Armadito Antivirus Documentation

Version 1.0

Teclib

27 February 2017

Table des matières

1 Introduction **3**

2 Compilation **5**
2.1 Compilation sur Linux 5
2.2 Compilation sur Windows 10

3 Installation **15**
3.1 Installation sur Linux 15
3.2 Installation sur Windows 15

4 Configuration **17**
4.1 Configuration sous Linux 17
4.2 Configuration sur Windows 18

5 Contribuer **21**

6 Licences **23**

7 FAQ **25**

8 Captures d'écran **27**
8.1 Le tableau de bord 28
8.2 L'interface de scan 28
8.3 Le journal 28

Armadito Antivirus est un antivirus open source, qui protège vos ordinateurs et vos serveurs contre les malwares et les virus. Il inclut la détection classique de logiciels malveillants basés sur les signatures et fournit des modules de détection heuristiques novateurs pour les fichiers binaires (MS-Windows et GNU / Linux) et pour les documents PDF.

CHAPITRE 1

Introduction

Armadito Antivirus est un antivirus open-source qui protège vos ordinateurs personnels et serveurs contre les malwares. Il inclut la détection classique de logiciels malveillants basés sur des signatures et fournit des modules de détection heuristiques novateurs pour les fichiers binaires (MS-Windows et GNU / Linux) et pour les documents PDF.

Ce projet est conçu d'une manière qui permet à quiconque d'améliorer les techniques de détection des contenus malicieux. En effet, les experts en sécurité peuvent développer leurs propres modules d'analyse. N'ayant pas à se préoccuper sur le fonctionnement du reste de l'antivirus, ils sont alors à même de se concentrer uniquement sur les techniques anti-malwares. Cette posssibilité d'implémentation modulaire vise à encourager les gens à tester et à développer librement des techniques expérimentales.

Pour une utilisation personnelle, une interface graphique simple a été développé avec des technologies web (AngularJs / HTML5 / CSS3). L'interface est par conséquent multi-plateforme (Windows, Linux, Mac OS X). Elle donne accès à toutes les fonctionnalités de l'antivirus : analyse à la demande, protection en temps réel, zone de quarantaine, journal de détection des menaces, etc.

CHAPITRE 2

Compilation

Compilation sur Linux

Sur linux, les sources de l'antivirus sont compilées en utilisant automake et autoconf pour générer les fichiers Makefiles. Vous pouvez compiler chaque partie séparément par nous-mêmes.

Les sources sont accessibles publiquement sur github, dans des dépôts git séparés :

- Core : https://github.com/armadito/armadito-av
- WebUI : https://github.com/armadito/armadito-web-ui
- SystrayUI : https://github.com/armadito/armadito-systray-ui
- Module Clamav : https://github.com/armadito/armadito-mod-clamav
- Module PDF : https://github.com/armadito/armadito-mod-pdf
- Module H1 : https://github.com/armadito/armadito-mod-h1
- Module Yara : https://github.com/armadito/armadito-mod-yara

Armadito core

Armadito core correspond à la librairie *libarmadito*. Les symboles exportés depuis cette librairie permettent à tous les modules d'utiliser la même API.

Les sources sont accessibles au public sur github.com, vous pouvez les obtenir avec la commande suivante :

```
$ git clone https://github.com/armadito/armadito-av.git -b DEV
```

Les prérequis

Pour construire libarmadito, vous avez besoin des outils suivants :

- automake / autoconf
- Fabrication GNU
- Compilateur C
- glib, libmagic, libxml2, libmicrohttpd, bibliothèques libcurl et en-têtes

Pour installer les en-têtes nécessaires :

- Ubuntu :

```
apt-get install automake autoconf libtool libglib2.0-dev libmagic-dev libxml2-dev libjson-c-dev libm
```

- Fedora :

```
dnf install automake autoconf libtool glib2-devel file-devel libxml2-devel json-c-devel libmicrohttpd
```

Configuration

Une fois le repo cloné, vous devez initialiser la compilation en utilisant automake et autoconf. Un script shell **autogen.sh** est prévu pour faciliter cette étape :

```
$ ./autogen.sh
+ aclocal --force
+ libtoolize --force --automake --copy
+ autoheader --force
+ automake --foreign --add-missing --force-missing --copy
+ autoconf --force
```

Cela générera les fichiers **Makefile.in** et **configure**.

configure script prend les options suivantes :

--prefix=PREFIX install architecture-independent files in PREFIX [default is /usr/local]

--enable-debug enable debugging [default is yes]

Le répertoire ** PREFIX ** sera utilisé par ** make install **. L'utilisation du PREFIX est préférable étant donné que les modules et l'interface graphique ont besoin de l'installation préalable de libarmadito.

