Rekall Forensics Documentation

Release 1.7.2

The Rekall Team

Jan 29, 2018

Contents:

CHAPTER 1

EFilter - A query language for Rekall.

The Rekall framework is plugin based. This is what makes it so extensible. Developers can add many different plugins to implement different analysis techniques and produce different data.

Historically, plugins had no restriction over the type of output they produced. While some plugins put thought into producing structured output, others produced output which was only usable by humans, since it was largely unstructured. As the needs for automation increased, it soon became obvious that plugin output needs to be machine parseable in some way.

For example, consider the humble *pslist (APIPslist)* plugin - a simple plugin which just displays the list of running processes in tabular form. Initially this plugin produced a number of columns such as process name, pid etc. Some users required the binary path, and that was added. Then some users requires restricing the listed processed by various means, such as a list of pids, process name regular expression, start time etc.

Then some users wanted to combine the output from several plugins in some way. For example, show all the vad regions from a the "chrome" process.

It soon became obvious that we could not just keep adding more and more flags to each plugin to control the way the plugin worked. The same kind of filtering was repeating in many plugins (e.g. filter by process names) and it was difficult to anticipate how users would like to combine plugins in the future.

We wanted to create a mechanism that gave users control over which results they wanted to see, how to filter the output and how to combine the output from several plugins together.

The idea of building a framework to facilitate arbitrary queries was born. We chose to model the query language after SQL which is widely understood, and this is how EFilter was born.

1.1 What is EFilter?

EFilter is an SQL like query language for combining, filtering and customizing the output of Rekall plugins. Just like in SQL, EFilter queries are used to generate a customized output, however, unlike a database query, EFilter runs Rekall plugins to generate data dynamically, rather than look at stored data.

Lets look at a simple EFilter query:

```
select proc.name, pid from pslist() where pid > 4
```

This query contains three main parts:

1. The pslist() plugin will be executed and produce a set of rows. Each row contains several columns.
2. The filter condition follows the "where" operator and specifies a condition. EFilter will evaluate the condition on each row emitted from the plugin and only matching rows will be displayed.
3. The output is then produced in two columns which are derived from each emitted row.

1.2 Describing Plugins

In order for EFilter to work, each plugin must produce structured output in a specified format. We have seen before that plugins produce a sequence of rows, with each row having several columns. Each cell is a specific type of object.

Let us examine the *pslist()* plugin again. To get information about each plugin output we can use the *describe (Describe)* plugin:

```
[1] Live (API) 16:18:50> describe pslist, max_depth=1
Field                                                   Type
---------------------------------------------------     ----
proc                                                    LiveProcess
. as_dict                                               method
. cmdline                                               list
. connections                                           list
. cpu_affinity                                          list
. cpu_percent                                           float
. cpu_times                                             pcputimes
. create_time                                           float
. cwd                                                   str
. environ                                               dict
. exe                                                   str
. get_process_address_space                             method
. gids                                                  pgids
Name                                                    str
pid                                                     int
ppid                                                    int
Thds                                                    int
wow64                                                   bool
start                                                   UnixTimeStamp
```

In the above example, we see that the plugin generates a *Name* columen with a type of string, *pid* and *ppid* columns which are integers as well as a more complex type, such as a UnixTimeStamp.

We can also see the field *proc* which is of type *LiveProcess*. This more complex type is like a python dictionary itself, and contains multiple members.

Note: In Rekall each plugin is free to produce any output - the output types of each plugin are not defined in advance (since they might change depending on the profile, OS version etc). Therefore it is difficult to predict in advance what each column will contain.

The describe plugin therefore needs to actually run the plugin and it inspects the output of the first row produced. While this works most of the time, it is often not possible to get a sensible result without supplying proper arguments. For example, consider the *glob (IRGlob)* plugin. When run with no arguments it does not produce any results (since there is nothing to glob). Therefore *describe (Describe)* will produce incorrect results.

To solve this predicament it is possible to run the describe() plugin with the *args* parameter, which should be a python dict of parameters to be passed to the plugin. This way the plugin maybe run with reasonable parameters and produce reasonable results.

We can apply operators on the cells emitted by a specific plugin to generate the desired output. For example, suppose we wanted to show the command line for each running process. We can see the *proc* object contains a *cmdline* field, and so we can simply issue:

```
select proc.name, proc.cmdline from pslist()
```

Note that the cmdline is a list (it is the process's argv), and so Rekall will display it as such using the special annotation:

```
[1] Live (API) 16:32:48> select proc.name, proc.cmdline from pslist() where proc.name␣
↪=~ "rekall"
             cmdline                   name
----------------------------------- -------
- 0:                                rekall
  /home/mic/projects/Dev/bin/python3
- 1:
  /home/mic/projects/Dev/bin/rekall
- 2:
  -v
- 3:
  --live
- 4:
  API
```

1.3 Operator rules.

EFilter is type aware and will try to do the right thing with each type if it makes sense. When the user applies an operator on a type, the operator will attempt to do something sensible (or else it will just return None). The operator should never raise an error.

For example consider the =~ operator which means a regular expression match. When we apply this operator on a single string, we expect that it match that string:

```
select * from pslist() where proc.name =~ "rekall"
```

If however we applied this operator on a list, we expect the row to match if any of the list items matches:

```
select * from pslist() where proc.cmdline =~ "--live"
```

Note that it is not an error to try to apply a regular expression to a non-string - it simply will never match. Therefore the following query will always return the empty set, since an integer can never match a regular expression:

```
select * from pslist() where proc.pid =~ "foobar"
```

1.4 Plugin arguments.

In the queries above we just ran the pslist plugin with no arguments. Most Rekall plugins, however, take some form of arguments. We can see the arguments that a plugin takes by consulting the plugin documentation or by appending "?" to the name of the plugin:

```
[1] Live (API) 21:12:35> pslist?
file:             rekall-core/rekall/plugins/response/processes.py
Plugin:           APIPslist (pslist)
:                 This is a Typed Plugin.
Positional Args:   pids: One or more pids of processes to select. (type:␣
↪ArrayIntParser)
Keyword Args:
  profile:    Name of the profile to load. This is the filename of the profile found␣
↪in the profiles directory. Profiles are searched in the profile path order (If␣
↪specified we disable autodetection).
  proc_regex: A regex to select a process by name. (type: RegEx)
  verbosity:  An integer reflecting the amount of desired output: 0 = quiet, 10 =␣
↪noisy. (type: IntParser)
```

It is possible to feed the result of an efilter query into the parameters from another plugin. Here is a trivial example:

```
[1] Live (API) 21:19:53> select * from pslist(pids: (select pid from pslist() where␣
↪proc.name =~ "rekall"))
   proc        Name    pid  ppid  Thds Hnds        wow64                 start                ␣
↪          binary
   -------------- -------- ------ ------ ---- ---- ----------------- ---------------------
↪- -----------------------------------
   rekall (7826)  rekall  7826  7746  105        False              2018-01-27␣
↪05:12:20Z  /home/mic/projects/Dev/bin/python3
```

Note the following about the subselect syntax:

1. Argument names are provided to the plugin with the ":" operator. This assigns the output of the sub-select as a list into the parameter.
2. The subselect must yield a single column. If the subselect yields more than one column, it is not clear which column should be assigned to the plugin parameter and Rekall will issue an error:

   ```
   [1] Live (API) 21:19:43> select * from pslist(pids: (select * from pslist() where␣
   ↪proc.name =~ "rekall"))
   2018-01-26 21:19:43,526:CRITICAL:rekall.1:Invalid Args: pids invalid: Arg pids␣
   ↪must be a list of integers.
   ```

3. The arg assigment operator tries to convert the subselect column into the type required by the parameter. This means that if the parameter expects an integer then the subselect should yield something which should be convertible to an integer:

   ```
   [1] Live (API) 21:26:02> select * from pslist(pids: (select proc.name from␣
   ↪pslist() where proc.name =~ "rekall"))
   2018-01-26 21:26:02,643:CRITICAL:rekall.1:Invalid Args: pids invalid: invalid␣
   ↪literal for int() with base 10: 'rekall'.
   ```

1.5 EFilter functions.

We have seen that EFilter offers operators to work on columns. In this section we see some of the more common functions and operators the language provides.

1.5.1 timestamp

The timestamp function converts its argument into a timestamp object. This allows Rekall to operate on the timestamp in a timezone aware way, compare it to other times etc.

1.6 Examples

The following are example queries which demonstrate how some plugins may be stringed together to achieve powerful combinations.

1.6.1 Finding Processes launched by a certain user.

Rekall has the *tokens (GetSIDs)* plugin which displays all the authorization tokens possessed by each process. Rekall also automatically resolves the token's SID to a username.

```
[1] hank.aff4 22:54:29> tokens()
Process                                      Sid                        Comment
-------------------------------------- ---------------- ------------------------------
↪----
0xfa8000c9e040 System                  4 S-1-5-18         Local System
0xfa8000c9e040 System                  4 S-1-5-32-544     Administrators
0xfa8000c9e040 System                  4 S-1-1-0          Everyone
0xfa8000c9e040 System                  4 S-1-5-11         Authenticated Users
0xfa8000c9e040 System                  4 S-1-16-16384     System Mandatory Level
```

Lets see all the processes started by "jessie":

```
[1] hank.aff4 22:56:14> select * from tokens() where Comment =~ 'User: jessie'
Process                                      Sid                                       ␣
↪Comment
----------------------------------- ---------------------------------------------- ---
↪----------
0xfa8002418440 regsvr32.exe     884 S-1-5-21-4270721788-567995706-2532315982-1003 ␣
↪User: jessie
0xfa8001417720 explorer.exe    1512 S-1-5-21-4270721788-567995706-2532315982-1003 ␣
↪User: jessie
0xfa8000f95b30 VBoxTray.exe    1964 S-1-5-21-4270721788-567995706-2532315982-1003 ␣
↪User: jessie
0xfa8000fdc780 miranda64.exe   2208 S-1-5-21-4270721788-567995706-2532315982-1003 ␣
↪User: jessie
0xfa80022e2230 dwm.exe         2520 S-1-5-21-4270721788-567995706-2532315982-1003 ␣
↪User: jessie
0xfa8000f7d1b0 taskhost.exe    2596 S-1-5-21-4270721788-567995706-2532315982-1003 ␣
↪User: jessie
0xfa8002376060 taskhost.exe    2848 S-1-5-21-4270721788-567995706-2532315982-1003 ␣
↪User: jessie
```

Lets view each process creation time and its full command line. The Process column is not simply a string. It is a full blown Rekall object which represents the kernel's _EPROCESS struct. We therefore can dereference individual members of _EPROCESS and retrieve additional information.

```
[1] hank.aff4 22:59:13> select Process, Process.CreateTime, Comment, Process.Peb.
↪ProcessParameters.CommandLine from tokens() where Comment =~ 'User: jessie'
Process                                CreateTime          Comment                    ␣
↪    CommandLine
```

```
----------------------------------- --------------------- ------------- ---------------
↪-------------------------------------
0xfa8002418440 regsvr32.exe     884 2015-08-10 02:00:45Z  User: jessie
0xfa8001417720 explorer.exe    1512 2015-08-10 02:00:41Z  User: jessie ␣
↪C:\Windows\Explorer.EXE
0xfa8000f95b30 VBoxTray.exe    1964 2015-08-10 02:01:05Z  User: jessie
↪"C:\Windows\System32\VBoxTray.exe"
0xfa8000fdc780 miranda64.exe   2208 2015-08-10 02:01:37Z  User: jessie  "C:\Program␣
↪Files (x86)\Miranda IM\miranda64.exe"
0xfa80022e2230 dwm.exe         2520 2015-08-10 02:00:41Z  User: jessie
↪"C:\Windows\system32\Dwm.exe"
0xfa8000f7d1b0 taskhost.exe    2596 2015-08-10 02:13:51Z  User: jessie  "taskhost.exe"
0xfa8002376060 taskhost.exe    2848 2015-08-10 02:00:40Z  User: jessie  "taskhost.
↪77exe"
```

1.6.2 Find files modified in the last 2 days.

When Rekall is run in live mode, it can examine files on the local filesystem. This is useful for incident response situations. One of the more useful plugins available in live mode is the *glob (IRGlob)* plugin which enumerate files on the local filesystem based on one or more glob expressions (similar to the shell glob). According to the plugin documentation, we see that the plugin accepts a repeated parameter called "globs" for all the glob expressions. Let's see all the files in the /etc/ directory:

```
[1] Live (API) 23:49:05> select * from glob(globs: "/etc/*")
path
----------------------------
/etc/papersize
/etc/logrotate.d
/etc/mime.types
/etc/kbd
```

Although the output appears to only contain a single column ("path"), we can see that the path is actually an object which contains a lot of information about each file.

```
[1] Live (API) 00:16:03> describe glob, args=dict(globs=["/etc/*"])
Field                                                 Type
-------------------------------------------------- ----
path                                                FileInformation
. filename                                          FileSpec
.. filesystem                                       str
.. name                                             str
.. path_sep                                         str
. session                                           -
. st_atime                                          float
. st_ctime                                          float
. st_dev                                            int
. st_gid                                            Group
.. gid                                              int
.. group_name                                       str
.. session                                          NoneType
. st_ino                                            int
. st_mode                                           Permissions
. st_mtime                                          float
. st_nlink                                          int
. st_size                                           int
. st_uid                                            User
```

```
.. homedir                                             str
.. session                                             NoneType
.. shell                                               str
.. uid                                                 int
.. username                                            str
```

In particular we see that the *path.st_mtime* is a float describing the file's modification time:

```
[1] Live (API) 00:29:08> select path.st_mtime, path from glob(globs: "/etc/*")
st_mtime                    path
-------------------- ----------------------------
1516590897.1290069   /etc/papersize
1516687780.2982903   /etc/logrotate.d
1446219570.0         /etc/mime.types
```

Since the field is a float, Rekall does not understand that it is actually a timestamp, and therefore we can not do any time arithmetic on it. We therefore need to explitely convert the modification time to a timestamp using the *timestamp* function.

```
[1] Live (API) 00:31:50> select timestamp(path.st_mtime) as mtime, path from␣
↪glob(globs: "/etc/*") where mtime > "2 days ago"
mtime                 path
--------------------- -----------------
2018-01-29 06:11:15Z  /etc/resolv.conf
2018-01-29 06:11:15Z  /etc/timezone
```

1. Note the explicit conversion to a timestamp. This allows Rekall to apply time related operators on this column.
2. The column is aliased as "mtime", which appears as the title of the first column. More importantly, the alias can be used in further calculations (specifically inside the where clause).
3. Note the human readable time specification "2 days ago". Rekall supports such convenient expressions, as well as exactly formatted times.

CHAPTER 2

Plugin Reference

2.1 Memory

2.1.1 Windows

analyze_struct (AnalyzeStruct)

A plugin to analyze a memory location.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
offset	SymbolAddress	A virtual address to analyze.
search	IntParser	How far back to search for pool tag.
size	IntParser	How many elements to identify.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The Windows kernel allocates memory from “pool space”. To ease debugging memory leaks, the kernel uses a unique “Pool Tag” to tag many allocations. Each kernel subsystem or driver would use a specific tag to keep track of its allocation.

We can use this fact when we look at some undocumented, or unknown memory region. This is what the *analyze_struct* plugin does:

1. It first searched back from the address of interest to determine if this address is part of a pool allocation. The plugin will report the pool tag of this allocation as well as its size and starting offset.
2. For each slot in the struct, the plugin assumes it is a pointer to something, and checks if whatever it is pointing to is a pool allocation or a known address.

We can use this to get an idea of what exists at this memory location and its struct layout.

In the below example, we pick an _EPROCESS from the output of *pslist* and search for pointers to it somewhere in kernel memory (There are many pointers! We just picked one for this example.). We then use the *analyze_struct* plugin to discover that the pointer resides in an allocation with the pool tag ‘ObHd’. We can search the kernel disassembly to realize this is an Object Handle. Note how we use grep to search for the little endian representation of the _EPROCESS address.

```
[1] win7.elf 23:14:38> pslist
  _EPROCESS            Name           PID   PPID   Thds    Hnds    Sess  Wow64        ␣
↪  Start                    Exit
-------------- -------------------- ----- ------ ------ -------- ------ ------ -------
↪----------------- ------------------------
....
0xfa8002ad0190 cmd.exe              2644   2616      2       66      1 True   2012-
↪10-01 14:40:20Z     -

[1] win7.elf 23:14:55> grep keyword="\x90\x01\xad\x02\x80\xfa"
....
    Offset                                   Data                                  ␣
↪         Comment
-------------- ----------------------------------------------------------------- -----
↪-----------------------------------
0xf8a0013d8ad8 60 40 a9 02 80 fa ff ff 01 00 00 00 00 00 00 00  `@..............
0xf8a0013d8ae8 90 01 ad 02 80 fa ff ff 01 00 00 00 00 00 00 00  ................
0xf8a0013d8af8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
...
[1] win7.elf 23:17:20> analyze_struct 0xf8a0013d8ae8

0xf8a0013d8ae8 is inside pool allocation with tag 'ObHd' (0xf8a0013d8a30)  and size␣
↪0x100
    Offset     Content
-------------- -------
           0x0 Data:0xfa8002ad0190 Tag:Pro\xe3 @0xfa8002ad0190 (0x530)
           0x8 Data:0x1
          0x10 Data:0x0
          0x18 Data:0x0
          0x20 Data:0x0
          0x28 Data:0x0
          0x30 Data:0xfa80017f9060 Tag:Pro\xe3 @0xfa80017f9060 (0x530)
          0x38 Data:0x1
          0x40 Data:0x730061006c
          0x48 Data:0x744e034d0110
          0x50 Data:0x490053004c
          0x58 Data:0xa4801280702
          0x60 Data:0x981e
          0x68 Data:0x100000000
          0x70 Data:0x0
[1] win7.elf 23:22:25> hex(struct.unpack("<I", 'ObHd')[0])
               Out<24> '0x6448624f'
[1] win7.elf 23:22:33> dis "nt!ObpInsertHandleCount"
----------------------> dis("nt!ObpInsertHandleCount")
Address      Rel             Op Codes                    Instruction                 ␣
↪Comment
------- -------------- -------------------- ---------------------------------------- -
↪------
------ nt!ObpInsertHandleCount ------: 0xf80002976010
  0xf80002976010            0x0 48895c2408           mov qword ptr [rsp + 8], rbx
  0xf80002976015            0x5 48896c2410           mov qword ptr [rsp + 0x10], rbp
....

  0xf80002976089           0x79 41b84f624864         mov r8d, 0x6448624f
  0xf8000297608f           0x7f e83cd3e4ff           call 0xf800027c33d0              ␣
↪        nt!ExAllocatePoolWithTag
```

```
0xf80002976094          0x84 4885c0              test rax, rax
0xf80002976097          0x87 0f84dacd0400        je 0xf800029c2e77                ␣
↪       nt!ExpProfileCreate+0x9d57
0xf8000297609d          0x8d 458bc5              mov r8d, r13d
```

atomscan (AtomScan)

Pool scanner for _RTL_ATOM_TABLE

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayInt-Parser	Kernel addresses of eprocess structs.
limit	IntParser	The length of data to search in each selected region.
method	ChoiceArray	Method to list processes.
pids	ArrayInt-Parser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
scan_kernel	Boolean	Scan the entire kernel address space.
scan_kernel_code	Boolean	Scan the kernel image and loaded drivers.
scan_kernel_nonpaged_pool	Boolean	Scan the kernel non-paged pool.
scan_kernel_paged_pool	Boolean	Scan the kernel paged pool.
scan_kernel_session_pools	Boolean	Scan session pools for all processes.
scan_physical	Boolean	Scan the physical address space only.
scan_process_memory	Boolean	Scan all of process memory. Uses process selectors to narrow down selections.
sort_by	String	Sort by [offset | atom | refcount]
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

atoms (Atoms)

Print session and window station atom tables.

From: http://msdn.microsoft.com/en-us/library/windows/desktop/ms649053.aspx

An atom table is a system-defined table that stores strings and corresponding identifiers. An application places a string in an atom table and receives a 16-bit integer, called an atom, that can be used to access the string. A string that has been placed in an atom table is called an atom name.

The global atom table is available to all applications. When an application places a string in the global atom table, the system generates an atom that is unique throughout the system. Any application that has the atom can obtain the string it identifies by querying the global atom table.

(The global atom tables are only global within each session).

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

Using this plugin you can find registered window messages, rogue injected DLL paths, window class names, etc.

Sample output:

```
  Offset(P)     Session    WindowStation           Atom      RefCount   HIndex    ␣
↪Pinned     Name
-------------- ---------- ------------------ -------------- ---------- ---------- ----
↪------ ----
0xf8a002871020 0          WinSta0                    0xc001 1          1          ␣
↪True       StdExit
0xf8a002871020 0          WinSta0                    0xc002 1          2          ␣
↪True       StdNewDocument
0xf8a002871020 0          WinSta0                    0xc003 1          3          ␣
↪True       StdOpenDocument
0xf8a002871020 0          WinSta0                    0xc004 1          4          ␣
↪True       StdEditDocument
0xf8a002871020 0          WinSta0                    0xc005 1          5          ␣
↪True       StdNewfromTemplate
0xf8a002871020 0          WinSta0                    0xc006 1          6          ␣
↪True       StdCloseDocument
0xf8a002871020 0          WinSta0                    0xc007 1          7          ␣
↪True       StdShowItem
0xf8a002871020 0          WinSta0                    0xc008 1          8          ␣
↪True       StdDoVerbItem
0xf8a002871020 0          WinSta0                    0xc009 1          9          ␣
↪True       System
0xf8a002871020 0          WinSta0                    0xc00a 1          10         ␣
↪True       OLEsystem
0xf8a002871020 0          WinSta0                    0xc00b 1          11         ␣
↪True       StdDocumentName
0xf8a002871020 0          WinSta0                    0xc00c 1          12         ␣
↪True       Protocols
0xf8a002871020 0          WinSta0                    0xc00d 1          13         ␣
↪True       Topics
0xf8a002871020 0          WinSta0                    0xc00e 1          14         ␣
↪True       Formats
0xf8a002871020 0          WinSta0                    0xc00f 1          15         ␣
↪True       Status
0xf8a002871020 0          WinSta0                    0xc010 1          16         ␣
↪True       EditEnvItems
0xf8a002811020 0          ------------------         0xc045 2          69         ␣
↪False      MSUIM.Msg.LBUpdate
0xf8a002811020 0          ------------------         0xc046 2          70         ␣
↪False      MSUIM.Msg.MuiMgrDirtyUpdate
0xf8a002811020 0          ------------------         0xc047 1          71         ␣
↪False      C:\Windows\system32\wls0wndh.dll
0xf8a002811020 0          ------------------         0xc048 27         72         ␣
↪False      {FB8F0821-0164-101B-84ED-08002B2EC713}
0xf8a002811020 0          ------------------         0xc049 2          73         ␣
↪False      MMDEVAPI
```

callback_scan (CallbackScan)

Print system-wide notification routines by scanning for them.

Note this plugin is quite inefficient - consider using the callbacks plugin instead.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

callbacks (Callbacks)

Enumerate callback routines.

This plugin just enumerates installed callback routines from various sources. It does not scan for them.

This plugin is loosely based on the original Volatility plugin of the same name but much expanded using new information.

Reference: <http://www.codemachine.com/notes.html>

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The Windows kernel has a facility to register callbacks for certain events. This is often misused by malware in order to gain persistence. The *callbacks* plugin enumerates these callbacks.

Since Rekall has an address resolver, we can often say more about what exists at each of the callback locations. Normally Rekall only tracks the profile for certain binaries (such as the kernel).

In the below example the callbacks plugins resolves the address of kernel symbols precisely since it has the kernel profile loaded. Other symbols are give approximately as their distance from the module's export table.

Suppose we want to verify what is the callback in the "wdf01000" driver. We can instruct the address resolver to download the profile from the Microsoft symbol server. Once the profile is downloaded, Rekall can determine the exact function name registered (wdf01000!FxpBugCheckCallback).

```
[1] win7.elf 00:59:59> callbacks
---------------------> callbacks()
                Type                      Offset        Callback                                ␣
↪   Symbol                     Details
------------------------------------ -------------- -------------- -------------------
↪------------------------------- -------
nt!PspLoadImageNotifyRoutine         0xf8000283e4a0 0xf800029acb68 nt!
↪EtwpTraceLoadImage
nt!PspCreateProcessNotifyRoutine     0xf8000283e720 0xf8000265af28 nt!
↪ViCreateProcessCallback
nt!PspCreateProcessNotifyRoutine     0xf8000283e728 0xf88001211330 ksecdd!
↪AcceptSecurityContext+0x230
nt!PspCreateProcessNotifyRoutine     0xf8000283e730 0xf8800112b910 cng!
↪SystemPrng+0x6a0
nt!PspCreateProcessNotifyRoutine     0xf8000283e738 0xf8800164c390 tcpip!
↪CreateProcessNotifyRoutineEx
nt!PspCreateProcessNotifyRoutine     0xf8000283e740 0xf88000d01b94 ci!
↪CiFreePolicyInfo+0xce84
nt!KeBugCheckCallbackListHead        0xfa80019c3ea0 0xf880014548f0 ndis!
↪NdisGetSharedDataAlignment+0x10            Ndis min
nt!KeBugCheckCallbackListHead        0xfa80019a4ea0 0xf880014548f0 ndis!
↪NdisGetSharedDataAlignment+0x10            Ndis min
nt!KeBugCheckCallbackListHead        0xfa80019a1ea0 0xf880014548f0 ndis!
↪NdisGetSharedDataAlignment+0x10            Ndis min
nt!KeBugCheckCallbackListHead        0xf80002c25400 0xf80002c0eef4 hal!
↪HalQueryMaximumProcessorCount+0x54c            ACPI x64
nt!KeBugCheckReasonCallbackListHead  0xfa80026549f8 0xf88000efd054 wdf01000+0x7a054  ␣
↪                                  PEAUTH
nt!KeBugCheckReasonCallbackListHead  0xfa8000927f88 0xf88000efd054 wdf01000+0x7a054  ␣
↪                                  monitor
```

```
[1] win7.elf 02:04:35> address_resolver "wdf01000"
---------------------> address_resolver("wdf01000") |
 Trying to fetch http://msdl.microsoft.com/download/symbols/wdf01000.pdb/
↪99521C1B360441A9A1EAECC9E5087A251/wdf01000.pd_
 Trying to fetch http://msdl.microsoft.com/download/symbols/wdf01000.pdb/
↪99521C1B360441A9A1EAECC9E5087A251/wdf01000.pd_
Extracting cabinet: /tmp/tmpnOmJvR/wdf01000.pd_
  extracting Wdf01000.pdb

All done, no errors.
                Out<1> Plugin: address_resolver

1] win7.elf 02:05:08> callbacks
----------------------> callbacks()
                Type                     Offset        Callback                   ␣
↪    Symbol                      Details
------------------------------------ -------------- -------------- -------------------
↪------------------------------ -------
nt!PspLoadImageNotifyRoutine         0xf8000283e4a0 0xf800029acb68 nt!
↪EtwpTraceLoadImage
nt!PspCreateProcessNotifyRoutine     0xf8000283e720 0xf8000265af28 nt!
↪ViCreateProcessCallback
nt!PspCreateProcessNotifyRoutine     0xf8000283e728 0xf88001211330 ksecdd!
↪AcceptSecurityContext+0x230
nt!PspCreateProcessNotifyRoutine     0xf8000283e730 0xf8800112b910 cng!
↪SystemPrng+0x6a0
nt!PspCreateProcessNotifyRoutine     0xf8000283e738 0xf8800164c390 tcpip!
↪CreateProcessNotifyRoutineEx
nt!PspCreateProcessNotifyRoutine     0xf8000283e740 0xf88000d01b94 ci!
↪CiFreePolicyInfo+0xce84
nt!KeBugCheckCallbackListHead        0xfa80019c3ea0 0xf880014548f0 ndis!
↪NdisGetSharedDataAlignment+0x10            Ndis min
nt!KeBugCheckCallbackListHead        0xfa80019a4ea0 0xf880014548f0 ndis!
↪NdisGetSharedDataAlignment+0x10            Ndis min
nt!KeBugCheckCallbackListHead        0xfa80019a1ea0 0xf880014548f0 ndis!
↪NdisGetSharedDataAlignment+0x10            Ndis min
nt!KeBugCheckCallbackListHead        0xf80002c25400 0xf80002c0eef4 hal!
↪HalQueryMaximumProcessorCount+0x54c         ACPI x64
nt!KeBugCheckReasonCallbackListHead  0xfa80026549f8 0xf88000efd054 wdf01000!
↪FxpBugCheckCallback                    PEAUTH
nt!KeBugCheckReasonCallbackListHead  0xfa8000927f88 0xf88000efd054 wdf01000!
↪FxpBugCheckCallback                    monitor
nt!KeBugCheckReasonCallbackListHead  0xfa80021f54b0 0xf88003edaf40 mouhid+0x3f40    ␣
↪                                mouhid
```

certscan (CertYaraScan)

Scan certificates in windows memory regions.

Plugin	Type	Description
binary_string	String	A binary string (encoded as hex) to search for. e.g. 000102[1-200]0506
context	IntParser	Context to print after the hit.
dtb	IntParser	The DTB physical address.
eprocess	ArrayInt-Parser	Kernel addresses of eprocess structs.
hits	IntParser	Total number of hits to report.
limit	IntParser	The length of data to search in each selected region.
method	ChoiceArray	Method to list processes.
pids	ArrayInt-Parser	One or more pids of processes to select.
pre_context	IntParser	Context to print before the hit.
proc_regex	RegEx	A regex to select a process by name.
scan_kernel	Boolean	Scan the entire kernel address space.
scan_kernel_code	Boolean	Scan the kernel image and loaded drivers.
scan_kernel_nonpaged_pool	Boolean	Scan the kernel non-paged pool.
scan_kernel_paged_pool	Boolean	Scan the kernel paged pool.
scan_kernel_session_pools	Boolean	Scan session pools for all processes.
scan_physical	Boolean	Scan the physical address space only.
scan_process_memory	Boolean	Scan all of process memory. Uses process selectors to narrow down selections.
string	String	A verbatim string to search for.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.
yara_expression	String	
yara_file	String	

check_pehooks (CheckPEHooks)

Checks a pe file mapped into memory for hooks.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
im-age_base	SymbolAd-dress	The base address of the pe image in memory.
thorough	Boolean	By default we take some optimization. This flags forces thorough but slower checks.
type	Choice	Type of hook to display.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

clipboard (Clipboard)

Extract the contents of the windows clipboard

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

cmdscan (CmdScan)

Extract command history by scanning for _COMMAND_HISTORY

Plugin	Type	Description
dtb	Int-Parser	The DTB physical address.
max_history	Int-Parser	Value of history buffer size. See HKEY_C URRENT_USERConsoleHistoryBufferSize for default.
verbosity	Int-Parser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The cmdscan plugin searches the memory of csrss.exe on XP/2003/Vista/2008 and conhost.exe on Windows 7 for commands that attackers entered through a console shell (cmd.exe). This is one of the most powerful commands you can use to gain visibility into an attackers actions on a victim system, whether they opened cmd.exe through an RDP session or proxied input/output to a command shell from a networked backdoor.

This plugin finds structures known as **COMMAND_HISTORY** by looking for a known constant value (**MaxHistory**) and then applying sanity checks. It is important to note that the **MaxHistory** value can be changed by right clicking in the top left of a cmd.exe window and going to Properties. The value can also be changed for all consoles opened by a given user by modifying the registry key HKCUConsoleHistoryBufferSize. The default is 50 on Windows systems, meaning the most recent 50 commands are saved. You can tweak it if needed by using the –max_history=NUMBER parameter.

The structures used by this plugin are not public (i.e. Microsoft does not produce PDBs for them), thus they're not available in WinDBG or any other forensic framework. They were reverse engineered by Michael Ligh from the conhost.exe and winsrv.dll binaries.

In addition to the commands entered into a shell, this plugin shows:

- The name of the console host process (csrss.exe or conhost.exe)
- The name of the application using the console (whatever process is using cmd.exe)
- The location of the command history buffers, including the current buffer count, last added command, and last displayed command
- The application process handle

Due to the scanning technique this plugin uses, it has the capability to find commands from both active and closed consoles.

### Notes

This plugin is pretty fragile since it relies on reversed structures in undocumented code. We are working on improving the situation here but there is a moderate chance that it will produce no results or garbage results.

### Sample Output

The following showing an operator using the winpmem acquisition tool to analyse the live memory of a Windows 7 machine.

```
win7.elf 22:15:39> cmdscan
-----------------> cmdscan()
**************************************************
CommandProcess: conhost.exe Pid: 2652
CommandHistory: 0x7ea40 Application: cmd.exe Flags: Allocated, Reset
CommandCount: 3 LastAdded: 2 LastDisplayed: 2
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x5c
Cmd    Address     Text
--- -------------- --------------------------------------------------
  0 0x00000005ea70 cd \Users\a\Desktop
  1 0x00000005b920 winpmem_1.1-write.exe -w -l
  2 0x0000000b3e70 vol.exe --profile Win7SP1x64 --file \\.\pmem
 15 0x000000040158
 16 0x00000007d3b0

**************************************************
CommandProcess: conhost.exe Pid: 2652
CommandHistory: 0xb40c0 Application: vol.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0xd4
Cmd    Address     Text
--- -------------- --------------------------------------------------
  0 0x0000001f77e0
  3 0x000000060ef0
  5 0x0000001f77e0
  8 0x000000060ef0
 10 0x0000001f77e0
 13 0x0000ffd96238
 14 0x00000007ec20
 15 0x0000001f7720
 23 0x0000000610a0
 24 0x0000000974e0
**************************************************
CommandProcess: conhost.exe Pid: 2652
CommandHistory: 0xb4410 Application: vol.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0xd8
Cmd    Address     Text
--- -------------- --------------------------------------------------
```

connscan (ConnScan)

Scan Physical memory for _TCPT_OBJECT objects (tcp connections)

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayInt-Parser	Kernel addresses of eprocess structs.
limit	IntParser	The length of data to search in each selected region.
method	ChoiceArray	Method to list processes.
pids	ArrayInt-Parser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
scan_kernel	Boolean	Scan the entire kernel address space.
scan_kernel_code	Boolean	Scan the kernel image and loaded drivers.
scan_kernel_nonpaged_pool	Boolean	Scan the kernel non-paged pool.
scan_kernel_paged_pool	Boolean	Scan the kernel paged pool.
scan_kernel_session_pools	Boolean	Scan session pools for all processes.
scan_physical	Boolean	Scan the physical address space only.
scan_process_memory	Boolean	Scan all of process memory. Uses process selectors to narrow down selections.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

Similar to the [connections](Connections.html) plugin, this plugin searches from _TCP_OBJECT structs. However, it employs pool scanning techniques.

### Notes

1. This plugin only works on versions of winsows prior to Win7.
2. Since the plugin may recover freed pool memory, the data may have been overwritten. This might produce garbage results for terminated connections.

### Sample output.

Note the nonsensical connection for local address *3.0.48.2* and the incorrect pid number below.

```
xp-laptop-2005-06-25.img 23:00:29> connscan
-------------------------------> connscan()
Offset(P)  Local Address             Remote Address                   Pid
---------- ------------------------- ------------------------- ----------
0x01370e70 192.168.2.7:1115          207.126.123.29:80               1916
0x01ed1a50 3.0.48.2:17985            66.179.81.245:20084       4287933200
0x01f0e358 192.168.2.7:1164          66.179.81.247:80                 944
0x01f11e70 192.168.2.7:1082          205.161.7.134:80                2392
0x01f35cd0 192.168.2.7:1086          199.239.137.200:80              1916
0x01f88e70 192.168.2.7:1162          170.224.8.51:80                 1916
0x020869b0 127.0.0.1:1055            127.0.0.1:1056                  2160
```

connections (Connections)

Print list of open connections [Windows XP Only]

This module enumerates the active connections from tcpip.sys.

Note that if you are using a hibernated image this might not work because Windows closes all sockets before hibernating. You might find it more effective to do conscan instead.

Active TCP connections are found in a hash table. The Hash table is given by the _TCBTable symbol. The size of the hash table is found in the _MaxHashTableSize variable.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

Prior to Windows 7, the windows TCP/IP stack uses objects of type _TCP_OBJECT to track TCP endpoints. These are the objects parsed by this module, hence this module will only be available on images from windows XP.

This module walks the _TCP_OBJECT hash tables and displays information related to the TCP endpoints.

### Notes

1. This plugin depends on exported debugging symbols, and therefore requires the correct tcpip profile to be loaded from the profile repository. See the [FAQ](/faq.html#profile) if you need to generate a profile.
2. For later versions of windows use the [netscan](Netscan.html) or the [netstat](Netstat.html) modules.

### Sample output

```
xp-laptop-2005-06-25.img 23:00:24> connections
---------------------------------> connections()
Offset (V) Local Address             Remote Address                Pid
---------- ------------------------- ------------------------- ------
0x820869b0 127.0.0.1:1055            127.0.0.1:1056              2160
0xffa2baf0 127.0.0.1:1056            127.0.0.1:1055              2160
0x8220c008 192.168.2.7:1077          64.62.243.144:80            2392
0x81f11e70 192.168.2.7:1082          205.161.7.134:80            2392
0x8220d6b8 192.168.2.7:1066          199.239.137.200:80          2392
```

consolescan (ConsoleScan)

Extract command history by scanning for _CONSOLE_INFORMATION

Plugin	Type	Description
dtb	Int-Parser	The DTB physical address.
his-tory_buffers	Int-Parser	Value of history buffer size. See HKEY_C URRENT_USERConsoleHistoryBufferSize for default.
max_history	Int-Parser	Value of history buffer size. See HKEY_C URRENT_USERConsoleHistoryBufferSize for default.
verbosity	Int-Parser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

consoles (Consoles)

Enumerate command consoles.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

Similar to [cmdscan](CmdScan.html) the consoles plugin finds commands that attackers typed into cmd.exe or executed via backdoors. However, instead of scanning for **COMMAND_HISTORY**, this plugin scans for **CONSOLE_INFORMATION**. The major advantage to this plugin is it not only prints the commands attackers typed, but it collects the entire screen buffer (input and output). For instance, instead of just seeing “dir”, you’ll see exactly what the attacker saw, including all files and directories listed by the “dir” command.

Additionally, this plugin prints the following:

- The original console window title and current console window title
- The name and pid of attached processes (walks a **LIST_ENTRY** to enumerate all of them if more than one)
- Any aliases associated with the commands executed. For example, attackers can register an alias such that typing “hello” actually executes “cd system”
- The screen coordinates of the cmd.exe console.

### Notes

This plugin is pretty fragile since it relies on reversed structures in undocumented code. We are working on improving the situation here but there is a moderate chance that it will produce no results or garbage results.

### Sample Output

```
win7.elf 22:23:10> consoles
**************************************************
ConsoleProcess: conhost.exe Pid: 2652
Console: 0xffd96200 CommandHistorySize: 50
HistoryBufferCount: 4 HistoryBufferMax: 4
OriginalTitle: Console2 command window
Title: Administrator: Console2 command window - vol.exe  --profile Win7SP1x64 --file␣
↪\\.\pmem
AttachedProcess: vol.exe Pid: 2920 Handle: 0xd8
AttachedProcess: vol.exe Pid: 2912 Handle: 0xd4
AttachedProcess: cmd.exe Pid: 2644 Handle: 0x5c
----
CommandHistory: 0xb4410 Application: vol.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0xd8
----
CommandHistory: 0xb40c0 Application: vol.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0xd4
----
CommandHistory: 0xb3ee0 Application: winpmem_1.1-write.exe Flags:
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x0
----
CommandHistory: 0x7ea40 Application: cmd.exe Flags: Allocated, Reset
CommandCount: 3 LastAdded: 2 LastDisplayed: 2
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x5c
Cmd #0 at 0x5ea70: cd \Users\a\Desktop
Cmd #1 at 0x5b920: winpmem_1.1-write.exe -w -l
Cmd #2 at 0xb3e70: vol.exe --profile Win7SP1x64 --file \\.\pmem
----
Screen 0x60ef0 X:117 Y:500
Dump:
```

```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>cd \Users\a\Desktop

C:\Users\a\Desktop>winpmem_1.1-write.exe -w -l
Will enable write mode
Loaded Driver.

C:\Users\a\Desktop>vol.exe --profile Win7SP1x64 --file \\.\pmem
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit (Intel)]
Type "copyright", "credits" or "license" for more information.

IPython 0.12.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.


The Volatility Memory Forensic Framework technology preview (3.0_tp2).

NOTE: This is pre-release software and is provided for evauation only. Please
check at http://volatility.googlecode.com/ for officially supported versions.

This program is free software; you can redistribute it and/or modify it under
the terms of the GNU General Public License.
Win7SP1x64:pmem 07:41:08> pslist
------------------------> pslist()
  Offset (V)   Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start␣
↪                Exit

-------------- -------------------- ------ ------ ------ -------- ------ ------ ------
↪-------------- ----------------
----
0xfa80008959e0 System                    4      0     85      502 ------  False 2012-
↪10-01 21:39:51   -

0xfa8001994310 smss.exe                272      4      2       29 ------  False 2012-
↪10-01 21:39:51   -
```

dlldump (DLLDump)

Dump DLLs from a process address space

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
dump_dir	String	Path suitable for dumping files.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
out_fd	String	A file like object to write the output.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
regex	RegEx	A Regular expression for selecting the dlls to dump.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

To extract a DLL from a process's memory space and dump it to disk for analysis, use the dlldump command. All the usual process selectors are supported. Additionally a regular expression can be specified for the DLL name to dump.

### Note

1. In order to dump any PE file from memory we need the PE header to be memory resident. Often this is not the case, and the header is flushed out of virtual memory. In this case it is still possible to dump parts of the PE image using the [vaddump](VADDump.html) plugin.
2. When dumping any binary from memory, it is not usually a perfect binary (i.e. you can not just run it). This is because the Import Address Table (IAT) reflects the patched version in memory and some pages may be missing. The resultant binary is probably only useful to analyses using a tool like IDA pro.

