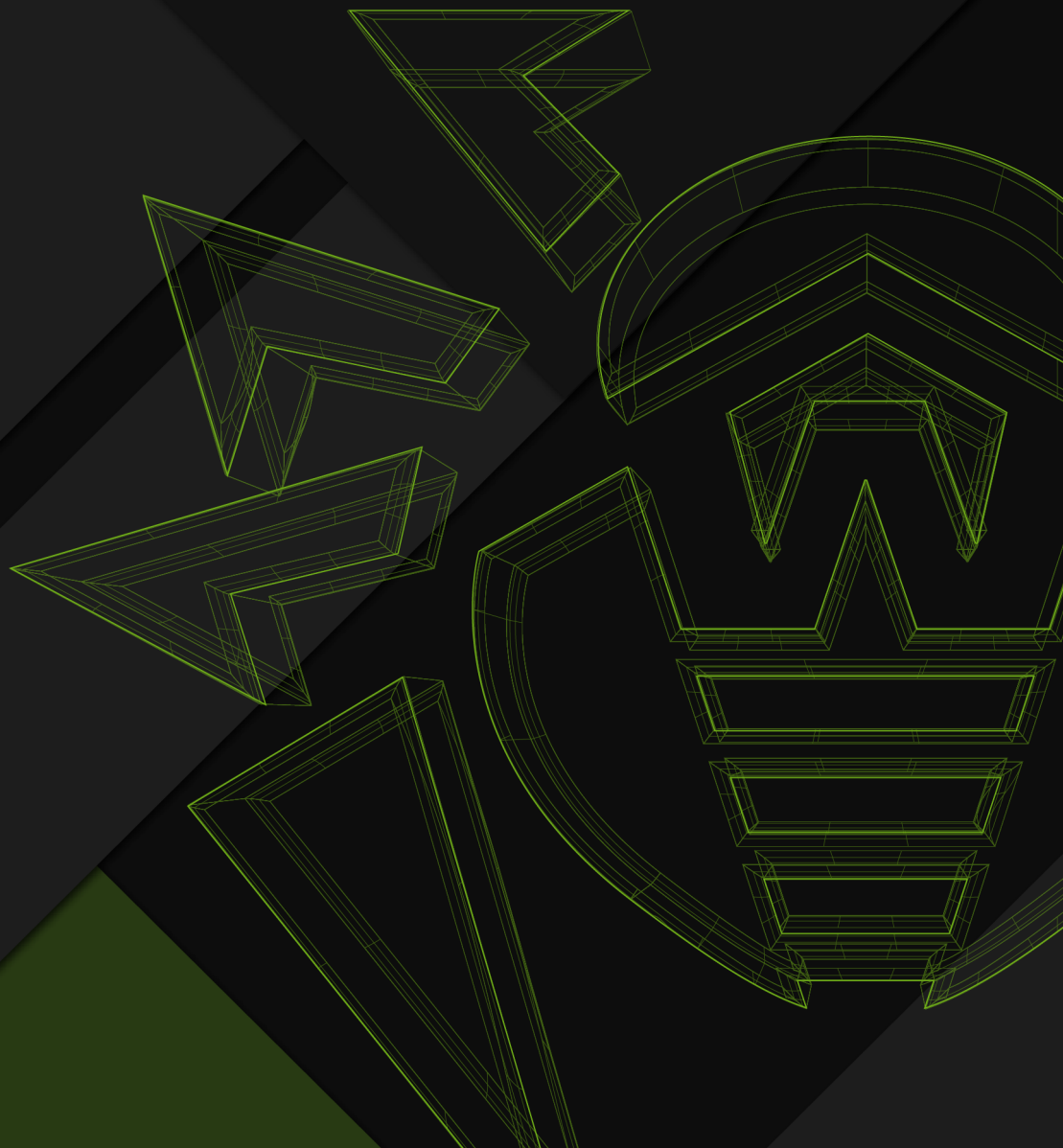




Study of a targeted attack on a Russian rail freight operator



© **Doctor Web, Ltd., 2024. All rights reserved.**

This document is the property of Doctor Web, Ltd. (hereinafter - Doctor Web). No part of this document may be reproduced, published or transmitted in any form or by any means for any purpose other than the purchaser's personal use without proper attribution.

Doctor Web develops and distributes Dr.Web information security solutions which provide efficient protection from malicious software and spam.

Doctor Web customers can be found among home users from all over the world and in government enterprises, small companies and nationwide corporations.

Dr.Web antivirus solutions are well known since 1992 for continuing excellence in malware detection and compliance with international information security standards. State certificates and awards received by the Dr.Web solutions, as well as the globally widespread use of our products are the best evidence of exceptional trust to the company products.

**Study of a targeted attack on a Russian rail freight operator
9/3/2024**

Doctor Web Head Office
2-12A, 3rd str. Yamskogo polya, Moscow, Russia, 125124

Website: www.drweb.com
Phone: +7 (495) 789-45-87

Refer to the official website for regional and international office information.

