FireEye | CYBER DEFENSE SUMMIT

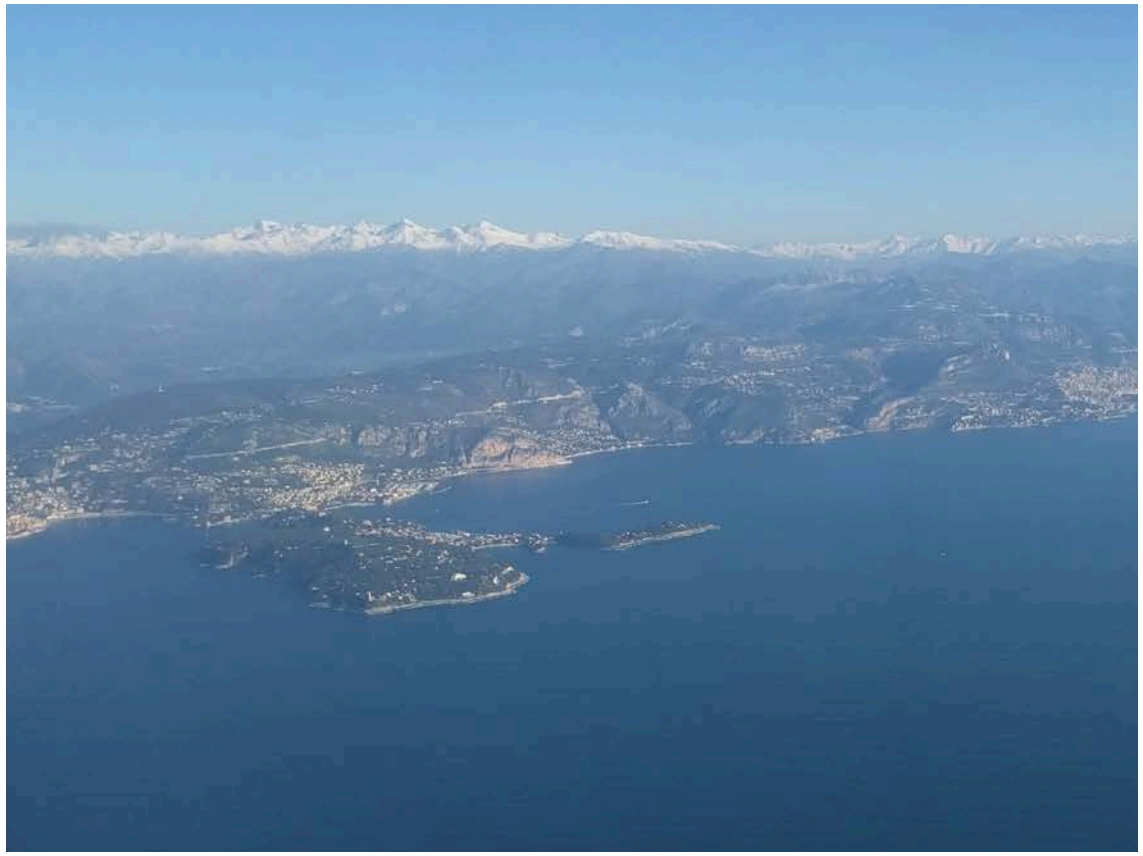
Hunting for PLATINUM

Mandiant consulting

OCTOBER 1 – 4, 2018 | WASHINGTON, D.C.

Adrien Bataille

- Senior Consultant at Mandiant
- 10 years experience
  - Incident Response
  - Malware analysis
  - Threat intelligence



Matias Bevilacqua

- Principal Consultant at Mandiant
- 15+ years experience
  - Incident Response
  - 5 years experience on IR/forensics R&D.
  - 2 years onsite focused on the Middle East threat landscape.

Disclaimer

"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."

Platinum overview

Microsoft Report in April 2016

- Low profile group
- Targets - South / Southeast Asia
- Custom backdoor and tools
- At least four zero-day exploits
- Hot-patching technique to inject code

Microsoft Blog post in June 2017

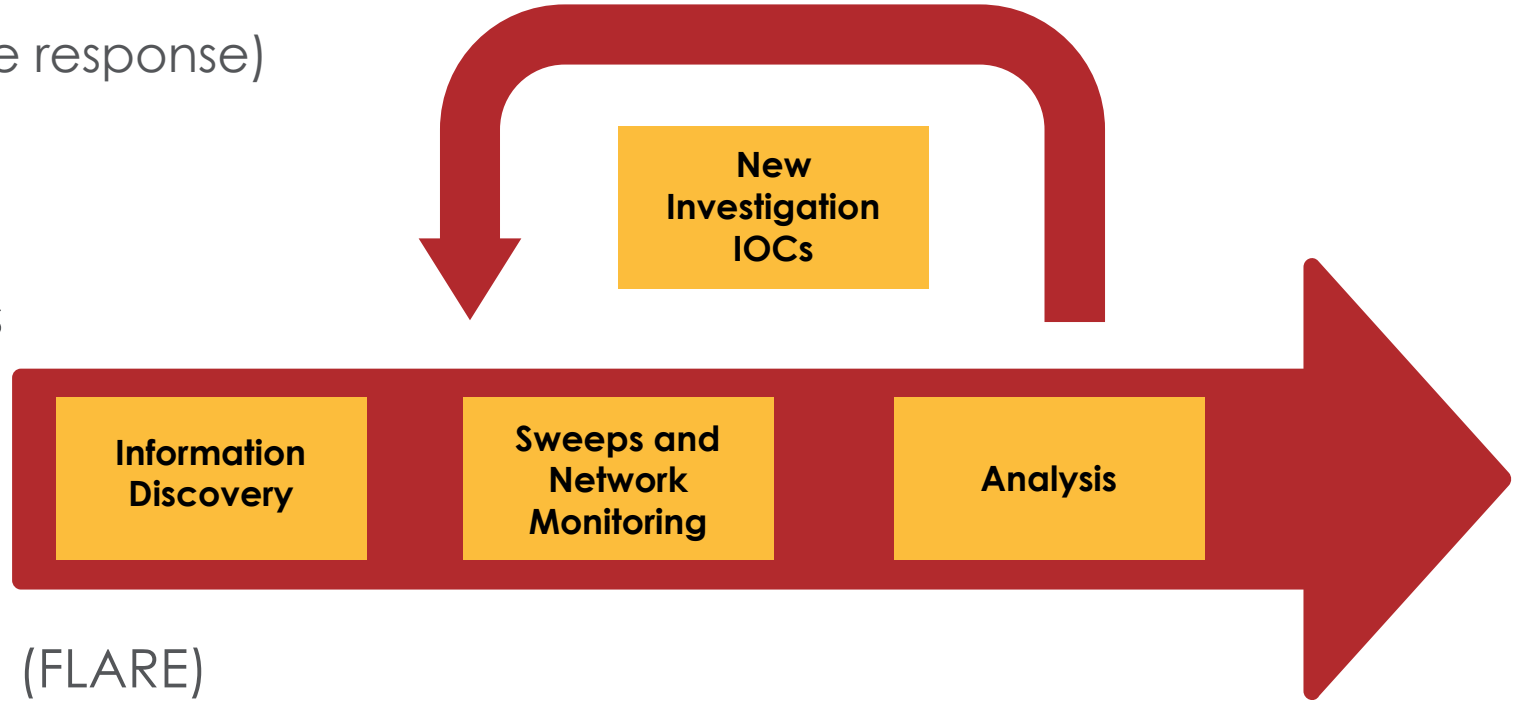
- Use of Intel AMT Serial over LAN for communications
- OS based firewall bypass
- Seen in file transfer tool

Mandiant case

- Mandiant called in to investigate suspicious activity on a handful of hosts
- Compromise confirmed
  - Breach attributed to a known threat group (not PLATINUM)
  - Use of Metasploit, webshells, ZXShell, plink… fairly low-tech and noisy.

