More evil: A deep look at Evilnum and its toolset

welivesecurity.com/2020/07/09/more-evil-deep-look-evilnum-toolset

Matías Porolli July 9, 2020

ESET has analyzed the operations of Evilnum, the APT group behind the Evilnum malware previously seen in attacks against financial technology companies. While said malware has been seen in the wild since at least 2018 and documented previously, little has been published about the group behind it and how it operates.

In this article we connect the dots and disclose a detailed picture of Evilnum's activities. The group's targets remain fintech companies, but its toolset and infrastructure have evolved and now consist of a mix of custom, homemade malware combined with tools purchased from Golden Chickens, a Malware-as-a-Service (MaaS) provider whose infamous customers include FIN6 and Cobalt Group.

Targets

According to ESET's telemetry, the targets are financial technology companies – for example, companies that offer platforms and tools for online trading. Although most of the targets are located in EU countries and the UK, we have also seen attacks in countries such as Australia and Canada. Typically, the targeted companies have offices in several locations, which probably explains the geographical diversity of the attacks.

The main goal of the Evilnum group is to spy on its targets and obtain financial information from both the targeted companies and their customers. Some examples of the information this group steals include:

- Spreadsheets and documents with customer lists, investments and trading operations
- Internal presentations
- Software licenses and credentials for trading software/platforms
- Cookies and session information from browsers
- Email credentials
- Customer credit card information and proof of address/identity documents

According to what we have seen during our investigation, the group has also gained access to IT-related information such as VPN configurations.

Overview of the attack

Targets are approached with spearphishing emails that contain a link to a ZIP file hosted on Google Drive. That archive contains several LNK (aka shortcut) files that extract and execute a malicious JavaScript component, while displaying a decoy document. These

shortcut files have “double extensions” to try to trick the user into opening them, thinking they are benign documents or pictures (in Windows, file extensions for known file types are hidden by default). The contents of one of the ZIP files are shown in Figure 1.

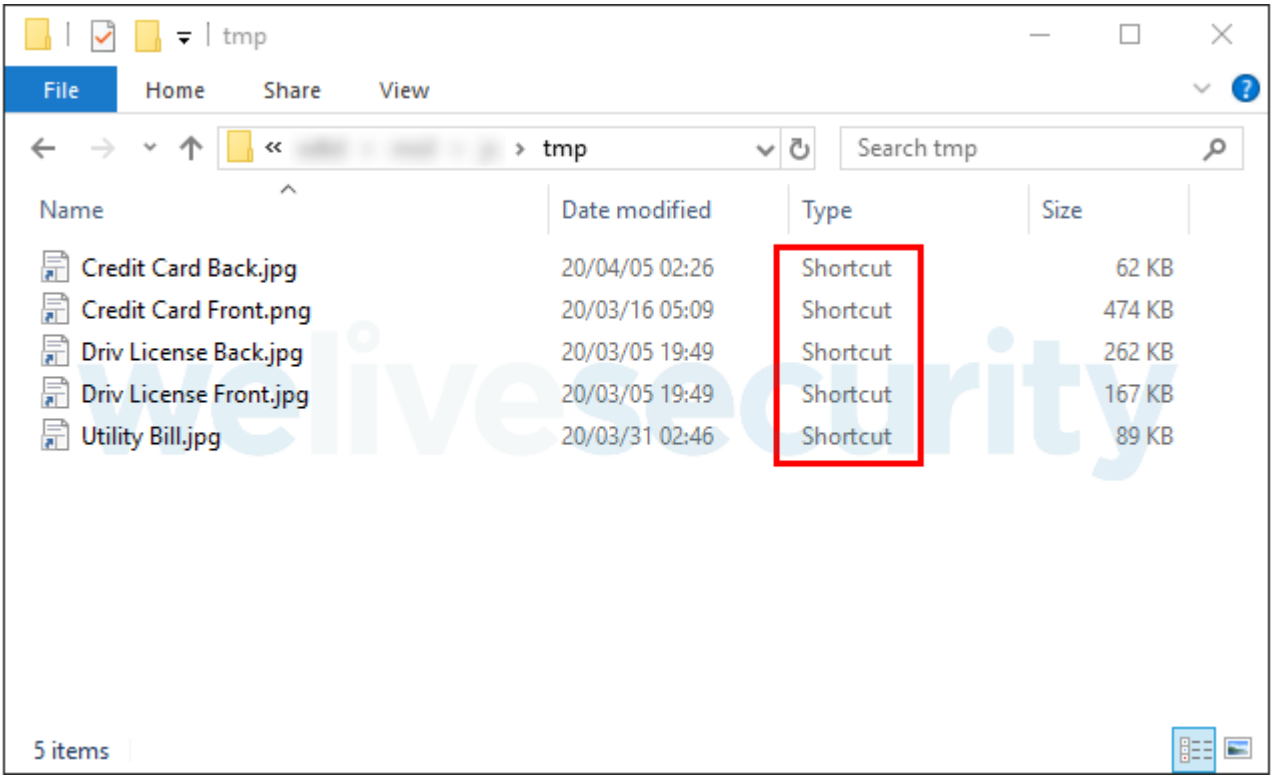

Figure 1. Malicious LNK files

Once a shortcut file is opened (it doesn’t matter which one, as they all do the same thing), it looks in the contents of its own file for lines with a specific marker and writes them to a .js file. Then this malicious JavaScript file is executed and it writes and opens a decoy file with the same name as the shortcut, but with the correct extension. It also deletes the shortcut file. The documents used as decoys are mostly photos of credit cards, identity documents, or bills with proof of address, as many financial institutions require these documents from their customers when they join, according to regulations (this is known as “Know Your Customer”). One such decoy is shown in Figure 2 (blurred for privacy).

Figure 2. Photo of the back of an ID card, used as a decoy

These decoy documents seem genuine, and we assume that they have been collected by this group during years of operation. Documents are collected actively in the group's current operations, as it targets technical support representatives and account managers, who regularly receive these kinds of documents from their customers. The group reuses the documents on different targets, unless the targets are from different regions.

