**Tencent IoT Hunter**

**使用文档**

**（2018.11.28）**

# 目录

[目录 2](#_Toc15291)

[一、 简介 4](#_Toc1544)

[二、 框架特点 4](#_Toc2610)

[（一） 良好的扩展接口 4](#_Toc23288)

[（二） 细粒度信息提取 4](#_Toc32139)

[（三） 第三方情报汇总 5](#_Toc4368)

[三、 兼容性 5](#_Toc30453)

[四、 提取信息详情 5](#_Toc27996)

[（一） 静态信息提取 5](#_Toc13720)

[（二） 动态信息提取 6](#_Toc12162)

[（三） 输出结果 7](#_Toc2840)

[五、 使用方法 9](#_Toc6345)

[（一） 静态信息提取工具 9](#_Toc22796)

[1． 使用命令行设置参数 9](#_Toc15406)

[2． 使用配置文件conf.py设置参数 10](#_Toc18231)

[3． 配置文件conf.py参数说明 10](#_Toc26682)

[（二） 动态提取工具 12](#_Toc24716)

[1． 分析环境搭建 12](#_Toc16504)

[2． 分析工具配置 12](#_Toc30013)

[3． 动态分析工具使用 13](#_Toc3010)

[六、 使用扩展接口 14](#_Toc21099)

[（一） 添加静态分析plugin 14](#_Toc9654)

[（二） 静态plugin调试 15](#_Toc9265)

[（三） 添加动态plugin 16](#_Toc13589)

[七、 数据可视化 17](#_Toc15645)

[（一） ES数据展示 17](#_Toc5594)

[（二） 弱口令的云图 17](#_Toc5715)

[（三） 地址位置展示 18](#_Toc7498)

[（四） 挖掘家族变种 19](#_Toc13807)

[八、 使用到的工具 19](#_Toc20305)

[九、 未来规划 20](#_Toc21864)

# 简介

Tencent IoT Hunter是为了收集IoT威胁情报而创建的框架工具，通过静态提取、动态提取、第三方网络平台获取信息的方式，全方位构建IoT病毒生命周期，为安全研究人员对IoT病毒的分析、研究、追踪提供详尽的威胁情报信息。

使用本框架工具可以非常精准的、细粒度的获取IoT样本中的相关恶意信息（CNC、Domain、function、etc.），这些恶意信息可以直接用来创建IoT恶意信息云查服务，而不需要分析人员对恶意信息进行二次确认，提升了恶意信息处理效率。

本框架提供了良好的扩展接口，使用者可编写个性化plugin来扩展信息提取的范围。通过提取出的信息，可快速搭建IoT威胁情报平台，快速进行可视化分析，挖掘IoT样本家族、变种的活跃度与分布情况。

# 框架特点

## 良好的扩展接口

此框架提供了扩展接口，使用此框架的安全人员可编写自己的plugin，来扩展信息提取的范围。为方便使用者编写自己的plugin，框架提供了基础类，新的plugin只需要继承基础类便可被直接执行，不需要编写额外的代码，让使用人专注于目标样本的信息提取。同时，框架会记录详细的log，方便使用者根据log记录来进行代码调试。

在编写plugin时，分析人员可定义自己的目标样本家族类型、传播方式、攻击的设备、攻击方式等，并可以精准获取CNC、Domain、弱口令字典、控制命令等恶意信息。

