Shadows From the Past Threaten Italian Enterprises

yoroi.company/research/shadows-from-the-past-threaten-italian-enterprises

November 30, 2020

11/30/2020

Introduction

The modern cyber threat landscape hides nasty surprises for companies, especially for the most structured and complex companies. Many times, threat actors develop very dangerous and effective techniques using tools and technologies in a smart, unattended way. This is the case of a particular cyber criminal group operating cyber intrusion against one of the most targeted and cyber-mature industry sectors: the Banking sector.

During the last years our defense and intelligence operations spotted and tracked some of the operations of a particular Threat Group targeting financial institutions worldwide, in Europe and especially in Italy, at least since 2015. This particular cyber criminal group has been recently publicly disclosed with the name UNC1945 by Mandiant (TH-239 in our internal KB), presenting the findings of a particular investigation they recently took part and referencing an intrusion potentially dated back to 2018.

After the publication of the Mandiant report, we decided to reach out and de-classify some technical details about this mysterious group that is threatening some of the largest Italian - and European - companies.

In particular, we'll describe how the group approached the difficult task of bringing a modern cyber arsenal to multi-decade old legacy systems.

The Challenge of Attribution

The attribution of this group to known threat actors is still smoky and far for certainty. In fact, even if CERT-Yoroi reserved intelligence information points to old intrusions attributed to Carbanak/Anunak romainan cells dated back 2015-2016, there is still no hard evidence linking to this particular group.

Anyway, despite the unclear purposes of their recent intrusion reported by Mandiant one thing is pretty clear: the group is capable of running long lasting operations.

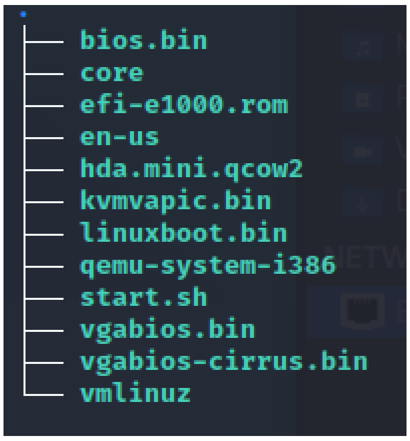
Operations running for years are not typical of the targeted ransomware operators afflicting thousands of companies nowadays. Most of those intrusions last weeks or at least months for a simple reason: persistence is really risky for the intruders, too much might cost them the whole profit opportunity.

In our experience, the actor behind TH-239 was historically financially motivated and well prepared in conducting intrusions in Enterprise grade - some time legacy - environments such as old Red Hat, Solaris OS and other Linux systems as well. A different skill set rather than most of the ransomware operators which are extremely good on Microsoft Windows environments. These much more rare abilities are typical of sophisticated groups such as those targeting high complex organizations such as Financial and Banking institutions, extremely characterized by legacy technologies, and almost unknown to most cyber criminals.

Technical Analysis

In this analysis, we want to deepen one of the post exploitation TTP used by the UNC1945 group to solve the huge problem of running modern attack tools on legacy systems. Do to so, the group in fact is using a custom QEMU linux virtual machine instance containing all the necessary tools adopted to achieve its objective.

This way all the operating system and dependencies issues become almost frictionless. In particular their portable virtual arsenal is based on QEMU images and looks like this:



```
.
├── bios.bin
├── core
├── efi-e1000.rom
├── en-us
├── hda.mini.qcow2
├── kvmvapic.bin
├── linuxboot.bin
├── qemu-system-i386
├── start.sh
├── vgabios.bin
├── vgabios-cirrus.bin
└── vmlinuz
```

Figure. Virtual Arsenal Structure

Let's go to analyze the various elements:

- **bios.bin:** is the QEMU seabios, an open source implementation of an x86 BIOS.
- **core:** is the original image of the Tiny Linux base
- **efi-e1000.rom:** QEMU custom ROM image.
- **en-us:** the mapping file for US keyboard layout.
- **hda.mini.qcow2:** QEMU QCOW2 Image (HD).
- **kvmapic.bin**: KVM in-kernel APIC support.
- **qemu-system-i386:** The QEMU PC System emulator;
- **start.sh:** the initialization script of the virtual machine.

The content of the starting script is quite simple:

```
!/bin/shif [ -z "$1" ];then  ./qemu-system-i386 -m 166 -kernel vmlinuz -initrd
core -hda hda.mini.qcow2 -append "tce=sda1 home=sda1 opt=sda1 noswap nozswap
superuser" -net nic -net user,tftp=/,hostfwd=tcp::2222-:22 -curseselse  ./qemu-
system-i386 -m 166 -kernel vmlinuz -initrd core -hda hda.mini.qcow2 -append
"tce=sda1 home=sda1 opt=sda1 noswap nozswap superuser" -net nic -net
user,tftp=/,hostfwd=tcp::19227-:22 -vnc none >/dev/null 2>&1 &fi
```

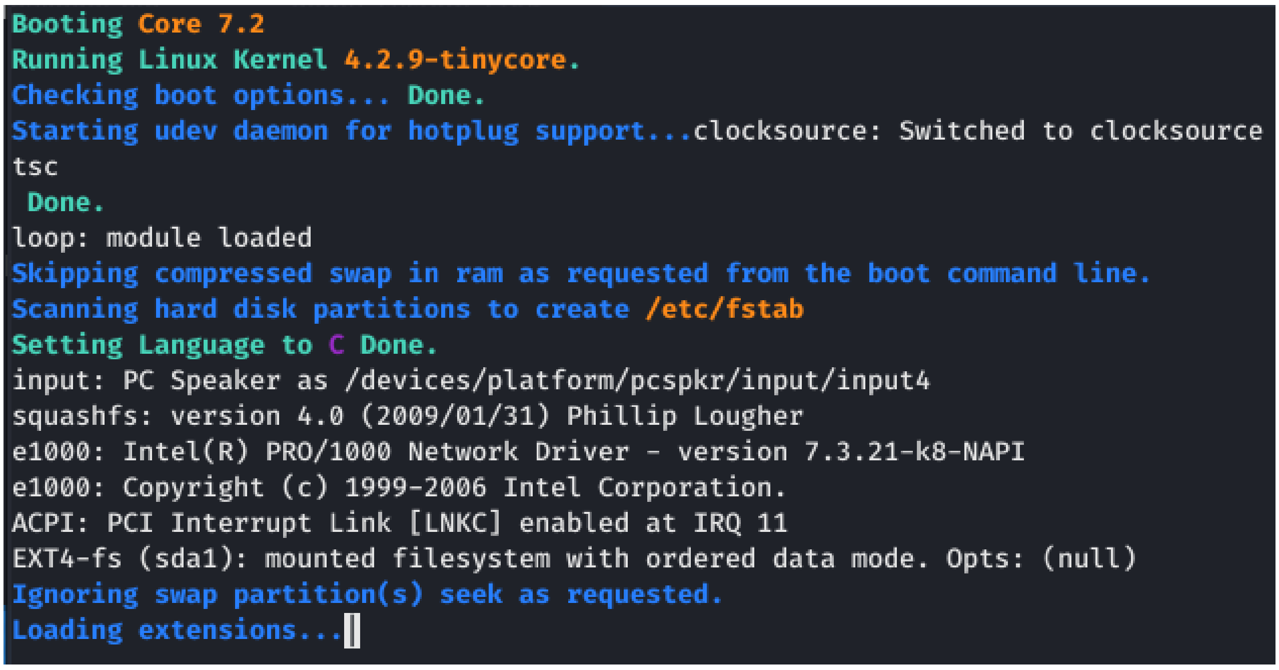
The bash script is quite easy. It has the purpose to launch the QEMU emulator system: if no parameter is provided ($1), the script executes the first branch enabling port forwarding between guest 22 to host 2222, using the display library "curses". This setup is to interact with the VM connecting through the host 2222 .