The JavaScript component is the first stage of the attack and can deploy other malware such as a C# spy component, Golden Chickens components or several Python-based tools. The name Evilnum was given to the C# component by other researchers in the past, but the JS component also has been referred to as Evilnum. We have named the group Evilnum as that is the name of their flagship malware, and we'll refer to the various malware pieces as components. An overview of these is shown in Figure 3.

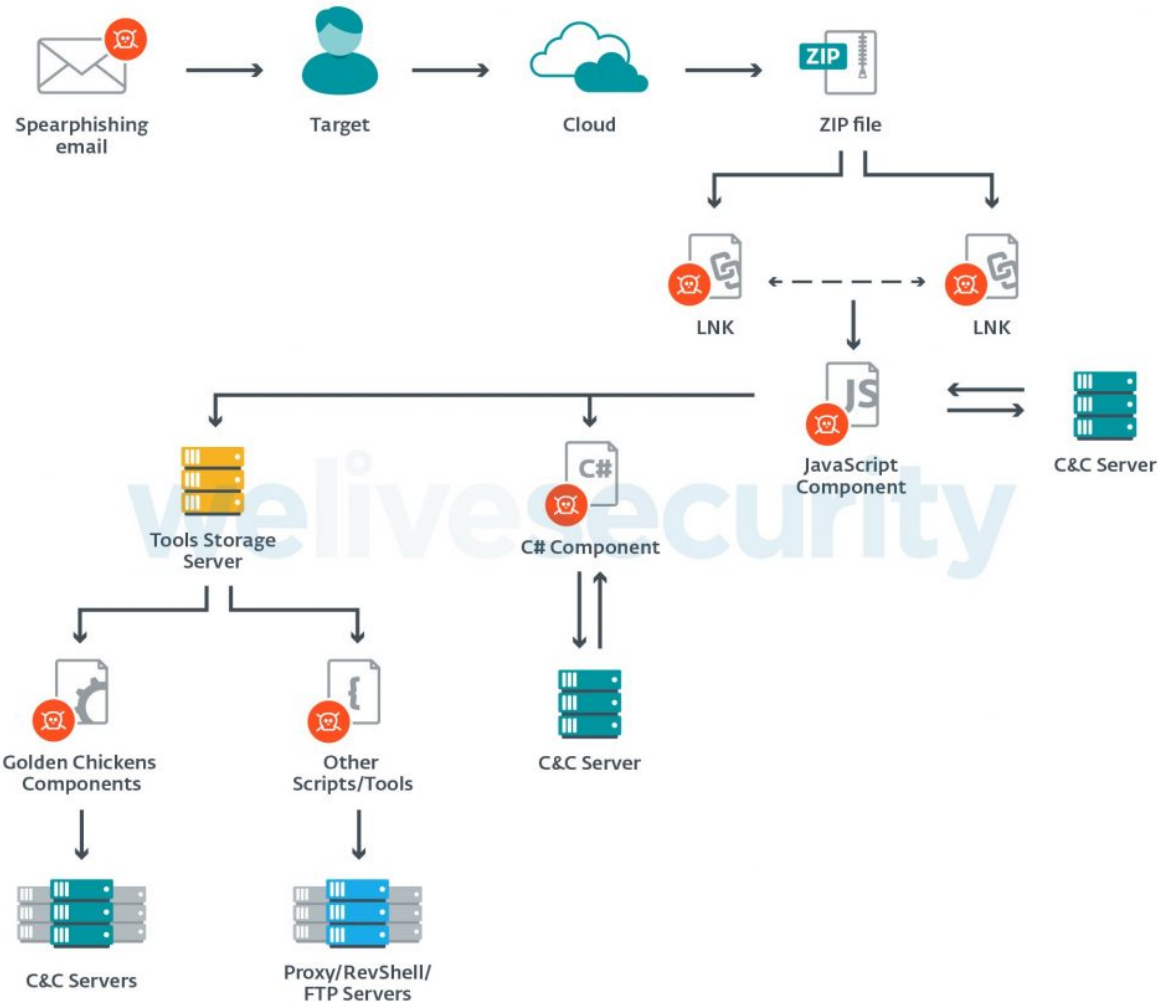

Figure 3. Evilnum components

Each of the various components has its own C&C server, and each component operates independently. The operators of the malware manually send commands to install additional components and use post-compromise scripts and tools if they consider them necessary.

Most servers used by the malware are referenced by IP addresses; domain names have not been used. The only exceptions are the C&C servers used by the Golden Chickens components; malware purchased from a MaaS provider, as we describe later.

Those referenced by an IP address can be split into two groups, based on the hosting provider. The majority of them are hosted with FreeHost, a Ukrainian provider. The rest are hosted in the Netherlands, with Dotsi.

JS Component: First compromise

This component communicates with a C&C server and acts as a backdoor without the need for any additional program. However, in most attacks that we have seen, the attackers deployed additional components as they saw fit and used the JS malware only as a first stage.

The first known mention of this JavaScript malware was in May 2018 in this pwncode article. The malware has changed since then and we illustrate these changes in Figure 4.

