**Tencent IoT Hunter**

**User Guide**

**（2018.11.28）**

# Contents

[Contents 2](#_Toc27170)

[I. Introduction 3](#_Toc18058)

[II. Framework features 3](#_Toc19754)

[a) Good extension interface 3](#_Toc19886)

[b) Fine-grained information extraction 4](#_Toc5927)

[c) Third-party intelligence aggregation 4](#_Toc290)

[III. Compatibility 4](#_Toc14765)

[IV. Information Extraction details 5](#_Toc3977)

[a) Static Information 5](#_Toc9905)

[b) Dynamic Information 6](#_Toc4469)

[c) Result 6](#_Toc4388)

[V. How to use 8](#_Toc8214)

[a) Static Information Extraction Tool 8](#_Toc8047)

[i. Setting parameters using the command Line 8](#_Toc23108)

[ii. Use config file: conf.py 9](#_Toc1754)

[iii. Configuration File (conf.py) Parameter description 9](#_Toc4806)

[b) Dynamic Analysis Tool 11](#_Toc32549)

[i. Analysis Environment Setup 11](#_Toc31252)

[ii. Analysis Tool Configuration 11](#_Toc14290)

[iii. Dynamic Analysis Tool Usage 12](#_Toc26339)

[VI. Use Extensible Plugins 13](#_Toc1279)

[a) Write Static Analysis Plugin 13](#_Toc15410)

[b) Static Plugin Debug 15](#_Toc26551)

[c) Write Dynamic Plugin 15](#_Toc8379)

[VII. Data visualization 16](#_Toc30245)

[a) ES Data Query 16](#_Toc6901)

[b) Weak Password Tag Cloud 16](#_Toc2578)

[c) C&C Geographic location visualization 17](#_Toc32257)

[d) Malware Variants Mining 18](#_Toc4239)

[VIII. Tools Used 18](#_Toc14147)

[IX. Future Work 19](#_Toc27584)

# Introduction

Tencent IoT Hunter is a framework tool which is developed to gather IoT threat intelligence. It focus on the whole IoT malware analysis life cycle in all directions through static information extraction, dynamic information extraction, and third-party network platform information. For security researchers it is much easier to do IoT malware analysis, research, track threat by using this intelligence framework.

Using this framework tool, the malicious information (CNC, Domain, function, etc.) within the IoT sample can be accurately and fine-grained obtained, which can be directly used to build the IoT threat cloud detection service. The advantage is that there is no need for analysts to reconfirm the malicious information, which greatly improves the efficiency of malicious information processing.

The framework provides a good extension interface, users can write plug-ins to extend the scope of information extraction to enhance this tool. Through the extracted information, we can quickly build a IoT threat intelligence platform, quickly visual analysis, mining IoT threat families, variants and monitor threat situation.

# Framework features

## Good extension interface

This framework provides an extension interface for security professionals using the framework to write their own plugin, to extend the scope of information extraction.

In order to facilitate the user to write their own plugin, framework provides a foundation class, the new plugin can be directly implemented by inheriting the base class, without the need to write additional code, so that users can focus on how to do the information extraction of the target sample.

At the same time, the framework records the detailed log. It is very easy for the user to debug the code by viewing the log file.

when writing plugin, analysts can define their own sample family type, propagation approach, target attack device, attack method, and can accurately obtain malicious information such as CNC, Domain, weak password dictionary, control commands and so on.

## Fine-grained information extraction

In the past, some analysis tools will extract coarse-grained information from samples, and security researchers need to make a second confirmation of whether it is malicious or not before using the information.

For example, a coarse-grained IP, is extracted from a sample, but it is not known whether the IP is malicious, so it cannot be used directly as a malicious IP, and the IP needs to be further confirmed if it is malicious.

But this framework tool use accurate feature matching method, which can accurately extract malicious information, the malicious information can be used directly without re-confirmation.

## Third-party intelligence aggregation

This framework attempts to get more IoT intelligence information through the open third-party network intelligence platform in order to provide more valuable reference information for users.

# Compatibility

* Python2, Python3
* Windows, Linux, （OS X not currently tested）

# Information Extraction details

This framework supports static and dynamic information extraction of ELF files on ARM, X86, X64, MIPS, Sparc, PowerPC platforms.