Compiler dans un répertoire distinct est fortement recommandé, sauf si vous voulez vraiment ajouter dans les sources des objets, des bibliothèques, des binaires et d'autres choses.

```
$ mkdir /home/joebar/build/libarmadito
```

L'appel typique du script de configuration est :

```
$ /home/joebar/armadito-av/libarmadito/configure --prefix=/home/joebar/install --enable-debug
```

Compilation

Une fois configuré, la compilation est facile :

```
$ make
```

Installation

Après la compilation, l'installation se fait par :

```
$ make install
```

Cela installera les bibliothèques, les outils, les fichiers d'en-tête ... dans les sous-répertoires du ** PREFIX ** Répertoire défini au moment de la configuration.

Armadito modules

La compilation est relativement la même pour les différents modules d'analyse de l'antivirus Armadito. Il n'est pas nécessaire de compiler/installer tous les modules pour utiliser l'antivirus. Le choix des modules est en effet totalement libre.

Avertissement : Assurez-vous d'avoir compilé **libarmadito** (core) avant de compiler un de ces modules.

Pré-requis

Pour compiler un module, vous avez besoin des outils suivants :
- automake/autoconf
- GNU make
- C compiler
- libarmadito (core)

Certains modules ont des dépendances supplémentaires :

armadito-mod-clamav :
- libclamav library and headers. Ubuntu : libclamav-dev

armadito-mod-yara :
- libyara library and headers. Ubuntu : libyara-dev

armadito-mod-h1 :
- pas de dépendances supplémentaires

armadito-mod-pdf :
- pas de dépendances supplémentaires

Clone

```
git clone -b DEV https://github.com/armadito/armadito-mod-clamav.git
git clone -b DEV https://github.com/armadito/armadito-mod-yara.git
git clone -b DEV https://github.com/armadito/armadito-mod-h1.git
git clone -b DEV https://github.com/armadito/armadito-mod-pdf.git
```

Configuration

Une fois le repo cloné, vous devez initialiser la compilation en utilisant automake et autoconf. Un script shell **autogen.sh** est d'ailleurs prévu pour faciliter cette étape :

```
$ ./autogen.sh
+ aclocal --force
+ libtoolize --force --automake --copy
+ automake --foreign --add-missing --force-missing --copy
+ autoconf --force
```

Cela générera les fichiers **Makefile.in** et **configure**.

configure script prend les options suivantes :

--prefix=PREFIX	installe l'architecture-independent de PREFIX [/usr/local]
--enable-debug	enable debugging [default is yes]

Le répertoire **PREFIX** sera utilisé par **make install**. Son utilisation est obligatoire, sauf pour la création d'un paquet ou l'installation dans les répertoires du système.

libarmadito utilise **pkg-config** pour spécifier les options de compilation relatives à libarmadito. Puisque le fichier de spécification **libarmadito.pc** pour **pkg-config** n'est pas situé dans le repertoire standard (habituel **/usr/lib/pkgconfig**), l'appel du script configure doit inclure la variable d'environnement **PKG_CONFIG_PATH**.

Compiler dans un répertoire distinct est fortement recommandé, sauf si vous voulez vraiment ajouter dans les sources des objets, des bibliothèques, des binaires et d'autres choses.

Par exemple :

```
$ REPO_GIT/configure --prefix=/home/joebar/install --enable-debug PKG_CONFIG_PATH=/home/joebar/instal
```

Notez que le chemin spécifié dans la valeur de **PKG_CONFIG_PATH** doit être cohérent avec le **PREFIX** utilisé dans l'installation libarmadito (voir **Armadito core** compilation sur linux).

Compilation

```
$ make
```

Installation

```
$ make install
```

Cela installera les bibliothèques, les outils, les fichiers d'en-tête ... dans les sous-répertoires du chemin **PREFIX** défini au moment de la configuration. Exemple : /home/joebar/install/...

