

# **Tencent IoT Hunter User Guide**

(2018. 11. 28)

# Contents

Contents.....	2
I. Introduction.....	3
II. Framework features.....	3
a) Good extension interface.....	3
b) Fine-grained information extraction.....	4
c) Third-party intelligence aggregation.....	4
III. Compatibility.....	4
IV. Information Extraction details.....	5
a) Static Information.....	5
b) Dynamic Information.....	6
c) Result.....	6
V. How to use.....	8
a) Static Information Extraction Tool.....	8
i. Setting parameters using the command Line.....	8
ii. Use config file: conf.py.....	9
iii. Configuration File (conf.py) Parameter description.....	9
b) Dynamic Analysis Tool.....	11
i. Analysis Environment Setup.....	11
ii. Analysis Tool Configuration.....	11
iii. Dynamic Analysis Tool Usage.....	12
VI. Use Extensible Plugins.....	13
a) Write Static Analysis Plugin.....	13
b) Static Plugin Debug.....	15
c) Write Dynamic Plugin.....	15
VII. Data visualization.....	16
a) ES Data Query.....	16
b) Weak Password Tag Cloud.....	16
c) C&C Geographic location visualization.....	17
d) Malware Variants Mining.....	18
VIII. Tools Used.....	18
IX. Future Work.....	19

# I. 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.

## II. Framework features

### a) 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.

## **b) 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.

## **c) 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.

# **III. Compatibility**

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

## IV. Information Extraction details

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

### d) 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

## e) 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

## f) Result

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

### Static Analysis Results JSON:

```

"machine_arch": "ARM", "packer": "upx",
"md5": "ebc376d877523c6cb673", "malicious_type": [
"sha1": "0c456349da336dc6d1c", "Botnet"
"virus_name": "Trojan.Linux.l",
"file_type": "elf", "attack_device": [
"cnc": [
[
"Router",
"Camera",
"206.189.126.143", "DVR",
"" "Printer",
"TV Box"
],
],
"sha256": "0d8159b9cc2bc7a54", "detect": 1,
"1a542eeeb28fea", "file_size": 128047,
"spread_way": [
"SSH", "malicious_family": [
"Telnet" "Mirai"
],
],
"weak_password": [
[
"root",
"vizxv"
],
[
"root",
"klv123"
],
[
"root",
"7ujMko0admin"
],
[
"root",
"7ujMko0vizxv"
]
],
"configuration": [
"\x05\x20",
"\x0f\x48",
"Connected To CNC\x00",
"shell\x00",
"enable\x00",
"system\x00",
"sh\x00",
"/bin/busybox SORA\x00",
"SORA: applet not found",
"ncorrect\x00",
"/bin/busybox ps\x00",
"/bin/busybox kill -9 \\",
"/proc/\x00",
"/exe\x00",
"/fd\x00",
"/maps\x00",
"/proc/net/tcp\x00",
"/status\x00",
],
],
"function": [
"attack_get_opt_str",
"attack_start",
"attack_parse",
"attack_get_opt_ip",
"attack_get_opt_int",
"attack_init",
"attack_method_udpplain",
"attack_method_std",
"attack_method_udpgener",
"attack_method_greeth",
"attack_method_greip",
"attack_method_udpvse",
"attack_method_tcpsyn",
"checksum_generic",
"checksum_tcpudp",
"killer_kill_by_port",
"killer_init",
"anti_gdb_entry",
"resolve_cnc_addr"
],
"main_function": [
"DDoS",
"Downloader"
],
"string": [
"206.189.126.143",
"/proc/stat",
"/proc/cpuinfo",
"processor",
"/sys/devices/system/cpu",
"/dev/null"
]
],
],

