Baldr vs The World

A credential thief's burst of creative energy delivers a dangerous new threat

Albert Zsigovits, SophosLabs
### Contents

**Baldr: An introduction** .................................................................................................................. 4

**How criminals lure victims to Baldr** ............................................................................................ 6

  - Buying a Baldr Trojan .................................................................................................................. 6

**Baldr’s global impact** .................................................................................................................. 8

**Baldr victims profile** .................................................................................................................... 9

**Baldr's global distribution** .......................................................................................................... 12

**Baldr's distribution methods** ...................................................................................................... 14

  - .ace archive vulnerability vector ............................................................................................ 14
  - Weaponized RTF file vector ...................................................................................................... 16

**Baldr malware functionality: How Baldr works** ........................................................................... 20

  - Initial system profiling ............................................................................................................. 20
  - Retrieving the list of installed programs ..................................................................................... 21
  - Obtaining OS profiling data ....................................................................................................... 21
  - GPU and RAM information ......................................................................................................... 22
  - Disk information .......................................................................................................................... 22
  - Dumping saved credentials and history from installed web browsers ......................................... 22
  - Gathering FTP logins .................................................................................................................. 25
  - XMPP credentials from instant messaging clients ....................................................................... 27
  - Dumping VPN configuration files ............................................................................................... 27
  - Coin wallets .................................................................................................................................. 28
  - Telegram credentials and data ...................................................................................................... 29
  - Taking desktop screenshots .......................................................................................................... 30
  - Exfiltration of stolen credentials and profile data ....................................................................... 31

**Baldr’s evolution and eccentricities** ............................................................................................... 32

  - Code obfuscation and differences from v2.0 to v3.0 .................................................................... 34
  - Malware control panel settings .................................................................................................... 36
  - Sleep function ............................................................................................................................... 37
  - Malware, SelfDelete thyself ........................................................................................................ 38
  - Baldr’s malware family connections ............................................................................................ 39
Baldr vs The World

Baldr C2 and the administration panel............................................................................................ 42
Panel structure..................................................................................................................................... 42
Panel overview (v2.2 vs v3.0) ........................................................................................................... 42
Loader .................................................................................................................................................... 43
Cookies converter ............................................................................................................................... 45
Search ................................................................................................................................................... 46
Settings ................................................................................................................................................. 48
  Login page (v2.2) ............................................................................................................................... 50
  Login page (v3.0) ............................................................................................................................... 50
v2.2 server-side code ............................................................................................................................ 50
  gate.php .............................................................................................................................................. 51
v3.0 server-side code ............................................................................................................................ 54
  auth.php .............................................................................................................................................. 55
  database.php ...................................................................................................................................... 56
  gate.php .............................................................................................................................................. 56
Miscellaneous C2 components ........................................................................................................... 59
Panel inception (vulnerabilities) ......................................................................................................... 60
Mass-tracking Baldr panels .................................................................................................................. 62
C2 communication ............................................................................................................................... 63
  v2.1 ...................................................................................................................................................... 63
  v2.2 ...................................................................................................................................................... 63
  v3.0 ...................................................................................................................................................... 64
Baldr's use of bulletproof hosting....................................................................................................... 66
Tracking Baldr's buyer ........................................................................................................................... 68
  Telegram preferred for customer contact .......................................................................................... 70
  Baldr goes on hiatus? ......................................................................................................................... 71
Appendix: Indicators of Compromise ................................................................................................. 72
  Files analyzed in this report .............................................................................................................. 72
  Additional Indicators of Compromise: ............................................................................................. 72
Baldr: An introduction

Gamers have found themselves in the crosshairs of criminals for as long as it has been possible to monetize the theft of game credentials. Since the beginning of 2019, SophosLabs has been tracking the activity of a malware family we’re calling Baldr that, initially at least, targeted gamers through the use of misleading online videos.

These videos present the malware as a tool to gain an unfair advantage in a number of different online games, but the real purpose of Baldr is to enable both the purchasers and its creator to engage in identity theft.

We first observed the Trojan being advertised for sale on Russian cybercrime-related forums at the end of January, 2019. By the following month, we saw its distribution begin to increase, along with the price the malware authors were charging to criminals. As its distribution increases, so do the variety of methods that Baldr customers use to infect customers, including the use of maliciously crafted .ace archives and Office documents, which are either hosted for download or emailed to victims.

