To the Moon and back(doors): Lunar landing in diplomatic missions

ESET Research

ESET researchers provide technical analysis of the Lunar toolset, likely used by the Turla APT group, that infiltrated a European ministry of foreign affairs

Filip Jurčacko

15 May 2024 • , 30 min. read

ESET researchers discovered two previously unknown backdoors – which we named LunarWeb and LunarMail – compromising a European ministry of foreign affairs (MFA) and its diplomatic missions abroad. We believe that the Lunar toolset has been used since at least 2020 and, given the similarities between the tools' tactics, techniques, and procedures (TTPs) and past activities, we attribute these compromises to the infamous Russia-aligned cyberespionage group Turla, with medium confidence. We recently presented our insights from this research at this year's ESET World conference and provide more details about our findings in this blogpost.

Key points of the blogpost:

- ESET Research discovered two previously unknown backdoors – LunarWeb and LunarMail – used in the compromise of a European MFA and its diplomatic missions.
- LunarWeb, deployed on servers, uses HTTP(S) for its C&C communications and mimics legitimate requests, while LunarMail, deployed on workstations, is persisted as an Outlook add-in and uses email messages for its C&C communications.
- Both backdoors employ the technique of steganography, hiding commands in images to avoid detection.
- Both backdoors utilize a loader that uses the DNS domain name for decryption of the payload, share portions of their codebases, and have the unusual capability of being able to execute Lua scripts.
- The loader can have various forms, including trojanized open-source software, demonstrating the advanced techniques used by the attackers.

Turla, also known as Snake, has been active since at least 2004, possibly even dating back to the late 1990s. Believed to be part of the Russian FSB, Turla mainly targets high-profile entities such as governments and diplomatic organizations in Europe, Central Asia, and the Middle East. The group is notorious for breaching major organizations, including the US Department of Defense in 2008 and the Swiss defense company RUAG in 2014. Over the past few years, we have documented a large part of Turla's arsenal on WeLiveSecurity.

Our current investigation began with the detection of a loader decrypting and running a payload, from an external file, on an unidentified server. This led us to the discovery of a previously unknown backdoor, which we named LunarWeb. Subsequently, we detected a similar chain with LunarWeb deployed at a diplomatic institution of a

European MFA. Notably, the attacker also included a second backdoor – which we named LunarMail – that uses a different method for command and control (C&C) communications.

During another attack, we observed simultaneous deployments of a chain with LunarWeb at three diplomatic institutions of this MFA in the Middle East, occurring within minutes of each other. The attacker probably had prior access to the domain controller of the MFA and utilized it for lateral movement to machines of related institutions in the same network.

Further examination uncovered additional pieces of the puzzle, including components utilized in the initial stage of the compromise and a limited number of commands issued by the attacker. The timestamps in the oldest samples and the versions of the libraries used suggest that this toolset has been operational since at least 2020, possibly earlier. Our technical analysis focuses on the techniques these backdoors employ, such as steganography, and communication methods.

Victimology

According to ESET telemetry, the compromised machines that we managed to identify belong to a European MFA and are primarily related to its diplomatic missions in the Middle East.

Technical analysis

Initial access

We don't know exactly how initial access was gained in any of the compromises. However, recovered installation-related components and attacker activity suggest possible spearphishing and abuse of misconfigured network and application monitoring software Zabbix. Potential Zabbix abuse is suggested by a LunarWeb installation component imitating Zabbix logs, and a recovered backdoor command used to get the Zabbix agent configuration. Additionally, evidence of spearphishing includes a Word document installing a LunarMail backdoor via a malicious macro.

Below, we provide details of the installation-related components and initial attacker activity.

Stage 0 – LunarWeb initial server compromise

While we don't have the full picture of the initial compromise, we found an installation-related component in one of the server compromises – a compiled version of an ASP.NET web page originating from following source files:

- <IIS_web_root>\aspnet_client\system_web.aspx
- <IIS_web_root>\aspnet_client\system_web.cs

The system_web.aspx filename is a known IoC of Hafnium, a China-aligned APT known for exploiting vulnerabilities in Microsoft Exchange Server software. However, we believe this is either a coincidence or a false flag.

When the system_web.aspx page is requested, it responds with a benign-looking Zabbix agent log. However, the page covertly expects a password in a cookie named SMSKey. If provided, the password (combined with the salt Microsoft.SCCM.Update.Manager) is used to derive an AES-256 key and IV for decrypting two embedded blobs, which are then dropped to two temporary files in a directory excluded from scanning.

While we don't know the password, the file sizes match further stages in the compromise chain – the Stage 1 loader and Stage 2 blob – containing the LunarWeb backdoor. Lastly, either the attacker or an unknown component renames and moves the two temporary files to their final destinations, and sets up persistence.

During our investigation, we found that the attacker already had network access, used stolen credentials for lateral movement, and took careful steps to compromise the server without raising suspicion. The attacker's steps included copying two log files over the network; these files were deliberately named to mimic Zabbix agent logs. The attacker moved them to the IIS web directory as the system_web page, and sent a HEAD request to the page with a password, which resulted in the creation of two files with .tmp filename extensions. The system_web page files were then deleted, and the dropped .tmp files containing Stages 1 and 2 were moved to the following locations:

- C:\Windows\System32\en-US\winnet.dll.mui
- C:\Windows\System32\DynamicAuth.bin

Finally, to maintain access and execute their code, the attacker set up a Group Policy extension in the registry using the Remote Registry service.

Stage 0 – LunarMail initial user compromise

In another compromise, we found an older malicious Word document, likely from a spearphishing email. Despite being a DOC file, it's actually in DOCX format, which is a ZIP archive that can hold extra content. This document has unusual components: 32- and 64-bit versions of a Stage 1 loader, and a Stage 2 blob containing the LunarMail backdoor.