```

## Dynamic Analysis Results JSON:

```

"openat(AT_FDCWD, \"/proc/sys/vm/mmap_min_addr\", O_RDONLY) = 4",
"openat(AT_FDCWD, \"/tmp/qemu-open.RbtMLK\", O_RDWR|O_CREAT|O_EXCL, 0600) = 4",
"openat(AT_FDCWD, \"/proc/net/route\", O_RDONLY) = 5",
"openat(AT_FDCWD, \"/dev/watchdog\", O_RDWR <unfinished ...>",
"openat(AT_FDCWD, \"/dev/misc/watchdog\", O_RDWR <unfinished ...>",
"openat(AT_FDCWD, \"/proc/400/\", O_RDONLY|O_NONBLOCK|O_DIRECTORY) = -1 ENOENT (No such file or directory)",
"openat(AT_FDCWD, \"/proc/401/\", O_RDONLY|O_NONBLOCK|O_DIRECTORY) = -1 ENOENT (No such file or directory)"

```

```

{
  "plugin_info": {
    "GetConnectIP": [
      "8.8.8.8",
      "194.143"
    ]
  },
}

```

```

[
  {
    "port": 80,
    "addr": "192.168.1.1",
    "recvfrom(3, \"PING :48 01\\r\\n\\n\", 4096, 0, NULL, NULL) = 16"
  },
  "irc_info": [
    {
      "ip.dst": "192.168.1.15",
      "irc.response": ":irc.fbi.Null NOTICE AUTH :*** Looking up your hostname...",
      "ip.src": "192.168.1.43"
    },
    {
      "ip.dst": "192.168.1.15",
      "irc.response": ":irc.fbi.Null NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead",
      "ip.src": "192.168.1.43"
    }
  ]
}

```

## V. How to use

### g) 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	single 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.

### i. 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\2983a7e5bc97996cd98dff4f78e95b2  
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
```

### **ii. Use config file: conf.py**

```
MAL_SAMPLES_DIR = r"F:\Samples"  
RESULT_OUTPUT_DIR = r"F:\result"
```

Run python script:

```
python iot_hunter.py
```

### **iii. 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 = "676xxxxxxxxxxxxxxxxxxxxxxxxxxxx7d1db"

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 VirusTotal 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 command 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"
```

## h) Dynamic Analysis Tool

### iv. 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 lot 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

### v. 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

```

## vi. Dynamic Analysis Tool Usage

**Single File Analysis:** lotHunterDynamic.py -f filename -d logdir

**File Directory Analysis:** lotHunterDynamic.py -f directory -d logdir

```

lotHunterDynamic.py -h
usage: lotHunterDynamic.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

## VI. Use Extensible Plugins

### i) 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["UNKNOWN"]
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)
```

## j) 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:

```
2018-11-27 16:56:41,046 - IDA_ANALYSIS_FILE - INFO - =====Analyze File Begin=====
2018-11-27 16:56:41,048 - IDA_ANALYSIS_FILE - INFO - Sample Path:F:\Samples\0019c77ad7f4f97ec492726e9aa8e15e↓
2018-11-27 16:56:41,053 - IDA_ANALYSIS_FILE - INFO - Load plugin:f:\IoTHunter\plugins\plugin_Gafgvt_ARM.py↓
2018-11-27 16:56:41,055 - IDA_ANALYSIS_FILE - INFO - Load plugin:f:\IoTHunter\plugins\plugin_Gafgvt_x86.py↓
2018-11-27 16:56:41,058 - IDA_ANALYSIS_FILE - INFO - Load plugin:f:\IoTHunter\plugins\plugin_Mirai.py↓
2018-11-27 16:56:41,059 - IDA_ANALYSIS_FILE - INFO - Plugin Num:3↓
2018-11-27 16:56:41,059 - IDA_ANALYSIS_FILE - INFO - Analyze result:0 From <__main__.GafgvtARM object at 0x058C6C10>↓
2018-11-27 16:56:41,061 - IDA_ANALYSIS_FILE - ERROR - analyze()↓
Traceback (most recent call last):↓
  File "f:\IoTHunter\ida_analysis_file.py", line 331, in analyze↓
    plugin_obj = plugin()↓
  File "f:\IoTHunter\plugins\plugin_Gafgvt_x86.py", line 18, in __init__↓
    aaa↓
NameError: global name 'aaa' is not defined↓
2018-11-27 16:56:41,062 - IDA_ANALYSIS_FILE - INFO - =====Analyze File Begin=====
```

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

## k) 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