## Static Information

**Defined information：**

* Virus Name
* Malware Type
* Family information
* Spreading Method
* Target Device
* Main attacking method

**Information extraction**

* Basic Information (file size, file type, platform, md5, sha1, sha256)
* C&C Address
* Domain
* IP
* URL
* UDP
* TCP
* DNS
* Malware Configuration
* Weak Password Dictionary
* All Strings
* Suspicious Strings
* all function Names
* Control Commands
* Packer information

## Dynamic Information

**Process information:**

* Process EXECVE: parameter information
* Process Clone

**File Operation information:**

* File open: file name
* File read: read data
* File write: writing data

**Socket Information:**

* Connect: ip
* Recvfrom : ip, data
* Sendto : ip, data
* Bind : ip

**Network communication information:**

* Network packets: ip, protocol
* HTTP Information: host, data
* TCP Information: ip, data
* UDP Information: ip, data
* IRC Information: ip,IRC Message

**Dynamic Analysis plug-in information:**

* Plug-in Name.
* plug-in Analysis results

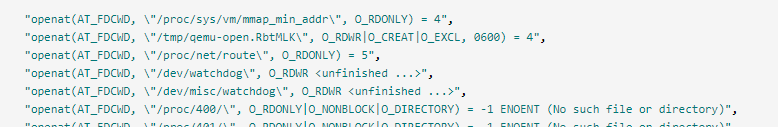
## Result

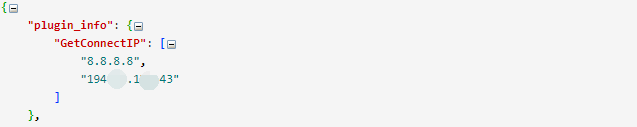
All analysis results are saved in the results file as json format:

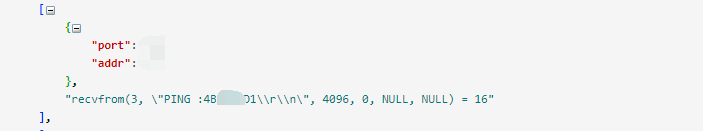
**Static Analysis Results JSON:**

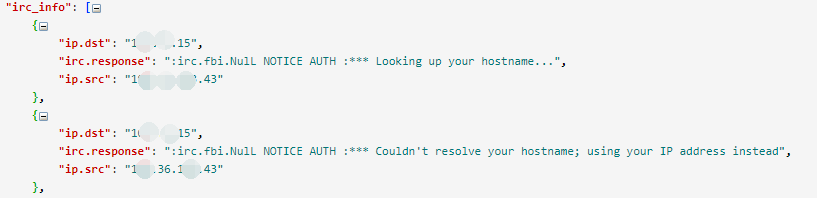


**Dynamic Analysis Results JSON:**









# How to use

## Static Information Extraction Tool

This tool can be set parameters using the command line or by a configuration file.

iot\_hunter.py -h

usage: iot\_hunter.py [-h] [-s SAMPLE\_DIR | -f SAMPLE\_PATH] [-o OUTPUT\_DIR]

[-v] [-c]

Tencent IoT Hunter

optional arguments:

-h, --help show this help message and exit

-s SAMPLE\_DIR samples folder path for analyzing.

-f SAMPLE\_PATH singal sample path for analyzing.

-o OUTPUT\_DIR output folder path for saving analysis result and log files.

-v, --virustotal try to get the sample info from VirusTotal.

-c, --clean clean result files,save all results to

result\_file\_detail\_info.txt.

### Setting parameters using the command Line

**Single File Analysis:：**

iot\_hunter.py -f F:\Samples\0019c77ad7f4f97ec492726e9aa8e15e -o F:\result

**Multi files Analysis:**

iot\_hunter.py -s F:\Samples -o F:\result

Sample Dir: F:\Samples

Output Dir: F:\result

F:\Samples\0019c77ad7f4f97ec492726e9aa8e15e

F:\Samples\2983a7e5bc97996cd98dffd4f78e95b2

Packed by UPX.