Armadito WebUI

Armadito WebUI s'appuie sur les technologies Web récentes pour fournir une interface graphique multiplateforme pour l'antivirus Armadito. Les sources sont accessibles au public sur github.com, vous pouvez les obtenir avec la commande suivante :

```
$ git clone https://github.com/armadito/armadito-web-ui.git -b DEV
```

Prérequis

Pour exécuter l'interface utilisateur graphique Armadito, vous avez besoin de :

— bower

Installation de bower (doit être effectué en tant que root aussi) :

```
$ npm install -g bower
```

Installation des dépendances

```
$ cd /home/joebar/armadito-web-ui
$ bower install
```

Configuration

Une fois le repo cloné, vous devez initialiser la compilation en utilisant automake et autoconf. Un script shell **autogen.sh** est fourni pour faciliter cette étape :

```
$ ./autogen.sh
+ aclocal --force
+ automake --foreign --add-missing --force-missing --copy
+ autoconf --force
```

Cela générera les fichiers **Makefile.in** et le script **configure**.

** configure ** script prend les options utiles suivantes :

--prefix=PREFIX install architecture-independent files in PREFIX [default is /usr/local]

Le répertoire **PREFIX** sera utilisé par **make install**. Son utilisation est obligatoire, à moins de construire un paquet et d'installer dans les répertoires système.

L'appel typique du script de configuration est :

```
$ /home/joebar/armadito-web-ui/configure --prefix=/home/joebar/install
```

Compilation

```
$ make
```

Installation

```
$ make install
```

Cela installera les bibliothèques, les outils, les fichiers d'en-tête ... dans les sous-répertoires du **PREFIX** défini au moment de la configuration.

Lancement de l'interface

Tout d'abord, le daemon Armadito doit être lancé.

Si c'est le cas, ouvrez votre navigateur Web et accédez à l'URL suivante :

http://localhost:8888/app/index.html

Débogage de l'interface

Une fois l'interface lancée :

— cliquez avec le bouton droit de la souris dans la fenêtre pour afficher le menu de débogage et sélectionnez «Inspecter» ou appuyez sur F12
— dans la fenêtre de l'inspecteur, sélectionnez l'onglet "console"

Compilation avec grunt

Installer grunt :

```
$ npm install -g grunt-cli
```

Exécuter *grunt* pour compiler et *grunt serve* pour l'aperçu.

Vous pouvez utiliser l'option "–force" si vous voulez compiler avec des warnings.

Compilation sur Windows

Danger : L'interface graphique est en cours de refonte et la nouvelle version n'est pas encore portée pour Windows.

Sous Windows, vous pouvez compiler des sources Armadito AV avec Visual Studio. Cela a été testé avec Visual Studio 2013. Vous devrez peut-être appliquer certaines modifications concernant la version de Visual Studio que vous utilisez.

Testé sur : Windows 7 64 bits avec Service Pack 1.

La solution Armadito pour Visual Studio est constitué des sous-projets suivants :

- **Driver**
  - ArmaditoGuard (sources)
  - ArmaditoGuard Package (installeur)
- **Libarmadito**
  - libarmadito (Armadito core)
- **Modules**
  - clamav_a6o (module clamav)
  - moduleH1 (module heuristique)
  - modulePDF (module PDF)
- **Service**
  - ArmaditoSvc (service d'analyse)
- **Installer**
  - ArmaditoGuard-setup (installation du driver)
  - Armadito-db-setup (projet d'installation pour les bases de données des modules)
  - Armadito-setup (projet d'installation pour armadito)

Libarmadito et modules

Les symboles exportés par libarmadito permettent à tous les modules d'utiliser la même API. Sous Windows, le résultat de la compilation est une DLL : ** libarmadito.dll **. Pour plus de simplicité, nous avons regroupé les dépendances windows dans une archive générée de manière automatique.

Pré-requis

- Microsoft Visual Studio 2013 (Community edition or more)
- Armadito windows dependencies archive (deps-x.zip)

Décompresser **deps-x.zip** dans le répertoire racine des sources armadito-av. Vous devriez avoir ces chemins de dépendances exactes :

```
SOMEWHERE\armadito-av\deps\glib\...
SOMEWHERE\armadito-av\deps\json-c\...
```

Compilation

Ouvrez la solution Armadito-AV à l'emplacement :

```
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\Armadito-AV.sln
```

Sélectionnez le projet correspondant dans l'Explorateur de solutions selon ce que vous souhaitez compiler :

- **Core** : Lib-armadito\libarmadito

— **Module clamav** : modules\clamav_a6o
— **Module PDF** : modules\modulePDF
— **Module H1** : modules\moduleH1

Enfin, lancez la compilation (Run).