They are installed using a VBA macro, executed on document opening, that does the following:

1. Calculates a victim ID from the computer name and informs its C&C server by pinging a specific URL with the ID in its subdomain.
2. Creates the directory %USERPROFILE%\Gpg4win and extracts the appropriate files from the extra content in the ZIP/DOCX – Stage 1 loader to gpgol.dll and Stage 2 blob to tempkeys.dat.
3. Sets up persistence via Outlook add-in registry settings and pings another URL containing the ID.

We did not obtain the whole document, but it probably contains a lure that is enticing enough, since it can't be accessed otherwise, to convince the victim to enable macros.

The paths and names used mimic Gpg4win's Outlook add-in, GpgOL. Once deployed, the Stage 1 loader appears in Outlook Add-Ins, as shown in Figure 1.

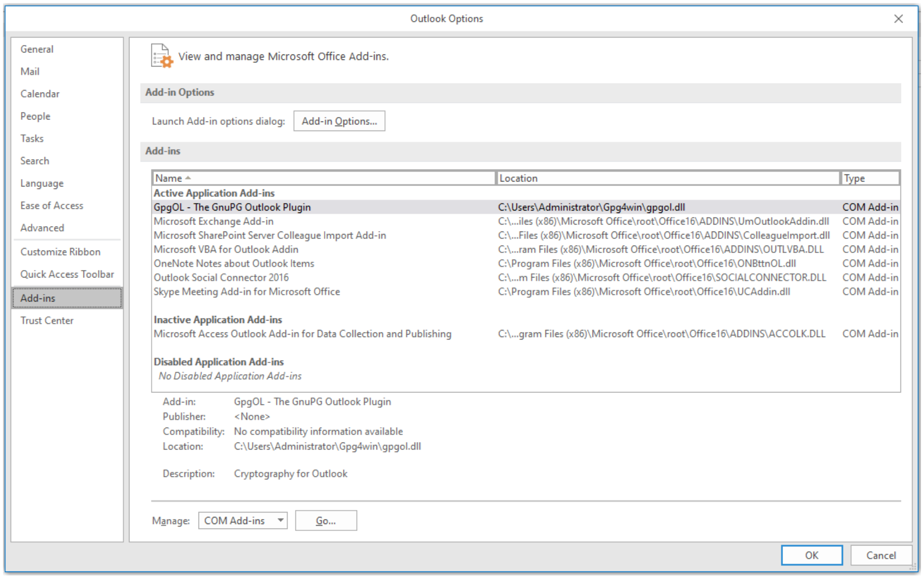

Figure 1. Malicious Outlook add-in

Lunar toolset

Following our analysis of the installers introduced in the previous section, we examine the loaders and finish with analysis of their payloads – two previously unknown backdoors. Figure 2 outlines the components in the two observed compromise chains.

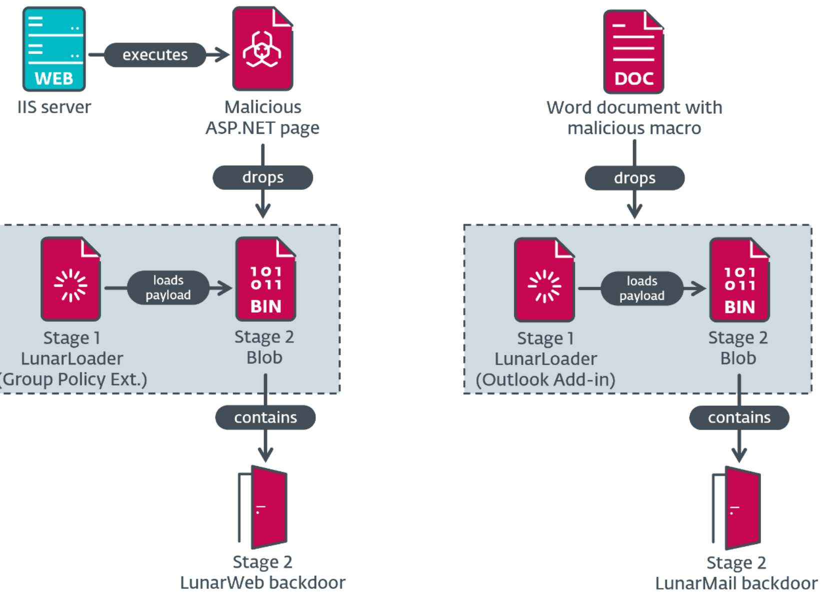

Figure 2. The two observed Lunar toolset compromise chains

Stage 1 – LunarLoader

The execution chain begins with a loader that we have named LunarLoader. It uses RC4, a symmetric key cipher, to decrypt the path to the Stage 2 blob and reads an encrypted payload from it. To ensure that only one loader instance is active, it attempts to open and then create a mailslot with a unique name, instead of a common synchronization object such as mutex or event. It also creates a decryption key, derived from the MD5 hash of the computer's DNS domain name, which it verifies. The payload is then decrypted using AES-256, resulting in a PE file. LunarLoader allocates memory for the PE image and decrypts the name of an exported function in the PE file, which is then run in a new thread. This function contains a reflective loader.

Using the DNS domain name for payload decryption serves as an execution guardrail. The loader correctly executes only in the targeted organization, which may hinder analysis if the domain name is not known.

LunarLoader can have a standalone form or be a part of trojanized open-source software. We observed one case of the latter, with a trojanized AdmPwd, which is a part of Windows Local Administrator Password Solution (LAPS).

We observed that LunarLoader uses three different persistence methods and several file paths, as shown in Table 1.

Table 1. Variants of LunarLoader

Persistence method	Loader path(s)	Host process	Note
Group policy extension	C:\Windows\System32\en-US\winnet.dll.mui C:\Program Files\LAPS\CSE\AdmPwd.dll*	svchost.exe -k GPSvcGroup	The AdmPwd dll is a known legitimate file path of Microsoft LAPS.
System DLL replacement	C:\Windows\System32\tapiperf.dll	wmiprvse.exe	Replacing a legitimate Windows DLL.
Outlook add-in	%USERPROFILE%\Gpg4win\gpgol.dll	outlook.exe	N/A

Stage 2 blob – payload container

The blob used in Stage 2 consists of four entries – including two unused strings, where the value of one is the base64-encoded version of the string freedom or death or freedom or death (yeah,we are alive), as shown in Figure 3, and 32-bit and 64-bit versions of the payload.