Investigative cycle

- Host inspection
  - HX used for analysis (triage, live response)
  - HIP for real-time alerting
- Network monitoring / analysis
  - PX network sensors
  - Client network infrastructure
- Malware reverse engineering (FLARE)



Suddenly things got interesting…

- What is this?…

```
C:\Program Files\Common Files\System\temp\alg.exe a -hpqwerty -r -dh ps_dl.dat "C:\Program Files\receptor\"

C:\Windows\Temp\alg.exe a -hpqwerty -r -ri3 -dh temp_001.dat agent_config.json ragent_install.log connector-deployment-guide.pdf AgentSetup_HIP_xAgent_Bundled.msi
```

- Live response analysis
  - Unknown malware in folder C:\Program Files\7-Zip\Lang\
  - Freshly installed
  - Regular Powershell event log clearing
  - Persistence unknown
  - Submitted to FLARE for triage, named REDPEPPER

```
C:\Program Files\7-Zip\Lang\bg.txt
C:\Program Files\7-Zip\Lang\br.txt
C:\Program Files\7-Zip\Lang\bz.txt
C:\Program Files\7-Zip\Lang\ca.txt
[…]
C:\Program Files\7-Zip\Lang\id.txt
C:\Program Files\7-Zip\Lang\iis.txt
C:\Program Files\7-Zip\Lang\io.txt
C:\Program Files\7-Zip\Lang\ios.txt
C:\Program Files\7-Zip\Lang\it.txt
C:\Program Files\7-Zip\Lang\iys.txt
C:\Program Files\7-Zip\Lang\ja.txt
C:\Program Files\7-Zip\Lang\kaa.txt
```

REDPEPPER

- C:\Program Files (x86)\7-zip\Lang\bz.txt
- C:\Program Files (x86)\7-zip\Lang\iis.txt
- C:\Program Files (x86)\7-zip\Lang\ios.txt
- C:\Program Files (x86)\7-zip\Lang\iys.txt

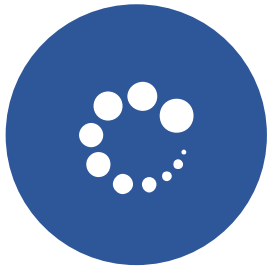
- WMI persistence
  - Filter "7zip_32"
  - Consumer triggered on event log message "The Security Center service entered the running state."
  - Base64 encoded powershell loader loading an encrypted shellcode and final payload
- Backdoor with statically linked OpenSSL (1.5MB)
  - 3DES encrypted configuration file
  - 3DES to HTTP/HTTPS C2
  - SSL communications
- Working hours and self-delete timer
- Evolution of Microsoft's "adbupd" backdoor
  - but does not trigger on the YARA rule

Hunting for REDPEPPER

- C:\Program Files (x86)\7-zip\Lang\bz.txt
- C:\Program Files (x86)\7-zip\Lang\iis.txt
- C:\Program Files (x86)\7-zip\Lang\ios.txt
- C:\Program Files (x86)\7-zip\Lang\iys.txt

PE metadata

- Known bad filenames
- Payload has single export name "GetStartObjectEx"
  - Encrypted on disk
  - Use of custom PE loader – won't appear as DLL export in running process
  - Search strings in process address space (noisy)



Persistence/Loading

- WMI filter and consumer
- Search for base64 encoded Powershell from "iis.txt"
  - Requires a special regex audit
  - Resource heavy
- EID 104 specific to Powershell: The Windows PowerShell log file was cleared

Real-time

- HIP alerts on obfuscated Powershell

Hunting for REDPEPPER

- C:\Program Files (x86)\7-zip\Lang\bz.txt
- C:\Program Files (x86)\7-zip\Lang\iis.txt
- C:\Program Files (x86)\7-zip\Lang\ios.txt
- C:\Program Files (x86)\7-zip\Lang\iys.txt

- Network
  - C2s from the configuration files
  - Old user-agents
    - IE7, IE8, IE9, IE10
    - Chrome 14.0.835.187
    - Firefox 3.6.13, 4.0, 5.0

```
GET /stocks/deals.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/5.0 (Windows NT 5.1) AppleWebkit/535.1 (KHTML, like Gecko) Chrome/1
4.0.835.187 Safari/535.1
Host: [redacted]
Connection: Keep-Alive
Cache-Control: no-cache
```

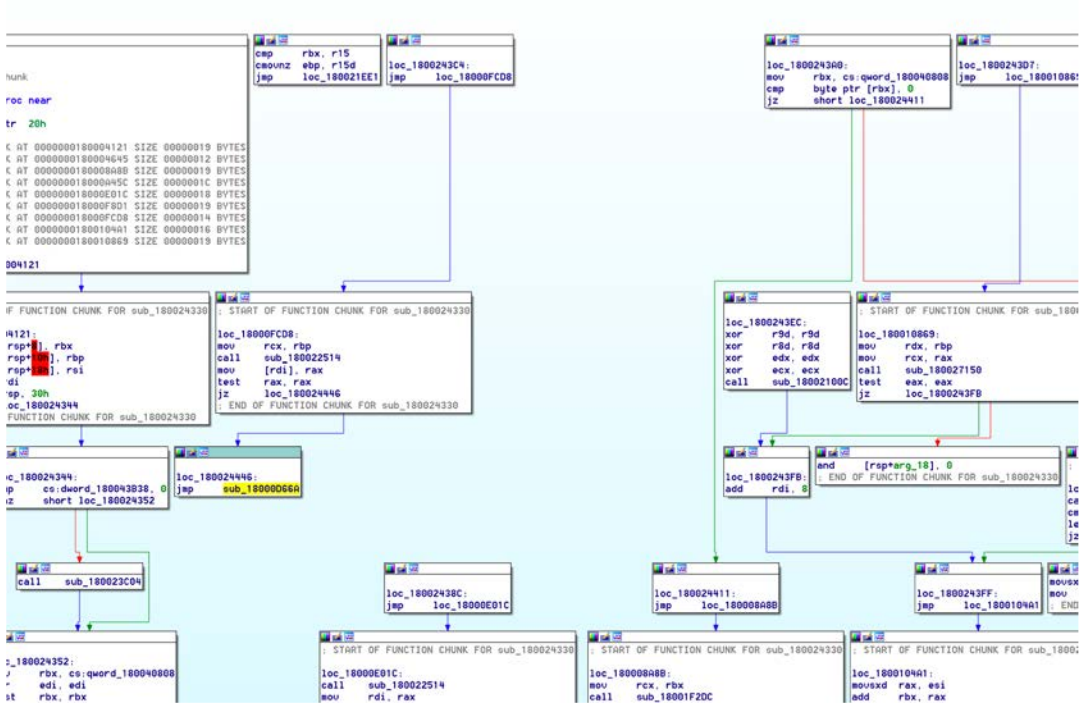
Continuing the investigation

- Hits on WMI and obfuscated PowerShell
  - REDPEPPER - 64 bit version with different filenames and path
  - Several unknown hits using a similar loading technique
    - *C:\ProgramData\Apple\setup\BonjourSupport.txt*
    - *C:\Program Files\Windows NT\TableTextService\TableTexTServiceDaGG.dll*
    - …
- Live response analysis suspects different malware
- *TableTexTServiceDaGG.dll* submitted to FLARE for analysis
- Was eventually named REDCURRY