```

## VII. 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.

### 1) ES Data Query

cnc	md5	virus_name	file_size	packer	malicious_type
▶ 178.62.194.120,	2931642db89531fafb1ac77206dc86df	Trojan.Linux.Mirai.caa	56,880	None	Botnet
▶ 206.189.217.84,	2989b5de79e0ab4417c10b64738a10a0	Trojan.Linux.Mirai.caa	56,584	None	Botnet
▶ 212.237.61.50,	2994d321a8c54542f94a2cb3b3655d96	Trojan.Linux.Mirai.caa	98,304	None	Botnet
▶ 80.211.69.98,	29e9daabfb796a4bdefa1654c4673c1a	Trojan.Linux.Mirai.caa	128,047	None	Botnet
▶ 46.148.20.36,	48616c70bdd7b973a54ed5352aeb2d8c	Trojan.Linux.Mirai.caa	56,880	upx	Botnet
▶ 165.227.115.67,	2b4655234cbfec1dfd89330d1b50783b	Trojan.Linux.Mirai.caa	120,875	None	Botnet
▶ 45.32.202.190,	2baa3729d4fd0e367718dba2d450fbc8	Trojan.Linux.Mirai.caa	66,688	None	Botnet
▶ 206.189.16.32,	2bfda0223a7e62aa907cbd9b32f2cd9c	Trojan.Linux.Mirai.caa	106,496	None	Botnet
▶ 142.93.207.201,	0ac23b17dafc17fa17f88ffe11a380a6	Trojan.Linux.Mirai.caa	59,680	upx	Botnet
▶ 206.189.168.196,	2de956ca8ddf7595e16dda1b781d1ea1	Trojan.Linux.Mirai.caa	114,688	None	Botnet

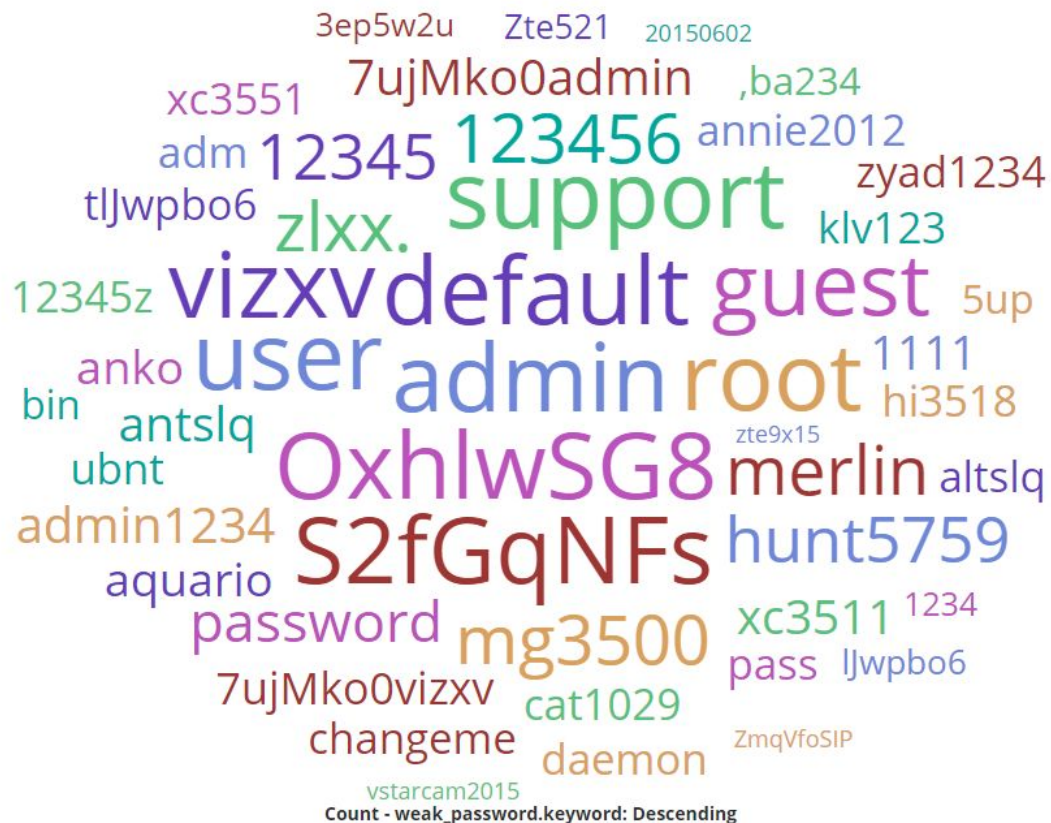
### m) Weak Password Tag Cloud

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.