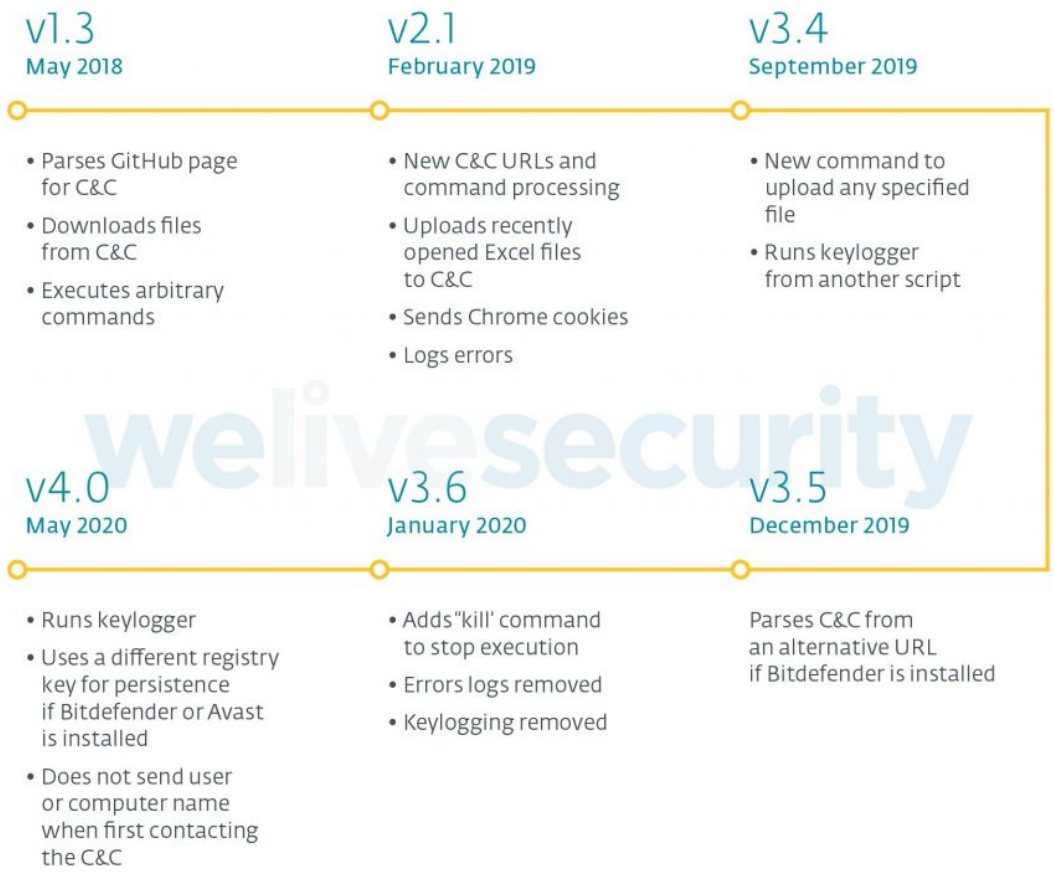

Figure 4. Timeline of changes in JS component

Differences between version 1.3 and the others are noteworthy, as the server-side code for the C&C was changed and commands are different. In that early version it was not possible to upload files to the C&C, only to download files to the victim's computer. Also, as new versions appeared, the malware was extended with some Python scripts (see the *Post-compromise toolset* section) and external tools such as ChromeCookiesView.

Despite the differences, the core functionalities remain the same in all versions, including the retrieval of the C&C server's address from GitHub, GitLab or Reddit pages created specifically for that purpose. Figure 5 shows an example of a Reddit page that is parsed by the malware to retrieve a C&C address.

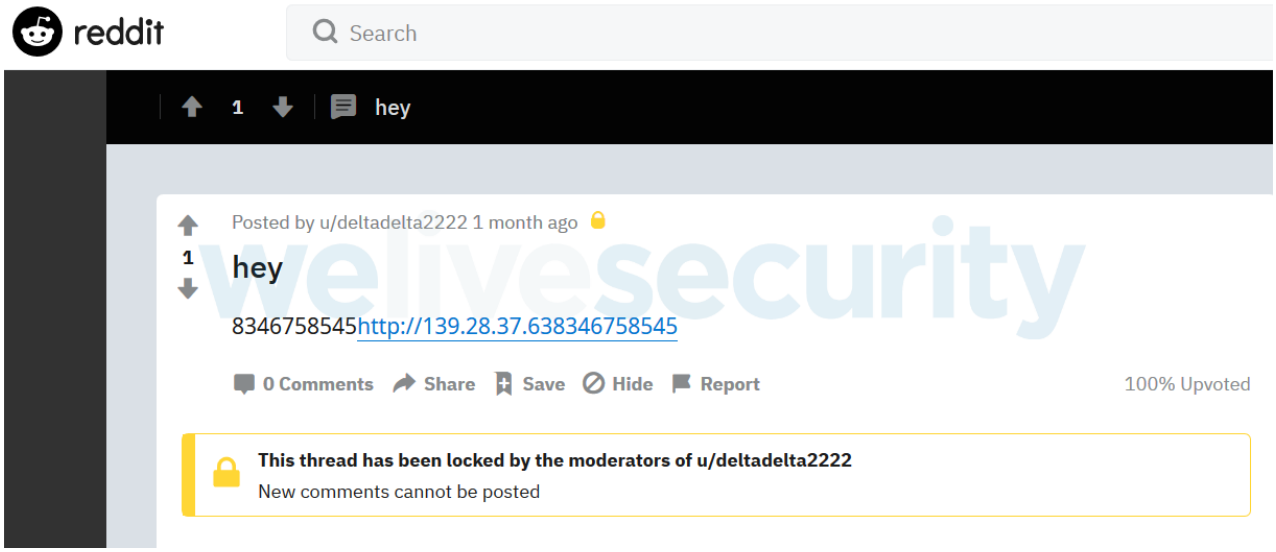

Figure 5. Reddit page with the C&C server for the JS component

This component achieves persistence through the Run registry key and has full backdoor capabilities: it can download and execute binaries, run arbitrary commands or upload files from the victim computer to the C&C server. We will not go into details about the technical aspects of this component, as a good analysis of the latest version was published recently by Prevailion.

C# Component: Evil, not so evil

In March 2019, Palo Alto Networks described malware with very similar functionality to the JS component, but coded in C#. That version (2.5) obtained the address of its C&C by dividing a number by 666, and was therefore named Evilnum by Palo Alto Networks researchers. Since then there have been new versions of the C# malware, the latest of them being version 4.0, which we first saw in April 2020. The number 666 is not used anymore and the PDB paths of the executables show that the developers call their malware “Marvel”. However, we will continue to name the malware Evilnum to avoid creating confusion.