Without any parameter, the script configures the access through the usage of SSH, because the "curses" library is not listed and the VNC server is disabled. However, both the branches allow communication through the SSH protocol, in the first case starting the communication at the port 2222, in the second at the port 19227.

After the first overview of the configuration of the QEMU environment, we decided to deepen into the Virtual Machine by executing it.

The QEMU system

Starting the QEMU system, we obtained the first screen of the configuration of the QEMU system.



```
Booting Core 7.2
Running Linux Kernel 4.2.9-tinycore.
Checking boot options... Done.
Starting udev daemon for hotplug support...clocksource: Switched to clocksource
tsc
 Done.
loop: module loaded
Skipping compressed swap in ram as requested from the boot command line.
Scanning hard disk partitions to create /etc/fstab
Setting Language to C Done.
input: PC Speaker as /devices/platform/pcspkr/input/input4
squashfs: version 4.0 (2009/01/31) Phillip Lougher
e1000: Intel(R) PRO/1000 Network Driver - version 7.3.21-k8-NAPI
e1000: Copyright (c) 1999-2006 Intel Corporation.
ACPI: PCI Interrupt Link [LNKC] enabled at IRQ 11
EXT4-fs (sda1): mounted filesystem with ordered data mode. Opts: (null)
Ignoring swap partition(s) seek as requested.
Loading extensions...
```

Figure. Arsenal boot up

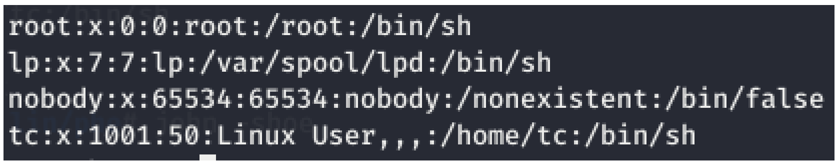
The specific linux distribution adopted by the attackers is Tinycore Linux 7.2, a lightweight distro running on a 4.2.9 Linux kernel. It is interesting to notice that the Tiny Core 7.2 release is about 4 years old, but we have evidence it was still part of their cyber-arsenal during 2020.

So, we can hypothesize that the creation and the configuration of the virtual QEMU environment is 2016 and the threat actor continues to use that since that period, and, thus, that the group is using this technique at least from 2016.

Navigating the filesystem, we discovered a huge amount of tools for exploitation, privilege escalation, lateral movement and exfiltration.

The Users

Navigating all the filesystem and dissecting every malicious capability have been a challenging activity because we had to explore an entire modular VM with such many interesting artifacts. So, we started to see which are the users contained inside the "/etc/passwd" file.

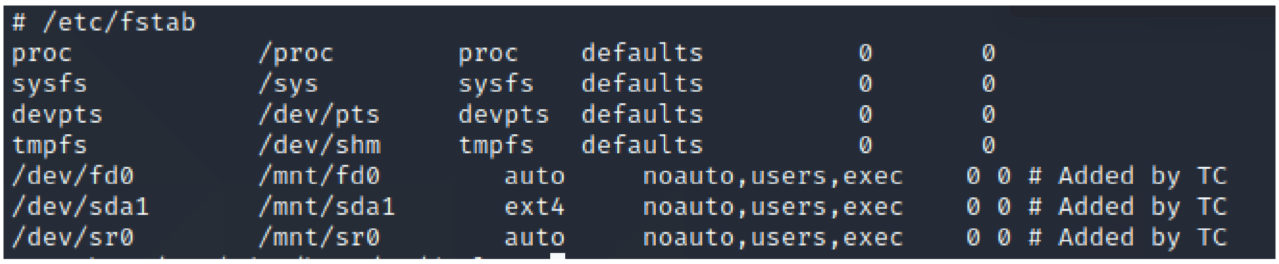


```
root:x:0:0:root:/root:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/false
tc:x:1001:50:Linux User,,,:/home/tc:/bin/sh
```

Figure. Virtual arsenal users configuration

Besides the "root" user, the base linux user with the highest level of privileges, we have, "lp" and "nobody" with no specific privileges; the first one is a sort of spooler service and the other one has no privileges and we didn't find anything regarding the directory "/nonexistent" inside the disk image.

The default user of the TinyCore Linux distribution is "tc" was configured with password " " (empty space) and it is the user basically adopted by the attackers. The confirmation of this hypothesis is inside "/etc/fstab" configuration file where the user "tc" has three entries, as we can see below:



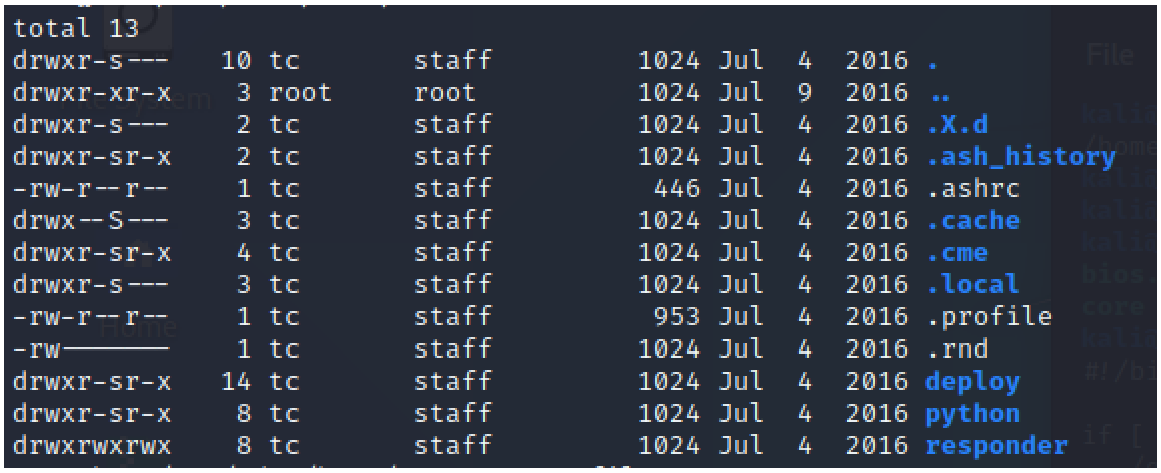
```
# /etc/fstab
proc            /proc        proc    defaults         0       0
sysfs           /sys         sysfs   defaults         0       0
devpts          /dev/pts     devpts  defaults         0       0
tmpfs           /dev/shm     tmpfs   defaults         0       0
/dev/fd0        /mnt/fd0        auto     noauto,users,exec     0 0 # Added by TC
/dev/sda1       /mnt/sda1       ext4     noauto,users,exec     0 0 # Added by TC
/dev/sr0        /mnt/sr0        auto     noauto,users,exec     0 0 # Added by TC
```

Figure. FSTAB configuration

This mode of adding devices onto the virtual system is provided thanks to the command line of QEMU hypervisor application though the command "-append" which has the purpose of appending a new virtual disk to the basic core of the linux system.

Inside the virtual disk there is the complete post-exploitation arsenal adopted by this new powerful threat actor and this particular configuration enables it to perform a completely modular arsenal. In fact, the attackers could update only the virtual device letting the core of the VM unaltered by simply replacing the virtual QCOW2 disk.

