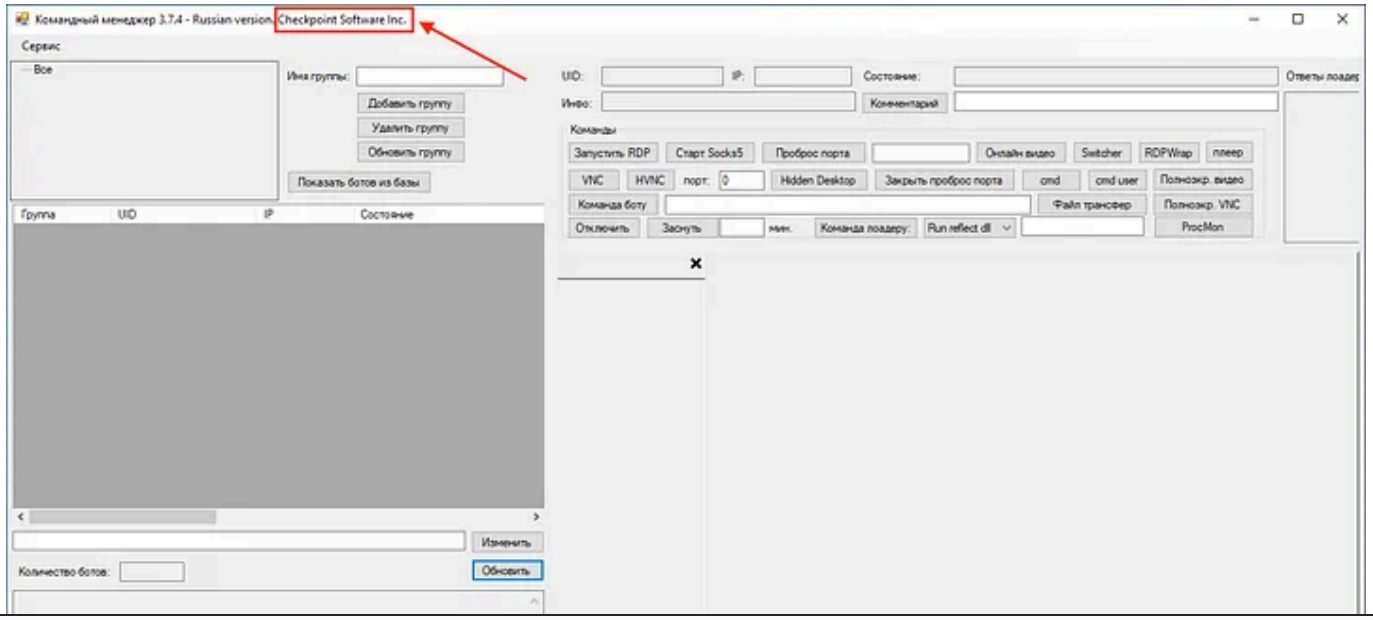
From pentest to APT attack: cybercriminal group FIN7 disguises its malware as an ethical hacker's

known as Tirion), a toolkit for reconnaissance and getting a foothold inside infected systems. Disguised as a legitimate cybersecurity company, the group distributes Lizar as a pentesting tool for Windows networks. This caught our attention and we did some research, the results of which we will share in this article.

A few words about FIN7

The APT group FIN7 was presumably founded back in 2013, but we will focus on its activities starting from 2020: that's when cybercriminals focused



Component name	Component description	Component activity
Lizar client	GUI software that FIN7 members use to control loaders on infected devices. It was designed to run on Windows OS	The software communicates with the server, sends commands to the loader on the infected machine (the loader) through the server and receives the result of the commands
Lizar server	The software that enables communication between the client and the loader	The software runs on a remote server
Lizar loader	Loader designed for downloading plug-ins	The loader communicates with the server and runs the necessary plug-ins on command from the server
Lizar plugins	Server-side plug-ins	The result of each plug-in is sent to the server and from the server to the client
Lizar plugins/extra	Client-side plug-ins	Plugins from the `plugins/extra` directory are transferred from the client to the server, then from the server to the loader (on the infected system)

Lizar client

Lizar client consistes of the following components:

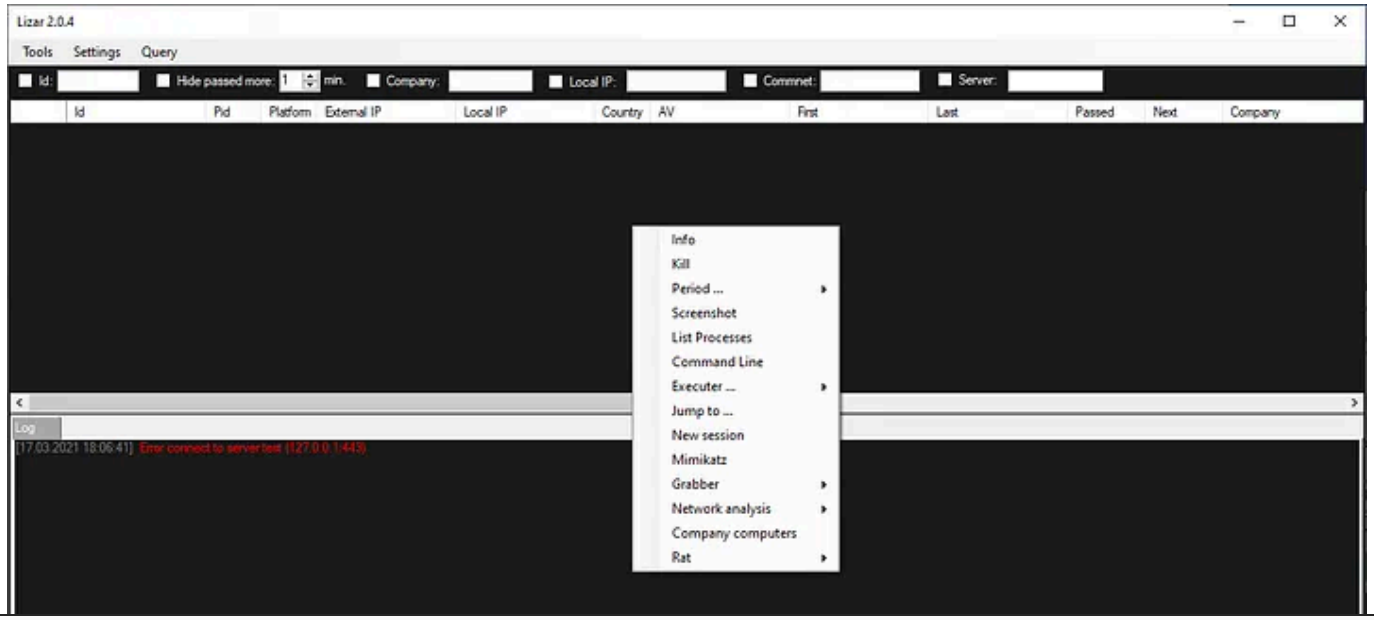
- `client.ini.xml` — XML configuration file;
- `client.exe` — client's main executable;
- `libwebp_x64.dll` — 64-bit version of `libwebp` library;
- `libwebp_x86.dll` — 32-bit version of `libwebp` library;
- `keys` — a directory with the keys for encrypting traffic between the client

Element group	Element name	Element description
`Servers → Server` Server configuration	`Name`	Server name displayed by the client
	`IP`	Server IP
	`Port`	Port which the server uses for listening
	`FileKey`	File with the key used to encrypt traffic between client and server
`JumperApp → App` Configuration of the application processes (target recepient of migration) on the infected OS	`Name`	Name of the process to which the loader can migrate

Characteristics	Value
File name	`client.exe`
SHA-256	`78a744a64d3afec117a9f5f11a9926d45d0064c5a46e3c0df5204476a1650099` (not present on VT)
File type	`PE32 executable for MS Windows (GUI) Intel 80386 Mono/.Net assembly`
Размер	238 080 bytes

