Learning to ChaCha with APT41

John Southworth

PwC UK

About me

John Southworth

Principal Threat Intelligence Analyst

- Tracking China-based threat actors (along with other threat actors)
- Malware reverse engineering
- Infrastructure tracking
- ♡ YARA rules

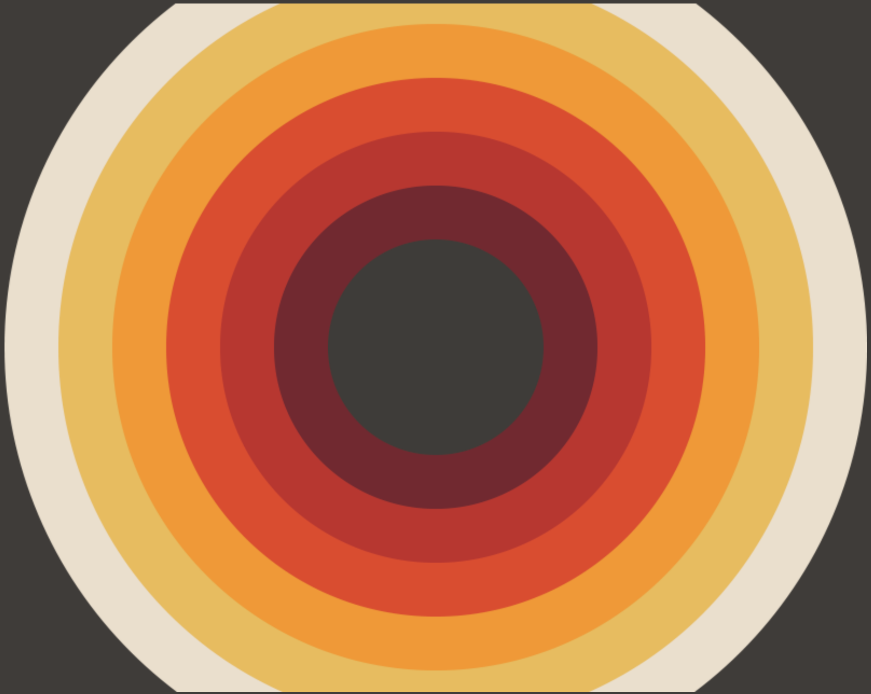
@BitsOfBinary

bitsofbinary

BitsOfBinary

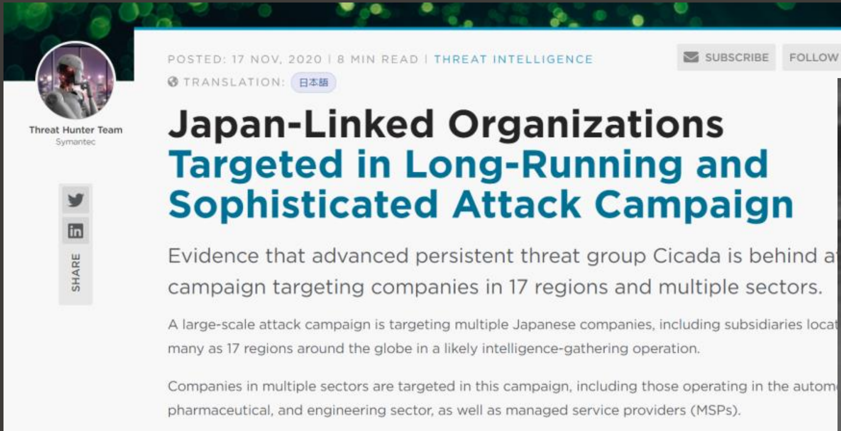
Structure of the talk

- Introduction
  - Red Apollo and MS13-098
  - Red Kelpie
- Malware analysis
  - Motnug + ChaChaLoader
  - Cobalt Strike + SideWalk
- Infrastructure analysis
- Targeting and attribution
- Further research



Indicators + YARA rules + Scripts:
https://github.com/PwCUK-CTO/TheSAS2021-Red-Kelpie

Red Apollo/APT10 recent open source reporting



POSTED: 17 NOV, 2020 | 8 MIN READ | THREAT INTELLIGENCE

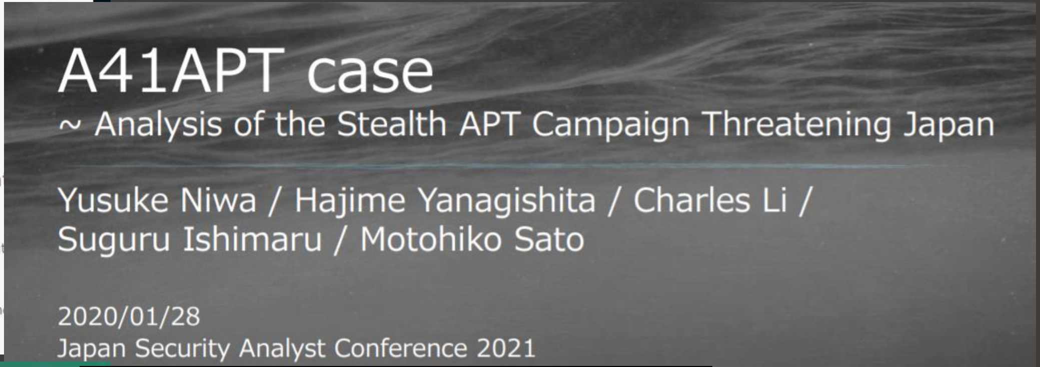
TRANSLATION: 日本語

Japan-Linked Organizations Targeted in Long-Running and Sophisticated Attack Campaign

Evidence that advanced persistent threat group Cicada is behind a campaign targeting companies in 17 regions and multiple sectors.

A large-scale attack campaign is targeting multiple Japanese companies, including subsidiaries locat many as 17 regions around the globe in a likely intelligence-gathering operation.

Companies in multiple sectors are targeted in this campaign, including those operating in the autom pharmaceutical, and engineering sector, as well as managed service providers (MSPs).

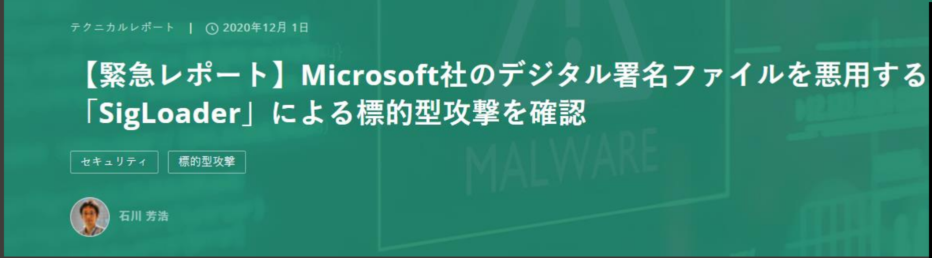


A41APT case

~ Analysis of the Stealth APT Campaign Threatening Japan

Yusuke Niwa / Hajime Yanagishita / Charles Li / Suguru Ishimaru / Motohiko Sato

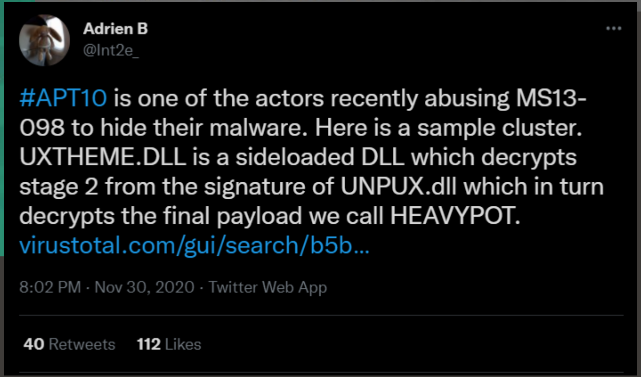
2020/01/28
Japan Security Analyst Conference 2021



テクニカルレポート | 2020年12月 1日

【緊急レポート】Microsoft社のデジタル署名ファイルを悪用する「SigLoader」による標的型攻撃を確認

石川 芳浩



Adrien B
@Int2e_

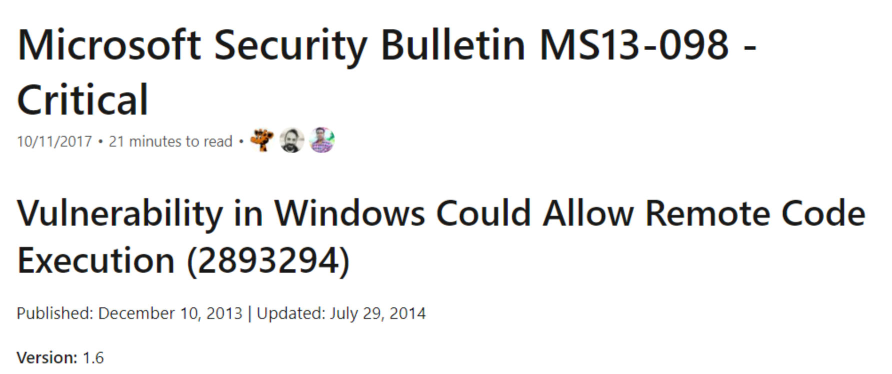
#APT10 is one of the actors recently abusing MS13-098 to hide their malware. Here is a sample cluster. UXTHEME.DLL is a sideloaded DLL which decrypts stage 2 from the signature of UNPUX.dll which in turn decrypts the final payload we call HEAVYPOT. virustotal.com/gui/search/b5b…