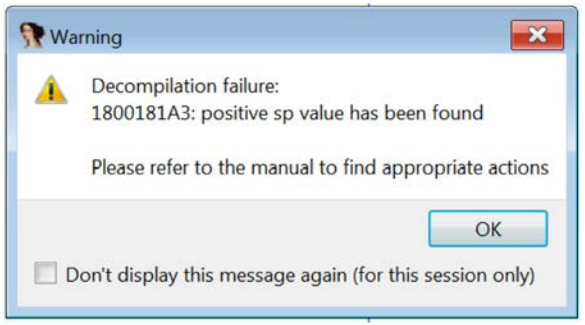
- C:\Program Files\WinRAR\Formats\tar.txt
- C:\Program Files\WinRAR\Formats\bz.txt
- C:\Program Files\WinRAR\Formats\cab.txt
- C:\Program Files\WinRAR\Formats\gz.txt

REDCURRY ("KB")

- Same loading technique as REDPEPPER
  - WMI persistence
  - Several base64 encoded files
- Final payload this time was packed, and obfuscated
- Keylogger, records keystrokes in encrypted files
- Found several versions
- Encryption keys slightly different per victim (a few bytes)
- Unpacked payload identified as Trojan_Win32_PlaKeylog_B by Microsoft YARA ruleset (PLATINUM)



Unpacked disassembly, duh.

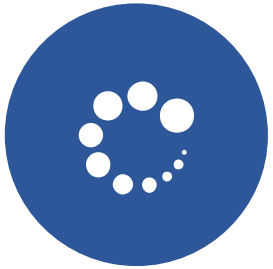


Hunting for REDCURRY

- C:\Program Files\Internet Explorer\F112props.dll
- C:\ProgramData\CyberLink\cbd\tokens.txt
- C:\Program Files\Windows NT\TableTextService\TableTexTServiceDaGG.dll

PE metadata

- Known bad filenames
- Payload has four exports "ExLinkFn", "ExLinkFnW", "ExUnlinkFn", "ExUnlinkFnW"
- Encrypted on disk
- Use of custom PE loader – won't appear as DLL export in running process
- Search strings in process address space



Persistence/Loading

- WMI filter and consumer
- Search for base64 encoded Powershell or "MZ" header
  - Requires a special regex audit
  - Resource heavy
- EID 104 specific to Powershell: The Windows PowerShell log file was cleared

Real-time

- HIP alerts on obfuscated Powershell (correlation observed to PS Empire)

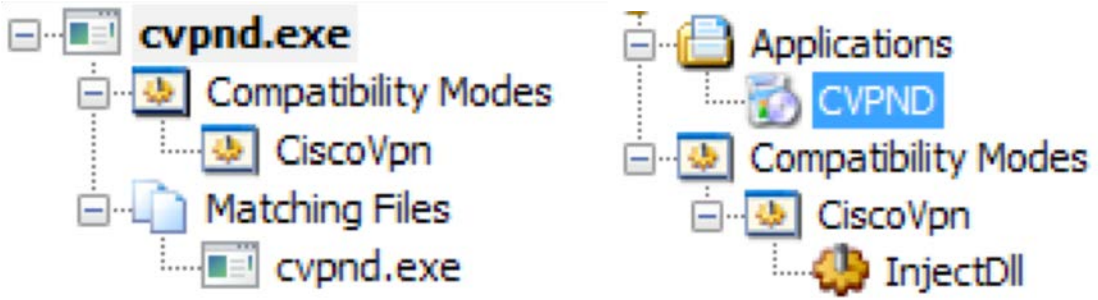
Hunting for more

- ShimCache/AmCache sweeps
- Simple & inaccurate description: records executed binaries (win)
- Good method to find files which may have been be deleted
- Hits for some of the preferred filenames of this threat actor
- Hits on « alg.exe » => live responses

- Unknown malware found!
  - C:\Program Files (x86)\Cisco Systems\VPN Client
  - Different loading than REDPEPPER and REDCURRY
  - Eventually named REDSALT.

C:\Program Files (x86)\Cisco Systems\VPN Client\cisco_cert_mgr.exe
C:\Program Files (x86)\Cisco Systems\VPN Client\cisconetworks.dat
C:\Program Files (x86)\Cisco Systems\VPN Client\cvpnd.exe
C:\Program Files (x86)\Cisco Systems\VPN Client\internal.ini
[...]
C:\Program Files (x86)\Cisco Systems\VPN Client\qt-mt335.dll
C:\Program Files (x86)\Cisco Systems\VPN Client\qtwidgets64.dll
C:\Program Files (x86)\Cisco Systems\VPN Client\routexchange.dll.mui
C:\Program Files (x86)\Cisco Systems\VPN Client\SetMTU.exe
C:\Program Files (x86)\Cisco Systems\VPN Client\VAInst64.exe
C:\Program Files (x86)\Cisco Systems\VPN Client\vpnapi.dll

REDSALT



- Backdoor functionality
- Persistence using ACI shims
  - Shim loads obfuscated 1st DLL (95% junk code)
  - 1st DLL decrypts and loads a second DLL (Rijndael)
  - 2nd DLL decrypts and loads resource (optional) which is packed payload
- Found several versions
  - ACI shim and loader tuned for each compromised host
  - Final payload often the same
- Final unpacked payload identified as Trojan_Win32_Plagon by Microsoft YARA rule

REDSALT

Persistence	• Service, COM, Winlogon events, ACI shims • Bootkit (pre-Vista)?
Configuration	• File or registry • Each entry is RC4 encrypted • Key changes for each sample
Active mode	• Encrypted file dropped on the system • HTTP/HTTPS with configured URLs (Wininet API)
Passive mode	• Door knocking - raw sockets wait for a "magic" packet then open ports • Capability of proxying to next host • Communications using IPV4/IPV6 or Intel AMT SOL on recent versions • Compatible with "Zc tool" referenced in MS report