## 细粒度信息提取

以往的一些分析工具会将样本中的信息进行粗粒度提取，需要进行二次确认是否为恶意，才能对信息进行使用。

例如，从一个样本中粗粒度的提取到了一个IP，但这个IP是否为恶意并不知晓，因此不能直接拿来当成恶意IP来使用，需要对这个IP进一步确认是否为恶意IP才能使用。

而本框架工具在静态提取信息时，使用精准的特征匹配方式，可以精准的提取出恶意信息，无需再进行二次确认便可将恶意信息直接使用。

## 第三方情报汇总

本框架尝试通过公开的第三方网络情报平台，拉取更多IoT情报信息，以便为使用者提供更多有价值的参考信息。

# 兼容性

* Python2、Python3
* Windows、Linux、（OS X暂时未测试）

# 提取信息详情

本框架支持ARM、X86、X64、MIPS、Sparc、PowerPC等平台上的ELF文件静态、动态信息提取。

## 静态信息提取

**定义信息：**

* 病毒名
* 恶意类型
* 家族信息
* 传播方式
* 攻击设备
* 主要攻击方式

**提取信息：**

* 基本信息（文件大小、文件类型、运行平台、md5、sha1、sha256）
* 控制端地址
* 域名
* IP
* URL
* UDP
* TCP
* DNS
* 配置信息
* 弱口令字典
* 所有字符串
* 可疑字符串
* 所有函数名
* 控制命令
* 加壳信息

## 动态信息提取

**进程信息：**

* 进程EXECVE 执行：参数信息
* 进程clone 执行信息

**文件操作信息：**

* 文件打开：文件名
* 文件读取：读取数据
* 文件写入：写入数据

**Socket信息：**

* Connect信息：ip
* Recvfrom 信息：ip，数据
* Sendto信息：ip，数据
* Bind信息：ip

**网络通信信息：**

* 网络通信包信息：ip，协议
* HTTP信息：host，数据
* TCP信息：ip，数据
* UDP信息：ip, 数据
* IRC 信息：ip，IRC消息

**动态分析插件信息：**

* 插件名称
* 分析插件结果

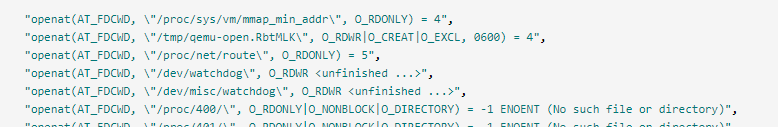
## 输出结果

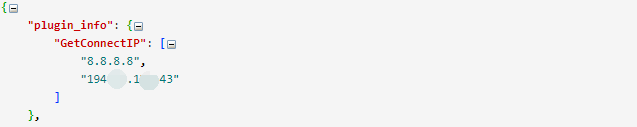
所有分析结果会以json的形式保存到结果文件中：

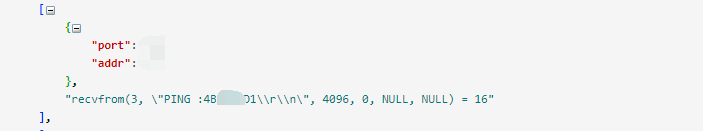
**静态分析结果JSON:**

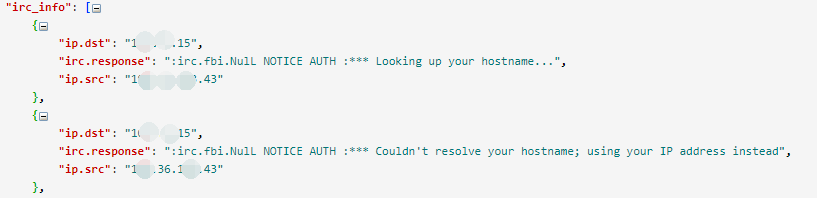


**动态分析数据JSON：**









# 使用方法

## 静态信息提取工具

此工具可使用命令行设置参数，也可以通过修改配置文件来设置参数。

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.

### 使用命令行设置参数

单样本分析：

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

多样本分析：

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

拉取样本VirusTotal相关信息：

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

将分析结果导入Elasticsearch：

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

### 使用配置文件conf.py设置参数

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

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

直接调用py脚本即可：

python iot\_hunter.py

### 配置文件conf.py参数说明

必需设置的参数：

IDA分析工具路径，https://www.hex-rays.com

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

按需配置的参数：

样本目录

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

结果输出目录

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

UPX工具路径，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"