### Sample output

```
win8.1.raw 14:51:37> dlldump proc_regex="winpmem", dump_dir="/tmp/"
-------------------> dlldump(proc_regex="winpmem", dump_dir="/tmp/")
  _EPROCESS     Name                   Base        Module                Dump File
-------------- ----------------- -------------- --------------------- ---------
0xe0000204a900 winpmem_1.5.2.   0x000000020000 winpmem_1.5.2.exe     module.2628.
↪3d04a900.20000.winpmem_1.5.2.exe
0xe0000204a900 winpmem_1.5.2.   0x7ff87f320000 ntdll.dll             module.2628.
↪3d04a900.7ff87f320000.ntdll.dll
0xe0000204a900 winpmem_1.5.2.   0x000076f50000 wow64.dll             module.2628.
↪3d04a900.76f50000.wow64.dll
0xe0000204a900 winpmem_1.5.2.   0x000076fa0000 wow64win.dll          module.2628.
↪3d04a900.76fa0000.wow64win.dll
0xe0000204a900 winpmem_1.5.2.   0x000077010000 wow64cpu.dll          module.2628.
↪3d04a900.77010000.wow64cpu.dll
```

dtbscan (DTBScan)

Scans the physical memory for DTB values.

This plugin can compare the DTBs found against the list of known processes to find hidden processes.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
limit	IntParser	Stop scanning after this many mb.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The PFN database can be used to resolve a physical address to its virtual address in the process address space. Since processes must have unique page tables, and therefore a unique DTB, we can enumerate all unique page tables on the system.

Using this technique allows us to locate hidden processes. We simply check each physical page and locate its DTB (or page table directory base) offset. We then match the DTB to a known process DTB. If the DTB is not known this is a strong indication that the process is hidden.

### Sample output

```
win8.1.raw 16:23:50> dtbscan
-------------------> dtbscan()
```

```
     DTB           VAddr        _EPROCESS     Image Name           Known
-------------- -------------- -------------- -------------------- -----
0x0000001a7000 0xf6fb7dbed000 0xe00000074580 System               True
0x0000118a3000 0xf6fb7dbed000 0xe00002073900 explorer.exe         True
0x00000923e000 0xf6fb7dbed000 0xe000020ea900 svchost.exe          True
0x000036ea3000 0xf6fb7dbed000 0xe000006208c0 taskhost.exe         True
0x000004c01000 0xf6fb7dbed000 0xe000000ce080 wininit.exe          True
0x00000d0a4000 0xf6fb7dbed000 0xe000022c6900 MsMpEng.exe          True
0x0000093c4000 0xf6fb7dbed000 0xe000020df080 svchost.exe          True
0x0000348c6000 0xf6fb7dbed000 0xe00001e2f700 dwm.exe              True
0x000011504000 0xf6fb7dbed000 0xe000007a3080 svchost.exe          True
0x000007c94000 0xf6fb7dbed000 0xe00001f22080 cmd.exe              True
0x00002fe03000 0xf6fb7dbed000 0xe00002043900 conhost.exe          True
0x00002f8ce000 0xf6fb7dbed000 0xe00001299900 SearchIndexer.       True
0x0000207b9000 0xf6fb7dbed000 0xe00002645080 VBoxTray.exe         True
```

devicetree (DeviceTree)

Show device tree.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayInt-Parser	Kernel addresses of eprocess structs.
limit	IntParser	The length of data to search in each selected region.
method	ChoiceArray	Method to list processes.
pids	ArrayInt-Parser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
scan_kernel	Boolean	Scan the entire kernel address space.
scan_kernel_code	Boolean	Scan the kernel image and loaded drivers.
scan_kernel_nonpaged_pool	Boolean	Scan the kernel non-paged pool.
scan_kernel_paged_pool	Boolean	Scan the kernel paged pool.
scan_kernel_session_pools	Boolean	Scan session pools for all processes.
scan_physical	Boolean	Scan the physical address space only.
scan_process_memory	Boolean	Scan all of process memory. Uses process selectors to narrow down selections.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

Windows uses a layered driver architecture, or driver chain so that multiple drivers can inspect or respond to an IRP. Rootkits often insert drivers (or devices) into this chain for filtering purposes (to hide files, hide network connections, steal keystrokes or mouse movements). The devicetree plugin shows the relationship of a driver object to its devices (by walking _DRIVER_OBJECT.DeviceObject.NextDevice) and any attached devices (_DRIVER_OBJECT.DeviceObject.AttachedDevice).

### Notes

In the current implementation this plugin uses scanning methods to locate the driver and device objects. This is an inefficient method which is also susceptible to false positives and active subversion. We are working on converting this plugin to use the [object_tree](ObjectTree.html) plugin to directly parse kernel driver structures.

### Sample output

```
[snip]
DRV 0x2bb31060 \Driver\winpmem
---| DEV 0xfa80019ba060 pmem FILE_DEVICE_UNKNOWN
DRV 0x2bb36600 \Driver\TermDD
---| DEV 0xfa80019ff040 - FILE_DEVICE_8042_PORT
------| ATT 0xfa80019ff980 - - \Driver\mouclass FILE_DEVICE_MOUSE
---| DEV 0xfa80019e2040 - FILE_DEVICE_8042_PORT
------| ATT 0xfa80019e2960 - - \Driver\kbdclass FILE_DEVICE_KEYBOARD
[snip]
```

In the above we can see that the winpmem driver has a device called "pmem". We also can see the mouse and keyboard drivers attached to the terminal services driver.

driverirp (DriverIrp)

Driver IRP hook detection

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayInt-Parser	Kernel addresses of eprocess structs.
limit	IntParser	The length of data to search in each selected region.
method	ChoiceArray	Method to list processes.
pids	ArrayInt-Parser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
regex	RegEx	Analyze drivers matching REGEX
scan_kernel	Boolean	Scan the entire kernel address space.
scan_kernel_code	Boolean	Scan the kernel image and loaded drivers.
scan_kernel_nonpaged_pool	Boolean	Scan the kernel non-paged pool.
scan_kernel_paged_pool	Boolean	Scan the kernel paged pool.
scan_kernel_session_pools	Boolean	Scan session pools for all processes.
scan_physical	Boolean	Scan the physical address space only.
scan_process_memory	Boolean	Scan all of process memory. Uses process selectors to narrow down selections.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

Windows drivers export a table of functions called the IRP **MajorFunction** table. In that table, the driver installs function handlers to handle verious types of requests from userspace. A common way to hook a legitimate driver is to replace these function pointers with a malicious function.

Many drivers forward their IRP functions to other drivers for legitimate purposes, so detecting hooked IRP functions based on containing modules is not a good method. Instead, we print everything and let you be the judge. The command also checks for Inline hooks of IRP functions and optionally prints a disassembly of the instructions at the IRP address (pass –verbosity to enable this).

This command outputs information for all drivers, unless you specify a regular expression filter.

### Notes

In the current implementation this plugin uses scanning methods to locate the driver and device objects. This is an inefficient method which is also susceptible to false positives and active subversion. We are working on converting this plugin to use the [object_tree](ObjectTree.html) plugin to directly parse kernel driver structures.

### Sample output

In the below we see that the pmem driver handles the **IRP_MJ_CREATE**, **IRP_MJ_CLOSE**, **IRP_MJ_READ** and **IRP_MJ_DEVICE_CONTROL** IRP types.

```
win8.1.raw 16:15:36> driverirp regex="pmem"
-------------------> driverirp(regex="pmem")
**************************************************
DriverName: pmem
DriverStart: 0xf800025ca000
DriverSize: 0x10000
DriverStartIo: 0x0
   - Func Name                            Func Addr    Module
---- ------------------------------------ -------------- ------
   0 IRP_MJ_CREATE                        0xf800025cb210 \??
↪\C:\Users\test\AppData\Local\Temp\pmeA86F.tmp
   1 IRP_MJ_CREATE_NAMED_PIPE             0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
   2 IRP_MJ_CLOSE                         0xf800025cb270 \??
↪\C:\Users\test\AppData\Local\Temp\pmeA86F.tmp
   3 IRP_MJ_READ                          0xf800025cbfa0 \??
↪\C:\Users\test\AppData\Local\Temp\pmeA86F.tmp
   4 IRP_MJ_WRITE                         0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
   5 IRP_MJ_QUERY_INFORMATION             0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
   6 IRP_MJ_SET_INFORMATION               0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
   7 IRP_MJ_QUERY_EA                      0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
   8 IRP_MJ_SET_EA                        0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
   9 IRP_MJ_FLUSH_BUFFERS                 0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
  10 IRP_MJ_QUERY_VOLUME_INFORMATION      0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
  11 IRP_MJ_SET_VOLUME_INFORMATION        0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
  12 IRP_MJ_DIRECTORY_CONTROL             0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
  13 IRP_MJ_FILE_SYSTEM_CONTROL           0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
  14 IRP_MJ_DEVICE_CONTROL                0xf800025cb300 \??
↪\C:\Users\test\AppData\Local\Temp\pmeA86F.tmp
  15 IRP_MJ_INTERNAL_DEVICE_CONTROL       0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
  16 IRP_MJ_SHUTDOWN                      0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
  17 IRP_MJ_LOCK_CONTROL                  0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
  18 IRP_MJ_CLEANUP                       0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
  19 IRP_MJ_CREATE_MAILSLOT               0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
  20 IRP_MJ_QUERY_SECURITY                0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
  21 IRP_MJ_SET_SECURITY                  0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
  22 IRP_MJ_POWER                         0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
```

```
 23 IRP_MJ_SYSTEM_CONTROL              0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
 24 IRP_MJ_DEVICE_CHANGE               0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
 25 IRP_MJ_QUERY_QUOTA                 0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
 26 IRP_MJ_SET_QUOTA                   0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
 27 IRP_MJ_PNP                         0xf802d31131b8␣
↪\SystemRoot\system32\ntoskrnl.exe
```

driverscan (DriverScan)

Scan for driver objects _DRIVER_OBJECT

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayInt-Parser	Kernel addresses of eprocess structs.
limit	IntParser	The length of data to search in each selected region.
method	ChoiceArray	Method to list processes.
pids	ArrayInt-Parser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
scan_kernel	Boolean	Scan the entire kernel address space.
scan_kernel_code	Boolean	Scan the kernel image and loaded drivers.
scan_kernel_nonpaged_pool	Boolean	Scan the kernel non-paged pool.
scan_kernel_paged_pool	Boolean	Scan the kernel paged pool.
scan_kernel_session_pools	Boolean	Scan session pools for all processes.
scan_physical	Boolean	Scan the physical address space only.
scan_process_memory	Boolean	Scan all of process memory. Uses process selectors to narrow down selections.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

To find **_DRIVER_OBJECT**s in physical memory using pool tag scanning, use this plugin. This is another way to locate kernel modules, although not all kernel modules have an associated **_DRIVER_OBJECT**.

The usual way for malware to enter Ring 0 is via loading a kernel driver of some sort. A malicious kernel driver is a strong indication that malware is running in Ring 0.

### Notes

1. Like other pool scanning plugins, this plugin may produce false positives since it essentially carves **_DRIVER_OBJECT** structures out of memory. On the other hand, this plugin may reveal drivers which have been unloaded.

### Sample output

```
win8.1.raw 16:17:29> driverscan
-------------------> driverscan()
    Offset(P)     #Ptr #Hnd     Start            Size       Service Key          Name    ␣
↪     Driver Name
- -------------- ---- ---- -------------- -------------- -------------------- --------
↪---- -----------
```

```
...
  0x00003e569c60     3     0 0xf80000b14000         0x10000 pcw                        pcw     ␣
↪     \Driver\pcw
  0x00003e569e60     3     0 0xf80000aeb000         0x29000 VBoxGuest                ␣
↪VBoxGuest     \Driver\VBoxGuest
  0x00003e59e590    17     0 0xf80000c26000        0x118000 NDIS                       NDIS    ␣
↪     \Driver\NDIS
  0x00003e5a1060     8     0 0xf80000ec5000        0x27f000 Tcpip                      Tcpip   ␣
↪     \Driver\Tcpip
  0x00003eb8d870     3     0 0xf800025ca000         0x10000 pmem                       pmem    ␣
↪     \Driver\pmem
  0x00003f066e60     3     0 0xf80001c69000          0xe000 monitor                    monitor␣
↪     \Driver\monitor
....
```

dumpfiles (DumpFiles)

Dump files from memory.

The interface is loosely based on the Volatility plugin of the same name, although the implementation is quite different.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
dump_dir	String	Path suitable for dumping files.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
file_objects	ArrayIntParser	Kernel addresses of _FILE_OBJECT structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

hooks_eat (EATHooks)

Detect EAT hooks in process and kernel memory

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

vacbs (EnumerateVacbs)

Enumerate all blocks cached in the cache manager.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

evtlogs (EvtLogs)

Extract Windows Event Logs (XP/2003 only)

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The evtlogs command extracts and parses binary event logs from memory. Binary event logs are found on Windows XP and 2003 machines, therefore this plugin only works on these architectures. These files are extracted from VAD of the services.exe process, parsed and shown as output.

### Notes

1. This plugin will only work on Windows XP/2003. Modern windows systems use evtx event log format. We are still working on supporting these logs.

### Sample output

```
xp-laptop-2005-06-25.img 16:43:19> evtlogs
----------------------------------> evtlogs()
TimeWritten Filename Computer Sid Source Event Id Event Type Message
----------- --------- --------- --- ------ --------- ---------- --------
2004-05-05 19:36:55+0000 SecEvent.Evt MOIT-A-PHXMOD2 S-1-5-18 Security 612 Success '-
↪';'+';'+';'+';'+';'+';'-';'-';'-';'-';'+';'+';'+';'+';'+';'+';'+';'+';'MOIT-A-
↪PHXMOD2$';'BALTIMORE';'(0x0,0x3E7)'
2004-05-05 19:36:56+0000 SecEvent.Evt MOIT-A-PHXMOD2 S-1-5-18 Security 618 Success
↪'MOIT-A-PHXMOD2$';'BALTIMORE';'(0x0,0x3E7)';'PolEfDat: <binary data> (none);  '
2004-05-05 19:37:03+0000 SecEvent.Evt MOIT-A-PHXMOD2 S-1-5-18 Security 537 Failure
↪'AJ.Morning';'BALTIMORE';'11';'User32  ';'Negotiate';'MOIT-A-PHXMOD2';'0xC000005E';
↪'0x0'
2004-05-05 19:37:03+0000 SecEvent.Evt MOIT-A-PHXMOD2 S-1-5-21-487349131-2095749132-
↪2248483902-19753 Security 528 Success 'AJ.Morning';'BALTIMORE';'(0x0,0x113AD)';'2';
↪'User32  ';'Negotiate';'MOIT-A-PHXMOD2';'{5c92d34f-85d3-2f5d-d036-759d7c97bfd7}'
2004-05-05 19:37:32+0000 SecEvent.Evt MOIT-A-PHXMOD2 S-1-5-19 Security 528 Success
↪'LOCAL SERVICE';'NT AUTHORITY';'(0x0,0x3E5)';'5';'Advapi  ';'Negotiate';'';'
↪{00000000-0000-0000-0000-000000000000}'
2004-05-05 19:37:33+0000 SecEvent.Evt MOIT-A-PHXMOD2 S-1-5-21-487349131-2095749132-
↪2248483902-19753 Security 596 Failure '619be804-cde6-484f-aff4-2a5e588d6eef';'';'';
↪'0x57'
```

filescan (FileScan)

Scan Physical memory for _FILE_OBJECT pool allocations

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
limit	IntParser	The length of data to search in each selected region.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
scan_kernel	Boolean	Scan the entire kernel address space.
scan_kernel_code	Boolean	Scan the kernel image and loaded drivers.
scan_kernel_nonpaged_pool	Boolean	Scan the kernel non-paged pool.
scan_kernel_paged_pool	Boolean	Scan the kernel paged pool.
scan_kernel_session_pools	Boolean	Scan session pools for all processes.
scan_physical	Boolean	Scan the physical address space only.
scan_process_memory	Boolean	Scan all of process memory. Uses process selectors to narrow down selections.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

To find FILE_OBJECTs in physical memory using pool tag scanning, use the filescan command. This will find open files even if a rootkit is hiding the files on disk and if the rootkit hooks some API functions to hide the open handles on a live system.

The plugin also resolves back the **_FILE_OBJECT** into the ownning process. This works only if the **_FILE_OBJECT** is actually in use (it does not work for closed files).

### Notes

1. Like other pool scanning plugins, this plugin may produce false positives since it essentially carves **_FILE_OBJECT** structures out of memory. On the other hand, this plugin may reveal files which have been closed or freed.
2. When inspecting the output, the **#Hnd** column indicates the number of handles to this **_FILE_OBJECT**. Objects in use will have a non zero value here and are likely to not be freed.
3. The plugin displays the physical address of the **_FILE_OBJECT** found. It may be possible to derive their virtual address using the [ptov](PtoV.html) plugin. Alternatively, specify the *scan_in_kernel* option, to ensure scanning occurs in the kernel address space.

### Sample output

```
win8.1.raw 16:55:44> filescan scan_in_kernel=True
-------------------> filescan(scan_in_kernel=True)
      Offset        #Ptr #Hnd Access      Owner       Owner Pid Owner Name        Name
- -------------- ------ ---- ------ -------------- --------- ---------------- ----
  0xe000000421e0      17    0 RW-rwd -------------- ---- ---------------- \$Directory
  0xe00000057d70      14    0 R--rwd -------------- ---- ----------------␣
↪\Windows\System32\AuthBroker.dll
  0xe000000599d0   32758    1 R--rw- 0xe00000074580    4 System           ␣
↪\Windows\CSC\v2.0.6
  0xe000000686e0      19    0 RW-rwd -------------- ---- ---------------- \$Directory
  0xe0000006a1f0      19    0 RW-rwd -------------- ---- ---------------- \$Directory
  0xe0000006b5a0      16    0 R--r-d -------------- ---- ----------------␣
↪\Windows\Fonts\modern.fon
  0xe0000006d8c0       4    0 R--r-d -------------- ---- ----------------␣
↪\Windows\System32\negoexts.dll
```

```
  0xe0000006dc40     16   0 R--r-- -------------- ---- ----------------␣
↪\Windows\Fonts\meiryob.ttc
  0xe0000006e1f0  29617   1 ------ 0xe0000204a900 2628 winpmem_1.5.2.   \Connect
  0xe0000006edd0     16   0 R--rwd -------------- ---- ----------------␣
↪\Windows\System32\msctf.dll
  0xe00000079270     16   0 R--r-- -------------- ---- ----------------␣
↪\Windows\Cursors\aero_up.cur
  0xe0000007abc0     12   0 R--rwd -------------- ---- ----------------␣
↪\Windows\System32\puiobj.dll
  0xe0000007ba90     18   0 RW-rwd -------------- ---- ---------------- \$Directory
  0xe0000007e070      3   0 R--r-- -------------- ---- ----------------␣
↪\Windows\Fonts\segoeui.ttf
  0xe0000007e360      4   0 RW-rwd -------------- ---- ---------------- \
↪$ConvertToNonresident
  0xe0000007e890      7   0 R--r-d -------------- ---- ----------------␣
↪\Windows\System32\usbmon.dll
  0xe0000007f360  32768   1 R--r-d 0xe000000ce080  432 wininit.exe     ␣
↪\Windows\System32\en-GB\user32.dll.mui
  0xe0000007f980      4   0 R--r-d -------------- ---- ----------------␣
↪\Windows\System32\KBDUK.DLL
  0xe000000b1d90     17   0 RW-rwd -------------- ---- ---------------- \$Directory
  0xe000000b1f20      5   0 R--r-d -------------- ---- ----------------␣
↪\Windows\System32\AppXDeploymentServer.dll
  0xe000000b4610     12   0 R--rwd -------------- ---- ----------------␣
↪\Windows\SysWOW64\winmmbase.dll
  0xe000000b6820      1   1 RWD--- 0xe00000074580    4 System          ␣
↪\Windows\System32\config\RegBack\SECURITY
  0xe000000b6a50  32766   1 RW---- 0xe00000074580    4 System          ␣
↪\Windows\System32\config\SECURITY.LOG2
```

show_referrer_alloc (FindReferenceAlloc)

Show allocations that refer to an address.

Plugin	Type	Description
address	IntParser	The address to display
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

gahti (Gahti)

Dump the USER handle type information.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

tokens (GetSIDs)

Print the SIDs owning each process token.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

In windows a process runs with a set of *Tokens*. These tokens are used to enforce Windows Mandatory ACL system. From a forensic point of view it is interesting to see what tokens a process is running with.

For non system processes, the process will also possess the token of the user who started it.

### Sample output

In the below we can see that this cmd.exe process was started by the user *test* with SID *S-1-5-21-1077689984-2177008626-1601812314-1001*.

```
win8.1.raw 22:41:01> tokens
-------------------> tokens()
Process           Pid   Sid                                                   Comment
----------------- ----- ----------------------------------------------------- -------
...
cmd.exe           888   S-1-5-21-1077689984-2177008626-1601812314-1001     User: test
cmd.exe           888   S-1-5-21-1077689984-2177008626-1601812314-513      Domain Users
cmd.exe           888   S-1-1-0                                               Everyone
cmd.exe           888   S-1-5-114
cmd.exe           888   S-1-5-21-1077689984-2177008626-1601812314-1002
cmd.exe           888   S-1-5-32-544                                          ␣
↪Administrators
cmd.exe           888   S-1-5-32-545                                          Users
cmd.exe           888   S-1-5-4                                               Interactive
cmd.exe           888   S-1-2-1                                               Console␣
↪Logon (Users who are logged onto the physical console)
cmd.exe           888   S-1-5-11                                              ␣
↪Authenticated Users
cmd.exe           888   S-1-5-15                                              This␣
↪Organization
cmd.exe           888   S-1-5-113
cmd.exe           888   S-1-5-5-0-126935                                      Logon␣
↪Session
cmd.exe           888   S-1-2-0                                               Local␣
↪(Users with the ability to log in locally)
cmd.exe           888   S-1-5-64-10                                           NTLM␣
↪Authentication
cmd.exe           888   S-1-16-12288                                          High␣
↪Mandatory Level
...
```

getservicesids (GetServiceSids)

Get the names of services in the Registry and return Calculated SID

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The getservicesids command calculates the SIDs for services on a machine. The service names are taken from the registry ("SYSTEMCurrentControlSetServices")

### Sample output

```
win8.1.raw 16:58:23> getservicesids
-------------------> getservicesids()
SID                                                                      Service Name
----------------------------------------------------------------------- ------------
S-1-5-80-3476726845-1218940557-3240126423-1396283824-3706223860         .NET CLR Data
S-1-5-80-3749761688-76038143-2425834820-4129736068-309120712            .NET CLR␣
↪Networking
S-1-5-80-4151353957-356578678-4163131872-800126167-2037860865           .NET CLR␣
↪Networking 4.0.0.0
S-1-5-80-603392709-3706100282-1779817366-3290147925-2109454977          .NET Data␣
↪Provider for Oracle
S-1-5-80-1168016597-2140435647-491797002-352772175-817350590            .NET Data␣
↪Provider for SqlServer
S-1-5-80-1135273183-3738781202-689480478-891280274-255333391            .NET Memory␣
↪Cache 4.0
S-1-5-80-255220978-1106536095-1636044468-311807000-281316439            .NETFramework
S-1-5-80-799694863-4024754253-4060439485-3284853837-2852070736          1394ohci
S-1-5-80-3459415445-2224257447-3423677131-2829651752-4257665947         3ware
S-1-5-80-550892281-1246201444-2906082186-2301917840-2280485454          ACPI
S-1-5-80-2670625634-2386107419-4204951937-4094372046-2600379021         acpiex
S-1-5-80-3267050047-1503497915-401953950-2662906978-1179039408          acpipagr
```

guess_guid (GuessGUID)

Try to guess the exact version of a kernel module by using an index.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
minimal_match	IntParser	The minimal number of comparison points to be considered. Sometimes not all comparison points can be used since they may not be mapped.
module	String	The name of the module to guess.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

handles (Handles)

Print list of open handles for each process

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
named_only	Boolean	Output only handles with a name .
object_types	ArrayStringParser	Types of objects to show.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

This plugin displays the handle table of processes. The handle table in the process stores securable kernel objects.

When a user mode process obtains a securable kernel object, they receive a handle to it - i.e. an integer which is the location in the handle table, rather than the raw kernel level pointer. User processes then use the handle to operate of the kernel level object. For example, if a process opens a file the **_FILE_OBJECT** will be stored in the handle table, and the userspace code will receive the offset into the handle table.

This plugin is especially useful to find all resources that are opened by a user space program, such as open files, registry keys etc. In fact any of the objects shown by the [object_types](ObjectTypes.html) plugin are stored in the handle table as can be seen by this module.

All the usual process selectors are supported. Additionally, it is possible to filter the output by using a comma separated list of handle types (as can be seen by the [object_types](ObjectTypes.html) plugin.

### Sample output

In the following output we see the winpmem acquisition tool's handle table. Note that it has an open file to the raw device *Devicepmem* and the output file of *DeviceHarddiskVolume2tempwin8.1.raw*.

```
win8.1.raw 18:00:43> handles proc_regex="winpmem"
-------------------> handles(proc_regex="winpmem")
  Offset (V)      Pid     Handle         Access     Type             Details
-------------- ------ -------------- -------------- ---------------- -------
0xe00001f82f20   2628            0x4       0x12019f File                ␣
↪\Device\ConDrv\Reference
0xe00001d17e00   2628           0x10       0x100020 File                ␣
↪\Device\HarddiskVolume2\Windows
0xe00001f546b0   2628           0x18       0x12019f File                ␣
↪\Device\ConDrv\Input
0xe00001eef800   2628           0x1c       0x12019f File                ␣
↪\Device\ConDrv\Output
0xe00001eef800   2628           0x20       0x12019f File                ␣
↪\Device\ConDrv\Output
0xe00001d0db80   2628           0x24       0x100020 File                ␣
↪\Device\HarddiskVolume2\temp
0xe0000006e1f0   2628           0x28       0x12019f File                ␣
↪\Device\ConDrv\Connect
0xe00000637480   2628           0x30       0x1f0001 ALPC Port
0xe000006bd290   2628           0x34       0x1f0003 Event
0xe00001ed6060   2628           0x38            0x1 WaitCompletionPacket
0xe00001ecd080   2628           0x3c       0x1f0003 IoCompletion
0xe00001ec7060   2628           0x40        0xf00ff TpWorkerFactory
0xe00000778320   2628           0x44       0x100002 IRTimer
0xe00001ecfb80   2628           0x48            0x1 WaitCompletionPacket
0xe00001a629d0   2628           0x4c       0x100002 IRTimer
0xe00001ec8f90   2628           0x50            0x1 WaitCompletionPacket
0xe00002048970   2628           0x54          0x804 EtwRegistration
0xe0000077dd00   2628           0x58       0x100003 Semaphore
0xe00001d1b340   2628           0x5c       0x100001 File             \Device\CNG
0xe000006b82c0   2628           0x60       0x100003 Semaphore
0xe00001d0c6e0   2628           0x64       0x120196 File                ␣
↪\Device\HarddiskVolume2\temp\win8.1.raw
0xe000007db2f0   2628           0x74       0x1f0003 Event
0xe000023eda60   2628           0x78          0x804 EtwRegistration
0xe000024c56c0   2628           0x7c          0x804 EtwRegistration
0xe00001f803e0   2628           0x80          0x804 EtwRegistration
0xe00000813330   2628           0x84       0x1f0003 Event
0xe00001254440   2628           0x88       0x1fffff Thread           TID 3420 PID 2628
0xe0000061ebb0   2628           0x8c       0x1f0001 ALPC Port
0xe00001d0c340   2628           0x90       0x12019f File             \Device\pmem
```

hivedump (HiveDump)

Prints out a hive

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

hives (Hives)

List all the registry hives on the system.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

hooks_iat (IATHooks)

Detect IAT/EAT hooks in process and kernel memory

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

imageinfo (ImageInfo)

List overview information about this image.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

This plugin prints an overview of certain parameters of the image.

### Notes

1. Since Rekall does not require users to select the profiles manually this plugin is not required to be run prior to any analysis. In fact the plugin itself needs to have accurate profiles loaded. It therefore does not server the same purpose as in previous version of the software.

### Sample output

```
win8.1.raw 18:00:48> imageinfo
-------------------> imageinfo()
Fact                 Value
-------------------- -----
Kernel DTB           0x1a7000
```

```
NT Build                9600.winblue_gdr.130913-2141
NT Build Ex             9600.16404.amd64fre.winblue_gdr.130913-2141
Signed Drivers          -
Time (UTC)              2014-01-24 21:20:05+0000
Time (Local)            2014-01-24 21:20:05+0000
Sec Since Boot          764.359375
NtSystemRoot            C:\Windows
*************** Physical Layout ***************
Physical Start  Physical End  Number of Pages
-------------- -------------- ---------------
0x000000001000 0x00000009f000 158
0x000000100000 0x000000102000 2
0x000000103000 0x00003fff0000 261869
```

impscan (ImpScan)

Scan for calls to imported functions.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

### Sample output

```
win8.1.raw 18:30:34> impscan proc_regex="dwm.exe"
-------------------> impscan(proc_regex="dwm.exe")
************************************************
Process dwm.exe PID 692
     IAT            Call      Module                Function
-------------- -------------- -------------------- --------
0x7ff7474f4000 0x7ff87f2c369c sechost.dll          ␣
↪ConvertStringSecurityDescriptorToSecurityDescriptorW
0x7ff7474f4030 0x7ff87b48beb0 uxtheme.dll          CloseThemeData
0x7ff7474f4038 0x7ff87b4bfc80 uxtheme.dll          OpenThemeData
0x7ff7474fa020 0x7ff87e4b5d34 msvcrt.dll           382
0x7ff7474fa030 0x7ff87e4b5f18 msvcrt.dll           410
0x7ff7474fa050 0x7ff87e4b9948 msvcrt.dll           144
0x7ff7474fa058 0x7ff87e4babc0 msvcrt.dll           129
0x7ff7474fa0e0 0x7ff87e4b468c msvcrt.dll           35
0x7ff7474fa0e8 0x7ff87e4b1cd4 msvcrt.dll           36
0x7ff7474fa120 0x7ff87f38f85c ntdll.dll            1252
0x7ff7474fa128 0x7ff87f36e384 ntdll.dll            1229
0x7ff7474fa130 0x7ff87c9a3dec KERNELBASE.dll       170
0x7ff7474fa138 0x7ff87f33c31c ntdll.dll            815
0x7ff7474fa148 0x7ff87f383270 ntdll.dll            RtlInitializeCriticalSection
0x7ff7474fa158 0x7ff87f36d100 ntdll.dll            RtlAcquireSRWLockShared
0x7ff7474fa168 0x7ff87f36b810 ntdll.dll            RtlLeaveCriticalSection
0x7ff7474fa170 0x7ff87c9a24f4 KERNELBASE.dll       157
0x7ff7474fa180 0x7ff87f36e50c ntdll.dll            1228
0x7ff7474fa188 0x7ff87f35db60 ntdll.dll            RtlAcquireSRWLockExclusive
0x7ff7474fa190 0x7ff87f36b550 ntdll.dll            867
```

```
0x7ff7474fa1a0 0x7ff87c9a14a0 KERNELBASE.dll       635
0x7ff7474fa1c8 0x7ff87c9a1440 KERNELBASE.dll       481
0x7ff7474fa1e8 0x7ff87f37c7c0 ntdll.dll            RtlSetLastWin32Error
0x7ff7474fa1f8 0x7ff87f366b90 ntdll.dll            928
0x7ff7474fa200 0x7ff87f3620d0 ntdll.dll            RtlAllocateHeap
0x7ff7474fa208 0x7ff87c9ac960 KERNELBASE.dll       684
0x7ff7474fa218 0x7ff87c9a14e0 KERNELBASE.dll       554
0x7ff7474fa230 0x7ff87edd3184 KERNEL32.DLL         GetStartupInfoW
0x7ff7474fa238 0x7ff87edd3074 KERNEL32.DLL         SetPriorityClass
```

hooks_inline (InlineHooks)

Detect API hooks in process and kernel memory

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

inspect_heap (InspectHeap)

Inspect the process heap.

This prints a lot of interesting facts about the process heap. It is also the foundation to many other plugins which find things in the process heaps.

NOTE: Currently we only support Windows 7 64 bit.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
free	Boolean	Also show freed chunks.
heaps	ArrayIntParser	Only show these heaps (default show all)
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

kdbgscan (KDBGScan)

Scan for possible _KDDEBUGGER_DATA64 structures.

The scanner is detailed here: http://moyix.blogspot.com/2008/04/finding-kernel-global-variables-in.html

The relevant structures are detailed here: http://doxygen.reactos.org/d3/ddf/include_2psdk_2wdbgexts_8h_source.html

We can see that _KDDEBUGGER_DATA64.Header is:

```
typedef struct _DBGKD_DEBUG_DATA_HEADER64 {
  LIST_ENTRY64    List;
  ULONG           OwnerTag;
  ULONG           Size;
}
```

We essentially search for an owner tag of "KDBG", then overlay the _KDDEBUGGER_DATA64 struct on it. We test for validity by reflecting through the Header.List member.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
full_scan	Boolean	Scan the full address space.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

Windows keeps a store of some useful global variables in a structure called **_KDDEBUGGER_DATA64**. This information is used by the microsoft kernel debugger in order to bootstap the analysis of a crash dump.

Rekall no longer uses the Kernel Debugger Block for analysis - instead accurate global symbol information are fetched from Microsoft PDB files containing debugging symbols.

### Notes

1. Previous versions of Rekall used the KDBG heavily for analysis, and by extension used this plugin. Currently the KDBG is not used by Rekall at all so this plugin is not all that useful.

kpcr (KPCR)

A plugin to print all KPCR blocks.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

Windows maintains per-processor information for each physical CPU in the system. This plugin displays this infomation.

### Sample output

```
win8.1.raw 21:15:09> kpcr
-------------------> kpcr()
**************************************************
Property                       Value
------------------------------ -----
Offset (V)                     0xf802d3307000
KdVersionBlock                 Pointer to -
IDT                            0xf802d4a43080
GDT                            0xf802d4a43000
CurrentThread                 : 0xe00001254440 TID 3420 (winpmem_1.5.2.:2628)
IdleThread                    : 0xf802d335fa80 TID 0 (System:0)
Details                       : CPU 0 (GenuineIntel @ 2517 MHz)
CR3/DTB                       : 0x1a7000
```

ldrmodules (LdrModules)

Detect unlinked DLLs

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

There are many ways to hide a DLL. One of the ways involves unlinking the DLL from one (or all) of the linked lists in the PEB. However, when this is done, there is still information contained within the VAD (Virtual Address Descriptor) which identifies the base address of the DLL and its full path on disk. To cross-reference this information (known as memory mapped files) with the 3 PEB lists, use the ldrmodules command.

For each memory mapped PE file, the ldrmodules command prints True or False if the PE exists in the PEB lists.

```
win8.1.raw 22:17:36> ldrmodules proc_regex="winpmem"
-------------------> ldrmodules(proc_regex="winpmem")
Pid      Process                    Base           InLoad InInit InMem MappedPath
-------- -------------------- -------------- ------ ------ ----- ----------
2628     winpmem_1.5.2.       0x0000753b0000 False False False␣
↪\Windows\SysWOW64\KernelBase.dll
2628     winpmem_1.5.2.       0x000000020000 True  False True  \temp\winpmem_1.5.2.exe
2628     winpmem_1.5.2.       0x000076c30000 False False False␣
↪\Windows\SysWOW64\kernel32.dll
2628     winpmem_1.5.2.       0x000074a40000 False False False␣
↪\Windows\SysWOW64\cryptbase.dll
2628     winpmem_1.5.2.       0x000074a50000 False False False␣
↪\Windows\SysWOW64\sspicli.dll
2628     winpmem_1.5.2.       0x000077010000 True  True  True ␣
↪\Windows\System32\wow64cpu.dll
2628     winpmem_1.5.2.       0x000076f50000 True  True  True ␣
↪\Windows\System32\wow64.dll
2628     winpmem_1.5.2.       0x000076fa0000 True  True  True ␣
↪\Windows\System32\wow64win.dll
2628     winpmem_1.5.2.       0x000075250000 False False False␣
↪\Windows\SysWOW64\rpcrt4.dll
2628     winpmem_1.5.2.       0x7ff87f320000 True  True  True ␣
↪\Windows\System32\ntdll.dll
2628     winpmem_1.5.2.       0x000077020000 False False False␣
↪\Windows\SysWOW64\ntdll.dll
2628     winpmem_1.5.2.       0x0000749e0000 False False False␣
↪\Windows\SysWOW64\bcryptprimitives.dll
2628     winpmem_1.5.2.       0x000074ff0000 False False False␣
↪\Windows\SysWOW64\advapi32.dll
2628     winpmem_1.5.2.       0x000076f10000 False False False␣
↪\Windows\SysWOW64\sechost.dll
2628     winpmem_1.5.2.       0x000074d80000 False False False␣
↪\Windows\SysWOW64\msvcrt.dll
```

Since the PEB and the DLL lists that it contains all exist in user mode, its also possible for malware to hide (or obscure) a DLL by simply overwriting the path. Tools that only look for unlinked entries may miss the fact that malware could overwrite *C:bad.dll* to show *C:windowssystem32kernel32.dll*. So you can also pass the *verbosity=10* parameter to ldrmodules to see the full path of all entries.

For concrete examples, see [ZeroAccess Misleads Memory-File Link](http://blogs.mcafee.com/mcafee-labs/zeroaccess-misleads-memory-file-link) and [QuickPost: Flame & Volatility](http://mnin.blogspot.com/2012/06/quickpost-flame-volatility.html).

```
win8.1.raw 22:17:41> ldrmodules proc_regex="winpmem", verbosity=10
-------------------> ldrmodules(proc_regex="winpmem", verbosity=10)
Pid      Process                    Base      InLoad InInit InMem MappedPath
-------- -------------------- --------------- ------ ------ ----- ----------
2628     winpmem_1.5.2.       0x0000753b0000 False False False␣
↪\Windows\SysWOW64\KernelBase.dll
2628     winpmem_1.5.2.       0x000000020000 True  False True  \temp\winpmem_1.5.2.exe
  Load Path: C:\temp\winpmem_1.5.2.exe : winpmem_1.5.2.exe
  Mem Path: C:\temp\winpmem_1.5.2.exe : winpmem_1.5.2.exe
2628     winpmem_1.5.2.       0x000076c30000 False False False␣
↪\Windows\SysWOW64\kernel32.dll
2628     winpmem_1.5.2.       0x000074a40000 False False False␣
↪\Windows\SysWOW64\cryptbase.dll
2628     winpmem_1.5.2.       0x000074a50000 False False False␣
↪\Windows\SysWOW64\sspicli.dll
2628     winpmem_1.5.2.       0x000077010000 True  True  True ␣
↪\Windows\System32\wow64cpu.dll
  Load Path: C:\Windows\system32\wow64cpu.dll : wow64cpu.dll
  Init Path: C:\Windows\system32\wow64cpu.dll : wow64cpu.dll
  Mem Path: C:\Windows\system32\wow64cpu.dll : wow64cpu.dll
2628     winpmem_1.5.2.       0x000076f50000 True  True  True ␣
↪\Windows\System32\wow64.dll
  Load Path: C:\Windows\SYSTEM32\wow64.dll : wow64.dll
  Init Path: C:\Windows\SYSTEM32\wow64.dll : wow64.dll
  Mem Path: C:\Windows\SYSTEM32\wow64.dll : wow64.dll
2628     winpmem_1.5.2.       0x000076fa0000 True  True  True ␣
↪\Windows\System32\wow64win.dll
  Load Path: C:\Windows\system32\wow64win.dll : wow64win.dll
  Init Path: C:\Windows\system32\wow64win.dll : wow64win.dll
  Mem Path: C:\Windows\system32\wow64win.dll : wow64win.dll
2628     winpmem_1.5.2.       0x000075250000 False False False␣
↪\Windows\SysWOW64\rpcrt4.dll
2628     winpmem_1.5.2.       0x7ff87f320000 True  True  True ␣
↪\Windows\System32\ntdll.dll
  Load Path: C:\Windows\SYSTEM32\ntdll.dll : ntdll.dll
  Init Path: C:\Windows\SYSTEM32\ntdll.dll : ntdll.dll
  Mem Path: C:\Windows\SYSTEM32\ntdll.dll : ntdll.dll
2628     winpmem_1.5.2.       0x000077020000 False False False␣
↪\Windows\SysWOW64\ntdll.dll
2628     winpmem_1.5.2.       0x0000749e0000 False False False␣
↪\Windows\SysWOW64\bcryptprimitives.dll
2628     winpmem_1.5.2.       0x000074ff0000 False False False␣
↪\Windows\SysWOW64\advapi32.dll
2628     winpmem_1.5.2.       0x000076f10000 False False False␣
↪\Windows\SysWOW64\sechost.dll
2628     winpmem_1.5.2.       0x000074d80000 False False False␣
↪\Windows\SysWOW64\msvcrt.dll
```

### Notes

1. Wow64 processes (i.e. 32 bit processes on 64 bit windows) will not show any 32 bit DLLs in any of the loader lists. This is normal (and you will see the Dlls loaded from the WindowsWow64 directory.

load_profile (LoadWindowsProfile)

Loads the profile into the session.

If the profile does not exist in the repositories, fetch and build it from the symbol server. This plugin allows the user to change resolution of selected binaries by forcing the fetching of symbol files from the symbol server interactively.

Plugin	Type	Description
guid	String	The guid of the module.
module_name	String	The name of the module (without the .pdb extensilon).
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

malfind (Malfind)

Find hidden and injected code

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
dump_dir	String	Path suitable for dumping files.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The malfind command helps find hidden or injected code/DLLs in user mode memory, based on characteristics such as VAD tag and page permissions.

Note: malfind does not detect DLLs injected into a process using **CreateRemoteThread->LoadLibrary**. DLLs injected with this technique are not hidden and thus you can view them with dlllist. The purpose of malfind is to locate DLLs that standard methods/tools do not see.

Here is an example of using it to detect the presence of Zeus. The first memory segment (starting at 0x2aa0000) was detected because it is executable, marked as private (not shared between processes) and has a VadS tag… which means there is no memory mapped file already occupying the space. Based on a disassembly of the data found at this address, it seems to contain some API hook trampoline stubs.

The second memory segment (starting at 0x3080000) was detected because it contained an executable that isn't listed in the PEB's module lists.