The three most important directories are "deploy", "python" and "responder".

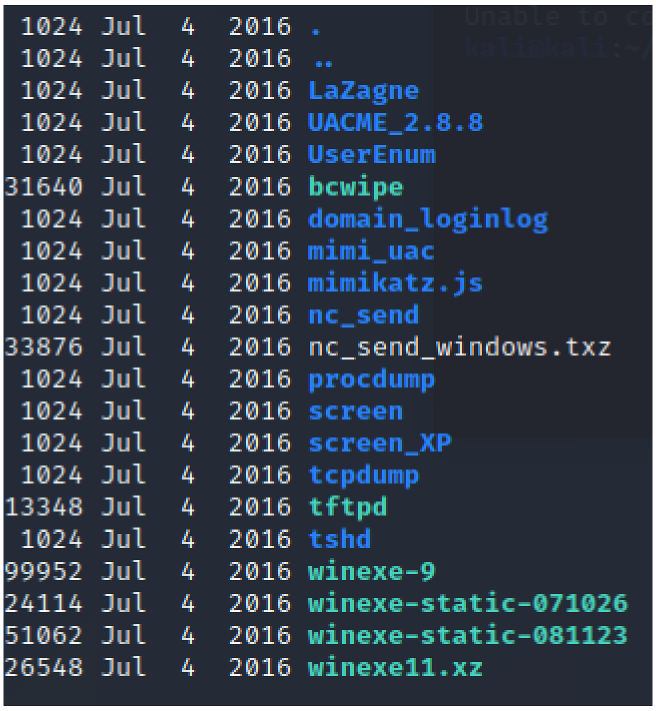


```
total 13
drwxr-s---   10 tc      staff      1024 Jul  4  2016 .
drwxr-xr-x    3 root    root       1024 Jul  9  2016 ..
drwxr-s---    2 tc      staff      1024 Jul  4  2016 .X.d
drwxr-sr-x    2 tc      staff      1024 Jul  4  2016 .ash_history
-rw-r--r--    1 tc      staff       446 Jul  4  2016 .ashrc
drwx--S---    3 tc      staff      1024 Jul  4  2016 .cache
drwxr-sr-x    4 tc      staff      1024 Jul  4  2016 .cme
drwxr-s---    3 tc      staff      1024 Jul  4  2016 .local
-rw-r--r--    1 tc      staff       953 Jul  4  2016 .profile
-rw-------    1 tc      staff      1024 Jul  4  2016 .rnd
drwxr-sr-x   14 tc      staff      1024 Jul  4  2016 deploy
drwxr-sr-x    8 tc      staff      1024 Jul  4  2016 python
drwxrwxrwx    8 tc      staff      1024 Jul  4  2016 responder
```

Figure. TC user folder structure

The "Deploy" Directory

This directory contains many tools part of the cyber arsenal. It is composed of many types of executable binaries, scripts and archives such as .py, .exe, .txz, or .js. This is an indication that the repository is virtual-arsenal was originally designed to target different kind of machines, as needed.

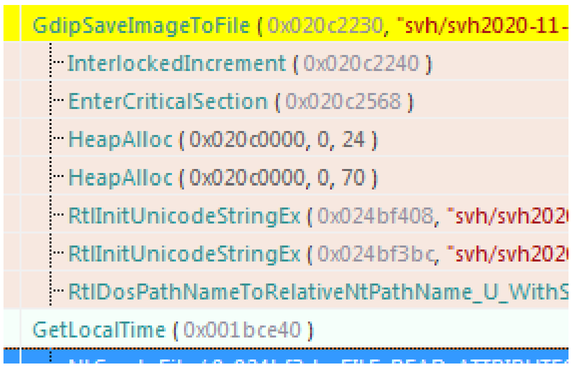


```
 1024 Jul  4  2016 .
 1024 Jul  4  2016 ..
 1024 Jul  4  2016 LaZagne
 1024 Jul  4  2016 UACME_2.8.8
 1024 Jul  4  2016 UserEnum
31640 Jul  4  2016 bcwipe
 1024 Jul  4  2016 domain_loginlog
 1024 Jul  4  2016 mimi_uac
 1024 Jul  4  2016 mimikatz.js
 1024 Jul  4  2016 nc_send
33876 Jul  4  2016 nc_send_windows.txz
 1024 Jul  4  2016 procdump
 1024 Jul  4  2016 screen
 1024 Jul  4  2016 screen_XP
 1024 Jul  4  2016 tcpdump
13348 Jul  4  2016 tftpd
 1024 Jul  4  2016 tshd
99952 Jul  4  2016 winexe-9
24114 Jul  4  2016 winexe-static-071026
51062 Jul  4  2016 winexe-static-081123
26548 Jul  4  2016 winexe11.xz
```

Figure. Deploy directory contents

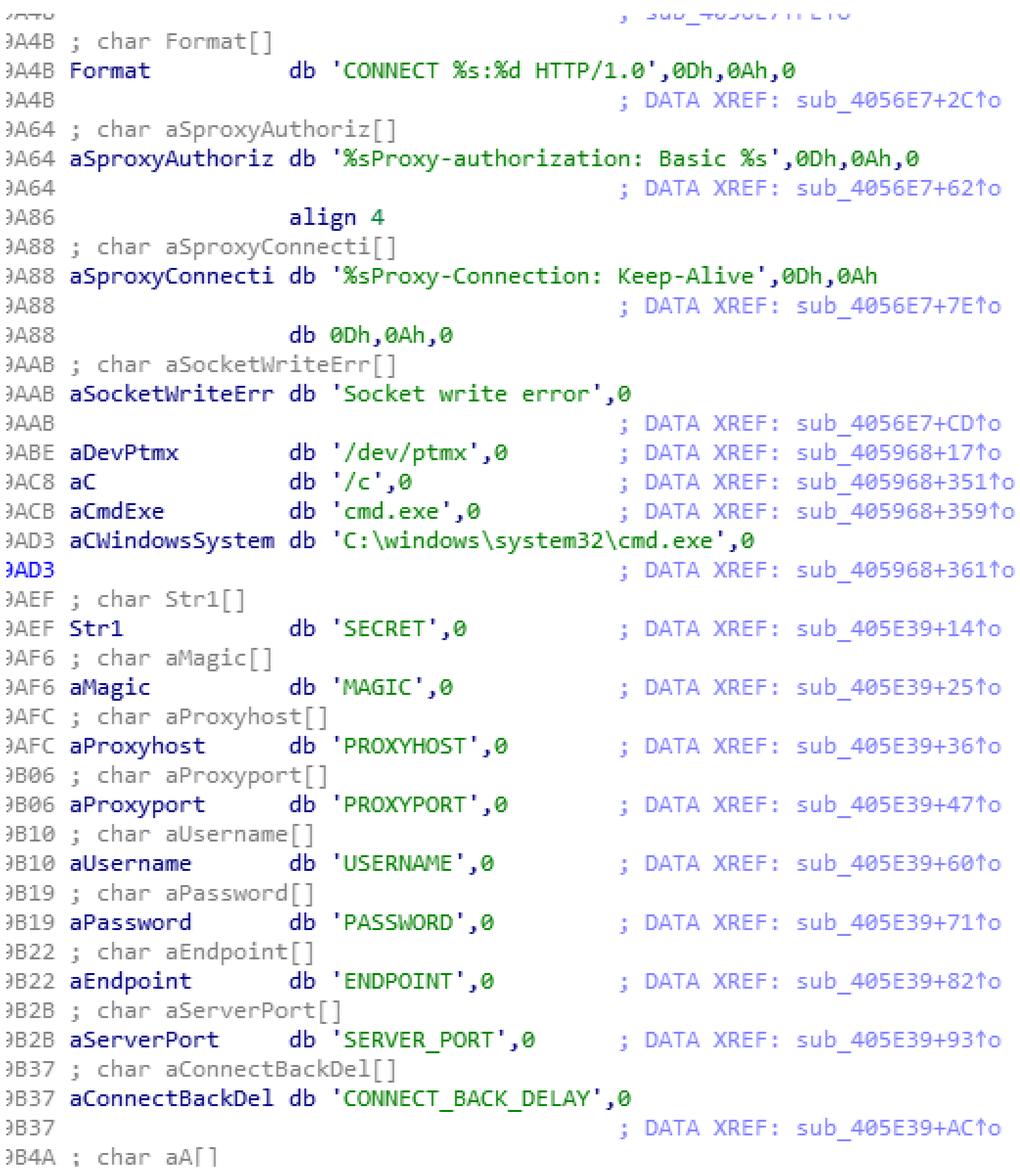
Many of the sub-folder in this directory references known tools, most of them publicly available. For instance:

- LaZagne: historical tool to harvest credentials from heterogeneous environments such as Microsoft Windows, Linux based systems and Apple's OSX.
- UACME: tool to bypass user access control restrictions on Microsoft Windows environments.
- Mimikatz: maybe the most popular post exploitation and privilege escalation tool even nowadays.
- Bcwipe: a legit data wiping software able to permanently and selectively delete files in a unrecoverable way, showing the group have particular care in remaining unnoticed.
- procdump: a sysinternal command-line utility to monitor and dump process memory.
- Screen and screen_xp: contains custom recognizance tools to monitor the desktop screen of their targets.
- tshd: contains a TinyShell backdoor - typical unix tool - but customized and compiled for Windows environments.



```
GdipSaveImageToFile ( 0x020c2230, "svh/svh2020-11-
  InterlockedIncrement ( 0x020c2240 )
  EnterCriticalSection ( 0x020c2568 )
  HeapAlloc ( 0x020c0000, 0, 24 )
  HeapAlloc ( 0x020c0000, 0, 70 )
  RtlInitUnicodeStringEx ( 0x024bf408, "svh/svh202
  RtlInitUnicodeStringEx ( 0x024bf3bc, "svh/svh202
  RtlDosPathNameToRelativeNtPathName_U_WithS
GetLocalTime ( 0x001bce40 )
```

Figure. custom screen capture tool dynamics



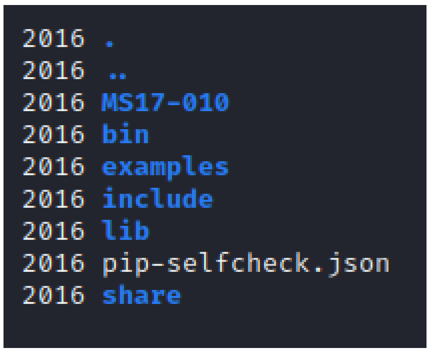
```
A4B ; char Format[]
A4B Format          db 'CONNECT %s:%d HTTP/1.0',0Dh,0Ah,0
A4B                                     ; DATA XREF: sub_4056E7+2C↑o
A64 ; char aSproxyAuthoriz[]
A64 aSproxyAuthoriz db '%sProxy-authorization: Basic %s',0Dh,0Ah,0
A64                                     ; DATA XREF: sub_4056E7+62↑o
A86                 align 4
A88 ; char aSproxyConnecti[]
A88 aSproxyConnecti db '%sProxy-Connection: Keep-Alive',0Dh,0Ah
A88                                     ; DATA XREF: sub_4056E7+7E↑o
A88                 db 0Dh,0Ah,0
AAB ; char aSocketWriteErr[]
AAB aSocketWriteErr db 'Socket write error',0
AAB                                     ; DATA XREF: sub_4056E7+CD↑o
ABE aDevPtmx        db '/dev/ptmx',0        ; DATA XREF: sub_405968+17↑o
AC8 aC              db '/c',0               ; DATA XREF: sub_405968+351↑o
ACB aCmdExe         db 'cmd.exe',0          ; DATA XREF: sub_405968+359↑o
AD3 aCWindowsSystem db 'C:\windows\system32\cmd.exe',0
AD3                                     ; DATA XREF: sub_405968+361↑o
AEF ; char Str1[]
AEF Str1            db 'SECRET',0           ; DATA XREF: sub_405E39+14↑o
AF6 ; char aMagic[]
AF6 aMagic          db 'MAGIC',0            ; DATA XREF: sub_405E39+25↑o
AFC ; char aProxyhost[]
AFC aProxyhost      db 'PROXYHOST',0        ; DATA XREF: sub_405E39+36↑o
B06 ; char aProxyport[]
B06 aProxyport      db 'PROXYPORT',0        ; DATA XREF: sub_405E39+47↑o
B10 ; char aUsername[]
B10 aUsername       db 'USERNAME',0         ; DATA XREF: sub_405E39+60↑o
B19 ; char aPassword[]
B19 aPassword       db 'PASSWORD',0         ; DATA XREF: sub_405E39+71↑o
B22 ; char aEndpoint[]
B22 aEndpoint       db 'ENDPOINT',0         ; DATA XREF: sub_405E39+82↑o
B2B ; char aServerPort[]
B2B aServerPort     db 'SERVER_PORT',0      ; DATA XREF: sub_405E39+93↑o
B37 ; char aConnectBackDel[]
B37 aConnectBackDel db 'CONNECT_BACK_DELAY',0
B37                                     ; DATA XREF: sub_405E39+AC↑o
B4A ; char aA[]
```

Figure. snippet from the customized TSHD backdoor for Windows

There are also some tools statically compiled such as winexe. The reason is to have tools that are self-contained and ready to execute and does not require any additional installations on the target machine.

The “Python” Directory

This directory is quite different from the previous one. It contains various exploits mainly written in python language. The first folder contains exploits for the well known MS17-010 vulnerability also known as the EternalBlue family. The code has been forked form the open source github repository MS17-010 and contains python scripts to conduct EternalBlue, EternalChampion, EternalSynergy, EternalRomance attacks, along with old Windows 2000 exploits.



```
2016 .
2016 ..
2016 MS17-010
2016 bin
2016 examples
2016 include
2016 lib
2016 pip-selfcheck.json
2016 share
```

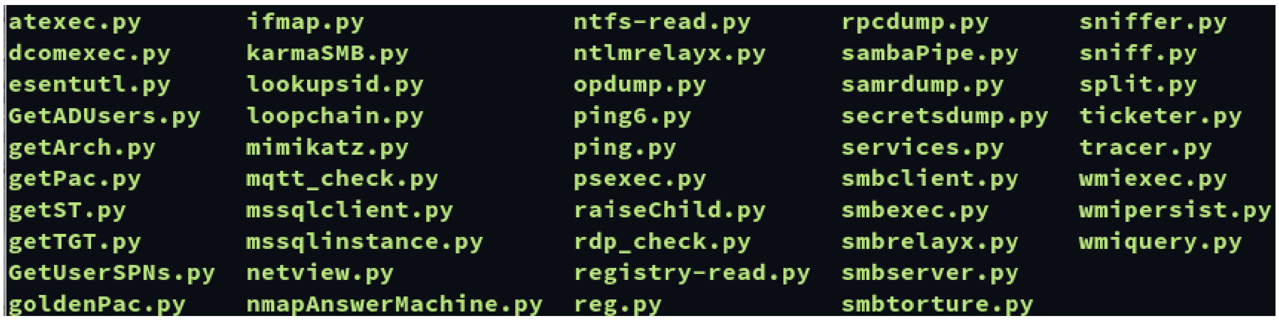
Figure. The "Python" directory

Inside the "python" directory, we also noticed the "pip-selfcheck.json" file, an installation artifact reporting the latest update of the cyber-arsenal dates back to 2018 as shown below:

```
{"last_check":"2018-05-26T06:06:28Z","pypi_version":"10.0.1"}
```

Inside the "example" sub-folder there are many reconnaissance tools able to gather technical information from the target systems via SMB, WMI, ActiveDirectory, System registry, NFS, Netbios and so on. Tools really useful to gather information about the Microsoft Windows perimeter of the victim network.