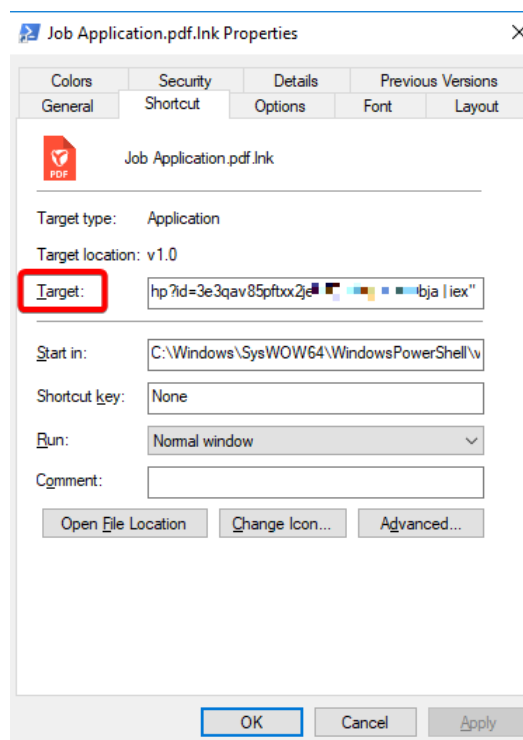
Introduction

Spear phishing is a popular method of delivering malware to computers in large organizations. It differs from regular phishing in that the attackers gather information in advance and personalize the message they send to encourage the victim to perform an action that will result in a security breach. The primary targets are either high-level employees with access to valuable information, or employees in departments that interact with multiple recipients. This is especially true for HR staff who receive many emails from strangers that have attachments in a variety of formats. This vector was chosen by the threat actors to attack a major freight operator in March 2024.

General Information About the Attack and the Tools Involved

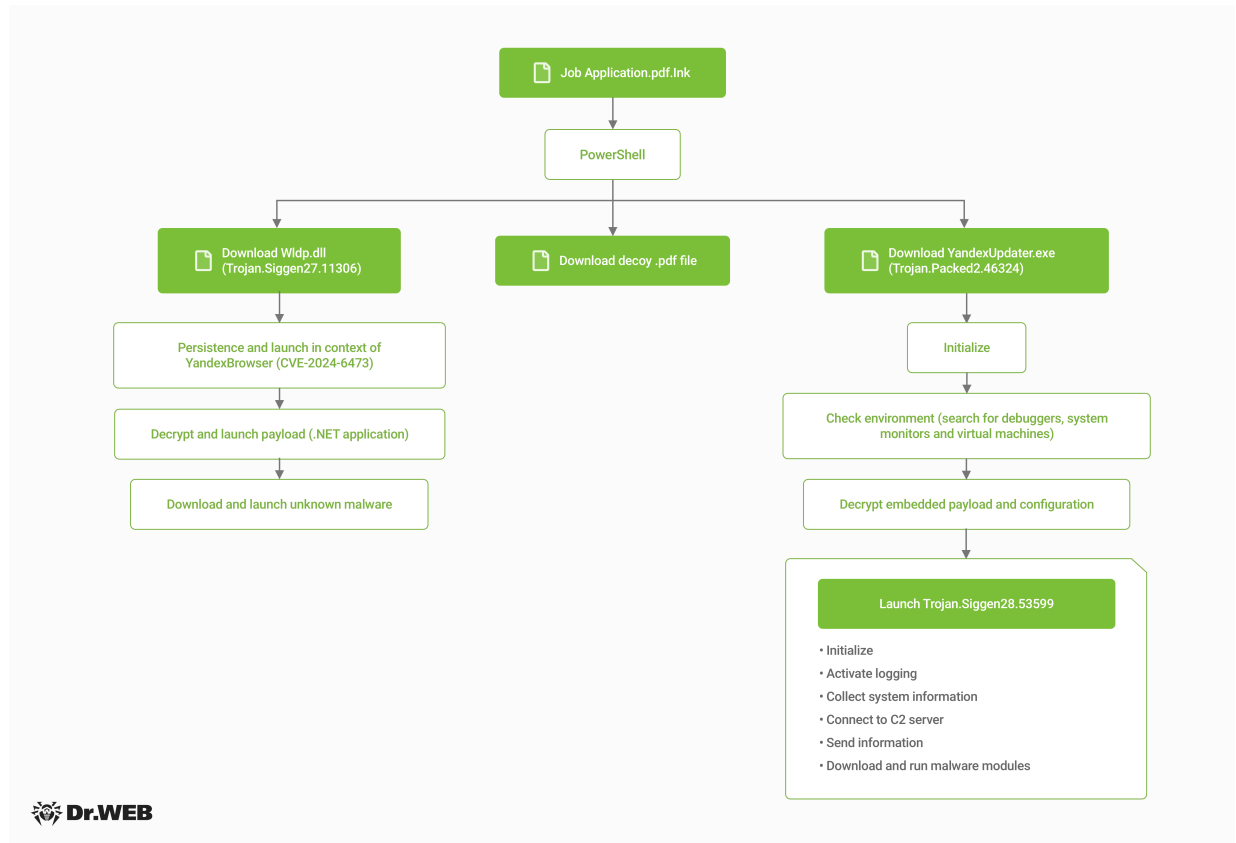
In March 2024, a large Russian company in the rail freight industry contacted Doctor Web. A suspicious email with an attachment caught the attention of their information security department. After trying to determine the threat posed by the attached file, they contacted our specialists. After reviewing the request, our analysts concluded that the company had almost been the victim of spear phishing. The goal of the perpetrators was to gather system information and launch modular malware on a compromised PC.