If you want to save extracted copies of the memory segments identified by malfind, just supply an output directory with the *dump_dir* parameter. In this case, an unpacked copy of the Zeus binary that was injected into explorer.exe would be written to disk.

```
zeus2x4.vmem 22:53:43> malfind proc_regex="explorer"
---------------------> malfind(proc_regex="explorer")
**************************************************f pid 1752
Process: explorer.exe Pid: 1752 Address: 0x2aa0000
Vad Tag: VadS Protection: EXECUTE_READWRITE
Flags: CommitCharge: 1, MemCommit: 1, PrivateMemory: 1, Protection: 6

 0x2aa0000 b8 35 00 00 00 e9 a9 d1 e6 79 68 6c 02 00 00 e9  .5.......yhl....
 0x2aa0010 b4 63 e7 79 8b ff 55 8b ec e9 7c 11 d7 79 8b ff  .c.y..U...|..y..
 0x2aa0020 55 8b ec e9 01 32 77 74 8b ff 55 8b ec e9 7c 60  U....2wt..U...|`
 0x2aa0030 72 74 8b ff 55 8b ec e9 ca e9 72 74 8b ff 55 8b  rt..U.....rt..U.
```

```
0x02aa0000      b835000000           MOV EAX, 0x35
0x02aa0005      e9a9d1e679           JMP 0x7c90d1b3
0x02aa000a      686c020000           PUSH DWORD 0x26c
0x02aa000f      e9b463e779           JMP 0x7c9163c8
0x02aa0014      8bff                 MOV EDI, EDI
0x02aa0016      55                   PUSH EBP
0x02aa0017      8bec                 MOV EBP, ESP
0x02aa0019      e97c11d779           JMP 0x7c81119a
0x02aa001e      8bff                 MOV EDI, EDI
0x02aa0020      55                   PUSH EBP
0x02aa0021      8bec                 MOV EBP, ESP
0x02aa0023      e901327774           JMP 0x77213229
0x02aa0028      8bff                 MOV EDI, EDI
0x02aa002a      55                   PUSH EBP
0x02aa002b      8bec                 MOV EBP, ESP
0x02aa002d      e97c607274           JMP 0x771c60ae
0x02aa0032      8bff                 MOV EDI, EDI
0x02aa0034      55                   PUSH EBP
0x02aa0035      8bec                 MOV EBP, ESP
0x02aa0037      e9cae97274           JMP 0x771cea06
0x02aa003c      8bff                 MOV EDI, EDI
0x02aa003e      55                   PUSH EBP
0x02aa003f      8bec                 MOV EBP, ESP
0x02aa0041      e9e8327774           JMP 0x7721332e
0x02aa0046      8bff                 MOV EDI, EDI
0x02aa0048      55                   PUSH EBP
0x02aa0049      8bec                 MOV EBP, ESP
0x02aa004b      e9494d7274           JMP 0x771c4d99
0x02aa0050      8bff                 MOV EDI, EDI
0x02aa0052      55                   PUSH EBP
0x02aa0053      8bec                 MOV EBP, ESP
0x02aa0055      e99d827274           JMP 0x771c82f7
0x02aa005a      8bff                 MOV EDI, EDI
0x02aa005c      55                   PUSH EBP
0x02aa005d      8bec                 MOV EBP, ESP
0x02aa005f      e9ef927574           JMP 0x771f9353
0x02aa0064      8bff                 MOV EDI, EDI
0x02aa0066      55                   PUSH EBP
0x02aa0067      8bec                 MOV EBP, ESP
0x02aa0069      e9fe897374           JMP 0x771d8a6c
0x02aa006e      6a2c                 PUSH 0x2c
0x02aa0070      68187b1c77           PUSH DWORD 0x771c7b18
0x02aa0075      e957797274           JMP 0x771c79d1
0x02aa007a      8bff                 MOV EDI, EDI
0x02aa007c      55                   PUSH EBP
0x02aa007d      8bec                 MOV EBP, ESP
0x02aa007f      e9ac3d016f           JMP 0x71ab3e30
0x02aa0084      8bff                 MOV EDI, EDI
0x02aa0086      55                   PUSH EBP
0x02aa0087      8bec                 MOV EBP, ESP
0x02aa0089      e99e4b016f           JMP 0x71ab4c2c
0x02aa008e      8bff                 MOV EDI, EDI
0x02aa0090      55                   PUSH EBP
0x02aa0091      8bec                 MOV EBP, ESP
0x02aa0093      e96768016f           JMP 0x71ab68ff
0x02aa0098      8bff                 MOV EDI, EDI
0x02aa009a      55                   PUSH EBP
```

```
0x02aa009b      8bec                    MOV EBP, ESP
0x02aa009d      e9598b977b              JMP 0x7e418bfb
0x02aa00a2      8bff                    MOV EDI, EDI
0x02aa00a4      55                      PUSH EBP
0x02aa00a5      8bec                    MOV EBP, ESP
0x02aa00a7      e9130d997b              JMP 0x7e430dbf
0x02aa00ac      8bff                    MOV EDI, EDI
0x02aa00ae      55                      PUSH EBP
**************************************************
Process: explorer.exe Pid: 1752 Address: 0x3080000
Vad Tag: VadS Protection: EXECUTE_READWRITE
Flags: CommitCharge: 52, MemCommit: 1, PrivateMemory: 1, Protection: 6

 0x3080000 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00  MZ..............
 0x3080010 b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ........@.......
 0x3080020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
 0x3080030 00 00 00 00 00 00 00 00 00 00 00 00 c0 00 00 00  ................

0x03080000      4d                      DEC EBP
0x03080001      5a                      POP EDX
0x03080002      90                      NOP
0x03080003      0003                    ADD [EBX], AL
0x03080005      0000                    ADD [EAX], AL
0x03080007      000400                  ADD [EAX+EAX], AL
0x0308000a      0000                    ADD [EAX], AL
0x0308000c      ff                      DB 0xff
0x0308000d      ff00                    INC DWORD [EAX]
0x0308000f      00b800000000            ADD [EAX+0x0], BH
0x03080015      0000                    ADD [EAX], AL
0x03080017      004000                  ADD [EAX+0x0], AL
0x0308001a      0000                    ADD [EAX], AL
0x0308001c      0000                    ADD [EAX], AL
0x0308001e      0000                    ADD [EAX], AL
```

mftdump (MftDump)

Enumerate MFT entries from the cache manager.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

mimikatz (Mimikatz)

Extract and decrypt passwords from the LSA Security Service.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

miranda (Miranda)

A mixin for plugins which require a valid kernel address space.

Args: dtb: A potential dtb to be used.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

modscan (ModScan)

Scan Physical memory for _LDR_DATA_TABLE_ENTRY objects.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayInt-Parser	Kernel addresses of eprocess structs.
limit	IntParser	The length of data to search in each selected region.
method	ChoiceArray	Method to list processes.
pids	ArrayInt-Parser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
scan_kernel	Boolean	Scan the entire kernel address space.
scan_kernel_code	Boolean	Scan the kernel image and loaded drivers.
scan_kernel_nonpaged_pool	Boolean	Scan the kernel non-paged pool.
scan_kernel_paged_pool	Boolean	Scan the kernel paged pool.
scan_kernel_session_pools	Boolean	Scan session pools for all processes.
scan_physical	Boolean	Scan the physical address space only.
scan_process_memory	Boolean	Scan all of process memory. Uses process selectors to narrow down selections.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The modscan command finds LDR_DATA_TABLE_ENTRY structures by scanning physical memory for pool tags. This can pick up previously unloaded drivers and drivers that have been hidden/unlinked by rootkits.

### Notes

1. Like other pool scanning plugins, this plugin may produce false positives since it essentially carves **_LDR_DATA_TABLE_ENTRY** structures out of memory. On the other hand, this plugin may reveal files which have been closed or freed.

### Sample output

In this example we can identify the pmem driver which was loaded from a temporary location.

```
win8.1.raw 23:27:24> modscan
-------------------> modscan()
  Offset(P)     Name                      Base            Size       File
-------------- -------------------- -------------- -------------- ----
0x000001ce507e                      0x20c483483824     0xebc08b44
0x00003ce163b0 mrxsmb.sys           0xf80002174000        0x6d000␣
↪\SystemRoot\system32\DRIVERS\mrxsmb.sys
0x00003ce17610 mrxsmb20.sys         0xf80002000000        0x39000␣
↪\SystemRoot\system32\DRIVERS\mrxsmb20.sys
```

```
0x00003ce1e830 mpsdrv.sys           0xf8000215d000        0x17000␣
↪\SystemRoot\System32\drivers\mpsdrv.sys
0x00003ce4cf30 Ndu.sys              0xf800022cd000        0x1d000␣
↪\SystemRoot\system32\drivers\Ndu.sys
0x00003ce4df20 mrxsmb10.sys         0xf80002282000        0x4b000␣
↪\SystemRoot\system32\DRIVERS\mrxsmb10.sys
0x00003ce80170 peauth.sys           0xf800022ea000        0xa9000␣
↪\SystemRoot\system32\drivers\peauth.sys
0x00003ce8b010 srvnet.sys           0xf8000239e000        0x43000␣
↪\SystemRoot\System32\DRIVERS\srvnet.sys
0x00003ce8bc20 secdrv.SYS           0xf80002393000         0xb000␣
↪\SystemRoot\System32\Drivers\secdrv.SYS
0x00003ceae280 tcpipreg.sys         0xf800023e1000        0x12000␣
↪\SystemRoot\System32\drivers\tcpipreg.sys
0x00003ceae520 srv2.sys             0xf800024ec000        0xad000␣
↪\SystemRoot\System32\DRIVERS\srv2.sys
0x00003cec9ee0                      0x665602050006            0x0
0x00003ceede60 srv.sys              0xf80002400000        0x98000␣
↪\SystemRoot\System32\DRIVERS\srv.sys
0x00003cf44eb0 mslldp.sys           0xf80002498000        0x16000␣
↪\SystemRoot\system32\DRIVERS\mslldp.sys
0x00003d144160 rspndr.sys           0xf80001caf000        0x18000␣
↪\SystemRoot\system32\DRIVERS\rspndr.sys
0x00003d145a50 lltdio.sys           0xf80001c9b000        0x14000␣
↪\SystemRoot\system32\DRIVERS\lltdio.sys
0x00003d18c850 HTTP.sys             0xf80002043000        0xfa000␣
↪\SystemRoot\system32\drivers\HTTP.sys
0x00003d29b010 pmeA86F.tmp          0xf800025ca000        0x10000 \??
↪\C:\Users\test\AppData\Local\Temp\pmeA86F.tmp
0x00003d655520 HdAudio.sys          0xf80001d45000        0x66000␣
↪\SystemRoot\system32\drivers\HdAudio.sys
0x00003d6593e0 tunnel.sys           0xf800024ae000        0x2d000␣
↪\SystemRoot\system32\DRIVERS\tunnel.sys
```

version_modules (ModVersions)

Try to determine the versions for all kernel drivers.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
name_regex	RegEx	Filter module names by this regex.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

Each time a windows binary is built using the Microsoft Visual Studio compiler suite a new unique GUID is generated for this file. The GUID is used to link the executable and the pdb file (which contains debugging symbols).

The GUID is embedded in the executable in an *RSDS* record (i.e. the record has a signature starting with the letters *RSDS*). Rekall can scan for this signature in order to identify the executable version.

This plugin scans for the version string for each loaded kernel module. Use the [version_scan](VersionScan.html) module to search for RSDS signatures in physical memory.

### Sample output

```
win7_trial_64bit.dmp.E01 23:48:26> version_modules
  Offset (V)    Name                    GUID/Version                        PDB
```

```
-------------- -------------------- -------------------------------- -----------------
↪-------------
0xf800027f4b0c ntoskrnl.exe         C07170995AA8441B952E3B9AE3F3754B2 ntkrnlmp.pdb
0xf8000262deb4 hal.dll              0C72B43B8AC64E22AB88B564E69330372 hal.pdb
0xf88002d34af4 wanarp.sys           7BA2309F029F4DE7878AED80636C2D132 wanarp.pdb
0xf8800183eed4 TDI.SYS              C519554437F04B63BC39FF4E69578DC42 tdi.pdb
0xf88000d95b24 volmgrx.sys          C047BA32ABCB4A948CBB8930F352B1032 volmgrx.pdb
0xf88003de7c60 dump_dumpfve.sys     A2CC4DFB86424750871BCB8E1E841E3C1 dumpfve.pdb
0xf880019d4e00 watchdog.sys         79ACBD31D1BD428A8311AD9D5DCDEAA61 watchdog.pdb
0xf8800111004c cng.sys              F0AA00E320D4468A9D3F7078E2AE2BF52 cng.pdb
0xf88002c2e648 csc.sys              56B7C3B9040B47D9821E6A57E6A5AE4A1 csc.pdb
0xf88000c02f48 CI.dll               5F1BDC2205AC402CB0F09FC7CF17A3701 ci.pdb
0xf88003c3f2dc USBD.SYS             BE6200B21204452DADDF85CED51A5BDE1 usbd.pdb
0xf88002d0a1fc netbios.sys          084EB51DBDE844CF9EAD3B5FDFABDC721 netbios.pdb
0xf88000cc80a0 mcupdate.dll         8C7A27566CD54FB9A00AF26B5BF941651 mcupdate_
↪GenuineIntel.pdb
0xf8800145c920 ndis.sys             40D6C85AC9F74887A652601839A1F56D2 ndis.pdb
0xf880019eb04c rdpencdd.sys         C299649119AC4CC888F37C32A216781A1 RDPENCDD.pdb
0xf88003814d08 srv.sys              20C4A475BE954C10997EAD2C623E40C32 srv.pdb
0xf88003a52c10 raspptp.sys          C9106AFB80474EFCAF9384DA26CC35622 raspptp.pdb
0xf880019b42ec VIDEOPRT.SYS         1B0FC2CC31FE41CEBEAC4ABB7375EA481 videoprt.pdb
0xf88000fda340 PCIIDEX.SYS          2C4F146DA2774ACEA1D5499284DDDB271 pciidex.pdb
0xf88003c2962c HIDCLASS.SYS         1815DD7E268B4BB9BCD5226204CFEC9C1 hidclass.pdb
0xf88000fd105c intelide.sys         B72598DF61A84806B7AC593BA128300C1 intelide.pdb
0xf88003a37320 raspppoe.sys         39B224364B9042649CA0CDB8270762931 raspppoe.pdb
0xf88000e040ec atapi.sys            4E82D8C0AB5A41799B979539D280167D1 atapi.pdb
0xf88002cba464 netbt.sys            840D3E3C828C4D60A905DC82D8CBF8FA2 netbt.pdb
0xf880011f647c kbdclass.sys         D5F7E088FAF44B60A3774197A9ADEEC01 kbdclass.pdb
0xf88000e361f0 amdxata.sys          8D1A5FFBAEEA4D388F8B7B3B9378C3671 amdxata.pdb
0xf880031abb04 srvnet.sys           608D364BC5524794BD70C89773BD51EF2 srvnet.pdb
0xf880028fa614 bowser.sys           26FAC99A52F8439E9A5B8B4B37F90D5B1 bowser.pdb
0xf88002ddb6f4 dfsc.sys             827F5D478C94478299C7FEC7FEE4DAFA1 dfsc.pdb
0xf880011bf9dc fvevol.sys           2FBEA7856251499B87C65A29FC51E6191 fvevol.pdb
0xf80000bc13b0 kdcom.dll            ACC6A823A2844D22B68CD5D48D42381F2 kdcom.pdb
0xf88000fbe5a4 volmgr.sys           39E92F60716140C38C723CDF21B956CD2 volmgr.pdb
0xf88000f5c108 msisadrv.sys         09A612E6691847ED98E4F36F3CC9EE641 msisadrv.pdb
0xf8800183127c tdx.sys              FB912A34EB1A44EC9F65E250879944B52 tdx.pdb
0xf8800119f10c rdyboost.sys         20E6E50C6F9B42589E18D96AD84608DB1 rdyboost.pdb
```

modules (Modules)

Print list of loaded kernel modules.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
name_regex	RegEx	Filter module names by this regex.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

To view the list of kernel drivers loaded on the system, use the modules command. This walks the doubly-linked list of **_LDR_DATA_TABLE_ENTRY** structures pointed to by **PsLoadedModuleList**.

It cannot find hidden/unlinked kernel drivers, however [modscan](ModScan.html) serves that purpose. Also, since this plugin uses list walking techniques, you typically can assume that the order the modules are displayed in the output is the order they were loaded on the system.

### Notes

1. The Base address is the location where the kernel module's PE header is mapped. For example you can examine information about the module's IAT/EAT using the [peinfo](PEInfo.html) plugin, providing the base address.

### Sample output

```
win8.1.raw 23:35:19> modules
-------------------> modules()
  Offset (V)    Name                      Base           Size      File
-------------- --------------------- -------------- -------------- ----
0xe00000057620 ntoskrnl.exe          0xf802d3019000       0x781000␣
↪\SystemRoot\system32\ntoskrnl.exe
0xe00000057530 hal.dll               0xf802d379a000        0x6f000␣
↪\SystemRoot\system32\hal.dll
0xe000000557c0 storahci.sys          0xf800006d9000        0x1d000␣
↪\SystemRoot\System32\drivers\storahci.sys
0xe0000149ade0 mssmbios.sys          0xf800018c4000         0xc000␣
↪\SystemRoot\System32\drivers\mssmbios.sys
0xe000013871e0 Npfs.SYS              0xf800008ba000        0x14000␣
↪\SystemRoot\System32\Drivers\Npfs.SYS
0xe00000055d50 volmgrx.sys           0xf80000393000        0x5f000␣
↪\SystemRoot\System32\drivers\volmgrx.sys
0xe00002145a50 lltdio.sys            0xf80001c9b000        0x14000␣
↪\SystemRoot\system32\DRIVERS\lltdio.sys
0xe00000055e40 volmgr.sys            0xf8000045d000        0x15000␣
↪\SystemRoot\System32\drivers\volmgr.sys
0xe00000054950 fwpkclnt.sys          0xf80001144000        0x6c000␣
↪\SystemRoot\System32\drivers\fwpkclnt.sys
0xe00000054c60 NETIO.SYS             0xf80000d3e000        0x79000␣
↪\SystemRoot\system32\drivers\NETIO.SYS
0xe000014b3500 kbdclass.sys          0xf80001a1f000        0x10000␣
↪\SystemRoot\System32\drivers\kbdclass.sys
0xe00001339b50 drmk.sys              0xf80001c00000        0x1c000␣
↪\SystemRoot\system32\drivers\drmk.sys
0xe00000054b70 ksecpkg.sys           0xf80000db7000        0x34000␣
↪\SystemRoot\System32\Drivers\ksecpkg.sys
0xe00000054100 CLASSPNP.SYS          0xf80000800000        0x56000␣
↪\SystemRoot\System32\drivers\CLASSPNP.SYS
```

mutantscan (MutantScan)

Scan for mutant objects _KMUTANT

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayInt-Parser	Kernel addresses of eprocess structs.
limit	IntParser	The length of data to search in each selected region.
method	ChoiceArray	Method to list processes.
pids	ArrayInt-Parser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
scan_kernel	Boolean	Scan the entire kernel address space.
scan_kernel_code	Boolean	Scan the kernel image and loaded drivers.
scan_kernel_nonpaged_pool	Boolean	Scan the kernel non-paged pool.
scan_kernel_paged_pool	Boolean	Scan the kernel paged pool.
scan_kernel_session_pools	Boolean	Scan session pools for all processes.
scan_physical	Boolean	Scan the physical address space only.
scan_process_memory	Boolean	Scan all of process memory. Uses process selectors to narrow down selections.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

This plugin uses pool scanning techniques to find **_KMUTANT** objects.

Mutants implement a “named semaphore” in windows. This is used by malware to ensure only a single copy of the malware is running at the same time. By analyzing the name of the Mutant that a specific malware strand is using it is possible to tell immediately if the malware is running on the machine.

For more information, see Andreas Schuster’s [Searching for Mutants](http://computer.forensikblog.de/en/2009/04/searching_for_mutants.html).

### Notes

1. Like other pool scanning plugins, this plugin may produce false positives since it essentially carves **_KMUTANT** structures out of memory.
2. It is more efficient to search for named mutants using the [object_tree](ObjectTree.html) plugin - since it does not use pool scanning techniques.
3. When inspecting the output, the **#Hnd** column indicates the number of handles to this **_KMUTANT**. Objects in use will have a non zero value here and are likely to not be freed.

### Sample output

```
win8.1.raw 23:46:56> mutantscan scan_in_kernel=1
-------------------> mutantscan(scan_in_kernel=1)
    Offset(P)      #Ptr #Hnd Signal     Thread           CID  Name
- -------------- ------ ---- ------ -------------- ---------- ----
  0xe0000007f810      3   2  1    0x000000000000            ␣
↪C::Users:test:AppData:Local:Microsoft:Windows:Explorer:thumbcache_sr.db!dfMaintainer
  0xe0000007f8d0      3   2  1    0x000000000000            ␣
↪C::Users:test:AppData:Local:Microsoft:Windows:Explorer:thumbcache_1600.db!
↪dfMaintainer
  0xe000000b8d00  32722   1  1    0x000000000000            BcdSyncMutant
  0xe00000624240  32769   1  0    0xe00000624700   556:1396 F659A567-8ACB-4E4A-92A7-
↪5C2DD1884F72
  0xe000006f4a60  32768   1  0    0xe000006dc080 2332:2460 Instance2:  ESENT␣
↪Performance Data Schema Version 255
  0xe00001253080  32768   1  0    0xe000007fd080   880:3144 Instance3:  ESENT␣
↪Performance Data Schema Version 255
```

```
  0xe00001262360      2  1 1    0x000000000000           ARC_AppRepSettings_Mutex
  0xe00001272530      5  4 1    0x000000000000          ␣
↪C::Users:test:AppData:Local:Microsoft:Windows:Explorer:iconcache_1024.db!
↪dfMaintainer
  0xe000012725f0      5  4 1    0x000000000000          ␣
↪C::Users:test:AppData:Local:Microsoft:Windows:Explorer:iconcache_256.db!dfMaintainer
  0xe000012726b0      5  4 1    0x000000000000          ␣
↪C::Users:test:AppData:Local:Microsoft:Windows:Explorer:iconcache_96.db!dfMaintainer
  0xe00001272770      5  4 1    0x000000000000          ␣
↪C::Users:test:AppData:Local:Microsoft:Windows:Explorer:iconcache_48.db!dfMaintainer
  0xe00001272ac0 131007  4 1    0x000000000000          ␣
↪C::Users:test:AppData:Local:Microsoft:Windows:Explorer:iconcache_32.db!dfMaintainer
  0xe0000128e1e0 131005  4 1    0x000000000000          ␣
↪C::Users:test:AppData:Local:Microsoft:Windows:Explorer:iconcache_16.db!dfMaintainer
  0xe0000129a2c0  32734  1 1    0x000000000000           SmartScreen_AppRepSettings_
↪Mutex
  0xe000012c7950 131061  4 1    0x000000000000          ␣
↪C::Users:test:AppData:Local:Microsoft:Windows:Explorer:iconcache_idx.db!
↪IconCacheInit
  0xe000012c7a10      5  4 1    0x000000000000          ␣
↪C::Users:test:AppData:Local:Microsoft:Windows:Explorer:iconcache_wide_alternate.db!
↪dfMaintainer
  0xe000012c7ad0      5  4 1    0x000000000000          ␣
↪C::Users:test:AppData:Local:Microsoft:Windows:Explorer:iconcache_exif.db!
↪dfMaintainer
  0xe000012c7b90      5  4 1    0x000000000000          ␣
↪C::Users:test:AppData:Local:Microsoft:Windows:Explorer:iconcache_wide.db!
↪dfMaintainer
  0xe000012c7c50      5  4 1    0x000000000000          ␣
↪C::Users:test:AppData:Local:Microsoft:Windows:Explorer:iconcache_sr.db!dfMaintainer
...
```

object_tree (ObjectTree)

Visualize the kernel object tree.

Ref: http://msdn.microsoft.com/en-us/library/windows/hardware/ff557762(v=vs.85).aspx

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
type_regex	RegEx	Filter the type of objects shown.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The windows kernel has the notion of a **Kernel Object**. Objects are managed by the kernel through a dedicated API. Kernel Objects are typically used to manage resources which the kernel manages on behalf of user space, for example, open files are managed via the **_FILE_OBJECT** object.

Objects can be named using a directory structure not unlike a filesystem. Objects are placed inside an **_OBJECT_DIRECTORY** object which contains other objects, including other directories. This means that named kernel objects forma tree in memory.

It is possible to discover all currently in-use named objects by following this object tree in memory, which is what this plugin does. This is an alternative to the scanning approach employed by plugins like **psscan**, **driverscan** etc.

### Notes

1. The object tree only tracks named objects. So for example Process objects are typically not tracked here, but Mutants, SymbolicLinks etc are.
2. It is possible to filter objects by types. So for example to enumerate all Mutants one would use the **type_regex="Mutant"** option.
3. *SymbolicLinks* also contain the timestamp when they were created. Note that SymbolicLinks are typically used to provide userspace access to a kernel driver (via the *CreateFile* api), so a timestamp here is a good indication of when a driver was loaded.

### Sample output

```
# Enumeate all drivers
win7.elf 01:25:12> object_tree type_regex="Driver"
-----------------> object_tree(type_regex="Driver")
_OBJECT_HEADER Type                 Name
-------------- -------------------- --------------------
0xfa80025e5d10 Driver               . mrxsmb10
0xfa80025e1190 Driver               . mrxsmb
0xfa8001953940 Driver               . mrxsmb20
....

# We can examine a specific object using the virtual offset.

win7.elf 01:28:18> x=profile._OBJECT_HEADER(0xfa80019fb8d0)
win7.elf 01:28:34> print x.get_object_type()
Driver

# We can dereference the exact object contained in this header (in this case
#  _DRIVER_OBJECT.

win7.elf 01:28:40> print x.Object
[_DRIVER_OBJECT _DRIVER_OBJECT] @ 0xFA80019FB900
  0x00 Type                [short:Type]: 0x00000004
  0x02 Size                [short:Size]: 0x00000150
  0x08 DeviceObject       <_DEVICE_OBJECT Pointer to [0xFA80019FB550] (DeviceObject)>
  0x10 Flags               [unsigned long:Flags]: 0x00000012
  0x18 DriverStart        <Void Pointer to [0xF88003B45000] (DriverStart)>
  0x20 DriverSize          [unsigned long:DriverSize]: 0x0000B000
  0x28 DriverSection      <Void Pointer to [0xFA80019FB7C0] (DriverSection)>
  0x30 DriverExtension    <_DRIVER_EXTENSION Pointer to [0xFA80019FBA50]␣
↪(DriverExtension)>
  0x38 DriverName         [_UNICODE_STRING DriverName] @ 0xFA80019FB938 (\Driver\rdpbus)
  0x48 HardwareDatabase <_UNICODE_STRING Pointer to [0xF80002B59558]␣
↪(HardwareDatabase)>
  0x50 FastIoDispatch     <_FAST_IO_DISPATCH Pointer to [0x00000000] (FastIoDispatch)>
  0x58 DriverInit         <Function Pointer to [0xF88003B4D1B0] (DriverInit)>
  0x60 DriverStartIo      <Function Pointer to [0x00000000] (DriverStartIo)>
  0x68 DriverUnload       <Function Pointer to [0xF88003B4B480] (DriverUnload)>
  0x70 MajorFunction      <IndexedArray 28 x Pointer @ 0xFA80019FB970>
win7.elf 01:29:01> print x.Object.DriverName
\Driver\rdpbus
```

In the next example we search for SymbolicLinks for the pmem device and discover when the pmem driver was loaded.

```
win7.elf 01:38:53> object_tree type_regex="Symbolic"
0xf8a0003a58a0 SymbolicLink         . Root#MS_PPPOEMINIPORT#0000#{cac88484-7515-4c03-
↪82e6-71a87abac361}-> \Device\00000034 (2012-10-01 21:39:55+0000)
0xf8a0003c1030 SymbolicLink         . Root#*ISATAP#0000#{ad498944-762f-11d0-8dcb-
↪00c04fc3358c}-> \Device\00000001 (2012-10-01 21:39:51+0000)
```

```
0xf8a00007fda0 SymbolicLink         . WMIAdminDevice-> \Device\WMIAdminDevice (2012-
↪10-01 21:39:45+0000)
0xf8a0056e8dd0 SymbolicLink         . pmem-> \Device\pmem (2012-10-01 14:40:44+0000)
0xf8a0001111c0 SymbolicLink         . Root#MS_NDISWANIP#0000#{cac88484-7515-4c03-82e6-
↪71a87abac361}-> \Device\00000032 (2012-10-01 21:39:55+0000)
0xf8a0003bef20 SymbolicLink         . Root#MS_NDISWANBH#0000#{cac88484-7515-4c03-82e6-
↪71a87abac361}-> \Device\00000031 (2012-10-01 21:39:55+0000)
0xf8a000006f40 SymbolicLink         . Global-> \GLOBAL?? (2012-10-01 21:39:45+0000)
```

object_types (Objects)

Displays all object Types on the system.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The windows kernel has the notion of a **Kernel Object**. Objects are managed by the kernel through a dedicated API. Kernel Objects are typically used to manage resources which the kernel manages on behalf of user space, for example, open files are managed via the **_FILE_OBJECT** object.

There is a fixed number of kernel objects, each is described by an **_OBJECT_TYPE** structure, the address of which can be found at the **ObpObjectTypes** symbol.

### Notes

1. Each time a new object is created by the kernel, the **Number of Objects** count increases. For every free's object, this number decreases. The counter therefore represents the total number of active instances of this object type.
2. The number of kernel objects varies between windows kernel version. In order to find the size of the **ObpObjectTypes** array, Rekall uses the reference count on the **Type** object type - each kernel object type has a unique **_OBJECT_TYPE** structure.
3. The **Number of Objects** count also has forensic significance. For example the total number of **Process** objects represents the total number of _EPROCESS structures in current use (Note that a process may be terminated but the _EPROCESS is still kept in use).

### Sample output

The below output indicates that there should be 41 processes active, and 548 threads.

```
win7.elf 01:39:36> object_types
-----------------> object_types()
Index  Number Objects PoolType         Name
----- --------------- ---------------- ----
    2              42 NonPagedPool     Type
    3              40 PagedPool        Directory
    4             173 PagedPool        SymbolicLink
    5             704 PagedPool        Token
    6               3 NonPagedPool     Job
    7              41 NonPagedPool     Process
    8             548 NonPagedPool     Thread
    9               0 NonPagedPool     UserApcReserve
   10               1 NonPagedPool     IoCompletionReserve
...
```

pedump (PEDump)

Dump a PE binary from memory.

Plugin	Type	Description
address_space	AddressSpace	The address space to use.
dtb	IntParser	The DTB physical address.
image_base	SymbolAddress	The address of the image base (dos header).
out_fd	String	A file like object to write the output.
out_file	String	The file name to write.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

Windows executable files (PE Files) are mapped into memory from disk. This plugin can dump arbitrary PE files from memory (whether they are executables, DLLs, kernel modules etc). All we require is the PE file's mapped base addresses (i.e. the location in the virtual address space where the MZ header resides.

The image_base offset can be specified using a named address as usual. So for example, to specify a kernel module it is sufficient to just name it (e.g. pedump "nt" - will dump the kernel image).

This plugin is used by the **dlldump**, **moddump**, **procdump** etc plugins.

### Note

1. In order to dump any PE file from memory we need the PE header to be memory resident. Often this is not the case, and the header is flushed out of virtual memory. In this case it is still possible to dump parts of the PE image using the [vaddump](VADDump.html) plugin.
2. When dumping any binary from memory, it is not usually a perfect binary (i.e. you can not just run it). This is because the Import Address Table (IAT) reflects the patched version in memory and some pages may be missing. The resultant binary is probably only useful to analyses using a tool like IDA pro.

pfn (PFNInfo)

Prints information about an address from the PFN database.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
pfn	IntParser	The PFN to examine.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

psscan (PSScan)

Scan Physical memory for _EPROCESS pool allocations.

Status flags: E: A known _EPROCESS address from pslist. P: A known pid from pslist.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayInt-Parser	Kernel addresses of eprocess structs.
limit	IntParser	The length of data to search in each selected region.
method	ChoiceArray	Method to list processes.
pids	ArrayInt-Parser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
scan_kernel	Boolean	Scan the entire kernel address space.
scan_kernel_code	Boolean	Scan the kernel image and loaded drivers.
scan_kernel_nonpaged_pool	Boolean	Scan the kernel non-paged pool.
scan_kernel_paged_pool	Boolean	Scan the kernel paged pool.
scan_kernel_session_pools	Boolean	Scan session pools for all processes.
scan_physical	Boolean	Scan the physical address space only.
scan_process_memory	Boolean	Scan all of process memory. Uses process selectors to narrow down selections.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

Pool scanning is a technique for discovering kernel data structures based on signatures. It is essentially the memory forensic equivalent of carving. The **psscan** plugin carves for **_EPROCESS** structures in memory.

By default the plugin scans in the physical address space. Any hits are resolved into the virtual address space by following the lists. If **scan_in_kernel** is specified, the scanning occurs in kernel space.

### Notes

1. Like other pool scanning plugins, this plugin may produce false positives since it essentially carves **_EPROCESS** structures out of memory. On the other hand, this plugin may reveal files which have been closed or freed.
2. The plugin displays the physical address of the **_EPROCESS** found. It may be possible to derive their virtual address using the [ptov](PtoV.html) plugin. Alternatively, specify the *scan_in_kernel* option, to ensure scanning occurs in the kernel address space.

pstree (PSTree)

Print process list as a tree

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

This plugin displays all known processes in a tree form (i.e. the process parents with their children). This is useful to see which process launched another process.

### Notes

- **Sometimes malware will launch a processes called "lsass.exe" or** "csrss.exe". This plugin helps to highlight discrepencies since these processes are normally only launched from known processes.
- **Using the verbose=1 flag will also print the command lines of each** process as determined by three methods: - cmd: **task.Peb.ProcessParameters.CommandLine - path: **task.Peb.ProcessParameters.ImagePathName - audit: **task.SeAuditProcessCreationInfo.ImageFileName.Name**

### Sample output

```
win7.elf 14:55:19> pstree verbose=1
Name                                        Pid    PPid    Thds    Hnds  Time
---------------------------------------- ------ ------ ------ ------ ------------------
↪-------
 0xFA8002259060:csrss.exe                   348    340      9    436 2012-10-01␣
↪21:39:57+0000
    cmd: %SystemRoot%\system32\csrss.exe ObjectDirectory=\Windows SharedSection=1024,
↪20480,768 Windows=On SubSystemType=Windows ServerDll=basesrv,1␣
↪ServerDll=winsrv:UserServerDllInitialization,3␣
↪ServerDll=winsrv:ConServerDllInitialization,2 ServerDll=sxssrv,4 ProfileControl=Off␣
↪MaxRequestThreads=16
    path: C:\Windows\system32\csrss.exe
    audit: \Device\HarddiskVolume2\Windows\System32\csrss.exe
 0xFA8000901060:wininit.exe                 384    340      3     75 2012-10-01␣
↪21:39:57+0000
    cmd: wininit.exe
    path: C:\Windows\system32\wininit.exe
    audit: \Device\HarddiskVolume2\Windows\System32\wininit.exe
. 0xFA800206D5F0:services.exe               480    384     11    208 2012-10-01␣
↪21:39:58+0000
     cmd: C:\Windows\system32\services.exe
     path: C:\Windows\system32\services.exe
     audit: \Device\HarddiskVolume2\Windows\System32\services.exe
.. 0xFA80024F85D0:svchost.exe               236    480     19    455 2012-10-01␣
↪14:40:01+0000
      cmd: C:\Windows\system32\svchost.exe -k LocalService
      path: C:\Windows\system32\svchost.exe
      audit: \Device\HarddiskVolume2\Windows\System32\svchost.exe
```

pagefiles (Pagefiles)

Report all the active pagefiles.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

pool_tracker (PoolTracker)

Enumerate pool tag usage statistics.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The Windows kernel allocates memory from a shared pool. In order to track memory leaks and to aid in debugging, pool allocations typically have fixed tags indicating the component which allocated the memory. For example, in windows 8, allocating an _EPROCESS struct will result in a pool allocation with a tag of *Proc*.

To aid in debugging, Windows tracks pool allocation in a special table found by the symbol **PoolTrackTable**. This table can show the total number of allocation and deallocations associated with a particular pool tag.

From a forensic point of view, this information can be useful to assess the number of outstanding allocations. For example we can see how many live processes we expect to be preset.

### Notes

1. Just because the process is terminated does not mean the _EPROCESS structure is immediately deallocated. Windows might keep these structures alive for some time for various reasons. A discrepancy here is at best a hint that something does'nt add up.

### Sample output

```
win8.1.raw 15:29:07> pool_tracker
Tag                NP Alloc   NP Bytes              P Alloc     P Bytes
---- -------------------- ---------- -------------------- ----------
 DMV                  1 (0)          0                 0 (0)          0
8042                  6 (4)       4048                12 (0)          0
ACPI                  4 (0)          0                 0 (0)          0
AFGp                  1 (0)          0                 0 (0)          0
ALPC             3211 (770)     434240                 0 (0)          0
ARFT                  0 (0)          0               151 (3)        192
AcpA                  2 (2)        160                 0 (0)          0
AcpB                  0 (0)          0               121 (0)          0
...
Pprl                  0 (0)          0                 3 (0)          0
Ppsu                  0 (0)          0            1394 (223)      18512
Prcr                  5 (4)       5440                13 (0)          0
Proc                137 (48)     91328                 0 (0)          0
PsFn                136 (0)          0                 0 (0)          0
...

win8.1.raw 15:36:40> pslist
--------------------> pslist()
  Offset (V)    Name                    PID    PPID    Thds      Hnds    Sess   Wow64 Start␣
↪                   Exit
-------------- -------------------- ------ ------ ------ -------- ------ ------ ------
↪------------------ ------------------------
DEBUG:root:Listed 48 processes using PsActiveProcessHead
DEBUG:root:Listed 43 processes using CSRSS
DEBUG:root:Listed 47 processes using PspCidTable
DEBUG:root:Listed 45 processes using Sessions
DEBUG:root:Listed 45 processes using Handles
...
```

In the above example we see that there are 48 outstanding *_EPROCESS* objects and there are 48 members in the **PsActiveProcessHead** list.

pools (Pools)

Prints information about system pools.

Ref: http://illmatics.com/Windows%208%20Heap%20Internals.pdf https://media.blackhat.com/bh-dc-11/Mandt/BlackHat_DC_2011_Mandt_kernelpool-wp.pdf https://immunityinc.com/infiltrate/archives/kernelpool_ infiltrate2011.pdf http://gate.upm.ro/os/LABs/Windows_OS_Internals_Curriculum_Resource_Kit-ACADEMIC/WindowsResearchKernel-WRK/WRK-v1.2/base/ntos/ex/pool.c

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

printkey (PrintKey)

Print a registry key, and its subkeys and values

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

privileges (Privileges)

Prints process privileges.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

procdump (ProcExeDump)

Dump a process to an executable file sample

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
dump_dir	String	Path suitable for dumping files.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
out_fd	String	A file like object to write the output.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

This plugin dumps the mapped PE files associated with a windows process. It is equivalent to calling **pedump** with an image base corresponding to the VAD section of the main process executable.

The **procdump** plugin is a thin wrapper around the **pedump** plugin.

### Sample output

```
win7.elf 14:42:55> procdump proc_regex="csrss", dump_dir="/tmp/"
**************************************************
Dumping csrss.exe, pid: 348    output: executable.csrss_exe_348.exe
**************************************************
Dumping csrss.exe, pid: 396    output: executable.csrss_exe_396.exe
```

procinfo (ProcInfo)

Dump detailed information about a running process.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The **procinfo** plugin displays basic information about a process. It takes all the usual process selectors (e.g. pid, name etc) and prints information about the PE file (using **peinfo**) as well as the process environment strings.

### Sample output

```
win7.elf 14:43:15> procinfo proc_regex="csrss"
**************************************************
Pid: 348 csrss.exe

Process Environment
   ComSpec=C:\Windows\system32\cmd.exe
   FP_NO_HOST_CHECK=NO
   NUMBER_OF_PROCESSORS=1
   OS=Windows_NT
   Path=C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;
↪C:\Windows\System32\WindowsPowerShell\v1.0\
   PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
   PROCESSOR_ARCHITECTURE=AMD64
   PROCESSOR_IDENTIFIER=Intel64 Family 6 Model 37 Stepping 2, GenuineIntel
   PROCESSOR_LEVEL=6
   PROCESSOR_REVISION=2502
   PSModulePath=C:\Windows\system32\WindowsPowerShell\v1.0\Modules\
   SystemDrive=C:
   SystemRoot=C:\Windows
   TEMP=C:\Windows\TEMP
   TMP=C:\Windows\TEMP
   USERNAME=SYSTEM
   windir=C:\Windows

PE Infomation
Attribute            Value
-------------------- -----
Machine              IMAGE_FILE_MACHINE_AMD64
TimeDateStamp        2009-07-13 23:19:49+0000
Characteristics      IMAGE_FILE_EXECUTABLE_IMAGE, IMAGE_FILE_LARGE_ADDRESS_AWARE
GUID/Age             E8979C26A0EE47A69575E54FA6C7F6BE1
PDB                  csrss.pdb
```

```
MajorOperatingSystemVersion 6
MinorOperatingSystemVersion 1
MajorImageVersion    6
MinorImageVersion    1
MajorSubsystemVersion 6
MinorSubsystemVersion 1

Sections (Relative to 0x497B0000):
Perm Name          VMA            Size
---- -------- -------------- --------------
xr-  .text    0x000000001000 0x000000000c00
-rw  .data    0x000000002000 0x000000000200
-r-  .pdata   0x000000003000 0x000000000200
-r-  .rsrc    0x000000004000 0x000000000800
-r-  .reloc   0x000000005000 0x000000000200

Data Directories:
-                                             VMA            Size
---------------------------------------- -------------- --------------
IMAGE_DIRECTORY_ENTRY_EXPORT             0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_IMPORT             0x0000497b17c4 0x00000000003c
IMAGE_DIRECTORY_ENTRY_RESOURCE           0x0000497b4000 0x0000000007f8
IMAGE_DIRECTORY_ENTRY_EXCEPTION          0x0000497b3000 0x00000000003c
IMAGE_DIRECTORY_ENTRY_SECURITY           0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_BASERELOC          0x0000497b5000 0x00000000000c
IMAGE_DIRECTORY_ENTRY_DEBUG              0x0000497b10a0 0x00000000001c
IMAGE_DIRECTORY_ENTRY_COPYRIGHT          0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_GLOBALPTR          0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_TLS                0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG        0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT       0x0000497b02b0 0x000000000030
IMAGE_DIRECTORY_ENTRY_IAT                0x0000497b1000 0x000000000098
IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT       0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR     0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_RESERVED           0x000000000000 0x000000000000

Import Directory (Original):
Name                                               Ord
-------------------------------------------------- -----
ntdll.dll!NtSetInformationProcess                  498
ntdll.dll!RtlSetHeapInformation                    1158
ntdll.dll!RtlSetUnhandledExceptionFilter           1179
ntdll.dll!NtTerminateProcess                       535
ntdll.dll!RtlVirtualUnwind                         1264
ntdll.dll!RtlLookupFunctionEntry                   1025
ntdll.dll!RtlCaptureContext                        635
ntdll.dll!NtTerminateThread                        536
ntdll.dll!RtlUnhandledExceptionFilter              1219
ntdll.dll!RtlSetProcessIsCritical                  1166
ntdll.dll!isspace                                  1900
ntdll.dll!RtlUnicodeStringToAnsiString             1222
ntdll.dll!RtlAllocateHeap                          613
ntdll.dll!RtlFreeAnsiString                        840
ntdll.dll!RtlNormalizeProcessParams                1041
CSRSRV.dll!CsrServerInitialization                 22
CSRSRV.dll!CsrUnhandledExceptionFilter             26

Export Directory:
    Entry      Stat Ord   Name
-------------- ---- ----- --------------------------------------------------
Version Information:
key                  value
-------------------- -----
CompanyName          Microsoft Corporation
FileDescription      Client Server Runtime Process
FileVersion          6.1.7600.16385 (win7_rtm.090713-1255)
InternalName         CSRSS.Exe
LegalCopyright       Microsoft Corporation. All rights reserved.
OriginalFilename     CSRSS.Exe
ProductName          Microsoft Windows Operating System
ProductVersion       6.1.7600.16385
```

ptov (PtoV)

Converts a physical address to a virtual address.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
physical_address	IntParser	The Virtual Address to examine.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

This plugin uses the **PFN Database** to convert a physical page to its virtual address. It is similar to the **pas2vas** plugin in this regard, but does not need to enumerate all address spaces prior to running (so it is a bit faster).