The latest version comes bundled in an MSI file (Windows Installer) and runs independent of the JS component. Furthermore, it has different C&Cs than the JS component. However, in all cases that we have seen, the C# component was downloaded and executed after the JavaScript malware gained initial access. The structure of this component is shown in Figure 6.

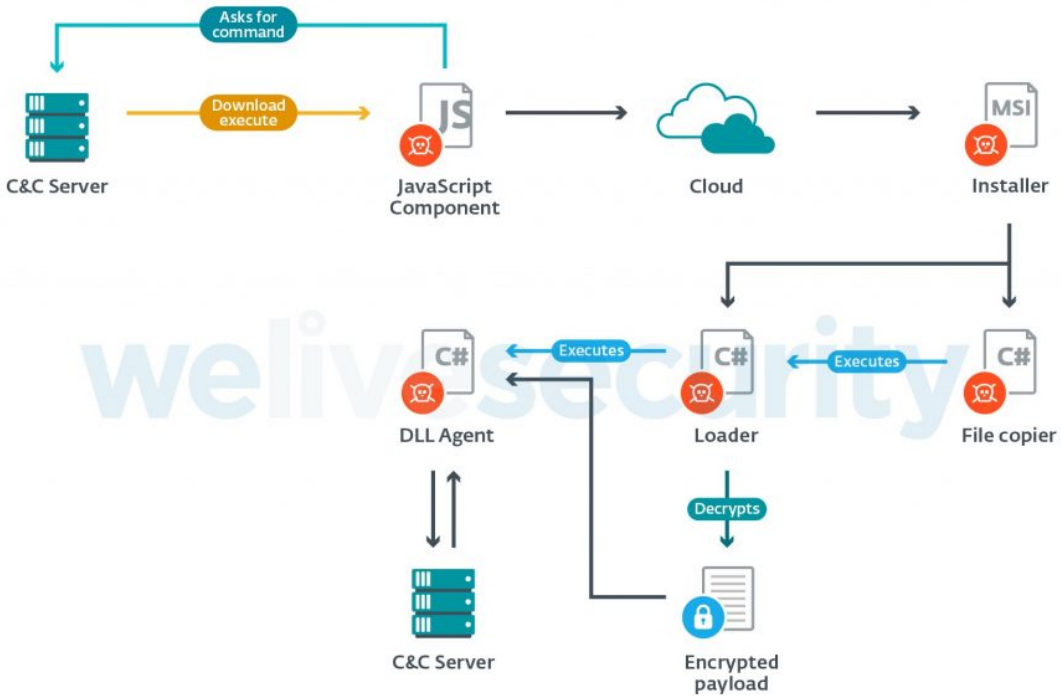

Figure 6. Parts of the C# component

When the MSI file is executed, three malicious components, along with some .NET Framework library files, are written to disk in %LOCALAPPDATA%\Microsoft\Mediia. The file copier is the first to be executed and its only purpose is to move the files to another location in %LOCALAPPDATA% (see the *Indicators of Compromise* section for the folder names). The loader is then executed and it loads and decrypts the contents of the file System.Memmory.dll, which is the actual malicious payload (DLL Agent) for the C# component. AES encryption is used for the DLL and for obfuscation of the strings in the payload. The same key and initialization vector are used to encrypt the strings in all of the different versions.

The IP address of the C&C server is hardcoded and in plain text. A GET request is sent for /Validate/valsrv and if the response body contains the text youwillnotfindthisanywhare, then the server is accepted. Otherwise, a GitLab page is parsed to get the IP address of a second server.

The following capabilities are present in version 4.0:

- Take screenshots if the mouse has been moved in a period of time, and send them to the C&C, base64 encoded. The image is stored in a file called SC4.P7D
- Run commands
- Run other binaries via cmd.exe
- Send information such as computer name, username and antivirus installed
- Persist in a compromised system by creating registry keys

Commands

The commands that can be sent to the malware are:

- killme: stops the malware and removes persistence
- mouse: moves the mouse. With this action a screenshot will be taken
- cookies: sends Chrome cookies to the C&C
- passwords: sends Chrome saved passwords. We believe they focus on Chrome not based on market share (after all, these are targeted attacks), but because of the ease of processing cookies and retrieving stored passwords
- Other commands to be run directly with cmd.exe

Version 2.5 was the first documented version of the C# component (first seen by ESET in December 2018). Then we saw v2.7.1 (November 2019), v3 (December 2019) and v4.0 (April 2020). The most important differences between the latest version of the malware and previous ones are:

- The main payload is a 32-bit DLL. Previously, it was a 64-bit EXE file.
- HTTPS communication in the latest version
- There is no "reverse" command anymore. It was used in previous versions to open a reverse shell. This is now done with other scripts

The JS and C# components are connected to each other: the latter takes screenshots whereas the former doesn't, but it has code that looks for screenshot files and sends them to its C&C server. The C# component also deletes all files with the .lnk extension in the %LOCALAPPDATA%\Temp folder, cleaning leftovers from the initial compromise by the JS component. So even if the C# component has limited functionalities (it can't download or upload files), it provides redundancy with a different C&C server and extra persistence in case the JS component is detected or removed from the victim's computer.

Golden Chickens components: TerraLoader family

In a small number of cases, the Evilnum group has also deployed some tools purchased from a Malware–as–a–Service provider. This term is used to describe malware authors who offer not only their malicious binaries, but also any necessary infrastructure (such as the C&C servers) and even technical support to their criminal customers.

In this case the MaaS provider is known as Golden Chickens and has other customers (apart from this group), such as FIN6 and Cobalt Group. Older versions of all the components that we describe in the following sections were seen previously, in an attack against eCommerce merchants that Visa attributed to FIN6 in February 2019. We believe that FIN6, Cobalt Group and Evilnum group are not the same, despite the overlaps in their toolsets. They just happen to share the same MaaS provider.