The tools in the "examples" folder are actually borrowed from the "Impacket" open source collection available on github.



```
atexec.py       ifmap.py              ntfs-read.py      rpcdump.py      sniffer.py
dcomexec.py     karmaSMB.py           ntlmrelayx.py     sambaPipe.py    sniff.py
esentutl.py     lookupsid.py          opdump.py         samrdump.py     split.py
GetADUsers.py   loopchain.py          ping6.py          secretsdump.py  ticketer.py
getArch.py      mimikatz.py           ping.py           services.py     tracer.py
getPac.py       mqtt_check.py         psexec.py         smbclient.py    wmiexec.py
getST.py        mssqlclient.py        raiseChild.py     smbexec.py      wmipersist.py
getTGT.py       mssqlinstance.py      rdp_check.py      smbrelayx.py    wmiquery.py
GetUserSPNs.py  netview.py            registry-read.py  smbserver.py
goldenPac.py    nmapAnswerMachine.py  reg.py            smbtorture.py
```

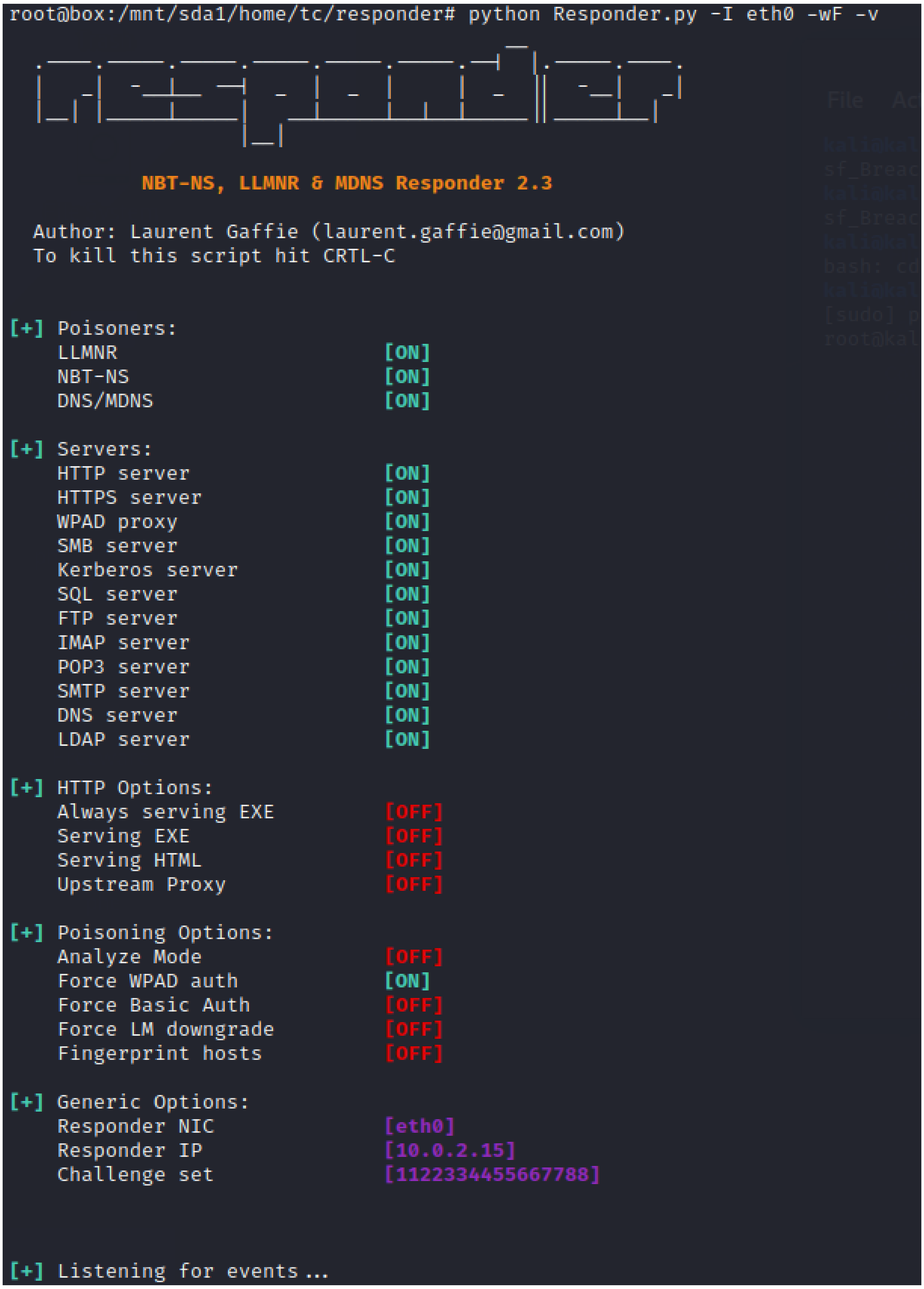
Figure. Contents of "example" sub-folder

The "Responder" Directory

This folder hosts the homonymous "Responder" tool. Responder is a LLMNR/NBT-NS/mDNS Poisoner written by Laurent Gaffie, and is able to listens on the wire for NetBIOS Name Service (NetBIOS) and Link-Local Multicast Name Resolution (LLMNR) broadcast and multicast requests for hostnames from other machines in the local subnet". Executed with the right parameters is able to snoop on NetBios and LLMNR in order to steal NTLMv1/v2 password hash.

The presence of this type of tool inside the cyber arsenal of the attacker means the threat group is really dangerous: man in the middle attacks across the company network are part of their modus operandi so their intrusions could be really dangerous in poorly segregated networks.