### Notes

1. The plugin currently only works for kernel addresses and for 4k pages. So for example this will not work reliably for pool memory (since Pool is allocated in 2mb pages).
2. If this plugin does not work for a certain address, try to use the **pas2vas** plugin.

### Sample output

```
win7.elf 15:22:57> vtop 0xfa8002635810
------------------> vtop(0xfa8002635810)
Virtual 0xfa8002635810 Page Directory 0x271ec000
pml4e@ 0x271ecfa8 = 0x4000863
pdpte@ 0x4000000 = 0x4001863
pde@ 0x4001098 = 0x2ac009e3
Large page mapped 0x2ae35810
Physical Address 0x2ac35810
win7.elf 15:23:05> ptov 0x2ac35810
------------------> ptov(0x2ac35810)
Physical Address 0x2ac35810 => Virtual Address 0xf6fd40035810
DTB @ 0x187000
PML4E @ 0x187f68
PDPTE @ 0x187fa8
PDE @ 0x4000000
PTE @ 0x40011a8
```

raw2dmp (Raw2Dump)

Convert the physical address space to a crash dump.

The Windows debugger (Windbg) works only with memory dumps stored in the proprietary ‘crashdump’ file format. This file format contains the following features:

1. Physical memory ranges are stored in a sparse way - there is a ‘Runs’ table which specifies the mapping between the physical offset and the file offset of each page. This allows the format to omit unmapped regions (unlike raw format which must pad them with zero to maintain alignment).
2. The crash dump header contains metadata about the image. Specifically, the header contain a copy of the Kernel Debugger Data Block (AKA the KDBG). This data is used to bootstrap the windows debugger by providing critical initial hints to the debugger.

Since the KDBG block is created at system boot and never used (until the crash dump is written) it is trivial for malware to overwrite it - making it really hard for responders since windbg will not be able to read the file. In later versions of windows, the kdbg is also obfuscated (See the function “nt!KdCopyDataBlock” which decrypts it.).

Rekall itself does not use the KDBG block any more, although older memory forensic tools still do use it. Rekall instead relies on accurate debugging symbols to locate critical kernel data structures, reducing the level of trust we place on the image itself (so Rekall is more resilient to manipulation).

In order to ensure that the windows debugger is able to read the produced crash dump, we recreate the kernel debugger block from the symbol information we already have.

NOTE: The crashdump file format can be deduced by:

dis ‘nt!IoFillDumpHeader’

This is the reference for this plugin.

Plugin	Type	Description
destination	String	The destination path to write the crash dump.
dtb	IntParser	The DTB physical address.
rebuild	Boolean	Rebuild the KDBG data block.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The Windows debugger (Windbg) works only with memory dumps stored in the proprietary ‘crashdump’ file format. This file format contains the following features:

1. Physical memory ranges are stored in a sparse way - there is a *Runs* table which specifies the mapping between the physical offset and the file offset of each page. This allows the format to omit unmapped regions (unlike raw format which must pad them with zero to maintain alignment).
2. The crash dump header contains metadata about the image. Specifically, the header contain a copy of the Kernel Debugger Data Block (AKA the **KDBG**). This data is used to bootstrap the windows debugger by providing critical initial hints to the debugger.

Since the **KDBG** block is created at system boot and never used (until the crash dump is written) it is trivial for malware to overwrite it - making it really hard for responders since windbg will not be able to read the file. In later versions of windows, the KDBG is also obfuscated (See the function *nt!KdCopyDataBlock* which decrypts it.).

Rekall itself does not use the **KDBG** block any more, although older memory forensic tools still do use it. Rekall instead relies on accurate debugging symbols to locate critical kernel data structures, reducing the level of trust we place on the image itself (so Rekall is more resilient to manipulation).

In order to ensure that the windows debugger is able to read the produced crash dump, we recreate the kernel debugger block from the symbol information we already have.

### Notes:

1. The crashdump file format can be deduced by: .. code-block:: text

   dis 'nt!IoFillDumpHeader'

   This is the reference for this plugin.

2. This plugin is really only useful in order to produce an image compatible with the windows debugger for the purpose of further investigation by the debugger. If you find that the windows debugger has a useful feature that Rekall does not have, please let us know so we can implement it in Rekall. We intend to replace the use of the windows debugger in digital forensics.

regdump (RegDump)

Dump all registry hives from memory into a dump directory.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
dump_dir	String	Path suitable for dumping files.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

services (Services)

Enumerate all services.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

sessions (Sessions)

List details on _MM_SESSION_SPACE (user logon sessions).

Windows uses sessions in order to separate processes. Sessions are used to separate the address spaces of windows processes.

Note that this plugin traverses the ProcessList member of the session object to list the processes - yet another list _EPROCESS objects are on.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

shimcachemem (ShimCacheMem)

Extract the Application Compatibility Shim Cache from kernel memory.

Plugin	Type	Description
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

show_allocation (ShowAllocation)

Show the allocation containing the address.

Plugin	Type	Description
address	ArrayIntParser	The address to display
dtb	IntParser	The DTB physical address.
length	IntParser	How many bytes after the address to display.
preamble	IntParser	How many bytes prior to the address to display.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

sockets (Sockets)

Print list of open sockets. [Windows xp only]

This module enumerates the active sockets from tcpip.sys

Note that if you are using a hibernated image this might not work because Windows closes all sockets before hibernating.

_ADDRESS_OBJECT are arranged in a hash table found by the _AddrObjTable symbol. The hash table has a size found by the _AddrObjTableSize symbol.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

This module enumerates the active sockets from tcpip.sys

Note that if you are using a hibernated image this might not work because Windows closes all sockets before hibernating.

_ADDRESS_OBJECT are arranged in a hash table found by the _AddrObjTable symbol. The hash table has a size found by the _AddrObjTableSize symbol.

svcscan (SvcScan)

Scan for Windows services

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

Windows uses services for long running processes. Serivces are managed by the “services.exe” process. The **svcscan** plugin scans the heap memory of the “services.exe” process for **_SERVICE_RECORD** records). These records describe the services which are loaded by the system, and even once the services are unloaded, we might find **_SERVICE_RECORD** records.

### Notes

1. Since loading kernel code is usually done by inserting a kernel driver, and kernel drivers are loaded through a service, this plugin will also show forensically significant kernel drivers loading.

2. This plugin relies on memory scanning and so it is not all that reliable. Often it will not reveal services which we know are running. However, it might also reveal services which have been deleted.
3. A better plugin is the **services** plugin which enumerates all services from the registry.

### Sample output

The below example shows a kernel driver being loaded as a service.

```
Offset: 0x26f7d6a10
Order: 402
Process ID: -
Service Name: WFPLWFS
Display Name: Microsoft Windows Filtering Platform
Service Type: SERVICE_KERNEL_DRIVER
Service State: SERVICE_RUNNING
Binary Path: \Driver\WFPLWFS
```

symlinkscan (SymLinkScan)

Scan for symbolic link objects

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
limit	IntParser	The length of data to search in each selected region.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
scan_kernel	Boolean	Scan the entire kernel address space.
scan_kernel_code	Boolean	Scan the kernel image and loaded drivers.
scan_kernel_nonpaged_pool	Boolean	Scan the kernel non-paged pool.
scan_kernel_paged_pool	Boolean	Scan the kernel paged pool.
scan_kernel_session_pools	Boolean	Scan session pools for all processes.
scan_physical	Boolean	Scan the physical address space only.
scan_process_memory	Boolean	Scan all of process memory. Uses process selectors to narrow down selections.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

A symbolic link is a kernel object which maps a device from one name in the kernel object tree to another name. Often a driver will set up a symbolic link to a "dos device name" to allow access to a kernel device from userspace.

For example, the pmem driver makes a symbolic link from **GLOBAL??pmem** to **Devicespmem** so that a user space program can use the **CreateFile** API to open a handle to **.pmem**.

This plugin scans for **_OBJECT_SYMBOLIC_LINK** objects using pool scanning techniques.

### Notes

1. Like other pool scanning plugins, this plugin may produce false positives since it essentially carves **_OBJECT_SYMBOLIC_LINK** structures out of memory. On the other hand, this plugin may reveal symlinks which have been closed or freed.

1. The interesting thing about a symlink is that it contains the timestamp of when it was created. This can be significant when determining when the system was compromised.
2. Since the *symlinkscan* plugin carves out **_OBJECT_SYMBOLIC_LINK** objects it has no context of where in the object tree the symlink exists. Hence it is unable to show parent object directories. A better plugin to use is the [object_tree](ObjectTree.html) plugin.

### Sample output

Here we see the **symlinkscan** plugin detecting the pmem link.

```
    Offset(P)      #Ptr   #Hnd Creation time            From To
- -------------- ------ ------ ------------------------ ---- ------------------------
↪-----------------------------------
  0x00000010d470      3      2 2014-01-24 22:07:29+0000 HDAUDIO#FUNC_01&VEN_8384&DEV_
↪7680&SUBSYS_83847680&REV_1034#4&136d1aa0&0&0001#{65e8773e-8f56-11d0-a3b9-
↪00a0c9223196} \Device\0000001e

  0x00000040e940      1      0 2014-01-24 22:07:23+0000 Psched \Device\Psched
  0x0000004e9490      2      1 2014-01-24 22:07:32+0000 DISPLAY#Default_Monitor#4&
↪d9dcf0b&0&UID0#{e6f07b5f-ee97-4a90-b076-33f57bf4eaa7} \Device\00000021
...
  0x00002be706f0      2      1 2014-01-24 22:07:32+0000 AppContainerNamedObjects␣
↪\Sessions\1\AppContainerNamedObjects
  0x00002bf89f20      2      1 2014-01-24 22:07:32+0000 Global \BaseNamedObjects
  0x00002c0b8270      2      1 2014-01-24 22:07:32+0000 1 \Sessions\1\BaseNamedObjects
  0x00002dbdbe00      1      0 2014-01-24 21:20:05+0000 pmem \Device\pmem
  0x00002f2b7240      1      0 2014-01-24 22:07:26+0000 HCD0 \Device\USBFDO-0
```

thrdscan (ThrdScan)

Scan physical memory for _ETHREAD objects

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayInt-Parser	Kernel addresses of eprocess structs.
limit	IntParser	The length of data to search in each selected region.
method	ChoiceArray	Method to list processes.
pids	ArrayInt-Parser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
scan_kernel	Boolean	Scan the entire kernel address space.
scan_kernel_code	Boolean	Scan the kernel image and loaded drivers.
scan_kernel_nonpaged_pool	Boolean	Scan the kernel non-paged pool.
scan_kernel_paged_pool	Boolean	Scan the kernel paged pool.
scan_kernel_session_pools	Boolean	Scan session pools for all processes.
scan_physical	Boolean	Scan the physical address space only.
scan_process_memory	Boolean	Scan all of process memory. Uses process selectors to narrow down selections.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

Pool scanning is a technique for discovering kernel data structures based on signatures. It is essentially the memory forensic equivalent of carving. The **thrdscan** plugin carves for **_KTHREAD** structures in memory.

By default the plugin scans in the physical address space. Any hits are resolved into the virtual address space by following the lists. If **scan_in_kernel** is specified, the scanning occurs in kernel space.

### Notes

1. Like other pool scanning plugins, this plugin may produce false positives since it essentially carves **_KTHREAD** structures out of memory. On the other hand, this plugin may reveal files which have been closed or freed.
2. The plugin displays the physical address of the **_KTHREAD** found. It may be possible to derive their virtual address using the [ptov](PtoV.html) plugin. Alternatively, specify the *scan_in_kernel* option, to ensure scanning occurs in the kernel address space.
3. This plugin is the pool scanning variant of the [threads](Threads.html) plugin.

### Sample output

The below is an example of running **thrdscan** over a windows system. Note that we can still see exited threads. Rekall resolves the start address of the thread (i.e. the function which started running in this thread). This helps to identify what the thread is supposed to be doing.

```
win8.1.raw 18:52:26> thrdscan
  Offset(P)        PID    TID Start Address  Create Time              Exit Time          ␣
↪         Process          Symbol
-------------- ------ ------ -------------- ------------------------ ------------------
↪-------- ---------------- ------
0x0000001ab080   2332   3976 0x7ff87f35b5c0 -                        -                  ␣
↪         svchost.exe      \Windows\System32\ntdll.dll!TpPostWork+0x4a0
0x000000230880   2392   3212 0x7ff6670fd0bc -                        2014-01-24␣
↪21:18:44+0000 VBoxTray.exe     \Windows\System32\VBoxTray.exe!+0xd0bc
0x00000025e080   3644   1068 0x7ff7a4831070 -                        -                  ␣
↪         conhost.exe      \Windows\System32\conhost.exe!+0x1070
0x000000261080    880   2440 0x7ff866dbaf44 -                        -                  ␣
↪         svchost.exe      \Windows\System32\wuaueng.dll!+0x3af44
0x000000261880    880   3512 0x7ff87f35b5c0 -                        -                  ␣
↪         svchost.exe      \Windows\System32\ntdll.dll!TpPostWork+0x4a0
0x0000002d6080   3644   3688 0x7ff7a4833060 -                        -                  ␣
↪         conhost.exe      \Windows\System32\conhost.exe!+0x3060
0x0000002e1080    976   3932 0x7ff877104924 -                        2014-01-24␣
↪21:18:37+0000 svchost.exe      \Windows\System32\sysmain.dll!+0x94924
0x0000002e1880    880   3324 0x7ff87f35b5c0 -                        -                  ␣
↪         svchost.exe      \Windows\System32\ntdll.dll!TpPostWork+0x4a0
0x00000035d080    880   1752 0x7ff866dbaf44 -                        -                  ␣
↪         svchost.exe      \Windows\System32\wuaueng.dll!+0x3af44
0x000000558080    880   3524 0x7ff87f35b5c0 -                        -                  ␣
↪         svchost.exe      \Windows\System32\ntdll.dll!TpPostWork+0x4a0
0x000000613080    880   3496 0x7ff866dbaf44 -                        -                  ␣
↪         svchost.exe      \Windows\System32\wuaueng.dll!+0x3af44
0x000000613880   3400   3648 0x7ff87f35b5c0 -                        -                  ␣
↪         MpCmdRun.exe     \Windows\System32\ntdll.dll!TpPostWork+0x4a0
0x000000668080    880   3524 0x7ff87f35b5c0 -                        -                  ␣
↪         svchost.exe      \Windows\System32\ntdll.dll!TpPostWork+0x4a0
0x0000006c0080    880   3692 0x7ff8733911b0 -                        -                  ␣
↪         svchost.exe      \Windows\System32\aelupsvc.dll!+0x11b0
0x0000006ce080    880   3180 0x7ff866d81f3c -                        -                  ␣
↪         svchost.exe      \Windows\System32\wuaueng.dll!+0x1f3c
0x000002bd2080    880   3736 0x7ff866dbaf44 -                        -                  ␣
↪         svchost.exe      \Windows\System32\wuaueng.dll!+0x3af44
0x00000370a080    976   3932 0x7ff877104924 -                        2014-01-24␣
↪21:18:37+0000 svchost.exe      \Windows\System32\sysmain.dll!+0x94924
```

```
0x00000370a880    880   3324 0x7ff87f35b5c0 -                        -                  ␣
↪         svchost.exe      \Windows\System32\ntdll.dll!TpPostWork+0x4a0
0x000004eef080    880   3692 0x7ff8733911b0 -                        -                  ␣
↪         svchost.exe      \Windows\System32\aelupsvc.dll!+0x11b0
0x0000051a4874 2124654 30318413 0xffe800000000 -                        -               ␣
↪            ----------------
0x000005d8a080    880   3692 0x7ff8733911b0 -                        -                  ␣
↪         svchost.exe      \Windows\System32\aelupsvc.dll!+0x11b0
0x000009f5d080   2332   3928 0x7ff87f35b5c0 -                        -                  ␣
↪         svchost.exe      \Windows\System32\ntdll.dll!TpPostWork+0x4a0
0x00000cbde080   2392   3880 0x7ff6670fd0bc -                        2014-01-24␣
↪21:18:24+0000 VBoxTray.exe      \Windows\System32\VBoxTray.exe!+0xd0bc
0x00000dbdb080   2392   4084 0x7ff6670fd0bc -                        2014-01-24␣
↪21:19:27+0000 VBoxTray.exe      \Windows\System32\VBoxTray.exe!+0xd0bc
0x00000f345080    880   1532 0x7ff866dbaf44 -                        -                  ␣
↪         svchost.exe      \Windows\System32\wuaueng.dll!+0x3af44
0x00000f345880    880   2932 0x7ff87f35b5c0 -                        -                  ␣
↪         svchost.exe      \Windows\System32\ntdll.dll!TpPostWork+0x4a0
0x00000f413080      4   3176 0xf802d3613418 -                        -                  ␣
↪         System           nt!MiStoreEvictThread
```

threads (Threads)

Enumerate threads.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The **threads** plugin iterates over all processes and lists all threads in all processes. This is the list walking version of the [thrdscan](ThrdScan.html) plugin.

### Sample output

```
   _ETHREAD       PID     TID  Start Address  Process          Symbol
-------------- ------ ------ -------------- ---------------- ------
0xe00000089880      4      8 0xf802d3509ec8 System           nt!Phase1Initialization
0xe0000011f040      4     12 0xf802d3154c04 System           nt!PopIrpWorkerControl
0xe0000011f880      4     16 0xf802d312f868 System           nt!PopIrpWorker
0xe0000011e040      4     20 0xf802d312f868 System           nt!PopIrpWorker
0xe0000011e880      4     24 0xf802d31551c0 System           nt!PopFxEmergencyWorker
0xe0000011d040      4     28 0xf802d3520f14 System           nt!
↪ExpWorkerThreadBalanceManager
0xe0000011d880      4     32 0xf802d30533a8 System           nt!ExpWorkerThread
0xe0000011c880      4     36 0xf802d314cb04 System           nt!
↪ExpWorkerFactoryManagerThread
0xe00000120040      4     40 0xf802d3146fdc System           nt!KiExecuteDpc
0xe00000120880      4     44 0xf802d314f764 System           nt!
↪MiDereferenceSegmentThread
0xe00000124040      4     48 0xf802d3151a8c System           nt!MiModifiedPageWriter
0xe00000124880      4     52 0xf802d314de28 System           nt!KeBalanceSetManager
```

```
0xe00000123040       4      56 0xf802d314bc18 System           nt!KeSwapProcessOrStack
0xe00000122040       4      64 0xf802d314cd68 System           nt!
↪CcQueueLazyWriteScanThread
0xe00000122880       4      68 0xf802d3154b9c System           nt!FsRtlWorkerThread
0xe00000121040       4      72 0xf802d3154b9c System           nt!FsRtlWorkerThread
0xe00000133040       4      76 0xf802d3492540 System           nt!EtwpLogger
0xe00000133880       4      80 0xf802d30533a8 System           nt!ExpWorkerThread
0xe00000137040       4      84 0xf802d314c94c System           nt!MiMappedPageWriter
....
```

timers (Timers)

Print kernel timers and associated module DPCs.

Ref: http://computer.forensikblog.de/en/2011/10/timers-and-times.html

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The windows kernel has a mechanism for drivers to schedule Deferred Procedure Calls (DPCs) wither periodically or in a future time. This mechanism is useful for malware which wants to remain persistant, but not necessarily run at all times (This reduces the malware's footprint).

The kernel uses **_KTIMER** objects to keep track of these DPCs. Depending on the exact OS version, the timers are arranged in slightly different data structures:

- On Window XP there is a symbol **KiTimerTableListHead** which enumerates all timer hash tables.
- On windows 7, the timer list is at **_KPCR.PrcbData.TimerTable.TimerEntries**.

Since Windows 7, PatchGuard was introduced. This uses the timer table to schedule periodic runs. Microsoft felt it was necessary to protect PatchGuard by obfuscating all DPC pointers in the timer table. This unfortunately also obfuscates all other timers, including ones possibly used by malware.

Rekall is able to de-obfuscate these DPC address and resolve them back to their correct module. Rekall will also indicate when the timer is due to go off.

### Sample output

```
win8.1.raw 22:25:53> timers
Table      Offset      DueTime(H)  DueTime              Period(ms)    Signaled    Routine␣
↪    Module
----- -------------- ---------- -------------------- ---------- ---------- -----------
↪--- --------------------
2   0xe00001a58708 0x0000000001f0df8a92 2014-01-24 21:21:14+0000       1000          ␣
↪Yes 0xf80000298480 wdf01000 + 0x8480
8   0xf802d32ecd00 0x0000000001c789ad30 2014-01-24 21:20:05+0000          0             -
↪ 0xf802d311b194 nt!CcScanDpc
9   0xf802d32bcce0 0x0000010c0d9d767529 2015-01-01 00:00:00+0000          0             -
↪ 0xf802d32467b4 nt!ExpNextYearDpcRoutine
9   0xf802d32ac920 0x0000000001e478b3c5 2014-01-24 21:20:53+0000          0             -
↪ 0xf802d3116abc nt!CmpLazyFlushDpcRoutine
13  0xf80002146660 0x0000000001f3302411 2014-01-24 21:21:18+0000      43348          ␣
↪Yes 0xf80002140c44 bowser + 0x3c44
15  0xf8000072e320 0x00000000c877502ee7 2014-01-25 21:02:20+0000          0             -
↪ 0xf80000719230 storport + 0x23230
```

```
17  0xf800024cbb28 0x0000000001fdfb093c 2014-01-24 21:21:36+0000      28348         ␣
↪Yes 0xf800024af550 tunnel + 0x1550
18  0xe0000127ff40 0x0000000002f06baf46 2014-01-28 21:28:23+0000          0            -
↪ 0xf80000b31394 volsnap + 0x2394
21  0xe0000137bb40 0x0000000001f0df8a92 2014-01-24 21:21:14+0000       1000         ␣
↪Yes 0xf8000194a860 usbport + 0x2860
24  0xe00000203b88 0x0000000002534bd8cd 2014-01-24 21:23:59+0000          0            -
↪ 0xf80001a930a4 battc + 0x10a4
38  0xe00001493278 0x0000000001f1249ec9 2014-01-24 21:21:14+0000          0            -
↪ 0xf80000c2ac30 ndis + 0x4c30
38  0xe00002327228 0x00000000024c651b42 2014-01-24 21:23:47+0000     944848            -
↪ 0xf8000249cbb4 mslldp + 0x4bb4
38  0xe000013f7ef8 0x00000000324d602123 2014-01-25 03:07:25+0000   21600000            -
↪ 0xf80001491cf0 dxgkrnl + 0x19cf0
38  0xf802d32ea250 0x0000000001d163bc04 2014-01-24 21:20:21+0000      60000         ␣
↪Yes 0xf802d3116bac nt!IopIrpStackProfilerTimer
40  0xf80000e981c0 0x0000000002840a55a8 2014-01-24 21:25:21+0000          0            -
↪ 0xf80000e94c9c mup + 0x1c9c
```

unloaded_modules (UnloadedModules)

Print a list of recently unloaded modules.

Ref: http://volatility-labs.blogspot.de/2013/05/movp-ii-22-unloaded-windows-kernel_22.html

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

For debugging purposes windows keeps a list of the last few kernel modules to have been unloaded. Sometimes if malware inserts a kernel component, and then removes it this will leave traces in this list.

### Sample output

The below sample shows that *win32dd* was used to acquire this sample, and that the Honeynet project's [capture tools](https://projects.honeynet.org/capture-hpc/browser/capture-hpc/branches/dev/capture-client/KernelDrivers/CaptureKernelDrivers) were used.

```
130115b.w32 22:53:17> unloaded_modules
INFO:root:Detected kernel base at 0x804D7000-
Name                   Start       End       Time
-------------------- ---------- ---------- ----
Sfloppy.SYS          0xf8383000 0xf8386000 2013-01-15 22:06:06+0000
Cdaudio.SYS          0xf89c2000 0xf89c7000 2013-01-15 22:06:06+0000
processr.sys         0xf88aa000 0xf88b3000 2013-01-15 22:06:06+0000
splitter.sys         0xf8bc6000 0xf8bc8000 2013-01-15 22:06:41+0000
aec.sys              0xb1be6000 0xb1c09000 2013-01-15 22:06:41+0000
swmidi.sys           0xb1d06000 0xb1d14000 2013-01-15 22:06:41+0000
DMusic.sys           0xb1cf6000 0xb1d03000 2013-01-15 22:06:41+0000
drmkaud.sys          0xf8c9f000 0xf8ca0000 2013-01-15 22:06:41+0000
kmixer.sys           0xb1b1b000 0xb1b46000 2013-01-15 22:06:51+0000
kmixer.sys           0xb14df000 0xb150a000 2013-01-15 22:08:04+0000
kmixer.sys           0xb14df000 0xb150a000 2013-01-15 22:09:21+0000
win32dd.sys          0xb160a000 0xb1616000 2013-01-15 22:27:39+0000
fastdumpx86.sys      0xf8942000 0xf8948000 2013-01-15 22:30:55+0000
CaptureFileMonitor.sys 0xb1c3a000 0xb1c3d000 2013-01-15 22:35:48+0000
```

```
CaptureRegistryMonitor.sys 0xf8c1e000 0xf8c20000 2013-01-15 22:39:51+0000
CaptureProcessMonitor.sys 0xf8c0e000 0xf8c10000 2013-01-15 22:39:52+0000
CaptureFileMonitor.sys 0xb15ba000 0xb15bd000 2013-01-15 22:39:52+0000
```

userassist (UserAssist)

Print userassist registry keys and information

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

userhandles (UserHandles)

Dump the USER handle tables

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
free	Boolean	Also include free handles.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
type	RegEx	Filter handle type by this Regular Expression.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

users (Users)

Enumerate all users of this system.

Ref: samparse.pl from RegRipper.

# copyright 2012 Quantum Analytics Research, LLC # Author: H. Carvey, keydet89@yahoo.com

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

vad (VAD)

Concise dump of the VAD.

Similar to windbg's !vad.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
offset	IntParser	Only print the vad corresponding to this offset.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
regex	RegEx	A regular expression to filter VAD filenames.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The windows kernel manages process memory using the Virtual Address Descriptor tree. The VAD is a tree of mapped memory regions into the process address space. The VAD regions are used to manage the process address space (i.e. its page tables).

The **vad** plugin displays all the vad regions in the process and their properties.

### Notes

1. The *start* and *end* columns refer to the page number of the region. To convert from an address to page number simply multiply (or divide) by 0x1000.
2. If a memory region is mapped from a file (e.g. via the **mmap** call) the filename will be shown.
3. Most executables (e.g. dlls) are mapped with the EXECUTE_WRITECOPY permission. This is so that the executable pages are shared between all processes. As soon as a process attempts to write to that region the binary will be mapped EXECUTE_READWRITE.
4. When a dll is mapped into the vad, the PE header is placed at the vad's start address. This means that you can dump the dll by simply passing the vad's start address to [pedump](PEDump.html) as the image base.

### Sample output

```
win7_trial_64bit.dmp.E01 23:10:34> vad 1232
**************************************************
Pid: 1232 grrservice.exe
     VAD       lev      start            end        com -        -      Protect           ␣
↪    Filename
-------------- --- -------------- -------------- ---- -------- ------ ------------------
↪--- --------
0xfa80020877a0 1          0x73660        0x736bb    6 Mapped   Exe    EXECUTE_WRITECOPY␣
↪   \Windows\System32\wow64win.dll
0xfa8002083a50 2            0x400          0x427    8 Mapped   Exe    EXECUTE_WRITECOPY␣
↪   \Python27\grrservice.exe
0xfa800207fd80 3            0x290          0x293    0 Mapped          READONLY          ␣
↪   Pagefile-backed section
0xfa800205a6d0 4             0x50           0x8f    7 Private         READWRITE
0xfa80020848f0 5             0x40           0x40    0 Mapped   Exe    EXECUTE_WRITECOPY␣
↪   \Windows\System32\apisetschema.dll
0xfa800208b590 6             0x10           0x1f    0 Mapped          READWRITE         ␣
↪   Pagefile-backed section
0xfa8002066300 5             0x90          0x28f    3 Private         READWRITE
0xfa800208acd0 4            0x2b0          0x316    0 Mapped          READONLY          ␣
↪   \Windows\System32\locale.nls
0xfa8002082470 5            0x2a0          0x2a0    1 Private         READWRITE
0xfa80020aaad0 5            0x360          0x39f    7 Private         READWRITE
0xfa80020a0170 6            0x3a0          0x3df    7 Private         READWRITE
0xfa800207e180 3            0x830          0x92f   28 Private         READWRITE
0xfa800208aa30 4            0x580          0x58f    3 Private         READWRITE
```

```
0xfa800209f6d0 5           0x430          0x4af    1 Private        READWRITE
0xfa80020590f0 5           0x5f0          0x66f    6 Private        READWRITE
0xfa8001fea860 4         0x735d0        0x7361a    4 Mapped   Exe   EXECUTE_WRITECOPY␣
↪    \Windows\SysWOW64\apphelp.dll
0xfa80020a01c0 5           0xb30          0xd2f    3 Private        READWRITE
0xfa800209f680 6           0xd30          0xf2f    3 Private        READWRITE
0xfa8002087f00 5         0x73650        0x73657    2 Mapped   Exe   EXECUTE_WRITECOPY␣
↪    \Windows\System32\wow64cpu.dll
0xfa80020838a0 2         0x7efb0        0x7efd2    0 Mapped         READONLY         ␣
↪    Pagefile-backed section
0xfa8002087c00 3         0x760a0        0x7619f    3 Mapped   Exe   EXECUTE_WRITECOPY␣
↪    \Windows\SysWOW64\kernel32.dll
0xfa800208af80 4         0x74b50        0x74b95    3 Mapped   Exe   EXECUTE_WRITECOPY␣
↪    \Windows\SysWOW64\KernelBase.dll
0xfa8002087cb0 5         0x74a70        0x74a7b    2 Mapped   Exe   EXECUTE_WRITECOPY␣
↪    \Windows\SysWOW64\cryptbase.dll
0xfa8002085e30 6         0x736c0        0x736fe    3 Mapped   Exe   EXECUTE_WRITECOPY␣
↪    \Windows\System32\wow64.dll
0xfa800208a900 6         0x74a80        0x74adf    2 Mapped   Exe   EXECUTE_WRITECOPY␣
↪    \Windows\SysWOW64\sspicli.dll
0xfa800208b900 5         0x76000        0x7609f    5 Mapped   Exe   EXECUTE_WRITECOPY␣
↪    \Windows\SysWOW64\advapi32.dll
0xfa8002086430 4         0x76ce0        0x76dfe    0 Private Exe    EXECUTE_READWRITE
0xfa80020874f0 5         0x767b0        0x7685b    8 Mapped   Exe   EXECUTE_WRITECOPY␣
↪    \Windows\SysWOW64\msvcrt.dll
0xfa800208aaf0 6         0x763b0        0x7649f    2 Mapped   Exe   EXECUTE_WRITECOPY␣
↪    \Windows\SysWOW64\rpcrt4.dll
0xfa800208b1d0 6         0x76860        0x76878    4 Mapped   Exe   EXECUTE_WRITECOPY␣
↪    \Windows\SysWOW64\sechost.dll
0xfa80020839c0 5         0x771b0        0x7735b   12 Mapped   Exe   EXECUTE_WRITECOPY␣
↪    \Windows\System32\ntdll.dll
0xfa8001d47490 6         0x76f50        0x77049    0 Private Exe    EXECUTE_READWRITE
0xfa8002083930 6         0x77390        0x7750f    9 Mapped   Exe   EXECUTE_WRITECOPY␣
↪    \Windows\SysWOW64\ntdll.dll
0xfa800209f5e0 7         0x7efad        0x7efaf    3 Private        READWRITE
0xfa800204f6b0 3         0x7f0e0        0x7ffdf    0 Private        READONLY
0xfa8002084980 4         0x7efde        0x7efde    1 Private        READWRITE
0xfa8002084350 5         0x7efdb        0x7efdd    3 Private        READWRITE
0xfa800209f9b0 6         0x7efd5        0x7efd7    3 Private        READWRITE
0xfa8002083800 5         0x7efdf        0x7efdf    1 Private        READWRITE
0xfa800208b260 6         0x7efe0        0x7f0df    0 Mapped         READONLY         ␣
↪    Pagefile-backed section
0xfa800207c840 4         0x7ffe0        0x7ffef   -1 Private        READONLY
0xfa80020810b0 5         0x7fff0     0x7fffffef   -1 Private        READONLY
```

vaddump (VADDump)

Dumps out the vad sections to a file

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
dump_dir	String	Path suitable for dumping files.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
offset	IntParser	Only print the vad corresponding to this offset.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
regex	RegEx	A regular expression to filter VAD filenames.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

Although you can dump a process executable using the [procdump](ProcDump.html) plugin, this only dumps the main executable. For further analysis of a process it is useful to dump its entire address space. Since the address space is discontiguous it is best to dump it out one vad segment at a time.

### Sample output

```
win7_trial_64bit.dmp.E01 23:45:01> vaddump pid=1232, dump_dir="/tmp"
************* grrservice.exe (1232) *************
    Start            End            Length     Filename                                    ␣
↪                    Comment
-------------- -------------- -------------- ----------------------------------------
↪------------------- -------
0x000073660000 0x0000736bbfff        0x5bfff grrservice.exe.2f684a70.73660000-
↪736bbfff.dmp                 \Windows\System32\wow64win.dll
0x000000400000 0x000000427fff        0x27fff grrservice.exe.2f684a70.00400000-
↪00427fff.dmp                 \Python27\grrservice.exe
0x000000290000 0x000000293fff         0x3fff grrservice.exe.2f684a70.00290000-
↪00293fff.dmp                 Pagefile-backed section
0x000000050000 0x00000008ffff        0x3ffff grrservice.exe.2f684a70.00050000-
↪0008ffff.dmp
0x000000040000 0x000000040fff          0xfff grrservice.exe.2f684a70.00040000-
↪00040fff.dmp                 \Windows\System32\apisetschema.dll
0x000000010000 0x00000001ffff         0xffff grrservice.exe.2f684a70.00010000-
↪0001ffff.dmp                 Pagefile-backed section
0x000000090000 0x00000028ffff       0x1fffff grrservice.exe.2f684a70.00090000-
↪0028ffff.dmp
0x0000002b0000 0x000000316fff        0x66fff grrservice.exe.2f684a70.002b0000-
↪00316fff.dmp                 \Windows\System32\locale.nls
0x0000002a0000 0x0000002a0fff          0xfff grrservice.exe.2f684a70.002a0000-
↪002a0fff.dmp
0x000000360000 0x00000039ffff        0x3ffff grrservice.exe.2f684a70.00360000-
↪0039ffff.dmp
0x0000003a0000 0x0000003dffff        0x3ffff grrservice.exe.2f684a70.003a0000-
↪003dffff.dmp
0x000000830000 0x00000092ffff        0xfffff grrservice.exe.2f684a70.00830000-
↪0092ffff.dmp
0x000000580000 0x00000058ffff         0xffff grrservice.exe.2f684a70.00580000-
↪0058ffff.dmp
0x000000430000 0x0000004affff        0x7ffff grrservice.exe.2f684a70.00430000-
↪004affff.dmp
0x0000005f0000 0x00000066ffff        0x7ffff grrservice.exe.2f684a70.005f0000-
↪0066ffff.dmp
0x0000735d0000 0x00007361afff        0x4afff grrservice.exe.2f684a70.735d0000-
↪7361afff.dmp                 \Windows\SysWOW64\apphelp.dll
0x000000b30000 0x000000d2ffff       0x1fffff grrservice.exe.2f684a70.00b30000-
↪00d2ffff.dmp
```

```
0x000000d30000 0x000000f2ffff       0x1fffff grrservice.exe.2f684a70.00d30000-
↪00f2ffff.dmp
0x000073650000 0x000073657fff         0x7fff grrservice.exe.2f684a70.73650000-
↪73657fff.dmp                  \Windows\System32\wow64cpu.dll
0x00007efb0000 0x00007efd2fff        0x22fff grrservice.exe.2f684a70.7efb0000-
↪7efd2fff.dmp                  Pagefile-backed section
0x0000760a0000 0x00007619ffff        0xfffff grrservice.exe.2f684a70.760a0000-
↪7619ffff.dmp                  \Windows\SysWOW64\kernel32.dll
0x000074b50000 0x000074b95fff        0x45fff grrservice.exe.2f684a70.74b50000-
↪74b95fff.dmp                  \Windows\SysWOW64\KernelBase.dll
0x000074a70000 0x000074a7bfff         0xbfff grrservice.exe.2f684a70.74a70000-
↪74a7bfff.dmp                  \Windows\SysWOW64\cryptbase.dll
0x0000736c0000 0x0000736fefff        0x3efff grrservice.exe.2f684a70.736c0000-
↪736fefff.dmp                  \Windows\System32\wow64.dll
0x000074a80000 0x000074adffff        0x5ffff grrservice.exe.2f684a70.74a80000-
↪74adffff.dmp                  \Windows\SysWOW64\sspicli.dll
0x000076000000 0x00007609ffff        0x9ffff grrservice.exe.2f684a70.76000000-
↪7609ffff.dmp                  \Windows\SysWOW64\advapi32.dll
0x000076ce0000 0x000076dfefff       0x11efff grrservice.exe.2f684a70.76ce0000-
↪76dfefff.dmp
0x0000767b0000 0x00007685bfff        0xabfff grrservice.exe.2f684a70.767b0000-
↪7685bfff.dmp                  \Windows\SysWOW64\msvcrt.dll
0x0000763b0000 0x00007649ffff        0xeffff grrservice.exe.2f684a70.763b0000-
↪7649ffff.dmp                  \Windows\SysWOW64\rpcrt4.dll
0x000076860000 0x000076878fff        0x18fff grrservice.exe.2f684a70.76860000-
↪76878fff.dmp                  \Windows\SysWOW64\sechost.dll
0x0000771b0000 0x00007735bfff       0x1abfff grrservice.exe.2f684a70.771b0000-
↪7735bfff.dmp                  \Windows\System32\ntdll.dll
0x000076f50000 0x000077049fff        0xf9fff grrservice.exe.2f684a70.76f50000-
↪77049fff.dmp
0x000077390000 0x00007750ffff       0x17ffff grrservice.exe.2f684a70.77390000-
↪7750ffff.dmp                  \Windows\SysWOW64\ntdll.dll
0x00007efad000 0x00007efaffff         0x2fff grrservice.exe.2f684a70.7efad000-
↪7efaffff.dmp
0x00007f0e0000 0x00007ffdffff       0xefffff grrservice.exe.2f684a70.7f0e0000-
↪7ffdffff.dmp
0x00007efde000 0x00007efdefff          0xfff grrservice.exe.2f684a70.7efde000-
↪7efdefff.dmp
0x00007efdb000 0x00007efddfff         0x2fff grrservice.exe.2f684a70.7efdb000-
↪7efddfff.dmp
0x00007efd5000 0x00007efd7fff         0x2fff grrservice.exe.2f684a70.7efd5000-
↪7efd7fff.dmp
0x00007efdf000 0x00007efdffff          0xfff grrservice.exe.2f684a70.7efdf000-
↪7efdffff.dmp
0x00007efe0000 0x00007f0dffff        0xfffff grrservice.exe.2f684a70.7efe0000-
↪7f0dffff.dmp                  Pagefile-backed section
0x00007ffe0000 0x00007ffeffff         0xffff grrservice.exe.2f684a70.7ffe0000-
↪7ffeffff.dmp
0x00007fff0000 0x07fffffeffff  0x7ff7fffffff grrservice.exe.2f684a70.7fff0000-
↪7fffffeffff.dmp
...
win7_trial_64bit.dmp.E01 23:45:13> peinfo executable="/tmp/grrservice.exe.2f684a70.
↪760a0000-7619ffff.dmp"
Attribute            Value
-------------------- -----
Machine              IMAGE_FILE_MACHINE_I386
TimeDateStamp        2011-07-16 04:33:08+0000
Characteristics      IMAGE_FILE_32BIT_MACHINE, IMAGE_FILE_DLL,
                     IMAGE_FILE_EXECUTABLE_IMAGE
GUID/Age             0EB73428EC4E430FB8EDD94C5946855B2
PDB                  wkernel32.pdb
MajorOperatingSystemVersion 6
MinorOperatingSystemVersion 1
MajorImageVersion    6
MinorImageVersion    1
MajorSubsystemVersion 6
MinorSubsystemVersion 1

Sections (Relative to 0x760A0000):
Perm Name          VMA            Size
---- -------- -------------- --------------
xr-  .text    0x000000010000 0x0000000c0000
-rw  .data    0x0000000d0000 0x000000010000
-r-  .rsrc    0x0000000e0000 0x000000010000
-r-  .reloc   0x0000000f0000 0x000000010000

Data Directories:
-                                            VMA            Size
---------------------------------------- -------------- --------------
IMAGE_DIRECTORY_ENTRY_EXPORT             0x00007615f728 0x00000000aa1a
IMAGE_DIRECTORY_ENTRY_IMPORT             0x00007616a144 0x0000000001f4
IMAGE_DIRECTORY_ENTRY_RESOURCE           0x000076180000 0x000000000520
IMAGE_DIRECTORY_ENTRY_EXCEPTION          0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_SECURITY           0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_BASERELOC          0x000076190000 0x00000000ad3c
IMAGE_DIRECTORY_ENTRY_DEBUG              0x00007616feb8 0x000000000038
IMAGE_DIRECTORY_ENTRY_COPYRIGHT          0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_GLOBALPTR          0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_TLS                0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG        0x000076123330 0x000000000040
IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT       0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_IAT                0x0000760b0000 0x000000000ddc
IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT       0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR     0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_RESERVED           0x000000000000 0x000000000000

Import Directory (Original):
Name                                               Ord
-------------------------------------------------- -----
API-MS-Win-Core-RtlSupport-L1-1-0.dll!RtlUnwind    3
API-MS-Win-Core-RtlSupport-L1-1-0.dll!RtlCaptureContext 0
API-MS-Win-Core-RtlSupport-L1-1-0.dll!RtlCaptureStackBackTrace 1
ntdll.dll!NtCreateEvent                            227
ntdll.dll!NtDuplicateObject                        275
ntdll.dll!RtlConvertSidToUnicodeString             686
ntdll.dll!NtNotifyChangeKey                        337
ntdll.dll!RtlRunOnceInitialize                     1151
```

vadmap (VADMap)

Inspect each page in the VAD and report its status.