F:\Samples2989b5de79e0ab4417c10b64738a10a0

**Get VirusTotal Information:**

iot\_hunter.py -v -s F:\Samples -o F:\result

**Load results to Elasticsearch:**

import\_data\_to\_es.py -r F:\result\result\_ida\_file\_analysis.txt

### Use config file: conf.py

MAL\_SAMPLES\_DIR = r"F:\Samples"

RESULT\_OUTPUT\_DIR = r"F:\result"

Run python script:

python iot\_hunter.py

### Configuration File (conf.py) Parameter description

**Parameters that must be set:**

IDA PRO Path:https://www.hex-rays.com

IDA\_EXECUTABLE\_FILE\_PATH = r"C:\Program Files (x86)\IDA 6.5\idaq.exe"

**On-demand configuration:**

Samples Directory

MAL\_SAMPLES\_DIR = r"F:\Samples"

Result Output Directory

RESULT\_OUTPUT\_DIR = r"F:\result"

UPX Tool Path:https://github.com/upx/upx-testsuite

UPX\_EXECUTABLE\_FILE\_PATH = r"f:\tools\upx\i386-win32.pe\upx-3.95.exe"

VirusTotal key:https://www.virustotal.com

VIRUSTOTAL\_KEY = "676xxxxxxxxxxxxxxxxxxxxxxxxxxxxx7d1db"

If visit VirusTotal need Proxy,Please use the following code

PROXIES = {"http": "proxy.xxxx.com:8080", "https": "proxy.xxxx.com:8080"}

If you always try to get VirustTotal Information,Please set this parameter to True

VIRUSTOTAL\_ALWAYS\_GET = False

File Size Limit

FILE\_SIZE\_LIMIT = 10 \* 1024 \* 1024

Elasticsearch parameters:https://github.com/elastic/elasticsearch-py

ES\_HOST = "localhost:9200"

ES\_INDEX\_NAME = "iot\_threat"

ES\_TYPE\_NAME = "FileAnalysis"

**Default configuration:**

IDA Script Analysis File Result

IDA\_FILE\_ANALYSIS\_RESULT = r"result\_ida\_file\_analysis.txt"

VirusTotal Information File Name

VIRUSTOTAL\_RESULT = r"result\_virustotal.txt"

Summary of all results（IDA + VirusTotal）,Need use -c in commmand line.

FILE\_DETAIL\_INFO = r"result\_file\_detail\_info.txt"

Log File Configuration

IDA\_ANALYSIS\_LOGGER\_NAME = "IDA\_ANALYSIS\_FILE"

IDA\_FILE\_ANALYSIS\_LOG = r"log\_ida\_file\_analysis.log"

IOT\_HUNTER\_LOGGER\_NAME = "IOT\_HUNTER\_MAIN"

IOT\_HUNTER\_LOG = r"log\_iot\_hunter.log"

OTHER\_ERROR\_LOG = r"log\_other\_error.log"

ES\_LOGGER\_NAME = "IMPORT\_DATA\_TO\_ES"

ES\_IMPORT\_DATA\_LOG = "log\_import\_data\_to\_es.log"

Parameters used internally, try not to modify them.

If you want to modify, you need to modify both the associated file name and the directory name

IDA\_PYTHON\_SCRIPT = "ida\_analysis\_file.py"

IDA\_PLUGINS\_DIR\_NAME = "plugins"

## Dynamic Analysis Tool

### Analysis Environment Setup

The dynamic analysis environment needs to run the IOT sample in the virtual machine environment and monitor its behavior. For safety the IOT sample execution environment needs the virtual machine. This dynamic analysis tool is based on the VirtualBox. One Linux guest VM should be installed such as (Ubuntu).

Guest VM Installation:

1. Linux System,VBoxGuestAdditions\_5.2.22
2. QEMU,Use QEMU User Mode to emulate ARM, MIPS, PowerPC Iot files in x86/64 platform
3. Strace:monitor sample behaviors information
4. Tcpdump: capture network packets for analysis
5. Clean System with above tools, please use root account and save a snapshot named "analysis"

Host OS tools:

1. Tshark:To analyze packets file capture by tcpdump

### Analysis Tool Configuration

Configuration File:DynamicConfig.conf

[guest\_vm]