8:02 PM · Nov 30, 2020 · Twitter Web App

40 Retweets 112 Likes

Red Apollo loaders

- DESLoader/SigLoader/HEAVYHAND
- Loads a variety of malware families:
  - SodaMaster/DelfsCake/HEAVYPOT
  - P8RAT/GreetCake
  - Cobalt Strike
- Abuses MS13-098 to store encrypted payloads



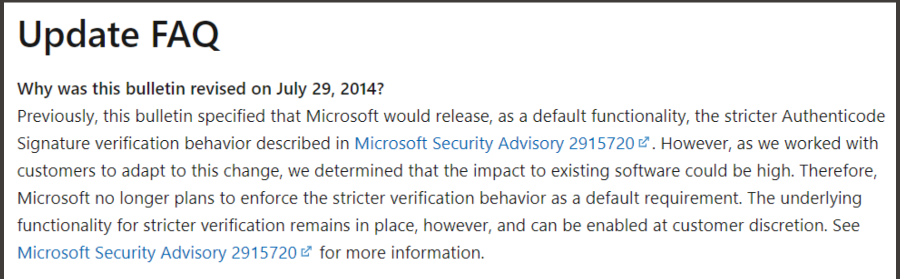
Microsoft Security Bulletin MS13-098 - Critical

10/11/2017 • 21 minutes to read •

Vulnerability in Windows Could Allow Remote Code Execution (2893294)

Published: December 10, 2013 | Updated: July 29, 2014

Version: 1.6



Update FAQ

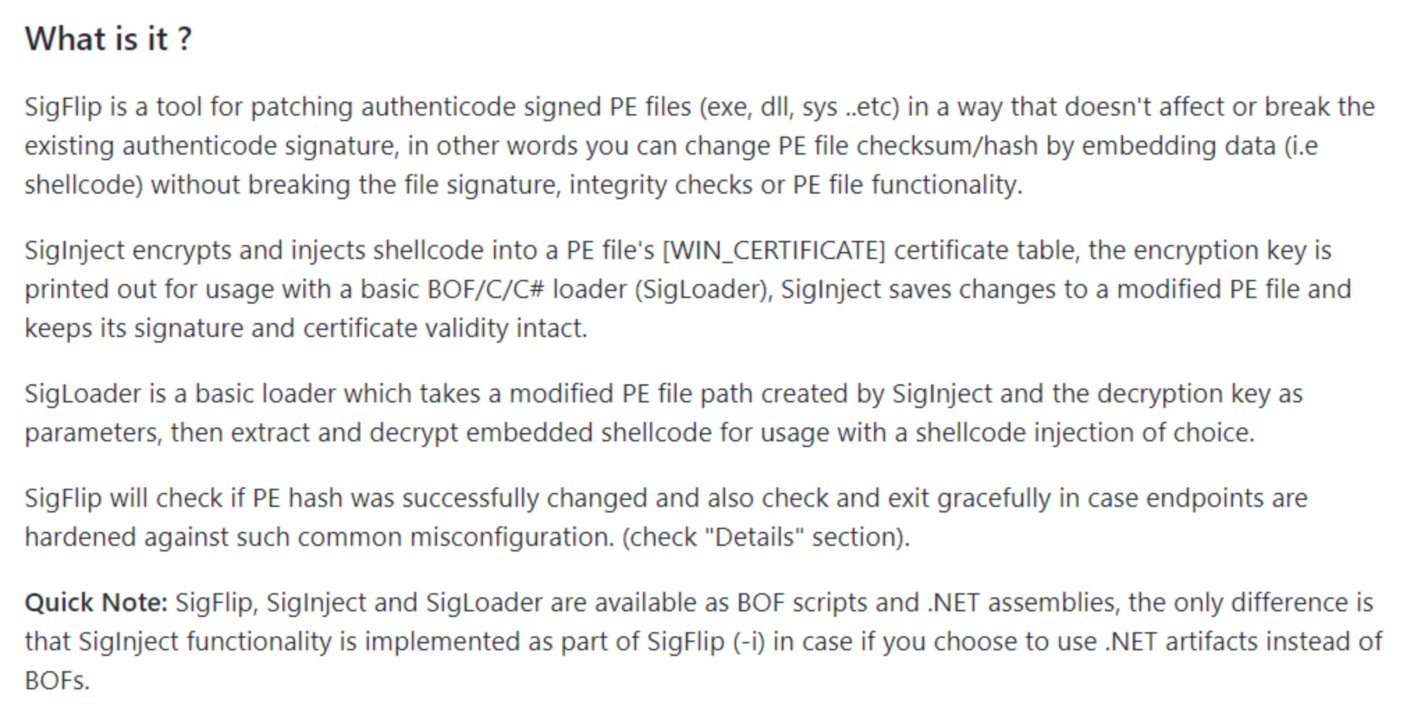
Why was this bulletin revised on July 29, 2014?

Previously, this bulletin specified that Microsoft would release, as a default functionality, the stricter Authenticode Signature verification behavior described in Microsoft Security Advisory 2915720. However, as we worked with customers to adapt to this change, we determined that the impact to existing software could be high. Therefore, Microsoft no longer plans to enforce the stricter verification behavior as a default requirement. The underlying functionality for stricter verification remains in place, however, and can be enabled at customer discretion. See Microsoft Security Advisory 2915720 for more information.

Sample DESLoader hash:

SHA-256	`1f1bcb03b008c4fdd462e7d2b5db5ca321ff6d56bbb22cddd39c82df1f6a038f`
File type	`Win64 DLL`
File size	`190,464 bytes`

Open source - https://github.com/med0x2e/SigFlip



What is it ?

SigFlip is a tool for patching authenticode signed PE files (exe, dll, sys ..etc) in a way that doesn't affect or break the existing authenticode signature, in other words you can change PE file checksum/hash by embedding data (i.e shellcode) without breaking the file signature, integrity checks or PE file functionality.

SigInject encrypts and injects shellcode into a PE file's [WIN_CERTIFICATE] certificate table, the encryption key is printed out for usage with a basic BOF/C/C# loader (SigLoader), SigInject saves changes to a modified PE file and keeps its signature and certificate validity intact.

SigLoader is a basic loader which takes a modified PE file path created by SigInject and the decryption key as parameters, then extract and decrypt embedded shellcode for usage with a shellcode injection of choice.

SigFlip will check if PE hash was successfully changed and also check and exit gracefully in case endpoints are hardened against such common misconfiguration. (check "Details" section).

Quick Note: SigFlip, SigInject and SigLoader are available as BOF scripts and .NET assemblies, the only difference is that SigInject functionality is implemented as part of SigFlip (-i) in case if you choose to use .NET artifacts instead of BOFs.