To carry out the attack, the criminals sent a phishing email disguised as a jobseeker's résumé to the company's email address. Attached to the email was an archive purporting to contain a PDF file with a job application. That file had the so-called "double" extension of `.pdf.lnk`. Hiding malicious objects by using double extensions is a common tactic employed by attackers to fool their victims. By default, Windows hides file extensions as a convenience to the user. And when a file has a "double" extension, the system only hides the last extension. In this case, the victim could see the first extension—`.pdf`, while the `.lnk` extension was hidden. Moreover, even if the display of full filenames is enabled, the `.lnk` extension is always hidden by the operating system.



Metadata stored in an Ink file

The real `.lnk` extension is an extension for shortcuts in Windows. In the Target field, you can specify the path to any operating system object, such as an executable file, and run it with the required parameters. This attack covertly launched the PowerShell command prompt, which downloaded from the attackers' website two malicious scripts, each of which launched its own payload.



Attack chain

The first was a decoy .pdf file and executable file called `YandexUpdater.exe`, which posed as a component for updating Yandex Browser (the name of the real component is `service_update.exe`). This executable is a malware dropper called **Trojan.Packed2.46324**, which, after conducting a series of checks to determine whether it is running in an emulated environment and whether debugging software is present, unpacks **Trojan.Siggen28.53599** on the compromised system. The latter has remote control capabilities, collects system information and downloads various malicious modules. In addition to these functions, the trojan also has anti-debugging capabilities. If antivirus, virtual machine and debugger processes are detected, the trojan overwrites its file with zeros and deletes it and the folder in which it was stored.



Клеблец Инна Федоровна

Женщина, 30 лет, родилась 10 марта 1994

+7 (952) 5704332

inna.kleblets@mail.ru — предпочитаемый способ связи

Проживает: г. Вологда

Гражданство: Россия

Готова к переезду, готова к командировкам. Не замужем

Желаемая должность и зарплата

Младший Frontend-разработчик (Ru)

Специализации:

— Python,SQL

55 000

руб.

Занятость: полная занятость

График работы: полный день

Желательное время в пути до работы: не имеет значения

Опыт работы — 5 лет 5 месяцев

Июль 2020 —

Ноябрь 2023

2 года 5 месяцев

“Универсал” ООО

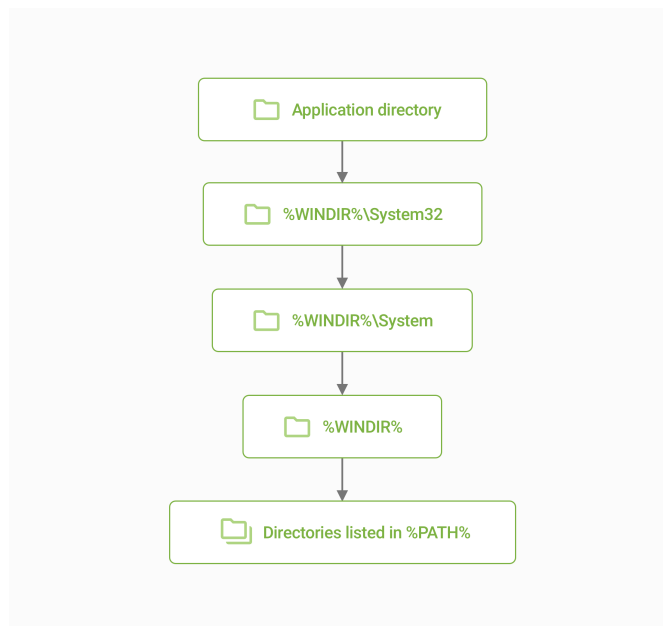
ООО «Универсал»

Младший Front end-разработчик

- Создание внешних оболочек сайта
- Обслуживание систем авторизации и платежей
- Интеграция графического контента
- Общение с клиентами в конечной фазе создание сайта

Decoy PDF file

The second payload consisted of a decoy PDF file and **Trojan.Siggen27.11306**. This trojan is a dynamic library (DLL) with an encrypted payload. A unique feature of this trojan is that it exploits the vulnerability of Yandex Browser to DLL Search Order Hijacking. In Windows, DLL files are libraries that applications use to store functions, variables and interface elements. When launched, the applications search for libraries in several data stores in a specific order, so attackers can try to “jump the queue” and place a malicious library in the folder where DLL searches are prioritized.



Simplified DLL search prioritization scheme

This trojan is stored in the hidden %LOCALAPPDATA%\Yandex\YandexBrowser\Application folder under the name wldp.dll. This is the directory where Yandex Browser is installed and where the browser looks for the libraries it needs at startup. In turn, the legitimate wldp.dll library, whose function is to ensure the security of application startup, is an OS system library and is located in the %WINDIR%\System32 folder. Since the malicious library is located in the Yandex Browser installation folder, it is loaded first. At the same time, it gets all the permissions of the main application: it can execute commands and create processes from the context of the browser, as well as inherit firewall rules for Internet access.