Le résultat de la compilation devrait se trouver dans le dossier suivant :

```
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\out\Debug
```

ou

```
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\out\Release
```

Si la compilation a été un succès, vous devriez avoir les fichiers correspondants :

Core :

```
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\out\[build_mode]\conf\armadito.conf
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\out\[build_mode]\glib-2-vs12.dll
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\out\[build_mode]\gmodule-2.vs12.dll
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\out\[build_mode]\gthread-2.vs12.dll
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\out\[build_mode]\libarmadito.dll
```

Module clamav :

```
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\out\[build_mode]\modules\clamav_a6o.dll
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\out\[build_mode]\libclamav.dll
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\out\[build_mode]\libeay32.dll
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\out\[build_mode]\ssleay32.dll
```

Module PDF :

```
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\out\[build_mode]\modules\modulePDF.dll
```

Module H1 :

```
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\out\[build_mode]\modules\moduleH1.dll
```

Armadito Windows Driver

Armadito Windows Driver est responsable de la protection en temps réel de l'antivirus Armadito.

Pré-requis

— Microsoft Visual Studio 2013 (édition communautaire ou plus)
— Armadito windows dependencies archive (deps-x.zip)
— Windows Driver Kit 8.1

Pour obtenir Windows Driver Kit 8.1 : <https://www.microsoft.com/en-us/download/details.aspx?id=42273>

Avertissement : Windows Driver Kit 8.1 va **seulement** avec MS Visual Studio 2013. Vous devez obtenir le WDK compatible avec votre version de Visual Studio.

Signature du driver

Ajout de votre certificat au magasin local

- Ouvrez l'outil Certificate Manager (certmgr.msc)
- Allez à **Certificats - Utilisateur actuel > Particulier > Certificats**
- Cliquez avec le bouton droit de la souris sur le dossier et choisissez **Toutes les tâches > Importer**
- Ensuite, suivez l'assistant pour importer votre certificat.

Signature avec votre certificat

- Ouvrez la solution Armadito-AV dans Visual Studio.
- Cliquez avec le bouton droit de la souris sur le projet **ArmaditoGuard** et sélectionnez **Propriétés**.
- Accédez à **Propriétés de configuration > Signature du pilote > Général**.
- **Signature mode > Signature du produit**.
- **Certificat de production > Sélectionnez dans le magasin** et sélectionnez votre certificat précédemment ajouté.
- Répétez les étapes précédentes pour le projet **ArmaditoGuard Package**.

Construire

Ouvrez la solution Armadito-AV à l'emplacement suivant :

```
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\Armadito-AV.sln
```

Premièrement, séléctionnez le projet **DriverArmaditoGuard** et lancez la compilation.

Puis, séléctionnez le projet **Driver\ArmaditoGuard Package** et lancez la compilation.

Enfin, séléctionnez le projet **Setup\ArmaditoGuard-setup** et lancez la compilation.

Le résultat de la compilation devrait se trouver dans le dossier suivant :

```
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\out\Debug
```

or

```
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\out\Release
```

Si la compilation a été un succès, vous devriez avoir les fichiers correspondants :

```
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\out\[build_mode]\driver\armaditoguard.cat
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\out\[build_mode]\driver\armaditoguard.inf
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\out\[build_mode]\driver\armaditoguard.sys
SOMEWHERE\armadito-av\build\windows\VS12\Armadito-AV\out\[build_mode]\driver\ArmaditoGuard-setup.exe
```

Armadito WebUI

Danger : L'interface utilisateur graphique est en cours de refonte et la nouvelle version n'est pas encore portée à Windows. Ce sera réglé dès que possible.

Pré-requis

Les outils nécessaires sont les suivants :
- node.js
- Git
- bower

Installer node.js et bower

Tout d'abord, téléchargez node.js depuis https://nodejs.org/en/download/, en utilisant l'installateur .msi ou le programme d'installation .exe, à votre choix.

Pendant l'installation, les choix de configuration par défaut sont OK.