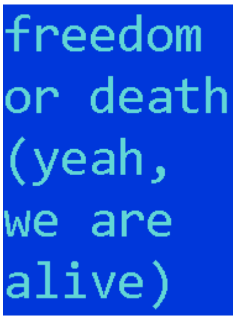

Figure 3. Decoded version of the string, which contains a message

While the purpose of the freedom or death string in the given context isn't explicitly explained, it's common for malware authors to include such strings for a variety of possible reasons, such as tracking different versions of their malware, to serve as a distraction or false lead for analysts, or simply as a form of signature or calling card. In some cases, we found strings instead of a 32-bit payload – such as the string shit happens.

We observed two different backdoors used as payloads. The backdoors seem to use the following DLL names in the export directory, with these suspected meanings:

- mswt[e].dll – web transport (LunarWeb)
- msmt[e].dll – mail transport (LunarMail)

The e suffix is used for the 64-bit versions. The observed file paths for the blob are listed in the IoCs section.

Stage 2 payload #1 – LunarWeb backdoor

LunarWeb, the first payload we discovered, is a backdoor that communicates with its C&C server using HTTP(S) and executes commands it receives. We observed that LunarWeb was deployed only on servers, not user workstations.

During its initialization, LunarWeb attempts to locate or create its state file, which contains entries related to its execution. Then it decrypts strings, mostly related to communication, using RC4 with the static key C1 82 A7 04 21 B6 40 C8 9A C3 79 AD F5 5F 72 86. It also collects victim identification data and uses it to calculate a victim ID, which is used in communications with the C&C server.

After conducting safety checks, the backdoor waits for a few hours before entering its communication loop. This delay is skipped on the backdoor's first run. The security checks include a limit of initial contact attempts with the C&C server, assessing the backdoor's lifespan, and checking C&C server accessibility. If any of the safety conditions fail, LunarWeb self-removes, deleting its files, including the Stage 1 loader and Stage 2 blob. However, the persistence method for the Stage 1 loader is left, potentially leaving detectable traces.

Configuration and state

LunarWeb's configuration is hardcoded into the binary, likely from manual source code changes. The configuration varies between samples, including the C&C servers, their unreachability threshold, the communication format, and the backdoor lifespan.

The backdoor maintains a 512-byte state structure, updated during execution and stored in a file. This file contains three state slots, accessed by index 0, 1, or 2 as shown in Figure 4. The first two slots are modifiable, but unused by this backdoor; only the third slot is used. State slots are encrypted using RC4 with key 99 53 EA 6A AB 29 44 EF BE 36 12 9E F2 3B 5E C9.

```
char __fastcall get_state_entry(basic_stringW *state_file, unsigned int entry_idx, s_entry *out_state_entry)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  res = 0;
  state_file_ = state_file;
  if ( state_file->len && entry_idx <= 2 )
  {
    if ( state_file->capacity >= 8ui64 )
      state_file_ = ret_arg_0(state_file->buff_or_ptr);
    FileW = CreateFileW(state_file_, GENERIC_READ, 0, 0i64, 3u, 0x80u, 0i64);
    v8 = FileW;
    if ( FileW != INVALID_HANDLE_VALUE )
    {
      if ( entry_idx << 9 == SetFilePointer(FileW, entry_idx << 9, 0i64, 0) )// idx * 512
      {
        memset(enc_buff, 0, sizeof(enc_buff));
        NumberOfBytesRead = 0;
        if ( ReadFile(v8, enc_buff, 512u, &NumberOfBytesRead, 0i64) && NumberOfBytesRead == 512 )
        {
          rc4_decrypt(&state_key, 16, enc_buff, 512, out_state_entry);
          res = 1;
        }
      }
      CloseHandle(v8);
    }
  }
  return res;
}
```

Figure 4. Hex-Rays decompilation showing state retrieval

The observed locations of the state files are listed in the IoCs section.

Information collection

LunarWeb collects the following information about its host computer:

- unique victim identification obtained via WMI queries:
  - operating system version with serial number,
  - BIOS version with serial number, and
  - domain name.
- further system information obtained via shell commands:
  - computer and operating system information (output of systeminfo.exe),
  - environment variables,
  - network adapters,
  - list of running processes,
  - list of services, and
  - list of installed security products.

The information is sent to the C&C server on first contact.

Communication

After initialization, LunarWeb communicates with its C&C server using HTTP(S), underneath which is a custom binary protocol with encrypted content.

LunarWeb employs three URLs (containing IP addresses instead of domains) for different purposes. One URL is used for first contact, uploading information about the host computer as described in the previous section. The two remaining URLs are used for getting commands, each being on a different server. We refer to these URLs below as command URLs.

To hide its C&C communications, LunarWeb impersonates legitimate-looking traffic, spoofing HTTP headers with genuine domains and commonly used attributes. It can also receive commands hidden in images. Impersonated attributes from each observed LunarWeb sample are shown in Table 2.