After the browser is launched, the malicious wldp.dll library decrypts the payload embedded in it. Note that the decryption is done twice. The first time it is done using a key generated from the hash of the path where the malicious DLL is located, and then using a global key embedded in the body of the trojan. The decryption results in shell code, the execution of which allows attackers to run an application, written in the .NET language, on the compromised system. This executable, in turn, downloads malware from the network. Unfortunately, at the time of our investigation, the server that the downloader was communicating with was down, and we were unable to determine what specific trojan was being downloaded in this case.

Having discovered this vulnerability in Yandex Browser, we submitted our findings to Yandex. The developers promptly released an updated version of Yandex Browser (24.7.1.380) where this vulnerability ([CVE-2024-6473](#)) is fixed.

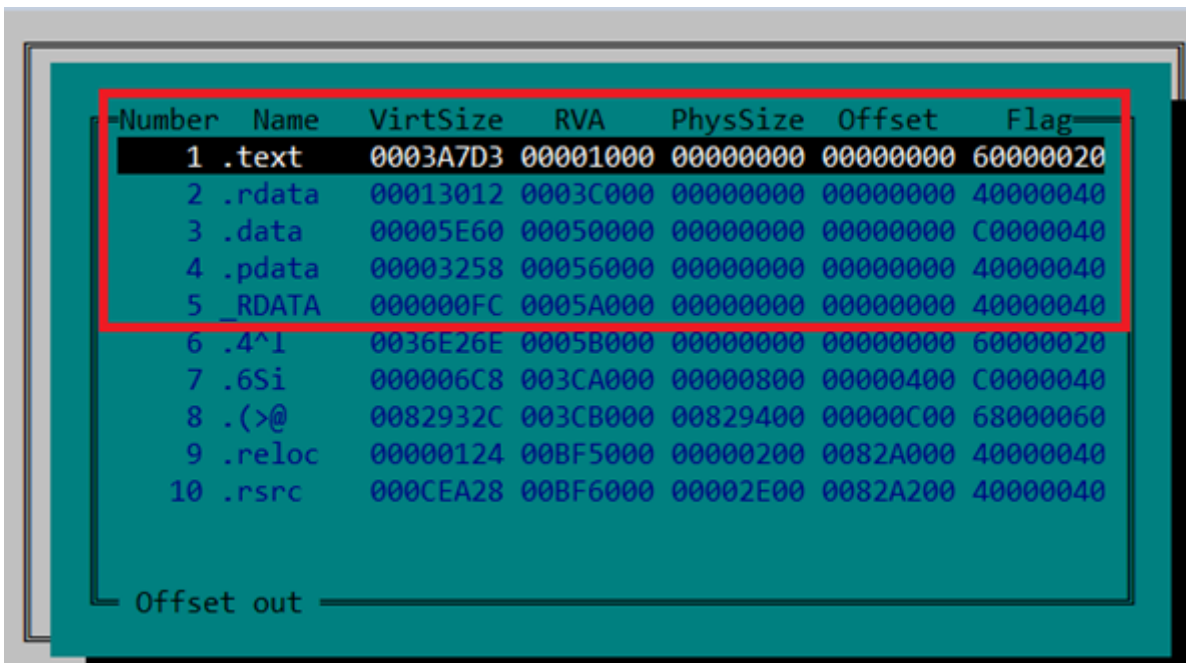
Operating Routine of Discovered Malware Samples

Trojan.Packed2.46324

A malicious Windows malware dropper program written in C++. The executable file is obfuscated with XOR encryption and packed with a custom packer. It is used to deliver **Trojan.Siggen28.53599** to compromised PCs.