Vérification de l'installation :

```
$ npm --version
2.15.1
```

Puis installez bower en utilisant :

```
$ npm install -g bower
```

Installer git

Pour utiliser bower, vous devez d'abord installer git.

Git pour windows est disponible ici : https://git-for-windows.github.io/

Vérification de l'installation :

```
    $ git version
2.8.1.windows.1
```

Installation des modules dans l'arborescence des sources

Utiliser bower à partir du répertoire **armadito-web-ui** pour installer les modules nécessaires :

```
$ cd SOMEWHERE/armadito-web-ui
    $ bower install
    $ npm install
```

Exécution de l'interface

Tout d'abord, le daemon Armadito doit être lancé.

Si c'est le cas, ouvrez votre navigateur Web et accédez à l'URL suivante :

http://localhost:8888/app/index.html

Débogage de l'interface

Une fois l'interface lancée :

— cliquez avec le bouton droit de la souris dans la fenêtre pour afficher le menu de débogage et sélectionnez «Inspecter» ou appuyez sur F12
— dans la fenêtre de l'inspecteur, sélectionnez l'onglet "console"

Compilation avec grunt

Installer grunt :

```
$ npm install -g grunt-cli
```

Exécuter *grunt* pour compiler et *grunt serve* pour l'aperçu.

Vous pouvez utiliser l'option "–force" si vous voulez compiler avec des warnings.

CHAPITRE 3

Installation

Installation sur Linux

Installation à partir de sources

Lorsque vous installez Armadito AV à partir de sources (soit tarball ou git clone), vous devez d'abord le compiler. Reportez-vous à la section **Compilation > Compilation sur Linux** de cette documentation pour obtenir des instructions détaillées.

Après la bonne configuration et la compilation des différentes parties (core, modules, webui), chaque partie est simplement installable par la commande suivante :

```
$ make install
```

Cela installera les bibliothèques, les outils, les fichiers d'en-tête ... dans les sous-répertoires du **PREFIX** défini au moment de la configuration.

Il faut prendre soin de bien configurer chaque partie avec le même **PREFIX** afin que les différents composants soient installés à leurs emplacements respectifs.

Installation à partir de paquets

Distributions Ubuntu

Les paquets pour les distributions Ubuntu sont disponibles à l'adresse suivante :

URL : <https://launchpad.net/~armadito/+archive/ubuntu/armadito-av>

```
sudo add-apt-repository ppa:armadito/armadito-av
```

Notez que ce PPA est expérimental et que l'interface utilisateur graphique n'est pas encore emballée de manière propre.

Installation sur Windows

Pré-requis

Packages redistribuables Visual C++ for Visual Studio 2013 :

— vcredist_x64.exe

— vcredist_x86.exe

URL : https://www.microsoft.com/fr-fr/download/details.aspx?id=40784

Installation avec l'installateur MSI

1. Téléchargez et installez les pré-requis (voir la section précédente)
2. Télécharger armadito-av msi :
   — Armadito-db-setup-0.10.0.msi
   — Armadito-setup-0.10.0.msi
3. Installer les bases de données des modules armadito-av :

   Lancez l'installeur Armadito-db-setup-0.10.0.msi
4. Installez armadito-av (service d'analyse + interface utilisateur grapique)

   Lancez le programme d'installation Armadito-setup-0.10.0.msi

Installation à partir de sources

1. Compilez les sources de l'antivirus à partir du projet de solution Visual Studio.

Suivez les instructions de la section **Compilation > Compilation sous Windows** de cette documentation.

2. Copiez les fichiers de bases de données du module dans le dossier adéquat :

```
SOMEWHERE\\armadito-av\\build\\windows\\VS\\Armadito-AV\\out\\[build_mode]\\modules\\DB
```

3. Installez le driver en exécutant la commande suivante dans une invite en tant qu'administrateur :

```
ArmaditoGuard-setup --install
```

4. Installez le service d'analyse en exécutant la commande suivante dans une invite de commandes en tant qu'administrateur :

```
ArmaditoSvc --installBoot (service lancé au démarrage de Windows)
```

ou

```
ArmaditoSvc --install (service lancé sur demande)
```

CHAPITRE 4

Configuration

Configuration sous Linux

La configuration de l'antivirus sur Linux est stocké dans deux fichiers :

— /etc/armadito/armadito.conf
— /etc/armadito/conf.d/on-access-linux.conf

Note : Si vous avez compilé à partir de sources, ces fichiers de configuration se trouvent dans votre répertoire **PREFIX**.