We consider Baldr an up-and-coming password stealer as we've observed its evolution through at least four major revisions over the past seven months. In that time, the malware's creator has added a raft of new features that put it in direct competition from better-known families. There has also been a bit of drama in the criminal underground, where the main developer and the principal distributor seem to have had a (somewhat public) falling out, with the distributor dropping Baldr as a product for sale. But the malware has not ceased functioning, and we expect it to re-emerge, possibly with a new name.

This paper provides a deep technical synopsis of Baldr malware, including its command-and-control administrative web panel, which several Baldr-using criminals carelessly left unprotected and downloadable from open directories. We've also come across what appear to be credential dumps generated by Baldr in files submitted to public repositories like Virustotal. While we will refrain from publishing victim details, we've anonymized and aggregated some of this data to illustrate the types of data most commonly stolen by operators of Baldr.

We also discuss some of the unique characteristics of Baldr’s killchain (implemented not by the malware's creator but by its criminal customer base) and its apparent relationship to other malware families, some of which Baldr itself delivers to victim machines as a malware distribution network.

The author wishes to thank Fraser Howard, Gabor Szappanos, and Andrew Brandt for their invaluable assistance in helping prepare this report.

Listed trademarks are the property of their respective owners. Third-party company, product, and service names are used for identification purposes only, and do not imply endorsement.
How criminals lure victims to Baldr

We first encountered Baldr as its distribution network started to spread through gaming circles. Several YouTube videos targeting specific online games were used as a lure for potential victims.

![Gaming-related Youtube videos distributing Baldr](image)

These videos were used to advertise tools that purport to give online game players one or more abilities to cheat in games such as Counter-Strike: Go or Apex Legends. The video details often contained a link that a viewer could use to download the tool. We also saw download links distributed in gaming-specific channels on both the Discord and Telegram chat services.

In addition to these distribution methods, we found instances where we found Baldr malware included with pirated versions of games offered for illicit download, as well as bundled along with maliciously modified installers of otherwise legitimate cryptocurrency miner software.

Buying a Baldr Trojan

We began tracking the cost the malware distributor charged Baldr customers in January, 2019, by following the online advertisements for the malware. The following chart traces the cost of the malware and overlays data about the malware's self-reported version numbers. The cost of a license for Baldr also includes access to future upgrades; The criminal customer receives a unique ID which then can be used to request updated builds from a Telegram chatbot.
Each major version update has offered additional new features, and corresponds to an increase in price, though the malware distributor has periodically offered discounts to potential customers. According to at least one posting on a dark web message board, more than 200 criminals have purchased a license to use Baldr.

However, it seems Baldr’s future is unpredictable: The makers of Baldr appear to have had a falling out with one of their larger distributors, and as this story went to press, the primary distributor appears to have stop working with the Baldr developers. Based on the nature of this type of criminal enterprise, we suspect that Baldr will once again be offered for sale, and the distribution issues are only temporary.
The distributor who has been primarily responsible for selling Baldr has, in the past, posted regular updates on sales numbers. Using those numbers, SophosLabs estimates that sales of Baldr have already brought in from $25,000 to about $32,000, but that is not the only source of criminal income for its creators, who also stand to make additional money from the sale of stolen credentials or payment card data.

**Baldr's global impact**

Baldr's scale remains relatively low when compared to more widely distributed credential theft malware, such as Trickbot. However, Baldr's impact is not insignificant and the malware has been gaining traction. Baldr has significant global distribution.

Although the stealer is being sold on, primarily, underground forums based in Russia, we discovered that, unusually, it was also executed in Russia at a scale that made Russia the third most highly targeted country for attacks.

We also detected significant volumes of Baldr infections in Indonesia, the United States, Singapore, Brazil, India, and Germany. The following heatmap shows counts of live infections detected, broken out by country in which the infection took place.

One reason we discovered for Russia's high detection rate is the fact that Baldr's customers – themselves criminals – sometimes run the malware samples themselves on machines they control, as a test to ensure the malware works as intended. Sometimes they do this
deliberately, but we discovered that, in some instances, the malware was executed accidentally on the computers belonging to the criminals who purchased the malware. This generates some statistical anomalies.

Further complicating matters are the fact that we determined that some copies of the malware were run in public sandboxes, which detonate the malware in a controlled environment for research or analysis purposes. Those machines are clearly not operated by actual victims, so they skew the numbers somewhat.