Trojan unpacking

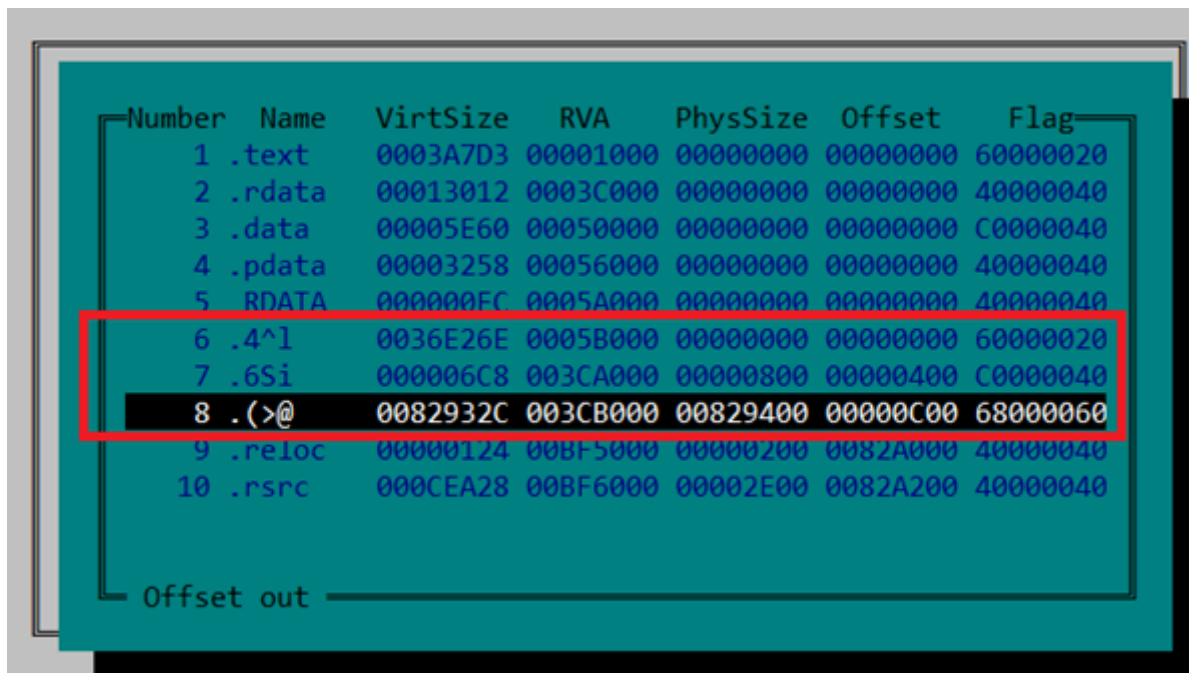
The trojan is packed using a custom packer that is characterized by the zero physical size of sections bearing "traditional" names such as ".text", ".rdata" and others.



Number	Name	VirtSize	RVA	PhysSize	Offset	Flag
1	.text	0003A7D3	00001000	00000000	00000000	60000020
2	.rdata	00013012	0003C000	00000000	00000000	40000040
3	.data	00005E60	00050000	00000000	00000000	C0000040
4	.pdata	00003258	00056000	00000000	00000000	40000040
5	._RDATA	000000FC	0005A000	00000000	00000000	40000040
6	.4^1	0036E26E	0005B000	00000000	00000000	60000020
7	.6Si	000006C8	003CA000	00000800	00000400	C0000040
8	.(>@	0082932C	003CB000	00829400	00000C00	68000060
9	.reloc	00000124	00BF5000	00000200	0082A000	40000040
10	.rsrc	000CEA28	00BF6000	00002E00	0082A200	40000040

Offset out _____

In this case, one of the following sections — ". (>@" — contains some code. It also contains the entry point.



Number	Name	VirtSize	RVA	PhysSize	Offset	Flag
1	.text	0003A7D3	00001000	00000000	00000000	60000020
2	.rdata	00013012	0003C000	00000000	00000000	40000040
3	.data	00005E60	00050000	00000000	00000000	C0000040
4	.pdata	00003258	00056000	00000000	00000000	40000040
5	.RDATA	000000FC	0005A000	00000000	00000000	40000040
6	.4^1	0036E26E	0005B000	00000000	00000000	60000020
7	.6Si	000006C8	003CA000	00000800	00000400	C0000040
8	.(>@	0082932C	003CB000	00829400	00000C00	68000060
9	.reloc	00000124	00BF5000	00000200	0082A000	40000040
10	.rsrc	000CEA28	00BF6000	00002E00	0082A200	40000040

Offset out

Once initialized, the source code containing the original entry point is extracted into empty sections.

Operating routine

Initialization

The trojan reads the `KUSER_SHARED_DATA` struct, which contains various system information. The trojan checks the value of this struct's `NtMajorVersion` field, which must equal 10.

It then initializes the struct containing the pointers to the `ntdll.dll`, `kernel32.dll`, `user32.dll`, `ole32.dll` libraries, which are used to call functions from this struct. Then the three threads responsible for anti-debugging are initialized. After that, the struct for working with the `wevtapi.dll` library is initialized and pointers to the libraries and various functions are obtained.

Use of WinAPI

The trojan uses WinAPI system functions via a wrapper struct that contains a table of functions, library pointers, library load addresses, and an anti-debugging flag.

The table contains the following functions:

- Functions for working with WinAPI, i.e., finding a function pointer and calling it
- Helper functions—ad hoc implementation of the `LoadLibrary` and `GetProcAddress` calls

- The configuration of input parameters for a range of functions

When launched, the trojan initializes its main struct. It does this by using a modified CRC32 algorithm to find library load addresses in the `PEB_LDR_DATA` system struct. The trojan uses two methods to access functions stored in the libraries:

- Ad hoc implementation of the `LoadLibrary` and `GetProcAddress` calls

The trojan has two functions that mimic the implementation of `LoadLibrary` and `GetProcAddress`. This method is used when the trojan needs access to an API contained in a library that has not yet been loaded into the process memory.

- Searching for libraries by their hashes in the `PEB_LDR_DATA` system struct

The trojan searches for a required library in the `PEB_LDR_DATA` struct using the `InMemoryOrderModuleList` list, which contains pointers to all the libraries loaded into the process memory and their names. The library name is matched by comparing the hash value generated using the modified CRC32 algorithm with the requisite library name. Next, the required library function is found in the table of exported library functions, in which case the function names are hashed in the same way. The library name and function are read using the modified CRC32 algorithm.