REDSALT – AMT SOL

```
*(v2 + 20) = 0;
v4 = CreateFileA("\\\\.\\COM3", 0xC0000000, 0, 0, 3u, 0, 0);
*(v2 + 8) = v4;
if ( *(v2 + 12) )
{
  if ( *(v2 + 16) )
  {
    if ( v4 != -1 )
    {
      DCB.DCBlength = 0;
      memset(&DCB.BaudRate, 0, 0x18u);
      CommTimeouts.ReadTotalTimeoutConstant = 500;
      CommTimeouts.WriteTotalTimeoutConstant = 500;
      v7 = 115200;
      v8 = 256000;
      v9 = 384000;
      v10 = 512000;
      v11 = 1024000;
      CommTimeouts.ReadIntervalTimeout = 200;
      CommTimeouts.ReadTotalTimeoutMultiplier = 0;
      CommTimeouts.WriteTotalTimeoutMultiplier = 0;
      if ( GetCommState(v4, &DCB) )
      {
        do
        {
          DCB.BaudRate = *(&v7 + v3);
          if ( !SetCommState(*(v2 + 8), &DCB) )
            break;
          ++v3;
        }
        while ( v3 < 5 );
        if ( SetCommTimeouts(*(v2 + 8), &CommTimeouts) )
          *(v2 + 4) = 1;
      }
```

```
if ( networkObj )
{
  if ( networkObj->flag_AMT )
  {
    result = AMT_Send(&g_AMTstruct, buf, len);
  }
  else
  {
    v5 = send(networkObj->socket, buf, len, flags);
    if ( v5 > 0 )
    {
      v1 = GetTickCount();
      networkObj->field_14 = 0;
      networkObj->timer = v1;
    }
    result = v5;
  }
}
else
{
  result = -1;
}
return result;
```

REDSALT - interactive mode

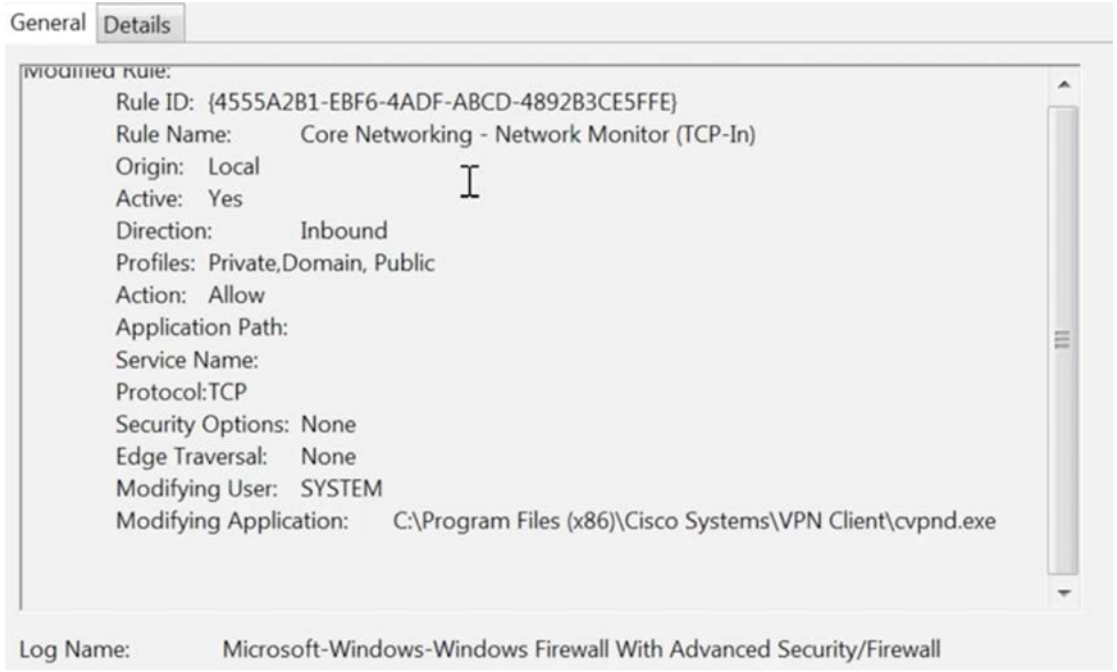
```
C:\Users\Administrator\Desktop>zc.exe 192.168.22.150 445 dcc
:3389:3389
~~~~

OK
.....
OK
dcct3st2018

[WINDOWS-PC2]>HP

1) Command:RS
2) Command:TS
3) Command:RC
4) Command:TN
5) Command:PN
6) Command:TTN
7) Command:HP
8) Command:IF
9) Command:UC
10) Command:EX
11) Command:[UI]
12) Command:[QR]
[WINDOWS-PC2]>
```

```
[WINDOWS-PC2]>PN
I:192.168.22.230
TP:445
C:dcct3st2018:D:1300:1300
...
OK
.....
...
dcct3st2018

