Chucky Documentation

Release 1

Alwin Maier, Fabian Yamaguchi, Ke Yang

April 12, 2016

This document is for this modified version of Chucky implementation and is also suitable for the original version (developed by Alwin Maier and Fabian Yamaguchi).

CHAPTER 1

Introduction of Chucky

Chucky is a missing check vulnerability detection method designed by Fabian Yamaguchi. It statically taints source code and identifies anomalous or missing conditions linked to security-critical objects. Chucky analyzes functions for anomalies. To this end, the usage of symbols used by a function is analyzed by comparing the checks used in conjunction with the symbol with those used in similar functions.

CHAPTER 2

The Implementation

This implemetation of Chucky interactive with the database parsed by joern (another tools developed by Fabian et al). After a Robust Parsing by joern, conditions, assignments and API symbols are extracted from every function and all the code information are stored in the graph database as Code Property Graphs including AST,CFG and DDG. Joern use Neo4j to store these information.

There are five step for Chucky to complete the analyze.

1. **Identification of sources and sinks.** The query symbol is given by user as an analyse target. So the first job of Chucky is to locate them in the database and find all the candidates(functions that use the query symbol). According to different symbol types, this can be achived by a group of a well defined gremlin query.

2. **Neighborhood discovery.**

   - Viewing the function as a document and defining the key words as the element concerned in the AST, Chucky describe each function as a symbol vector.
   - Chucky find the similarest top k functions to the query function by applying the information retrieval technique in this vector space.

   The first procedure is implemented by gremlin query in joern-tools and the second one is implemented by pure python.

3. **Lightweight tainting.**

   Idendify the the condition code of **if**, **while** and **for** in which there exists a symbol in the path from the source to the sink. These symbols may influence or be influenced by the the query symbol in each top k similarest functions. This step is also implemented by gremlin queries as such relations can be described as a path in the code property graph.

4. **Embedding of functions.**

   Describe each function as a sparse 0-1 vector according to the existence of the condition key words discovered by the pervious step.

5. **Anomaly detection.**

   Find the most significant missing word in the condition vector of the query function set off by the neighborhoods. The anomaly score of the query function is expressed by the percentage of time the significant missing key word exists in the neighborhoods.

All the analysis are based on the extensible query language defined in joern-tools by Gremlin and a wrapped inteface defined by python-joern.

For the orginal idea, please refer to Chucky: Exposing Missing Checks in Source Code for Vulnerability Discovery Fabian Yamaguchi, Christian Wressnegger, Hugo Gascon, and K. Rieck *ACM Conference on Computer and Communications Security (CCS)*

About the Modification.

1. **Refactor to clean the middle code.**
    - Replace sally embedding module by pure python code(transplant the code witten by Fabian) to remove the data exchange cost on disk.
    - Fix some bugs and make it more robust.
2. **Rewirte the KNN class to support the neighborhood selection strategy:**
    - Leverage the name(file name or function name) information and the caller set information of a function when it's usefull.
    - Kick some name irrelevant functions out, and set a robust threshold for the recomandation of good candidate.
3. **Add multi-source/sink support.**
    - Design a new option set for user to specify the multi-source/sink.
    - Use the combination of source/sink as the key feature to find candidate neighborhood.
    - Use the union of the tainted condition features as the condition embedding feature.
    - Refactor the job generation and remove the redundant function selection to improve the performance.
4. Add a report module to show the detail report.

Note: the advancement of the modification still needs to be judged and more evaluation is required. Connect Ke Yang(123yangke321@sina.com) for more information.

Although this is a NON-OFFICIAL document for Chucky, hope it will be helpful for people who are intersted in Chucky and working and studying in this area.

Contents:

3.1 Download and Installation

3.1.1 Dependencies

- joern >= 3.0 https://github.com/fabsx00/joern
- python-joern >=0.2 https://github.com/fabsx00/python-joern
- joern-tools >=0.1 https://github.com/fabsx00/joern-tools
- Neo4j >=2.1 http://www.neo4j.org

- Python 2.7

This version is for Debian & Ubuntu Linux only.