The Golden Chickens tools come as ActiveX components (OCX files) and all of them contain TerraLoader code, which serves as a common loader for the various payloads available to Golden Chickens' customers. These tools are used by Evilnum as follows:

- The attackers manually send a command to the JS or C# component to drop and execute a batch file from one of their servers.
- That batch file writes a malicious INF file and supplies it as a parameter to the Microsoft utility cmstp.exe, which executes a remote scriptlet specified in the INF file. This technique has been documented in the MITRE ATT&CK knowledge base as CMSTP; an example of how this technique is used may be found here. This technique has been used in the past by Cobalt, another financially motivated group.
- The remote scriptlet contains obfuscated JS code that drops an OCX file and executes it via regsvr32.exe.

The TerraLoader code performs several integrity checks before dropping the payload. These checks implement anti-debugging techniques and try to identify anomalies to prevent execution in sandboxed environments. Some of these techniques range from detecting incorrect parameters, filenames and extensions, to detecting hardware breakpoints or identifying specific modules loaded into the subject process. Should these checks all pass, the actual payload is decrypted and executed.

We have seen Evilnum deploy the following Golden Chickens payloads in their attacks:

- More_eggs
- A Meterpreter payload that we will call TerraPreter
- TerraStealer
- TerraTV

Researchers from Positive Technologies recently analyzed some tools used by the Cobalt group, including More_eggs version 6.6, which is one of the versions used by Evilnum group. They have a very good analysis of TerraLoader, so we suggest checking their report (section 4).

More_eggs

More_eggs is a JavaScript backdoor that communicates with a C&C server and accepts commands. It has been used in the past by other groups targeting financial companies. Evilnum uses it in conjunction with its homemade backdoors in order to provide redundancy and additional persistence on victim networks.

We have seen Evilnum use 32-bit ActiveX components with TerraLoader code that runs More_eggs versions 6.5, 6.6 and 6.6b – the latest available versions. They do so by dropping msxsl.exe (a command line transformation utility that is a legitimate Microsoft executable) and having it execute the JavaScript code, very similar to what was described in this article by IRIS.

The dropped JavaScript code is generated on the fly by the ActiveX component, and there are some considerations during analysis:

- The initial JS code that executes exe has a hardcoded absolute path, so executing it from another location or with another user will fail.

- The final More_eggs payload is encrypted with a key that has the hostname and processor family information appended at the end. An example key is:

cvyLMmtGSKmPMfzJjGyg552DESKTOP-FQAT01XIntel64 Family 6 Model 94 Stepping 3, GenuineIntel

The core functionalities are the same as described in the article linked above, although there is a new command, more_time, not mentioned there. This command is similar to the documented command via_c, which executes its parameter with cmd.exe /v /c <parameter>. The difference is that it additionally sends the output back to the C&C (via_c only sends whether or not the command succeeded).

TerraPreter

Evilnum group also uses 64-bit executables that decrypt and run a Meterpreter instance in memory. The use of Meterpreter gives them flexibility and the ability to run various payloads in a stealthy and extensible way.

The structure of these components and the integrity checks implemented were identified as TerraLoader code. That's why we refer to these components as TerraPreter. Decompiled code of the main malicious routine is shown in Figure 7.

```
__int64 malcode()
{
  signed int hDC; // [rsp+0h] [rbp-40h]
  signed __int64 Size; // [rsp+10h] [rbp-30h]
  signed int lpDecryptedPayload; // [rsp+20h] [rbp-20h]

  Dummy();
  Dummy();
  RC4_InitKey();
  Dummy();
  if ( CheckCommandLineArguments() == 1 )
    return 0i64;
  if ( CheckExtensionAndFileName() == 1 )
    return 0i64;
  Size = &DecryptedTarget - &EncryptedBuff;
  if ( &EncryptedBuff <= 0 || Size <= 0 )
    return 0i64;
  Dummy();
  lpDecryptedPayload = HeapCreate(0x40000u, Size, Size);
  Dummy();
  if ( !lpDecryptedPayload )
    return 0i64;
  if ( !RC4(&EncryptedBuff, Size, RC4Key) )
    return 0i64;
  Dummy();
  if ( !memcpy(lpDecryptedPayload, &EncryptedBuff, Size) )
    return 0i64;
  Dummy();
  hDC = GetModuleHandleW(0i64);
  Dummy();
  if ( hDC )
  {
    Dummy();
    GrayStringW(hDC, 1, lpDecryptedPayload, 1i64, 1, 1, 1, 1, 1);
  }
  return 0i64;
}
```

Figure 7. Decompiled code for Meterpreter Loader components

The routine labeled Dummy calls a series of APIs that don't do anything. The RC4 function initialization brute-forces the key to use by taking a base string and appending a number to it that is incremented in each iteration. It then decrypts a 16-byte buffer with the candidate key using RC4. If the decrypted buffer matches a hardcoded string, then that candidate key will be the chosen RC4 key for later use. We believe this may be a time-wasting countermeasure against emulators.

After the embedded buffer with the payload is decrypted, the malware will finally set a callback to the GrayStringW API function, pointing to the decrypted buffer. After going through many layers of decoding, Meterpreter's metsrv.dll is loaded in memory. From

this point on, what we see is regular Meterpreter behavior that has not been modified. However, we will continue to describe how communications are performed.

TerraPreter communicates with a C&C server using HTTPS and retrieves a series of commands. C&Cs we have seen contacted are cdn.lvsys[.]com and faxing-mon[.]best. The first one was redirected to d2nz6secq3489l.cloudfront[.]net. Every time a C&C receives a request, it sends different binary data XORed with a random 4-byte key. The malware reads the key to be used for decryption from the first 4 bytes of a 32-byte header that prefixes the encrypted data. Figure 8 shows an example.