This allows us to see the address translation status of each page in the VAD.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
end	IntParser	Stop reading at this offset.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
start	IntParser	Start reading from this page.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

vtop (VtoP)

Prints information about the virtual to physical translation.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

This plugin displays all the page translation steps needed to resolve a virtual address to a physical address.

### Notes

1. The plugin uses the current default address space to calculate the mapping. If you want to resolve the virtual address in a process space you will need to switch the process context first (i.e. use the [cc](SetProcessContext.html) plugin.

### Sample output

```
win7_trial_64bit.dmp.E01 23:52:53> vtop 0xfa8000a2d060
Virtual 0xfa8000a2d060 Page Directory 0x00187000
pml4e@ 0x187fa8 = 0x3c00863
pdpte@ 0x3c00000 = 0x3c01863
pde@ 0x3c01028 = 0x30c009e3
Large page mapped 0x30e2d060
Physical Address 0x30c2d060
```

win32k_autodetect (Win32kAutodetect)

Automatically detect win32k struct layout.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

dns_cache (WinDNSCache)

Dump the windows DNS resolver cache.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
hashtable	String	Optionally provide the hashtable
no_index	Boolean	Should we not use the index
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

desktops (WinDesktops)

Print information on each desktop.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

dlllist (WinDllList)

Prints a list of dll modules mapped into each process.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

Lists dll modules loaded into a process by following the doubly linked list of **LDR_DATA_TABLE_ENTRY** stored in in **_EPROCESS.Peb.Ldr.InLoadOrderModuleList**. DLLs are automatically added to this list when a process calls *LoadLibrary* (or some derivative such as *LdrLoadDll*) and they aren't removed until *FreeLibrary* is called and the reference count reaches zero.

All the usual process selectors are supported.

### Note

1. Wow64 processes (i.e. 32 bit binaries running on 64 bit windows) load dlls through a different mechanism.
2. Since the **InLoadOrderModuleList** is maintained in the process address space, it is simple to manipulate from Ring 3 (without kernel access). This means that this plugin may not show all the linked in DLLs.
3. A better plugin to use is the [ldrmodules](LdrModules.html) plugin, which uses the VAD to enumerate dlls. The VAD is maintained in kernel memory and therefore can only be accessed through Ring 0 access.

### Sample output

Below we see winpmem used to acquire the image of this Windows 8.1 system. Since winpmem is a 32 bit application, we see the wow64.dll dynamically loaded. Note that in this case, the 32 bit dlls will not show in the **InLoadOrderModuleList**. Using the [ldrmodules](LdrModules.html) plugin reveals all the 32 bit dlls loaded.

```
win8.1.raw 15:35:10> dlllist proc_regex="winpmem"
-------------------> dlllist(proc_regex="winpmem")
winpmem_1.5.2. pid: 2628
Command line : winpmem_1.5.2.exe  -2 win8.1.raw
Note: use ldrmodules for listing DLLs in Wow64 processes
```

```
     Base           Size      Load Reason/Count              Path
-------------- -------------- ------------------------------ ----
0x000000020000        0x2d000 LoadReasonStaticDependency     C:\temp\winpmem_1.5.2.exe
0x7ff87f320000       0x1a9000 LoadReasonStaticDependency    ␣
↪C:\Windows\SYSTEM32\ntdll.dll
0x000076f50000        0x49000 LoadReasonDynamicLoad         ␣
↪C:\Windows\SYSTEM32\wow64.dll
0x000076fa0000        0x68000 LoadReasonStaticDependency    ␣
↪C:\Windows\system32\wow64win.dll
0x000077010000         0x9000 LoadReasonStaticDependency    ␣
↪C:\Windows\system32\wow64cpu.dll
win8.1.raw 15:35:51> ldrmodules proc_regex="winpmem"
--------------------> ldrmodules(proc_regex="winpmem")
Pid      Process                   Base      InLoad InInit InMem MappedPath
-------- -------------------- -------------- ------ ------ ----- ----------
2628     winpmem_1.5.2.       0x0000753b0000 False False False␣
↪\Windows\SysWOW64\KernelBase.dll
2628     winpmem_1.5.2.       0x000000020000 True   False True  \temp\winpmem_1.5.2.exe
2628     winpmem_1.5.2.       0x000076c30000 False False False␣
↪\Windows\SysWOW64\kernel32.dll
2628     winpmem_1.5.2.       0x000074a40000 False False False␣
↪\Windows\SysWOW64\cryptbase.dll
2628     winpmem_1.5.2.       0x000074a50000 False False False␣
↪\Windows\SysWOW64\sspicli.dll
2628     winpmem_1.5.2.       0x000077010000 True   True   True ␣
↪\Windows\System32\wow64cpu.dll
2628     winpmem_1.5.2.       0x000076f50000 True   True   True ␣
↪\Windows\System32\wow64.dll
2628     winpmem_1.5.2.       0x000076fa0000 True   True   True ␣
↪\Windows\System32\wow64win.dll
2628     winpmem_1.5.2.       0x000075250000 False False False␣
↪\Windows\SysWOW64\rpcrt4.dll
2628     winpmem_1.5.2.       0x0ff87f320000 False False False␣
↪\Windows\System32\ntdll.dll
2628     winpmem_1.5.2.       0x000077020000 False False False␣
↪\Windows\SysWOW64\ntdll.dll
2628     winpmem_1.5.2.       0x0000749e0000 False False False␣
↪\Windows\SysWOW64\bcryptprimitives.dll
2628     winpmem_1.5.2.       0x000074ff0000 False False False␣
↪\Windows\SysWOW64\advapi32.dll
2628     winpmem_1.5.2.       0x000076f10000 False False False␣
↪\Windows\SysWOW64\sechost.dll
2628     winpmem_1.5.2.       0x000074d80000 False False False␣
↪\Windows\SysWOW64\msvcrt.dll
```

eventhooks (WinEventHooks)

Print details on windows event hooks

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

find_dtb (WinFindDTB)

A plugin to search for the Directory Table Base for windows systems.

There are a number of ways to find the DTB:

- Scanner method: Scans the image for a known kernel process, and read the DTB from its Process Environment Block (PEB).
- Get the DTB from the KPCR structure.
- Note that the kernel is mapped into every process's address space (with the exception of session space which might be different) so using any process's DTB from the same session will work to read kernel data structures. If this plugin fails, try psscan to find potential DTBs.

Plugin	Type	Description
process_name	String	The names of the processes to search for.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

### Notes

1. This is an internally used plugin for discovering the Directory Table Base (DTB) on windows systems. It is unlikely to be useful to a user by itself.

memdump (WinMemDump)

Dump windows processes.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
dump_dir	String	Path suitable for dumping files.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

To dump all addressable memory in a process, use the memdump plugin. This plugin enumerates the process page tables and writes them out into an external file. An index file is also created which can be used to find the virtual address of each byte in the output file.

You would typically use this output file in order to scan for e.g. virus signatures or other patterns in tools which do not understand virtual memory mappings.

The plugin accepts all the usual process filtering commands (e.g. by pid, proc_regex etc). Additionally if no filtering command is specified the plugin dumps the kernel's address space. (You can dump all processes by providing a **proc_regex** of '.').

### Notes

1. This plugin is very similar to the vaddump plugin, except that it dumps the page table, and not only the VAD tree. This plugin actually contains all memory currently accessible to the process (despite any possible manipulation of the VAD tree).
2. The process's virtual address space is typically fragmented and had large, unmapped gaps in it. Therefore this plugin does not just zero fill these gaps, rather it writes all addressable memory directly to the output file. This means that contiguous memory in the output file is not necessarily contiguous in memory.
3. To find out where a particular byte in the output file maps in the process virtual memory, check the index file (Example below).
4. Note that processes typically alway map the kernel in the upper memory region (i.e. above the symbol *MmHighestUserAddress*. This plugin does not dump the kernel portion of the address space, unless the **–all** parameter is specified.

### Sample output

```
win7.elf 00:30:52> memdump pid=2912, dump_dir="/tmp/"
-----------------> memdump(pid=2912, dump_dir="/tmp/")
**************************************************
Writing vol.exe 0xfa8002193060 to vol.exe_2912.dmp
win7.elf 00:30:55> ls -l /tmp/vol.exe_2912.dmp -h
-rw-r----- 1 scudette staff 2.2M Jun 18 00:30 /tmp/vol.exe_2912.dmp
win7.elf 00:30:59> less /tmp/vol.exe_2912.dmp.idx
 File Address      Length      Virtual Addr
-------------- -------------- --------------
0x000000000000 0x000000001000 0x000000010000
0x000000001000 0x000000001000 0x000000020000
0x000000002000 0x000000001000 0x000000021000
0x000000003000 0x000000001000 0x00000002f000
0x000000004000 0x000000001000 0x000000040000
0x000000005000 0x000000001000 0x000000050000
0x000000006000 0x000000001000 0x000000051000
```

memmap (WinMemMap)

Calculates the memory regions mapped by a process.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

To enumerate the address space of a process use this plugin.

It is not that useful in practice, unless you want to manually translate a physical address to a virtual address.

### Notes

1. It is not often necessary to dump the entire page tables of each process. Instead it is possible to first switch to the process context (using the **cc** plugin), and then use *vtop* to translate the virtual address to physical address.
2. Similar to the **memdump** plugin, we do not dump the kernel address space portion for processes unless the **all** parameter is specified.

### Sample output

```
win7.elf 00:54:22> memmap pid=2912
-----------------> memmap(pid=2912)
**************************************************
Process: 'vol.exe' pid: 2912

Dumping address space at DTB 0x271ec000

   Virtual        Physical          Size
-------------- -------------- --------------
0x000000010000 0x000007c4c000         0x1000
0x000000020000 0x00000818f000         0x1000
0x000000021000 0x000007e11000         0x1000
0x00000002f000 0x000008010000         0x1000
0x000000040000 0x00002428e000         0x1000
0x000000050000 0x000001e6b000         0x1000
0x000000051000 0x000007f49000         0x1000
```

messagehooks (WinMessageHooks)

List desktop and thread window message hooks.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

moddump (WinModDump)

Dump kernel drivers from kernel space.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
dump_dir	String	Path suitable for dumping files.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
out_fd	String	A file like object to write the output.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
regex	RegEx	A Regular expression for selecting the dlls to dump.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

netscan (WinNetscan)

Scan a Vista, 2008 or Windows 7 image for connections and sockets

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayInt-Parser	Kernel addresses of eprocess structs.
limit	IntParser	The length of data to search in each selected region.
method	ChoiceArray	Method to list processes.
pids	ArrayInt-Parser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
scan_kernel	Boolean	Scan the entire kernel address space.
scan_kernel_code	Boolean	Scan the kernel image and loaded drivers.
scan_kernel_nonpaged_pool	Boolean	Scan the kernel non-paged pool.
scan_kernel_paged_pool	Boolean	Scan the kernel paged pool.
scan_kernel_session_pools	Boolean	Scan session pools for all processes.
scan_physical	Boolean	Scan the physical address space only.
scan_process_memory	Boolean	Scan all of process memory. Uses process selectors to narrow down selections.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

netstat (WinNetstat)

Enumerate image for connections and sockets

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

pas2vas (WinPas2Vas)

Resolves a physical address to a virtual addrress in a process.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
offsets	ArrayIntParser	A list of physical offsets to resolve.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

In virtual (or protected) mode, the CPU can not access physical memory directly. Instead each memory access made by the CPU is translated using the MMU into a relevant physical address. This translation is controlled by page tables loaded in the memory address controlled by the CR3 register.

Each processes has a unique page table structure, and therefore a unique view of physical memory. In order to know what physical address is mapped to each virtual address you can use the **vtop** plugin. However, the reverse mapping is not so simple - there can be many virtual addresses mapped to the same physical address.

This plugin enumerates all virtual to physical mappings in one or more processes. It then builds a large lookup table in memory to be able to reverse the mapping. i.e. given a physical address, the plugin is able to determine the virtual address that maps to it, and in which processes it exists.

Forensically this can be used if you find an interesting string in the physical image (e.g. with a hex editor) and want to know which process has that physical memory mapped. Another use case is to detect shared memory between multiple processes.

### Notes

1. This plugin only enumerates the userspace portion of the process address space (since all processes share the same kernel address space).
2. The plugin may take a while to run while it builds its lookup table. The next time you run it it should be very fast. The lookup map is also stored in the session cache so you can use the **-s** parameter to store the session for next time.

### Sample output

In the following we see that the process *vol.exe* is a Wow64 process and maps **WindowsSysWOW64ws2_32.dll**. We want to know who else is using this dll. We first find the physical address of the mapped dll (note we need to switch to the correct process context first), then we use the **pas2vas** plugin to determine which other process has that physical page mapped.

```
win7.elf 12:29:35> pslist
  Offset (V)    Name                    PID    PPID   Thds     Hnds    Sess   Wow64  Start␣
↪                    Exit
-------------- -------------------- ------ ------ ------ -------- ------ ------ ------
↪------------------ ------------------------
...
0xfa8002193060 vol.exe                 2912   2644      1       19      1   True  2012-
↪10-01 14:41:03+0000 -
0xfa80017f9060 vol.exe                 2920   2912      4      169      1   True  2012-
↪10-01 14:41:03+0000 -
win7.elf 12:29:59> vad 2912
-----------------> vad(2912)
**************************************************
Pid: 2912 vol.exe
     VAD        lev     start           end         com -       -      Protect           ␣
↪    Filename
-------------- --- -------------- -------------- ---- ------- ------ -----------------
↪--- --------
0xfa80026f9d80 1          0x74400        0x7443e     3 Mapped  Exe    EXECUTE_WRITECOPY␣
↪    \Windows\System32\wow64.dll
...
0xfa80021da200 3          0x766c0        0x766f4     2 Mapped  Exe    EXECUTE_WRITECOPY␣
↪    \Windows\SysWOW64\ws2_32.dll
0xfa80026eb5e0 4          0x75ef0        0x75fdf     2 Mapped  Exe    EXECUTE_WRITECOPY␣
↪    \Windows\SysWOW64\rpcrt4.dll
...
0xfa80028f59d0 5          0x7fff0     0x7fffffef    -1 Private        READONLY
win7.elf 12:30:08> cc 2912
Switching to process context: vol.exe (Pid 2912@0xfa8002193060)

win7.elf 12:32:45> vtop 0x766c0000
-----------------> vtop(0x766c0000)
Virtual 0x766c0000 Page Directory 0x271ec000
pml4e@ 0x271ec000 = 0x700000008844867
pdpte@ 0x8844008 = 0x80000007845867
pde@ 0x7845d98 = 0x7b55847
```

```
pte@ 0x7b55600 = 0x1a58f005
PTE mapped@ 0x7b55600 = 0x1a58f000
Physical Address 0x1a58f000
win7.elf 12:32:53> pas2vas 0x1a58f000

   Physical       Virtual        Pid Name
-------------- -------------- ------ ----
0x00001a58f000 0x0000766c0000   2616 Console.exe
0x00001a58f000 0x0000766c0000   2920 vol.exe
0x00001a58f000 0x0000766c0000   2912 vol.exe
```

We see that *Console.exe* also maps the same dll - probably since it is also a Wow64 process which requires network access. .. _phys_map-WinPhysicalMap-plugin:

phys_map (WinPhysicalMap)

Prints the boot physical memory map.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

This plugin will simply print the kernels idea of the physical memory layout on a machine. Typically the physical address space is not contiguous (i.e. does not have RAM chip mapping all address ranges between 0 and the maximum number). This is because the BIOS needs to leave gaps for DMA devices to be mapped.

The BIOS sets up an initial mapping and communicates the mapping to the kernel through a BIOS service call (Or EFI call) which can be done while the kernel still boots (In real mode). The kernel then keeps this information and returns it through the **MmGetPhysicalMemoryRanges()** function.

### Notes

1. It is rather easy to manipulate this information to subvert acquisition. Most acquisition tools use this information to determine where it is safe to read and to avoid reading from DMA mapped memory.

### Sample output

```
win8.1.raw 15:19:26> phys_map
-------------------> phys_map()
  Phys Start      Phys End    Number of Pages
-------------- -------------- ---------------
0x000000001000 0x00000009f000 158
0x000000100000 0x000000102000 2
0x000000103000 0x00003fff0000 261869
```

yarascan_physical (WinPhysicalYaraScanner)

An experimental yara scanner over the physical address space.

Yara does not provide a streaming interface, which means that when we scan for yara rules we can only ever match strings within the same buffer. This is a problem for physical address space scanning because each page (although it might appear to be contiguous) usually comes from a different process/mapped file.

Therefore we need a more intelligent way to apply yara signatures on the physical address space:

1. The original set of yara rules is converted into a single rule with all the strings from all the rules in it. The rule has a condition "any of them" which will match any string appearing in the scanned buffer.

2. This rule is then applied over the physical address space.

3. For each hit we derive a context and add the hit to the context.

4. Finally we test all the rules within the same context with the original rule set.

Plugin	Type	Description
context	IntParser	Context to print after the hit.
hits	IntParser	Quit after finding this many hits.
limit	IntParser	The length of data to search.
pre_context	IntParser	Context to print before the hit.
start	IntParser	Start searching from this offset.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.
yara_ast	String	If provided we scan for this yara expression specified in the yara JSON AST.
yara_expression	String	If provided we scan for this yara expression specified in the yara DSL.

pslist (WinPsList)

List processes for windows.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The **pslist** plugin list all the processes on windows using a variety of methods. Since it is required by all plugins which has process selectors, this plugin will, by default, list processes using all methods.

The output of this plugin is typically cached in the session, so the first time it is run there might be a slight delay while all methods are used, but subsequent invokations should be almost instant.

Currently the following process listing methods are used:

- PsActiveProcessHead: This method follows the doubly linked list found by the symbol **PsActiveProcessHead**. It is the simplest and fastest method for listing processes, but it is easily subverted by simply removing an _EPROCESS struct from this list.
- CSRSS: The client-server runtime service is responsible for monitoring all running processes. It therefore maintains open handles to running processes. This method locates the *csrss.exe* process and enumerates its handle table finding all handles to processes. Note that this will not typically find the csrss.exe proces itself, nor system processes which were started before it.
- PspCidTable: The PspCidTable is a handle table for process and thread client IDs [Ref](http://uninformed.org/index.cgi?v=3&a=7&p=6). The process's pid is the index into this table. This method enumerates the table in order to find all processes. (Note a rootkit can easily remove a process from this table).
- Sessions: This enumerates all the processes in all windows sessions (**SessionProcessLinks** member of **_MM_SESSION_SPACE** struct).
- Handles: The enumerates all handle tables (Which are found on a list from the symbol **HandleTableListHead**) and collects their owning process (The **QuotaProcess** member).

### Sample output

```
  Offset (V)    Name                          PID    PPID    Thds      Hnds    Sess  Wow64 Start␣
↪                   Exit
-------------- -------------------- ------ ------ ------ -------- ------ ------ ------
↪------------------ ------------------------
DEBUG:root:Listed 48 processes using PsActiveProcessHead
DEBUG:root:Listed 43 processes using CSRSS
DEBUG:root:Listed 47 processes using PspCidTable
DEBUG:root:Listed 45 processes using Sessions
DEBUG:root:Listed 45 processes using Handles
0xe00000074580 System                       4      0      97 -------- ------  False 2014-
↪01-24 22:07:24+0000 -
0xe00001499040 smss.exe                   292      4       2 -------- ------  False 2014-
↪01-24 22:07:24+0000 -
0xe0000212c900 svchost.exe                372    528      15 --------      0  False 2014-
↪01-24 21:07:51+0000 -
0xe00001be1280 csrss.exe                  380    372       8 --------      0  False 2014-
↪01-24 22:07:32+0000 -
0xe000000ce080 wininit.exe                432    372       1 --------      0  False 2014-
↪01-24 22:07:32+0000 -
0xe000000d9280 csrss.exe                  440    424       9 --------      1  False 2014-
↪01-24 22:07:32+0000 -
```

rammap (WinRammap)

Scan all physical memory and report page owners.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
end	IntParser	Physical memory address to end displaying.
start	IntParser	Physical memory address to start displaying.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

ssdt (WinSSDT)

Enumerate the SSDT.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The System Service Descritor Table is the main interface to the kernel from user space. In the past, malware used to install hook in this SSDT in order to intercept userspace->kernel calls. In more recent versions of Windows, Microsoft has implemented **PatchGuard** specifically to prevent these kinds of hooks. Therefore, its very rare to see these kinds of hooks any more.

The **ssdt** plugin enumerates the the SSDT table and resolves the addresses back to the names of the functions. Windows has two SSDTs - one for the kernel and one for the GUI subsystem (win32k driver).

An intalled ssdt hook will appear as a function in a different module (or an unknown module).

### Sample output

```
win7.elf 15:35:25> ssdt
************ Table 0 @ 0xf80002691b00 ************
    Entry          Target     Symbol
-------------- -------------- ------
           0x0 0xf80002aa2190 nt!NtMapUserPhysicalPagesScatter
           0x1 0xf80002988a00 nt!NtWaitForSingleObject
           0x2 0xf80002688dd0 nt!NtCallbackReturn
           0x3 0xf800029abb10 nt!NtReadFile
           0x4 0xf800029a9bb0 nt!NtDeviceIoControlFile
           0x5 0xf800029a4ee0 nt!NtWriteFile
           0x6 0xf8000294adc0 nt!NtRemoveIoCompletion
           0x7 0xf80002947f10 nt!NtReleaseSemaphore
           0x8 0xf8000299fda0 nt!NtReplyWaitReceivePort
           0x9 0xf80002a71e20 nt!NtReplyPort
...
         0x18c 0xf8000297a92c nt!NtWaitForKeyedEvent
         0x18d 0xf800026a1010 nt!NtWaitForWorkViaWorkerFactory
         0x18e 0xf80002ab0b00 nt!NtWaitHighEventPair
         0x18f 0xf80002ab0b90 nt!NtWaitLowEventPair
         0x190 0xf80002678fc4 nt!NtWorkerFactoryWorkerReady
************ Table 1 @ 0xf960001a1c00 ************
    Entry          Target     Symbol
-------------- -------------- ------
           0x0 0xf96000195580 win32k!NtUserGetThreadState
           0x1 0xf96000192630 win32k!NtUserPeekMessage
           0x2 0xf960001a3c6c win32k!NtUserCallOneParam
           0x3 0xf960001b1dd0 win32k!NtUserGetKeyState
           0x4 0xf960001ab1ac win32k!NtUserInvalidateRect
           0x5 0xf960001a3e70 win32k!NtUserCallNoParam
           0x6 0xf9600019b5a0 win32k!NtUserGetMessage
           0x7 0xf9600017fbec win32k!NtUserMessageCall
...
         0x334 0xf96000153b80 win32k!NtUserValidateHandleSecure
         0x335 0xf960001acd9c win32k!NtUserWaitForInputIdle
         0x336 0xf960001a6304 win32k!NtUserWaitForMsgAndEvent
         0x337 0xf960001acef0 win32k!NtUserWindowFromPhysicalPoint
         0x338 0xf960001ae06c win32k!NtUserYieldTask
         0x339 0xf960001a6b84 win32k!NtUserSetClassLongPtr
         0x33a 0xf96000181ca0 win32k!NtUserSetWindowLongPtr
```

sigscan (WinSigScan)

Runs a signature scans against physical, kernel or process memory.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

virt_map (WinVirtualMap)

Prints the Windows Kernel Virtual Address Map.

Windows allocates virtual address ranges to various purposes. This plugin deduces the virtual address map.

On 32 bit windows, the kernel virtual address space can be managed dynamically. This plugin shows each region and what it is used for.

Note that on 64 bit windows the address space is large enough to not worry about it. In that case, the offsets and regions are hard coded.

http://www.woodmann.com/forum/entry.php?219-Using-nt!_MiSystemVaType-to-navigate-dynamic-kernel-address-space-in-Window

The kernel debugger shows the virtual address map using the !vm extension. For example:

> !vm 20 System Region Base Address NumberOfBytes

NonPagedPool : ffff810000000000 100000000000 Session : ffff910000000000 8000000000 SpecialPoolPaged : ffff978000000000 8000000000 SystemCache : ffff988000000000 100000000000 SystemPtes : ffffae8000000000 100000000000 UltraZero : ffffc00000000000 100000000000 PageTables : ffffd40000000000 8000000000 PagedPool : ffffd48000000000 100000000000 SpecialPoolNonPaged : ffffe50000000000 8000000000 PfnDatabase : ffffe80000000000 38000000000 Cfg : ffffebdd84214da8 28000000000 HyperSpace : ffffee8000000000 10000000000 SystemImages : fffff80000000000 8000000000

Rekall uses this information to refine its operations to increase both efficiency and correctness. For example, when scanning objects which should exist in non paged pools, by default, Rekall only examines the NonPagedPool region. This speeds up operations as well as reducing false positives from unrelated memory regions.

Later kernel version (Windows 10+) use a global nt!MiVisibleState to maintain state information, including the virtual address map. This plugin implements support for various versions.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

yarascan (WinYaraScan)

Scan using yara signatures.

Plugin	Type	Description
binary_string	String	A binary string (encoded as hex) to search for. e.g. 000102[1-200]0506
context	IntParser	Context to print after the hit.
dtb	IntParser	The DTB physical address.
eprocess	ArrayInt-Parser	Kernel addresses of eprocess structs.
hits	IntParser	Quit after finding this many hits.
limit	IntParser	The length of data to search in each selected region.
method	ChoiceArray	Method to list processes.
pids	ArrayInt-Parser	One or more pids of processes to select.
pre_context	IntParser	Context to print before the hit.
proc_regex	RegEx	A regex to select a process by name.
scan_kernel	Boolean	Scan the entire kernel address space.
scan_kernel_code	Boolean	Scan the kernel image and loaded drivers.
scan_kernel_nonpaged_pool	Boolean	Scan the kernel non-paged pool.
scan_kernel_paged_pool	Boolean	Scan the kernel paged pool.
scan_kernel_session_pools	Boolean	Scan session pools for all processes.
scan_physical	Boolean	Scan the physical address space only.
scan_process_memory	Boolean	Scan all of process memory. Uses process selectors to narrow down selections.
string	String	A verbatim string to search for.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.
yara_expression	String	If provided we scan for this yara expression.
yara_file	String	The yara signature file to read.

address_resolver (WindowsAddressResolver)

A windows specific address resolver plugin.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
symbol	ArrayString	List of symbols to lookup
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

psxview (WindowsPsxView)

Find hidden processes with various process listings

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

cc (WindowsSetProcessContext)

A cc plugin for windows.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
eprocess	ArrayIntParser	Kernel addresses of eprocess structs.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

windows_stations (WindowsStations)

Displays all the windows stations by following lists.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

times (WindowsTimes)

Return current time, as known to the kernel.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

2.1.2 Linux

arp (Arp)

print the ARP table.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

arp returns the list of IPv4 network neighbour entries in the kernel cache.

Rekall uses the *neigh_tables* kernel symbol and walks the neighbour tables to show the entries.

### Sample output

```
Windows7_VMware(Win7x64+Ubuntu686,Ubuntu64)_VBox(XPSP3x86).ram 12:09:00> arp
-------------------------------------------------------------------------> arp()
                                    IP Address                   MAC           Device
--------------------------------------------- -------------------- ---------------
                              ff02::1:ff57:f719    33:33:ff:57:f7:19            eth0
                                       ff02::16    33:33:00:00:00:16            eth0
```

```
                                  192.168.239.2     00:50:56:e5:38:b6            eth0
                                192.168.239.254     00:50:56:f7:25:d0            eth0
```

banner (Banner)

Prints the Linux banner information.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

banner output provides the same information as running *uname -a* on the host.

### Sample output

```
Windows7_VMware(Win7x64+Ubuntu686,Ubuntu64)_VBox(XPSP3x86).ram 12:17:38> banner
---------------------------------------------------------------------> banner()
Banner
--------------------------------------------------------------------------------
Linux version 3.11.0-12-generic (buildd@allspice) (gcc version 4.8.1 (Ubuntu/Linaro 4.
↪8.1-10ubuntu7) ) #19-Ubuntu SMP Wed Oct 9 16:20:46 UTC 2013 (Ubuntu 3.11.0-12.19-
↪generic 3.11.3)
```

bash (BashHistory)

Scan the bash process for history.

Based on original algorithm by Andrew Case.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
method	ChoiceArray	Method to list processes (Default uses all methods).
pids	ArrayInt-Parser	One or more pids of processes to select.
proc_regex	RegEx	The processes we should examine.
scan_entire_address_space	Boolean	Scan the entire process address space, not only the heap.
task	ArrayInt-Parser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The Bourne Again Shell maintains a history a history of all commands that have been executed in the current session in memory. *bash* is a plugin that provides a chronologically ordered list of commands executed by each bash process, grouped by pid.

### Notes

- Only commands executed in each bash session are stored in memory. So if you're looking for commands for exitted bash sessions you may be more lucky by looking at the disk .bash_history file if logging wasn't disabled.

### Sample output

```
Windows7_VMware(Win7x64+Ubuntu686,Ubuntu64)_VBox(XPSP3x86).ram 12:27:35> bash
----------------------------------------------------------------------> bash()
   Pid Name                 Timestamp                Command
------ -------------------- ------------------------ --------------------
  1335 bash                 2014-03-04 17:16:31+0000 uname -a
```

check_afinfo (CheckAFInfo)

Verifies the operation function pointers of network protocols.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The plugin identifies the location of each function pointer of different network protocols. If located within the kernel or a loaded module, rekall will give such information as well as its kernel-space address.

If malware dynamically allocates memory and copies code there to handle these functions, the Module column will appear as Unknown.

### Sample output .. code-block:: text

Windows7_VMware(Win7x64+Ubuntu686,Ubuntu64)_VBox(XPSP3x86).ram 13:51:35> check_afinfo ——————————————————————> check_afinfo() Constant Name Member Address Module ————————— ————————— ————— ——————
tcp4_seq_afinfo seq_fops.llseek 0xffff811c9250 Kernel tcp4_seq_afinfo seq_fops.read 0xffff811c9460 Kernel tcp4_seq_afinfo seq_fops.release 0xffff812157d0 Kernel udplite6_seq_afinfo seq_ops.show 0xffff816a1300 Kernel udplite6_seq_afinfo seq_fops.llseek 0xffff811c9250 Kernel udplite6_seq_afinfo seq_fops.read 0xffff811c9460 Kernel udplite6_seq_afinfo seq_fops.release 0xffff812157d0 Kernel udp6_seq_afinfo seq_ops.show 0xffff816a1300 Kernel udp6_seq_afinfo seq_fops.llseek 0xffff811c9250 Kernel udp6_seq_afinfo seq_fops.read 0xffff811c9460 Kernel udp6_seq_afinfo seq_fops.release 0xffff812157d0 Kernel udplite4_seq_afinfo seq_ops.show 0xffff8164f9e0 Kernel udplite4_seq_afinfo seq_fops.llseek 0xffff811c9250 Kernel udplite4_seq_afinfo seq_fops.read 0xffff811c9460 Kernel udplite4_seq_afinfo seq_fops.release 0xffff812157d0 Kernel udp4_seq_afinfo seq_ops.show 0xffff8164f9e0 Kernel udp4_seq_afinfo seq_fops.llseek 0xffff811c9250 Kernel udp4_seq_afinfo seq_fops.read 0xffff811c9460 Kernel udp4_seq_afinfo seq_fops.release 0xffff812157d0 Kernel

check_creds (CheckCreds)

Checks if any processes are sharing credential structures

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
method	ChoiceArray	Method to list processes (Default uses all methods).
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

In order for rootkits to elevate the privileges of a given process, they need to alter the current effective identifier of a process. Before kernel 2.6, this was done by setting a couple of integers in the process task to the desired ID.

After 2.6, credentials are handled internally via the *task_struct->cred* member. Likely due to laziness or a poor attempt at remaining stealth, some rootkits simply reuse the *cred* member of tasks that have the desired credentials (most often ID 0: *root*).

This plugin reports the location of the *cred* member of each task. When this structure is being reused, you'll see more than one line of output with the same *cred* address.

### Sample output

```
Windows7_VMware(Win7x64+Ubuntu686,Ubuntu64)_VBox(XPSP3x86).ram 15:40:12> check_creds
-----------------------------------------------------------------------> check_creds()
     Cred      PID      Command
-------------- -------- --------------------
0x88003b86c900 966      dbus-daemon
0x88003c766480 1031     systemd-logind
0x88003c1a7380 1056     getty
0x88003c1d2180 1103     irqbalance
0x88003c1d23c0 1290     kauditd
0x88003c1a6c00 1058     getty
0x880036b2e840 1132     atd
0x88003b96d080 1055     getty
0x88003c767440 1335     bash
0x88003c1a6cc0 1074     sshd
0x88003c1d2c00 1131     cron
0x88003cbc0900 1160     login
0x88003c183140 1081     acpid
0x88003b9ded80 1042     getty
0x88003b9dee40 1049     getty
0x88003c1a78c0 1176     whoopsie
0x88003c69a480 1486     dnsmasq
0x88003cbc1440 1199     libvirtd
```

check_idt (CheckIdt)

Checks if the IDT has been altered

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

This plugin tries to identify the symbol name or location of each entry in the Interrupt Descriptor Table.

The IDT holds a list of gate descriptors. These descriptor can be task, trap or interrupt descriptors.

- Interrupt Gates are invoked via the *int* instruction. System calls, for example, can be invoked in Linux with an *int 0x80* instruction.
- Trap Gates are also invoked via the *int* instruction but don't modify the IF flag in the EFLAGS register.
- Task Gates were originally intended to facilitate task switching but are mostly not used nowadays.

The plugin provides 6 columns as output:

- __Index__: The gate number.
- __Address__: The kernel address of the gate handler.
- __Type__: Whether this is an int/trap/task gate.
- __Present__: If the gate descriptor is present.

- __DPL__: Descriptor Privilege Level. The highest ring that is allowed to call this gate.
- __Symbol__: The kernel symbol that the address points to. If it's unknown but within the kernel image, it will be *Kernel*. Otherwise, *Unknown*.

### Notes

- A value of *Kernel* in the __Symbol__ column means "as part of the kernel image", not that it's anywhere in the kernel address space.
- Rekall currently only validates the IDT at the address pointed by the kernel symbol *idt_table*. Note that on a running system, the current IDT may be different as it can be changed via the x86 *lidt* instruction.
- Entries 0x00 to 0x1F are reserved by Intel for processor exceptions.