Table 3. Characteristics of `client.exe`

Column name	Column contents
`Id`	Bot info `{bot name}:{bot id}:{pid}`, where: • `bot id` — checksum of system information (see Lizar loader for bot ID generation algorithm) • `pid` — the identifier of the process where the loader is active (if the loader is inactive, `pid` is set to `0`)
`Pid`	Same value as `pid` from the `Id`
`Platform`	The bitness and the type of process where the loader is active: • `x86 loader` is a 32-bit `EXE` or `DLL` • `x86.net loader` is a 32-bit `EXE` or `DLL` file, written using the .NET Framework platform • `x64 loader` is a 64-bit `EXE` or `DLL` • `x64.net loader` is a 64-bit `EXE` or `DLL` file, written using the .NET Framework platform
`External IP`	External IP address of the infected system running the loader (not always displayed correctly)



- `List Processes` — get a list of processes (Fig. 7). The plugin for this command is located on the server. If the plugin is successful, the list of processes will appear in a separate window.

Figure 10. `Network analysis` command in the Lizar client GUI

- `Rat` — run Carbanak (`RAT`). The IP address and port of the server and admin panel are set via the `client.ini.xml` configuration file (Fig. 11).

Date and time of the last detected server version compilation: `Fri Feb 19 16:16:25 2021`.

The application is run using the Wine utility with the pre-installed Wine Mono (`wine-mono-5.0.0-x86.msi`).

The server application directory includes the following components:

- `client/keys` — directory with encryption keys for proper communication with the client;

- `x64` — directory containing the `SQLite.interop.dll` auxiliary library file

Example contents of the configuration file:

- `System.Data.SQLite.dll` — auxiliary library file.

As we have already mentioned, Lizar loader and Lizar plugins run on the infected system and can logically be combined into the Lizar bot component. The bot's modular architecture makes the tool scalable and allows for independent development of all components.

We've detected three kinds of bots: DLLs, EXEs and PowerShell scripts, which execute a DLL in the address space of the PowerShell process.

The pseudocode of the main loader function, along with the reconstructed function structure, is shown in Fig. 13

The following are some of the actions the `x_Init` function performs:

1. Generate a random key `g_ConfigKey31` using the function `SystemFunction036`. This key is used to encrypt and decrypt the configuration data.

2. Obtain system information and calculate the checksum from the information received (Fig. 14).

3. Retrieve the current process ID (the checksum and `PID` of the loader process are displayed in the `Id` column in the client application).

4. Calculate the checksum from the previously received checksum and the current process ID (labelled `g_BotId` in Figure 13).

5. Decrypt configuration data: list of IP addresses, list of ports for each server. Configuration data is decrypted on 31-byte `g_LoaderKey` with `XOR` algorithm. After decryption, the data is re-encrypted on `g_ConfigKey31` with an `XOR` algorithm. The `g_LoaderKey` is also used when encrypting data sent to

3. Depending on the command and loader bitness, the server finds a suitable plugin from the `plugins` directory, then sends the loader a request containing the command and the body of the plugin (e.g., `Screenshot{bitness}.dll`).
4. The loader executes the plugin and stores the result of the plugin's execution in a specially allocated area of memory on the heap.
5. The server retrieves the results of plugin execution and sends them on to the client.

- ListProcess32.dll
- ListProcess64.dll
- mimikatz32.dll
- mimikatz64.dll
- NetSession32.dll
- NetSession64.dll
- rat32.dll

The plugin can run the following components:

- EXE file from the `%TEMP%` directory;
- PowerShell script from the `%TEMP%` directory, which is run using the following command: `{path to powershell.exe} -ex bypass -noprof -nolog -nonint -f {path to the PowerShell script}`;
- DLL in memory;
- shellcode.

Contrary to its name, this plugin has no grabber functionality and is a typical PE loader.

Although attackers call it a grabber, the loaded PE file actually performs the functions of other types of tools, such as a stealer.

Both versions of the plugin are used as client-side grabber loaders: `PswRdInfo64` and `PswInfoGrabber64`.

Jumper32.dll/Jumper64.dll

The plugin is designed to migrate the loader to the address space of another process. Injection parameters are set in the Lizar client configuration file. It should be noted that this plugin can be used not only to inject the loader, but also to execute other PE files in the address space of the specified process.

Figure 21 shows the main function of the plugin.

- by creating a process with a certain name and performing an injection into it;
- by creating a process with the same name as the current one and performing an injection into it.