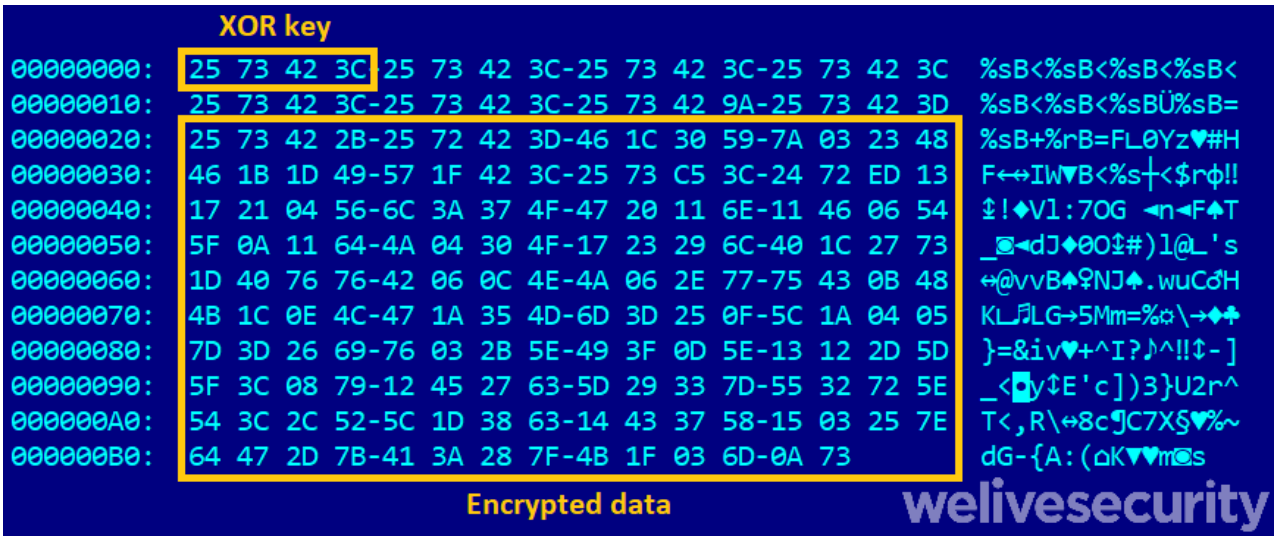

Figure 8. Data sent by the C&C

The first command sent by the C&C is core_patch_url, which changes the last part of the URL for subsequent requests. Then core_negotiate_tlv_encryption is sent by the C&C, along with its public key. From this point on, messages will be encrypted before they are XORed.

TerraStealer and TerraTV

TerraStealer is also known as SONE or Stealer One. It scans for many browsers, email, FTP and file transfer applications, to steal cookies and credentials. One of the binaries we analyzed had logging activated. Part of one such log is shown in Figure 9.

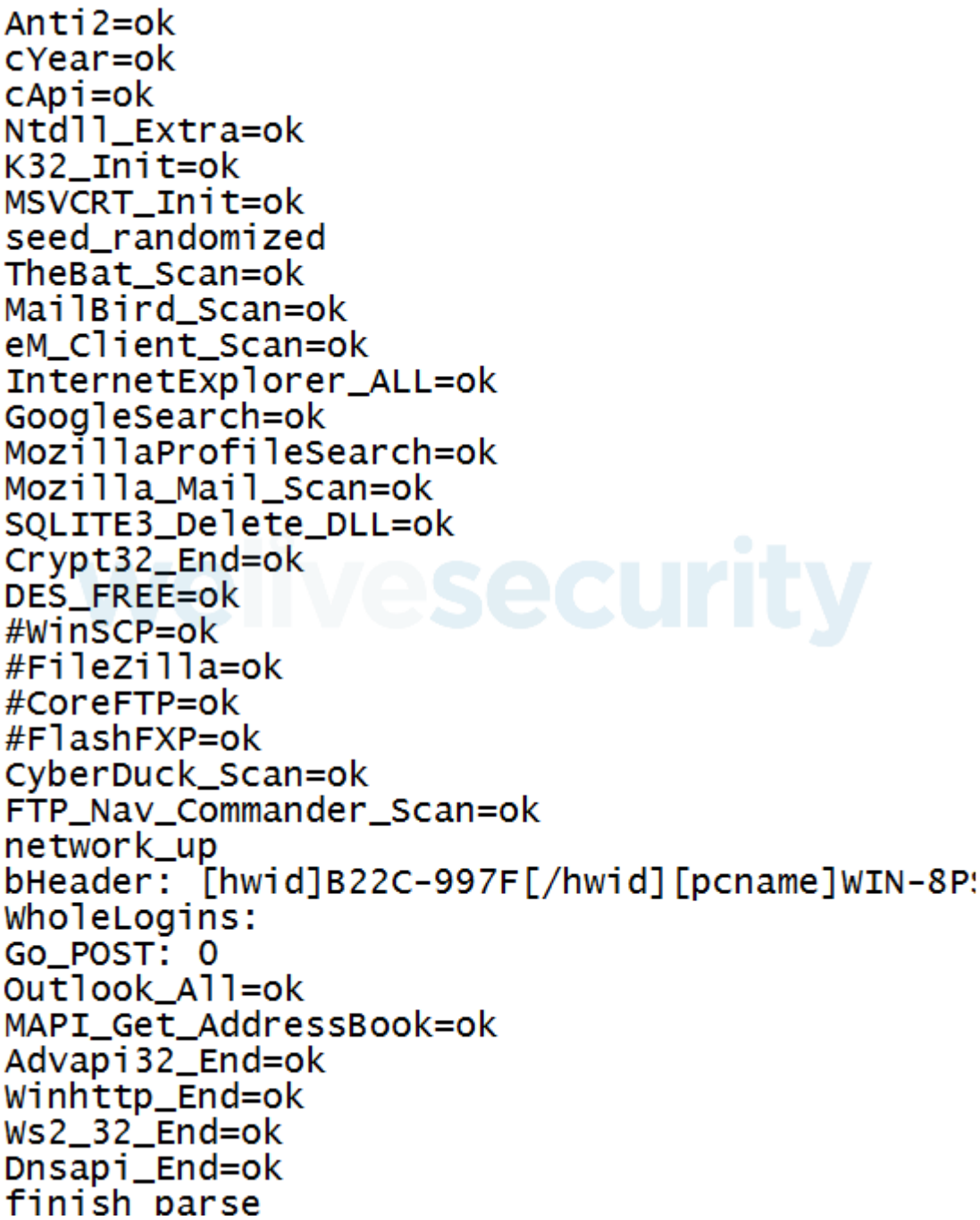

Figure 9. TerraStealer log

Another component used by this group is a variant of TerraTV. It runs a legitimate TeamViewer application but hides its user interface elements, so that the operators of the malware can connect to the compromised computer undetected.