Debugger evasion

Checking the debug registers

The trojan obtains the context of the parent thread and checks that the values of the `Dr0-Dr7` registers are set to 0.

Checking for a debugged environment

In the `KUSER_SHARED_DATA` struct, the trojan checks the first two bits in the `KdDebuggerEnabled` field; the value of these bits must be set to 0.

Using the `NtQueryInformationProcess` function, the trojan checks for the presence of a debugger by reading various parameters of the `PROCESSINFOCLASS` struct: `ProcessDebugFlags`, `ProcessDebugPort`, `ProcessDebugObjectHandle`, `ProcessTlsInformation`.

Checking for debugger drivers

The trojan scans the `%WINDIR%\System32\drivers` directory for files that indicate the presence of debugging software. It then calculates the filename hash using the modified CRC32 algorithm and compares the result to the blacklist hashes.

Each check is timed, and if the check fails, a flag is set in a global variable that is checked at various stages of execution. If the flag check fails, the trojan terminates its process.

Environment check

To protect itself from running in a virtualized environment, the trojan scans a number of OS logs, using the `wevtapi.dll` library. It searches for the below strings:

In the logs `Microsoft-Windows-Shell-Core/Operational, System, Application`:

- `\npcap.sys`
- `Wireshark`
- `API_Monitor`
- `apimonitor`
- `API Monitor`
- `rohitab.com`
- `hex-rays.com`
- `processhacker.sys`
- `ProcessHacker`
- `PROCMON2`
- `ida64.exe`

In the logs `Microsoft-Windows-Storage-Storport/Operational, System, Microsoft-Windows-Storsvc/Diagnostic, Microsoft-Windows-StorageSpaces-Driver/Operational, Microsoft-Windows-Partition/Diagnostic, Microsoft-Windows-Kernel-PnP/Configuration, Application`:

- `VMTools`
- `VMUpgradeHelper`
- `VirtualBox Guest`
- `VBoxService.exe`
- `VBOX HARDDISK`
- `_FLOPPY_`
- `\VMWVM`
- `_VBOX&`
- `NECVMWar`
- `prl_`
- `VMware`

Additionally, in the Application log:

- `VMware Player`
- `VMware NAT Service`
- `\Device\VBoxNet`
- `Oracle VM VirtualBox`

After performing these checks, the dropper unpacks **Trojan.Siggen28.53599**, the payload contained in the resources. This is done using a modified RC4 algorithm; the key for the cipher is 8 bytes long. The dropper then decrypts the payload configuration, which is encrypted using the XOR cipher. The resulting configuration is stored in a string labeled with the characters "DANTEMARKER", which are overwritten.

After all the prep work is done, the dropper projects the payload into the memory and transfers control to it.

Trojan.Siggen28.53599

A malicious program for Windows written in C++. The main functionality of the trojan is to download and manage modules received from its C2 server.

Operating routine

The trojan has a number of basic and helper structs that are initialized at startup and stored as pointers in global variables.

Basic structs:

WinAPI wrapper

The trojan uses WinAPI system functions via a wrapper struct that contains a table of functions, library pointers, library load addresses, and an anti-debugging flag.

The table contains the following functions:

- Functions for working with WinAPI, i.e., for finding a function pointer and calling it
- Helper functions—the ad hoc implementation of the `LoadLibrary` and `GetProcAddress` calls
- The configuration of the input parameters for a range of functions

When launched, the trojan initializes its main struct. It does this by using a modified CRC32 algorithm to find library load addresses in the `PEB_LDR_DATA` system struct. The trojan uses two methods to access functions stored in the libraries:

- Ad hoc implementation of the `LoadLibrary` and `GetProcAddress` calls

The trojan has two functions that mimic the implementation of `LoadLibrary` and `GetProcAddress`. This method is used when the trojan needs access to an API contained in a library that has not yet been loaded into the process memory.

- Searching for libraries by their hashes in the `PEB_LDR_DATA` system struct

The trojan searches for a required library in the `PEB_LDR_DATA` struct, using the `InMemoryOrderModuleList` list, which contains pointers to all the libraries loaded into the process memory and their names. The library name is matched by comparing the hash value generated using the modified CRC32 algorithm with the requisite library name. Next, the required library function is found in the table of exported library functions, in which case the function names are hashed in the same way. The library name and function are read using the modified CRC32 algorithm.

Logger struct

This is a struct whose main purpose is to generate the application log. The log contains information about errors and the step currently being executed.

System information collector struct

The main purpose of this struct is to collect system information and send it to the C2 server.

C2 server communication struct

This struct ensures the interaction with the C2 server. It contains a struct for working with the `winhttp.dll` library and information about the control server: the port, IP address and routing table.

Module and configuration struct

The main function of this struct is to manage the operation of modules and their configurations. It contains vectors that describe the modules, their configuration, and auxiliary system information.

Manager struct

The main purpose of this struct is to control program operation and ensure interaction between other structs. It holds pointers to all the other primary structs: WinAPI wrapper, logger, communication, and configuration.

