APAF Documentation

Release 0.1

Michele Orrù

November 22, 2012

CONTENTS

APAF, aka Anonymous Python Application Framework, is a multi-platform *build system* framework and a *library* for developing Python/Twisted based server applications, exposed as Tor Hidden Service, easy to be installed and managed on multiple platforms (Windows, OSX, Debian) with a particular focus for desktop environments.

CHAPTER

ONE

METADATA

Author: Michele Orrù, aka maker

Project: Google Summer of code 2012

Further informations:

- **tor-dev mailing list:**
  - project proposal http://archives.seul.org/or/dev/Apr-2012/msg00031.html
  - status reports: http://archives.seul.org/or/dev/Jun-2012/msg00014.html
- wikis and any other huge description of the apaf project

CHAPTER

TWO

REFERENCES

1. Tor blog
2. Any other site mentioning apaf

CHAPTER

THREE

FEATURES

APAF is a python library based on **twisted** and **txtorcon**. It can be used for launching a standalone web application exposing itself via hidden service, or as server application.

Multiplatform. APAF is tested on *Windows XP*, *Ubuntu 12.04* and *Mac OSX*.

Portable. Apart from the python package itself, APAF can be built as `.app`, using py2app, and `.exe`, using py2exe for windows.

Easy. exposing your application with APAF is as simple as writing a class and 3 clicks (ehi, I am still working on this). A good starting point may be the `staticfileserver.py` example.

Secure We make sure that all APAF's outbounding connecction will pass through Tor, and that its fingerprint is reduced to a minimum.

> **Warning:** we *do not guarantee* that the application built with APAF will not leak data outside tor, neither the same application will make APAF easily recognizable. The developer building applications with APAF should properly audit the applications they are packaging to check for possible leaks.

See Also:

See the threat model *page* for further informations.

CHAPTER

FOUR

INSTALLATION

APAF can be installed using either `pip` or `easy_install`. There should be somewhere also builds for debian.

```
$ pip install git+https://github.com/globaleaks/APAF.git
```

CHAPTER

FIVE

RUNNING

To try out apaf, just run:

```
$ python apaf/main.py
```

In case you are not interested in user-experience and similar bullshits, `--debug` option could be helpful.

CHAPTER

SIX

CODING

Follows below the table of libraries.

6.1 APAF package

6.1.1 File System Structure

APAF comes with a very simple *file system structure*.

```
- apaf/
  |- test/
- datadir/
  |- contrib/
  |- config/
  |- services/
- docs/
```

Apart from the standard package directory `apaf/` , which will be covered further in *APAF utilities*, and `docs/` that is the directory you are currently reading from, `datadir` contains everything for building a new apaf `apaf.core.Service`.

`contrib/` holds every bundles binary provided with the final application. If you are going to use external binary tools, you can place them here and access afterwarss via `apaf.config.binary_kits`

`config/` contains every configuration file previously created with `apaf.config.Config`.

`services/` is the root directory for external services. By default, it contains some examples, but feel free to change those :D.

6.1.2 Utility classes

If you are a web application developer interested in using APAF for exposing its application to the dark net with a .onion domain, that's the right place for you.

APAF communicates with external application using the class *apaf.core.Service*, which exports some metadata such as name *apaf.core.Service.name*, descirption *apaf.core.Service.desc*, exc.; serves the twisted's factory class *twisted.internet.protocol.factory*; and has its own configuration file in yaml syntax.

At the bottom of this document you will see some informations about also *apaf.testing* module. It is supposed to be used from each new application inside its `'test/'` directory.

Services implementation

Services are object exposing informations about how they need to be set up (callbacks), their metadata (class attributes) and user configuration (service.config static attribute)

Configuration helper

Each *apaf.core.Service* class has its *apaf.config.Config* instance, which interfaces to pyYaml for provinding a simple configuration file, writted in a human-readable format such is yaml and organized with default fields.

class `apaf.config.`**`Config`**(*config_file*, *defaults*)
Configuration class

`reset`()
Restores default configuration.

Testing

Cyclone, unlike tornado, does not provide any testing module. APAF tries to meet unit testing needs providing `apaf.testing`. In this module you will find some decorators and some helper functions.

Decorators

`@apaf.testing.`**`Page`**

`@apaf.testing.`**`json`**

Functions

`apaf.testing.`**`start_mock_apaf`**()

6.1.3 JSON APIs

GET /services # returns the list of running services

GET /services/<servicename> # returns informations about the service <servicename> # 404 if <servicename> was not found

GET /services/<servicename>/start , GET /services/<servicename>/stop # Start/stop a service. Returns a json object {'result': state} where state # is a boolean

GET /config # return a key:value dictionary for each item in the config file.

PUT /config ## XXX must be post. ### # given a dictionary {key:value}, substitute each item with name *key* in # the dictionary with *value*. # In case of error, return a json object {'error':message}

GET /tor/<spkey> # Wraps the command GETINFO <spkey> to the controlport. # return 404 in case the command is not valid, a json object {'error':message} # otherwise

POST /auth/login # oAuth login - not implemented.