Anyway, the 2.3 version of the Responder toolkit dates back to 2016, suggesting this tool was one of the first tools installed on the virtual-arsenal.



```
root@box:/mnt/sda1/home/tc/responder# python Responder.py -I eth0 -wF -v
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 2.3

  Author: Laurent Gaffie (laurent.gaffie@gmail.com)
  To kill this script hit CRTL-C


[+] Poisoners:
    LLMNR                      [ON]
    NBT-NS                     [ON]
    DNS/MDNS                   [ON]

[+] Servers:
    HTTP server                [ON]
    HTTPS server               [ON]
    WPAD proxy                 [ON]
    SMB server                 [ON]
    Kerberos server            [ON]
    SQL server                 [ON]
    FTP server                 [ON]
    IMAP server                [ON]
    POP3 server                [ON]
    SMTP server                [ON]
    DNS server                 [ON]
    LDAP server                [ON]

[+] HTTP Options:
    Always serving EXE         [OFF]
    Serving EXE                [OFF]
    Serving HTML               [OFF]
    Upstream Proxy             [OFF]

[+] Poisoning Options:
    Analyze Mode               [OFF]
    Force WPAD auth            [ON]
    Force Basic Auth           [OFF]
    Force LM downgrade         [OFF]
    Fingerprint hosts          [OFF]

[+] Generic Options:
    Responder NIC              [eth0]
    Responder IP               [10.0.2.15]
    Challenge set              [1122334455667788]



[+] Listening for events...
```

Figure. Responder configuration

Inside the "Responder" directory there is also the "log" folder created during the first execution of the tool.

```
07/09/2016 11:39:19 PM - Responder Started: ['./Responder.py', '-I', 'eth0']
07/09/2016 11:39:19 PM - Responder Config: Settings class:
```

The Virtual-Arsenal Capabilities

After the analysis, we identified all the tools stored inside the malicious virtual machine and we are able to classify them into the following categories:

- Enumeration (T1261):
  Using tools to enumerate users through various protocols such as LPAD, NETBIOS, Active Directory and RPC.
- Network communication (TA0011):
  - Network pivoting communication using tcp, tftp, http tunnels
  - Packet manipulation tools like inpacket or scapy
- OS Credential Dumping (T1003):
  - Harvesting credential stored inside user and system configuration through open-source tools like LaZagne.
  - Obtaining higher privileges and authentication tokens using escalation tools like Mimikatz or system process dumping, and bypassing OS protection such as UAC.
- Exploit (T1404):
  compromising remote systems abusing known 1-Day vulnerabilities such as the MS17-010 ones.
- Man-in-the-Middle (T1557):
  Engaging their targets with Man in the Middle attack to manipulate network interaction to obtain hashes and tokens (e.g. through Responder)
- Living-off-the-Land tools (T1218):
  Many legit tools such as sysinternal tools such as “procdump” are ready to be abused in the toolkit, also linux utilities like “winexe”, “find”, ”touch” “grep”, “head”, “less” or “wget” have been cross-compiled to be ready for deployment on Windows environments too.

All of them are compiled in different ways (static or dynamic) and written in different languages (javascript, assembly, python etc.). This TTP confirms that the QEMU virtual machine is a complete medium of post-exploitation framework, which could be customized according to every need coming from the attacker.

If we think about the classic Lockheed Martin Cyber Kill Chain, we can collocate the Virtual Machine role in its mid and latest phase, the “Lateral Movements” and the “Action and Objectives”. In fact, the QEMU virtual machine was manually controlled through an SSH tunnel by the attackers, and it has been used as a powerful framework.

Conclusion

This QEMU based virtual arsenal is not the only tool in the satchel of this blurred threat actor. They are also able to leverage 0Day exploits - such as the CVE-2020-14871 described in Early Warning bulletin N031120 - and to leverage completely custom implants and tools to get in and move laterally even in the most segregated network. Also, the pivoting abilities of the group are really notable and the customization of the thsd backdoor observed in the virtual-arsenal is just one element of their modus operandi.

Follow-up reports will better describe how these actors have leveraged their edge abilities to threaten even the most cyber-mature companies.