Table 2. Impersonated attributes

Host	User-Agent	Request-URI / Filename
win8.ipv6.microsoft.com	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:80.0) Gecko/20100101 Firefox/80.0	(Non-impersonating URIs)

Host	User-Agent	Request-URI / Filename
i1.c1.eset.com	Host: EES Update (Windows; U; 64bit; BPC 9.0.2047.0; OS: 10.0.16299 SP 0.0 NT; TDB 57524; TPCAT 0; CL 1.0.0; x64c; APP ees; ASP 0.0; FW 32.0; PX 1; CD 1; RA 1; UBR 2166; HVCI 0; SHA256 1; WU 3; HWF: DA7506AC-AB57-4C28-BC32-E6D90B48B66F; PLOC en_us; PCODE 111.0.0; PAR 0; ATH -1; DC 0; PLID 375-GTM-VO6; SEAT 62f587f1; RET 5004) [*sic*]	update.ver.signed livegrid
<MFA_country_news_site>	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0	(Non-impersonating URIs)
ctldl.windowsupdate.com	Microsoft-CryptoAPI/6.1	/msdownload/update/v3/static/trustedr/en/authrootstl.cab /msdownload/update/v3/static/trustedr/en/disallowedcertstl.cab /msupload/update/v3/static/trustedr/stats
ctldl.windowsupdate.com	Microsoft-CryptoAPI/6.1	/msdownload/update/v3/static/trustedr/en/authrootstl.cab /msdownload/update/v3/static/trustedr/stats

Notable examples of impersonation include Windows services (Teredo, Windows Update) and updates of ESET products. In cases of ESET impersonation, the attackers copied the User-Agent (where they slipped in a Host header) and other headers used by updates of our product. Strangely, they spoofed a nonexistent domain in the Host header.

Victim identification is included in HTTP requests, either in a cookie or a URL query parameter. The first method uses randomly generated cookies with a 16-byte identifier (possibly a campaign ID) and a victim ID. The second method appends the victim ID twice to the URL. The suspected campaign ID is present in samples using the second method but is not used. LunarWeb can also use an HTTP proxy server for C&C communications, if needed.

Receiving commands

LunarWeb collects commands from the C&C server via a GET request to the command URL. The request and response format vary across five supported formats, with a hardcoded value determining which to use. Table 3 provides an overview of these formats. We observed usage of formats 2, 3, and 4.

Table 3. Communication formats for getting commands

Format	Command request filename example	Response, extraction, decoding	Response decryption, decompression	Note
0	N/A	Base64	RSA	Short commands only (RSA-4096 512-byte limit).
1	N/A	None	RSA	Short commands only (RSA-4096 512-byte limit).
2	update.ver.signed disallowedcertstl.cab (impersonation specific)	Base64 or none	RSA, AES, zlib	Decoding is skipped in instances where this format is actually used.
3	<random_5_alnum>.jpg	JPG	RSA, AES, zlib	The data is inside a JPG comment.
4	<random_5_alnum>.gif	GIF	RSA, AES, zlib	The data is inside GIF data blocks.

Depending on the communication format, the data received from the C&C server might need decoding using the base64 algorithm or extraction from an image. JPGs are scanned for the comment marker FF FE, while GIFs are parsed using the giflib library. In both cases, the interesting data is embedded in the structures of the image format and not hidden in individual pixels of an image, as in LSB steganography for example.

Communication formats 0 and 1, though not observed, simply decrypt resulting data using RSA-4096. Formats 2, 3, and 4 are more complex. The resulting data starts with an encrypted AES seed, decrypted with RSA-4096 and used to derive a session key. This session key is then used to decrypt the rest of the data using AES-256, followed by zlib decompression.

After decryption and, if needed, decompression, the received data results in a command package. This package, possessing a unique ID, is compared to the last processed ID, stored in the backdoor's state. If they are different, the backdoor processes the package and updates the last ID. The package may hold multiple commands. Each command is executed, and its output sent to the C&C server in a single format, with no steganography, as described in the ensuing Exfiltrating data section.

To perform cryptographic operations, LunarWeb utilizes a statically linked Mbed TLS library. It has two embedded RSA-4096 keys: one for decrypting incoming data and one for encrypting outgoing data. Both use standard parameters and are outlined in our GitHub repository.

Exfiltrating data

First, data is zlib-compressed and encrypted using AES-256, with a session key and IV derived from the data's size, also producing a hash-based message authentication code (HMAC).

For AES encryption, a random 32-byte AES seed is generated and encrypted using RSA-4096. The seed is used to derive a session key in a PBKDF-like manner, SHA-256 hashing the seed and an IV 8,192 times. The same key derivation happens when decrypting received data. The derivation algorithm and encryption code was copied from an older Mbed TLS sample program that was removed from the library in 2021.

Finally, the encrypted data, along with decryption and integrity metadata, is sent. If output data exceeds 1.33 MB after compression, it is split into multiple parts of random size (384–512 KB).

POST requests to the C&C server include impersonation headers and victim identification, and their sending is delayed by a sleep of 34 to 40 seconds. Interestingly, each command package received contains an output URL, which is where to send the result. This could be a different URI on the same C&C server, or a completely different server. In the limited number of command packages that we observed, the output URL was the same as the command URL.

Commands

LunarWeb supports common backdoor capabilities, including file and process operations, and running shell commands, including ones via PowerShell. One of the commands stands out, with the rather uncommon capability of being able to run Lua code.

The full list of supported commands, with additional details, is shown in Table 4.

Table 4. Overview of LunarWeb commands

Type	Command	Details
0	Run shell commands via a BAT file and get output	Runs specified shell commands via a temporary BAT file %TEMP%\<random_9_alnum_chars>.bat. The output is retrieved via a pipe (also applies to the next four commands).
1	Run shell commands and get Unicode output	Runs the shell commands on the command line via cmd.exe /c and /U option for Unicode output.
2	Run shell commands and get output	Runs the shell commands on the command line via cmd.exe /c.
3	Run PowerShell commands via a PS1 file and get output	Runs specified PowerShell commands via a temporary script file %TEMP%\<random_12_alnum_chars>.ps1.
4	Run PowerShell commands and get output	Runs specified PowerShell commands via powershell.exe -command.
5	Run Lua code	Lua code is executed using the statically linked LuaCOM library and the Lua library, version 5.1.5. These libraries, along with the command, were not present in the single 32-bit version of the LunarWeb backdoor that we observed.
6	Write file	Specifies the file path and content to write.
7	Read file	Uses file mapping to access content instead of the regular ReadFile API.
8	Get victim identification via WMI	Obtains victim identification information using WMI queries, the same information as described in the *Information collection* section.
9	No operation	N/A
10	Update state entry in third slot	Updates an entry in the state used by the backdoor (index 2), adjusting break duration before communication loop and after C&C contact failure..
11	Set state content in first slot	Sets the content of the state in the first slot (index 0), but its purpose is unknown.
12	Set state content in second slot	Sets the content of the state in the second slot (index 1), but its purpose is unknown.
13	Create process and get output	Creates an arbitrary process with a specified command line and retrieves its output via a pipe.
14	Zip specified path(s)	Creates a ZIP archive with specified files and directories, via the statically linked Zipper library.