Analyse à la demande

Vous pouvez configurer le fonctionnement de l'analyse à la demande dans **/etc/armadito/armadito.conf** :

```
[on-demand]
white-list-dir = "/boot"; "/dev"; "/etc"; "/proc"; "/run"; "/sys"; "/var"
mime-types="*"
modules="clamav"; "moduleH1"
max-size = 10048576
```

— **white-list-dir** : Liste des répertoires exclus de l'analyse à la demande.
— **mime-types** : Types de fichiers MIME analysés lors d'une analyse à la demande.
— **modules** : Modules utilisés par l'analyse à la demande.
— **max-size** : Taille maximale des fichiers scannés lors de l'analyse à la demande.

Analyse en temps réel

L'analyse en temps réel d'Armadito AV s'appuie principalement sur l'API fanotify. Vous pouvez trouver plus d'informations sur la façon dont il fonctionne en lisant la page de manuel : *fanotify man7 <http ://man7.org/linux/man-pages/man7/fanotify.7.html> _*.

La configuration peut être effectuée en modifiant **/etc/armadito/conf.d/on-access-linux.conf** :

```
[on-access]
enable=1
enable-permission=1
enable-removable-media=1
mount="/home"
directory="/var/tmp"; "/tmp"
```

```
white-list-dir = "/bin"; "/boot"; "/dev"; "/etc"; "/lib"; "/lib32"; "/lib64"
mime-types = "application/x-executable"; "application/pdf"; "application/zip"
modules = "clamav"
max-size = 10048576
```

- **enable** : Activez (1) ou désactivez (0) l'analyse à l'accès.
- **enable-permission** : validation (1) ou désactivation (0) vérification d'autorisation.
  - Si **activé**, les fichiers détectés comme malveillants seront bloqués par Armadito AV.
  - Si **désactivé**, les fichiers détectés comme malveillants ne seront notifiés.
- **enable-removable-media** : active (1) ou désactive (0) la surveillance des médias amovibles.
  - Si **activé**, des points de montage de supports amovibles seront ajoutés à la volée à la liste de surveillance.
- **mount** : liste des répertoires qui seront surveillés par des points de montage. C'est à dire. En utilisant FAN_MARK_MOUNT.
- **directory** : liste des répertoires qui seront surveillés par un marquage récursif de tous les sous-répertoires.
- **white-list-dir** : liste des répertoires exclus de l'analyse à la demande.
- **mime-types** : Types de fichiers MIME analysés lors d'une analyse à la demande.
- **modules** : Modules utilisés par l'analyse à la demande.
- **max-size** : Taille maximale des fichiers scannés lors d'une analyse à la demande.

Alertes de virus

Lorsqu'un contenu malicieux est détecté par Armadito AV, un rapport d'alerte est généré et stocké dans un emplacement défini dans la configuration. Cela peut être configuré en modifiant **/etc/armadito/armadito.conf** :

```
[alert]
alert-dir = "/var/spool/armadito"
```

- **alert-dir** : dossier dans lequel les alertes seront stockées.

Quarantaine

Pour isoler les fichiers infectés, Armadito AV peut placer les fichiers détectés en quarantaine. **/etc/armadito/armadito.conf** contient la configuration de la quarantaine :

```
[quarantine]
enable = 0
quarantine-dir = "/var/lib/armadito/quarantine"
```

- **enable** : active (1) ou désactive (0) la quarantaine.
- **quarantine-dir** : répertoire où seront déplacés les fichiers mis en quarantaine.

Configuration sur Windows

La configuration de l'antivirus sur Windows peut être modifié dans le fichier suivant :

- **<install_dir>**\Armadito-av\conf\armadito.conf

Note : By default, **install_dir** is *C :\Program Files\Teclib*.

Analyse à la demande

Vous pouvez configurer le fonctionnement de l'analyse à la demande dans **<install_dir>\Armadito-av\conf\armadito.conf** :

```
[on-demand]
white-list-dir = "C:\\wl-dir1\\"; "C:\\wl-dir2\\"
mime-types="*"
modules="clamav"; "moduleH1"
max-size = 10048576
```

- **white-list-dir** : Liste des répertoires exclus de l'analyse à la demande (non encore implémentée).
- **mime-types** : Types de fichiers MIME analysés lors d'une analyse à la demande.
- **modules** : Modules utilisés par l'analyse à la demande.
- **max-size** : Taille maximale des fichiers numérisés lors de l'analyse à la demande.