Appendix

Indicator of Compromise

Hash:

```
fb7426ad06ee17fae29e4a46e36d92e7ba7a7cefaeeac2741eca6c535a1b3128
tc/deploy/LaZagne/LaZagne-32bits
afa814290bfea15a47d3462ae32d94f82e66ee888f7c51caf34b3212723c22ad
tc/deploy/LaZagne/laZagne.exe
d3d4f88012dc5b7deec6c54bef21e17f720d58aa00c8a809eb36d47038ca8db8
tc/deploy/LaZagne/LaZagne-64bits
05732e84de58a3cc142535431b3aa04efbe034cc96e837f93c360a6387d8faad
tc/deploy/procdump/procdump.exe
5b0f3ad95531dc16bb8de255186be66bf134fbdea4c1fbee38c807d98992c20c
tc/deploy/procdump/run.bat
dd8a9c4c59a7c7b07f21a6b3ac60405ee4c796cb3b268a9f6bd07fcdfc25cebd
tc/deploy/screen/svhost.exe
d6d4b69a277eac02b8b79c5e734f80d6cf1e0a4e967729a20079f7815de53794
tc/deploy/UACME_2.8.8/UacInfo64.exe
5afa7bd2ec1cc2abc91b37b0f800e2af11f3c796450c618e0f40e41efe756640
tc/deploy/UACME_2.8.8/Akagi64.exe
0f04ed31f345a3fcfe2e6a4c9022f02847df785ff9cd82147fccea5122646eba
tc/deploy/UACME_2.8.8/Akagi32.exe
b90cbef385708ae3a47b4fc96299e3f7c2979af439ed79cb20b355718fa263f4
tc/deploy/UserEnum/userslist.txt
edf35acd8eeeba68af9113afdedc21dc7d4ecb549da09195a4a9f25f4eb9941f
tc/deploy/UserEnum/UserEnum_LDAP.py
6dd57af6bd2049a6382ac7169e44aadef9353b905feb75e96abaae57199d4188
tc/deploy/UserEnum/README.md
c1f409a02ad9584551a17a7321db2ad448c6b2e5731d224201c5a173fb873cce
tc/deploy/UserEnum/UserEnum_NBS.py
54ecffbe2e97a127ee6820c891ba13f0b7ea5558b33e1ee731bdb772e5b97deb
tc/deploy/UserEnum/UserEnum_RPC.py
ffa5e945163ffb23d26a5dde041802219b03692e7af409e621ef92d6692dfbaf
tc/deploy/tshd/head.exe
df4e2115c80d07ca4345ba92053dcc38c4002554677a04509d02669a50ab86bf
tc/deploy/tshd/cygwin1.dll
c2ef6fc419630d566154f8372e94859df8141d02805bc7bce39c726a1ffef7c1
tc/deploy/tshd/grep.exe
70c5e7cd2926bb9849cffa6ae1c5559baf0ec4e3c896ae28bf219c9008f4c2c7
tc/deploy/tshd/find.exe
365ac8a166174bbc89fb24b21bfcd0b015950495bdf384ab830dd96d25e4cee3
tc/deploy/tshd/w_conf.exe
3faebbd216d5e94b696288d3089fff6ecb29fc23e97ceb2ff355341ac740d6a5
tc/deploy/tshd/uudecode.exe
2c73707fc79ff78846cc3c85383d47e46e495ef223d58e1e2933787fcfc2566a
tc/deploy/tshd/uuencode.exe
1ac28b748404d58b9f0c62d1ee65e3b444c9ad3ac0abea299238090b764bc25b
tc/deploy/tshd/wget.exe
0dd4d924c9069992dd7b3e007c0f3ca149b7fb1ce0dfb74b37c7efc6e1aebb46
tc/deploy/tshd/svchost.exe
ca301cde5b700ef7160cdf1f3acc6710da59958b8613dbe0abd2fd8120dfc0ed
tc/deploy/tshd/tail.exe
73723541d7a3c60456d69f0edff955bfde9db6e255821f6aee11f5f2a8a6466b
tc/deploy/tshd/run.xml
dffffc9bfaa0b41674bbffcf93764f5d04e218a454dc5ab93a830f8ee19722a7
tc/deploy/tshd/touch.exe
edf649392001017219a27e07b9c33b3a1ebd074d1f0b769a6e8928833271b1c3
tc/deploy/nc_send/sendnc.py
a70334114ee71a28aab1f992a1a6ff5b894433066859f8bf87fe117b6b0dd288
tc/deploy/nc_send/outlooks.exe
7d33f24ae4c7b3024d5cec2a31420be857f0e547de8971dd6dea169119d4f348
tc/deploy/nc_send/tar.exe
875f234ed1c172ad8deca6d9c35270c1426d25765653600b2b899efa9f9a966f
```

```
tc/deploy/nc_send/nc.tar
50cf763baf747c0094885bc1d129fb97211c618e316ea476c0dfeffeddf9db42
tc/deploy/nc_send/mimi32.exe
b3ec0b621d523f8182cf8409cb1c3d29553d56442e75dcac964dfae82d2c1bc9
tc/deploy/nc_send/sendncmim.py
632be2363c7a13be6d5ce0dca11e387bd0a072cc962b004f0dcf3c1f78982a5a
tc/deploy/nc_send/mimi64.exe
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
tc/deploy/nc_send/name_mimi_sysinit.exe
23e5fd457a251d3f87920727a12bcf2e70b30901597309564015eddd12b94a1c
tc/deploy/nc_send/mimi_old.exe
f5c1304be270e271e902f0229ab8d876c9ed63cbf4fe926dd1ab61f7335187a4
tc/deploy/mimi_uac/nc32.exe
a0ee6605e8fc9fd2c397b7cd7ddd1664b08e2e6c9f72ab9e658ec5859876da67
tc/deploy/mimi_uac/nc64.exe
8edb867830e81c060f715b365be834677949f3fb7b7649852e6087a0f8adb115
tc/deploy/mimi_uac/uac_status.txt
c27382fd82bd4af92905144b6b219c3b75cb001081f9ae683115d50d2df8382a
tc/deploy/mimi_uac/update.js
24aed7103a56557d5eddf54460cc9ae59e3f1ddc695c9d403e141c61e96059af
tc/deploy/mimi_uac/a.ps1
43c48658feb11645b32e26b8dd3db2736083047e1088cf813da75c95d50d17c5
tc/deploy/domain_loginlog/domain_loginlog.exe
77820dc7dba2412933210d7ea40c11640500b54d3ac14d317187a9c2f6a96645
tc/deploy/winexe-static-071026
8530dc274a9154a916af2595c4d48824e8b4015745cd452a634ae55d2786c902
tc/deploy/screen_XP/svhost.exe
0abc3962c668e457beb043c2455e30585e1da8732ab42e0130fd729a8dc7ebc4  tc/python/MS17-
010/mysmb.py
f64024054e3e3c9cf011b27d768bd3692fa9430ae3ec39e4e7a18133d364300c  tc/python/MS17-
010/infoleak_uninit.py
515284a34b406042f0eea228a5adaf705f674826999525f2d7c7f13512d5dc72  tc/python/MS17-
010/nc8887.read
1b9833f28868d5a39d927f1a18f89073b82c322574b2214228201e35088314ed  tc/python/MS17-
010/npp_control.py
34022a65a3eb93b109ed4c6e1233c6404197818a70f51ab654e2c7e474ee2539  tc/python/MS17-
010/eternalblue_exploit8.py
f5c1304be270e271e902f0229ab8d876c9ed63cbf4fe926dd1ab61f7335187a4  tc/python/MS17-
010/outlooks.exe
64cf03ed475f4486147cd2cedb78db4aa7164f57b3a2c25776ed1fb28853d7e5  tc/python/MS17-
010/nc8887_.py
0b0949ea092aea52f258865b278702aa1d55558a3a349805fb970ee1439f7964  tc/python/MS17-
010/checker.py
b60da0fe1946329c43fbde55fa3543510830b04959605e4b9f8ea75d4451d445  tc/python/MS17-
010/exploit.py
a2ccf5c039464e67ff0a372f91f6e89999ee7c0ea44a6cba493e0aec28954023  tc/python/MS17-
010/shellcode/eternalblue_sc_merge.py
493e3faaef103c8afd4d713b1447c5489e551892f42eba1b9383532024cdd107  tc/python/MS17-
010/shellcode/eternalblue_kshellcode_x64.asm
a9fdd64ccc3dfba679dd796d4e3e42e1581f48555fb47d4662f1cb4191fe1a71  tc/python/MS17-
010/shellcode/sc_all.bin
312d0e8913c9d1037669a73ce07f8df98af2a6a3c9c72cb2fbd29a7857686379  tc/python/MS17-
010/shellcode/eternalblue_kshellcode_x86.asm
c174f89004c2fb3e91ab8233794d055340cd2a9520dc2be8b938ebccf1c74a74  tc/python/MS17-
010/eternalblue7_exploit.py
7a0774c5872df12686735efc631aa83a78bf1b6211d77ccd9a2ae0ff0adfb58c  tc/python/MS17-
010/nc8887.py
e9073f672596429eab45efe3e79e36e361fb220b71f4c47b32edbc6c51544494  tc/python/MS17-
010/eternalsynergy_leak.py
```

```
3f106f73b51516fccb1d62265248ee03ccadf86377d66ef53a672729096d2cf3  tc/python/MS17-
010/eternalromance_poc.py
2e2332d9119ca0075db133111ef9dfd5577cedc8df25d6a603755005a787178c  tc/python/MS17-
010/BUG.txt
eb53bc507b64d43b3702bcabd662eddbbf468d8144e8d611fcca78bd7101cd08  tc/python/MS17-
010/del_outlooks.py
2c84ce6f127ac559658ad2f5cbb5ead99c0bce27feae2f2cffcf0f1a5bc77f19  tc/python/MS17-
010/eternalblue_poc.py
c49ec4e145fa4dfc63b5fe6655a84056304e61f776e3a4125b507d9f6d5fb315  tc/python/MS17-
010/eternalromance_poc2.py