Helper structs:

Structs for working with cryptography: SHA-1, SHA-256

Structs for working with auxiliary libraries: `bcrypt.dll`, `winhttp.dll`

Structs for storing various flags

Debugger evasion

When launched, the trojan also initializes 3 threads to evade debuggers:

Checking the debug registers

The trojan obtains the context of the parent thread and checks that the values of the `Dr0-Dr7` registers are set to 0.

Checking for debugged environment

In the `KUSER_SHARED_DATA` struct, the trojan checks the first two bits in the `KdDebuggerEnabled` field; the value of these bits must be set to 0.

Using the `NtQueryInformationProcess` function, the trojan checks for the presence of a debugger by reading various parameters of the `PROCESSINFOCLASS` struct:

`ProcessDebugFlags`, `ProcessDebugPort`, `ProcessDebugObjectHandle`,
`ProcessTlsInformation`.

Checking for debugger drivers

The trojan scans the `%WINDIR%\System32\drivers` directory for files that indicate the presence of debugging software. It then calculates the filename hash, using the modified CRC32 algorithm, and compares the result to the blacklist hashes.

Checking for a copy of the trojan in the system

After initialization, the trojan attempts to create a mutex which is a Base64 SHA-1 encoded hash of the `MachineGuid` string value. If the attempt to capture the mutex fails, the line "Found another agent running. Exiting..." is written to the application log and the trojan is terminated.

Verifying keys and creating a handshake

The trojan checks for an existing handshake. This is done by accessing the key in the "Microsoft Software Key Storage Provider" CNG key storage, using the `NCryptOpenKey` function. The key name is a SHA-256 hash of the `MachineGuid` value. If such a key does not exist, the availability of the network connection is checked: if the connection is established, the trojan initiates the creation of the handshake:

- A copy of the RSA internal key from the "Microsoft Software Key Storage Provider" CNG key storage is created
- A key is received from the server
- This key is stored in the storage with a name corresponding to the SHA-256 hash of the `MachineGuid` value.

The trojan parses the incoming packet for a new C2 server address and port number. Once the handshake is established, the following system information is sent to the C2 server: CPU architecture, OS name, user interface type, installed application identifiers, disk information, user names, and locale.

Basic functions

The trojan performs the following actions:

- Loads and unloads modules
- Sends messages to the C2 server about its operation or errors
- Changes the configuration of modules
- Updates the trojan body, if necessary

Module structure and configuration

A module is a dynamic library that is projected into the memory and has the following exportable functions:

- Start
- Stop
- Configure
- GetID
- GetStatus
- SetStatus
- GetStarted
- GetHandler
- Destroy
- PushErrorCMR

The module identifier indicates its function, i.e., knowing the identifier, you can determine which tasks are sent by the C2 server.

Identifier	Purpose
238	Inject
27	Purpose unknown
44	Purpose unknown

JSON with module configuration

```
{
  "triggers": [
    {
      "schedule": "<str_value>",
      "process": "<str_value>",
      "repetitions": "<int_value>",
      "sendCmr": {
        "name": "<str_value>",
        "interval": "<int_value>"
      }
    }
  ]
}
```

Result of the module's execution

After the module completes its task, it generates a response

```
{
  "CommandModuleResponse": "<str_value>",
  "requestId": "<int_value>",
  "moduleId": "<int_value>",
  "exitCode": "<int_value>",
  "info": "Error" //this field is only shown if there was an error in the
module's operation; otherwise this field is missing
}
```

Trojan update

While in operation, the trojan checks the availability of its update flag. If this flag is set, the trojan performs a series of system checks, and, based on the results, selects one of the two update strategies. If antivirus software is detected on the compromised PC, the trojan is updated through the loading of shellcode; otherwise, it is updated using the Inject module.

Checking for antivirus software

The trojan body contains a list of hashes of antivirus program names. When scanning for antivirus software, the trojan obtains a list of processes and calculates the hashes of running applications, which are compared to the following hardcoded list:

- msmtpeng
- mssense
- avastsvc
- dwservice
- avp
- nortonsecurity
- coreserviceshell
- avguard
- fshoster32

- vsserv
- mbam
- adawareSERVICE
- avgsvc
- wrsa

Shellcode for directory removal

Input arguments:

- directory name

Actions performed:

- Looks up the `kernel32.dll` library address in the `PEB_LDR_DATA` struct
- Gets the functions for the shellcode's operation from the library export results, where the library name and function names are determined by their hashes
- Determines the path to the `%LOCALAPPDATA%\EROCS\` directory
- Overwrites with zeros and deletes all the files in the specified directory
- Deletes the directory itself

Shellcode for restarting the trojan

Actions performed:

- Gets the list of processes, using the `NtQuerySystemInformation` function (the `SystemProcessInformation` parameter); checks that the `UniqueProcessId` field is equal to `0x434F5245`
- Deletes the `HKEY_CURRENT_USER\Software\Uninstall` key

Self-deletion

The trojan also has a self-deletion function that is triggered when the "deadline" registry key, which is responsible for the trojan lifetime and is updated when a new command from the C2 server is received, is set to a specific value. The trojan also initiates the self-deletion procedure, using WinAPI if errors are detected during the above checks. In this case, it performs the following actions:

- Deletes its directory
- Deletes the handshake
- Deletes registry keys created while the trojan is in operation

Sending messages to the C2 server

The following User-Agent values are used by the trojan to send messages:

Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:88.0) Gecko/20100101
Firefox/88.0

Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:79.0) Gecko/20100101
Firefox/79.0

Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/90.0.4430.93 Safari/537.3

Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/80.0.3987.149 Safari/53

Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/79.0.3945.88 Safari/537.3

All outgoing and incoming messages are encrypted using RSA.