[WINDOWS7-PC3]>
```

```
@C:\Windows\Temp\>del m64.exe
del m64.exe

Deleting: C:\Windows\Temp\m64.exe with 1 pass(es)
    1 file deleted with 1 passes
```

Hunting for REDSALT

- Event logs
  - Program-Telemetry logs for InjectDLL shims are a gold mine
  - Firewall rule changes (hardcoded names)
- Persistence – search for small sdb
- Unique export "StartThreadAtWinLogon"
  - Search in memory
  - Stack string in 1st stage DLL



Hunting for REDSALT

Compatibility fix applied to C:\Windows\SysWOW64\netstat.EXE.
Fix information: InjectDll, {3432bc96-d181-4529-b261-1a3964961b6c}, 0x40206.

- Example event logs for InjectDll shim

500 | Information | Compatibility fix applied to C:\Program Files (x86)\Cisco Systems\VPN Client\cvpnd.exe. Fix information: InjectDll, {3432bc96-d181-4529-b261-1a3964961b6c}, 0x00010206.

500 | Information | Compatibility fix applied to C:\Windows\SysWOW64\netsh.exe. Fix information: InjectDll, {3432bc96-d181-4529-b261-1a3964961b6c}, 0x00040206.

500 | Information | Compatibility fix applied to C:\Windows\SysWOW64\netsh.exe. Fix information: InjectDll, {3432bc96-d181-4529-b261-1a3964961b6c}, 0x00040206.

500 | Information | Compatibility fix applied to C:\Windows\SysWOW64\netsh.exe. Fix information: InjectDll, {3432bc96-d181-4529-b261-1a3964961b6c}, 0x00040206.

500 | Information | Compatibility fix applied to C:\Windows\SysWOW64\netsh.exe. Fix information: InjectDll, {3432bc96-d181-4529-b261-1a3964961b6c}, 0x00040206.

500 | Information | Compatibility fix applied to C:\Windows\SysWOW64\ipconfig.EXE. Fix information: InjectDll, {3432bc96-d181-4529-b261-1a3964961b6c}, 0x00040206.

500 | Information | Compatibility fix applied to C:\Windows\SysWOW64\netstat.EXE. Fix information: InjectDll, {3432bc96-d181-4529-b261-1a3964961b6c}, 0x00040206.

500 | Information | Compatibility fix applied to C:\Windows\SysWOW64\netstat.EXE. Fix information: InjectDll, {3432bc96-d181-4529-b261-1a3964961b6c}, 0x00040206.

500 | Information | Compatibility fix applied to C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.EXE. Fix information: InjectDll, {3432bc96-d181-4529-b261-1a3964961b6c}, 0x00040206.

500 | Information | Compatibility fix applied to C:\Windows\SysWOW64\systeminfo.EXE. Fix information: InjectDll, {3432bc96-d181-4529-b261-1a3964961b6c}, 0x00040206.

500 | Information | Compatibility fix applied to C:\Windows\SysWOW64\netstat.EXE. Fix information: InjectDll, {3432bc96-d181-4529-b261-1a3964961b6c}, 0x00040206.

500 | Information | Compatibility fix applied to C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.EXE. Fix information: InjectDll, {3432bc96-d181-4529-b261-1a3964961b6c}, 0x00040206.

500 | Information | Compatibility fix applied to C:\Windows\SysWOW64\systeminfo.EXE. Fix information: InjectDll, {3432bc96-d181-4529-b261-1a3964961b6c}, 0x00040206.

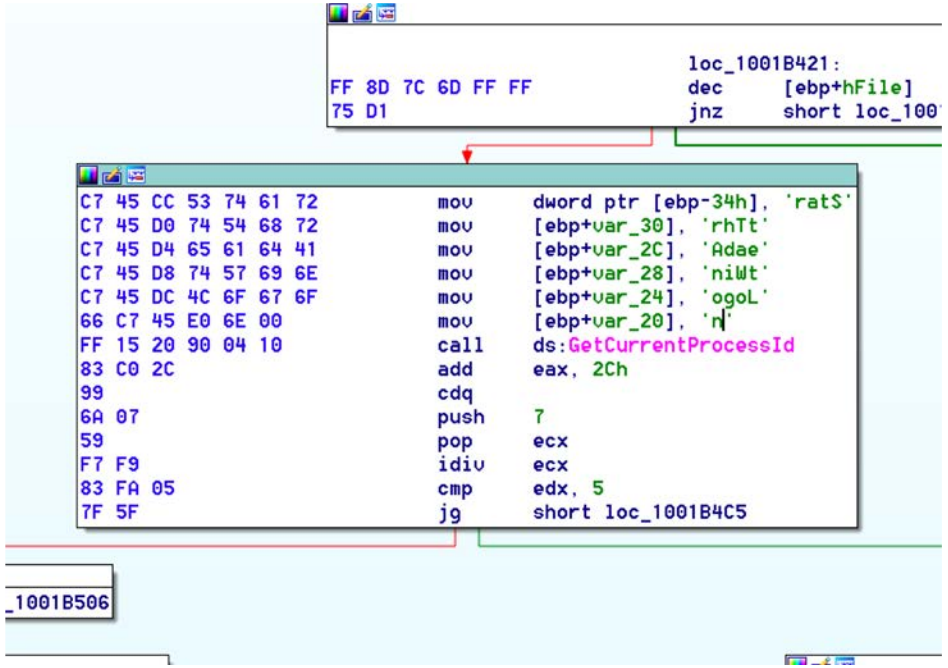
500 | Information | Compatibility fix applied to C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.EXE. Fix information: InjectDll, {3432bc96-d181-4529-b261-1a3964961b6c}, 0x00040206.

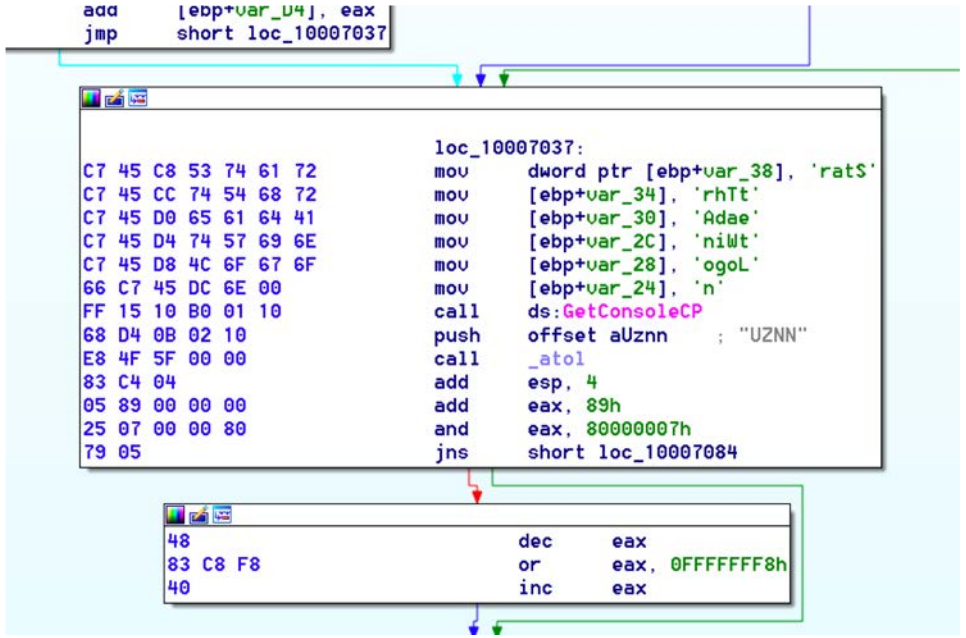
500 | Information | Compatibility fix applied to C:\Windows\SysWOW64\cmd.EXE. Fix information: InjectDll, {3432bc96-d181-4529-b261-1a3964961b6c}, 0x00040206.

Hunting for REDSALT

- Stack strings in all first stage DLLs
- Only offset on stack changes

- C:\Program Files (x86)\Cisco Systems\VPN Client\qtwidgets64.dll
- C:\Program Files (x86)\Hewlett-Packard\HP Hotkey Support\hpbluetooth32.dll
- C:\Program Files\DVD Maker\en-US\WM2CLIP.dll
- C:\Program Files\DVD Maker\en-US\WMM4CLIP.dll
- C:\Program Files (x86)\SnapComms\Client\673\SnapclientSSes.dll
- C:\Program Files (x86)\VERITAS\VxPBX\bin\vxlis_64.dll

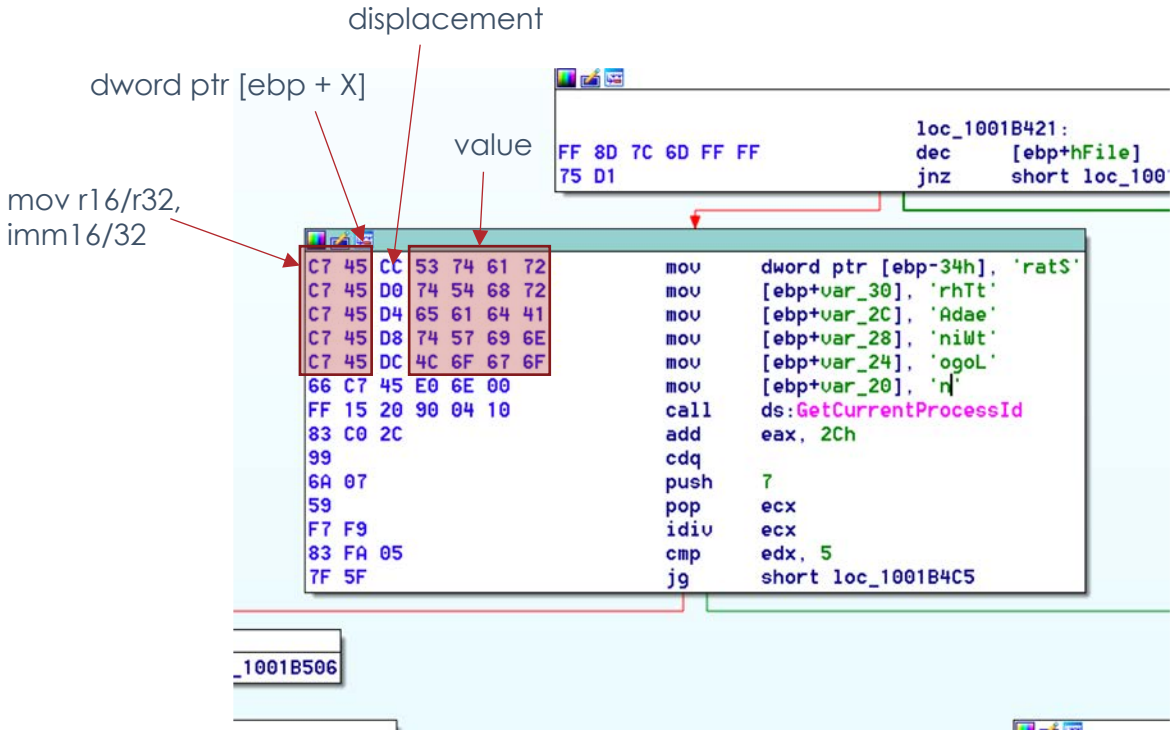


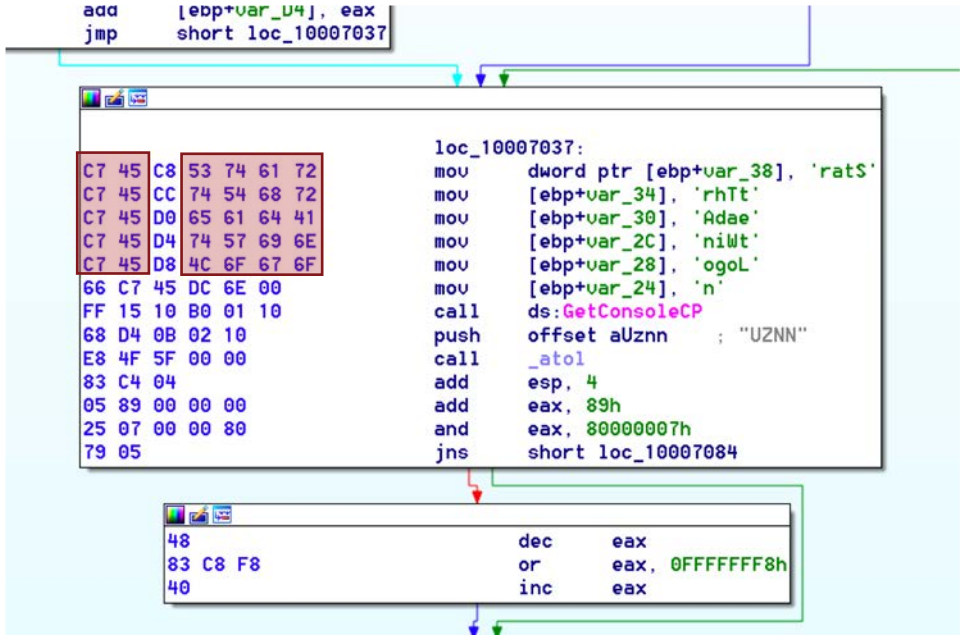


Hunting for REDSALT

- Remove displacement and keep static portion of the instruction

- C:\Program Files (x86)\Cisco Systems\VPN Client\qtwidgets64.dll
- C:\Program Files (x86)\Hewlett-Packard\HP Hotkey Support\hpbluetooth32.dll
- C:\Program Files\DVD Maker\en-US\WM2CLIP.dll
- C:\Program Files\DVD Maker\en-US\WMM4CLIP.dll
- C:\Program Files (x86)\SnapComms\Client\673\SnapclientSSes.dll
- C:\Program Files (x86)\VERITAS\VxPBX\bin\vxlis_64.dll