Finally, to understand why some countries seem to be more targeted than others, we would have to develop a better understanding of who make up Baldr's customer base. The abundance of logs of Russian victims suggest that some of Baldr's customers may be Russian groups that target victims within Russia.

Breaking down Baldr infections by countries (data labeled "Indonesia" includes Singapore infections)

**Baldr victims profile**

During the course of our investigation, we came across files unintentionally uploaded to public malware repositories that contain data stolen from victims of Baldr. While we have provided this information to law enforcement, we thought it might be informative to anonymize this data and use it to illustrate the most common types of information that Baldr has already managed to steal from some victims.
On the following two infographics, we’ve aggregated data in ways that we hope illustrates what victims lost most frequently to Baldr.

The first word cloud features the 50 mail domains where people had accounts most commonly stolen by Baldr criminals. While it should come as no surprise to find Microsoft, Google, or Yahoo dominating the most frequently used (and therefore, stolen) domains, it was significant to see Russia-based mail.ru, as well as encrypted mail provider Protonmail. And how many people still have Rocketmail accounts?

![The 50 e-mail domains most frequently stolen by Baldr criminals](image)

The second word cloud shows the 25 web services where credentials were most frequently stolen. The presence of several gaming or gaming-peripheral websites targeted for credential theft by Baldr probably comes down to how criminals initially spread Baldr, on videos touting game cheating tools. Sony. Battle.net, Steam, and Epic were all represented, as well as Twitch and Discord.
The typical victim uses many of the very same services that most people use on a daily basis: social media, file repositories, streaming, shopping sites, and (of course) online gaming services. Baldr’s victims are subject to identity theft, or credit card fraud as a result of the stolen information gained directly or indirectly. One indirect method is that a criminal may sell a stolen credential from the gaming (Battle.Net, Steam, Epic Games, Sony PSN) or shopping services (Amazon, eBay, PayPal) on so-called "carder" sites, which are marketplaces for stolen data.

Digging deeper, we found that one of Baldr’s distributors also supplies logs to the individual who is re-selling stolen credentials. This looks to be a rather interesting business model, since the distributor can double down on the stolen victim logs, selling stolen credential/card information to earn a little extra.
Baldr's main distributor claims to be the source for supplying logs for resale to a carder

**Baldr's global distribution**

While looking to enrich our statistics, we performed a relationship analysis using VirusTotal Graph over the course of several months. The following two snapshots were taken on the 22nd of March, and on the 6th of May, with the number of "related" nodes in the map illustrating the growing size of the Baldr botnet.

At the time of the first snapshot, only a couple of dozen Baldr samples were present on VirusTotal. Six weeks later, the number had grown in the range of several hundreds of unique samples.

From a different perspective, it is interesting to look at how many different Baldr samples are appearing daily, over time. Baldr had its largest distribution days around mid-May, while the
numbers appear to drop off in early June. We will discuss why that's happening later in the paper.

Unique Baldr samples appearing in-the-wild per day, April-June 2019

With its latest update, VirusTotal Intelligence now offers new keywords which enable us to sift through the results that are stemming from dynamic analysis. One search key we could use to cluster these samples is this one, which lets us hunt for unique commonalities between hundreds of similar samples:

Baldr behavior_processes:choice behavior_network:gate.php
Baldr's distribution methods

As previously discussed, Baldr began as a malware whose distribution was linked to online videos that purport to show a tool that gives online gamers an unfair advantage over their competitors. But as the malware's customer base grew, so did the variety of methods we saw to send the malware to victims. In this section, we'll describe two of the more interesting varieties of distribution methods we observed in the wild.

.ace archive vulnerability vector

One of the vectors used to distribute Baldr exploits a vulnerability in the file archiving tool WinRAR that was discovered in February, 2019. Designated CVE-2018-20250, it's a path traversal vulnerability in a library called unacev2.dll. WinRAR uses the DLL to extract ACE archives, a less common archive format, and the DLL hasn't been updated in more than a decade.

When a victim opens an ACE archive that has been specifically formulated to exploit this vulnerability, the DLL can be made to extract a file to an arbitrary path and then execute that file.

In the .ace file exploits we observed delivering Baldr, the malicious .ace file's filename field has been tampered with. This means that WinRAR completely ignores the destination folder designated by the user (or by WinRAR by default), and instead it evaluates the filename as a relative path. This introduces several opportunities for shenanigans.
For successful exploitation, the ace archive needs to be in one of the %USERHOME% directories (Desktop or Downloads will work). Once the user clicks the maliciously crafted ACE file, the PE32 file inside the archive gets dumped to a hardcoded relative file path. An attacker can craft an ACE file with an absolute file path to the Startup folders, for example. Anything inside of that folder would launch after system boot, an old but effective persistence technique.