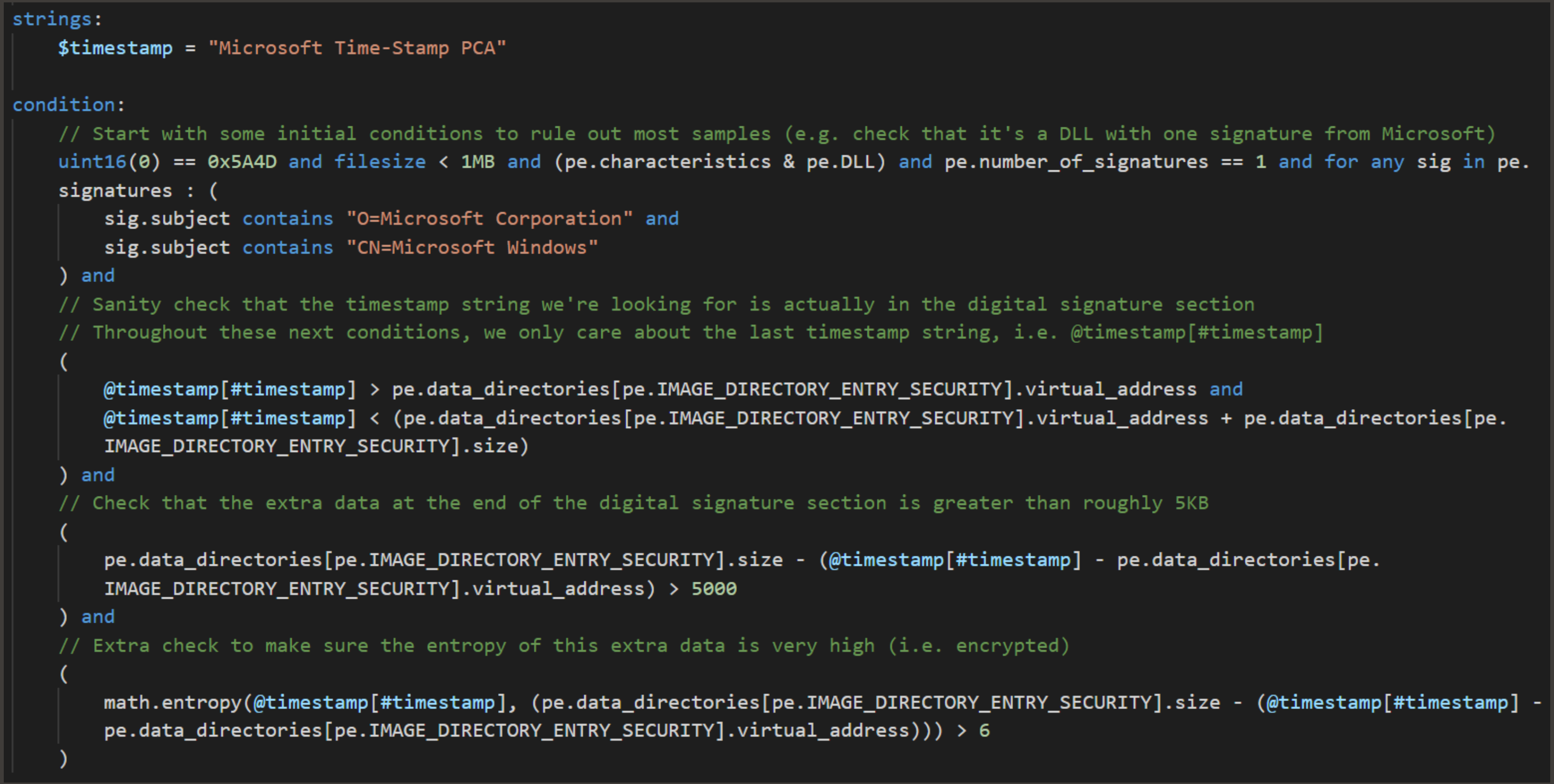
Tracking MS13-098 encoded payloads

- Look for signed binaries with organisation “`Microsoft Corporation`”, and common name “`Microsoft Windows`”
- Find the last occurrence of “`Microsoft Time-Stamp PCA`”, and check it is within “`IMAGE_DIRECTORY_ENTRY_SECURITY`”
- Check that from the end of this timestamp the rest of this section is greater than 5KB (i.e. there is extra data at the end of the digital signature)
- Finally, check that this extra data has high entropy

Alternatively, @Int2e_ has a similar rule:

https://twitter.com/Int2e_/status/1330975808941330432

YARA rule for suspected MS13-098 binaries



```
strings:
    $timestamp = "Microsoft Time-Stamp PCA"

condition:
    // Start with some initial conditions to rule out most samples (e.g. check that it's a DLL with one signature from Microsoft)
    uint16(0) == 0x5A4D and filesize < 1MB and (pe.characteristics & pe.DLL) and pe.number_of_signatures == 1 and for any sig in pe.
    signatures : (
        sig.subject contains "O=Microsoft Corporation" and
        sig.subject contains "CN=Microsoft Windows"
    ) and
    // Sanity check that the timestamp string we're looking for is actually in the digital signature section
    // Throughout these next conditions, we only care about the last timestamp string, i.e. @timestamp[#timestamp]
    (
        @timestamp[#timestamp] > pe.data_directories[pe.IMAGE_DIRECTORY_ENTRY_SECURITY].virtual_address and
        @timestamp[#timestamp] < (pe.data_directories[pe.IMAGE_DIRECTORY_ENTRY_SECURITY].virtual_address + pe.data_directories[pe.
        IMAGE_DIRECTORY_ENTRY_SECURITY].size)
    ) and
    // Check that the extra data at the end of the digital signature section is greater than roughly 5KB
    (
        pe.data_directories[pe.IMAGE_DIRECTORY_ENTRY_SECURITY].size - (@timestamp[#timestamp] - pe.data_directories[pe.
        IMAGE_DIRECTORY_ENTRY_SECURITY].virtual_address) > 5000
    ) and
    // Extra check to make sure the entropy of this extra data is very high (i.e. encrypted)
    (
        math.entropy(@timestamp[#timestamp], (pe.data_directories[pe.IMAGE_DIRECTORY_ENTRY_SECURITY].size - (@timestamp[#timestamp] -
        pe.data_directories[pe.IMAGE_DIRECTORY_ENTRY_SECURITY].virtual_address))) > 6
    )
```

Pivoting to Red Kelpie

- Rule works nicely, picks up a bunch of extra Red Apollo samples
- Can't confirm all of them without the loaders
- Then picks up a MS13-098 sample that is loaded by a different loader than DESLoader…
- Appears related to APT41 during initial analysis, specifically the Motnug loader

Note: timing of MS13-098 samples observed suggests Red Apollo used this technique first

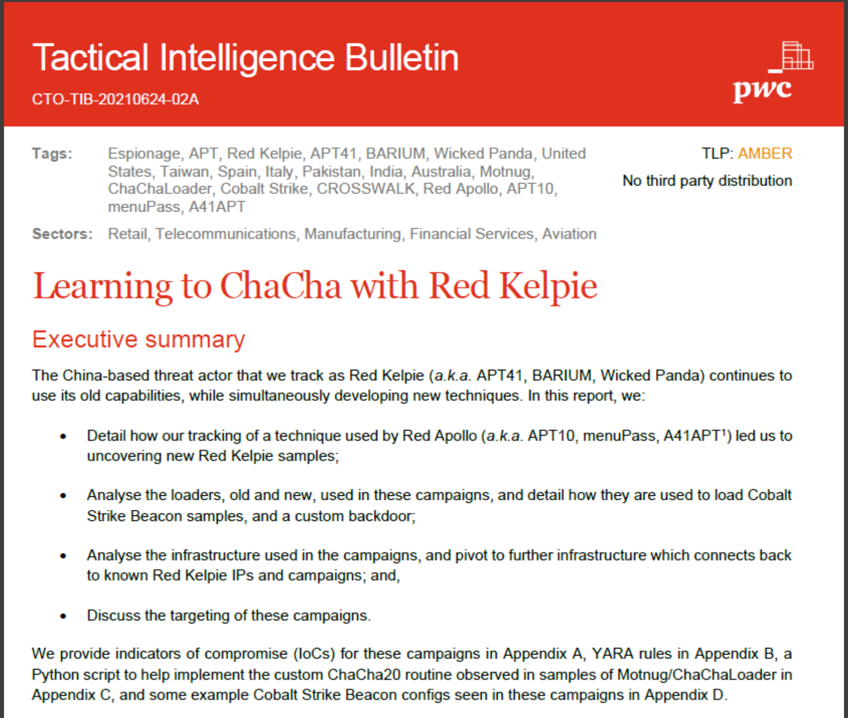
First Red Apollo sample: seen 9th July 2020

First Red Kelpie sample: seen 12th November 2020

I MADE THIS
YOU MADE THIS ?
APT10
MS13-098
APT41
MS13-098
APT41
I MADE THIS
MS13-098
APT41

Threat actor: Red Kelpie

- China-based threat actor
- Aliases*: APT41, BARIUM
- Targets: Telecommunications, technology, manufacturing, financial services, etc.
- Tools: CROSSWALK, ShadowPad, Motnug, and many more



Tactical Intelligence Bulletin

CTO-TIB-20210624-02A

pwc

Tags: Espionage, APT, Red Kelpie, APT41, BARIUM, Wicked Panda, United States, Taiwan, Spain, Italy, Pakistan, India, Australia, Motnug, ChaChaLoader, Cobalt Strike, CROSSWALK, Red Apollo, APT10, menuPass, A41APT

TLP: AMBER

No third party distribution

Sectors: Retail, Telecommunications, Manufacturing, Financial Services, Aviation

Learning to ChaCha with Red Kelpie

Executive summary

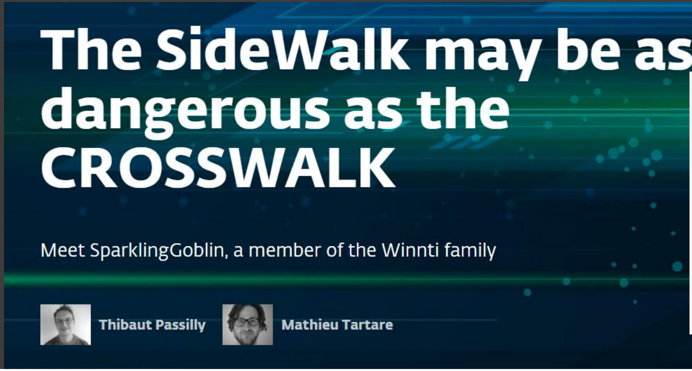
The China-based threat actor that we track as Red Kelpie (*a.k.a.* APT41, BARIUM, Wicked Panda) continues to use its old capabilities, while simultaneously developing new techniques. In this report, we:

- Detail how our tracking of a technique used by Red Apollo (*a.k.a.* APT10, menuPass, A41APT[1]) led us to uncovering new Red Kelpie samples;
- Analyse the loaders, old and new, used in these campaigns, and detail how they are used to load Cobalt Strike Beacon samples, and a custom backdoor;
- Analyse the infrastructure used in the campaigns, and pivot to further infrastructure which connects back to known Red Kelpie IPs and campaigns; and,
- Discuss the targeting of these campaigns.