```
$stack_string = { C7 45 ?? 53 74 61 72 C7 45 ?? 74 54 68 72 C7 45 ?? 65 61 64 41 C7 45 ?? 74 57 69 6E C7 45 ?? 4C 6F 67 6F }
```

REDSALT Encryption

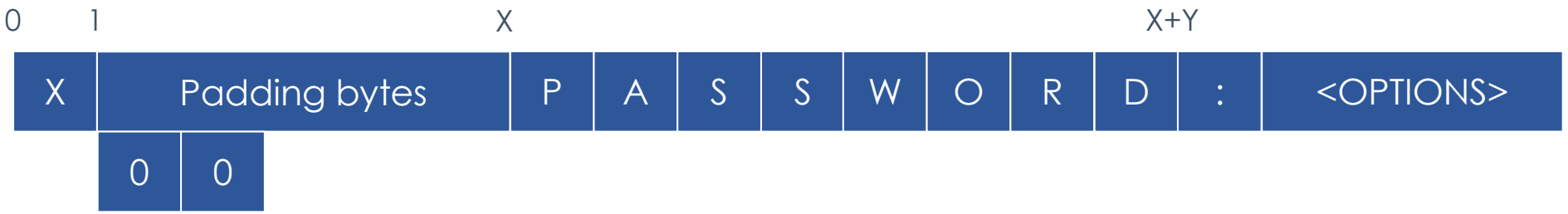
- Blowfish
- Hardcoded keys K1 and K2 for network communications
- C2 key (file and internet)
  - K = md5(xor(K1, K2))
  - K2 is a slight shift from K1
- Door-knocking / Internal initial packets
  - K = md5(K1)
  - Configured password for authentication
- K1 and K2 unchanged for years according to available samples
- Blowfish small block size makes for potential SNORT sigs
  - More chances to find fixed patterns to hit on

Hunting for REDSALT

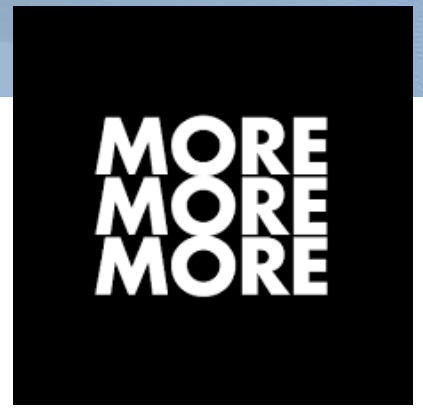
- Network C2 comms
  - Malware looks for content starting with *JOB_NO=*
  - Padding at the beginning of the data up to 8 bytes
  - Compute all possible values for C2 for the first 8 bytes and encrypt with K
    - 1JOB_NO=
    - 20JOB_NO
    - 300JOB_N
    - …

Hunting for REDSALT

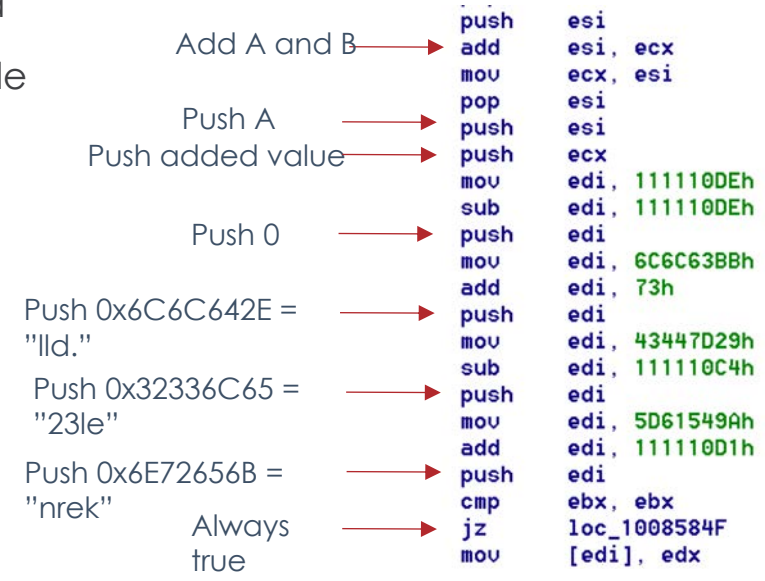
- Network internal comms
  - Malware looks for content starting with campaign password
  - Padding at the beginning of the data up to 8 bytes
    - If known password, compute possible values like for C2
    - If unknown password, lucky hit possible ☺
      - 8 byte padding: 80000000
    - Mistake to put padding zeroes at the beginning

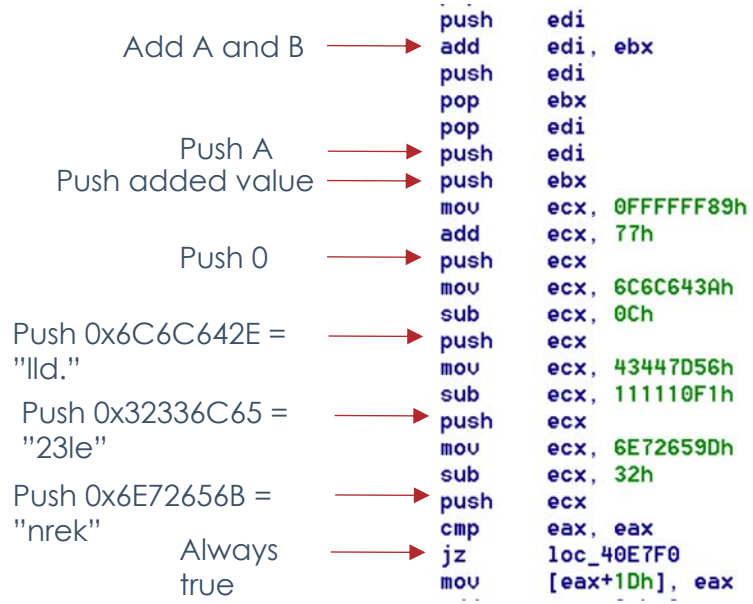


Going further



- REDPEPPER and REDCURRY use the same custom packer
- Sweeping for the packer may yield some results
- Packer basics
  - Exports and imports rewritten on DLL load
  - Same methodology used for each sample
  - Registers, constants, instructions change
- Some slight patterns





Example YARA rule for ESI