### Sample output

```
$ python rekall/rekal.py --de -f ~/projects/actaeon64/memory_images/Windows7_
↪VMware\(Win7x64+Ubuntu686\,Ubuntu64\)_VBox\(XPSP3x86\).ram --profile_path ../rekall-
↪profiles/ --profile_path ../my-profiles/  --ept 0x17725001E check_idt
Index    Address                   Type Present DPL Symbol
----- -------------- ------------------ ------- --- ------------------------------
  0x0 0xffff816f6970    32-bit Int Gate       1   0 divide_error
  0x1 0xffff816ecc80    32-bit Int Gate       1   0 Kernel
  0x2 0xffff816ed0b0    32-bit Int Gate       1   0 nmi
  0x3 0xffff816eccc0    32-bit Int Gate       1   3 int3
  0x4 0xffff816f69a0    32-bit Int Gate       1   3 overflow
  0x5 0xffff816f69d0    32-bit Int Gate       1   0 bounds
  0x6 0xffff816f6a00    32-bit Int Gate       1   0 invalid_op
  0x7 0xffff816f6a30    32-bit Int Gate       1   0 device_not_available
  0x8 0xffff816f6a60    32-bit Int Gate       1   0 double_fault
  0x9 0xffff816f6a90    32-bit Int Gate       1   0 coprocessor_segment_overrun
  0xa 0xffff816f6ac0    32-bit Int Gate       1   0 invalid_TSS
  0xb 0xffff816f6af0    32-bit Int Gate       1   0 segment_not_present
  0xc 0xffff816ecd00    32-bit Int Gate       1   0 stack_segment
  0xd 0xffff816ecdc0    32-bit Int Gate       1   0 general_protection
  0xe 0xffff816ecdf0    32-bit Int Gate       1   0 page_fault
  0xf 0xffff816f6b20    32-bit Int Gate       1   0 spurious_interrupt_bug
 0x10 0xffff816f6b50    32-bit Int Gate       1   0 coprocessor_error
 0x11 0xffff816f6b80    32-bit Int Gate       1   0 alignment_check
 0x12 0xffff816ece50    32-bit Int Gate       1   0 machine_check
 0x13 0xffff816f6bb0    32-bit Int Gate       1   0 simd_coprocessor_error
 0x14 0xffff81d260b4    32-bit Int Gate       1   0 Unknown
 0x15 0xffff81d260bd    32-bit Int Gate       1   0 Unknown
 0x16 0xffff81d260c6    32-bit Int Gate       1   0 Unknown
 0x17 0xffff81d260cf    32-bit Int Gate       1   0 Unknown
 0x18 0xffff81d260d8    32-bit Int Gate       1   0 Unknown
 0x19 0xffff81d260e1    32-bit Int Gate       1   0 Unknown
 0x1a 0xffff81d260ea    32-bit Int Gate       1   0 Unknown
 0x1b 0xffff81d260f3    32-bit Int Gate       1   0 Unknown
 0x1c 0xffff81d260fc    32-bit Int Gate       1   0 Unknown
 0x1d 0xffff81d26105    32-bit Int Gate       1   0 Unknown
 0x1e 0xffff81d2610e    32-bit Int Gate       1   0 Unknown
 0x1f 0xffff81d26117    32-bit Int Gate       1   0 Unknown
 0x20 0xffff816f5e00    32-bit Int Gate       1   0 irq_move_cleanup_interrupt
 0x21 0xffff816f5a04    32-bit Int Gate       1   0 Kernel
 0x22 0xffff816f5a08    32-bit Int Gate       1   0 Kernel
 0x23 0xffff816f5a0c    32-bit Int Gate       1   0 Kernel
 0x24 0xffff816f5a10    32-bit Int Gate       1   0 Kernel
 0x25 0xffff816f5a14    32-bit Int Gate       1   0 Kernel
 0x26 0xffff816f5a18    32-bit Int Gate       1   0 Kernel
```

```
0x27 0xffff816f5a20    32-bit Int Gate       1   0 Kernel
0x28 0xffff816f5a24    32-bit Int Gate       1   0 Kernel
0x29 0xffff816f5a28    32-bit Int Gate       1   0 Kernel
0x2a 0xffff816f5a2c    32-bit Int Gate       1   0 Kernel
0x2b 0xffff816f5a30    32-bit Int Gate       1   0 Kernel
0x2c 0xffff816f5a34    32-bit Int Gate       1   0 Kernel
0x2d 0xffff816f5a38    32-bit Int Gate       1   0 Kernel
0x2e 0xffff816f5a40    32-bit Int Gate       1   0 Kernel
0x2f 0xffff816f5a44    32-bit Int Gate       1   0 Kernel
0x30 0xffff816f5a48    32-bit Int Gate       1   0 Kernel
0x31 0xffff816f5a4c    32-bit Int Gate       1   0 Kernel
0x32 0xffff816f5a50    32-bit Int Gate       1   0 Kernel
0x33 0xffff816f5a54    32-bit Int Gate       1   0 Kernel
0x34 0xffff816f5a58    32-bit Int Gate       1   0 Kernel
0x35 0xffff816f5a60    32-bit Int Gate       1   0 Kernel
0x36 0xffff816f5a64    32-bit Int Gate       1   0 Kernel
0x37 0xffff816f5a68    32-bit Int Gate       1   0 Kernel
0x38 0xffff816f5a6c    32-bit Int Gate       1   0 Kernel
0x39 0xffff816f5a70    32-bit Int Gate       1   0 Kernel
0x3a 0xffff816f5a74    32-bit Int Gate       1   0 Kernel
0x3b 0xffff816f5a78    32-bit Int Gate       1   0 Kernel
0x3c 0xffff816f5a80    32-bit Int Gate       1   0 Kernel
0x3d 0xffff816f5a84    32-bit Int Gate       1   0 Kernel
0x3e 0xffff816f5a88    32-bit Int Gate       1   0 Kernel
0x3f 0xffff816f5a8c    32-bit Int Gate       1   0 Kernel
0x40 0xffff816f5a90    32-bit Int Gate       1   0 Kernel
0x41 0xffff816f5a94    32-bit Int Gate       1   0 Kernel
0x42 0xffff816f5a98    32-bit Int Gate       1   0 Kernel
0x43 0xffff816f5aa0    32-bit Int Gate       1   0 Kernel
0x44 0xffff816f5aa4    32-bit Int Gate       1   0 Kernel
0x45 0xffff816f5aa8    32-bit Int Gate       1   0 Kernel
0x46 0xffff816f5aac    32-bit Int Gate       1   0 Kernel
0x47 0xffff816f5ab0    32-bit Int Gate       1   0 Kernel
0x48 0xffff816f5ab4    32-bit Int Gate       1   0 Kernel
0x49 0xffff816f5ab8    32-bit Int Gate       1   0 Kernel
0x4a 0xffff816f5ac0    32-bit Int Gate       1   0 Kernel
0x4b 0xffff816f5ac4    32-bit Int Gate       1   0 Kernel
0x4c 0xffff816f5ac8    32-bit Int Gate       1   0 Kernel
0x4d 0xffff816f5acc    32-bit Int Gate       1   0 Kernel
0x4e 0xffff816f5ad0    32-bit Int Gate       1   0 Kernel
0x4f 0xffff816f5ad4    32-bit Int Gate       1   0 Kernel
0x50 0xffff816f5ad8    32-bit Int Gate       1   0 Kernel
0x51 0xffff816f5ae0    32-bit Int Gate       1   0 Kernel
0x52 0xffff816f5ae4    32-bit Int Gate       1   0 Kernel
0x53 0xffff816f5ae8    32-bit Int Gate       1   0 Kernel
0x54 0xffff816f5aec    32-bit Int Gate       1   0 Kernel
0x55 0xffff816f5af0    32-bit Int Gate       1   0 Kernel
0x56 0xffff816f5af4    32-bit Int Gate       1   0 Kernel
0x57 0xffff816f5af8    32-bit Int Gate       1   0 Kernel
0x58 0xffff816f5b00    32-bit Int Gate       1   0 Kernel
0x59 0xffff816f5b04    32-bit Int Gate       1   0 Kernel
0x5a 0xffff816f5b08    32-bit Int Gate       1   0 Kernel
0x5b 0xffff816f5b0c    32-bit Int Gate       1   0 Kernel
0x5c 0xffff816f5b10    32-bit Int Gate       1   0 Kernel
0x5d 0xffff816f5b14    32-bit Int Gate       1   0 Kernel
0x5e 0xffff816f5b18    32-bit Int Gate       1   0 Kernel
0x5f 0xffff816f5b20    32-bit Int Gate       1   0 Kernel
0x60 0xffff816f5b24    32-bit Int Gate       1   0 Kernel
```

```
0x61 0xffff816f5b28    32-bit Int Gate       1   0 Kernel
0x62 0xffff816f5b2c    32-bit Int Gate       1   0 Kernel
0x63 0xffff816f5b30    32-bit Int Gate       1   0 Kernel
0x64 0xffff816f5b34    32-bit Int Gate       1   0 Kernel
0x65 0xffff816f5b38    32-bit Int Gate       1   0 Kernel
0x66 0xffff816f5b40    32-bit Int Gate       1   0 Kernel
0x67 0xffff816f5b44    32-bit Int Gate       1   0 Kernel
0x68 0xffff816f5b48    32-bit Int Gate       1   0 Kernel
0x69 0xffff816f5b4c    32-bit Int Gate       1   0 Kernel
0x6a 0xffff816f5b50    32-bit Int Gate       1   0 Kernel
0x6b 0xffff816f5b54    32-bit Int Gate       1   0 Kernel
0x6c 0xffff816f5b58    32-bit Int Gate       1   0 Kernel
0x6d 0xffff816f5b60    32-bit Int Gate       1   0 Kernel
0x6e 0xffff816f5b64    32-bit Int Gate       1   0 Kernel
0x6f 0xffff816f5b68    32-bit Int Gate       1   0 Kernel
0x70 0xffff816f5b6c    32-bit Int Gate       1   0 Kernel
0x71 0xffff816f5b70    32-bit Int Gate       1   0 Kernel
0x72 0xffff816f5b74    32-bit Int Gate       1   0 Kernel
0x73 0xffff816f5b78    32-bit Int Gate       1   0 Kernel
0x74 0xffff816f5b80    32-bit Int Gate       1   0 Kernel
0x75 0xffff816f5b84    32-bit Int Gate       1   0 Kernel
0x76 0xffff816f5b88    32-bit Int Gate       1   0 Kernel
0x77 0xffff816f5b8c    32-bit Int Gate       1   0 Kernel
0x78 0xffff816f5b90    32-bit Int Gate       1   0 Kernel
0x79 0xffff816f5b94    32-bit Int Gate       1   0 Kernel
0x7a 0xffff816f5b98    32-bit Int Gate       1   0 Kernel
0x7b 0xffff816f5ba0    32-bit Int Gate       1   0 Kernel
0x7c 0xffff816f5ba4    32-bit Int Gate       1   0 Kernel
0x7d 0xffff816f5ba8    32-bit Int Gate       1   0 Kernel
0x7e 0xffff816f5bac    32-bit Int Gate       1   0 Kernel
0x7f 0xffff816f5bb0    32-bit Int Gate       1   0 Kernel
0x80 0xffff816f72e0    32-bit Int Gate       1   3 ia32_syscall
0x81 0xffff816f5bb8    32-bit Int Gate       1   0 Kernel
0x82 0xffff816f5bc0    32-bit Int Gate       1   0 Kernel
0x83 0xffff816f5bc4    32-bit Int Gate       1   0 Kernel
0x84 0xffff816f5bc8    32-bit Int Gate       1   0 Kernel
0x85 0xffff816f5bcc    32-bit Int Gate       1   0 Kernel
0x86 0xffff816f5bd0    32-bit Int Gate       1   0 Kernel
0x87 0xffff816f5bd4    32-bit Int Gate       1   0 Kernel
0x88 0xffff816f5bd8    32-bit Int Gate       1   0 Kernel
0x89 0xffff816f5be0    32-bit Int Gate       1   0 Kernel
0x8a 0xffff816f5be4    32-bit Int Gate       1   0 Kernel
0x8b 0xffff816f5be8    32-bit Int Gate       1   0 Kernel
0x8c 0xffff816f5bec    32-bit Int Gate       1   0 Kernel
0x8d 0xffff816f5bf0    32-bit Int Gate       1   0 Kernel
0x8e 0xffff816f5bf4    32-bit Int Gate       1   0 Kernel
0x8f 0xffff816f5bf8    32-bit Int Gate       1   0 Kernel
0x90 0xffff816f5c00    32-bit Int Gate       1   0 Kernel
0x91 0xffff816f5c04    32-bit Int Gate       1   0 Kernel
0x92 0xffff816f5c08    32-bit Int Gate       1   0 Kernel
0x93 0xffff816f5c0c    32-bit Int Gate       1   0 Kernel
0x94 0xffff816f5c10    32-bit Int Gate       1   0 Kernel
0x95 0xffff816f5c14    32-bit Int Gate       1   0 Kernel
0x96 0xffff816f5c18    32-bit Int Gate       1   0 Kernel
0x97 0xffff816f5c20    32-bit Int Gate       1   0 Kernel
0x98 0xffff816f5c24    32-bit Int Gate       1   0 Kernel
0x99 0xffff816f5c28    32-bit Int Gate       1   0 Kernel
0x9a 0xffff816f5c2c    32-bit Int Gate       1   0 Kernel
```

```
0x9b 0xffff816f5c30    32-bit Int Gate       1   0 Kernel
0x9c 0xffff816f5c34    32-bit Int Gate       1   0 Kernel
0x9d 0xffff816f5c38    32-bit Int Gate       1   0 Kernel
0x9e 0xffff816f5c40    32-bit Int Gate       1   0 Kernel
0x9f 0xffff816f5c44    32-bit Int Gate       1   0 Kernel
0xa0 0xffff816f5c48    32-bit Int Gate       1   0 Kernel
0xa1 0xffff816f5c4c    32-bit Int Gate       1   0 Kernel
0xa2 0xffff816f5c50    32-bit Int Gate       1   0 Kernel
0xa3 0xffff816f5c54    32-bit Int Gate       1   0 Kernel
0xa4 0xffff816f5c58    32-bit Int Gate       1   0 Kernel
0xa5 0xffff816f5c60    32-bit Int Gate       1   0 Kernel
0xa6 0xffff816f5c64    32-bit Int Gate       1   0 Kernel
0xa7 0xffff816f5c68    32-bit Int Gate       1   0 Kernel
0xa8 0xffff816f5c6c    32-bit Int Gate       1   0 Kernel
0xa9 0xffff816f5c70    32-bit Int Gate       1   0 Kernel
0xaa 0xffff816f5c74    32-bit Int Gate       1   0 Kernel
0xab 0xffff816f5c78    32-bit Int Gate       1   0 Kernel
0xac 0xffff816f5c80    32-bit Int Gate       1   0 Kernel
0xad 0xffff816f5c84    32-bit Int Gate       1   0 Kernel
0xae 0xffff816f5c88    32-bit Int Gate       1   0 Kernel
0xaf 0xffff816f5c8c    32-bit Int Gate       1   0 Kernel
0xb0 0xffff816f5c90    32-bit Int Gate       1   0 Kernel
0xb1 0xffff816f5c94    32-bit Int Gate       1   0 Kernel
0xb2 0xffff816f5c98    32-bit Int Gate       1   0 Kernel
0xb3 0xffff816f5ca0    32-bit Int Gate       1   0 Kernel
0xb4 0xffff816f5ca4    32-bit Int Gate       1   0 Kernel
0xb5 0xffff816f5ca8    32-bit Int Gate       1   0 Kernel
0xb6 0xffff816f5cac    32-bit Int Gate       1   0 Kernel
0xb7 0xffff816f5cb0    32-bit Int Gate       1   0 Kernel
0xb8 0xffff816f5cb4    32-bit Int Gate       1   0 Kernel
0xb9 0xffff816f5cb8    32-bit Int Gate       1   0 Kernel
0xba 0xffff816f5cc0    32-bit Int Gate       1   0 Kernel
0xbb 0xffff816f5cc4    32-bit Int Gate       1   0 Kernel
0xbc 0xffff816f5cc8    32-bit Int Gate       1   0 Kernel
0xbd 0xffff816f5ccc    32-bit Int Gate       1   0 Kernel
0xbe 0xffff816f5cd0    32-bit Int Gate       1   0 Kernel
0xbf 0xffff816f5cd4    32-bit Int Gate       1   0 Kernel
0xc0 0xffff816f5cd8    32-bit Int Gate       1   0 Kernel
0xc1 0xffff816f5ce0    32-bit Int Gate       1   0 Kernel
0xc2 0xffff816f5ce4    32-bit Int Gate       1   0 Kernel
0xc3 0xffff816f5ce8    32-bit Int Gate       1   0 Kernel
0xc4 0xffff816f5cec    32-bit Int Gate       1   0 Kernel
0xc5 0xffff816f5cf0    32-bit Int Gate       1   0 Kernel
0xc6 0xffff816f5cf4    32-bit Int Gate       1   0 Kernel
0xc7 0xffff816f5cf8    32-bit Int Gate       1   0 Kernel
0xc8 0xffff816f5d00    32-bit Int Gate       1   0 Kernel
0xc9 0xffff816f5d04    32-bit Int Gate       1   0 Kernel
0xca 0xffff816f5d08    32-bit Int Gate       1   0 Kernel
0xcb 0xffff816f5d0c    32-bit Int Gate       1   0 Kernel
0xcc 0xffff816f5d10    32-bit Int Gate       1   0 Kernel
0xcd 0xffff816f5d14    32-bit Int Gate       1   0 Kernel
0xce 0xffff816f5d18    32-bit Int Gate       1   0 Kernel
0xcf 0xffff816f5d20    32-bit Int Gate       1   0 Kernel
0xd0 0xffff816f5d24    32-bit Int Gate       1   0 Kernel
0xd1 0xffff816f5d28    32-bit Int Gate       1   0 Kernel
0xd2 0xffff816f5d2c    32-bit Int Gate       1   0 Kernel
0xd3 0xffff816f5d30    32-bit Int Gate       1   0 Kernel
0xd4 0xffff816f5d34    32-bit Int Gate       1   0 Kernel
```

```
0xd5 0xffff816f5d38    32-bit Int Gate       1   0 Kernel
0xd6 0xffff816f5d40    32-bit Int Gate       1   0 Kernel
0xd7 0xffff816f5d44    32-bit Int Gate       1   0 Kernel
0xd8 0xffff816f5d48    32-bit Int Gate       1   0 Kernel
0xd9 0xffff816f5d4c    32-bit Int Gate       1   0 Kernel
0xda 0xffff816f5d50    32-bit Int Gate       1   0 Kernel
0xdb 0xffff816f5d54    32-bit Int Gate       1   0 Kernel
0xdc 0xffff816f5d58    32-bit Int Gate       1   0 Kernel
0xdd 0xffff816f5d60    32-bit Int Gate       1   0 Kernel
0xde 0xffff816f5d64    32-bit Int Gate       1   0 Kernel
0xdf 0xffff816f5d68    32-bit Int Gate       1   0 Kernel
0xe0 0xffff816f5d6c    32-bit Int Gate       1   0 Kernel
0xe1 0xffff816f5d70    32-bit Int Gate       1   0 Kernel
0xe2 0xffff816f5d74    32-bit Int Gate       1   0 Kernel
0xe3 0xffff816f5d78    32-bit Int Gate       1   0 Kernel
0xe4 0xffff816f5d80    32-bit Int Gate       1   0 Kernel
0xe5 0xffff816f5d84    32-bit Int Gate       1   0 Kernel
0xe6 0xffff816f5d88    32-bit Int Gate       1   0 Kernel
0xe7 0xffff816f5d8c    32-bit Int Gate       1   0 Kernel
0xe8 0xffff816f5d90    32-bit Int Gate       1   0 Kernel
0xe9 0xffff816f5d94    32-bit Int Gate       1   0 Kernel
0xea 0xffff816f5d98    32-bit Int Gate       1   0 Kernel
0xeb 0xffff816f5da0    32-bit Int Gate       1   0 Kernel
0xec 0xffff816f5da4    32-bit Int Gate       1   0 Kernel
0xed 0xffff816f5da8    32-bit Int Gate       1   0 Kernel
0xee 0xffff816f5dac    32-bit Int Gate       1   0 Kernel
0xef 0xffff816f5ef0    32-bit Int Gate       1   0 apic_timer_interrupt
0xf0 0xffff816f5db4    32-bit Int Gate       1   0 Kernel
0xf1 0xffff816f5db8    32-bit Int Gate       1   0 Kernel
```

check_modules (CheckModules)

Compares module list to sysfs info, if available.

Sysfs contains a kset objects for a number of kernel objects (kobjects). One of the ksets is the “module_kset” which holds references to all loaded kernel modules.

Each struct module object holds within it a kobj struct for reference counting. This object is referenced both from the struct module and the sysfs kset.

This plugin traverses the kset and resolves the kobj back to its containing object (which is the struct module itself). We then compare the struct module with the list of known modules (which is obtained by traversing the module’s list member). So if a module were to simply unlink itself from the list, it would still be found by its reference from sysfs.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

check_proc_fops (CheckProcFops)

Checks the proc filesystem for hooked f_ops.

Plugin	Type	Description
all	Boolean	Specify to see all the fops, even if they are known.
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

check_proc_fops checks the file operations pointers of each open file in the proc filesystem. Some rootkits hook these operations in order to implement process hiding.

In order to determine if an operation pointer is hooked, rekall checks that the pointer resides within a known module or the kernel image.

If a pointer is found outside of these bounds, it will be reported.

### Notes

- To obtain a list of all checked function pointers, use the *–all* parameter.

### Sample output

Expect blank output on clean systems.

```
pmem 15:44:30> check_proc_fops
-------------> check_proc_fops()
   DirEntry    Path                                                 Member                ␣
↪     Address      Module
-------------- -------------------------------------------------- --------------------
↪ -------------- --------------------
pmem 15:44:35>
```

check_syscall (CheckSyscall)

Checks if the system call table has been altered.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

check_syscall checks if every syscall handler points to a known symbol in the profile. All the default syscall handlers for a given kernel should be exported along with the profile and if this handler is changed, Rekall will detect it.

### Notes

1. Unknown symbols are reported as *Unknown*.
2. Only the handler pointers are checked. If the original handler is still being used but it has been patched in memory, no hook detection will be done.

### Sample output

```
```

check_ttys (CheckTTY)

Checks tty devices for hooks.

Some malware insert a hook into the ops struct of the tty driver. This plugin enumerates all tty_struct objects and checks if their ops handlers have been subverted.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

check_task_fops (CheckTaskFops)

Check open files in tasks for f_ops modifications.

Plugin	Type	Description
all	Boolean	Specify to see all the fops, even if they are known.
dtb	IntParser	The DTB physical address.
method	ChoiceArray	Method to list processes (Default uses all methods).
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

check_task_fops checks the file operations pointers of each running process' open files. Rootkits may hook these function pointers in order to control operation of specific tasks.

In order to determine if an operation pointer is hooked, rekall checks that the pointer resides within a known module or the kernel image.

If a pointer is found outside of these bounds, it will be reported.

### Notes

- To obtain a list of all checked function pointers, use the *–all* parameter.

### Sample output

Expect blank output on clean systems.

```
pmem 15:44:30> check_task_fops
-------------> check_proc_fops()
   DirEntry     Path                                              Member              ␣
↪     Address     Module
-------------- -------------------------------------------------- --------------------
↪ -------------- --------------------
pmem 15:44:35>
```

cpuinfo (CpuInfo)

Prints information about each active processor.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

### Sample output

```
[1] Windows7_VMware(Win7x64+Ubuntu686,Ubuntu64)_VBox(XPSP3x86).ram 16:07:43> cpuinfo
---------------------------------------------------------------------------> cpuinfo()
CPU         Vendor                                                     Model
---- -------------------- ---------------------------------------------------------------
↪--------------------
0    GenuineIntel         Intel(R) Core(TM) i7 CPU         930  @ 2.80GHz
1    GenuineIntel         Intel(R) Core(TM) i7 CPU         930  @ 2.80GHz
2    GenuineIntel         Intel(R) Core(TM) i7 CPU         930  @ 2.80GHz
3    GenuineIntel         Intel(R) Core(TM) i7 CPU         930  @ 2.80GHz
```

heapdump (HeapChunkDumper)

Dumps allocated/freed chunks from selected processes

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
dump_dir	String	Path suitable for dumping files.
main_arena	IntParser	The main_arena pointer either extracted from the statically linked ELF binary or from the libc library.
mal-loc_par	IntParser	The malloc_par pointer either extracted from the linked ELF binary or from the libc library.
method	ChoiceAr-ray	Method to list processes (Default uses all methods).
pids	ArrayInt-Parser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
task	ArrayInt-Parser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

heapobjects (HeapObjects)

Prints the structs of heap objects (such as allocated chunks, arenas, . . .)

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
main_arena	IntParser	The main_arena pointer either extracted from the statically linked ELF binary or from the libc library.
malloc_par	IntParser	The malloc_par pointer either extracted from the linked ELF binary or from the libc library.
method	ChoiceArray	Method to list processes (Default uses all methods).
pids	ArrayIntParser	One or more pids of processes to select.
print_allocated	Boolean	prints all allocated chunk structs
print_freed	Boolean	prints all freed chunk structs
print_mallinfo	Boolean	prints statistic information, similar to glibc's mallinfo
print_mmapped	Boolean	prints all MMAPPED chunk structs
proc_regex	RegEx	A regex to select a process by name.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

heapinfo (HeapOverview)

Tries to gather a list of all arenas/heaps and all allocated chunks.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
main_arena	IntParser	The main_arena pointer either extracted from the statically linked ELF binary or from the libc library.
malloc_par	IntParser	The malloc_par pointer either extracted from the linked ELF binary or from the libc library.
method	ChoiceArray	Method to list processes (Default uses all methods).
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

heapsearch (HeapPointerSearch)

Searches all chunks for the given string, regex or pointer(s).

Plugin	Type	Description
chunk_addresses	ArrayIntParser	Expects address(es) belonging to a chunk(s) of interest, and prints all chunks having a pointer somewhere into the data part of that chunk(s).
dtb	IntParser	The DTB physical address.
main_arena	IntParser	The main_arena pointer either extracted from the statically linked ELF binary or from the libc library.
malloc_par	IntParser	The malloc_par pointer either extracted from the linked ELF binary or from the libc library.
method	ChoiceArray	Method to list processes (Default uses all methods).
pids	ArrayIntParser	One or more pids of processes to select.
pointers	ArrayIntParser	Prints chunks that contain exactly the given pointer(s). The pointer(s) can be given as (hexa)decimal numbers.
proc_regex	RegEx	A regex to select a process by name.
regex	str	Searches all chunks with the given regex and prints all hits.
search_struct	Boolean	Includes the malloc_struct fields in the search process such as size and fd/bk for bin chunks (but not its own prev_size field). This is normally not desired and hence deactivated by default.
string	str	Searches all chunks for the given string and prints all hits.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

heaprefs (HeapReferenceSearch)

Examines the data part of the given chunk for references to other chunks.

Plugin	Type	Description
chunk_addresses	ArrayIntParser	The address(es) belonging to chunks of interest. Those chunks are then examined for references to other chunks.
dtb	IntParser	The DTB physical address.
main_arena	IntParser	The main_arena pointer either extracted from the statically linked ELF binary or from the libc library.
malloc_par	IntParser	The malloc_par pointer either extracted from the linked ELF binary or from the libc library.
method	ChoiceArray	Method to list processes (Default uses all methods).
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

hostname (Hostname)

A mixin for those plugins requiring a physical address space.

Args:

physical_address_space: The physical address space to use. If not specified we use the following options:

1. session.physical_address_space,
2. Guess using the load_as() plugin,
3. Use session.kernel_address_space.base.

Plugin	Type	Description
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

iomem (IOmem)

mimics /proc/iomem.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

### Sample output

```
[1] Windows7_VMware(Win7x64+Ubuntu686,Ubuntu64)_VBox(XPSP3x86).ram 16:22:13> iomem
---------------------------------------------------------------------------> iomem()
   Resource          Start            End         Name
-------------- -------------- -------------- ----
0xffff81c3abc0  0x000000000000 0x00ffffffffff
0x88003fff9b00 . 0x000000000000 0x000000000fff reserved
0x88003fff9b38 . 0x000000001000 0x00000009ebff System RAM
0x88003fff9b70 . 0x00000009ec00 0x00000009ffff reserved
0x88003d112200 . 0x0000000a0000 0x0000000bffff PCI Bus 0000:00
0xffff81c1aac0 . 0x0000000c0000 0x0000000c7fff Video ROM
0x88003fff9ba8 . 0x0000000ca000 0x0000000cbfff reserved
0xffff81c1ab00 .. 0x0000000ca000 0x0000000cafff Adapter ROM
0x88003d112238 . 0x0000000d0000 0x0000000d3fff PCI Bus 0000:00
0x88003d112270 . 0x0000000d4000 0x0000000d7fff PCI Bus 0000:00
0x88003d1122a8 . 0x0000000d8000 0x0000000dbfff PCI Bus 0000:00
0x88003fff9be0 . 0x0000000dc000 0x0000000fffff reserved
0xffff81c1aca0 .. 0x0000000f0000 0x0000000fffff System ROM
0x88003fff9c18 . 0x000000100000 0x00003fedffff System RAM
0xffff81c1a6a0 .. 0x000001000000 0x0000016f9945 Kernel code
0xffff81c1a6e0 .. 0x0000016f9946 0x000001d0e7ff Kernel data
0xffff81c1a660 .. 0x000001e6d000 0x000001fcffff Kernel bss
0x88003fff9c50 . 0x00003fee0000 0x00003fefefff ACPI Tables
0x88003fff9c88 . 0x00003feff000 0x00003fefffff ACPI Non-volatile Storage
0x88003fff9cc0 . 0x00003ff00000 0x00003fffffff System RAM
0x88003d1122e0 . 0x0000c0000000 0x0000febfffff PCI Bus 0000:00
0x88003d1a0488 .. 0x0000c0000000 0x0000c0007fff 0000:00:0f.0
0x88003d1a1488 .. 0x0000c0008000 0x0000c000bfff 0000:00:10.0
0x88003d202680 .. 0x0000e5b00000 0x0000e5bfffff
0x88003d1da680 .. 0x0000e5c00000 0x0000e5cfffff PCI Bus 0000:1a
0x88003d1d2680 .. 0x0000e5d00000 0x0000e5dfffff PCI Bus 0000:12
```

```
0x88003d1ca680 .. 0x0000e5e00000 0x0000e5efffff
0x88003d201680 .. 0x000000000000 0x000000000000 -
0x88003fff9d30 . 0x0000fec00000 0x0000fec0ffff reserved
0x88003fff9e00 .. 0x0000fec00000 0x0000fec003ff IOAPIC 0
0x88003fff9e80 . 0x0000fed00000 0x0000fed003ff HPET 0
0x88003d2ca500 .. 0x0000fed00000 0x0000fed003ff pnp 00:07
0xffff81c25cc0 . 0x0000fee00000 0x0000fee00fff Local APIC
0x88003fff9d68 .. 0x0000fee00000 0x0000fee00fff reserved
0x88003fff9da0 . 0x0000fffe0000 0x0000ffffffff reserved
```

ifconfig (Ifconfig)

Gathers active interfaces.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

### Sample output

```
[1] Windows7_VMware(Win7x64+Ubuntu686,Ubuntu64)_VBox(XPSP3x86).ram 16:12:17> ifconfig
--------------------------------------------------------------------------->␣
↪ifconfig()
   Interface         Ipv4Address              MAC                Flags
---------------- -------------------- ------------------ --------------------
lo               127.0.0.1            00:00:00:00:00:00  IFF_LOOPBACK, IFF_UP
eth0             192.168.239.129      00:0C:29:57:F7:19  IFF_BROADCAST, IFF_MULTICAST,
↪ IFF_UP
```

keepassx (Keepassx)

Gathers password entries for keepassx. The retrieved content of those entries comprises the username, title, URL and Comment.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
main_arena	IntParser	The main_arena pointer either extracted from the statically linked ELF binary or from the libc library.
mal-loc_par	IntParser	The malloc_par pointer either extracted from the linked ELF binary or from the libc library.
method	ChoiceAr-ray	Method to list processes (Default uses all methods).
pids	ArrayInt-Parser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
task	ArrayInt-Parser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

memdump (LinMemDump)

Dump the addressable memory for a process.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
dump_dir	String	Path suitable for dumping files.
method	ChoiceArray	Method to list processes (Default uses all methods).
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

memmap (LinMemMap)

Dumps the memory map for linux tasks.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
method	ChoiceArray	Method to list processes (Default uses all methods).
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

moddump (LinModdump)

Dumps loaded kernel modules.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
dump_dir	String	Dump directory.
regexp	RegEx	Regexp on the module name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

pstree (LinPSTree)

Shows the parent/child relationship between processes.

This plugin prints a parent/child relationship tree by walking the task_struct.children and task_struct.sibling members.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

pas2vas (LinPas2Vas)

Resolves a physical address to a virtual addrress in a process.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
method	ChoiceArray	Method to list processes (Default uses all methods).
offsets	ArrayIntParser	A list of physical offsets to resolve.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

vaddump (LinVadDump)

Dump the VMA memory for a process.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
dump_dir	String	Path suitable for dumping files.
method	ChoiceArray	Method to list processes (Default uses all methods).
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

vtop (LinVtoP)

Describe virtual to physical translation on Linux platforms.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
method	ChoiceArray	Method to list processes (Default uses all methods).
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

yarascan (LinYaraScan)

Scan using yara signatures.

Plugin	Type	Description
binary_string	String	A binary string (encoded as hex) to search for. e.g. 000102[1-200]0506
context	IntParser	Context to print after the hit.
dtb	IntParser	The DTB physical address.
hits	IntParser	Quit after finding this many hits.
method	ChoiceArray	Method to list processes (Default uses all methods).
pids	ArrayIntParser	One or more pids of processes to select.
pre_context	IntParser	Context to print before the hit.
proc_regex	RegEx	A regex to select a process by name.
scan_kernel	Boolean	Scan the entire kernel address space.
scan_physical	Boolean	Scan the physical address space only.
scan_process_memory	Boolean	Scan all of process memory. Uses process selectors to narrow down selections.
string	String	A verbatim string to search for.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.
yara_expression	String	If provided we scan for this yara expression.
yara_file	String	The yara signature file to read.

address_resolver (LinuxAddressResolver)

A Linux specific address resolver plugin.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
symbol	ArrayString	List of symbols to lookup
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

dmesg (LinuxDmesg)

Gathers dmesg buffer.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

### Sample output

```
[1] Windows7_VMware(Win7x64+Ubuntu686,Ubuntu64)_VBox(XPSP3x86).ram 16:07:44> dmesg
---------------------------------------------------------------------------> dmesg()
Timestamp Facility Level                                     Message
--------- -------- ----- -------------------------------------------------------------
↪-------------------
     0.00 0        LOG_INFO Initializing cgroup subsys cpuset
     0.00 0        LOG_INFO Initializing cgroup subsys cpu
     0.00 0        LOG_INFO Initializing cgroup subsys cpuacct
     0.00 0        LOG_INFO Linux version 3.11.0-12-generic (buildd@allspice) (gcc␣
↪version 4.8.1 (Ubuntu/Linaro 4.8.1-10ubuntu7) ) #19-Ubuntu SMP Wed Oct 9 16:20:46␣
↪UTC 2013 (Ubuntu 3.11.0-12.19-generic 3.11.3)
```

```
    0.00 0         LOG_INFO Command line: BOOT_IMAGE=/vmlinuz-3.11.0-12-generic root=/
↪dev/mapper/ubuntu--vmware--vg-root ro
    0.00 0         LOG_INFO KERNEL supported cpus:
    0.00 0         LOG_INFO   Intel GenuineIntel
    0.00 0         LOG_INFO   AMD AuthenticAMD
    0.00 0         LOG_INFO   Centaur CentaurHauls
    0.00 0         LOG_INFO Disabled fast string operations
    0.00 0         LOG_INFO e820: BIOS-provided physical RAM map:
    0.00 0         LOG_CRIT BIOS-e820: [mem 0x0000000000000000-0x000000000009ebff]␣
↪usable
    0.00 0         LOG_CRIT BIOS-e820: [mem 0x000000000009ec00-0x000000000009ffff]␣
↪reserved
    0.00 0         LOG_CRIT BIOS-e820: [mem 0x00000000000ca000-0x00000000000cbfff]␣
↪reserved
    0.00 0         LOG_CRIT BIOS-e820: [mem 0x00000000000dc000-0x00000000000fffff]␣
↪reserved
    0.00 0         LOG_CRIT BIOS-e820: [mem 0x0000000000100000-0x000000003fedffff]␣
↪usable
    0.00 0         LOG_CRIT BIOS-e820: [mem 0x000000003fee0000-0x000000003fefefff]␣
↪ACPI data
    0.00 0         LOG_CRIT BIOS-e820: [mem 0x000000003feff000-0x000000003fefffff]␣
↪ACPI NVS
    0.00 0         LOG_CRIT BIOS-e820: [mem 0x000000003ff00000-0x000000003fffffff]␣
↪usable
    0.00 0         LOG_CRIT BIOS-e820: [mem 0x00000000f0000000-0x00000000f7ffffff]␣
↪reserved
    0.00 0         LOG_CRIT BIOS-e820: [mem 0x00000000fec00000-0x00000000fec0ffff]␣
↪reserved
    0.00 0         LOG_CRIT BIOS-e820: [mem 0x00000000fee00000-0x00000000fee00fff]␣
↪reserved
    0.00 0         LOG_CRIT BIOS-e820: [mem 0x00000000fffe0000-0x00000000ffffffff]␣
↪reserved
    0.00 0         LOG_INFO NX (Execute Disable) protection: active
    0.00 0         LOG_INFO SMBIOS 2.4 present.
    0.00 0         LOG_INFO DMI: VMware, Inc. VMware Virtual Platform/440BX Desktop␣
↪Reference Platform, BIOS 6.00 07/31/2013
    0.00 0         LOG_INFO Hypervisor detected: VMware
    0.00 0         LOG_CRIT e820: update [mem 0x00000000-0x00000fff] usable ==>␣
↪reserved
    0.00 0         LOG_CRIT e820: remove [mem 0x000a0000-0x000fffff] usable
    0.00 0         LOG_INFO
```

find_dtb (LinuxFindDTB)

A scanner for DTB values. Handles both 32 and 64 bits.

The plugin also autodetects when the guest is running as a XEN ParaVirtualized guest and returns a compatible address space.

Plugin	Type	Description
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

pslist (LinuxPsList)

Gathers active tasks by walking the task_struct->task list.

It does not display the swapper process. If the DTB column is blank, the item is likely a kernel thread.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
method	ChoiceArray	Method to list processes (Default uses all methods).
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

psxview (LinuxPsxView)

Find hidden processes comparing various process listings.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
method	ChoiceArray	Method to list processes.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

cc (LinuxSetProcessContext)

A cc plugin for windows.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
method	ChoiceArray	Method to list processes (Default uses all methods).
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

sigscan (LinuxSigScan)

Runs a signature scans against physical, kernel or process memory.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
method	ChoiceArray	Method to list processes (Default uses all methods).
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

vadmap (LinuxVADMap)

Inspect each page in the VAD and report its status.

This allows us to see the address translation status of each page in the VAD.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
end	IntParser	Stop reading at this offset.
method	ChoiceArray	Method to list processes (Default uses all methods).
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
start	IntParser	Start reading from this page.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

lsmod (Lsmod)

Gathers loaded kernel modules.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

Rekall walks the list at kernel symbol *modules* to provide the list of modules.

### Sample output

```
[1] Windows7_VMware(Win7x64+Ubuntu686,Ubuntu64)_VBox(XPSP3x86).ram 16:22:54> lsmod
-----------------------------------------------------------------------------> lsmod()

******************** Overview ********************
   Virtual        Core Start    Total Size         Name
-------------- -------------- ---------- --------------------
0xffffa038d120 0xffffa038b000      12880 ipt_MASQUERADE
0xffffa0383180 0xffffa0381000      13011 iptable_nat
```

lsmod_sections (LsmodSections)

Display all the ELF sections of kernel modules.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

lsmod_parameters (Lsmod_parameters)

Display parameters for all kernel modules.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

lsof (Lsof)

Lists open files.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
method	ChoiceArray	Method to list processes (Default uses all methods).
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

Rekall walks the process table and dereferences each of the *task.files.fds* for each kernel task.

### Sample output

```
$ python rekall/rekal.py -f ~/memory_images/Windows7_VMware\(Win7x64+Ubuntu686,
↪Ubuntu64\)_VBox\(XPSP3x86\).ram  --ept 0x00017725001e - lsof
[...]
libvirtd             1199           0       13 -                      0 -         -
libvirtd             1199           0       14            0           0         0␣
↪socket:/NETLINK[0]
libvirtd             1199           0       15            0           0     12987␣
↪socket:/UNIX[12987]
libvirtd             1199           0       16 -                      0 -         ␣
↪proc
libvirtd             1199           0       17            0           0         0␣
↪socket:/NETLINK[0]
libvirtd             1199           0       18            0           0      8902 /
↪run/libvirt/network/nwfilter.leases
libvirtd             1199           0       19            0           0      7861 -
bash                 1335           0        0 -                      0 -         -
bash                 1335           0        1 -                      0 -         -
bash                 1335           0        2 -                      0 -         -
bash                 1335           0      255 -                      0 -         -
```

mcat (Mcat)

Returns the contents available in memory for a given file.

Ranges of the file that are not present in memory are returned blank.

Plugin	Type	Description
device	String	Name of the device to match.
dtb	IntParser	The DTB physical address.
dump_dir	String	Path suitable for dumping files.
path	String	Path to the file.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

You can find the list of files in memory by using the *mls* plugin.

mfind (Mfind)

Finds a file by name in memory.

Plugin	Type	Description
device	String	Name of the device to match.
dtb	IntParser	The DTB physical address.
path	String	Path to the file.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

mfind can and will find multiple files if more than one file potentially matches the path. You can use the *–device* parameter to filter out by device name if you want to target a specific mountpoint.

### Sample output

```
[1] Linux-3.2.0-4-686-pae.E01 12:56:58> mfind "/etc/passd"
--------------------------------------> mfind("/etc/passd")
[1] Linux-3.2.0-4-686-pae.E01 12:58:00> mfind "/etc/passwd"
--------------------------------------> mfind("/etc/passwd")
Files on device /dev/disk/by-uuid/55bda481-150f-442e-b781-231a904cebd1 mounted at /.
   Perms        uid         gid          size                 mtime                  ␣
↪atime                    ctime                inode                              path
----------- ---------- ---------- -------------- ------------------------ ------------
↪------------ ------------------------ ---------- -----------------------------------
↪-------------------------
-rw-r--r--           0          0            942 2013-12-03 12:21:50+0000 2014-11-28␣
↪10:59:14+0000 2013-12-03 12:21:50+0000        128 /etc/passwd
[1] Linux-3.2.0-4-686-pae.E01 12:58:05> mfind "/dev/pts/0"
--------------------------------------> mfind("/dev/pts/0")
[1] Linux-3.2.0-4-686-pae.E01 12:58:12> mfind "/dev/pts"
--------------------------------------> mfind("/dev/pts")
Files on device devpts mounted at /dev/pts.
   Perms        uid         gid          size                 mtime                  ␣
↪atime                    ctime                inode                              path
----------- ---------- ---------- -------------- ------------------------ ------------
↪------------ ------------------------ ---------- -----------------------------------
↪-------------------------
drwxr-xr-x           0          0              0 2014-11-28 11:40:08+0000 2014-11-28␣
↪11:40:08+0000 2014-11-28 11:40:08+0000          1 /dev/pts
Files on device udev mounted at /dev.
   Perms        uid         gid          size                 mtime                  ␣
↪atime                    ctime                inode                              path
----------- ---------- ---------- -------------- ------------------------ ------------
↪------------ ------------------------ ---------- -----------------------------------
↪-------------------------
drwxr-xr-x           0          0             40 2014-11-28 11:40:08+0000 2014-11-28␣
↪11:40:08+0000 2014-11-28 11:40:08+0000       1137 /dev/pts
```

mls (Mls)

Lists the files in a mounted filesystem.