d20a5fce1a3fdd7b031e1f20a78206187541d6ba10d9e2d0a6472526cea2c746  tc/python/MS17-
010/eternalchampion_poc.py
8e21af3c2840ff374ef5c4f98d5bd665482241c66d7fe1172023cb67ece80079  tc/python/MS17-
010/README.md
99a4ded26895422707f7c92eca9c9d64212cc033c50010fb027fe32ab55386d9  tc/python/MS17-
010/eternalblue_exploit7.py
2dfe1fc676fe8f5c949ac7a15491b4081a8dd8d11a3baa3442be539fe7e12e26  tc/python/MS17-
010/42315.py
896610790bfa3554722518d81cd7692ba3cc963d1fd82bc6c57f7b2df7962625  tc/python/MS17-
010/eternalchampion_leak.py
d37670ef452b3850d2a7d590ab3bee83902f3644cdba4e9b52fe8a2deb85402f  tc/python/MS17-
010/eternalsynergy_poc.py
5e58130b5598379d83300aea616d3f21aa6037e50aa41bac59dcfb993873868c  tc/python/MS17-
010/zzz_exploit.py
dd4798af2c60dd83852bb9f097bf82b332e6408d0c9362a477592397468553ca  tc/python/MS17-
010/eternalchampion_poc2.py
89e30158f62eb2e60763bc9701d750e61ede148793b411c45898c0b36f467b78  tc/python/MS17-
010/deluser.py
e2022ebe819476f715b10e441ef13171317c91cfaf553302c90b77ea686b72b5  tc/python/MS17-
010/eternalromance_leak.py
5dd68dc09454edb1fa4f847819e3ae48fbedacef5413f2a9ef9957602a6ad97a  tc/python/MS17-
010/adduser.py
624bcdae55baeef00cd11d5dfcfa60f68710a02  responder/LICENSE
7ca9a5cb7034b04ea6060c7f7804997b9f8ba411  responder/README.md
0833c52e7c2afce7ff357ee496bc2b3c6662edba  responder/Responder.conf
aca7fc6f37f789ef7a5816dce83ac4efaaa76a35  responder/Responder.py
90fa9a2d1db2e143eb2999f80a615f997b916407  responder/certs/gen-self-signed-cert.sh
cde48c263fe556d21f0dacfee746b73a9d0f843c  responder/certs/responder.crt
9439fb0577b485bea2ab5515d22ce028b1edefd7  responder/certs/responder.key
3c1be1e572a4a475a4499d0c87979005a4927a1a  responder/files/AccessDenied.html
da35e993ca6b2f8a73bef404a32391ae2a6f6b3e responder/files/BindShell.exe
1bb89403f629f02091397a7c34e99c2e35b7e74  responder/poisoners/LLMNR.py
5e4d413602268e9bdabd48c486ecf164c3a188ea  responder/poisoners/LLMNR.pyc
aebe1412a78a904badfa7cbed4f3ece351af6a55 responder/poisoners/MDNS.py
a048219fd8fcef6a98cb2be309135e44efcec006  responder/poisoners/MDNS.pyc
5860c2fd3cd1a4e7203ff943753a7fbf656951dd  responder/poisoners/NBTNS.py
27deef9453ee7102f0bc42380a355124b6ad7d6a responder/ poisoners/NBTNS.pyc
da39a3ee5e6b4b0d3255bfef95601890afd80709  responder/poisoners/__init__.py
3e6de0af6b7c6c1aaaae7f4b5ecc3fc43c5b7859  responder/poisoners/__init__.pyc
874ed910bf04d409d3639c2e14776c452eb1755e  responder/tools/BrowserListener.py
8b1aac92e1a185855a4b5a2f55e14b9817f95aaa  responder/tools/DHCP.py
4b22f17fd4ec78fe4b11a98ad882970a7c55b9ef  responder/tools/DHCP_Auto.sh
cf80e9f2f99ed0778fe3ac209324bf9a84be6b1f  responder/tools/FindSMB2UPTime.py
ad09d51ecbdddd57cd0e1845ed6bd7a1c863a196  responder/tools/FindSQLSrv.py
511326ff4ed1876b6b59ccfe6b8471a27309ea00  responder/tools/Icmp-Redirect.py
3e6b7ffe886764b31757ce1bb0fd9a1854246a97  responder/tools/RelayPackets.py
23622c8ff7baea6cac44f08ff681b37a9ee9fdd1  responder/tools/SMBRelay.py
a29540e984808a029425be53ca93ce3f8fd79a27  responder/servers/Browser.py
5b1f743c91c868204348de30c2ba0dcc03b87833  responder/servers/Browser.pyc
```

```
e5dc55eecb82c1d40a4b3492ced9bf19f2dae0b4  responder/servers/DNS.py
40fa18da8bed7c4b8c42a5a038408988ab31ff82  responder/servers/DNS.pyc
7765a0a1b66a58ae487cf76c2ec43f88c767e8dd  responder/servers/FTP.py
6c03f7b5451a3b8bdc693a02dc61d0b78269a5d6  responder/servers/FTP.pyc
adaa025b5cf015769213738ead37ce7f032d203d  responder/servers/HTTP.py
d21d59dd2136e5fea0634a1cc6c7b36511025f67  responder/servers/HTTP.pyc
26939dee3f00cfca80ae62745fc4b8a987e93a49  responder/servers/HTTP_Proxy.py
10f4d968f6410179ea03a330984136cb6fffc83c  responder/servers/HTTP_Proxy.pyc
088b3ece595950ab4e471e4763bbd400ff1fca1f  responder/servers/IMAP.py
9f8014a0150badc732d07cfc381785975dee6cd7  responder/servers/IMAP.pyc
2e1ace2fc5a63000cf71510a02e3221880c094a4  responder/servers/Kerberos.py
6ceee9bf498f443817d8603ac1692c639ad8a59e  responder/servers/Kerberos.pyc
0bdac1da920a1c8177cf4f2abc147710f1b5ec09  responder/servers/LDAP.py
4cfc6d04b0311f27d170c860eb8ff7e05769a502  responder/servers/LDAP.pyc
e62c3b201a99501a626c35dc084a2201f59e2bd6  responder/servers/MSSQL.py
f49f6878303358fa15f87bd05041515605aab802  responder/servers/MSSQL.pyc
a095b50743189513f2c62033127dd5ab23e4c3ec  responder/servers/POP3.py
32dd92688f6ab0c57acbeb63ff3b093d90440a7e  responder/servers/POP3.pyc
8e229944b428f675d27a5c99e71496907a9a17d1  responder/servers/SMB.py
2533c2a30cb4f06d8c5fa9259e7d3d42ea40ca22  responder/servers/SMB.pyc
a472ed6415e0c00c4c4320468dcb65256138ffa5  responder/servers/SMTP.py
ad49be1fe6e6e732b875bac888d693a7c39f8132  responder/servers/SMTP.pyc
da39a3ee5e6b4b0d3255bfef95601890afd80709  responder/servers/__init__.py
369a6285d5047feadc3bad34bd5d914766672b81  responder/servers/__init__.pyc
ef7d632acf72b04b6cf7500b21c4c8efe7612d4a  responder/fingerprint.py
 4c9b5680c7575e0c08c8ff1511e4553c0136c743  responder/fingerprint.pyc
a8236535d40dfdd2ae9b24aecbf1ba7e65313ce3  responder/odict.py
718d1d180ff58c198713bb27544f5f8fe5b35fc6  responder/odict.pyc
cc8af2c71cc4cead9ee99c268e075d2e0cb8f32d  responder/packets.py
 de600e4c57efc08ac86c63ee3a003955fc00bdc6  responder/packets.pyc
2cdc364e88955d8176e32e2c35026d325883c157  responder/settings.py
7e73cf9b4b59db4df95c8d00915ba2370b8cb538  responder/settings.pyc
9e6a51c50ec8c9bec41ca2c060c2029f36997a67  responder/utils.py
0a26c8278351f8969e7b1c63f8e4ed69cc087925  responder/utils.pyc
```

Yara Rules:

```
import "pe"
import "math"

rule svchost_backdoor_UNC1945{
        meta:
        description = "Yara Rule for svchost.exe backdoor of the UNC1945 arsenal"
        hash=”428b47caf74ce986bc3688262355d5b7”
        author = "Yoroi Malware Zlab"
        last_updated = "2020_11_20"
        tlp = "white"
        category = "informational"

strings:
$s1="PROXYHOST"
$s2="PROXYPORT"
$s3="USERNAME"
$s4="PASSWORD"
$s5="ENDPOINT"
$s6="/cygdrive/c/windows/system32.log"
$s7={40 20 36 [4] C7 40 24 36 [4] C7 40 28}

condition:
   uint16(0) == 0x5A4D and
   uint32(uint32(0x3C)) == 0x00004550 and pe.number_of_sections == 5 and

for any i in (0..pe.number_of_sections -1 ) : (
math.entropy(pe.sections[i].raw_data_offset, pe.sections[i].raw_data_size) >
7.2781 and pe.sections[i].name==".data" ) and ( pe.sections[2].name=="/4" ) and
all of them }
```

This blog post was authored by Luigi Martire, Antonio Pirozzi and Luca Mella of Yoroi Malware ZLAB