Some of the commands can output an error message referring to the commands as tasks – Format of the task is incorrect.

We were able to recover a command package that contained multiple shell commands used for reconnaissance executed via command 1, collecting the following: system and OS Information, user information, network configuration and connections, environment variables, scheduled tasks, installed programs and security products, firewall settings, directory listings, Kerberos tickets and sessions, shared resources, Group Policy, and local group memberships. Additionally, a read file command (7) was used to retrieve Zabbix configuration from a specified file path.

Stage 2 payload #2 – LunarMail backdoor

The second backdoor, which we call LunarMail, shares many similarities with LunarWeb. The main difference is the communication method – LunarMail uses email for communication with its C&C server.

This backdoor is designed to be deployed on user workstations, not servers – because it is persisted and intended to run as an Outlook add-in. A high-level overview of how LunarMail operates is shown in Figure 5.

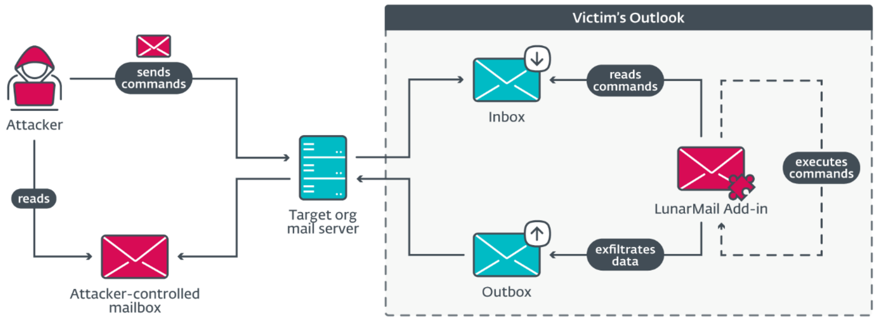

Figure 5. LunarMail operation

LunarMail shares ideas of its operation with LightNeuron, another Turla backdoor that uses email messages for C&C purposes. Although both use a similar exfiltration method, we did not find any code similarities between the two backdoors. Other Turla backdoors with similar operation include Outlook backdoor.

Initialization

During its initialization, the backdoor decrypts a string used to initialize a regex object that is used as a filter to search for the email profile to use for C&C purposes, which we describe later. The regex expression, and other strings in the backdoor, are encrypted using RC4 with the static key E3 7C 9E B0 DF D1 46 48 B4 AE 8A 5F 2A A1 78 7B.

To interact with Outlook, the backdoor dynamically resolves the necessary Outlook Messaging API (MAPI) functions.

On each run, the backdoor creates a directory in the path %TEMP%\{<random_guid>}, used as a staging directory for data exfiltration.

Configuration and state

Similar to LunarWeb, LunarMail's configuration entries are hardcoded in the binary. It also maintains a state file, with a single state (unlike LunarWeb, which has multiple state slots).

The configuration likely consists of conditions to find an Outlook profile for C&C communications, default exfiltration configuration, and the backdoor's lifespan limit.

The state is persisted in the file %LOCALAPPDATA%\Microsoft\Outlook\outlk.share with a 668-byte structure, updated during execution. It stores, among others, a timestamp of the last executed command and current staging directory. On subsequent runs, the previous staging directory is deleted and replaced with a new one.

Information collection

On first run, the LunarMail backdoor collects the following information:

- environment variables, and
- recipients of all sent email messages (email addresses).

Additionally, a batch file with shell commands to obtain further system information is decrypted but never executed.

In certain error cases, such as failure to collect the aforementioned information, the email addresses of available Outlook profiles are collected.

Communication and commands

Running inside Outlook, the LunarMail backdoor communicates with its C&C server – receiving commands and exfiltrating data – using email messages, via the Outlook Messaging API (MAPI).

Profile search

To communicate, LunarMail first searches for suitable Outlook profiles provided by Microsoft Exchange. The profile conditions include having only four default folders (Inbox, Sent, Deleted, and Outbox), containing the domain of the targeted institution in the email address, and not matching a regex pattern for various legitimate institutional emails.

The first matching profile sends initial information. For further communication, the inboxes of profile candidates are searched for command-containing emails. This approach avoids hardcoding profiles and makes identification harder. Additionally, commands can set a specific profile to use, which is persisted in the backdoor's state.

Receiving commands

LunarMail identifies a profile with commands by searching email messages and attempting to parse their attachments. The attachment must be a single PNG image with the .png extension, with the size of less than or equal to 10 MB. It then attempts to parse IDAT chunks of the PNG file, looking for an AES seed, an exfiltration configuration, and commands chunks. All these components are zlib-compressed and encrypted, the first using RSA-4096 and the latter two using AES.

Interestingly, the chunks must adhere to the PNG format with verified CRCs, resulting in a valid, but noisy-looking image due to encrypted, compressed content.

LunarMail uses the same cryptography as LunarWeb, including the Mbed TLS library, two RSA-4096 keys (listed in our GitHub repository), and usage of AES-256 with the same key derivation algorithm. The decompressed chunk with AES-encrypted content has a similar structure to that seen in LunarWeb.

The decrypted, decompressed exfiltration configuration has a specific structure including configuration ID, email address, subject, body, and attachment name and extension.

The exfiltration configuration structure mirrors LunarWeb's command package metadata, specifying the command outputs' destination and an ID to avoid duplicate commands, stored in state. Once decrypted and decompressed, LunarMail commands have a structure identical to LunarWeb's. Each parsed command is executed, storing output in the staging directory for exfiltration.

