

Tencent IoT Hunter

使用文档

(2018. 11. 28)

目录

目录.....	2
一、 简介.....	3
二、 框架特点.....	3
（一） 良好的扩展接口.....	3
（二） 细粒度信息提取.....	3
（三） 第三方情报汇总.....	4
三、 兼容性.....	4
四、 提取信息详情.....	4
（一） 静态信息提取.....	4
（二） 动态信息提取.....	5
（三） 输出结果.....	6
五、 使用方法.....	8
（一） 静态信息提取工具.....	8
1. 使用命令行设置参数.....	8
2. 使用配置文件 conf.py 设置参数.....	9
3. 配置文件 conf.py 参数说明.....	9
（二） 动态提取工具.....	11
1. 分析环境搭建.....	11
2. 分析工具配置.....	11
3. 动态分析工具使用.....	12
六、 使用扩展接口.....	13
（一） 添加静态分析 plugin.....	13
（二） 静态 plugin 调试.....	14
（三） 添加动态 plugin.....	15
七、 数据可视化.....	16
（一） ES 数据展示.....	16
（二） 弱口令的云图.....	16
（三） 地址位置展示.....	17
（四） 挖掘家族变种.....	18
八、 使用到的工具.....	18
九、 未来规划.....	19

一、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。

二、框架特点

（一）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。

（二）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，才能对信息进行处理。

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。

（三）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。

三、兼容性

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

四、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。

（一）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

（二）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

（三）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:

```

"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": [ "malicious_family": [
"SSH", "Mirai"
"Telnet" ],
],
"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"
]
],
],

```

动态分析数据 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": "192.168.1.15",
    "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"
    }
  ],
]
```

五、使用方法

(一) 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	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.

1. 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\2983a7e5bc97996cd98dff4f78e95b2
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
```

2. 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
```

3. 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 = "676xxxxxxxxxxxxxxxxxxxxxxxxxxxx7d1db"

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"
```

（二）动态提取工具

1. 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

2. 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

```

3. 动态分析工具使用

单样本分析: lotHunterDynamic.py -f 文件名 -d log 目录

多样本分析: lotHunterDynamic.py -d 目录 -d log 目录

```

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

```

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

六、使用扩展接口

（一）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["UNKNOWN"]
    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)
```

（二）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:

```
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_Gafgyt_ARM.py↓
2018-11-27 16:56:41,055 - IDA_ANALYSIS_FILE - INFO - Load plugin:f:\IoTHunter\plugins\plugin_Gafgyt_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__.GafgytARM 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_Gafgyt_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=====
```

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

（三）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
```

七、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。

(一) ES 数据展示

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

(二) 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。

(四) 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。

md5	configuration
49c75da 3513859 79c6215 7e32082 912b	\x05\x39, \x07\xbe, DaddyL33T Infected Your shit \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 shit \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 shit 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 shit \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 aa4ae63d 5766	\x02\xa4, rootmodz.site\x00, senpai.site\x00, \x87], seraph just fucked your shit 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

八、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/>)

九、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