访问VirusTotal需要代理时，设置此参数

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

始终尝试接取样本的VirusTotal时，设置此参数为True

VIRUSTOTAL\_ALWAYS\_GET = False

样本大小限制

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

Elasticsearch相关参数，https://github.com/elastic/elasticsearch-py

ES\_HOST = "localhost:9200"

ES\_INDEX\_NAME = "iot\_threat"

ES\_TYPE\_NAME = "FileAnalysis"

可保持默认的参数：

IDA脚本输出的分析结果文件

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

VirusTotal拉取的样本信息文件

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

汇总所有信息的结果（IDA + VirusTotal），需要使用-c参数

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

Log信息文件

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"

内部使用的参数，尽量不要修改，

如要修改，需一并修改相关文件名与目录名

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

IDA\_PLUGINS\_DIR\_NAME = "plugins"

## 动态提取工具

### 分析环境搭建

动态分析环境需要在虚拟机环境中运行IoT样本，监控其行为，IoT样本执行环境需要虚拟机，本动态分析工具基于VirtualBox开发，需要安装在Guest VM安装linux系统如（Ubuntu）。

Guest VM工具安装：

1. Linux 系统，VBoxGuestAdditions\_5.2.22
2. QEMU虚拟机，通过QEMU User Mode 模拟支持运行ARM, MIPS, PowerPC等多平台IOT文件
3. Strace：监控样本运行的系统调用信息
4. Tcpdump: 记录样本执行的网络包，用于分析网络行为
5. 纯净的系统安装上述工具后，使用root用户，保存镜像名为analysis

Host 分析工具：

1. Tshark：用于分析tcpdump生成的pcap包

### 分析工具配置

配置文件：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

### 动态分析工具使用

**单样本分析**：IotHunterDynamic.py –f 文件名 –d log目录

**多样本分析**：IotHunterDynamic.py –d 目录 –d log目录

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

**分析框架流程：**

1. 获取待分析样本
2. 启动分析虚拟机
3. 发送文件到虚拟机
4. 发送分析脚本到虚拟机
5. 执行样本，记录行为
6. 从分析机获取日志
7. 解析日志，获取文件，网络，进程信息
8. 调用用户自定义解析插件
9. 生成样本json格式分析结果

# 使用扩展接口

## 添加静态分析plugin

使用者可根据目标样本编写自己的信息提取插件plugin，将编写好的plugin放在plugins目录中即可直接执行。

框架提供了基础类PluginParent，此类提供了基础信息：

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需要继承PluginParent类，并实现analyze函数，在此函数中对所需字段进行填充，在代码尾部调用add\_plugin(派生类名)即可：

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):

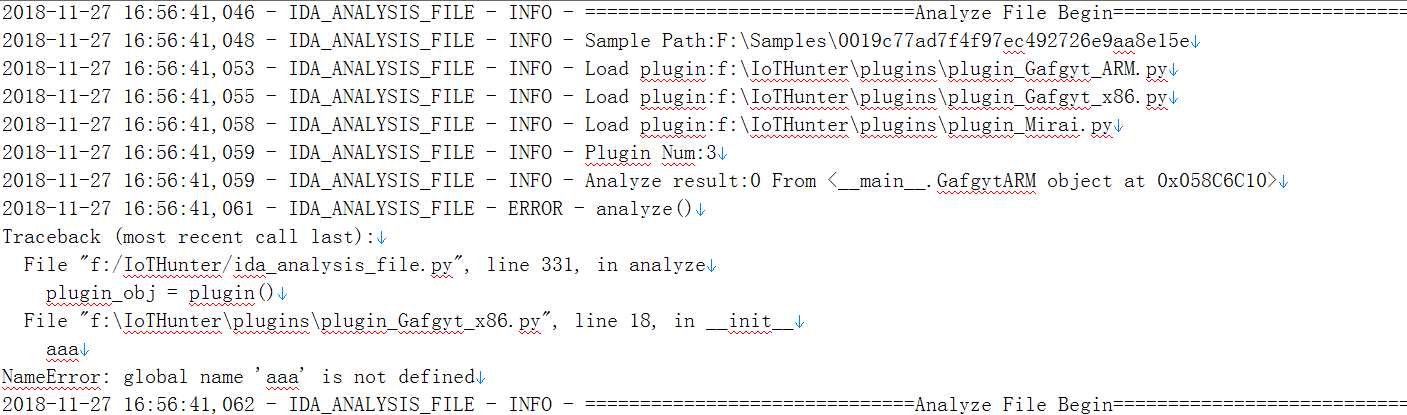
...