The .ace archives crafted to exploit this bug contain a filename field which has been modified in a special way: The path starts with the distinctive string `C:\C:C:..` which, when read by the ACE plugin in WinRAR, fools the filename parser.
After the next reboot, the system executes the Baldr stealer that has been dropped into the Startup folder.

**Weaponized RTF file vector**

Another interesting delivery vector for Baldr employs a maliciously-crafted RTF file that exploits a vulnerability in Microsoft Office 2007, 2010, 2013, and 2016, with the designation CVE-2018-0802. The exploit only works on versions of Office which have been patched to fix an earlier bug in the Equation Editor (designated CVE-2017-11882), for which Microsoft released an update in November, 2017.

As a patch for CVE-2018-0802 was released in January, 2018, this exploit only affects a relatively small population of the overall Microsoft Office userbase. Why criminals chose to use this particular vulnerability to distribute malware more than a year after the patch was released remains a mystery, as subsequent updates to Microsoft Office have essentially removed the vulnerable Equation Editor component from Office, altogether.

Here’s an illustration of the CVE-2018-0802 Baldr killchain from Sophos Central, when we detonated a sample in a controlled environment and allowed our Intercept X EDR product to
Quickly glancing through the RTF file, we see there’s an OLE equation object inside named Equation.3.

Using Didier Stevens’ rtfdump.py tool, we can easily extract the objects inside the RTF file.

Looking at object 7, we see an embedded executable:
And looking inside of object 10 (Equation.3), we can sift through the exploit's streams.

Stream 5 (Equation Native) contains a command that will run the executable embedded inside object 7.

This stream has a very well documented structure. Again, using Didier’s format-bytes.py tool, and defining a header scheme we can cross-reference the headers with their corresponding bytes seen in the stream:
On line 18 we can spot the *font name* with an additional 26 bytes long command plus the 100 bytes long [0x20] byte padding (spaces).
Baldr malware functionality: How Baldr works

At its most fundamental level, Baldr functions as a tool to profile a victim's computer, and steal as much valuable data as quickly as possible. In the following chapter, we go into great detail about what information Baldr collects, and, mechanically, how it accomplishes each step in the process. Baldr typically runs for 15-30 seconds, during which time everything in the following section of this report takes place.

A typical data collection package contains a richly detailed profile of the computer itself and its setup, and data scraped from locations on the victim's filesystem commonly used by a variety of programs to store saved credentials.

The malware we analyzed does not perform a man-in-the-browser hijack and steal credentials as they are entered, but merely grabs anything that looks like it might contain useful or valuable data, including Bitcoin wallets, VPN profiles, and of course saved passwords from FTP clients, IM and chat services, and email clients. Baldr can scrape the saved passwords, cookies, and other information from at least 22 different web browsers and will relieve you of your hard-won cryptocurrency if you use one of 14 wallets the malware is capable of raiding.

Initial system profiling

The malware starts by collecting the following profiling information about the victim's machine, and the network to which it is connected:

- Geo information
  - IP
  - Country Code
  - Country
  - State Name
  - City
  - Time zone
  - ZIP
  - ISP
  - Coordinates

- Machine information
  - Username
  - PC Name
  - UUID
  - HWID
  - OS Version
  - CPU Model
  - GPU Model
  - RAM information
  - MAC Address
  - Screen Resolution
  - System Language
  - Layout Language
  - PC Boot Time
  - Drive List
  - Drive Model
  - Drive Serial Number
  - Disk Size
  - Disk Signature
  - Installed Programs List
  - Running Processes List
Retrieving the list of installed programs

Baldr enumerates the list of installed programs from the HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall path in the Windows Registry. `DisplayName` corresponds to the name of the program, and `DisplayVersion` its version number. The malware concatenates these two strings together, and puts each one on a new line.

```
private static void 83762095()
{
    List<string> list = new List<string>();
    try
    {
        string name = "SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall";
        using (RegistryKey registryKey = Registry.LocalMachine.OpenSubKey(name))
        {
            foreach (string name2 in registryKey.GetSubKeyNames())
            {
                using (RegistryKey registryKey2 = registryKey.OpenSubKey(name2))
                {
                    try
                    {
                        string text = registryKey2.GetValue("DisplayName").ToString();
                        string str = registryKey2.GetValue("DisplayVersion").ToString();
                        if (!string.IsNullOrEmptyWhitespace(text))
                        {
                            list.Add(text + " + str + "r\n");
                        }
                    }
                }
            }
        }
    }
}
```