When executed, TerraTV drops several signed TeamViewer components into C:\Users\Public\Public Documents\57494E2D3850535046373333503532\. The dropped files are shown in Figure 10.

ACTIVEDS.dll is not signed and it is where the malicious code resides. There is a Windows DLL with that same name in the system folder, but since the malicious DLL is in the same directory as the TeamViewer executable, it is found first, and therefore is loaded instead of the Windows DLL. This is known as DLL search order hijacking. This

ACTIVEDS.dll hooks several API calls in the TeamViewer executable to hide the application's tray icon and to capture login credentials. The part of the code where the hooks are set is shown in Figure 11.

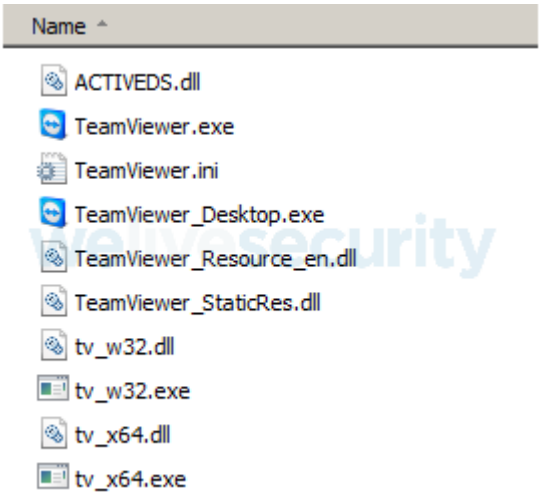

Figure 10. TeamViewer files dropped by TerraTV

```
  v15 = 0;
  hModule = LoadLibraryW_0(LibFileName);        // kernel32.dll
  if ( !hModule )
    return (GetLastError_0)();
  v16 = LoadLibraryW_0(aUser32Dll_0);
  if ( !v16 )
    return (GetLastError_0)();
  v13 = LoadLibraryW_0(aShell32Dll_0);
  if ( !v13 )
    return (GetLastError_0)();
  v12 = LoadLibraryW_0(aGdi32Dll_0);
  if ( !v12 )
    return GetLastError_0(0, v13, hModule);
  v0 = GetProcAddress_0(hModule, aCreateprocessw);
  dword_743EC7D4 = SetHook(v0, sub_743D6370);
  v1 = GetProcAddress_0(hModule, aCreatemutexw);
  dword_743EC7D0 = SetHook(v1, sub_743D5F00);
  v2 = GetProcAddress_0(v13, aShellexecuteex);
  dword_743EC7BC = SetHook(v2, sub_743D63D0);
  v3 = GetProcAddress_0(v13, aShellNotifyico);
  dword_743EC7B4 = SetHook(v3, sub_743D5EE0);
  v4 = GetProcAddress_0(v16, aShowwindow);
  dword_743EC7AC = SetHook(v4, sub_743D6460);
  v5 = GetProcAddress_0(v16, aCreatewindowex);
  dword_743EC7B8 = SetHook(v5, sub_743D6090);
  v6 = GetProcAddress_0(v16, aCreatedialogpa);
  dword_743EC7C0 = SetHook(v6, sub_743D6300);
  v7 = GetProcAddress_0(v16, aIswindowvisibl);
  dword_743EC7C8 = SetHook(v7, sub_743D5EF0);
  v8 = GetProcAddress_0(v16, aSetwindowtextw);
  dword_743EC7B0 = SetHook(v8, writeIDPass);
  v9 = GetProcAddress_0(v16, aDefwindowprocw);
  dword_743EC7CC = SetHook(v9, sub_743D61C0);
  v10 = GetProcAddress_0(v12, aBitblt);
  dword_743EC7C4 = SetHook(v10, sub_743D5DB0);
  return v15;
}
```

Figure 11. Hooks set for TeamViewer

The Windows API call DefWindowProcW (called several times by the TeamViewer executable to process messages directed to its main window) is hooked with a routine that writes TeamViewer's ID and password to the file %APPDATA%\log_CZ72kGqTdU.txt. With these credentials, and TeamViewer running with no visible tray icon or window, the operators of the malware can remotely control the computer, via its GUI, at any time.

Post-compromise toolset

The malicious components previously mentioned are frequently extended with several additional tools in the Evilnum group's arsenal. In most of the compromises we have seen, the attackers utilized publicly available tools, but have also developed some custom scripts. Usually they keep their tools in password-protected archives on their servers and decompress them on a victim's PC as needed.

Python-based tools

- Reverse shell over SSL script: A very short script that takes the server and port as command line arguments.
- SSL proxy that uses PythonProxy, junction, plink and stunnel. It can also connect to an FTP server or use pysoxy. We have seen the script being used with the "proxy" setting and 185.62.189[.]210 as the server.
- LaZagne to retrieve stored passwords
- IronPython along with libraries for taking screenshots, keylogging and recording DirectSound audio

Other publicly available tools

- PowerShell scripts: for example, Bypass-UAC
- Several NirSoft utilities; for example, Mail PassView, to retrieve passwords from email clients, and ProduKey, to get Microsoft Office and Windows Licenses

Conclusion

The Evilnum group has been operating for at least two years and was active at the time of this writing. It has an infrastructure for its operations with several different servers: one for communications with the JS component, another for the C# component, a different one for storing its tools and exfiltrated data, proxy server, and so on. This group targets fintech companies that provide trading and investment platforms for their customers. The targets are very specific and not numerous. This, and the group's use of legitimate tools in its attack chain, have kept its activities largely under the radar. Thanks to our telemetry data we were able to join the dots and discover how the group operates, uncovering some overlaps with other known APT groups. We think this and other groups share the same MaaS provider, and the Evilnum group cannot yet be associated with any previous attacks by any other APT group.