GET /auth/logout # logout

6.2 Threat Model

21:15 < rransom> What security properties should APAF provide? What attacks (or classes of attacks) should APAF prevent or resist? This section describes the classes of attacks the APAF should prevent/resist.

6.2.1 The Application

present briefly the application and describe how the user shall interact with it The Anonymous Python Application Framework is built and delivered as a standalone application, and consists in a simple static file server.

Double clicking the executable, a new browser tab will show the configuration page, on which the user can select the destination folder and edit advanced options.

Entry Points: Hidden Service port selected in the configuration page, telnet login? Flowing Data: documents selected from the user

6.2.2 Key Scenarios

who is going to use the application?

The application may be used from:

- a generic anonymous user intending not to share its identity;
- an anonymous activist;
- a non-profit organization with a low budget;
- an anonymous user interested in sharing documents but unable to host a server

6.2.3 Vulnerabilities

a list of the vulnerable corners of the application.

- gain access to the configuration page on the login may lead to serving system dirrectories;
- possible xss;
- executable infection;
- data leakage outside tor;
- *read twisted documentation, which kind of authentication does twisted support?*

6.2.4 Attacks

the attack a malicious user may perform

- bruteforce over the login form;
- the .exe/.app contains, compressed, all the python standard library in pyc format. Replacing one of these byte-code libraries may lead to the control of the applciation.
- denial of service

6.2.5 Security Controls

Precautions for attacks As far as I know, -onion hostnames, by thir own, provide a secutipry mechanisms to avoid poisoning or man in the middle attacks.

The user shall be advised with very clear messages in the configuration page about the consequences of editing a certain box.

6.3 Setting up the APAF build System on Mac OS X

This tutorial winn guide you through the installation of the apaf and its dependencies on a Mac Os X 10.6 environment

Warning: This tutorial has been tested only on Mac OS X 10.6 and 10.7 (by mogui) .

Note: This tutorial will start assuming you are on a clean environment. If you have already installed Python, you may consider start reading from *Download APAF*

6.3.1 Install GnuPG

Install GnuPG as a tool to to verify the various software download:

https://github.com/downloads/GPGTools/GPGTools/GPGTools-20120318.dmg

6.3.2 Install Python

Download Python 2.7 for Mac Os X from http://www.python.org/ftp/python/2.7.3/python-2.7.3-macosx10.6.dmg

Verify signature of application from http://www.python.org/ftp/python/2.7.3/python-2.7.3-macosx10.6.dmg.asc .

Install the software following the wizards.

6.3.3 Install Setuptools and pip

Download setup tools:

```
wget http://pypi.python.org/packages/source/s/setuptools/setuptools-0.6c11.tar.gz
tar xvzf setuptools-0.6c11.tar.gz
cd setuptools-0.6c11
python2.7 setup.py install
```

Install Pip: :: python2.7 /Library/Frameworks/Python.framework/Versions/2.7/bin/easy_install-2.7 pip

6.3.4 Install Git

Since github lets you download a simple *.zip* of the latest revision of your application, git is not indispensable. But certainly it will be comfortable to stay up to date with the software development

http://git-scm.com/download/mac

6.3.5 Extract Tor binary

In order to extract the Mac OS X tor's binary we need to download TBB that's packaged as a zip file:: cd APAF/datadir/contrib/ wget –no-check-certificate https://www.torproject.org/dist/torbrowser/osx/TorBrowser-2.2.35-12-osx-i386-en-US.zip

Then extract the Tor binary with the following command line by using 7zip for OSX:: $ unzip TorBrowser-2.2.35-12-osx-i386-en-US.zip

Then move the binary in the current directory:: $mv TorBrowser_en-US.app/Contents/MacOS/tor .

Obtaining APAF

APAF has not stable versions yet. You can download the latest revision from git at:

```
$ git clone https://github.com/Globaleaks/APAF.git
```

Onnce downloaded, cd into `apaf` and install its dependencies. :: cd apaf pip -r requirements.txt

6.3.6 Build Apaf Application

:: cd ../../ python2.7 setup.py py2app

Now in dist/ you will find "apaf.app"

6.4 Setting up the APAF build System on debian

install debian build tools http://ghantoos.org/2008/10/19/creating-a-deb-package-from-a-python-setuppy/ http://wiki.debian.org/Python/Packaging

apt-get install build-essential dpkg-dev debhelper devscripts fakeroot

install apaf dependencies

$ apt-get install tor python-twisted python-pip $ pip install pygeoip ipaddr pyYAML

checkout branch debian

build debian with

$ python setup.py sdist -d .. update changelog (how?)

$ debuild

6.4.1 Updating relase

Use *debchange* to change your changelog with *debchange -a*, then make a new release *debchange -r* Build apaf with *python setup.py sdist* and, after being sure to have set up you gpg configuration, make the debian package with *$ dpkg-buildpackage -i -Ifakeroot*.

Done!

6.5 Setting up the APAF build System on Windows

This tutorial winn guide you through the installation of the apaf and its dependencies in a Windows environment.

Warning: This tutorial has been tested only on Windows XP sp3 and Windows 7.