# guest os configuration, username, password,vm name

name=ubuntu11.04

username=root

password=root

#sample to run path

runpath=/home/root/Desktop/

#host os path to put strace log and tcpdump pcap file

host\_log\_path=f:\vm\_share\strace.log

host\_log\_tcpdump=f:\vm\_share\tcpdump.pcap

#guest os path to put strace, tcpdump, guestanalyzer.py

vm\_log\_path=/home/justin/Desktop/strace.log

vm\_log\_tcpdump=/home/justin/Desktop/tcpdump.pcap

guest\_analyzer\_path=/home/justin/Desktop

[vbox]

virtualbox\_path = D:\Program Files\Oracle\VirtualBox

[analyzer]

max\_strace\_lines=20000

strace\_log\_path=f:\vm\_share\strace.log

tshark\_path=c:\Program Files\Wireshark\tshark.exe

host\_log\_tcpdump=f:\vm\_share\tcpdump.pcap

### Dynamic Analysis Tool Usage

**Single File Analysis:** IotHunterDynamic.py –f filename –d logdir

**File Directory Analysis:**IotHunterDynamic.py –f directory –d logdir

IotHunterDynamic.py -h

usage: IotHunterDynamic.py [-h] [-f FILENAME] [-d FILE\_DIR] [-o OUT\_DIR]

optional arguments:

-h, --help show this help message and exit

-f FILENAME, --filename FILENAME

File to Analyze

-d FILE\_DIR, --file\_dir FILE\_DIR

Files directory to Analyze

-o OUT\_DIR, --out\_dir OUT\_DIR

log output directory

**Analysis Steps:**

1. Get Sample to Analyze
2. Start Analysis VM
3. Send file to VM
4. Send Analysis Tool to VM
5. Run Target File, monitor behaviors
6. Fetch results from VM
7. Analyze log to get file,network,process information
8. Apply user plugins
9. Generate final json report

# Use Extensible Plugins

## Write Static Analysis Plugin

Users can write their own information extraction plugin, then put the plugin in plugins directory, which can be directly executed.

The framework provides the base class PluginParent, which provides the basic information:

class PluginParent():

malicious\_type = []

malicious\_family = []

spread\_way = []

attack\_device = []

main\_function = []

cnc = []

ip = []

domain = []

url = []

udp = []

tcp = []

dns = []

configuration = []

weak\_password = []

suspicious\_string = []

bot\_command = []

other\_info = []

detect = ENUM\_DETECT\_RESULT["UNKNOW"]

virus\_name = ""

\_\_metaclass\_\_ = ABCMeta

@abstractmethod

def analyze(self, \*argv):

return False

Plugin needs to inherit the PluginParent class and implement the analyze function, which populates the required fields by calling add\_plugin (derived class name) at the end of the code:

import re

from util import \*

class MiraiARM(PluginParent):

def \_\_init\_\_(self):

self.malicious\_type = ["Botnet"]

self.malicious\_family = ["Mirai"]

self.spread\_way = ["SSH", "Telnet"]

self.attack\_device = ["Router", "Camera", "DVR", "Printer", "TV Box"]

self.main\_function = ["DDoS", "Downloader"]

self.virus\_name = "Trojan.Linux.Mirai.caa"

self.configuration = []

self.weak\_password = []

self.cnc = []

self.detect = 0

def analyze(self, \*argv):

self.get\_configuration(key)

self.get\_cnc()

self.get\_weak\_password(key)

...

def get\_cnc():

...

def get\_weak\_password(self, key):

...

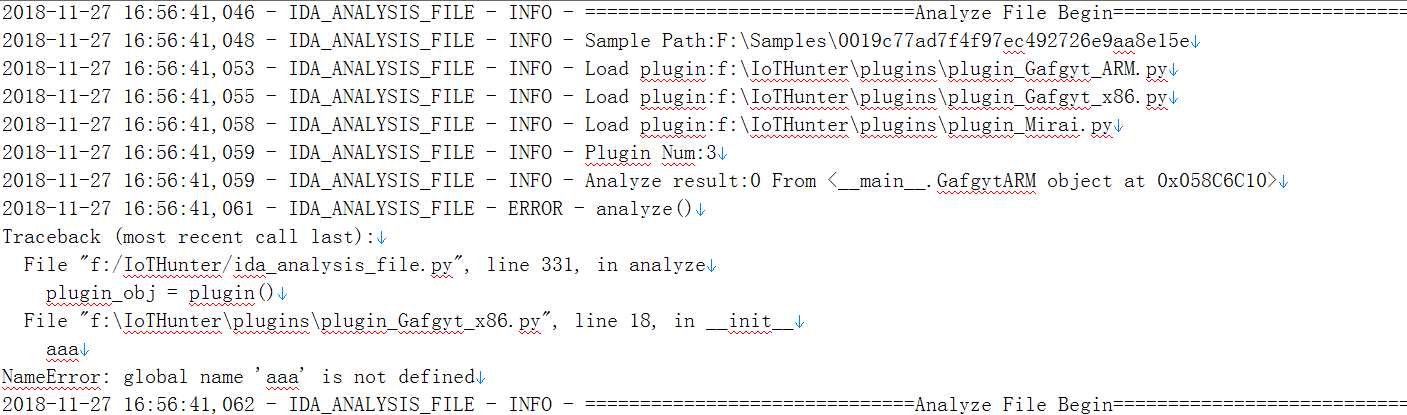
add\_plugin(MiraiARM)