Getting list of installed programs from the Windows Registry

Obtaining OS profiling data

Baldr’s next task is to produce system information, using the `Win32_OperatingSystem` WMI class; The malware uses a complex series of `if` statements to determine the version number. Once the correct version is detected, that number gets stored in variable `6bd967dc`.

```
private static void 25e49f4c()
{
    string 6bd967dc = "";
    string text = "";
    string text2 = "";
    try
    {
        ManagementObject managementObject = new ManagementObject("Win32_OperatingSystem@");
        6bd967dc = managementObject["SerialNumber"].ToString();
        text = managementObject["Caption"].ToString();
        text2 = managementObject["OSArchitecture"].ToString();
        if (text.Contains("8"))
        {
            text = "Windows 8";
        }
        if (text.Contains("8.1"))
        {
            text = "Windows 8.1";
        }
        if (text.Contains("10"))
        {
            text = "Windows 10";
        }
    }
}
```

Getting Operating System information with a WMI query
**GPU and RAM information**

CPU\GPU\RAM information is also gathered via similar WMI queries:

<table>
<thead>
<tr>
<th>Information</th>
<th>WMI query</th>
<th>Stored in variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU</td>
<td>select * from Win32_VideoController</td>
<td>2577f0df</td>
</tr>
<tr>
<td>RAM</td>
<td>Select TotalPhysical from Win32_ComputerSystem</td>
<td>5a05f91c</td>
</tr>
</tbody>
</table>

**Disk information**

To gather system information, the stealer calls several objects and takes a WMI query as input. HDD information is being evaluated by calling the following WMI class:

- Win32_DiskDrive

And are correlated with the *associators*:

- Win32_DiskDriveToDiskPartition
- Win32_LogicalDiskToPartition

All this information gets appended to a file named `information.log`.

**Dumping saved credentials and history from installed web browsers**

In the next operation, the malware dumps all saved credentials from each of the victim's currently installed browsers into a file called `passwords.log`.

Baldr can enumerate the following web browsers; They all use similar techniques to store credentials and cache data:

- YandexBrowser
- Zotero
- Waterfox
- Thunderbird
- Opera
- Supermedium
- Songbird2
- SeaMonkey
- Scout
- Pale Moon
- Opera Neon
- Mozilla
• Firefox
• Fast Web Browser
• Edge Dev
• Edge SxS
• Dragon
• Citrio
• Chrome
• Chrome Beta
• Brave Browser
• Torch
• Vivaldi

On top of this, the complete browser cache is subject to exfiltration, as the malware dumps the following types of data into different files:

• Saved autocomplete information → autocomplete.txt
• Saved credit card information → cards.txt
• Browser cookies → cookies.txt
• Browsing history → history.txt
• All domains visited → cookieDomains.log
There are 6 files of interest for the browser credential theft operation, and the malware gathers data from these files:

- Cookies.sqlite
- Places.sqlite
- Formhistory.sqlite
- Logins.json
- Key3.db
- Key4.db

Firefox stores its cookies in `cookies.sqlite` inside a table called "moz_cookies".

Places.sqlite contains a list of all the web pages a user visited, but also stores bookmarks and attributes for visited sites.Forensically speaking, this is the single most important file for forensic investigators (or crooks) to examine. The information located here can be easily cross-referenced with the cookies.sqlite, formhistory.sqlite, and permissions.sqlite files to understand a victim’s browsing habits.

Next, the malware searches for logins.json, key3.db and key4.db and then it searches inside of logins.json for a few specific keywords using regular expressions:

- hostname
- encryptedPassword
- encryptedUsername
Hostname, encryptedPassword, encryptedUsername are selected from logins.json

The `formhistory.sqlite` file holds not only all the data that is used for filling out forms, but search keywords as well.

The `Sqlite files` can be opened with an add-on called Sqlite Manager

**Gathering FTP logins**

The malware also steals saved FTP credentials from common FTP client applications, such as FileZilla and Total Commander.

First, the malware tries to look up the GHISLER folder from `%AppData%` and then enumerates the files to find `wcx_ftp.ini`.