add\_plugin(MiraiARM)

## 静态plugin调试

由于IDAPython命令行调用的方式遇到异常时会自动退出，并不会保存代码异常信息，因此，对编写plugin的使用者来说非常不方便，无法定位自己代码的问题，为此，本框架提供了log记录，使用者在编写plugin并执行后，可查看log来定位异常问题。

log\_ida\_file\_analysis.log：



使用者也可以使用logger.info("xxx")方式，打印自己的log信息。

## 添加动态plugin

使用者可根据目标样本编写自己的动态信息提取插件plugin，将编写好的plugin放在DynamicPlugins\目录中即可直接执行。

插件开发：需实现analyze 和 get\_result两个接口，分析框架会调用所有的插件，并记录插件结果，analyze函数参数behaviors记录了样本的行为数据。用户可以进行自定义分析，获取特定结果。

插件示例：获取样本connect所有目的ip列表

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

# 数据可视化

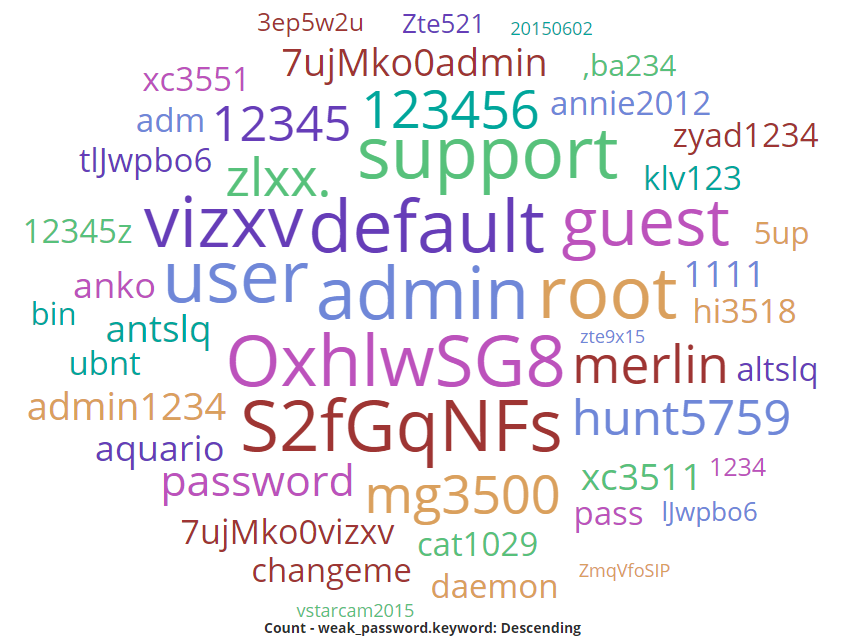
框架内提供了将分析数据导入到Elasticsearch的功能，使用者可快速搭建IoT数据挖掘平台，进行数据可视化分析。可以快速挖掘IoT家族的变种、传播广度、活跃度、新的弱口令利用等。