We provide indicators of compromise (IoCs) for these campaigns in Appendix A, YARA rules in Appendix B, a Python script to help implement the custom ChaCha20 routine observed in samples of Motnug/ChaChaLoader in Appendix C, and some example Cobalt Strike Beacon configs seen in these campaigns in Appendix D.

*Note: these are not 1:1 mappings of the threat actor, this is based on our visibility

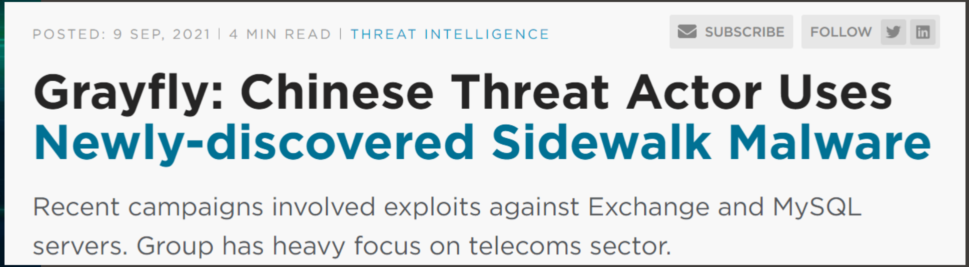
Related APT41 research



The SideWalk may be as dangerous as the CROSSWALK

Meet SparklingGoblin, a member of the Winnti family

Thibaut Passilly Mathieu Tartare

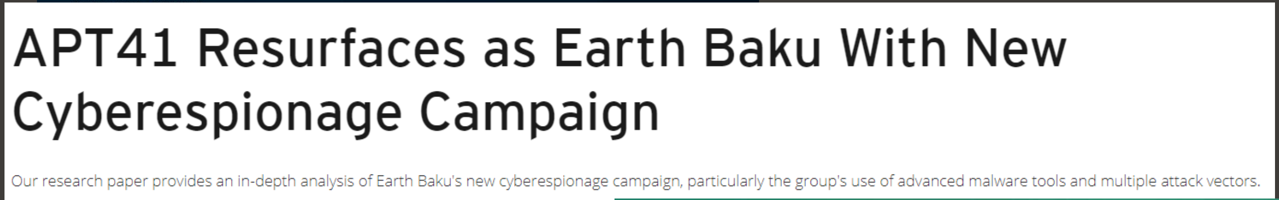


POSTED: 9 SEP, 2021 | 4 MIN READ | THREAT INTELLIGENCE

SUBSCRIBE FOLLOW

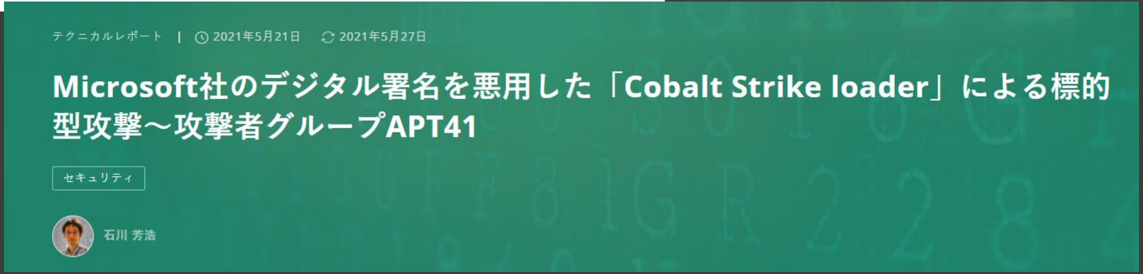
Grayfly: Chinese Threat Actor Uses Newly-discovered Sidewalk Malware

Recent campaigns involved exploits against Exchange and MySQL servers. Group has heavy focus on telecoms sector.



APT41 Resurfaces as Earth Baku With New Cyberespionage Campaign

Our research paper provides an in-depth analysis of Earth Baku's new cyberespionage campaign, particularly the group's use of advanced malware tools and multiple attack vectors.



テクニカルレポート | 2021年5月21日 2021年5月27日

Microsoft社のデジタル署名を悪用した「Cobalt Strike loader」による標的型攻撃～攻撃者グループAPT41

セキュリティ

石川 芳浩

Hypotheses around MS13-098 exploitation

1. Red Apollo shared the MS13-098 technique with Red Kelpie
2. Red Apollo used the technique first, and Red Kelpie found it independently (or copied Red Apollo)
3. Red Apollo and Red Kelpie both got the technique from the same source

Note: US Department of Justice has indicted members of both APT10 and APT41

References:

https://www.justice.gov/opa/pr/two-chinese-hackers-associated-ministry-state-security-charged-global-computer-intrusion

https://www.justice.gov/opa/pr/seven-international-cyber-defendants-including-apt41-actors-charged-connection-computer

Motnug analysis

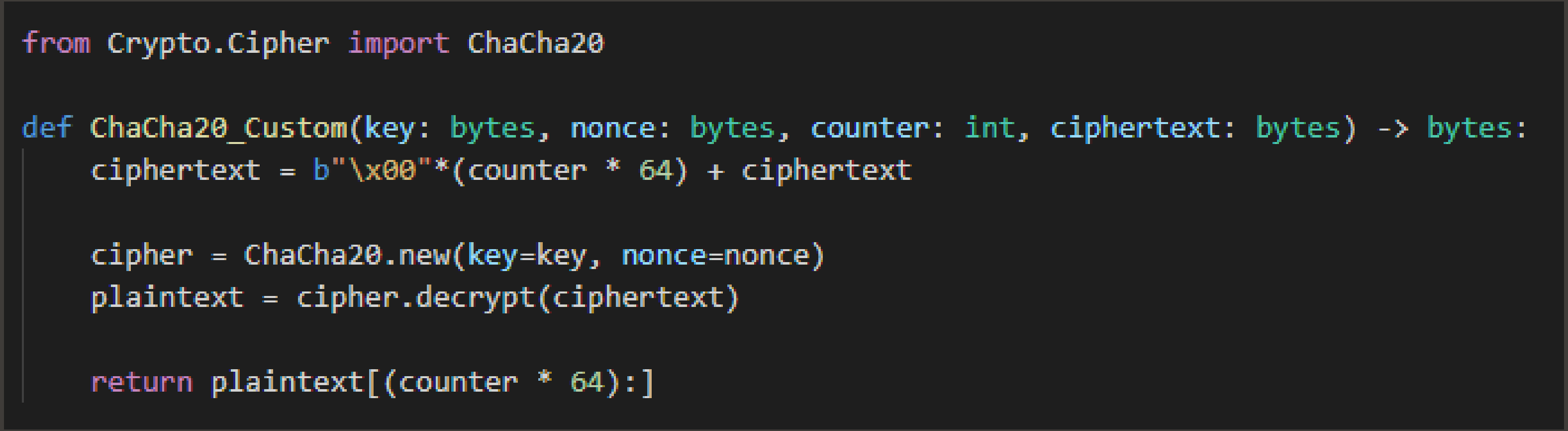
- Loader used by Red Kelpie since at least 2019
- Named Motnug by ESET
- Known to load CROSSWALK
- Can either load a file from itself, or from another file on disk
- Decrypt payloads via AES-128 CBC, or custom ChaCha20 routine

Sample hash:

SHA-256	fad80dc36a59d1cc67f3c4f5deb2650ca7f5abac43858bf38b46f60d6bb4b196
File type	Win64 DLL
File size	101,376 bytes

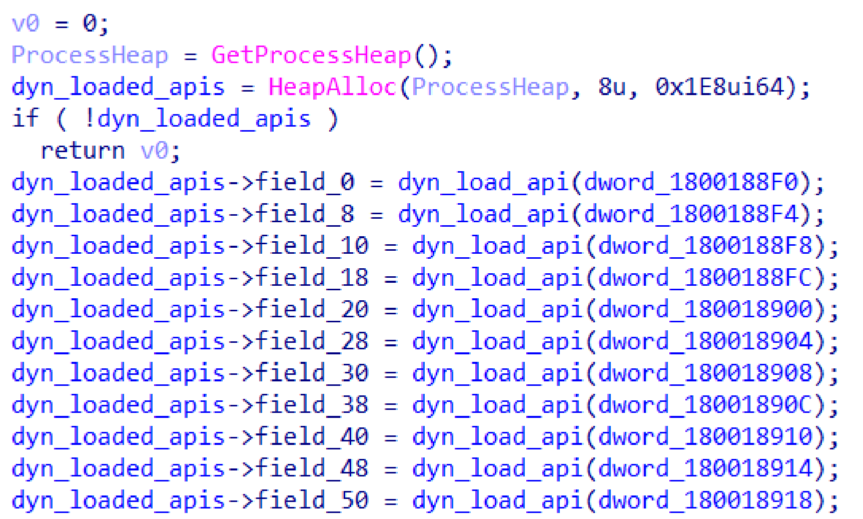
Custom ChaCha20

- ChaCha20 counter manually set to a value higher than 0
- Not all ChaCha20 implementations allow for manual setting of the counter
- Hacky approach: feed *n* 64-byte blocks in, where *n* is the counter you want to set