The file `wcx_ftp.ini` being searched for
One interesting observation is that Total Commander stores the FTP password in a proprietary encrypted format that offers no real, practical security:

```
[wx_fbin]
1 [connections]
2 1=totalcmd ftp
3 default=totalcmd ftp
4 [totalcmd ftp]
5 host=
6 username=
7 password=F35B4357FB9EFB05A06A6E
8 passmode=0
9 MLSD=-1
10 [default]
11 passmode=0
```

*Stored Total Commander FTP scheme*

FileZilla is the other target of the stealer when looking at FTP stealing functions. Both `recent_servers.xml` and `sitemanager.xml` are exfiltrated from the `AppData\Roaming\FileZilla` folder.

```
string path = folderPath + FileZilla\; 
if (Directory.Exists(path))
{
    foreach (FileInfo fileInfo in new DirectoryInfo(path).GetFiles())
    {
        if (fileInfo.Name.Contains("recent_servers.xml"))
        {
            list.Add(new file_entry
            {
                filename = "FTP\FileZilla\" + fileInfo.Name,
                filedata = ea10584c37f3f35c(fileInfo.FullName)
            });
        }
        if (fileInfo.Name.Contains("sitemanager.xml"))
        {
            list.Add(new file_entry
            {
                filename = "FTP\FileZilla\" + fileInfo.Name,
                filedata = ea10584c37f3f35c(fileInfo.FullName)
            });
        }
    }
}
```

*Looking for FileZilla xml files on the system*

Passwords here are encoded in base64, which offers no real security.
XMPP credentials from instant messaging clients

Baldr searches for the following chat/instant messaging clients.

- PidginPsi
- Psi+
- Jabber

The malware can collect Jabber configuration files as well. These xml files always contain the credentials in plain-text to access a given XMPP server. The following files are inspected and enumerated if found:

- `.purple\accounts.xml`
- `\Psi\profiles\default\accounts.xml`
- `\Psi\profiles\default\accounts.xml`
- `Jabber\pidgin_accounts.xml`
- `Jabber\psiplus_accounts.xml`
- `Jabber\psi_accounts.xml`

Dumping VPN configuration files

As private VPN services have gained popularity, credential stealing malware now routinely targets the configuration files for popular services, such as ProtonVPN and NordVPN, which contain credentials. Baldr enumerates both files from the `AppData\Local` folder, where the VPN client software stores its `user.config` VPN profile files.
We found that credentials stored in NordVPN’s *user.config* file are first encrypted with the Windows Data Protection API, and then the encrypted blob gets base64-encoded. NordVPN and ProtonVPN configuration profiles are very similar to each other hence the capability for both.

**Coin wallets**

Baldr steals any or all of the following wallet types it can find on the victim's computer (because of course it does).

- Bitcoin
- Zcash
- Litecoin
- Monero
- Bytecoin
- ElectronCash
- MultiDoge
- DigiByte
- Electrum
- Bitcoin
- Actinium
- Exodus
- Ethereum
- Jaxx Liberty
Enumeration is done via recursion lookup for files with the `.wallet` extension.

![Code snippet](image)

**Method for finding virtual currency wallets**

**Telegram credentials and data**

The stealer will sift through the active processes and look for a process that has Telegram in its name. Once the process is identified, the malware queries the filesystem path to the executable, and then creates a folder named `\tdata` in the directory path. The malware also searches for the folder `AppData\Roaming\Telegram Desktop`, and dumps the Telegram-specific folders:

- `D877F783D5D3EF8C`*
- `D877F783D5D3EF8C\map0`
- `D877F783D5D3EF8C\map1`
- `D877F783D5D3EF8C0`*
- `D877F783D5D3EF8C1`*
Understanding and implementing capability for all these applications and their corresponding custom file formats certainly required a lot of time and effort.

**Taking desktop screenshots**

Because a picture is worth 1000 words, the malware takes a screenshot of the current active desktop. The height and the width of the desktop screen gets assigned to variables `text` and `text2`, converts them to integers, stores them in variables `num` and `num2`, and passes them to the method labeled `d359ba2b.ee11870e3` as parameters.
Inside the `ee11870e3` class, the `CopyFromScreen` method grabs a screenshot from the desktop and passes it to `MemoryStream`. Then the `MemoryStream` gets dumped and saved into a jpeg file called `screen.jpeg`.