3.1.2 Clone from git Repository

To clone it from Git repository, run the following commands in the terminal:

```
$ sudo apt-get install git #skip this command if you have git already installed
$ git clone https://github.com/yangke/chucky-ng.git
```

3.2 Usage

3.2.1 Example 1

```
$ python chucky.py --p length -n 25 --interactive
```

3.2.2 Example 2

```
$ python chucky.py -p length --callee png_free -var slength -n 3 -l png_handle_sCAL -r
```

3.2.3 Usage Pattern

Suppose we have already parsed the code and we have configured and started the neo4j database service. (For parsing the code and database configuration please refer to the document of joern. Don't worry, the section *A Quick Start Example* will also mention a little about this.):

```
$ cd chucky-ng/chucky
$ python chucky.py [-h] [-f FUNCTION] [--callee CALLEES [CALLEES ...]]
             [-p PARAMETERS [PARAMETERS ...]]
             [-var VARIABLES [VARIABLES ...]] -n N_NEIGHBORS
             [-c CHUCKY_DIR] [-o OUTPUT_REPORT_DIRECTORY] [-r]
             [--interactive] [-l LIMIT] [-d | -v | -q]
```

optional arguments:

```
-h, --help           Show this help message and exit.
-f FUNCTION, --function FUNCTION
                     Specify the function to analysis.
                     If this option is configured, the analysis will only perform on this function.
-n N_NEIGHBORS, --n-neighbors N_NEIGHBORS
                     Number of neighbours to consider for neighborhood discovery.
-c CHUCKY_DIR, --chucky-dir CHUCKY_DIR
                     The directory holding chucky's data such as cached
                     symbol embeddings and possible annotations of sources and sinks.
-o OUTPUT_REPORT_DIRECTORY, --output-report-directory OUTPUT_REPORT_DIRECTORY
                     The report output directory of chucky. For each target function under
                     analyzation chucky will generate a detail report.
-r, --report         Output the detail report for each function under analyzation.
--interactive        Enable interactive mode.
-l LIMIT, --limit LIMIT
                     Limit analysis to functions with given name.
```

```
-d, --debug         Enable debug output.
-v, --verbose       Increase verbosity.
-q, --quiet         Be quiet during processing.
```

source_sinks:

```
--callee CALLEES [CALLEES ...]
                    Specify the identifier name of callee type source/sink.
-p PARAMETERS [PARAMETERS ...], --parameter PARAMETERS [PARAMETERS ...]
                    Specify the identifier name of parameter type source/sink.
-var VARIABLES [VARIABLES ...], --variable VARIABLES [VARIABLES ...]
                    Specify the identifier name of variable type source/sink.
```

To get a quick start, please see *A Quick Start Example*.

3.3 A Quick Start Example

Suppose we are the planning to analyse the code of image processing library LibPNG(version 1.2.44).

3.3.1 Download and Extract

Download and extract the the source code of libPNG.

```
$ wget http://sourceforge.net/projects/libpng/files/libpng12/older-releases/1.2.44/libpng-1.2.44.tar
$ tar xvzf libpng-1.2.44.tar.gz
```

3.3.2 Generate the graph database

Run the following command:

```
$ joern libpng-1.2.44
```

A hidden directory `.joernIndex` will be generated under the current directory(suppose the current directory is `$TEST`).

3.3.3 Configure Database Server

Configure the graph database server Neo4j

Assume `$NEO4J_HOME` is the install directory of your Neo4j(Note that current joern only support 1.9.* version serials). Edit the file `$NEO4J_HOME/conf/server.properties`. As an example, for neo4j-1.9.7, you should open the file `neo4j-1.9.7/conf/neo4j-server.properties`.

Then change:

```
#org.neo4j.server.database.location=data/graph.db
```

to:

```
#org.neo4j.server.database.location=$TEST/.joernIndex
```

and save it.

3.3.4 Start Neo4j

Start Neo4j database.

```
$ $NEO4J_HOME/bin/neo4j start
```

Go to your chucky directory `chucky-ng/chucky` and run a chucky analysis.

```
$python chucky.py --parameter length -n 25 |sort -r -k 1
```

Then Chucky will generate the report to the screen:

```
0.88000     process_data                        132644      Parameter       png_uint_32     length
0.88000     png_write_chunk_start                21892      Parameter       png_uint_32     length
0.88000     png_handle_sCAL                       7855      Parameter       png_uint_32     length
0.88000     png_handle_pCAL                       7142      Parameter       png_uint_32     length
0.88000     png_handle_hIST                       6432      Parameter       png_uint_32     length
0.48000     png_push_handle_zTXt                130041      Parameter       png_uint_32     length
0.48000     png_push_handle_tEXt                129600      Parameter       png_uint_32     length
0.48000     png_push_handle_iTXt                130979      Parameter       png_uint_32     length
0.48000     png_handle_zTXt                       9120      Parameter       png_uint_32     length
0.48000     png_handle_tEXt                       8636      Parameter       png_uint_32     length
```