## Static Plugin Debug

Because IDAPython command line calls automatically exits when it encounters an exception, and does not save code exception information, it is very inconvenient for users who write plugin to locate their own code problems.

This framework provides the log record, user can view the log to locate the exception problems.

log\_ida\_file\_analysis.log：



User can use logger.info("xxx") to print own log information.

## Write Dynamic Plugin

Users can write their own dynamic plugin, put the plugin in DynamicPlugins directory, which can be directly executed.

Plugin development: Plugin need to implement analyze and get\_result two interfaces, analysis framework will call all plugins, and generate plug-in results. Function analyze parameter behaviors records the behavior information of the sample. Users can perform custom analysis to obtain wanted results.

Plugin Example:Get All Connected IP

class GetConnectIP():

"""plugin to get connect ip list"""

def \_\_init\_\_(self):

self.ip\_list = []

def analyze(self, behaviors):

hit = 0

for data in behaviors.socket\_log['connect']:

hit = 1

addr = data['addr']

if addr not in self.ip\_list:

self.ip\_list.append(data['addr'])

return hit

def get\_result(self):

return self.ip\_list

# Data visualization

The framework provides the function of importing analysis data into Elasticsearch, and users can quickly build a IoT data mining platform to carry out data visualization analysis. IoT family variants, spread methods, activity, new weak password can be quickly mining.