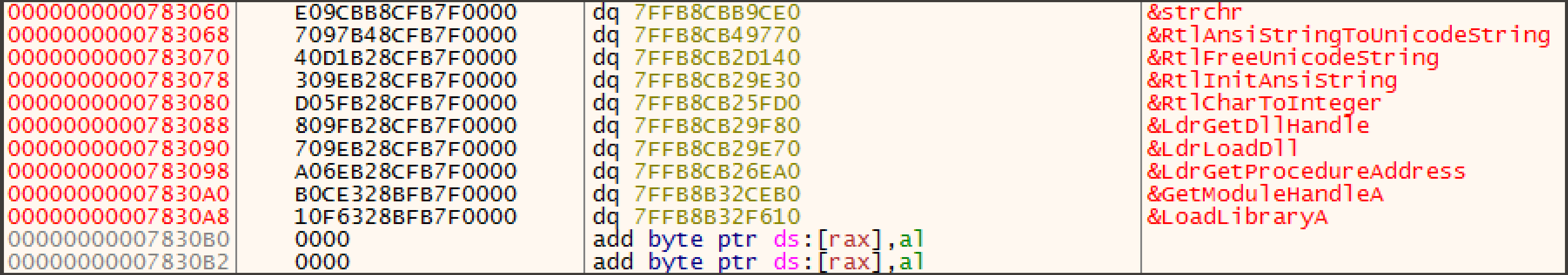


```python
from Crypto.Cipher import ChaCha20

def ChaCha20_Custom(key: bytes, nonce: bytes, counter: int, ciphertext: bytes) -> bytes:
    ciphertext = b"\x00"*(counter * 64) + ciphertext

    cipher = ChaCha20.new(key=key, nonce=nonce)
    plaintext = cipher.decrypt(ciphertext)

    return plaintext[(counter * 64):]
```

Motnug: API structure



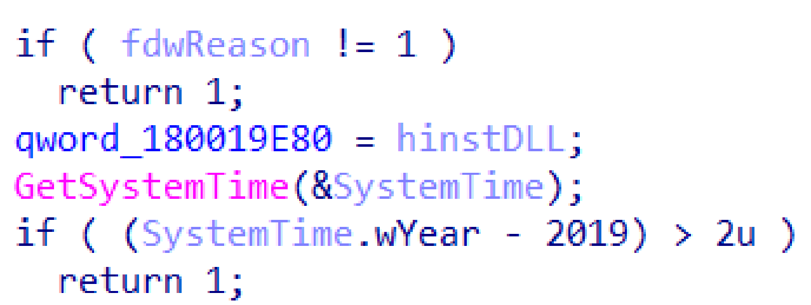
```
v0 = 0;
ProcessHeap = GetProcessHeap();
dyn_loaded_apis = HeapAlloc(ProcessHeap, 8u, 0x1E8ui64);
if ( !dyn_loaded_apis )
  return v0;
dyn_loaded_apis->field_0 = dyn_load_api(dword_1800188F0);
dyn_loaded_apis->field_8 = dyn_load_api(dword_1800188F4);
dyn_loaded_apis->field_10 = dyn_load_api(dword_1800188F8);
dyn_loaded_apis->field_18 = dyn_load_api(dword_1800188FC);
dyn_loaded_apis->field_20 = dyn_load_api(dword_180018900);
dyn_loaded_apis->field_28 = dyn_load_api(dword_180018904);
dyn_loaded_apis->field_30 = dyn_load_api(dword_180018908);
dyn_loaded_apis->field_38 = dyn_load_api(dword_18001890C);
dyn_loaded_apis->field_40 = dyn_load_api(dword_180018910);
dyn_loaded_apis->field_48 = dyn_load_api(dword_180018914);
dyn_loaded_apis->field_50 = dyn_load_api(dword_180018918);
```



```
0000000000783060     E09CBB8CFB7F0000   dq 7FFB8CBB9CE0              &strchr
0000000000783068     7097B48CFB7F0000   dq 7FFB8CB49770              &RtlAnsiStringToUnicodeString
0000000000783070     40D1B28CFB7F0000   dq 7FFB8CB2D140              &RtlFreeUnicodeString
0000000000783078     309EB28CFB7F0000   dq 7FFB8CB29E30              &RtlInitAnsiString
0000000000783080     D05FB28CFB7F0000   dq 7FFB8CB25FD0              &RtlCharToInteger
0000000000783088     809FB28CFB7F0000   dq 7FFB8CB29F80              &LdrGetDllHandle
0000000000783090     709EB28CFB7F0000   dq 7FFB8CB29E70              &LdrLoadDll
0000000000783098     A06EB28CFB7F0000   dq 7FFB8CB26EA0              &LdrGetProcedureAddress
00000000007830A0     B0CE328BFB7F0000   dq 7FFB8B32CEB0              &GetModuleHandleA
00000000007830A8     10F6328BFB7F0000   dq 7FFB8B32F610              &LoadLibraryA
00000000007830B0     0000               add byte ptr ds:[rax],al
00000000007830B2     0000               add byte ptr ds:[rax],al
```

Motnug: time-based guardrail

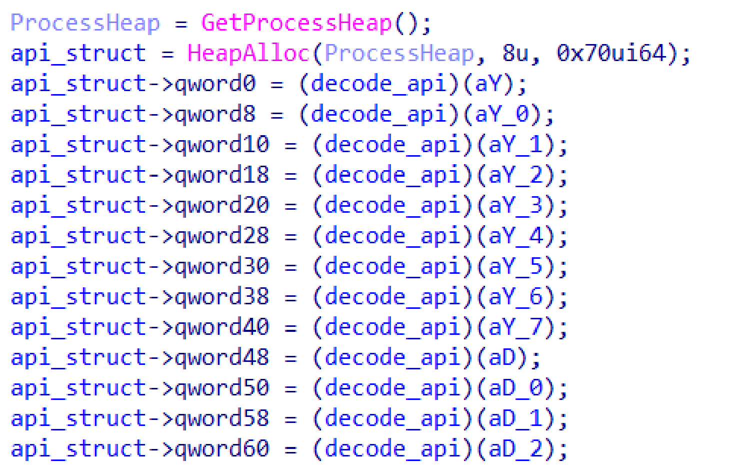
- Some variants contain a guardrail to stop Motnug executing if it fails
- If the year is greater than 2021, or less than 2019, the loader will not execute



```
if ( fdwReason != 1 )
  return 1;
qword_180019E80 = hinstDLL;
GetSystemTime(&SystemTime);
if ( (SystemTime.wYear - 2019) > 2u )
  return 1;
```

Introducing ChaChaLoader

- Called StealthVector by Trend Micro
- Abuses MS13-098 (how I found it initially)
- Updated version of Motnug?
  - Same custom ChaCha20 routine
  - Very similar custom API structure
  - All compilation timestamps after Motnug
- Extra features in the loader (more anti-analysis)
- More customisable configuration