Analyse en temps réel

Vous pouvez configurer le fonctionnement de l'analyse en temps réel dans **<install_dir>\Armadito-av\conf\armadito.conf** :

```
[on-access]
enable = 0
mime-types="*"
modules="clamav"
```

- **enable** : activer (1) ou désactiver (0) la protection en temps réel.
- **mime-types** : Types de fichiers MIME numérisés lors de l'analyse en temps réel.
- **modules** : Modules utilisés par la protection en temps réel.

Alertes de virus

Pas encore implémenté.

Quarantaine

Pour isoler les fichiers infectés, Armadito AV peut placer les fichiers détectés en quarantaine. **<dir_installation>\Armadito-av\conf\armadito.conf** contient la configuration de la quarantaine :

```
[quarantine]
enable = 0
quarantine-dir = "quarantine"
```

- **enable** : Activer (1) ou désactiver (0) la quarantaine.
- **quarantine-dir** : Sous-répertoire où seront déplacés les fichiers mis en quarantaine.

CHAPITRE 5

Contribuer

Etant donné qu'Armadito antivirus est un projet open-source, nous invitons grandement les développeurs, ingénieurs, partenaires, clients ou quiconque désireux de renforcer les technologies à collaborer à l'évolution du logiciel.

En devenant membre de la communauté, vous partagez vos idées et astuces, échangez vos connaissances et votre expertise avec d'autres membres afin d'assurer le développement de la solution Armadito.

Vous pouvez participer au projet de différentes façons, par exemple :

- installation de l'antivirus
- soumission des rapports de bugs
- suggestion de nouvelles fonctionnalités
- développement de modules

CHAPITRE 6

Licences

Armadito antivirus peut être divisé en plusieurs sous-projets, qui sont les suivants :

- Armadito core
- Armadito web ui
- Armadito systray ui
- Armadito windows driver
- Armadito module clamav
- Armadito module PDF
- Armadito module H1

Armadito core

- **Description** : Librairie principale de l'Antivirus.
- **License** : LGPLv3
- **Github** : armadito-av

Armadito web ui

- **Description** : Interface graphique en angularJS.
- **License** : GPLv3
- **Github** : armadito-web-ui

Armadito systray ui

- **Description** : Intégration dans la zone de notification.
- **License** : GPLv3
- **Github** : armadito-systray-ui

Armadito windows driver

- **Description** : Driver pour l'analyse en temps réel sur Windows.
- **License** : MS-PL
- **Github** : armadito-windows-driver

Armadito module clamav

- **Description** : Module d'analyse utilisant le moteur Clamav.
- **License** : GPLv3
- **Github** : armadito-mod-clamav

Armadito module PDF

- **Description** : Module d'analyse des fichiers PDF.
- **License** : GPLv3
- **Github** : armadito-mod-pdf

Armadito module H1

- **Description** : Module d'analyse heuristique des fichiers binaires (PE/ELF). Basé sur le travail du projet DAVFI (http://www.davfi.fr/)
- **License** : GPLv3
- **Github** : armadito-mod-h1