Plugin	Type	Description
device	String	Name of the device to match.
dtb	IntParser	The DTB physical address.
path	String	Path to the file.
recursive	Boolean	Recursive listing
unallocated	Boolean	Show files that have no inode information.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

### Sample output

```
$ PYTHONPATH=. python rekall/rekal.py -f Linux-3.2.0-4-686-pae.E01 --profile_path ../
↪my-profiles/ https://raw.githubusercontent.com/google/rekall-profiles/master/ - mls
↪"/"
Files on device /dev/disk/by-uuid/55bda481-150f-442e-b781-231a904cebd1 mounted at /.
   Perms       uid        gid          size                mtime                  ␣
↪atime                   ctime             inode                              path
----------- ---------- ---------- ------------- ------------------------ ------------
↪------------ ------------------------ ---------- -----------------------------------
↪-------------------------
---------            0          0             0 -                        -           ␣
↪              -                                  0 /
drwxr-xr-x           0          0          4096 2013-12-03 12:18:39+0000 2012-06-01␣
↪01:12:42+0000 2013-12-03 12:18:39+0000        576 /bin
drwxr-xr-x           0          0          4096 2013-12-03 12:14:16+0000 2013-12-03␣
↪12:19:41+0000 2013-12-03 12:14:16+0000        574 /dev
drwxr-xr-x           0          0          4096 2014-11-28 11:40:10+0000 2014-11-28␣
↪11:41:27+0000 2014-11-28 11:40:10+0000         15 /etc
drwxr-xr-x           0          0          4096 2013-12-03 13:25:13+0000 2014-01-28␣
↪11:40:22+0000 2013-12-03 13:25:13+0000        247 /lib
drwxr-xr-x           0          0          4096 2013-12-03 12:12:06+0000 2013-12-03␣
↪12:15:21+0000 2013-12-03 12:12:06+0000         17 /media
drwxr-xr-x           0          0          4096 2013-10-07 15:25:28+0000 2013-10-07␣
↪15:25:28+0000 2013-12-03 12:12:10+0000        571 /proc
drwx------           0          0          4096 2014-02-24 13:05:51+0000 2014-11-28␣
↪11:02:32+0000 2014-02-24 13:05:51+0000        570 /root
drwxr-xr-x           0          0          4096 2013-12-03 12:20:20+0000 2013-12-03␣
↪12:20:20+0000 2013-12-03 12:20:20+0000        569 /run
drwxr-xr-x           0          0          4096 2013-12-03 12:20:20+0000 2013-07-18␣
↪03:10:56+0000 2013-12-03 12:20:20+0000        230 /sbin
drwxr-xr-x           0          0          4096 2012-06-10 07:11:32+0000 2012-06-10␣
↪07:11:32+0000 2013-12-03 12:12:10+0000        734 /selinux
drwxr-xr-x           0          0          4096 2013-07-18 03:10:52+0000 2013-07-18␣
↪03:10:52+0000 2013-12-03 12:12:11+0000        568 /sys
drwxrwxrwxt          0          0          4096 2014-11-28 11:40:10+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:10+0000        567 /tmp
drwxr-xr-x           0          0          4096 2013-12-03 12:12:13+0000 2014-01-28␣
↪12:14:39+0000 2013-12-03 12:12:13+0000        168 /usr
drwxr-xr-x           0          0          4096 2013-12-03 12:12:13+0000 2013-12-03␣
↪12:19:03+0000 2013-12-03 12:12:13+0000         12 /var
**************************************************
Files on device devtmpfs mounted at /.
   Perms       uid        gid          size                mtime                  ␣
↪atime                   ctime             inode                              path
----------- ---------- ---------- ------------- ------------------------ ------------
↪------------ ------------------------ ---------- -----------------------------------
↪-------------------------
lrwxrwxrwx           0          0             9 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:10+0000 2014-11-28 11:40:09+0000       3464 /MAKEDEV
```

```
---------            0           0              0 -                        -          ␣
↪            -                                    0 /autofs
---------            0           0              0 -                        -          ␣
↪            -                                    0 /block
crw-------T          0           0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       3041 /btrfs-control
---------            0           0              0 -                        -          ␣
↪            -                                    0 /bus
drwxr-xr-x           0           0           2440 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:08+0000 2014-11-28 11:40:09+0000       1184 /char
crw-------           0           0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1037 /console
lrwxrwxrwx           0           0             11 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:10+0000 2014-11-28 11:40:09+0000       3030 /core
---------            0           0              0 -                        -          ␣
↪            -                                    0 /cpu
crw-------           0           0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1129 /cpu_dma_latency
---------            0           0              0 -                        -          ␣
↪            -                                    0 /disk
lrwxrwxrwx           0           0             13 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:10+0000 2014-11-28 11:40:09+0000       3034 /fd
crw-rw-rw-           0           0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1031 /full
---------            0           0              0 -                        -          ␣
↪            -                                    0 /fuse
crw-------           0           0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       3721 /hidraw0
crw-------           0           0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1113 /hpet
drwxr-xr-x           0           0            260 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:10+0000 2014-11-28 11:40:09+0000       1114 /input
crw-------           0           0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1034 /kmsg
srw-rw-rw-           0           0              0 2014-11-28 11:40:10+0000 2014-11-28␣
↪11:40:10+0000 2014-11-28 11:40:10+0000       4761 /log
crw-------T          0           0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       3042 /loop-control
---------            0           0              0 -                        -          ␣
↪            -                                    0 /loop0
brw-rw----T          0           6              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       4253 /loop1
brw-rw----T          0           6              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       4256 /loop2
brw-rw----T          0           6              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       4259 /loop3
brw-rw----T          0           6              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       4264 /loop4
brw-rw----T          0           6              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       4267 /loop5
brw-rw----T          0           6              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       4271 /loop6
brw-rw----T          0           6              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       4274 /loop7
---------            0           0              0 -                        -          ␣
↪            -                                    0 /mapper
crw-r-----T          0          15              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1027 /mem
```

```
---------            0          0              0 -                        -          ␣
↪             -                                  0 /net
crw-------           0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1130 /network_latency
crw-------           0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1131 /network_throughput
crw-rw-rw-           0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1028 /null
crw-------           0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1035 /oldmem
crw-r-----T          0         15              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1029 /port
---------            0          0              0 -                        -          ␣
↪             -                                  0 /ppp
crw-------           0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1116 /psaux
crw-rw-rw-           0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1107 /ptmx
drwxr-xr-x           0          0             40 2014-11-28 11:40:08+0000 2014-11-28␣
↪11:40:08+0000 2014-11-28 11:40:08+0000       1137 /pts
crw-rw-rw-           0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1032 /random
lrwxrwxrwx           0          0              4 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:10+0000 2014-11-28 11:40:09+0000       3731 /root
---------            0          0              0 -                        -          ␣
↪             -                                  0 /rtc
crw-------           0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1117 /rtc0
lrwxrwxrwx           0          0              8 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       3947 /shm
crw-------           0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1106 /snapshot
---------            0          0              0 -                        -          ␣
↪             -                                  0 /snd
---------            0          0              0 -                        -          ␣
↪             -                                  0 /sndstat
lrwxrwxrwx           0          0             15 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:10+0000 2014-11-28 11:40:09+0000       3040 /stderr
lrwxrwxrwx           0          0             15 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:10+0000 2014-11-28 11:40:09+0000       3036 /stdin
lrwxrwxrwx           0          0             15 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:10+0000 2014-11-28 11:40:09+0000       3038 /stdout
crw-rw-rw-           0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1036 /tty
crw-------           0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1038 /tty0
crw-------           0          5              0 2014-11-28 11:41:20+0000 2014-11-28␣
↪11:41:20+0000 2014-11-28 11:40:16+0000       1043 /tty1
crw-------           0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1052 /tty10
crw-------           0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1053 /tty11
crw-------           0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1054 /tty12
crw-------           0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1055 /tty13
crw-------           0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1056 /tty14
```

```
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1057 /tty15
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1058 /tty16
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1059 /tty17
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1060 /tty18
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1061 /tty19
crw-rw----          0          5              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:10+0000       1044 /tty2
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1062 /tty20
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1063 /tty21
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1064 /tty22
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1065 /tty23
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1066 /tty24
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1067 /tty25
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1068 /tty26
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1069 /tty27
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1070 /tty28
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1071 /tty29
crw-rw----          0          5              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:10+0000       1045 /tty3
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1072 /tty30
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1073 /tty31
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1074 /tty32
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1075 /tty33
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1076 /tty34
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1077 /tty35
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1078 /tty36
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1079 /tty37
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1080 /tty38
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1081 /tty39
crw-rw----          0          5              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:10+0000       1046 /tty4
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1082 /tty40
```

```
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1083 /tty41
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1084 /tty42
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1085 /tty43
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1086 /tty44
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1087 /tty45
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1088 /tty46
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1089 /tty47
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1090 /tty48
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1091 /tty49
crw-rw----          0          5              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:10+0000       1047 /tty5
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1092 /tty50
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1093 /tty51
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1094 /tty52
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1095 /tty53
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1096 /tty54
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1097 /tty55
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1098 /tty56
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1099 /tty57
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1100 /tty58
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1101 /tty59
crw-rw----          0          5              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:10+0000       1048 /tty6
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1102 /tty60
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1103 /tty61
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1104 /tty62
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1105 /tty63
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1049 /tty7
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1050 /tty8
crw-------          0          0              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1051 /tty9
crw-rw----T         0         20              0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1112 /ttyS0
```

```
crw-rw----T          0         20               0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1109 /ttyS1
crw-rw----T          0         20               0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1110 /ttyS2
crw-rw----T          0         20               0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1111 /ttyS3
---------            0          0               0 -                          -          ␣
↪              -                                  0 /uinput
crw-rw-rw-           0          0               0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1033 /urandom
crw-------           0          0               0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1039 /vcs
crw-------           0          0               0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1041 /vcs1
crw-------           0          0               0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       3897 /vcs2
---------            0          0               0 -                          -          ␣
↪              -                                  0 /vcs3
crw-------           0          0               0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       3907 /vcs4
crw-------           0          0               0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       3912 /vcs5
crw-------           0          0               0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       3917 /vcs6
crw-------           0          0               0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1040 /vcsa
crw-------           0          0               0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1042 /vcsa1
crw-------           0          0               0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       3898 /vcsa2
crw-------           0          0               0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       3903 /vcsa3
crw-------           0          0               0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       3908 /vcsa4
crw-------           0          0               0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       3913 /vcsa5
crw-------           0          0               0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       3918 /vcsa6
---------            0          0               0 -                          -          ␣
↪              -                                  0 /vda
---------            0          0               0 -                          -          ␣
↪              -                                  0 /vda1
---------            0          0               0 -                          -          ␣
↪              -                                  0 /vda2
---------            0          0               0 -                          -          ␣
↪              -                                  0 /vda5
crw-------           0          0               0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1026 /vga_arbiter
frw-r-----           0          4               0 2014-11-28 11:40:20+0000 2014-11-28␣
↪11:40:10+0000 2014-11-28 11:40:20+0000       4753 /xconsole
crw-rw-rw-           0          0               0 2014-11-28 11:40:09+0000 2014-11-28␣
↪11:40:09+0000 2014-11-28 11:40:09+0000       1030 /zero
**************************************************
```

Note that sometimes you may have to specify the right device in order to only get the data you want. Like in this example. Use the –device parameter in that case.

```
$ PYTHONPATH=. python rekall/rekal.py -f Linux-3.2.0-4-686-pae.E01 --profile_path ../
↪my-profiles/ https://raw.githubusercontent.com/google/rekall-profiles/master/ - mls
↪"/" --device="/dev/disk/by-uuid/55bda481-150f-442e-b781-231a904cebd1"
```

mount (Mount)

Lists the mount points.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

### Sample output

```
[1] Linux-3.2.0-4-686-pae.E01 12:56:57> mount
---------------------------------------> mount()
                      Device                                                 Path         ␣
↪                      Type             flags
-------------------------------------------------- -----------------------------------
↪--------------- -------------- --------------------
proc                                                /proc                                 ␣
↪                 proc           rw, nodev, noexec, nosuid, relatime
devpts                                              /dev/pts                              ␣
↪                 devpts         rw, noexec, nosuid, relatime
tmpfs                                               /run/lock                             ␣
↪                 tmpfs          rw, nodev, noexec, nosuid, relatime
tmpfs                                               /run/shm                              ␣
↪                 tmpfs          rw, nodev, noexec, nosuid, relatime
udev                                                /dev                                  ␣
↪                 devtmpfs       rw, relatime
tmpfs                                               /run                                  ␣
↪                 tmpfs          rw, noexec, nosuid, relatime
rpc_pipefs                                          /var/lib/nfs/rpc_pipefs               ␣
↪                 rpc_pipefs     rw, relatime
/dev/disk/by-uuid/55bda481-150f-442e-b781-231a904cebd1 /                                  ␣
↪                     ext4           rw, relatime
devtmpfs                                            /                                     ␣
↪                 devtmpfs       rw, relatime
sysfs                                               /sys                                  ␣
↪                 sysfs          rw, nodev, noexec, nosuid, relatime
```

netstat (Netstat)

Print the active network connections.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

notifier_chains (NotifierChainPlugin)

Outputs and verifies kernel notifier chains.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The Linux kernel can notify modules on certain events. This is done by subscribing to a notifier chain. A notifier chain is an ordered list of functions the kernel will call when an event is triggered.

Rekall analyzes the following notifier chains and shows whether there's any callback function registered:

- The *keyboard_notifier_list*, which is used to notify on keyboard events and some keyloggers use.
- *vt_notifier_list*, which is used to notify when there's events on a virtual terminal and could be used to assist in monitoring ttys.

Normally, no callbacks will be registered in any of these notifier chains and Rekall will produce no output.

### Sample output

```
$ PYTHONPATH=. python rekall/rekal.py -f Linux-3.2.0-4-686-pae.E01 --profile_path ../
↪my-profiles/ https://raw.githubusercontent.com/google/rekall-profiles/master/ -␣
↪notifier_chains
    Chain symbol        Index Priority  Address          Module                    ␣
↪    Symbol
------------------------- ----- -------- ---------- -------------------- -------------
↪---------------------------
```

psaux (PSAux)

Gathers processes along with full command line and start time.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
method	ChoiceArray	Method to list processes (Default uses all methods).
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

pkt_queues (PacketQueues)

Dumps the current packet queues for all known open sockets.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

pidhashtable (PidHashTable)

List processes by enumerating the pid hash tables.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
method	ChoiceArray	Method to list processes (Default uses all methods).
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

maps (ProcMaps)

Gathers process maps for linux.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
method	ChoiceArray	Method to list processes (Default uses all methods).
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
task	ArrayIntParser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

### Sample output

```
[1] Windows7_VMware(Win7x64+Ubuntu686,Ubuntu64)_VBox(XPSP3x86).ram 17:18:41> maps
-----------------------------------------------------------------------> maps()
  Pid        Start              End        Flags      Pgoff      Major  Minor      Inode ␣
↪                                           File Path
-------- -------------- -------------- ------ -------------- ------ ------ -----------
↪-- ------------------------------------------------------------------------------
966      0x000000000000 0x000000000000 ---    0x000000000000 0      0      0
1031     0x000000400000 0x00000043a000 r-x    0x000000000000 -      -      -           ␣
↪    -
1031     0x000000639000 0x00000063a000 r--    0x000000039000 -      -      -           ␣
↪    -
1031     0x00000063a000 0x00000063b000 rw-    0x00000003a000 -      -      -           ␣
↪    -
1031     0x0000012be000 0x0000012df000 rw-    0x000000000000 0      0      0           ␣
↪    [heap]
1031     0x000000000000 0x000000000000 ---    0x000000000000 0      0      0
1042     0x000000000000 0x000000000000 ---    0x000000000000 0      0      0
1056     0x000000400000 0x000000407000 r-x    0x000000000000 -      -      0           ␣
↪    /sbin/getty
1056     0x000000606000 0x000000607000 r--    0x000000006000 -      -      0           ␣
↪    /sbin/getty
1056     0x000000607000 0x000000608000 rw-    0x000000007000 -      -      0           ␣
↪    /sbin/getty
1056     0x000000608000 0x00000060a000 rw-    0x000000000000 0      0      0
1056     0x000000000000 0x000000000000 ---    0x000000000000 0      0      0
1058     0x000000400000 0x000000407000 r-x    0x000000000000 -      -      0           ␣
↪    /sbin/getty
1058     0x000000606000 0x000000607000 r--    0x000000006000 -      -      0           ␣
↪    /sbin/getty
1058     0x000000607000 0x000000608000 rw-    0x000000007000 -      -      0           ␣
↪    /sbin/getty
1058     0x000000608000 0x00000060a000 rw-    0x000000000000 0      0      0
```

```
1058     0x00000194c000 0x00000196d000 rw-    0x000000000000 0      0      0          ␣
↪   [heap]
1058     0x7f44e0f56000 0x7f44e1493000 r--    0x000000000000 252    0      660935     ␣
↪   /usr/lib/locale/locale-archive
1058     0x000000000000 0x000000000000 ---    0x000000000000 0      0      0
1074     0x7f8f09279000 0x7f8f09285000 r-x    0x000000000000 -      -      0          ␣
↪   /lib/x86_64-linux-gnu/libnss_files-2.17.so
1074     0x7f8f09285000 0x7f8f09484000 ---    0x00000000c000 -      -      0          ␣
↪   /lib/x86_64-linux-gnu/libnss_files-2.17.so
1074     0x7f8f09484000 0x7f8f09485000 r--    0x00000000b000 -      -      0          ␣
↪   /lib/x86_64-linux-gnu/libnss_files-2.17.so
1074     0x7f8f09485000 0x7f8f09486000 rw-    0x00000000c000 -      -      0          ␣
↪   /lib/x86_64-linux-gnu/libnss_files-2.17.so
1074     0x7f8f09486000 0x7f8f09491000 r-x    0x000000000000 -      -      -          ␣
↪   -
1074     0x7f8f09491000 0x7f8f09690000 ---    0x00000000b000 -      -      -          ␣
↪   -
1074     0x7f8f09690000 0x7f8f09691000 r--    0x00000000a000 -      -      -          ␣
↪   -
1074     0x7f8f09691000 0x7f8f09692000 rw-    0x00000000b000 -      -      -          ␣
↪   -
[...]
```

zsh (Zsh)

Extracts the zsh command history, similar to the existing bash plugin.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
main_arena	IntParser	The main_arena pointer either extracted from the statically linked ELF binary or from the libc library.
mal-loc_par	IntParser	The malloc_par pointer either extracted from the linked ELF binary or from the libc library.
method	ChoiceAr-ray	Method to list processes (Default uses all methods).
pids	ArrayInt-Parser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
task	ArrayInt-Parser	Kernel addresses of task structs.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

2.1.3 OSX

check_trap_table (CheckTrapTable)

Checks the traps table for hooks.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

address_resolver (DarwinAddressResolver)

A Darwin specific address resolver plugin.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
symbol	ArrayString	List of symbols to lookup
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

allproc (DarwinAllProcCollector)

A mixin for plugins which require a valid kernel address space.

Args: dtb: A potential dtb to be used.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

arp (DarwinArp)

Show information about arp tables.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

boot_cmdline (DarwinBootParameters)

Prints the kernel command line.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

check_syscalls (DarwinCheckSysCalls)

Checks the syscall table.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

dmesg (DarwinDMSG)

Print the kernel debug messages.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

dead_fileprocs (DarwinDeadFileprocCollector)

A mixin for plugins which require a valid kernel address space.

Args: dtb: A potential dtb to be used.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

dead_procs (DarwinDeadProcessCollector)

Lists dead processes using the proc allocation zone.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

dumpcompressedmemory (DarwinDumpCompressedPages)

Dumps all compressed pages.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
dump_dir	String	Path suitable for dumping files.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

dump_zone (DarwinDumpZone)

Dumps an allocation zone’s contents.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

find_dtb (DarwinFindDTB)

Tries to find the DTB address for the Darwin/XNU kernel.

As the XNU kernel developed over the years, the best way of deriving this information changed. This class now offers multiple methods of finding the DTB. Calling find_dtb should automatically select the best method for the job, based on the profile. It will also attempt to fall back on less ideal ways of getting the DTB if the best way fails.

find_kaslr (DarwinFindKASLR)

A scanner for KASLR slide values in the Darwin kernel.

The scanner works by looking up a known data structure and comparing its actual location to its expected location. Verification is a similar process, using a second constant. This takes advantage of the fact that both data structures are in a region of kernel memory that maps to the physical memory in a predictable way (see ID_MAP_VTOP).

Human-readable output includes values of the kernel version string (which is used for validation) for manual review, in case there are false positives.

handles (DarwinHandles)

Walks open files of each proc and collects the fileproc.

This is the same algorithm as lsof, but aimed at just collecting the fileprocs, without doing anything with them, or sorting.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
pids	ArrayIntParser	One or more pids of processes to select.
proc	ArrayIntParser	Kernel addresses of proc structs.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

ip_filters (DarwinIPFilters)

Check IP Filters for hooks.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

ifconfig (DarwinIfnetCollector)

A mixin for plugins which require a valid kernel address space.

Args: dtb: A potential dtb to be used.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

lsmod (DarwinLsmod)

Lists all kernel modules.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

lsof (DarwinLsof)

Walks open files of each proc in order and prints PID, FD and the handle.

Each process has an array of pointers to fileproc structs - the offset into the array is the file descriptor and each fileproc struct represents a handle on some resource. A type field in the fileproc determines the type of the resource pointed to from the fileproc (e.g. vnode, socket, pipe. . .).

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

machine_info (DarwinMachineInfo)

Show information about this machine.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

maps (DarwinMaps)

Display the process maps.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
pids	ArrayIntParser	One or more pids of processes to select.
proc	ArrayIntParser	Kernel addresses of proc structs.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

memdump (DarwinMemDump)

Dumps the memory map for darwin tasks.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
dump_dir	String	Path suitable for dumping files.
pids	ArrayIntParser	One or more pids of processes to select.
proc	ArrayIntParser	Kernel addresses of proc structs.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

memmap (DarwinMemMap)

Prints the memory map for darwin tasks.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
pids	ArrayIntParser	One or more pids of processes to select.
proc	ArrayIntParser	Kernel addresses of proc structs.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

mount (DarwinMount)

Show mount points.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

netstat (DarwinNetstat)

Prints all open sockets we know about, from any source.

Netstat will display even connections that lsof doesn't know about, because they were either recovered from an allocation zone, or found through a secondary mechanism (like system call handler cache).

On the other hand, netstat doesn't know the file descriptor or, really, the process that owns the connection (although it does know the PID of the last process to access the socket.)

Netstat will also tell you, in the style of psxview, if a socket was only found using some of the methods available.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

notifiers (DarwinNotifiers)

Detects hooks in I/O Kit IONotify objects.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

psaux (DarwinPSAUX)

List processes with their commandline.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
pids	ArrayIntParser	One or more pids of processes to select.
proc	ArrayIntParser	Kernel addresses of proc structs.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

pas2vas (DarwinPas2Vas)

Resolves a physical address to a virtual addrress in a process.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
offsets	ArrayIntParser	A list of physical offsets to resolve.
pids	ArrayIntParser	One or more pids of processes to select.
proc	ArrayIntParser	Kernel addresses of proc structs.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

pgrphash (DarwinPgrpHashCollector)

A mixin for plugins which require a valid kernel address space.

Args: dtb: A potential dtb to be used.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

phys_map (DarwinPhysicalMap)

Prints the EFI boot physical memory map.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

pidhash (DarwinPidHashProcessCollector)

A mixin for plugins which require a valid kernel address space.

Args: dtb: A potential dtb to be used.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

pstree (DarwinPsTree)

A mixin for plugins which require a valid kernel address space.

Args: dtb: A potential dtb to be used.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

pslist (DarwinPslist)

A mixin for plugins which require a valid kernel address space.

Args: dtb: A potential dtb to be used.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
pids	ArrayIntParser	One or more pids of processes to select.
proc	ArrayIntParser	Kernel addresses of proc structs.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

psxview (DarwinPsxView)

A mixin for plugins which require a valid kernel address space.

Args: dtb: A potential dtb to be used.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
pids	ArrayIntParser	One or more pids of processes to select.
proc	ArrayIntParser	Kernel addresses of proc structs.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

route (DarwinRoute)

Show routing table.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

dead_sessions (DarwinSessionZoneCollector)

A mixin for plugins which require a valid kernel address space.

Args: dtb: A potential dtb to be used.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

sessions (DarwinSessions)

Finds sessions by walking their global hashtable.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

cc (DarwinSetProcessContext)

A cc plugin for windows.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
pids	ArrayIntParser	One or more pids of processes to select.
proc	ArrayIntParser	Kernel addresses of proc structs.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

sigscan (DarwinSigScan)

Runs a signature scans against physical, kernel or process memory.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
pids	ArrayIntParser	One or more pids of processes to select.
proc	ArrayIntParser	Kernel addresses of proc structs.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

dead_sockets (DarwinSocketZoneCollector)

A mixin for plugins which require a valid kernel address space.

Args: dtb: A potential dtb to be used.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

open_sockets (DarwinSocketsFromHandles)

Looks up handles that point to a socket and collects the socket.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

sysctl (DarwinSysctl)

Dumps the sysctl database.

On OSX the kernel is configured through the sysctl mechanism. This is analogous to /proc or /sysfs on Linux. The configuration space is broken into MIBs - or hierarchical namespace.

https://developer.apple.com/library/mac/documentation/Darwin/Reference/ManPages/man8/sysctl.8.html

For example:

net.inet.ip.subnets_are_local net.inet.ip.ttl net.inet.ip.use_route_genid

This is implemented via a singly linked list of sysctl_oid structs. The structs can be on the following types:

- CTLTYPE_INT means this MIB will handle an int.
- CTLTYPE_STRING means this MIB will handle a string.
- CTLTYPE_QUAD means this MIB will handle a long long int.
- CTLTYPE_NODE means this is a node which handles a sublevel of MIBs. It is actually a pointer to a new sysctl_oid_list which handles the sublevel.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

dead_ttys (DarwinTTYZoneCollector)

A mixin for plugins which require a valid kernel address space.

Args: dtb: A potential dtb to be used.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

tasks (DarwinTaskProcessCollector)

A mixin for plugins which require a valid kernel address space.

Args: dtb: A potential dtb to be used.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

terminals (DarwinTerminals)

Lists open ttys.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

unp_sockets (DarwinUnpListCollector)

Walks the global list of sockets in uipc_usrreq.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

vadmap (DarwinVADMap)

Inspect each page in the VAD and report its status.

This allows us to see the address translation status of each page in the VAD.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
end	IntParser	Stop reading at this offset.
pids	ArrayIntParser	One or more pids of processes to select.
proc	ArrayIntParser	Kernel addresses of proc structs.
proc_regex	RegEx	A regex to select a process by name.
start	IntParser	Start reading from this page.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

vaddump (DarwinVadDump)

Dump the VMA memory for a process.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
dump_dir	String	Path suitable for dumping files.
pids	ArrayIntParser	One or more pids of processes to select.
proc	ArrayIntParser	Kernel addresses of proc structs.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

vtop (DarwinVtoP)

Describe virtual to physical translation on darwin platforms.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
pids	ArrayIntParser	One or more pids of processes to select.
proc	ArrayIntParser	Kernel addresses of proc structs.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

yarascan (DarwinYaraScan)

Scan using yara signatures.

Plugin	Type	Description
binary_string	String	A binary string (encoded as hex) to search for. e.g. 000102[1-200]0506
context	IntParser	Context to print after the hit.
dtb	IntParser	The DTB physical address.
hits	IntParser	Quit after finding this many hits.
pids	ArrayInt-Parser	One or more pids of processes to select.
pre_context	IntParser	Context to print before the hit.
proc	ArrayInt-Parser	Kernel addresses of proc structs.
proc_regex	RegEx	A regex to select a process by name.
scan_kernel	Boolean	Scan the entire kernel address space.
scan_physical	Boolean	Scan the physical address space only.
scan_process_memory	Boolean	Scan all of process memory. Uses process selectors to narrow down selections.
string	String	A verbatim string to search for.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.
yara_expression	String	If provided we scan for this yara expression.
yara_file	String	The yara signature file to read.

zones (DarwinZoneCollector)

A mixin for plugins which require a valid kernel address space.

Args: dtb: A potential dtb to be used.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

dead_vnodes (DarwinZoneVnodeCollector)

A mixin for plugins which require a valid kernel address space.

Args: dtb: A potential dtb to be used.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

2.2 Live

2.2.1 General

file_yara (FileYaraScanner)

Yara scanner which operates on files.

Plugin	Type	Description
binary_string	String	A binary string (encoded as hex) to search for. e.g. 000102[1-200]0506
context	IntParser	Context to print after the hit.
hits	IntParser	Quit after finding this many hits.
paths	Array	Paths to scan.
pre_context	IntParser	Context to print before the hit.
string	String	A verbatim string to search for.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.
yara_expression	String	If provided we scan for this yara expression.
yara_file	String	The yara signature file to read.

hexdump_file (IRDump)

Hexdump files from disk.

Plugin	Type	Description
case_insensitive	Bool	Globs will be case insensitive.
filesystem	Choices	The virtual filesystem implementation to glob in.
globs	ArrayString	List of globs to return.
length	IntParser	Maximum length to dump.
path_sep	String	Path separator character (/ or)
root	String	Root directory to glob from.
rows	IntParser	Number of bytes per row
start	IntParser	An offset to hexdump.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.
width	IntParser	Number of bytes per row

find (IRFind)

List files recursively from a root path.

Plugin	Type	Description
root	String	The root directory to start search from.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

glob (IRGlob)

Search for files by filename glob.

This code roughly based on the Glob flow in GRR.

Plugin	Type	Description
case_insensitive	Bool	Globs will be case insensitive.
filesystem	Choices	The virtual filesystem implementation to glob in.
globs	ArrayString	List of globs to return.
path_sep	String	Path separator character (/ or)
root	String	Root directory to glob from.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

hash (IRHash)

Plugin	Type	Description
hash	ChoiceArray	One or more hashes to calculate.
paths	Array	Paths to hash.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

stat (IRStat)

Plugin	Type	Description
paths	Array	Paths to stat.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

osquery (OSQuery)

Runs the OSQuery query and emit the results.

Note that the columns emitted depend on osquery itself so we can not predict in advance the table format.

Plugin	Type	Description
osquery_path	String	The path to the osquery binary (default osqueryi).
query	String	The OSQuery query to run.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

system_info (SystemInfo)

Just emit information about the agent.

The output format is essentially key value pairs. This is useful for efilter queries.

Plugin	Type	Description
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

2.2.2 API

lsof (APILsof)

A plugin which lists all open files.

Plugin	Type	Description
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

pslist (APIPslist)

A live pslist plugin using the APIs.

Plugin	Type	Description
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

cc (APISetProcessContext)

A cc plugin for setting process context to live mode.

Plugin	Type	Description
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

maps (IRMaps)

Examine the process memory maps.

Plugin	Type	Description
offset	SymbolAddress	Only print the vad corresponding to this offset.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
regex	RegEx	A regular expression to filter VAD filenames.
verbosity	IntParser	With high verbosity print more information on each region.

vaddump (IRVadDump)

Dump the VMA memory for a process.

Plugin	Type	Description
dump_dir	String	Path suitable for dumping files.
offset	SymbolAddress	Only print the vad corresponding to this offset.
pids	ArrayIntParser	One or more pids of processes to select.
proc_regex	RegEx	A regex to select a process by name.
regex	RegEx	A regular expression to filter VAD filenames.
verbosity	IntParser	With high verbosity print more information on each region.

address_resolver (LinuxAPIAddressResolver)

A Linux specific address resolver plugin.

Plugin	Type	Description
symbol	ArrayString	List of symbols to lookup
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

yarascan (ProcessYaraScanner)

Yara scan process memory using the ReadProcessMemory() API.

Plugin	Type	Description
binary_string	String	A binary string (encoded as hex) to search for. e.g. 000102[1-200]0506
context	IntParser	Context to print after the hit.
hits	IntParser	Quit after finding this many hits.
pids	ArrayIntParser	One or more pids of processes to select.
pre_context	IntParser	Context to print before the hit.
proc_regex	RegEx	A regex to select a process by name.
string	String	A verbatim string to search for.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.
yara_expression	String	If provided we scan for this yara expression.
yara_file	String	The yara signature file to read.

2.3 Filesystem

2.3.1 NTFS

fls (FLS)

Plugin	Type	Description
path	String	Path to print stats for.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

fstat (FStat)

Print information by filename.

Plugin	Type	Description
path	String	Path to print stats for.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

idump (IDump)

Dump a part of an MFT file.

Plugin	Type	Description
id	IntParser	Id of attribute to dump.
mft	IntParser	MFT entry to dump.
type	IntParser	Attribute type to dump.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

iexport (IExport)

Extracts files from NTFS.

For each specified MFT entry, dump the file to the specified dump directory. The filename is taken as the longest filename of this MFT entry.

Plugin	Type	Description
dump_dir	String	Path suitable for dumping files.
id	IntParser	Id of attribute to dump.
mft	IntParser	MFT entry to dump.
type	IntParser	Attribute type to dump.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

ils (ILS)

List files in an NTFS image.

Plugin	Type	Description
mfts	ArrayIntParser	MFT entries to list.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

istat (IStat)

Print information related to an MFT entry.

Plugin	Type	Description
mfts	ArrayIntParser	MFT entries to list.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

2.4 General

2.4.1 Utilities

aff4acquire (AFF4Acquire)

Copy the physical address space to an AFF4 file.

NOTE: This plugin does not require a working profile - unless the user also wants to copy the pagefile or mapped files. In that case we must analyze the live memory to gather the required files.

Plugin	Type	Description
also_mapped_files	Boolean	Also get mapped or opened files (requires a profile)
also_memory	Boolean	Also acquire physical memory. If not specified we acquire physical memory only when no other operation is specified.
also_pagefile	Boolean	Also get the pagefile/swap partition (requires a profile)
append	Boolean	Append to the current volume.
compression	String	The compression to use.
destination	String	The destination file to create.
destination_url	String	The destination AFF4 URL to create.
files	ArrayStringParser	Also acquire files matching the following globs.
gce_credentials	String	The GCE service account credentials to use.
gce_credentials_path	String	A path to the GCE service account credentials to use.
max_file_size	IntParser	Maximum file size to acquire.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

aff4dump (AFF4Dump)

Dump the entire resolver contents for an AFF4 volume.

Plugin	Type	Description
gce_credentials	String	The GCE service account credentials to use.
gce_credentials_path	String	A path to the GCE service account credentials to use.
long	Boolean	Include additional information about each stream.
regex	RegEx	Regex of filenames to dump.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.
volume	String	Volume to list.

aff4export (AFF4Export)

Exports all the streams in an AFF4 Volume.

Plugin	Type	Description
dump_dir	String	Path suitable for dumping files.
gce_credentials	String	The GCE service account credentials to use.
gce_credentials_path	String	A path to the GCE service account credentials to use.
regex	RegEx	Regex of filenames to dump.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.
volume	String	Volume to list.

aff4ls (AFF4Ls)

List the content of an AFF4 file.

Plugin	Type	Description
gce_credentials	String	The GCE service account credentials to use.
gce_credentials_path	String	A path to the GCE service account credentials to use.
long	Boolean	Include additional information about each stream.
regex	RegEx	Regex of filenames to dump.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.
volume	String	Volume to list.

api (APIGenerator)

Generate the plugin API document.

Plugin	Type	Description
output_file	String	If specified we write the API into this file in YAML.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

session_api (APISessionGenerator)

Plugin	Type	Description
output_file	String	If specified we write the API into this file in YAML.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

artifact_collector (ArtifactsCollector)

Collects artifacts.

Plugin	Type	Description
artifact_files	ArrayStringParser	A list of additional yaml files to load which contain artifact definitions.
artifacts	ArrayStringParser	A list of artifact names to collect.
copy_files	Bool	Copy files into the output.
create_timeline	Bool	Also generate a timeline file.
definitions	ArrayStringParser	An inline artifact definition in yaml format.
output_path	String	Path suitable for dumping files.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.
writer	Choices	Writer for artifact results.

artifact_list (ArtifactsList)

List details about all known artifacts.

Plugin	Type	Description
all	Bool	Show all artifacts.
labels	ArrayString-Parser	Filter by these labels.
regex	RegEx	Filter the artifact name.
sup-ported_os	ArrayString-Parser	If specified show for these OSs, otherwise autodetect based on the current image.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

artifact_view (ArtifactsView)

Plugin	Type	Description
artifacts	ArrayStringParser	A list of artifacts to display
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

build_index (BuildIndex)

Generate a profile index file based on an index specification.

The index specification is currently a yaml file with the following structure:

```
base_symbol: (string) # OPTIONAL Compute ALL offsets as relative to this
  symbol. This includes MaxOffset and MinOffset.
symbols: (array of dicts) # A list of symbols to index.
  -
  name: (string) # Symbol name
  data: (string) # Data that should be at the symbol's offset
  shift: (int) # OPTIONAL Adjust symbol offset by this number
```

Example:

```
path: win32k.sys
symbols:
  -
    # The name of the symbol we test for.
    name: "??_C@_1BO@KLKIFHLC@?$AAG?$AAU?$AAI?$AAF?$AAo?$AAn?$AAt?$AA?4?$AAH?$AAe?
↪$AAi?$AAg?$AAh?$AAt?$AA?$AA@"

    # The data we expect to find at that offset.
    data: "47005500490046006f006e0074002e00480065006900670068007400"

  -
    name: "wcschr"
    shift: -1
    data: "90"
```

The result is an index profile. This has an $INDEX section which is a dict, with keys being the profile name, and values being a list of (offset, match) tuples. For example:

```
{
 "$INDEX": {
   "tcpip.sys/AMD64/6.0.6001.18000/0C1A1EC1D61E4508A33F5212FC1B37202": [[1184600,
     "495053656344656c657465496e626f756e644f7574626f756e64536150616972"]],
   "tcpip.sys/AMD64/6.0.6001.18493/29A4DBCAF840463298F40190DD1492D02": [[1190376,
     "495053656344656c657465496e626f756e644f7574626f756e64536150616972"]],
   "tcpip.sys/AMD64/6.0.6002.18272/7E79532FC7E349C690F5FBD16E3562172": [[1194296,
     "495053656344656c657465496e626f756e644f7574626f756e64536150616972"]],
   },
   "$METADATA": {
      "ProfileClass": "Index",
      "Type": "Profile",
      "MaxOffset": 546567,
      "MinOffset": 0
    }
}
```

For example:

```
{
 "$INDEX": {
  "tcpip.sys/AMD64/6.0.6001.18000/0C1A1EC1D61E4508A33F5212FC1B37202": [[1184600,
↪"495053656344656c657465496e626f756e644f7574626f756e64536150616972"]],
  "tcpip.sys/AMD64/6.0.6001.18493/29A4DBCAF840463298F40190DD1492D02": [[1190376,
↪"495053656344656c657465496e626f756e644f7574626f756e64536150616972"]],
  "tcpip.sys/AMD64/6.0.6002.18272/7E79532FC7E349C690F5FBD16E3562172": [[1194296,
↪"495053656344656c657465496e626f756e644f7574626f756e64536150616972"]],
 "$METADATA": {
  "ProfileClass": "Index",
  "Type": "Profile"
  }
 }
```

build_local_profile (BuildProfileLocally)

Download and builds a profile locally in one step.

We store the profile in the first repository in the profile_path which must be writable. Usually this is a caching repository so the profile goes in the local cache.

simple_certdump (CertDump)

Dump certs found by cert scan.

Plugin	Type	Description
dump_dir	String	Path suitable for dumping files.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

simple_certscan (CertScan)

Dump RSA private and public SSL keys from the physical address space.

Plugin	Type	Description
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

This plugin is similar to the [cert_vad_scan](CertVadScan.html) plugin. It attempts to detect DER encoded X509 certificates or RSA private keys in physical memory.

Optionally, if a dump directory is provided the DER encoded certificates are also dumped to files in the specified directory.

### Sample Output

```
win8.1.raw 22:07:35> certscan
-------------------> certscan()
   Address      Type       Length     Description
-------------- ---------- ---------- -----------
0x000000030c95 X509       1287       /C=US/ST=Washington/L=Redmond/O=Microsoft␣
↪Corporation/CN=Microsoft Windows
0x00000003119c X509       1499       /C=US/ST=Washington/L=Redmond/O=Microsoft␣
↪Corporation/CN=Microsoft Windows Production PCA 2011
0x000000031b94 X509       1653       /C=US/ST=Washington/L=Redmond/O=Microsoft␣
↪Corporation/CN=Microsoft Time-Stamp PCA 2010
0x000000032209 X509       1246       /C=US/ST=Washington/L=Redmond/O=Microsoft␣
↪Corporation/OU=MOPR/OU=nCipher DSE ESN:F528-3777-8A76/CN=Microsoft Time-Stamp␣
↪Service
0x00000017114e X509       1499       /C=US/ST=Washington/L=Redmond/O=Microsoft␣
↪Corporation/CN=Microsoft Windows Production PCA 2011
0x000000171b46 X509       1653       /C=US/ST=Washington/L=Redmond/O=Microsoft␣
↪Corporation/CN=Microsoft Time-Stamp PCA 2010
```

collect (Collect)

Collect instances of struct of type 'type_name'. This plugin will find all other plugins that produce 'type_name' and merge all their output. For example, running collect 'proc' will give you a rudimentary psxview. This plugin is mostly used by other plugins, like netstat and psxview.

Plugin	Type	Description
type_name	String	The type (struct) to collect.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

convert_profile (ConvertProfile)

Convert a profile from another program to the Rekall format.

The Rekall profile format is optimized for loading at runtime. This plugin produces a Rekall profile from a variety of sources, including:

- Linux debug compiled kernel module (see tool/linux/README)
- OSX Dwarfdump outputs.

Plugin	Type	Description
converter	String	The name of the converter to use. If not specified autoguess.
out_file	String	Path for output file.
profile_class	String	The name of the profile implementation to specify. If not specified, we autodetect.
source	String	Filename of profile to read.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

Rekall profiles are JSON files which contain information specific to a particular software version. For example, Rekall requires a Linux Kernel profile to be able to analyze a memory image of the Linux kernel.

The *convert_profile* plugin converts profiles other formats to the standard JSON format used by Rekall. There are two main use cases:

1. If you have an old Volatility profile, this plugin will parse that.
2. When building a Linux kernel profile, the build system produces a debug enabled kernel module inside a Zip file. In this case you can use the *convert_profile* plugin to parse the DWARF stream from the debug module and produce the JSON file required.

The below example demonstrates how to build and convert a Linux profile locally for live analysis:

```
rekall/tools/linux# make profile
make -C /usr/src/linux-headers-3.13.0-74-generic CONFIG_DEBUG_INFO=y M=`pwd` modules
make[1]: Entering directory `/usr/src/linux-headers-3.13.0-74-generic'
  Building modules, stage 2.
    MODPOST 2 modules
    make[1]: Leaving directory `/usr/src/linux-headers-3.13.0-74-generic'
    cp module.ko module_dwarf.ko
    zip "3.13.0-74-generic.zip" module_dwarf.ko /boot/System.map-3.13.0-74-generic /
↪boot/config-3.13.0-74-generic
    updating: module_dwarf.ko (deflated 65%)
    updating: boot/System.map-3.13.0-74-generic (deflated 79%)
    updating: boot/config-3.13.0-74-generic (deflated 75%)

rekall/tools/linux# rekal convert_profile 3.13.0-74-generic.zip 3.13.0-74-generic.json
rekall/tools/linux# rekal --profile 3.13.0-74-generic.json -f /proc/kcore pslist
     task_struct           Name           PID    PPID   UID    GID         DTB        ␣
↪     Start Time        Binary
     -------------- -------------------- ------ ------ ------ ------ -------------- --
↪---------------------- ------
     0x8804285f0000 init                      1      0      0      0 0x000426592000 ␣
↪   2016-01-29 12:50:31Z /sbin/init
     0x8804285f1800 kthreadd                  2      0      0      0 -              ␣
↪   2016-01-29 12:50:31Z -
     0x8804285f3000 ksoftirqd/0               3      2      0      0 -              ␣
↪   2016-01-29 12:50:31Z -
```

dt (DT)

Print a struct or other symbol.

Really just a convenience function for instantiating the object and printing all its members.

Plugin	Type	Description
address_space	AddressSpace	The address space to use.
member_offset	IntParser	If specified we only show the member at this offset.
offset	IntParser	Name of a struct definition.
target	String	Name of a struct definition.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The *dt* plugin prints all the fields within a data structure and optionally, their contents.

In the example below, we create an *_EPROCESS* instance over a specific virtual address (this was taken from the output of the *pslist* plugin). The *dt* plugin displays all the fields in the struct. If there is a nested struct, the *dt* plugin shows a tree view of the nested struct as well.

Note that if an address is not specified, the *_EPROCESS* object will simply be instantiated over address 0 and all offsets will be relative to the begining of the struct. This is very useful when deciphering assembly code which dereferences members of the struct.