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

For any inquiries, or to make sample submissions related to the subject, contact us at threatintel@eset.com.

Special thanks to Ignacio Sanmillan for his help with the analysis of the Golden Chickens components.

MITRE ATT&CK techniques

Tactic	ID	Name	Description
Initial Access	T1192	Spearphishing Link	Emails contain a link to download a compressed file from an external server.
Execution	T1191	CMSTP	cmstp.exe is used to execute a remotely hosted scriptlet that drops a malicious ActiveX file.
	T1059	Command-Line Interface	cmd.exe is used to execute commands and scripts.
	T1129	Execution through Module Load	The malicious payload for the version 4.0 C# component is loaded from a DLL. TerraTV loads a malicious DLL to enable silent use of TeamViewer.
	T1061	Graphical User Interface	TerraTV malware allows remote control using TeamViewer.
	T1086	PowerShell	Evilnum group executes LaZagne and other PowerShell scripts after their JS component has compromised a target.
	T1117	Regsvr32	Evilnum group uses regsvr32.exe to execute their Golden Chickens tools.
	T1064	Scripting	Initial compromise and post-compromise use several JavaScript, Python and PowerShell scripts.
	T1218	Signed Binary Proxy Execution	msiexec.exe is used to install the malicious C# component.
	T1204	User Execution	Victims are lured to open LNK files that will install a malicious JS component.
	T1047	Windows Management Instrumentation	WMI is used by the JS component to obtain information such as which antivirus product is installed.
	T1220	XSL Script Processing	More_eggs malware uses msxsl.exe to invoke JS code from an XSL file.

Tactic	ID	Name	Description
Persistence	T1060	Registry Run Keys / Startup Folder	Registry Run keys are created in order to persist by the JS and C# components, as well as More_eggs
	T1108	Redundant Access	Evilnum components are independent and provide redundancy in case one of them is detected and removed.
	T1179	Hooking	TerraTV malware hooks several API calls in TeamViewer.
Defense Evasion	T1038	DLL Search Order Hijacking	TerraTV malware has TeamViewer load a malicious DLL placed in the TeamViewer directory, instead of the original Windows DLL located in a system folder.
	T1088	Bypass User Access Control	A PowerShell script is used to bypass UAC.
	T1116	Code Signing	Some of the Golden Chickens components are malicious signed executables. Also, Evilnum group uses legitimate (signed) applications such as cmstp.exe or msxsl.exe as a defense evasion mechanism.
	T1090	Connection Proxy	Connection to a proxy server is set up with post-compromise scripts.
	T1140	Deobfuscate/Decode Files or Information	Encryption, encoding and obfuscation are used in many Evilnum malware components.
	T1107	File Deletion	Both JS and C# components delete temporary files and folders created during the initial compromise.
	T1143	Hidden Window	TerraTV runs TeamViewer with its window and tray icon hidden.
	T1036	Masquerading	The C# component has its payload in system.memory.dll , which masquerades as a benign .NET Framework DLL.
	T1112	Modify Registry	Evilnum modifies the registry for different purposes, mainly to persist in a compromised system (for example, by using a registry's Run key).

Tactic	ID	Name	Description
	T1027	Obfuscated Files or Information	Encryption, encoding and obfuscation is used in many Evilnum malware components.
	T1497	Virtualization/Sandbox Evasion	The Golden Chickens components implement several integrity checks and evasion techniques.
Credential Access	T1003	Credential Dumping	Scripts and tools such as LaZagne are used to retrieve stored credentials.
	T1503	Credentials from Web Browsers	The C# component retrieves stored passwords from Chrome.
	T1056	Input Capture	Custom Python scripts have been used for keylogging.
	T1539	Steal Web Session Cookie	Evilnum malware steals cookies from Chrome.
Discovery	T1012	Query Registry	More_eggs queries the registry to know if the user has admin privileges.
	T1063	Security Software Discovery	Both the JS and C# components search for installed antivirus software.
	T1518	Software Discovery	TerraStealer malware looks for specific applications.
	T1082	System Information Discovery	Information about the system is sent to the C&C servers.
Collection	T1074	Data Staged	Data is stored in a temporary location before it is sent to the C&C.
	T1005	Data from Local System	The JS component (v2.1) has code to exfiltrate Excel files from the local system.
	T1114	Email Collection	TerraStealer malware targets email applications.
	T1056	Input Capture	Keystrokes are logged with a Python script.

Tactic	ID	Name	Description
	T1113	Screen Capture	Screenshots are taken by some Evilnum malware components.
Command and Control	T1043	Commonly Used Port	HTTP and HTTPS are used for C&C communication.
	T1132	Data Encoding	Some of the data sent to the C&C is base64-encoded.
	T1008	Fallback Channels	The JS and C# components can obtain a new C&C by parsing third-party webpages if the original C&C is down.
	T1104	Multi-Stage Channels	Evilnum malware uses independent C&C servers for its various components.
	T1219	Remote Access Tools	TerraTV malware uses TeamViewer to give control of the compromised computer to the attackers.
	T1105	Remote File Copy	Files are uploaded to/downloaded from a C&C server.
	T1071	Standard Application Layer Protocol	HTTP and HTTPS are used for C&C.
	T1032	Standard Cryptographic Protocol	More_eggs malware uses RC4 to encrypt data to be sent to the C&C.
	T1102	Web Service	GitHub, GitLab, Reddit and other websites are used to store C&C server information.
Exfiltration	T1022	Data Encrypted	Some Evilnum components encrypt data before sending it to the C&C.
	T1048	Exfiltration Over Alternative Protocol	Scripts are manually deployed by the malware operators to send data to an FTP server.
	T1041	Exfiltration Over Command and Control Channel	Data is exfiltrated over the same channel used for C&C.

Matías Porolli 9 Jul 2020 - 11:30AM

Newsletter