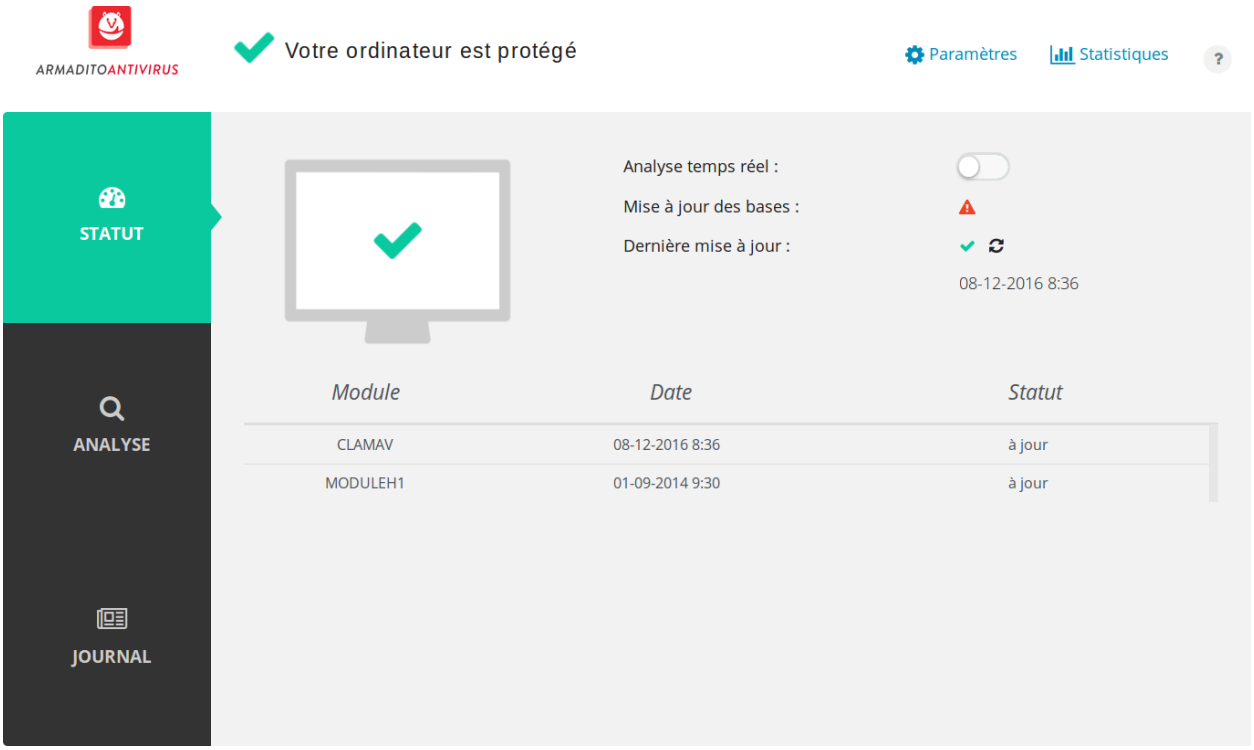
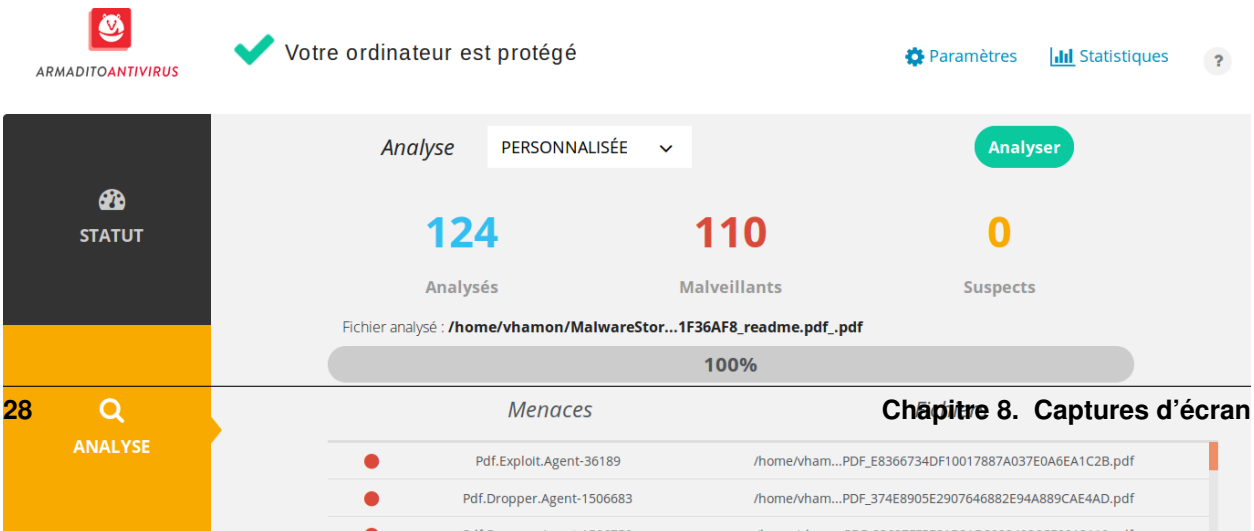
CHAPITRE 7

FAQ

CHAPITRE 8

Captures d'écran

Le tableau de bord



L'interface de scan