```
rule packer_esi_EP_787
{

    strings:

    $esi1 = { BE ?? 10 11 11 81 EE ?? 10 11 11 56 BE ?? ?? ?? ?? 81 (C6|EE) ?? (10|11) 11 11 56 BE }
    $esi2 = { BE ?? 10 11 11 81 EE ?? 10 11 11 56 BE ?? ?? ?? ?? 81 (C6|EE) ?? 00 00 00 56 BE }
    $esi4 = { BE ?? 11 11 11 81 EE ?? 11 11 11 56 BE ?? ?? ?? ?? 81 (C6|EE) ?? (10|11) 11 11 56 BE }
    $esi5 = { BE ?? 11 11 11 81 EE ?? 11 11 11 56 BE ?? ?? ?? ?? 81 (C6|EE) ?? 00 00 00 56 BE }
    $esi7 = { BE ?? EF EE EE 81 C6 ?? 10 11 11 56 BE ?? ?? ?? ?? 81 (C6|EE) ?? (10|11) 11 11 56 BE }
    $esi8 = { BE ?? EF EE EE 81 C6 ?? 10 11 11 56 BE ?? ?? ?? ?? 81 (C6|EE) ?? 00 00 00 56 BE }
    $esi10 = { BE ?? FF FF FF 81 C6 ?? 00 00 00 56 BE ?? ?? ?? ?? 81 (C6|EE) ?? (10|11) 11 11 56 BE }
    $esi11 = { BE ?? FF FF FF 81 C6 ?? 00 00 00 56 BE ?? ?? ?? ?? 81 C6 ?? 00 00 00 56 BE }
    $esi12 = { BE ?? FF FF FF 81 C6 ?? 00 00 00 56 BE ?? ?? ?? ?? 81 EE ?? 00 00 00 56 BE }
    $esi13 = { BE ?? EE EE EE 81 C6 ?? 11 11 11 56 BE ?? ?? ?? ?? 81 (C6|EE) ?? (10|11) 11 11 56 BE }
    $esi14 = { BE ?? EE EE EE 81 C6 ?? 11 11 11 56 BE ?? ?? ?? ?? 81 (C6|EE) ?? 00 00 00 56 BE }

    $jmp = { E8 05 00 00 00 (EB|E9) ?? ?? ?? ?? 8B FF 55 8B EC }

    //$mz = { 4D 5A }

    condition:

    $jmp and any of ($esi*)