Rekall also uses “virtual members” on structs, mostly placed there for convenience or to support multiple versions of the same struct. We can see in this case that the fields “name” and “pid” are virtual members since their offset is -1. These represent the name and the pid of the process in all operating systems.

```
[1] win7.elf 19:34:27> dt session.profile._EPROCESS(0xfa8002a94060)
---------------------> dt(session.profile._EPROCESS(0xfa8002a94060))
[_EPROCESS _EPROCESS] @ 0xfa8002a94060
Offset                   Field                    Content
------ ------------------------------ -------
            0x-1    RealVadRoot                         [_MMADDRESS_NODE BalancedRoot] @␣
↪0xFA8002A944A8
. 0xfa8002a9449c    Tag                                  [String:Tag]: '\x14\xd0\x02\x00'
. 0xfa8002a944a8    u1                                  [<unnamed-5580> u1] @␣
↪0xFA8002A944A8
.. 0xfa8002a944a8    Balance                              [BitField(0-2):Balance]:␣
↪0x00000000
.. 0xfa8002a944a8    Parent                              <_MMADDRESS_NODE Pointer to␣
↪[0xFA8002A944A8] (Parent)>
. 0xfa8002a944b0    LeftChild                           <_MMADDRESS_NODE Pointer to␣
↪[0x00000000] (LeftChild)>
. 0xfa8002a944b8    RightChild                          <_MMADDRESS_NODE Pointer to␣
↪[0xFA8002A92710] (RightChild)>
. 0xfa8002a944c0    StartingVpn                          [unsigned long long:StartingVpn]:␣
↪0x00000000
. 0xfa8002a944c8    EndingVpn                            [unsigned long long:EndingVpn]:␣
↪0x00000000
            0x-1    dtb                                 112128000
            0x-1    name                                 [String:ImageFileName]: 'Console.
↪exe\x00'
            0x-1    pid                                  [unsigned int:UniqueProcessId]:␣
↪0x00000A38
  0xfa8002a94060    Pcb                                 [_KPROCESS Pcb] @ 0xFA8002A94060
. 0xfa8002a94060    Header                              [_DISPATCHER_HEADER Header] @␣
↪0xFA8002A94060
.. 0xfa8002a94060    Lock                                 [long:Lock]: 0x00580003
.. 0xfa8002a94060    Type                                 [Enumeration:Type]: 0x00000003␣
↪(ProcessObject)
.. 0xfa8002a94061    Abandoned                            [unsigned char:Abandoned]:␣
↪0x00000000
.. 0xfa8002a94061    Absolute                             [BitField(0-1):Absolute]:␣
↪0x00000000
.. 0xfa8002a94061    Coalescable                          [BitField(1-2):Coalescable]:␣
↪0x00000000
.. 0xfa8002a94061    EncodedTolerableDelay                [BitField(3-
↪8):EncodedTolerableDelay]: 0x00000000
.. 0xfa8002a94061    KeepShifting                         [BitField(2-3):KeepShifting]:␣
↪0x00000000
.. 0xfa8002a94061    Signalling                           [unsigned char:Signalling]:␣
↪0x00000000
.. 0xfa8002a94061    TimerControlFlags                    [unsigned␣
↪char:TimerControlFlags]: 0x00000000
.. 0xfa8002a94062    CounterProfiling                     [BitField(2-
↪3):CounterProfiling]: 0x00000000
.. 0xfa8002a94062    CpuThrottled                         [BitField(0-1):CpuThrottled]:␣
↪0x00000000
```

```
.. 0xfa8002a94062    CycleProfiling                   [BitField(1-2):CycleProfiling]:␣
↪0x00000000
.. 0xfa8002a94062    Hand                             [unsigned char:Hand]: 0x00000058
.. 0xfa8002a94062    Reserved                         [BitField(3-8):Reserved]:␣
↪0x0000000B
.. 0xfa8002a94062    Size                             [unsigned char:Size]: 0x00000058
.. 0xfa8002a94062    ThreadControlFlags               [unsigned␣
↪char:ThreadControlFlags]: 0x00000058
.. 0xfa8002a94063    ActiveDR7                        [BitField(0-1):ActiveDR7]:␣
↪0x00000000
.. 0xfa8002a94063    DebugActive                      [unsigned char:DebugActive]:␣
↪0x00000000
.. 0xfa8002a94063    DpcActive                        [unsigned char:DpcActive]:␣
↪0x00000000
.. 0xfa8002a94063    Expired                          [BitField(7-8):Expired]:␣
↪0x00000000
```

describe (Describe)

Describe the output of a plugin.

Plugin	Type	Description
args	dict	args to run the plugin with.
max_depth	IntParser	The maximum depth to follow mappings.
plugin_name	String	A plugin or plugin name to describe.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

dis (Disassemble)

Disassemble the given offset.

Plugin	Type	Description
address_space	AddressSpace	The address space to use.
branch	Boolean	If set we follow all branches to cover all code.
canonical	Boolean	If set emit canonical instructions. These can be used to develop signatures.
end	IntParser	The end address to disassemble up to.
length	IntParser	The number of instructions (lines) to disassemble.
mode	Choices	Disassemble Mode (AMD64 or I386). Defaults to 'auto'.
offset	SymbolAddress	An offset to disassemble. This can also be the name of a symbol with an optional offset. For example: tcpip!TcpCovetNetBufferList.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

This plugin is used to disassemble memory regions. The offset to disassemble may be given as:

- An address in the current default address space (See the [cc](SetProcessContext.html) plugin for an explaination of the default address space).
- The name of a kernel module with an optional symbol name. The symbol may be an experted symbol, or non-exported symbol as defined in the pdb file for that kernel module.

### Notes

1. When using the interactive console you can complete symbol names by double tapping the [tab] key. For example **dis "nt!KiSetTi[tab][tab]**.
2. Rekall attempts to resolve addresses in the disassembly back to known symbol names. Additionally, for indirect operations, Rekall also prints the current value of the memory location. This feature is especially useful for understanding where indirect jumps are going - without needing to consider PE import tables etc. This works since the IAT is already patched into memory, hence Rekall can completely ignore IAT resoltion (unlike a standalone PE analyser like IDA).

### Sample output

Here we disassemble the kernel function **KiSetTimerEx** to observe the DPC pointer obfuscation that Patch Guard uses on 64 bit Windows 7. We can see the names of the symbols used and their current values, as well as the name of internally called functions.

```
win7.elf 23:48:14> dis "nt!KiSetTimerEx"
-----------------> dis("nt!KiSetTimerEx")
   Address       Rel Op Codes             Instruction                    Comment
-------------- ---- -------------------- ------------------------------ -------
------ nt!KiSetTimerEx ------
0xf8000269d4f0    0 48895c2408           MOV [RSP+0x8], RBX
0xf8000269d4f5    5 4889542410           MOV [RSP+0x10], RDX
0xf8000269d4fa    A 55                   PUSH RBP
0xf8000269d4fb    B 56                   PUSH RSI
0xf8000269d4fc    C 57                   PUSH RDI
0xf8000269d4fd    D 4154                 PUSH R12
0xf8000269d4ff    F 4155                 PUSH R13
0xf8000269d501   11 4156                 PUSH R14
0xf8000269d503   13 4157                 PUSH R15
0xf8000269d505   15 4883ec50             SUB RSP, 0x50
0xf8000269d509   19 488b05f09b2200       MOV RAX, [RIP+0x229bf0]        ␣
↪0x6D7CFFA404933FBB nt!KiWaitNever
0xf8000269d510   20 488b1dc19c2200       MOV RBX, [RIP+0x229cc1]        ␣
↪0x933DD660CFFF8004 nt!KiWaitAlways
0xf8000269d517   27 4c8bb424b0000000     MOV R14, [RSP+0xb0]
0xf8000269d51f   2F 4933de               XOR RBX, R14
0xf8000269d522   32 488bf1               MOV RSI, RCX
0xf8000269d525   35 450fb6f9             MOVZX R15D, R9B
0xf8000269d529   39 480fcb               BSWAP RBX
0xf8000269d52c   3C 418bf8               MOV EDI, R8D
0xf8000269d52f   3F 4833d9               XOR RBX, RCX
0xf8000269d532   42 8bc8                 MOV ECX, EAX
0xf8000269d534   44 48d3cb               ROR RBX, CL
0xf8000269d537   47 4833d8               XOR RBX, RAX
0xf8000269d53a   4A 450f20c4             MOV R12, CR8
0xf8000269d53e   4E b802000000           MOV EAX, 0x2
0xf8000269d543   53 440f22c0             MOV CR8, RAX
0xf8000269d547   57 65488b2c2520000000   MOV RBP, [GS:0x20]
0xf8000269d550   60 33d2                 XOR EDX, EDX
0xf8000269d552   62 488bce               MOV RCX, RSI
0xf8000269d555   65 e8f6b0ffff           CALL 0xf80002698650            nt!
↪KiCancelTimer
0xf8000269d55a   6A 48895e30             MOV [RSI+0x30], RBX
```

dump (Dump)

Hexdump an object or memory location.

You can use this plugin repeateadely to keep dumping more data using the "p _" (print last result) operation:

In [2]: dump 0x814b13b0, address_space="K" ——–> dump(0x814b13b0, address_space="K") Offset Hex Data ———- ———————————————— —————- 0x814b13b0 03 00 1b 00 00 00 00 00 b8 13 4b 81 b8 13 4b 81 .........K...K.

Out[3]: <rekall.plugins.core.Dump at 0x2967510>

In [4]: p _ ——–> p(_) Offset Hex Data ———- ———————————————— —————- 0x814b1440 70 39 00 00 54 1b 01 00 18 0a 00 00 32 59 00 00 p9..T.......2Y.. 0x814b1450 6c 3c 01 00 81 0a 00 00 18 0a 00 00 00 b0 0f 06 l<.............. 0x814b1460 00 10 3f 05 64 77 ed 81 d4 80 21 82 00 00 00 00 ..?.dw....!.....

Plugin	Type	Description
address_space	AddressSpace	The address space to use.
data	String	Dump this string instead.
length	IntParser	Maximum length to dump.
offset	SymbolAddress	An offset to hexdump.
rows	IntParser	Number of rows to dump
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.
width	IntParser	Number of bytes per row

If you need to produce a hexdump of a region of memory, use the *dump* plugin. This plugin accepts a single symbol name or address in the default address space (see the *cc* plugin).

The *dump* plugin will also show which symbol address is known to exist in every offset displayed. This is done via the Rekall address resolver. If colors are enabled, known symbols are highlighted in different colors both in the comment field and inside the hexdump area itself.

In the below example we dump the 'SeTcbPrivilege' symbol from the nt kernel. Also shown are other symbols located in the vicinity.

```
[1] win7.elf 22:32:36> dump "nt!SeTcbPrivilege"
---------------------> dump("nt!SeTcbPrivilege")
Offset                                    Data                                                ␣
↪       Comment
-------------- ----------------------------------------------------------------- -----
↪-----------------------------------
0xf80002b590b8 07 00 00 00 00 00 00 00 44 02 01 00 80 f9 ff ff  ........D....... nt!
↪SeTcbPrivilege, nt!NlsOemToUnicodeData
0xf80002b590c8 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00  ................ nt!
↪VfRandomVerifiedDrivers, nt!TunnelMaxEntries, nt!ExpBootLicensingData
0xf80002b590d8 bc 00 00 00 00 10 00 00 00 00 ff 07 80 f8 ff ff  ................ nt!
↪ExpLicensingDescriptorsCount, nt!CmpStashBufferSize, nt!ExpLicensingView
0xf80002b590e8 e8 f5 00 00 a0 f8 ff ff e8 45 7a 05 a0 f8 ff ff  .........Ez..... nt!
↪CmpHiveListHead
0xf80002b590f8 1c 00 00 00 80 f9 ff ff 16 00 00 00 00 00 00 00  ................ nt!
↪NlsAnsiToUnicodeData, nt!SeSystemEnvironmentPrivilege
```

dwarfparser (DwarfParser)

Parse the dwarf file and dump a vtype structure from it.

Plugin	Type	Description
dwarf_filename	String	The filename of the PDB file.
profile_class	String	The name of the profile implementation.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

This plugin is mostly used by the convert_profile plugin.

elf_sections (ELFSections)

Plugin	Type	Description
binary_path	String	Path to the ELF binary.
header_offset	IntParser	Offset to the ELF header.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

elf_versions_needed (ELFVerNeeded)

Plugin	Type	Description
binary_path	String	Path to the ELF binary.
header_offset	IntParser	Offset to the ELF header.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

elf_versions_symbols (ELFVerSymbols)

Plugin	Type	Description
binary_path	String	Path to the ELF binary.
header_offset	IntParser	Offset to the ELF header.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

ewfacquire (EWFAcquire)

Copy the physical address space to an EWF file.

Plugin	Type	Description
destination	String	The destination file to create. If not specified we write output.E01 in current directory.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

Rekall supports many different image formats. One of the popular formats is the EWF or E01 formats. It is a compressible format for forensic images.

The *ewfacquire* plugin will copy the physical address space into an EWF file. This can be used to acquire memory (e.g. when Rekall is used in live mode) or to convert a memory image from another format to EWF format.

Note that the EWF format is not an open format. The variant written by Rekall is not necessarily interchangeable with other implementations. We usually recommend using *aff4acquire* over *ewfacquire* because the AFF4 format can contain multiple streams and can also keep important metadata.

```
[1] win7.elf 23:02:22> ewfacquire destination="/tmp/test.E01"
---------------------> ewfacquire(destination="/tmp/test.E01")
 Writing 352Mb
```

fetch_pdb (FetchPDB)

Fetch the PDB file for an executable from the Microsoft PDB server.

Plugin	Type	Description
dump_dir	String	Path suitable for dumping files.
guid	String	The GUID of the pdb file. If provided, the pdb filename must be provided in the –pdb_filename parameter.
pdb_filename	String	The filename of the executable to get the PDB file for.
verbosity	Int-Parser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The Microsoft Visual Studio compiler stores debugging information for each binary built in a PDB file. Each binary contains a unique GUID which can be used to fetch the correct PDB file from the public Microsoft symbol server.

The *fetch_pdb* plugin is used to fetch the correct PDB file from the symbol server. You will need to provide the name of the PDB file and the GUID - both of these are found from the PE headers of the binary.

Note that this plugin is mainly used by the *build_local_profile* plugin and by the *manage_repo* plugins, but might also be useful on its own. Usually you need to *parse_pdb* after fetching it so a profile can be generated for Rekall to use.

In the example below we find the GUID and pdb file name of an executable from the image, then use the *fetch_pdb* plugin to fetch it. Note that PDB files are compressed using CAB on the symbol server so we need *cabextract* installed locally.

```
[1] win7.elf 23:08:40> peinfo "termdd"
          Attribute                                     Value
------------------------------ -------------------------------------------------------
↪-----
Machine                        IMAGE_FILE_MACHINE_AMD64
TimeDateStamp                  2009-07-14 00:16:36Z
Characteristics                IMAGE_FILE_DLL, IMAGE_FILE_EXECUTABLE_IMAGE,
                               IMAGE_FILE_LARGE_ADDRESS_AWARE
GUID/Age                       2A530717E88549BB92DBB72C224EC2B11
PDB                            termdd.pdb
MajorOperatingSystemVersion    6
MinorOperatingSystemVersion    1
MajorImageVersion              6

....

[1] win7.elf 23:09:12> fetch_pdb pdb_filename="termdd.pdb", guid=
↪"2A530717E88549BB92DBB72C224EC2B11"
 Trying to fetch http://msdl.microsoft.com/download/symbols/termdd.pdb/
↪2A530717E88549BB92DBB72C224EC2B11/termdd.pd_
 Trying to fetch http://msdl.microsoft.com/download/symbols/termdd.pdb/
↪2A530717E88549BB92DBB72C224EC2B11/termdd.pd_
Extracting cabinet: /tmp/tmpXkEgyu/termdd.pd_
  extracting termdd.pdb

All done, no errors.
```

which_plugin (FindPlugins)

Find which plugin(s) are available to produce the desired output.

Plugin	Type	Description
producers_only	Boolean	Only include producers: plugins that output only this struct and have no side effects.
type_name	String	The name of the type we're looking for. E.g.: 'proc' will find psxview, pslist, etc.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

grep (Grep)

Search an address space for keywords.

Plugin	Type	Description
address_space	AddressSpace	Name of the address_space to search.
context	IntParser	Context to print around the hit.
keyword	ArrayString	The binary strings to find.
limit	String	The length of data to search.
offset	IntParser	Start searching from this offset.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

Sometimes we want to search for some data in the address space. Although we can use *yarascan* to do this, it is typically slower than just running the *grep* plugin. Note that the plugin can scan the entire address space efficiently (i.e. it will automatically skip over sparse memory regions).

One of the more interesting uses of the *grep* plugin is looking for references. For example, suppose we wanted to see who has a reference to a particular _EPROCESS structure.

In the below example, we pick an _EPROCESS from the output of *pslist* and search for pointers to it somewhere in kernel memory (There are many pointers! We just picked one for this example.). We then use the *analyze_struct* plugin to discover that the pointer resides in an allocation with the pool tag 'ObHd'. We can search the kernel disassembly to realize this is an Object Handle. Note how we use grep to search for the little endian representation of the _EPROCESS address.

```
[1] win7.elf 23:14:38> pslist
  _EPROCESS            Name          PID   PPID   Thds    Hnds    Sess  Wow64   ␣
↪  Start                     Exit
-------------- -------------------- ----- ------ ------ -------- ------ ------ -------
↪----------------- ------------------------
....
0xfa8002ad0190 cmd.exe              2644   2616      2       66      1 True   2012-
↪10-01 14:40:20Z     -

[1] win7.elf 23:14:55> grep keyword="\x90\x01\xad\x02\x80\xfa"
....
    Offset                                    Data                                 ␣
↪          Comment
-------------- ----------------------------------------------------------------- -----
↪-----------------------------------
0xf8a0013d8ad8 60 40 a9 02 80 fa ff ff 01 00 00 00 00 00 00 00  `@..............
0xf8a0013d8ae8 90 01 ad 02 80 fa ff ff 01 00 00 00 00 00 00 00  ................
0xf8a0013d8af8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
...
[1] win7.elf 23:17:20> analyze_struct 0xf8a0013d8ae8

0xf8a0013d8ae8 is inside pool allocation with tag 'ObHd' (0xf8a0013d8a30)  and size␣
↪0x100
```

```
    Offset     Content
-------------- -------
           0x0 Data:0xfa8002ad0190 Tag:Pro\xe3 @0xfa8002ad0190 (0x530)
           0x8 Data:0x1
          0x10 Data:0x0
          0x18 Data:0x0
          0x20 Data:0x0
          0x28 Data:0x0
          0x30 Data:0xfa80017f9060 Tag:Pro\xe3 @0xfa80017f9060 (0x530)
          0x38 Data:0x1
          0x40 Data:0x730061006c
          0x48 Data:0x744e034d0110
          0x50 Data:0x490053004c
          0x58 Data:0xa4801280702
          0x60 Data:0x981e
          0x68 Data:0x100000000
          0x70 Data:0x0
[1] win7.elf 23:22:25> hex(struct.unpack("<I", 'ObHd')[0])
               Out<24> '0x6448624f'
[1] win7.elf 23:22:33> dis "nt!ObpInsertHandleCount"
----------------------> dis("nt!ObpInsertHandleCount")
Address      Rel             Op Codes                    Instruction                              ␣
↪Comment
------- -------------- -------------------- ---------------------------------------- -
↪------
------ nt!ObpInsertHandleCount ------: 0xf80002976010
  0xf80002976010            0x0 48895c2408           mov qword ptr [rsp + 8], rbx
  0xf80002976015            0x5 48896c2410           mov qword ptr [rsp + 0x10], rbp
....

  0xf80002976089           0x79 41b84f624864         mov r8d, 0x6448624f
  0xf8000297608f           0x7f e83cd3e4ff           call 0xf800027c33d0                      ␣
↪        nt!ExAllocatePoolWithTag
  0xf80002976094           0x84 4885c0               test rax, rax
  0xf80002976097           0x87 0f84dacd0400         je 0xf800029c2e77                        ␣
↪        nt!ExpProfileCreate+0x9d57
  0xf8000297609d           0x8d 458bc5               mov r8d, r13d
```

imagecopy (ImageCopy)

Copies a physical address space out as a raw DD image

Rekall supports many different image formats. Image formats such as AFF4 and EWF are very convenient for long term storage and archiving of images. However, some other memory analysis tools do not support such a rich selection of image formats and might not be able to directly analyze some of these formats.

Sometimes we might want to verify something with another tool, and the RAW image format seems to be most widely supported. The *imagecopy* plugin copies the current physical address space into a RAW file. It pads sparse regions with NULL bytes.

Note that RAW images can not contain multiple streams (like the pagefile), nor do they support any metadata (such as registers). Hence the RAW image is vastly inferior. We do not recommend actually acquiring the image using the RAW format in the first place (use AFF4 or ELF). However, if Rekall is run in live mode, the *imagecopy* plugin will produce a RAW image of live memory.

In the following example we convert an EWF image to raw so Volatility can read it:

```
[1] win7.elf.E01 23:36:57> imagecopy "/tmp/foo.raw"
---------------------> imagecopy("/tmp/foo.raw")
Range 0x0 - 0x2cb00000
Range 0xe0000000 - 0x1000000
Range 0xf0400000 - 0x400000
Range 0xf0800000 - 0x4000
Range 0xffff0000 - 0x10000
               Out<27> Plugin: imagecopy

[1] win7.elf.E01 23:38:06> !python /home/scudette/projects/volatility/vol.py --
↪profile Win7SP1x64 -f /tmp/foo.raw pslist
Volatility Foundation Volatility Framework 2.5
Offset(V)          Name                    PID   PPID   Thds     Hnds   Sess  Wow64␣
↪Start                          Exit
------------------ -------------------- ------ ------ ------ -------- ------ ------ --
↪---------------------------- ------------------------------
0xfffffa80008959e0 System                    4      0     84      511 ------      0␣
↪2012-10-01 21:39:51 UTC+0000
0xfffffa8001994310 smss.exe                272      4      2       29 ------      0␣
↪2012-10-01 21:39:51 UTC+0000
0xfffffa8002259060 csrss.exe               348    340      9      436      0      0␣
↪2012-10-01 21:39:57 UTC+0000
```

info (Info)

Print information about various subsystems.

shell (InteractiveShell)

An interactive shell for Rekall.

json_render (JSONParser)

Renders a json rendering file, as produced by the JsonRenderer.

The output of any plugin can be stored to a JSON file using:

rekall -f img.dd –format json plugin_name –output test.json

Then it can be rendered again using:

rekall json_render test.json

This plugin implements the proper decoding of the JSON encoded output.

Plugin	Type	Description
file	String	The filename to parse.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

l (Lister)

A plugin to list objects.

Sometimes in the interactive console we receive a generator or a list. Use the *l* plugin to quickly print each value in the list.

In the below example we instantiate the PsActiveProcessHeadHook and walk the list of processes (this is one of the *pslist* methods).

```
[1] win7.elf 23:48:12> head = session.profile.get_constant_object("PsActiveProcessHead
↪", "_LIST_ENTRY")

[1] win7.elf 23:48:32> l head.list_of_type("_EPROCESS", "ActiveProcessLinks")
---------------------> l(head.list_of_type("_EPROCESS", "ActiveProcessLinks"))
[_EPROCESS _EPROCESS] @ 0xFA80008959E0 (pid=4)
  0x00 Pcb                          [_KPROCESS Pcb] @ 0xFA80008959E0
  0x160 ProcessLock                  [_EX_PUSH_LOCK ProcessLock] @ 0xFA8000895B40
  0x168 CreateTime                    [WinFileTime:CreateTime]: 0x506A0DA7 (2012-10-
↪01 21:39:51Z)
  0x170 ExitTime                      [WinFileTime:ExitTime]: 0x00000000 (-)
  0x178 RundownProtect               [_EX_RUNDOWN_REF RundownProtect] @ 0xFA8000895B58
  0x180 UniqueProcessId               [unsigned int:UniqueProcessId]: 0x00000004
  0x188 ActiveProcessLinks           [_LIST_ENTRY ActiveProcessLinks] @ 0xFA8000895B68
  ....
```

live (Live)

Launch a Rekall shell for live analysis on the current system.

Plugin	Type	Description
mode	Choices	Mode for live analysis.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

load_as (LoadAddressSpace)

Load address spaces into the session if its not already loaded.

load_plugin (LoadPlugins)

Load user provided plugins.

This probably is only useful after the interactive shell started since you can already use the –plugin command line option.

lookup (Lookup)

Lookup a global in the profile.

This plugin lets the user ask for a specific global constant in the active profile.

Plugin	Type	Description
constant	String	The constant to look up in the profile.
target	String	The type of the constant.
target_args	String	The target args
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

manage_repo (ManageRepository)

Manages the profile repository.

Plugin	Type	Description
build_target	StringParser	A single target to build.
builder_args	ArrayString-Parser	Optional args for the builder.
executable	String	The path to the rekall binary. This is used for spawning multiple processes.
force_build_index	Boolean	Forces building the index.
path_to_repository	String	The path to the profile repository
processes	IntParser	Number of concurrent workers.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

inspect_vaddr (MemoryTranslation)

Inspect the mapping of a virtual address.

Plugin	Type	Description
address	SymbolAddress	Virtual address to inspect.
dtb	IntParser	The DTB physical address.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

null (Null)

This plugin does absolutely nothing.

It is used to measure startup overheads.

address_resolver (PEAddressResolver)

A simple address resolver for PE files.

Plugin	Type	Description
dtb	IntParser	The DTB physical address.
symbol	ArrayString	List of symbols to lookup
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

peinfo (PEInfo)

Print information about a PE binary.

Plugin	Type	Description
address_space	String	The address space to use.
executable	String	If provided we create an address space from this file.
image_base	SymbolAddress	The base of the image.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

The **peinfo** plugin examines a PE file mapped into memory and displays a rich variety of information about it:

- Metadata about the file (architecture, build date etc)
- The PDB guid for the file.
- The list of sections and where they are mapped into the virtual address space
- The import directory.
- The export directory.
- A version resource strings that might exist in the executable.

### Notes

1. This plugin depends on having a valid mapped PE header into memory. Sometimes this is not the case, since under memory pressure the kernel will unmapped the PE headers (since they are not needed after loading).
2. This plugin also works on disk files (PE executable). Simply pass a filename parameter to have it print information about external files.

### Sample output

```
win8.1.raw 15:11:02> peinfo "nt"
-------------------> peinfo("nt")
Attribute            Value
-------------------- -----
Machine              IMAGE_FILE_MACHINE_AMD64
TimeDateStamp        2013-09-14 08:23:16+0000
Characteristics      IMAGE_FILE_EXECUTABLE_IMAGE, IMAGE_FILE_LARGE_ADDRESS_AWARE
GUID/Age             FD3D00D28EDC4527BB922BCC0509D2851
PDB                  ntkrnlmp.pdb
MajorOperatingSystemVersion 6
MinorOperatingSystemVersion 3
MajorImageVersion    6
MinorImageVersion    3
MajorSubsystemVersion 6
MinorSubsystemVersion 3

Sections (Relative to 0xF802D3019000):
Perm Name          VMA            Size
---- --------- -------------- --------------
xr-  .text     0x000000001000 0x00000028d600
xr-  NONPAGED  0x00000028f000 0x000000000200
xr-  POOLCODE  0x000000290000 0x000000002800
-rw  .data     0x000000293000 0x00000000be00
-r-  .reloc    0x000000778000 0x000000008e00
...

Data Directories:
-                                             VMA            Size
---------------------------------------- -------------- --------------
IMAGE_DIRECTORY_ENTRY_EXPORT             0xf802d36ab000 0x0000000135ff
IMAGE_DIRECTORY_ENTRY_IMPORT             0xf802d335b728 0x00000000012c
IMAGE_DIRECTORY_ENTRY_RESOURCE           0xf802d375f000 0x000000031d20
IMAGE_DIRECTORY_ENTRY_EXCEPTION          0xf802d331c000 0x00000003ed6c
IMAGE_DIRECTORY_ENTRY_SECURITY           0xf802d3725e00 0x000000002158
IMAGE_DIRECTORY_ENTRY_BASERELOC          0xf802d3791000 0x000000003cd4
IMAGE_DIRECTORY_ENTRY_DEBUG              0xf802d301a100 0x000000000038
IMAGE_DIRECTORY_ENTRY_COPYRIGHT          0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_GLOBALPTR          0x000000000000 0x000000000000
```

```
IMAGE_DIRECTORY_ENTRY_TLS                 0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG         0xf802d3033f20 0x000000000094
IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT        0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_IAT                 0xf802d335b000 0x000000000728
IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT        0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR      0x000000000000 0x000000000000
IMAGE_DIRECTORY_ENTRY_RESERVED            0x000000000000 0x000000000000

Import Directory (Original):
Name                                                Ord
-------------------------------------------------- -----
ext-ms-win-ntos-werkernel-l1-1-0.dll!WerLiveKernelCloseHandle 1
ext-ms-win-ntos-werkernel-l1-1-0.dll!WerLiveKernelOpenDumpFile 4
ext-ms-win-ntos-werkernel-l1-1-0.dll!WerLiveKernelCancelReport 0
ext-ms-win-ntos-werkernel-l1-1-0.dll!WerLiveKernelInitSystem 3
...
msrpc.sys!MesDecodeBufferHandleCreate               11
msrpc.sys!NdrMesTypeDecode3                         45

Export Directory:
    Entry      Stat Ord   Name
-------------- ---- ----- --------------------------------------------------
0xf802d30ed1f4 M    3     ntoskrnl.exe!AlpcGetHeaderSize (nt!AlpcGetHeaderSize)
0xf802d30ed080 M    4     ntoskrnl.exe!AlpcGetMessageAttribute (nt!
↪AlpcGetMessageAttribute)
0xf802d30ed19c M    5     ntoskrnl.exe!AlpcInitializeMessageAttribute (nt!
↪AlpcInitializeMessageAttribute)
0xf802d36a4004 -    6     ntoskrnl.exe!BgkDisplayCharacter (nt!BgkDisplayCharacter)
0xf802d36a40b8 -    7     ntoskrnl.exe!BgkGetConsoleState (nt!BgkGetConsoleState)
0xf802d36a40e0 -    8     ntoskrnl.exe!BgkGetCursorState (nt!BgkGetCursorState)
0xf802d36a4108 -    9     ntoskrnl.exe!BgkSetCursor (nt!BgkSetCursor)
0xf802d31d23c8 M    10    ntoskrnl.exe!CcAddDirtyPagesToExternalCache (nt!
↪CcAddDirtyPagesToExternalCache)
0xf802d3106888 M    11    ntoskrnl.exe!CcCanIWrite (nt!CcCanIWrite)
...
```

parse_pdb (ParsePDB)

Parse the PDB streams.

Plugin	Type	Description
concise	Boolean	Specify this to emit less detailed information.
dump_dir	String	Path suitable for dumping files.
out-put_filename	String	The name of the file to store this profile.
pdb_filename	String	The filename of the PDB file.
profile_class	String	The name of the profile implementation. Default name is derived from the pdb filename.
verbosity	Int-Parser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.
win-dows_version	String	The windows version (major.minor.revision) corresponding with this PDB. For example, Windows 7 should be given as 6.1

Rekall uses debugging symbols to analyze memory. Each time Microsoft compilers generate a binary (executable or DLL) they also emit debugging information in a separate PDB file. Rekall needs a profile for each binary of interest

(A profile is a JSON file containing important debugging information about the binary).

Use the *fetch_pdb* plugin to fetch the PDB file and the *parse_pdb* plugin to parse it and produce a JSON file for Rekall to use.

Note that normally this plugin is called by other plugins such as *build_local_profile* or automatically by Rekall. So users do not need to call this plugin directly in most cases.

```
[1] win7.elf 23:09:12> fetch_pdb pdb_filename="termdd.pdb", guid=
↪"2A530717E88549BB92DBB72C224EC2B11"
 Trying to fetch http://msdl.microsoft.com/download/symbols/termdd.pdb/
↪2A530717E88549BB92DBB72C224EC2B11/termdd.pd_
 Trying to fetch http://msdl.microsoft.com/download/symbols/termdd.pdb/
↪2A530717E88549BB92DBB72C224EC2B11/termdd.pd_
Extracting cabinet: /tmp/tmpXkEgyu/termdd.pd_
  extracting termdd.pdb

All done, no errors.
[1] win7.elf 23:55:07> parse_pdb pdb_filename="termdd.pdb", output_filename="termdd.
↪json"
               Out<59> Plugin: parse_pdb
[1] win7.elf 23:55:37> !head termdd.json
{
 "$CONSTANTS": {
  "ExEventObjectType": 41408,
  "Globals": 46144,
  "HotPatchBuffer": 45056,
  "IcaChannelDispatchTable": 45856,
  "IcaChargeForPostCompressionUsage": 46106,
  "IcaConnectionDispatchTable": 45632,
  "IcaDeviceObject": 46848,
  "IcaDisableFlowControl": 46105,
```

p (Printer)

A plugin to print an object.

This plugin is an alias to the print python command. Use it when you want to print something to the console.

raise_the_roof (RaisingTheRoof)

A plugin that exists to break your tests and make you cry.

agent (RekallAgent)

The Rekall DFIR Agent.

Plugin	Type	Description
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

moo (RekallBovineExperience3000)

Renders Bessy the cow and some beer.

This is a text renderer stress-test. It uses multiple features at the same time:

- Multiple coloring rules per line (this was a doozy).
- Two columns with colors next to each other.
- Text with its own newlines isn't rewrapped.
- It still wraps if it overflows the cell.
- Bovine readiness and international spirit.

run (Run)

A plugin which runs its argument (using eval).

Note: This plugin is only defined and available when using the main entry point. It is not available when Rekall is used as a library since it allows arbitrary code execution.

run_flow (RunFlow)

Run the flows specified.

Plugin	Type	Description
flow	String	A string encoding a Flow JSON object.
flow_filename	String	A filename containing an encoded Flow JSON object.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

search (Search)

Searches and recombines output of other plugins.

Search allows you to use the EFILTER search engine to filter, transform and combine output of most Rekall plugins. The most common use for this is running IOCs.

Some examples:

- Find the process with pid 1:

```
select * pslist() where proc.pid == 1
```

- Sort lsof output by file descriptor:

```
select * from lsof() order by fd
```

- Filter and sort through lsof in one step:

```
select * from lsof() where proc.name =~ "rekall" order by fd
```

You will probably need to use the *describe* plugin to help discover the exact column structure.

- regex match on array of strings - case insensitive.

```
(Windows)
select proc, proc.environ from pslist() where
  proc.environ.TMP =~ "temp"
```

```
(Linux)
select proc, proc.environ from pslist() where
   proc.environ.PATH =~ "home"
```

- Format using the hex() method, using *as* to name columns.

```
(Windows)
select hex(VAD.start) as start, hex(VAD.end) as end,
      Protect from vad(proc_regex: "rekal")

(Linux)
select hex(start) as start, hex(end) as end, filename
      from maps(proc_regex: "rekall")
```

- Autoselect column names - second column can not clash with first column name (should be hex, column 1).

```
(Windows)
select hex(VAD.start), hex(VAD.end), Protect
      from vad(proc_regex: "rekal")

(Linux)
select hex(start), hex(end), filename from maps(proc_regex: "rekall")
```

- Timestamp user function

```
select proc, timestamp(proc.create_time) from pslist()
```

- Yarascan with sub query

```
select * from file_yara(
   paths: (
    select path.filename from glob(
        "c:\windows\*.exe")).filename,
   yara_expression: "rule r1 {strings: $a = "Microsoft" wide "
        "condition: any of them}")
```

On Linux:

```
select * from file_yara(
      paths: (
        select path.filename from glob(
           "/home/*/.ssh/*")).filename,
      yara_expression: "rule r1 {strings: $a = "ssh-rsa" condition: any of them}")
```

- Parameter interpolations:

```
a =  "select * from file_yara(paths: ( "
     "select path.filename from glob({0})).filename, yara_expression: {1})"
search a, [r"c:\windows\*.exe",
     "rule r1 {strings: $a = "Microsoft" wide condition: any of them}"]
```

- WMI integration + unknown field:

```
select Result.Name, Result.SessionId, Result.foo
     from wmi("select * from Win32_Process")

select Result.Name, Result.BootDevice
     from wmi("select * from Win32_OperatingSystem")
```

- Describe WMI dynamic query

```
describe wmi, dict(query="select * from Win32_Process")
```

- Substitute a single string

```
select sub("Microsoft", "MS", Result.Name)
       from wmi("select * from Win32_OperatingSystem")
```

- Substiture an array

```
select sub("rekal", "REKALL", proc.cmdline) from pslist()
```

Plugin	Type	Description
query	String	The dotty/EFILTER query to run.
query_parameters	ArrayString	Positional parameters for parametrized queries.
silent	Boolean	Queries should fail silently.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

sdel (SessionDelete)

Delete a session.

slist (SessionList)

List the sessions available.

smod (SessionMod)

Modifies parameters of the current analysis session.

Any session parameters can be set here. For example:

smod colors="no", paging_limit=10, pager="less"

snew (SessionNew)

Creates a new session by cloning the current one.

The Rekall interactive console may be used to analyze several images at the same time. We do this by switching sessions. Each image has a unique session, but since none of the sessions are global, we can switch from one session to the next.

Rekall's session management commands can be used to switch between sessions.

The example below shows us loading a second session with a new image. We switch to the new session and list processes in it. We then switch back and delete the new session. Note how the prompt changes as we switch from one session to the other.

```
[1] win7.elf 23:55:46> snew filename="/home/scudette/images/win10.aff4"
Created session [2] /home/scudette/images/win10.aff4 (2)
               Out<61> Plugin: snew
[2] /home/scudette/images/win10.aff4 (2) 23:57:03> pslist
---------------------------------------------------> pslist()
```

```
  _EPROCESS             Name            PID   PPID   Thds    Hnds    Sess  Wow64        ␣
↪   Start                      Exit
-------------- -------------------- ----- ------ ------ -------- ------ ------ -------
↪----------------- ------------------------
0xe0003486d680 System                  4      0     82        -       - False  2015-
↪06-03 06:56:02Z     -
0xe00035e54040 smss.exe              260      4      2        -       - False  2015-
↪06-03 06:56:02Z     -
0xe00035b84080 csrss.exe             332    324      9        -       0 False  2015-
↪06-03 06:56:03Z     -
0xe0003489b280 wininit.exe           400    324      1        -       0 False  2015-
↪06-03 06:56:03Z     -
[2] /home/scudette/images/win10.aff4 (2) 23:57:09> sswitch 1
                                          Out<63> Plugin: sswitch
[1] win7.elf 23:57:12> pslist
---------------------> pslist()
  _EPROCESS             Name            PID   PPID   Thds    Hnds    Sess  Wow64        ␣
↪   Start                      Exit
-------------- -------------------- ----- ------ ------ -------- ------ ------ -------
↪----------------- ------------------------
0xfa80008959e0 System                  4      0     84      511       - False  2012-
↪10-01 21:39:51Z     -
0xfa80024f85d0 svchost.exe           236    480     19      455       0 False  2012-
↪10-01 14:40:01Z     -
0xfa8001994310 smss.exe              272      4      2       29       - False  2012-
↪10-01 21:39:51Z     -
0xfa8002259060 csrss.exe             348    340      9      436       0 False  2012-
↪10-01 21:39:57Z     -
[2] /home/scudette/images/win10.aff4 (2) 23:57:25> slist
  [1] win7.elf
* [2] /home/scudette/images/win10.aff4 (2)
                                          Out<68> Plugin: slist
[1] win7.elf 23:57:33> sdel 2
               Out<70> Plugin: sdel
[1] win7.elf 00:01:49> slist
* [1] win7.elf
               Out<73> Plugin: slist
```

sswitch (SessionSwitch)

Changes the current session to the session with session_id.

cc (SetPartitionContext)

A mixin for those plugins requiring a physical address space.

Args:

physical_address_space: The physical address space to use. If not specified we use the following options:

1. session.physical_address_space,
2. Guess using the load_as() plugin,
3. Use session.kernel_address_space.base.

Plugin	Type	Description
partition_number	IntParser	The partition to switch to.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

simple_yarascan (SimpleYaraScan)

A Simple plugin which only yarascans the physical Address Space.

This plugin should not trigger profile autodetection and therefore should be usable on any file at all.

Plugin	Type	Description
binary_string	String	A binary string (encoded as hex) to search for. e.g. 000102[1-200]0506
context	IntParser	Context to print after the hit.
hits	IntParser	Quit after finding this many hits.
limit	IntParser	The length of data to search.
pre_context	IntParser	Context to print before the hit.
start	IntParser	Start searching from this offset.
string	String	A verbatim string to search for.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.
yara_expression	String	If provided we scan for this yara expression.
yara_file	String	The yara signature file to read.

fls (TSKFls)

A mixin for those plugins requiring a physical address space.

Args:

physical_address_space: The physical address space to use. If not specified we use the following options:

1. session.physical_address_space,
2. Guess using the load_as() plugin,
3. Use session.kernel_address_space.base.

Plugin	Type	Description
dir_path	String	Directory path to print content of
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

mmls (TskMmls)

A mixin for those plugins requiring a physical address space.

Args:

physical_address_space: The physical address space to use. If not specified we use the following options:

1. session.physical_address_space,
2. Guess using the load_as() plugin,
3. Use session.kernel_address_space.base.

Plugin	Type	Description
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

version_scan (VersionScan)

Scan the physical address space for RSDS versions.

Plugin	Type	Description
name_regex	RegEx	Filter module names by this regex.
scan_filename	String	Optional file to scan. If not specified we scan the physical address space.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

When the Microsoft Compilers create a binary (Executable or DLL) they leave a unique GUID in the PE header, so that the corresponding PDB file can be located for this binary.

The GUID is encoded using a known signature and therefore we can scan for all GUIDs which might appear in the memory image. This is useful to locate the exact version of binaries running in the memory image. Often malware authors forget to disable PDB file generation in Visual Studio and the GUID remains in the malware. In that case scanning for a known malicious GUID can be a strong signature.

In the below example we scan the memory image for the exact version of the windows kernel. Note how hits can be restricted by using a regular expression.

```
[1] win7.elf 00:01:51> version_scan name_regex="krnl"
  Offset (P)             GUID/Version                          PDB
-------------- --------------------------------- ------------------------------
0x0000027bb5fc F8E2A8B5C9B74BF4A6E4A48F180099942 ntkrnlmp.pdb
```

vmscan (VmScan)

Scan the physical memory attempting to find hypervisors.

Once EPT values are found, you can use them to inspect virtual machines with any of the rekall modules by using the –ept parameter and specifying the guest virtual machine profile.

Supports the detection of the following virtualization techonlogies:

- Intel VT-X with EPT. Microarchitectures: + Westmere + Nehalem + Sandybridge + Ivy Bridge + Haswell
- Intel VT-X without EPT (unsupported page translation in rekall). + Penryn

For the specific processor models that support EPT, please check: http://ark.intel.com/products/virtualizationtechnology.

Plugin	Type	Description
image_is_guest	Boolean	The image is for a guest VM, not the host.
no_nested	Boolean	Don't do nested VM detection.
no_validation	Boolean	[DEBUG SETTING] Disable validation of VMs.
offset	IntParser	Offset in the physical image to start the scan.
quick	Boolean	Perform quick VM detection.
show_all	Boolean	Also show VMs that failed validation.
verbosity	IntParser	An integer reflecting the amount of desired output: 0 = quiet, 10 = noisy.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search