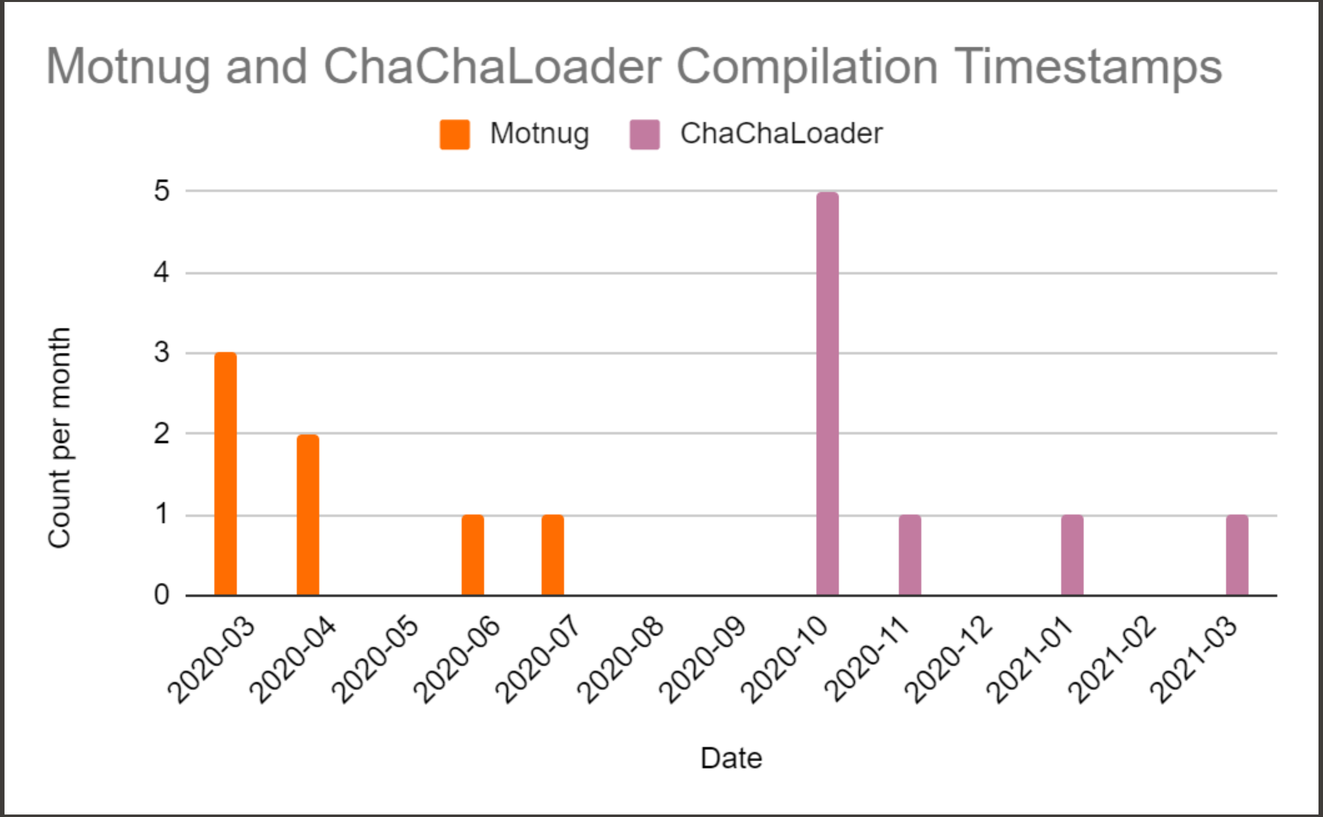


```
ProcessHeap = GetProcessHeap();
api_struct = HeapAlloc(ProcessHeap, 8u, 0x70ui64);
api_struct->qword0 = (decode_api)(aY);
api_struct->qword8 = (decode_api)(aY_0);
api_struct->qword10 = (decode_api)(aY_1);
api_struct->qword18 = (decode_api)(aY_2);
api_struct->qword20 = (decode_api)(aY_3);
api_struct->qword28 = (decode_api)(aY_4);
api_struct->qword30 = (decode_api)(aY_5);
api_struct->qword38 = (decode_api)(aY_6);
api_struct->qword40 = (decode_api)(aY_7);
api_struct->qword48 = (decode_api)(aD);
api_struct->qword50 = (decode_api)(aD_0);
api_struct->qword58 = (decode_api)(aD_1);
api_struct->qword60 = (decode_api)(aD_2);
```

Sample hash:

SHA-256	8da88951322fa7f464c13cb4a173d0c178f5e34a57957c9117b393133dd19925
File type	Win64 DLL
File size	93,696 bytes

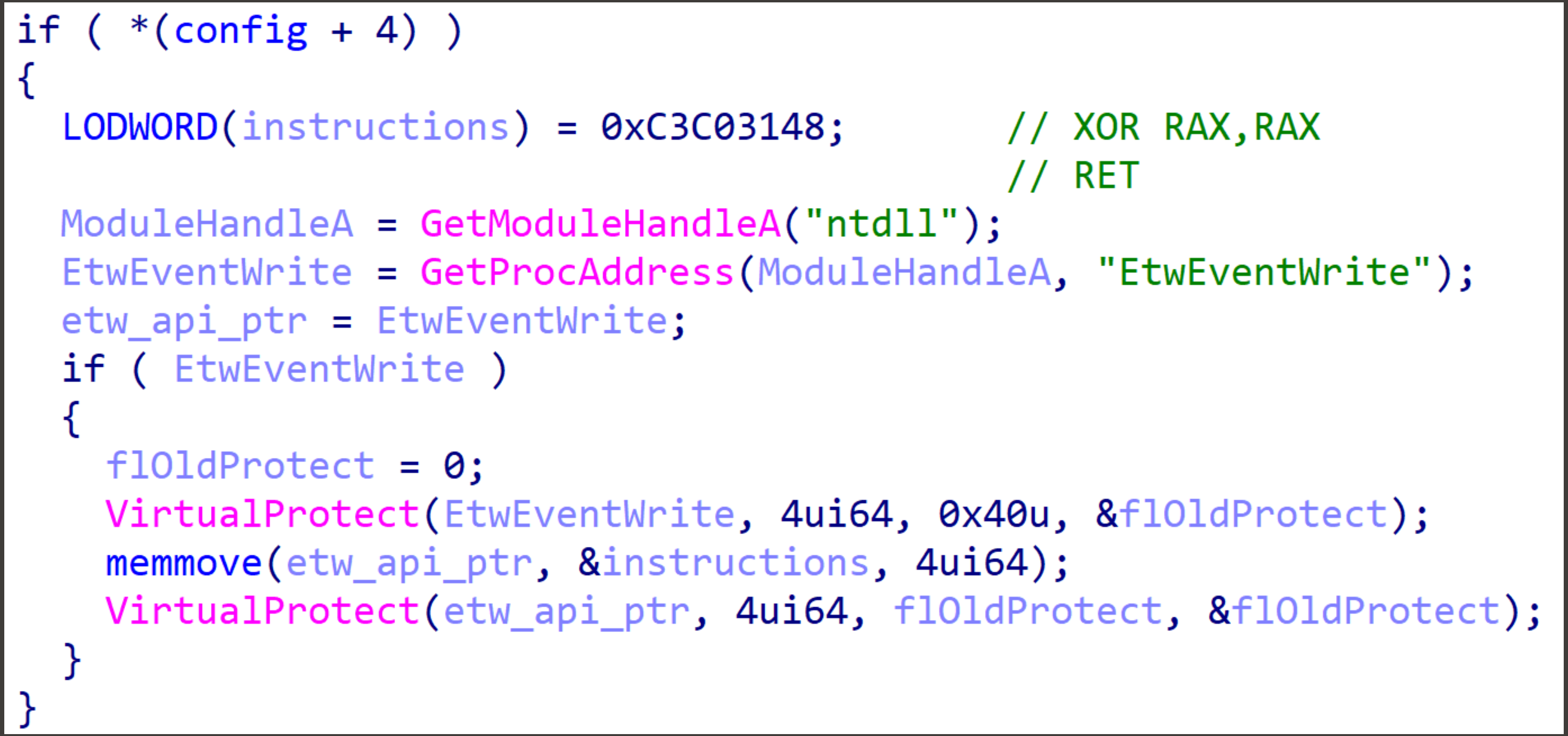
Swapping from Motnug to ChaChaLoader?



ChaChaLoader configuration - capability

Flag offset	Description
`0x4`	Exit if the username of the infected machine is not “`system`”
`0x8`	Attempt to create a mutex, and exit if it already exists
`0xC`	Delete the EXE which ChaChaLoader has been loaded into
`0x10`	Overwrite the API `EtwEventWrite` to always return 0 (i.e. attempt to block ETW logging)
`0x14`	Load the payload from within the loader itself (otherwise load it from a hardcoded filename provided in the config at offset `0x18`)

ETW event blocking



```
if ( *(config + 4) )
{
  LODWORD(instructions) = 0xC3C03148;       // XOR RAX,RAX
                                            // RET
  ModuleHandleA = GetModuleHandleA("ntdll");
  EtwEventWrite = GetProcAddress(ModuleHandleA, "EtwEventWrite");
  etw_api_ptr = EtwEventWrite;
  if ( EtwEventWrite )
  {
    flOldProtect = 0;
    VirtualProtect(EtwEventWrite, 4ui64, 0x40u, &flOldProtect);
    memmove(etw_api_ptr, &instructions, 4ui64);
    VirtualProtect(etw_api_ptr, 4ui64, flOldProtect, &flOldProtect);
  }
}
```

ChaChaLoader configuration - file to load

Config offset	Description
`0x18`	(Optional) the name of the file on disk to load the encoded payload from
`0x20`	The size of the encoded payload
`0x24`	The offset into the loader itself/the file on disk which it needs to load the encoded payload from
`0x28`	The CRC32 of the decoded payload
`0x2C`	The ChaCha20 cryptographic nonce used to decode the payload

And it loads.... Cobalt Strike! (for the most part)

Cobalt Strike URLs

- `hxxp://ns1[.]mssetting[.]com:53/v3/config/`
- `hxxp://www[.]mircoupdate[.]https443[.]net:443/jquery-3.3.1.min.js`
- `hxxp://ns1[.]extrsports[.]ru:53/topstories`
- `hxxp://www[.]corpsolution[.]net:443/massaction`
- `hxxp://ns[.]cloud01[.]tk:53/users/sign_in`
- `hxxp://ns[.]cloud20[.]tk:53/users/sign_in`