![Desktop Screen Screenshot](image)

**The desktop screen gets saved in a jpeg file**

**Exfiltration of stolen credentials and profile data**

Once the malware concludes the information collection procedures, it dispatches an exfiltration package in a single encrypted file (as of v3.0), inside of an HTTP POST request.

At the next section of the code, we explain how the malware constructs its C2 parameters: the `stringBuilder.Append` method concatenates the parameters then passes those on to the POST request.

```csharp
StringBuilder stringBuilder = new StringBuilder();
stringBuilder.Append("os=" + char609h.B5362F4fA.700U96e0 + ";");
stringBuilder.Append(string.Format("file=[0]", 53Be35588.9a82182F));
stringBuilder.Append(string.Format("cookie=[0]", 53Be35588.9927133F));
stringBuilder.Append(string.Format("pass=[0]", 53Be35588.90a56650));
stringBuilder.Append(string.Format("creds=[0]", 53Be35588.9a5e3661));
stringBuilder.Append(string.Format("wallets=[0]", 53Be35588.90a5615F));
stringBuilder.Append("id=");
stringBuilder.Append("version=");
byte[] array5 = df665383.0f6f14c7.GetBytes(stringBuilder.ToString());
```

Baldr's POST request decompiled
It is easy to intercept C&C communication with FakeNet and observe the exfiltrated data being sent out.

![FakeNet capture](image.png)

*Dumping victim information from FakeNet’s capture*

With version 2.x the data sent out is unencrypted, so we can take a stab at exporting the .zip file that contains all the information from the victim.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browsers</td>
<td>File folder</td>
<td></td>
</tr>
<tr>
<td>FTP</td>
<td>File folder</td>
<td></td>
</tr>
<tr>
<td>Jabber</td>
<td>File folder</td>
<td></td>
</tr>
<tr>
<td>Telegram</td>
<td>File folder</td>
<td></td>
</tr>
<tr>
<td>VPN</td>
<td>File folder</td>
<td></td>
</tr>
<tr>
<td>cookieDomains.log</td>
<td>Text Document</td>
<td>18 KB</td>
</tr>
<tr>
<td>information.log</td>
<td>Text Document</td>
<td>7 KB</td>
</tr>
<tr>
<td>passwords.log</td>
<td>Text Document</td>
<td>14 KB</td>
</tr>
<tr>
<td>screen.jpeg</td>
<td>JPEG image</td>
<td>121 KB</td>
</tr>
</tbody>
</table>

*List of files Baldr exfiltrated in the .zip package*

**Baldr’s evolution and eccentricities**

Baldr has been painstakingly built and improved upon over its existence. We’ve noted several characteristics that give rise to the idea that its creator is unafraid to break with tradition, at least as far as traditional malware characteristics go.
For instance, we’ve observed that even though its author has been adding features nonstop since Baldr’s inception, the Trojan is also missing any kind of fingerprinting component (like placing a Mutex on the system). Malware often employs mutexes to prevent the same executable from running multiple simultaneous instances on the same target machine. Baldr uses no method we can determine that would prevent multiple instances from running.

Maybe it’s an oversight, or maybe it’s intentional. As with the lack of a persistence mechanism, the lack of a mutex hasn’t gotten in the way of the Trojan’s relative success. It also has no ability today to propagate itself across a network. Doesn’t mean it won’t ever happen.

Baldr code uses multiple threads that constantly spawn and die during its execution. The malware launches child threads one after the other with `System.Threading.Thread` to complicate analysis, and to provide parallel execution. Looking at the source code, we can spot that all threads are started as background threads: these do not keep the managed execution environment running.

Here is a breakdown what each thread is responsible for:

<table>
<thead>
<tr>
<th>Thread Names</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>thread</td>
<td>Installed programs list</td>
</tr>
<tr>
<td>thread2</td>
<td>Active process list</td>
</tr>
<tr>
<td>thread3</td>
<td>Operating system version</td>
</tr>
<tr>
<td>thread4</td>
<td>Disk information</td>
</tr>
<tr>
<td>thread5</td>
<td>CPU information</td>
</tr>
<tr>
<td>thread6</td>
<td>RAM information</td>
</tr>
<tr>
<td>thread7</td>
<td>GPU information</td>
</tr>
<tr>
<td>thread8</td>
<td>Function for getting MAC address</td>
</tr>
</tbody>
</table>
Code obfuscation and differences from v2.0 to v3.0

Unfortunately for analysts, Baldr employs an excessive number of obfuscation layers (at last count, 9) that thwarts static code analysis. Can we hack it? (Yes we can!)