Notably, email messages that fail parsing for commands have their IDs cached to avoid repeated parsing, although the cache is not persisted and it is recreated on each backdoor execution. Emails successfully parsed for commands are deleted after processing.

Commands

In terms of command capabilities, LunarMail is simpler and features a subset of the commands found in LunarWeb. It can write a file, create a new process, and uniquely, take a screenshot and modify the C&C communication email address. While LunarMail lacks separate commands for running shell or PowerShell commands, it does support Lua scripts. When executed, commands write their output to files in the staging directory.

The full list of supported commands is shown in Table 5.

Table 5. Overview of LunarMail commands

Type	Command	Details
0	No operation	N/A
1	Write file	Specifies the file path and content to write.
2	Set the email address used for C&C communications	Sets a specific Outlook profile to use for C&C communications. The profile is specified by an email address, which is then persisted in the backdoor's state.
3	Create process and get output	Creates an arbitrary process with a specified command line, redirecting its output to the staging directory.
4	Take a screenshot	Utilizes GDI+ API to capture the entire screen and produce a JPG file.
5	Run Lua code	Just as in LunarWeb, Lua code is executed using the statically linked libraries LuaCOM and Lua.

Exfiltrating data – preparation

LunarMail searches its staging directory for output files produced by the backdoor, preparing them for exfiltration by embedding them in a PNG image or PDF document (depending on the attachment extension in the exfiltration configuration). Both PNG and PDF files are created using a valid content template.

For PNG files, a template matching the compromised institution's logo is used, indicating prior knowledge and preparation of the backdoor. To create a PNG that contains output files, LunarMail first generates a random 32-byte AES seed, used for encryption. Then it creates IDAT chunks with data and appends them to the PNG template. The chunks are similar to received commands, containing the following:

- Chunk with AES seed – RSA-4096 encrypted, zlib compressed.
- Chunk(s) with filename and content – AES-256 encrypted, zlib compressed.

Before compression and encryption, the output file name and content are wrapped into a structure that also contains a magic string 001035 that could be the backdoor version. Just like in received command PNGs, the created chunks follow the PNG specification and have their CRC checksum calculated, ensuring a valid image. To finalize the image, the IEND footer chunk is appended.

The second method, producing a PDF file, uses an encrypted template from the file %TEMP%\l4_mgrT.tmp. We have not observed this data file and the template's content is unknown, but probably it is a benign, unsuspicious document.

The output files with metadata are inserted at the end of the last stream in the PDF template, before the terminating endstream keyword. They are inserted in the following format and order:

1. Output files – variable sized, zlib compressed, AES-256 encrypted.
2. Metadata – fixed size (512 bytes), RSA-4096 encrypted.

The output filename and content are wrapped into the same structure as with the PNG, including the magic string, which is then compressed and encrypted.

The metadata contains information necessary for parsing and decrypting the structures of output files, including AES seed and output file positions in the PDF file.

After processing and embedding in the PNG or PDF file, files staged for exfiltration are deleted. The created file temporarily resides in the staging directory until exfiltration.

Exfiltrating data – transmission

Prepared PNG images or PDF documents containing output files are transmitted as attachments in emails to an attacker-controlled inbox, as per the exfiltration configuration. The default LunarMail setup includes a specific recipient email, subject header, message body, and attachment filename. The email content, although in the language of the compromised European MFA, appears machine translated due to its unnatural phrasing.

An exfiltration configuration from a received command overrides the default one. We have not recovered any commands so don't know if different email recipients, subjects, bodies, or attachment name or types are used across multiple commands.

If supported, the email body uses HTML format. The PNG is embedded as an image in the body, unlike the PDF. Figure 6 shows an illustration of an exfiltration email based on the default configuration. The email was translated, redacted, and the logo was changed by ESET Research, to not reveal the compromised institution.

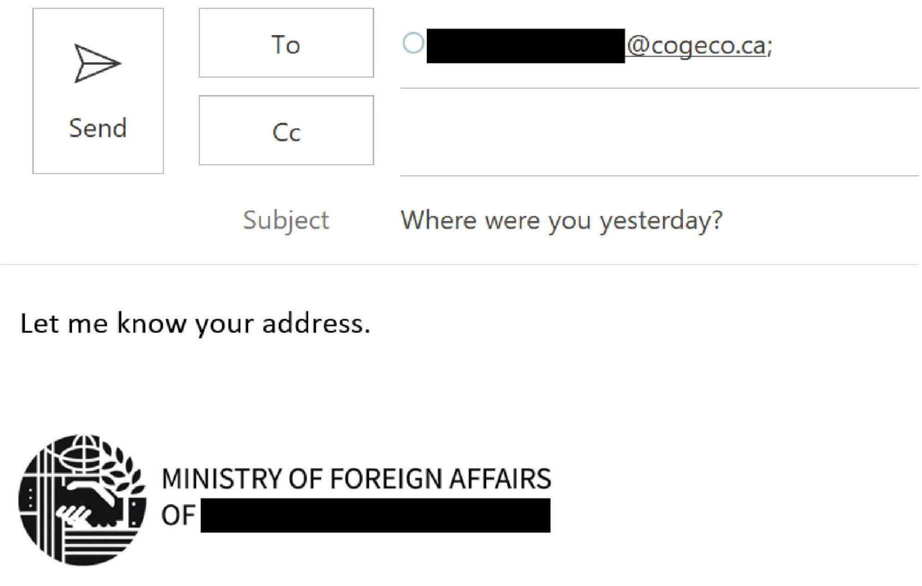

Figure 6. Illustration of an exfiltration email with data hidden in the image

Exfiltration email messages are sent with the PR_DELETE_AFTER_SUBMIT flag. In addition, any sent messages to the exfiltration address are deleted.

Conclusion

We have described two previously unknown backdoors used in compromises of a European government's institutions, which we attribute with medium confidence to the Russia-aligned APT group Turla.