Mixture of masquerading as Microsoft, cloud infrastructure, and sports themes

No direct connections back to known Red Kelpie infrastructure

Filenames of ChaChaLoader payloads

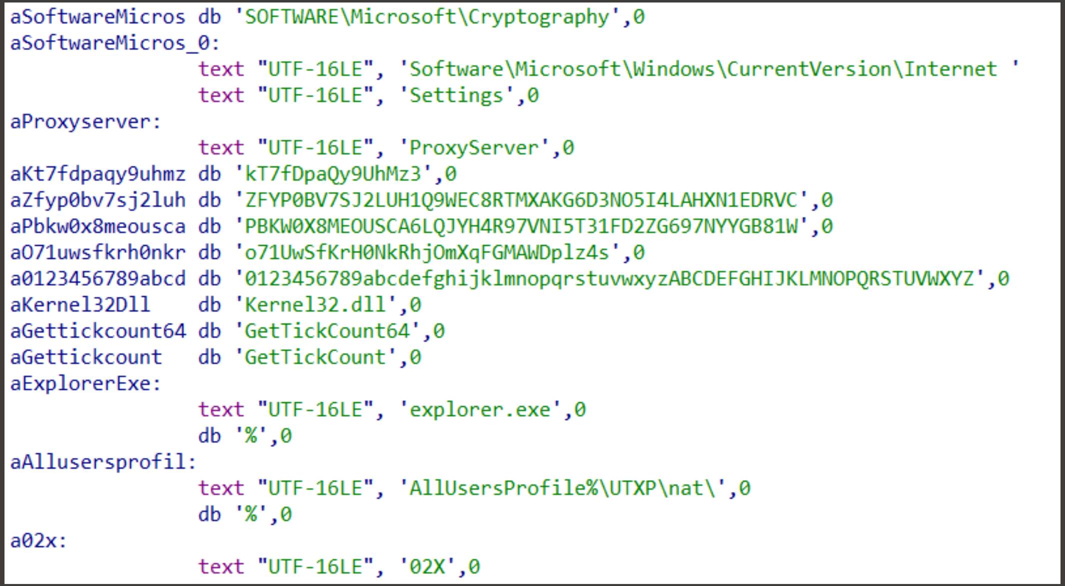
Filename	File type
`wlbsctrl.ax`	DLL
`KBDTAM131.dll`	DLL
`msiltcfg.tlb`	DLL
`google.temp`	DLL
`COMSysUpdate.ocx`	DAT
`normnfw.nls`	DAT

Filename	File type
`systeminfos.tmp`	N/A
`aadauthhelper.dll`	N/A
`dec1.dll`	N/A
`NTUSERS.DAT`	N/A
`spmsetc.sys`	N/A

Note: N/A file type for those I could not recover

SideWalk/ScrambleCross

- Also loaded by ChaChaLoader
- Modular backdoor
- Similar to CROSSWALK - shoutout to my colleague Adam (@malworms) for making this connection
- Shared ChaCha20 routine with ChaChaLoader



```
aSoftwareMicros db 'SOFTWARE\Microsoft\Cryptography',0
aSoftwareMicros_0:
                text "UTF-16LE", 'Software\Microsoft\Windows\CurrentVersion\Internet '
                text "UTF-16LE", 'Settings',0
aProxyserver:
                text "UTF-16LE", 'ProxyServer',0
aKt7fdpaqy9uhmz db 'kT7fDpaQy9UhMz3',0
aZfyp0bv7sj2luh db 'ZFYP0BV7SJ2LUH1Q9WEC8RTMXAKG6D3NO5I4LAHXN1EDRVC',0
aPbkw0x8meousca db 'PBKW0X8MEOUSCA6LQJYH4R97VNI5T31FD2ZG697NYYGB81W',0
aO71uwsfkrh0nkr db 'o71UwSfKrH0NkRhjOmXqFGMAWDplz4s',0
a0123456789abcd db '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ',0
aKernel32Dll    db 'Kernel32.dll',0
aGettickcount64 db 'GetTickCount64',0
aGettickcount   db 'GetTickCount',0
aExplorerExe:
                text "UTF-16LE", 'explorer.exe',0
                db '%',0
aAllusersprofil:
                text "UTF-16LE", 'AllUsersProfile%\UTXP\nat\',0
                db '%',0
a02x:
                text "UTF-16LE", '02X',0
```

C2 request pattern: `hxxps://194.156.98[.]89/UcocFuvS64a3Cto4/pGugspPEo0jsklzO`

Look for '`\/[A-Za-z0-9]{16}\/[A-Za-z0-9]{16}`' patterns in HTTP(S) requests

How is ChaChaLoader dropped?

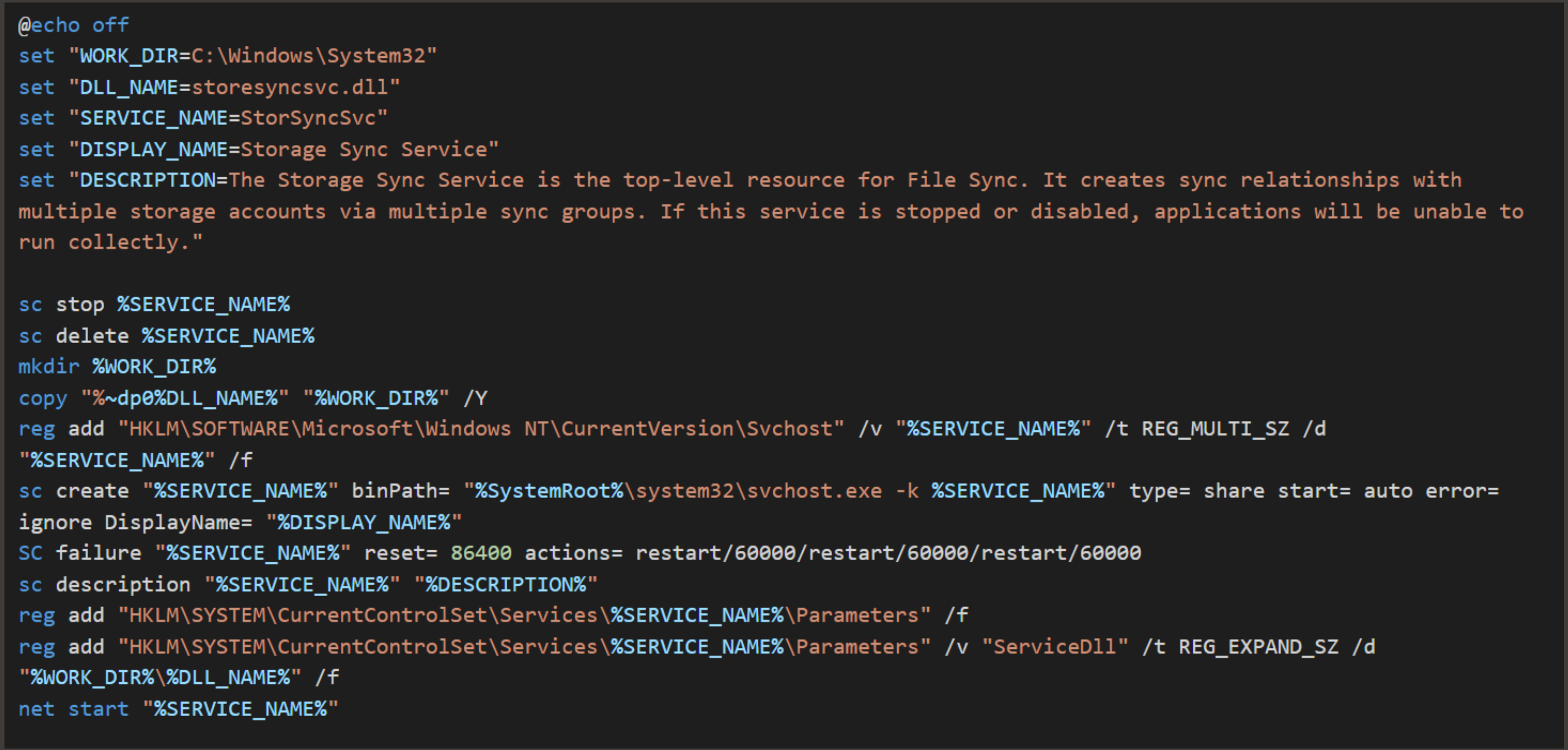
- Batch scripts observed loading ChaChaLoader
- Observed sample being dropped by self-extracting archive
- Open source research discussing the same batch scripts being dropped by exploits for vulnerabilities in Citrix, Cisco routers, etc.*

SFX archive:

SHA-256	2738449fd0d0a68dfb412646ca52b59c293f52a9af00acf3db85077d71534b66
File type	Win32 EXE
File size	542,586 bytes

Reference: https://www.fireeye.com/blog/threat-research/2020/03/apt41-initiates-global-intrusion-campaign-using-multiple-exploits.html

Red Kelpie Batch script



```
@echo off
set "WORK_DIR=C:\Windows\System32"
set "DLL_NAME=storesyncsvc.dll"
set "SERVICE_NAME=StorSyncSvc"
set "DISPLAY_NAME=Storage Sync Service"
set "DESCRIPTION=The Storage Sync Service is the top-level resource for File Sync. It creates sync relationships with
multiple storage accounts via multiple sync groups. If this service is stopped or disabled, applications will be unable to
run collectly."

sc stop %SERVICE_NAME%
sc delete %SERVICE_NAME%
mkdir %WORK_DIR%
copy "%~dp0%DLL_NAME%" "%WORK_DIR%" /Y
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost" /v "%SERVICE_NAME%" /t REG_MULTI_SZ /d
"%SERVICE_NAME%" /f
sc create "%SERVICE_NAME%" binPath= "%SystemRoot%\system32\svchost.exe -k %SERVICE_NAME%" type= share start= auto error=
ignore DisplayName= "%DISPLAY_NAME%"
SC failure "%SERVICE_NAME%" reset= 86400 actions= restart/60000/restart/60000/restart/60000
sc description "%SERVICE_NAME%" "%DESCRIPTION%"
reg add "HKLM\SYSTEM\CurrentControlSet\Services\%SERVICE_NAME%\Parameters" /f
reg add "HKLM\SYSTEM\CurrentControlSet\Services\%SERVICE_NAME%\Parameters" /v "ServiceDll" /t REG_EXPAND_SZ /d
"%WORK_DIR%\%DLL_NAME%" /f
net start "%SERVICE_NAME%"
```