Note: This tutorial will start assuming you are on a clean environment. If you have already installed Python, you may consider start reading further.

6.5.1 Requirements

- Python 2.7.3 http://www.python.org/download/releases/2.7.3/
- Twisted 12.0 http://twistedmatrix.com/trac/
- Setuptools 0.6-c11 http://pypi.python.org/pypi/setuptools
- Psutils 0.4.1 http://code.google.com/p/psutil/
- Py2exe 0.6.9 http://www.py2exe.org/
- Six 1.1.0 http://pypi.python.org/pypi/six
- PyGeoIP 0.2.3 http://code.google.com/p/pygeoip/
- Ipaddr 2.1.10 http://code.google.com/p/ipaddr-py/
- PyWin32 Build 20217 http://pypi.python.org/pypi/pywin32
- PyYAML 3.10 http://pyyaml.org/wiki/PyYAML
- 7zip 9.20 http://downloads.sourceforge.net/sevenzip/7z920.exe
- Gpg 4 win 2.1.1 http://www.gpg4win.org/download.html [in future]
- Git 1.7.3+ http://git-scm.com/download/win

6.5.2 Install GnuPG

Install GnuPG as a tool to to verify the various software download:

http://files.gpg4win.org/Beta/gpg4win-2.1.1-34299-beta.exe

6.5.3 Install Python

Download Python 2.7 from http://www.python.org/ftp/python/2.7.3/python-2.7.3.msi

Verify signature of application: http://www.python.org/ftp/python/2.7.3/python-2.7.3.msi.asc

Install the software following the wizard.

6.5.4 Install Setuptools

Download http://pypi.python.org/packages/2.7/s/setuptools/setuptools-0.6c11.win32-py2.7.exe#md5=57e1e64f6b7c7f1d2eddfc9746bbaf20

6.5.5 Install Pip

cd C:Python27Scripts C:Python27Scripts> easy_install.exe pip

6.5.6 Install psutil

Required for txtorconn

Download from http://psutil.googlecode.com/files/psutil-0.4.1.win32-py2.7.exe

6.5.7 Install Py2Exe

Url for py2exe: http://sourceforge.net/projects/py2exe/files/py2exe/0.6.9/py2exe-0.6.9.win32-py2.7.exe/download

6.5.8 Install PyWin32

Url for pywin32: http://sourceforge.net/projects/pywin32/files/pywin32/Build%20217/pywin32-217.win32-py2.7.exe/download

6.5.9 Install Twisted

Donwload from http://pypi.python.org/packages/2.7/T/Twisted/Twisted-12.0.0.win32-py2.7.msi

6.5.10 Install Zope.interface

Warning: Installing zope.interface with pip may lead to ImportError in building the APAF with py2exe.

Note: Tests on windows 7 show that, since easy_install behaves differently from pip.exe, using one instead of another during the setup of the environment may lead to problems afterwards when building the executable.

Install *zope.interface* using setuptools: :: C:Python27Scripts> easy_install.exe zope.interface

6.5.11 Install Six

cd C:Python27Scripts C:Python27Scripts> pip.exe install six

6.5.12 Install pygeoip

cd C:Python27Scripts C:Python27Scripts> pip.exe install pygeoip

6.5.13 Install ipaddr

cd C:Python27Scripts C:Python27Scripts> pip.exe install ipaddr

6.5.14 Install pyYAML

cd C:Python27Scripts C:Python27Scripts> pip.exe install pyYAML

6.5.15 Install Git

Since github lets you download a simple *.zip* of the latest revision of your application, git is not indispensable. But certainly it will be comfortable to stay up to date with the software development

http://git-scm.com/download/win

Then open a new Git shell from *Start>Git>Git Bash*.

6.5.16 Install Txtorcon

Txtorcon is not avaible on the Python Package Index, so you need to install it manually with git.

$ git clone https://github.com/meejah/txtorcon.git

Then install with pip: :: cd C:Python27Scripts C:Python27Scripts> pip.exe install C:pathoftxtorcon

6.5.17 Install Apaf

Download Apaf from Github:

$ git clone https://github.com/mmaker/APAF.git

6.5.18 Install 7zip

Download http://downloads.sourceforge.net/sevenzip/7z920.exe and install following the wizard.

It will place 7z.exe in "c:Program Files7-Zip7z.exe"

6.5.19 Extract Tor binary

Download the latest version of Tor binaries for Windows.

Go to download page https://www.torproject.org/download/download.html.en and download "Expert Bundle": https://www.torproject.org/dist/win32/tor-0.2.2.35-win32-1.exe

Now decompress the tor binary with 7zip and move it to contrib/ directory of APAF:

c:Program Files7-Zip7z.exe x tor-0.2.2.35-win32-1.exe tor.exe move tor.exe PATH_WHERE_IS_BUILD_ENVIRONMENT/contrib

6.5.20 Build Apaf Application

Here you are ready to use the apaf. To build the single *.exe* file, run

```
C:\path\of\user\APAF> C:\Python27\python.exe setup.py py2exe
```

6.6 Indices and tables

- *genindex*
- *modindex*
- *search*
- The Tor Project