Following table explains some of the key column.

column 1	column 2	column 3	column 6	column 7	column 15
anomaly score	function name	node id	query symbol	sinificant missing symbol	function location
0.88000	png_handle_sCAL	7855	length	length	libpng-1.2.44/pngrutil.c:1784:0:52039:56355

3.3.5 Analysis

For the vulnerable function **png_handle_sCAL** as reported in CVE-2011-2692, we can see from the result that it is ranked in top 5(all the top 5 functions have the highest anomaly score 0.88). This is because most of the similar functions(the first column shows the percentage) perform the check for the parameter **length**, howerver, **png_handle_sCAL** doesn't check it. We call these similar functions the neighborhoods of **png_handle_sCAL**. Chucky is a efficient tool for checking such statistically significant missing case.

3.4 Experiment Tutorial

This experiment tutorial help you to finish the evaluation described by this paper.

It's similar with the evaluation section Chucky: Exposing Missing Checks in Source Code for Vulnerability Discovery, but the ROC curves are generated by the middle result(The rank lists of similar functions).

To do the experiment, you should do the following steps:

1. Generate the code database.
2. Modify the code.
3. Run the automatic script.

3.4.1 Generate the Database

The database can be generated by joern(2.0-3.0) according to the method Fabian described in Chucky paper. That is, patch the vulnerability as the original version, then remove one check in one function from the original versions in a round robin fashion to generate such many code versions and then use joern to generate the code graph database for each vulnerable version. The version and the respective vulnerability number are listed below.

Project	Vulnerability	Declaration Type	Symbol	TYPE	#With Check	#Symbol Users	#F	LOC
firefox-4.0(/js)	CVE-2010-3183	uintN	argc	parameter	10	557	5649	372450
linux-2.6.34.13(/fs)	CVE-2010-2071	struct dentry*	dentry	parameter	8	1104	19178	955943
libpng-1.2.44	CVE-2011-2692	png_uint_32	length	parameter	19	29	473	40255
libtiff-3.9.4	CVE-2010-2067	TIFFDirEntry*	dir	parameter	9	75	609	332762
pidgin-2.7.3(/libpurple)	CVE-2010-3711	•	purple_base64_decode	callee	18	30	7390	332762

3.4.2 Modify the Code

1. Remove the # symbol at the head of the two lines in the `try` block of function analyze():

```
    #for n in nearestNeighbors:
    #    print str(n)+"\t"+n.location()
```

2. Comment out all the following code in `try` block(that means we just print the neighborhood selection result).
3. Define the environment variable `$NEO4J_HOME` to point it to your neo4j program directory.
4. Change the variable `cfgfile` in the script file `neighbor` to the absolute location of the configuration file `neo4j-server.properties`.
5. change the variable `line` in neighbor to the line of variable `org.neo4j.server.database.location` in the configuration file `conf/neo4j-server.properties` of your Neo4j database.

```
    line=11
```

5. Change the value of the `dbpath` to the location of all of your database.Note that the directory must be organized as `$dbpath/$projname/$funcname/.joernIndex`. The projenames and funcnames must be equal to the names listed in the script file `neighbor`.

3.4.3 Run the Auto-Script

```
$ cd chucky-ng/chucky
$ neighbor
$ python ROC.py
```

The shell script `neighbor` dump the result of KNN algorithm to the current file directory, then the `ROC.py` read the directory and generate the points in the directory named `ROC`.

3.4.4 Output Hierarchy

- The directory `neighbors` output by script neighbor will hold the hierarchy `$neighbors/$projname/$function_name`, for example, `neighbors/libpng/png_handle_cHRM`.
- The final ROC points will be generated in file `ROC/$projname-neighbors_ROC`, for example, `ROC/libpng-neighbors_ROC`).

At last, you can import these files of ROC point lists into drawing program to plot the diagram.

3.4.5 Details About the 64 Function

Here is the detail information about the 64 function for evaluation.