Infrastructure analysis

- Example extra infrastructure: `hxxps://work[.]getdns[.]tk/users/sign_in`
- `hxxp://ns1[.]freeemails[.]shop:53/fwlink`
- `hxxp://letwiki[.]com:443/massaction`

- Cobalt Strike malleable C2 profiles:
  - '`.tk`' and '`/users/sign_in`' (avoiding legit GitLab infrastructure)
  - '`http://ns1.`' on port 53
  - '`:443/massaction`'
- SideWalk C2 had a self signed certificate associated with APT41 (connects to CROSSWALK infrastructure, and APT41 infrastructure from FBI FLASH report*):

SHA-1	26f90f25a075d5658036d4ae493dcc77c2abc173
Serial	11095508213565976896
Common name	AS.website
Valid from	2019-10-08
Valid to	2029-10-05

*Reference: https://www.waterisac.org/system/files/articles/FLASH-AC-000133-TT-APT41.pdf

Targeting

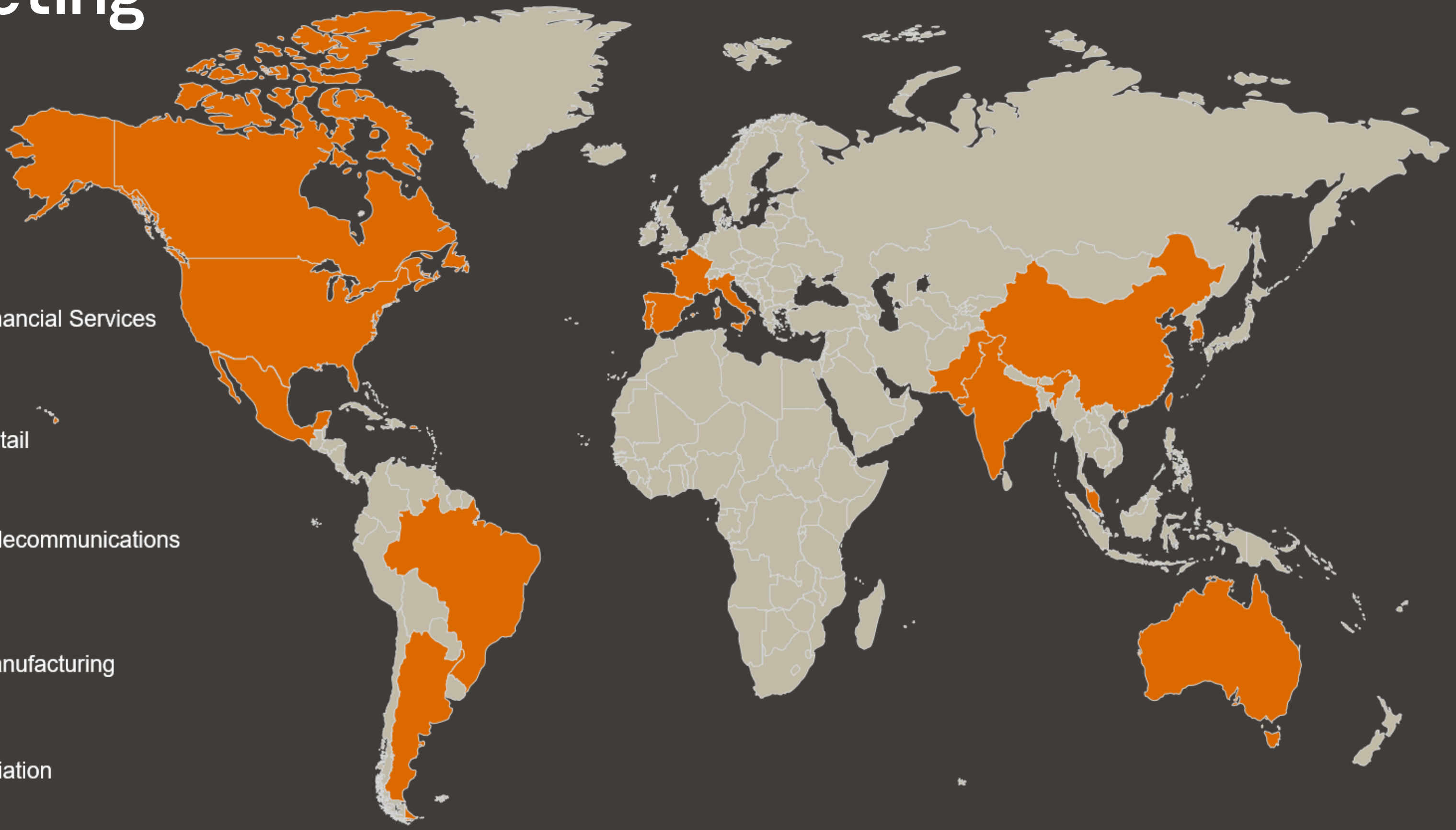
- Retail - Australasia, Europe
- Manufacturer - North America, Asia
- Financial services - Europe
- Telecommunications - Middle East
- Aviation - Asia*

- Further uploads to VirusTotal from other countries

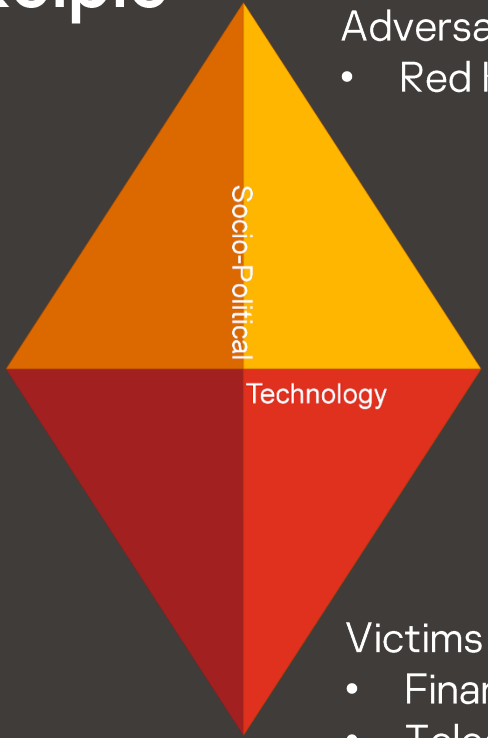
*Reference: https://blog.group-ib.com/colunmtk_apt41

Targeting

- Financial Services
- Retail
- Telecommunications
- Manufacturing
- Aviation



Attribution: Red Kelpie



Adversary

- Red Kelpie

Capability

- Motnug
- CROSSWALK
- ChaChaLoader
- SideWalk/ScrambleCross
- Cobalt Strike

Infrastructure

- Shared SSL cert
- Links to CROSSWALK infrastructure

Victims

- Financial services
- Telecommunications
- Manufacturing
- Retail
- Aviation

Further research – LNKs and BATs

- LNKs mentioned in Trend Micro research
- Unique icon locations:
  - `.\1.pdf`
  - `.\1.doc`
- One sample has same infrastructure as ChaChaLoader loaded sample (`119.45.238[.]189`)
- Cluster of “testing” LNKs uploaded from China

- Batch script dropped by CVE-2021-26084 (Confluence vulnerability)
- Loads Cobalt Strike downloader

Red Apollo references

Cicada:

https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/cicada-apt10-japan-espionage

A41APT:

https://jsac.jpcert.or.jp/archive/2021/pdf/JSAC2021_202_niwa-yanagishita_en.pdf

https://securelist.com/apt10-sophisticated-multi-layered-loader-ecipekac-discovered-in-a41apt-campaign/101519/

APT10:

https://www.lac.co.jp/lacwatch/report/20201201_002363.html

https://twitter.com/Int2e_/status/1333501729359466502

Red Kelpie references

SparklingGoblin:

https://www.welivesecurity.com/2021/08/24/sidewalk-may-be-as-dangerous-as-crosswalk/

Earth Baku:

https://www.trendmicro.com/en_us/research/21/h/apt41-resurfaces-as-earth-baku-with-new-cyberespionage-campaign.html

Grayfly:

https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/grayfly-china-sidewalk-malware

APT41:

https://www.lac.co.jp/lacwatch/report/20210521_002618.html

Thank you!

Indicators + YARA rules + Scripts:

https://github.com/PwCUK-CTO/SASCon2021-Red-Kelpie

@BitsOfBinary

#SASatHome | #TheSAS2021

Let’s Talk?

John Southworth

PwC UK