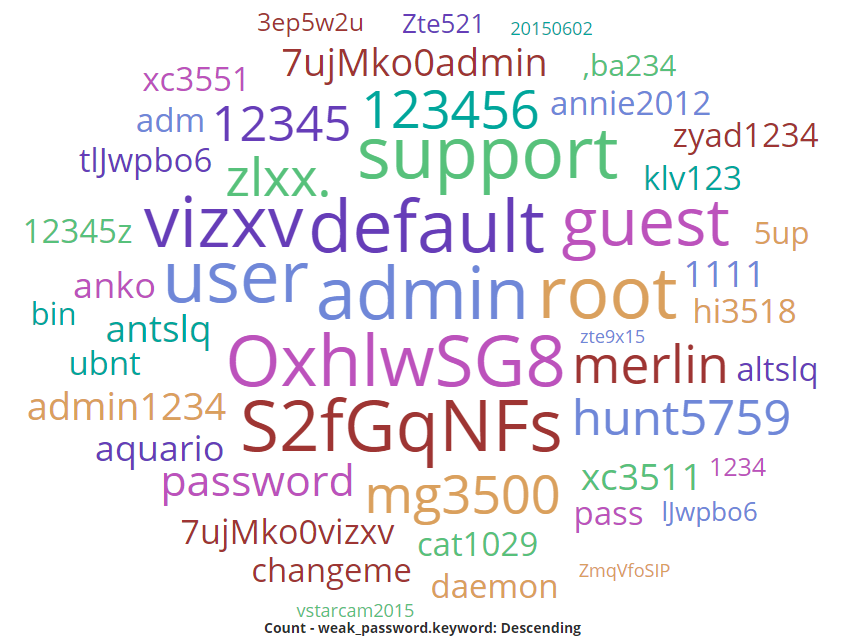
## ES Data Query



## Weak Password Tag Cloud

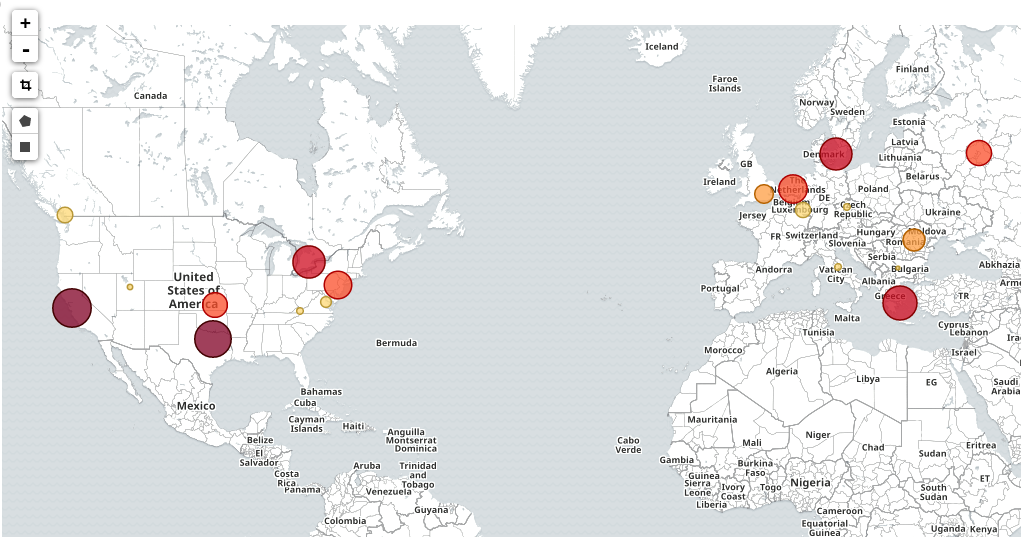
By extracting the weak passwords used in the IoT sample, we can observe which weak passwords are used most frequently.

To monitor all known weak passwords, when a new weak password is found, it is very likely that a new variant or a new weak password vulnerability appears.



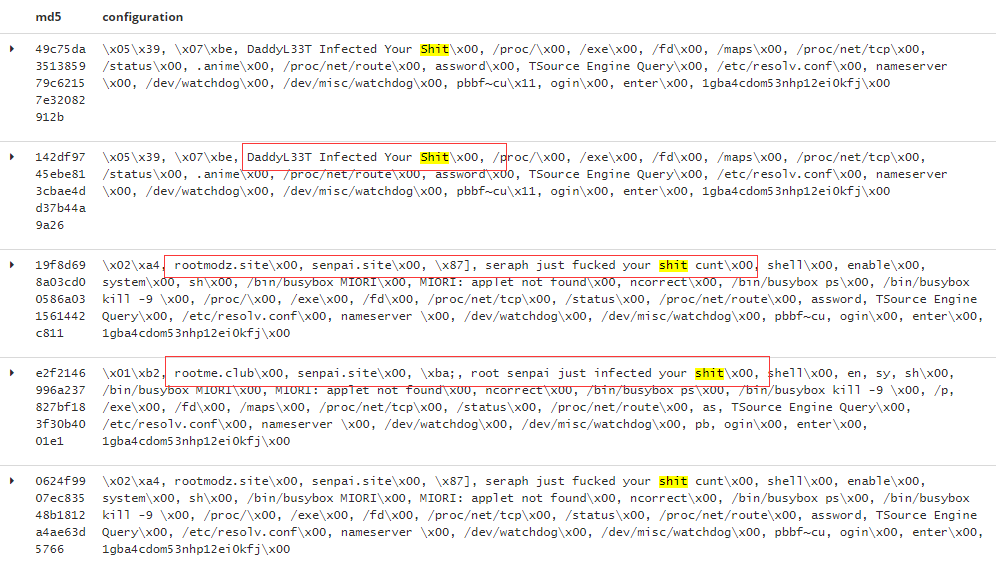
## C&C Geographic location visualization

By extracting the CNC address, the address location associated with the IoT family can be obtained.



## Malware Variants Mining

Through some keywords to carry on the mining, different malware family can be clustered.



# Tools Used

* IDA(https://www.hex-rays.com)
* IDA Python(https://github.com/idapython/src)
* UPX(https://github.com/upx/upx-testsuite)
* VirusTotal(https://www.virustotal.com)
* Elasticsearch(https://github.com/elastic/elasticsearch-py)
* Kibana(https://www.elastic.co/downloads)
* QEMU(https://www.qemu.org/)
* VirtualBox(<https://www.virtualbox.org/>)
* Strace(<https://strace.io/>)
* Tcpdump(<http://www.tcpdump.org/>)
* Wireshark(https://www.wireshark.org/)

# Future Work

* Include More plugins
* Extend capabilities to monitor new variants
* Include More third-party platform information
* Include More Behavioral Monitoring
* More IOT environment simulation