}
```

Hunting for the Packer

- Limited resource intensive hunts (nervous client…)
  - Needed to translate YARA into regex
  - Limited to PE files only
  - Limited number of possibilities
- Lots of interesting hits found on Exchange servers and Domain Controllers
  - Older variants of REDSALT, REDCURRY
  - REDMAIL
  - REDSAFFRON
  - pthreadvc3.dll
  - Vxlogagt.exe

Wait… REDSALT, REDCURRY again?

- REDSALT
  - COM and Winlogon Events for persistence
  - No first stage DLL with stack strings
  - Firewall IOC would have caught these

- C:\Program Files\Common Files\System\mslsf\mslsf64.dll
- C:\Program Files\Common Files\System\mslsf\msvtvc.DLL
- C:\Program Files\Veritas\NetBackup\bin\vxlogctl.exe
- C:\Program Files\Veritas\NetBackup\bin\vxlogcfg.exe
- C:\Program Files\Veritas\NetBackup\bin\vxlogcfg.dll

- REDCURRY
  - No persistence
  - No PowerShell event log clearing
  - No base64 encoded files
  - Memory only hunting would have failed too (malware no longer active)

- C:\Program Files\VMware\VMware Tools\Drivers\memctl\vmaudio.cat
- C:\Program Files\Windows Mail\Setup\wabimp.dll

- Overall, good example of how difficult it can be to write a generic IOC

REDMAIL ("GP")

> *"Transport agents let you install custom software that is created by Microsoft, by third-party vendors, or by your organization, on an Exchange server. This software can then process email messages that pass through the transport pipeline."*

- Microsoft Exchange transport agent "msdbagt.dll"
  - .NET file specified in Exchange configuration file agents.config
  - Blended in as "Microsoft Exchange ODB Agent"
- Obfuscated .NET calls a ***packed*** ☺ DLL "msdbeng.dll"
  - GP export – process email
  - PM export – uninstall / delete
- Configuration file "ExchPrf.inf" contains a list of targeted email addresses and a blacklist (spam)
- Compress and encrypt targeted emails to configured directory on disk
  - 128 bit AES keys generated using CryptGenRandom()
  - AES keys encrypted with hardcoded public RSA 1024 bit key
- Execute commands in an email from a configured sender as a batch script
- First sample in the environment dropped on **2010** (*recall first public report identified being MS April 2016*)
- No exfiltration mechanism… what?

- C:\Windows\SysWOW64\UCappi.dll
- C:\Windows\SysWOW64\UCmappi.dll
- C:\Windows\SysWOW64\Ucmmp.dll
- C:\Program Files\Windows Mail\setup\msadco.dll

REDSAFFRON

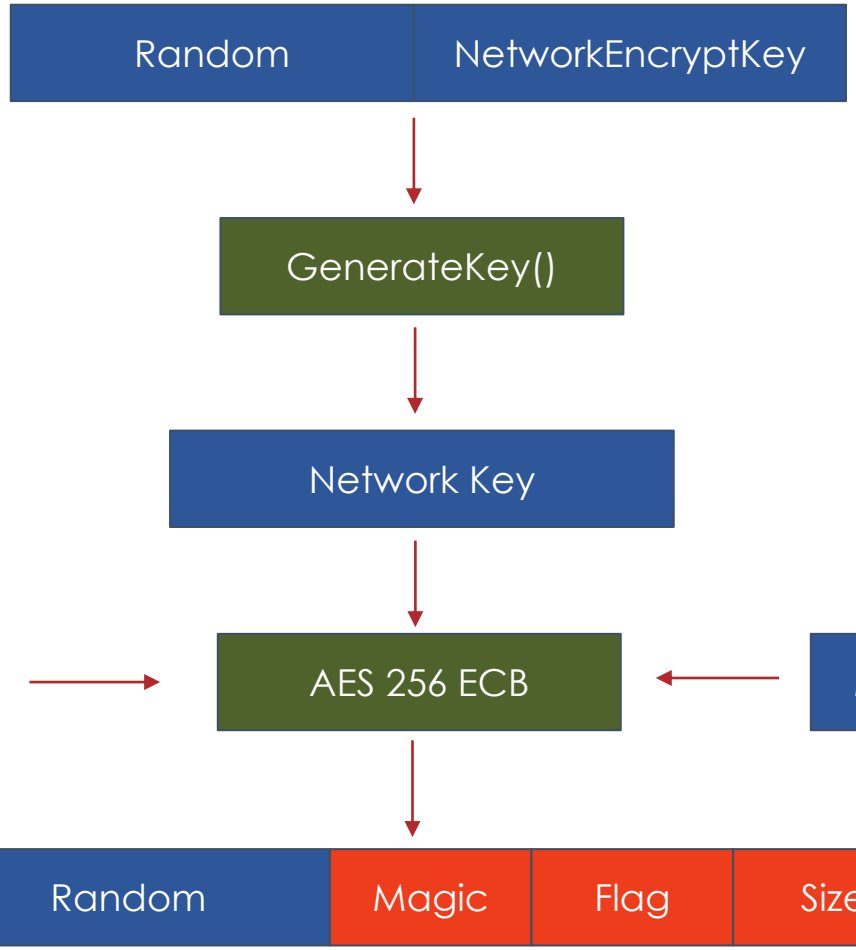
- Found on:
  - Exchange (2008, 2012) – **packed** ☺ but not obfuscated
  - Workstation (2015) – **packed** and obfuscated, loaded like Redcurry / Redpepper
- Main purpose is acting as a malware carrier for data exfiltration and communication between hosts
  - Interoperate up to 10 utilities/malware
  - Listening to raw socket "pings" to initiate coms on another port
- Configuration file encrypted using ICE
- Helper utility – "showcfg", "updatecfg", etc.
- Sample configurations retrieved
  - Start keylogger and unrecovered malware
  - Start keylogger and Redmail, send keylog data and encrypted emails to another workstation running REDSAFFRON

REDSAFFRON configuration

```
04. EradicateDays = -1
05. DaysToKeepFiles = 15
06. OfficeHour = 08:00;17:00
07. MinDiskSpace = 100
08. MaxArchiveSize = 10
09. MaxFolderSize = 100
10. MaxFileTransfer = 500
11. CommandDir = c:\progra~1\common~1\microsoft shared\Indexing Services\Index
12. ResultDir = c:\progra~1\common~1\microsoft shared\Indexing Services\Log
13. StorageDir = c:\progra~1\common~1\microsoft shared\Indexing Services\Store
14. TransitDir = c:\progra~1\common~1\microsoft shared\Indexing Services\Te
15. RunExeAs =
16. MonitoredDir1 = C:\Program Files\Common Files\microsoft shared\Indexing
17. MonitoredDir2 =
18. MonitoredDir3 =
19. MonitoredDir4 =
20. MonitoredDir5 =
<REDACTED>

[SERVER]
28. ProbePort = 3389
29. TransferPort = 443

[CLIENT]
30. PollTime = 1
31. PollRetries = 5
32. PollReboot = 1
33. TargetHostname1 =
34. TargetProbePort1 =
35. TargetTransferPort1 =
36. TargetHostname2 =
```

```
61. RemoveStoreSubDir1 = 0
62. KeepDirStructInArchive1 = 0
63. MultipleArchivePerSession1 = 1
64. ToolName2 =
65. FilePrefixID2 = KB2.1.8
66. ToolStoreDir2 = c:\program files\windows mail\setup | 2.dat
67. ToolUtilDir2 = c:\program files\windows mail\setup
68. ToolUtilFile2 = wabimp.dll
69. PreCmd1-2 = 0-1@rundll32 "c:\program files\windows mail\setup\wabimp.dll",ExLinkFn
70. PreCmd2-2 =
71. PreCmd3-2 =
72. PostCmd1-2 =
73. PostCmd2-2 =
74. PostCmd3-2 =
75. RemoveStoreSubDir2 = 0
76. KeepDirStructInArchive2 = 0
77. MultipleArchivePerSession2 = 1
78. ToolName3 = GP3.2.1
79. FilePrefixID3 = GP3.2.1
```

REDSAFFRON network encryption




```
  do
  {
    ++index;
    seed += *keyBytes++;
    keySize = strlen(inKey);
  }
  while ( index < keySize );
```

```
if ( networkKeySize > 0 )
{
  do
  {
    seed = 0x343FD * seed + 0x269EC3;          // rand
    randIndex = ((seed >> 16) & 0x7FFF) % 15; // get ran
    *(i++ + networkKey) = *(&hexTable + randIndex);// pu
  }
  while ( i < networkKeySize );
```

REDSAFFRON

- Weakness in key generation
  - Example 32 char NetworkEncryptKey + 16 byte rand()
    - Only 6096 different possible keys
    - 4000 possible keys with known rand() bytes
  - Packets can be brute forced without knowing NetworkEncryptKey or rand() bytes
    - Search for DWORD decrypted "magic" (0x1E3F2) at offset 16
    - Search for DWORD decrypted protocol flags at offset 20 (1, 2, 4, 6, 7, 18, 21, 22)
  - Suitable for targeted hunting in an engagement, not general detection (SNORT SO rules, Surricata LUAJIT, …)

```
do
{
  ++index;
  seed += *keyBytes++;
  keySize = strlen(inKey);
}
while ( index < keySize );
```

Conclusion

- Very interesting group with very good opsec
  - Time triggers to delete malware, secure delete, working hours configs…
  - References to our engagement found in keylogs…
  - Op paused when Mandiant arrived
    - Some malware deleted
    - Some malware was put on standby
    - No active malware communications captured
  - **Undetected for 9 years ☹**
- Low profile
- Take the best out of public knowledge and use it
- Not all IOCs are born equal!
- Trying to hide will sometimes reveal you ☺

FireEye

Thank you