The backdoors share a loader, bear code overlaps, and support similar commands, but they adopt different C&C communication methods. The first backdoor – LunarWeb – uses HTTP(S) and attempts to blend in by mimicking the traffic of legitimate services such as Windows Update. The second backdoor – LunarMail – piggybacks on Outlook and communicates via email messages, using either PNG images or PDF documents to exfiltrate data.

We observed varying degrees of sophistication in the compromises; for example, the careful installation on the compromised server to avoid scanning by security software contrasted with coding errors and different coding styles (which are not the scope of this blogpost) in the backdoors. This suggests multiple individuals were likely involved in the development and operation of these tools.

Although the described compromises are more recent, our findings show that these backdoors evaded detection for a more extended period and have been in use since at least 2020, based on artifacts found in the Lunar toolset.

IoCs

A comprehensive list of IoCs and samples can be found in our GitHub repository.

Files

SHA-1	Filename	Detection	Description
DE83C2C3FE68CB1BF96173E9EE3EA6161DCFB24A	App_Web_0bm4blbr.dll	MSIL/Agent.ERT	Compiled version of ASP.NET web page that installs LunarWeb.
9CEC3972FA35C88DE87BD66950E18B3E0A6DF77C	N/A	VBA/TrojanDownloader.Agent.ZJC	Malicious Word macro that installs LunarMail.
2ED792E39F7D56DE52BDF4AED96AFC898478BFDF	gpgol.dll	Win64/LunarLoader.B	LunarLoader (x64) used to load LunarMail.
F09E36553E48EBD42E60D9B25A390C0F57FF8DE0	gpgol.dll	Win32/LunarLoader.A	LunarLoader (x86) used to load LunarMail.
94A4CE9C75BC847E7BE59B96C4133D677D909414	tapiperf.dll	Win64/LunarLoader.C	LunarLoader (x64) used to load LunarWeb.
00006B30806F915911349D82BEEB1AEB9025ADB4	admpwd.dll	Win64/LunarLoader.A	LunarLoader (x64); a trojanized AdmPwd, used to load LunarWeb.
19D86CF2ED82EAE23E019706FAE8DAFC60552E85	AdmPwd.dll	Win64/LunarLoader.A	LunarLoader (x64); a trojanized AdmPwd, used to load LunarWeb.
795C4127D42FE8DFAF4510B406B52BA5BEDE8D3A	winnet.dll.mui	Win64/LunarLoader.B	LunarLoader (x64) used to load LunarWeb.
754FB657156643FD09A68EC9FC124528578CAB0C	N/A	Win32/LunarWeb.A	LunarWeb backdoor (x86).
FCAE66F6D95C78DC829688CC0F4C39BB5A57828B	N/A	Win64/LunarMail.A	LunarMail backdoor (x64).
67C6AEC8D129E610378EF52F8BF934886587932F	N/A	Win32/LunarMail.A	LunarMail backdoor (x86).
4C84110F1B10DF5FDD612759E210E44B0F0505EF	N/A	Win64/LunarWeb.A	LunarWeb backdoor (x64).
5D3975E57BDCB630A00FEBE5D405EEFB6D119D86	N/A	Win64/LunarWeb.A	LunarWeb backdoor (x64).
5EF771AFC96C24371D367448627609CFACB34A57	N/A	Win64/LunarWeb.A	LunarWeb backdoor (x64).
512E4FA7D6119270FF44A3B2A2359EE8825392EF	N/A	Win64/LunarWeb.A	LunarWeb backdoor (x64).

File paths

Stage 2 blob

C:\Windows\System32\DynamicAuth.bin

C:\Program Files\LAPS\CSE\admpwd.cache

C:\ProgramData\Microsoft\WinThumb\adcache.clb

C:\Windows\System32\perfcache.dat

%USERPROFILE%\Gpg4win\tempkeys.dat

LunarWeb state file

C:\ProgramData\Microsoft\Windows\Templates\content.tpl

C:\ProgramData\Microsoft\WinThumb\thumb.clb

C:\ProgramData\Microsoft\WinThumb\cfcache.clb

C:\Windows\System32\perfconfm.dat

LunarMail state file

%LOCALAPPDATA%\Microsoft\Outlook\outlk.share

Network

IP	Domain	Hosting provider	First seen	Details
N/A	thedarktower.av.master.dns-cloud[.]net	N/A	2020-02-01	Domain (Free DNS) pinged by malicious Word macro.
45.33.24[.]145	N/A	Akamai Connected Cloud	2020-05-20	C&C server of LunarWeb (compromised VPS).
45.79.93[.]87	N/A	Akamai Connected Cloud	2020-05-20	C&C server of LunarWeb (compromised VPS).
65.109.179[.]67	N/A	Hetzner Online GmbH	2023-10-29	C&C server of LunarWeb (compromised VPS).
74.50.80[.]35	N/A	Host Department NJ, LLC	2023-10-29	C&C server of LunarWeb.
82.165.158[.]86	N/A	IONOS SE	2022-08-03	C&C server of LunarWeb (compromised VPS).
82.223.55[.]220	N/A	IONOS SE	2022-08-03	C&C server of LunarWeb (compromised VPS).
139.162.23[.]113	N/A	Akamai Connected Cloud	2023-06-15	C&C server of LunarWeb (compromised VPS).
158.220.102[.]80	N/A	Contabo GmbH	2023-10-29	C&C server of LunarWeb.
161.97.74[.]237	N/A	Contabo GmbH	2023-06-15	C&C server of LunarWeb.
176.57.150[.]252	N/A	Contabo GmbH	2023-06-15	C&C server of LunarWeb.
212.57.35[.]174	N/A	Webglobe, a.s.	2023-06-02	C&C server of LunarWeb (compromised VPS).
212.57.35[.]176	N/A	Webglobe, a.s.	2023-06-02	C&C server of LunarWeb (compromised VPS).

Registry keys

HKCU\SOFTWARE\Classes\CLSID\{3115036B-547E-4673-8479-EE54CD001B9D}\

MITRE ATT&CK techniques

This table was built using version 15 of the MITRE ATT&CK framework.