## n) C&C Geographic location visualization

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



## o) Malware Variants Mining

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

md5	configuration
49c75da 3513859 79c6215 7e32082 912b	\x05\x39, \x07\xbe, DaddyL33T Infected Your <b>Shit</b> \x00, /proc/\x00, /exe\x00, /fd\x00, /maps\x00, /proc/net/tcp\x00, /status\x00, .anime\x00, /proc/net/route\x00, assword\x00, TSource Engine Query\x00, /etc/resolv.conf\x00, nameserver \x00, /dev/watchdog\x00, /dev/misc/watchdog\x00, pbbf~cu\x11, ogin\x00, enter\x00, 1gba4cdom53nhp12ei0kfj\x00
142df97 45ebe81 3cbae4d d37b44a 9a26	\x05\x39, \x07\xbe, DaddyL33T Infected Your <b>Shit</b> \x00, /proc/\x00, /exe\x00, /fd\x00, /maps\x00, /proc/net/tcp\x00, /status\x00, .anime\x00, /proc/net/route\x00, assword\x00, TSource Engine Query\x00, /etc/resolv.conf\x00, nameserver \x00, /dev/watchdog\x00, /dev/misc/watchdog\x00, pbbf~cu\x11, ogin\x00, enter\x00, 1gba4cdom53nhp12ei0kfj\x00
19f8d69 8a03cd0 0586a03 1561442 c811	\x02\xa4, rootmodz.site\x00, senpai.site\x00, \x87], seraph just fucked your <b>shit</b> cunt\x00, shell\x00, enable\x00, system\x00, sh\x00, /bin/busybox MIORI\x00, MIORI: applet not found\x00, ncorrect\x00, /bin/busybox ps\x00, /bin/busybox kill -9 \x00, /proc/\x00, /exe\x00, /fd\x00, /proc/net/tcp\x00, /status\x00, /proc/net/route\x00, assword, TSource Engine Query\x00, /etc/resolv.conf\x00, nameserver \x00, /dev/watchdog\x00, /dev/misc/watchdog\x00, pbbf~cu, ogin\x00, enter\x00, 1gba4cdom53nhp12ei0kfj\x00
e2f2146 996a237 827bf18 3f30b40 01e1	\x01\xb2, rootme.club\x00, senpai.site\x00, \xba;, root senpai just infected your <b>shit</b> \x00, shell\x00, en, sy, sh\x00, /bin/busybox MIORI\x00, MIORI: applet not found\x00, ncorrect\x00, /bin/busybox ps\x00, /bin/busybox kill -9 \x00, /p, /exe\x00, /fd\x00, /maps\x00, /proc/net/tcp\x00, /status\x00, /proc/net/route\x00, as, TSource Engine Query\x00, /etc/resolv.conf\x00, nameserver \x00, /dev/watchdog\x00, /dev/misc/watchdog\x00, pb, ogin\x00, enter\x00, 1gba4cdom53nhp12ei0kfj\x00
0624f99 07ec835 48b1812 a4ae63d 5766	\x02\xa4, rootmodz.site\x00, senpai.site\x00, \x87], seraph just fucked your <b>shit</b> cunt\x00, shell\x00, enable\x00, system\x00, sh\x00, /bin/busybox MIORI\x00, MIORI: applet not found\x00, ncorrect\x00, /bin/busybox ps\x00, /bin/busybox kill -9 \x00, /proc/\x00, /exe\x00, /fd\x00, /proc/net/tcp\x00, /status\x00, /proc/net/route\x00, assword, TSource Engine Query\x00, /etc/resolv.conf\x00, nameserver \x00, /dev/watchdog\x00, /dev/misc/watchdog\x00, pbbf~cu, ogin\x00, enter\x00, 1gba4cdom53nhp12ei0kfj\x00

## VIII. 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/>)

## IX. 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