Trojan.Siggen27.11306

A Windows trojan written in C. It is a DLL with an encrypted payload.

Operating routine

During initialization, the trojan sequentially creates two threads: one to decrypt the data and the other to execute the payload. Initially, the payload is encrypted with a key that is the path to the executable. During the first run, the trojan rebuilds the executable and covers it with another layer of encryption. This encryption binds the payload to the infected PC. The preparation phase consists of the following steps:

- A random salt is generated and stored in a new trojan body at a specific offset
- BIOS information is obtained
- This information is hashed using the salt generated in step 1, and the resulting hash is the key to encrypt the payload
- The payload is encrypted using a "custom" key

After this transformation, the trojan has two decryption stages:

Stage 1: Decryption using constants from the compromised PC

- The salt stored in the trojan body is taken at a specific offset
- The salt is used to create a hash of the BIOS information
- The payload is decrypted

Stage 2: Decryption of the payload encrypted with the default key

- The `ImagePathName` value is extracted from the `RTL_USER_PROCESS_PARAMETERS` struct—this field is a Unicode string whose length must be greater than `0x76` bytes (in our case the filename was `%LOCALAPPDATA%\Yandex\YandexBrowser\Application\Wldp.dll`)
- The last `0x76` bytes are read from the above value
- The hash of this value, which is the key for the symmetric algorithm, is generated
- The payload is decrypted

Encryption algorithm

A modified ChaCha20 algorithm is used as the symmetric encryption algorithm. The modification consists of an additional layer for key initialization: the input key undergoes one round of the algorithm, after which it becomes the key for the regular algorithm.

Hashing algorithm

A modified BLAKE2 algorithm is used as the hash function. The modification is that multiple repetitive hashes of the input data are used.

Payload

The payload is a shellcode generated using <https://github.com/TheWover/donut/tree/master>. This shellcode decrypts and downloads an MZPE file written in .NET, the main purpose of which is to launch a trojan downloaded from the Internet. The main body of the shellcode can be found at https://github.com/TheWover/donut/blob/master/loader_exe_x64.h.

The shellcode performs the following actions:

- Checks the flag responsible for executing the load in a separate or main thread
- Decrypts the MZPE file into a new allocated memory area
- Loads the `ole32.dll`, `oleaut32.dll`, `wininet.dll`, `mscoree.dll`, and `shell32.dll` libraries, using the `LoadLibraryA` function
- Loads the `WldpQueryDynamicCodeTrust`, `WldpIsClassInApprovedList`, `EtwEventWrite` and `EtwEventUnregister` functions, using the `GetProcAddress` function
- Initializes the AMSI interface
 - Loads `amsi.dll`
 - Loads the `AmsiInitialize`, `AmsiScanBuffer` and `AmsiScanString` functions
- Reads the value of the AMSI bypass flag; this flag is not set in this sample
- Downloads the .NET application

The .NET stager downloads other malware, saves it under the name `"YandexUpdater.exe"` and then launches it. At the time of our investigation, the file was no longer available on the server from which the malware was supposed to be downloaded, so we were unable to positively identify the downloaded software; however, we can assume that the file in question could be the same **Trojan.Packed2.46324**.

Conclusion

Thus, we see a multi-vector, multi-stage infection scheme with two different trojans that are delivered to a compromised system when a file from a phishing email is opened. Despite the complexity of the implementation, preventing and protecting against such attacks is quite simple:

- Raise employee awareness of information security issues (carefully check links and filenames, and do not open suspicious objects).
- Use software products that perform email filtering, such as [Dr.Web Mail Security Suite](#), to prevent the delivery of malicious emails and attachments.
- Install antivirus software, such as [Dr.Web Desktop Security Suite](#) and [Dr.Web Server Security Suite](#), on all network nodes, which will prevent a dangerous file from getting through when users are working on the Internet or block suspicious activities on user computers if a file was delivered on a USB drive.
- Regularly apply software updates that fix program bugs.

Appendix 1. Indicators of Compromise

SHA1 hashes

Trojan.Packed2.46324

34a4c5f28c7df23662962c3eaa0a15b7ae48b488: YandexUpdater.exe

Trojan.Siggen27.11306

60eaa4fd53b78227760864e6cf27b08bc4bdde72: Wldp.dll

Trojan.Siggen28.53599

853d6a17f0a1a4035b52699a447eeb4ad1ca6cf7

File artifacts

Job Application_202402523.rar (пароль: Инна)

Job Application.pdf.lnk

102fa066-cc9d-4a80-b3aa-12d5df196b42.pdf

Domains

infosecteam[.]info

IP addresses

109.248.147[.]132