Tactic	ID	Name	Description
Reconnaissance	T1591	Gather Victim Org Information	LunarMail's communication method indicates prior knowledge about compromised institutions.
Resource Development	T1583.002	Acquire Infrastructure: DNS Server	Stage 0 macro pings a domain from free DNS hosting provided by ClouDNS.
	T1583.003	Acquire Infrastructure: Virtual Private Server	Turla has used VPS hosting providers for C&C servers.
	T1584.003	Compromise Infrastructure: Virtual Private Server	Turla has used compromised VPSes for C&C purposes.
	T1586.002	Compromise Accounts: Email Accounts	Turla has used likely compromised email accounts for communication with the LunarMail backdoor.
	T1587.001	Develop Capabilities: Malware	Turla has developed custom malware, including loaders and backdoors.

Tactic	ID	Name	Description
Execution	T1047	Windows Management Instrumentation	LunarWeb obtains system information by using WMI queries.
	T1059	Command and Scripting Interpreter	LunarWeb and LunarMail can execute Lua scripts.
	T1059.001	Command and Scripting Interpreter: PowerShell	LunarWeb can execute PowerShell commands.
	T1059.003	Command and Scripting Interpreter: Windows Command Shell	LunarWeb can execute shell commands via cmd.exe.
	T1059.005	Command and Scripting Interpreter: Visual Basic	Stage 0 Word document contains a VBA macro.
	T1106	Native API	LunarWeb and LunarMail use various Windows APIs.
	T1204.002	User Execution: Malicious File	Stage 0 Word document with malicious macro must be opened by victim.
Persistence	T1137.006	Office Application Startup: Add-ins	LunarMail loader is persisted as an Outlook add-in.
	T1547	Boot or Logon Autostart Execution	A LunarWeb loader is persisted as a Group Policy extension.
	T1574	Hijack Execution Flow	A LunarWeb loader is persisted by replacing the system DLL tapiperf.dll.
Defense Evasion	T1027	Obfuscated Files or Information	LunarWeb and LunarMail are AES-256 encrypted on disk.
	T1027.003	Obfuscated Files or Information: Steganography	LunarMail stages exfiltration data into a PNG image or PDF document.
	T1027.007	Obfuscated Files or Information: Dynamic API Resolution	LunarMail dynamically resolves MAPI functions.
	T1027.009	Obfuscated Files or Information: Embedded Payloads	LunarMail installer has payloads embedded in a DOCX format document.
	T1036.005	Masquerading: Match Legitimate Name or Location	Filenames used by LunarWeb and LunarMail loading chains mimic legitimate files.
	T1070.004	Indicator Removal: File Deletion	LunarWeb and LunarMail can uninstall themselves by deleting their loading chain.
	T1070.008	Indicator Removal: Clear Mailbox Data	LunarMail deletes email messages used for C&C communications.
	T1140	Deobfuscate/Decode Files or Information	LunarWeb and LunarMail decrypt their strings using RC4.
	T1480.001	Execution Guardrails: Environmental Keying	LunarLoader decrypts its payload using a key derived from the DNS domain name.
	T1620	Reflective Code Loading	LunarWeb and LunarMail are executed using a reflective loader.
Discovery	T1007	System Service Discovery	LunarWeb retrieves a list of services.
	T1016	System Network Configuration Discovery	LunarWeb retrieves network adapter information.
	T1057	Process Discovery	LunarWeb retrieves a list of running processes.
	T1082	System Information Discovery	LunarWeb retrieves system information such as OS version, BIOS version, domain name, and environment variables. LunarMail retrieves environment variables.
	T1518.001	Software Discovery: Security Software Discovery	LunarWeb discovers installed security solutions via the WMI query wmic /Namespace:\\root\SecurityCenter2 Path AntiVirusProduct Get *.
Collection	T1005	Data from Local System	LunarWeb and LunarMail can upload files from the compromised machine.
	T1074.001	Data Staged: Local Data Staging	LunarMail stages data in a directory in %TEMP%.
	T1113	Screen Capture	LunarMail can capture screenshots.

Tactic	ID	Name	Description
	T1114.001	Email Collection: Local Email Collection	LunarMail collects recipients of sent email messages and can collect email addresses of Outlook profiles.
	T1560.002	Archive Collected Data: Archive via Library	LunarWeb and LunarMail use a statically linked zlib library for compression of collected data.
Command and Control	T1001.002	Data Obfuscation: Steganography	LunarWeb can receive commands hidden in JPG or GIF images. LunarMail receives commands hidden in PNG images and exfiltrates data hidden in PNG images or PDF documents.
	T1001.003	Data Obfuscation: Protocol Impersonation	LunarWeb impersonates legitimate domains in C&C communications by using a fake Host header and known URIs.
	T1071.001	Application Layer Protocol: Web Protocols	LunarWeb uses HTTP for C&C communications.
	T1071.003	Application Layer Protocol: Mail Protocols	LunarMail uses email messages for C&C communications.
	T1090.001	Proxy: Internal Proxy	LunarWeb can use an HTTP proxy for C&C communications.
	T1095	Non-Application Layer Protocol	Stage 0 macro pings the C&C server, utilizing ICMP protocol.
	T1132.001	Data Encoding: Standard Encoding	LunarWeb may receive base64-encoded data from the C&C server.
	T1573.001	Encrypted Channel: Symmetric Cryptography	LunarWeb and LunarMail encrypt C&C communications using AES-256.
	T1573.002	Encrypted Channel: Asymmetric Cryptography	LunarWeb and LunarMail encrypt the AES key used in C&C communications using RSA-4096.
Exfiltration	T1020	Automated Exfiltration	LunarWeb and LunarMail automatically exfiltrate collected data to the C&C server.
	T1030	Data Transfer Size Limits	LunarWeb splits exfiltrated data above 1.33 MB into multiple smaller chunks. LunarMail limits the size of email attachments containing exfiltrated data.
	T1041	Exfiltration Over C2 Channel	LunarWeb and LunarMail exfiltrate data over the C&C channel.