Firefox-4.0

Order	Function	Location
1	array_concat	js/src/jsarray.cpp
2	array_extra	js/src/jsarray.cpp
3	array_indexOfHelper	js/src/jsarray.cpp
4	array_slice	js/src/jsarray.cpp
5	array_splice	js/src/jsarray.cpp
6	array_unshift	js/src/jsarray.cpp
7	js::array_sort	js/src/jsarray.cpp
8	LookupGetterOrSetter	js/src/xpconnect/src/xpcquickstubs.cpp
9	DefineGetterOrSetter	js/src/xpconnect/src/xpcquickstubs.cpp
10	PropertyOpForwarder	js/src/xpconnect/src/xpcquickstubs.cpp

linux-2.6.34.13

Order	Function	Location
1	btrfs_xattr_acl_set	fs/btrfs/acl.c
2	jffs2_acl_setxattr	fs/jffs2/acl.c
3	ext2_xattr_set_acl	fs/ext2/acl.c
4	ext3_xattr_set_acl	fs/ext3/acl.c
5	ext4_xattr_set_acl	fs/ext4/acl.c
6	ocfs2_xattr_acl_set	fs/ocfs2/acl.c
7	generic_acl_set	fs/generic_acl.c
8	posix_acl_set	fs/reiserfs/xattr_acl.c

libpng-1.2.44

Order	Function	Location
1	png_handle_Bkgd	pngrutil.c
2	png_handle_cHRM	pngrutil.c
3	png_handle_gAMA	pngrutil.c
4	png_handle_iCCP	pngrutil.c
5	png_handle_IEND	pngrutil.c
6	png_handle_IHDR	pngrutil.c
7	png_handle_iTXt	pngrutil.c
8	png_handle_oFFs	pngrutil.c
9	png_handle_pHYs	pngrutil.c
10	png_handle_PLTE	pngrutil.c
11	png_handle_sBIT	pngrutil.c
12	png_handle_sCAL	pngrutil.c
13	png_handle_sPLT	pngrutil.c
14	png_handle_sRGB	pngrutil.c
15	png_handle_tEXt	pngrutil.c
16	png_handle_tIME	pngrutil.c
17	png_handle_tRNS	pngrutil.c
18	png_handle_unknown	pngrutil.c
19	png_handle_zTXt	pngrutil.c

tiff-3.9.4

Order	Function	Location
1	TIFFFetchByteArray	libtiff/tif_dirread.c
2	TIFFFetchLongArray	libtiff/tif_dirread.c
3	TIFFFetchPerSampleAnys	libtiff/tif_dirread.c
4	TIFFFetchPerSampleLongs	libtiff/tif_dirread.c
5	TIFFFetchPerSampleShorts	libtiff/tif_dirread.c
6	TIFFFetchShortArray	libtiff/tif_dirread.c
7	TIFFFetchShortPair	libtiff/tif_dirread.c
8	TIFFFetchString	libtiff/tif_dirread.c
9	TIFFFetchSubjectDistance	libtiff/tif_dirread.c

Pidgin-2.7.3

Order	Function	Location
1	digest_md5_handle_chanllenge	lipurple/protocols/jabber/auth_digest_md5.c
2	do_buddy_avatar_update_data	lipurple/protocols/jabber/useravatar.c
3	got_sessionreq	lipurple/protocols/msn/slp.c
4	jabber_data_create_from_xml	lipurple/protocols/jabber/data.c
5	jabber_ibb_parse	lipurple/protocols/jabber/ibb.c
6	jabber_scram_feed_parser	lipurple/protocols/jabber/auth_scram.c
7	jabber_vcard_parse	lipurple/protocols/jabber/buddy.c
8	jabber_vcard_parse_avatar	lipurple/protocols/jabber/presence.c
9	jabber_vacard_save_mine	lipurple/protocols/jabber/buddy.c
10	msim_msg_get_binary_from_element	lipurple/protocols/myspace/message.c
11	msn_oim_report_to_user	lipurple/protocols/msn/oim.c
12	msn_switchboard_shoe_ink	lipurple/protocols/msn/switchboard.c
13	purple_mime_decode_field	lipurple/util.c
14	purple_ntlm_parse_type2	lipurple/ntlm.c
15	scram_handle_challenge	lipurple/protocols/jabber/auth_scram.c
16	scram_handle_success	lipurple/protocols/jabber/auth_scram.c
17	yahoo_process_p2p	lipurple/protocols/yahoo/libymsg.c
18	yahoo_process_status	lipurple/protocols/yahoo/libymsg.c