## ES数据展示



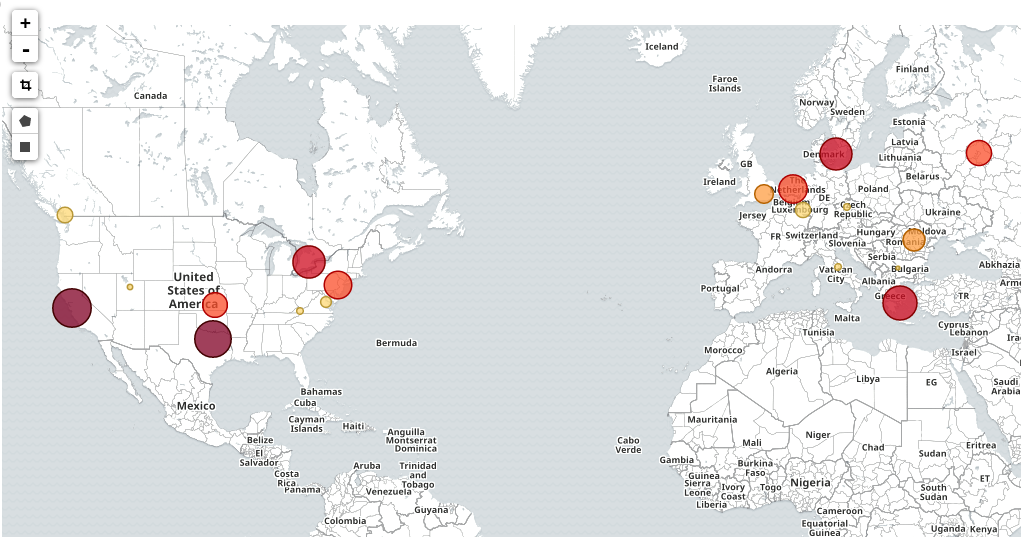
## 弱口令的云图

通过对IoT样本中使用的弱口令进行提取，可观察到哪些弱口令使用频度最高。而对所有已知弱口令进行监控，当出现新的弱口令时，说明很有可能出现新的变种或新的弱口令漏洞利用。



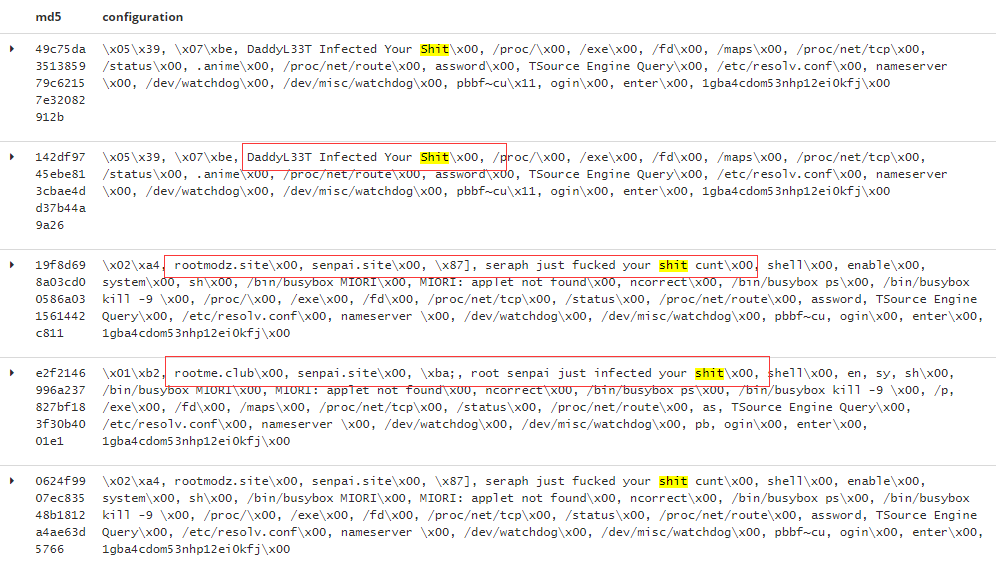
## 地址位置展示

通过提取的CNC地址，可以获取到IoT家族相关地址位置。



## 挖掘家族变种

通过一些关键字进行挖掘，可以聚类挖掘出不同的家族变种。



# 使用到的工具

* 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/)

# 未来规划

* 增加更多plugin
* 增加监控新变种的扩展功能
* 更多第三方平台信息接入
* 更多行为监控合入
* 更多真实IOT环境模拟