Let's take a closer look at each method.

Algorithm for injection by process ID

- If the flag is not set, the executable file is started by calling the `CreateProcessA` function (Fig. 24).

- `powerkatz_short64.dll`

NetSession32.dll/NetSession64.dll

The plugin is designed to retrieve information about all active network sessions on the infected server. For each session, the host address from which the connection is made can be retrieved, along with the name of the user initiating the connection.

The pseudocode of the function in which the information is received is

The plugin can take a JPEG screenshot on the infected system. The part of the function used to save the resulting image to the stream is shown below (Fig. 31).

plugins from the `plugins/extra` directory are transferred from the client to the server, then from the server to the loader (on the infected system).

List of files in the `plugins/extra` directory:

- ADRecon.ps1
- GetHash32.dll
- GetHash64.dll

GetHash32/GetHash64

The plugin is designed to retrieve user NTLM/LM hashes. The plugin is based on the code of the `lsadump` component from Mimikatz.

Fig. 32 shows a screenshot with pseudocode of exported `Entry` function (function names are chosen according to Mimikatz function names).

If the plugin fails to start with `SYSTEM` permissions, it will fill the buffer with the data shown in Fig. 33.

Figure 33. Buffer contents when running the plugin without `SYSTEM` permissions

match the names of the corresponding variables and functions from Mimikatz):

commands. The decompiled view of the `wmain` function is shown below:

A list of `powerkatz` functions that are absent from `powerkatz_short`:

- `kuhl_m_acr_clean`;
- `kuhl_m_busylight_clean`;
- `kuhl_m_c_rpc_clean`;
- `kuhl_m_c_rpc_init`;
- `kuhl_m_c_service_clean`;

- email accounts from Microsoft Outlook and Mozilla Thunderbird.

The `nss3.dll` library is used to retrieve sensitive data from the Firefox browser and is loaded from the directory with the installed browser (Fig. 39).

`History`, `places.sqlite`, `Login Data` are all `sqlite3` database files. To work with `sqlite3` databases the plugin uses functions from the `sqlite` library, statically linked with the resulting DLL, i.e. the plugin itself.

For Internet Explorer and Microsoft Edge browsers, the plugin retrieves user credentials using functions from the `vaultcli.dll` library that implements the functions of the `vaultcmd.exe` utility.

PswRdInfo64.dll

When started by another user (other than `SYSTEM`), the plugin attempts to collect credentials for RDP access to other hosts. Credentials are collected using `CredEnumerateW` function, with the `TERMSRV` string as the target.

Conclusion

As the analysis shows, Lizar is a diverse and complex toolkit. It is currently still under active development and testing, yet it is already being widely used to control infected computers, mostly throughout the United States.

```
4d933b6b60a097ad5ce5876a66c569e6f46707b934ebd3c442432711af195124
515b94290111b7be80e001bfa2335d2f494937c8619cfdaafb2077d9d6af06fe
61cfe83259640df9f19df2be4b67bb1c6e5816ac52b8a5a02ee8b79bde4b2b70
fbd2d816147112bd408e26b1300775bbaa482342f9b33924d93fd71a5c312cce
a3b3f56a61c6dc8ba2aa25bdd9bd7dc2c5a4602c2670431c5cbc59a76e2b4c54
e908f99c6753a56440127e54ce990adbc5128d10edc11622d548ddd67e6662ac
7d48362091d710935726ab4d32bf594b363683e8335f1ee70ae2ae81f4ee36ca
e894dedb4658e006c8a85f02fa5bbab7ecd234331b92be41ab708fa22a246e25
b8691a33aa99af0f0c1a86321b70437efcf358ace1cf3f91e4cb8793228d1a62
bd1e5ea9556cb6cba9a509eab8442bf37ca40006c0894c5a98ce77f6d84b03c7
98fbccd9c2e925d2f7b8bcfa247790a681497dfb9f7f8745c0327c43db10952f
552c00bb5fd5f10b105ca247b0a78082bd6a63e2bab590040788e52634f96d11
21db55edc9df9e096fc994972498cbd9da128f8f3959a462d04091634a569a96
```