First, we approached the problem with automatic de-obfuscators. We found out pretty quickly that those types of tools do not work in the context of this type of bespoke malware. Due to the complexity of the obfuscation, Baldr requires a bit more time and some manual labor to fully de-obfuscate the code.

The transition from the second to the third major release did not bring many changes. One significant improvement was an optimization of the server-side C2 code.

Baldr also changed how it takes screenshots: In v3.0 the developer decided to mask the screen.jpeg string hardcoded in the sample and use a base64 encoded version, instead. Literally, c2NyZWVuLmpwZWc=, the base64 representation of the "screen.jpeg" filename. Baldr decodes this string on-the-fly and writes out the contents of the array into the newly decoded base64-named file. If the developer did this to defeat detection, maybe they didn't think it through well enough.
In v3.0 Baldr encrypts its C2 parameters with a 4-byte XOR key, which the malware acquires from the C2 server response. It's not well protected, as we've observed both the plain-text and the encrypted parameters in memory. *cdc86873.Invoke* stores the C2 server address, hardcoded into each sample.

![Code snippet](image)

*The new base64-encoded screen.jpeg string and the conversion method*

The code snippet shown here demonstrates the encryption process. The malware concatenates the stored data, then connects to its C2 using `AbsoluteUri` which makes a POST request. The Content-Disposition header shows the malware sends home its exfiltrated data in a

![Webrequest output](image)

*HttpWebRequest calls the hard-coded C2 server*

C2 parameters in plain-text and in encrypted form seen in memory

Once the malware concatenates the stored data, it connects to its C2 using *AbsoluteUri* which makes a POST request. The Content-Disposition header shows the malware sends home its exfiltrated data in a
file named Encrypted.zip. (The content is easily decrypted with the 4-byte XOR key in hand, which we found one can retrieve from network traffic capture or memory analysis.)

Malware control panel settings

A new separator has been introduced in v3.0: `~;~`

Once the sample receives a response from the C2 server, it uses `DownloadString.Split` to split up the responses, separated by tilde-wrapped semicolons.

Based on the server settings, the malware will execute different operations. The settings are fetched and pushed onto `array2`.
We matched up the elements of the array with their corresponding panel settings:

<table>
<thead>
<tr>
<th>Settings array:</th>
<th>Panel PHP variables:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Array[0]</td>
<td>$telegram</td>
</tr>
<tr>
<td>Array[1]</td>
<td>$history</td>
</tr>
<tr>
<td>Array[2]</td>
<td>$autocomplete</td>
</tr>
<tr>
<td>Array[3]</td>
<td>$cards</td>
</tr>
<tr>
<td>Array[4]</td>
<td>$_cookies</td>
</tr>
<tr>
<td>Array[5]</td>
<td>$passwords</td>
</tr>
<tr>
<td>Array[6]</td>
<td>$jabber</td>
</tr>
<tr>
<td>Array[7]</td>
<td>$ftp</td>
</tr>
<tr>
<td>Array[8]</td>
<td>$screenshot</td>
</tr>
<tr>
<td>Array[9]</td>
<td>$selfDelete</td>
</tr>
<tr>
<td>Array[10]</td>
<td>$vpn</td>
</tr>
<tr>
<td>Array[12]</td>
<td>$executionTime</td>
</tr>
</tbody>
</table>

**Sleep function**

Baldr introduced the ability to receive an execution delay command from the C2 server in version 3. array2[12] is responsible for storing the delay time, but the value gets multiplied by 1000, since the Thread.Sleep method requires an integer parameter that is in milliseconds.
By default, there is no delay to the execution of the malware

A typo seen in ExecutionTime string

**Malware, SelfDelete thyself**

Version 3.0 seems to include a relatively small, additional routine that v2.x did not have. At the very end of the execution chain, once Baldr finished exfiltrating the data and closed the connection with the C2 server, it initiates a cleanup command through the local command processor:

<table>
<thead>
<tr>
<th>API sequence</th>
<th>Command:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ShellExecuteExW</td>
<td>cmd.exe /C choice /C Y /N /D Y /T 3 &amp;Del &quot;C:\baldr.exe&quot;</td>
</tr>
<tr>
<td>2. FindFirstFileExW</td>
<td>C:\Windows\system32\choice.exe</td>
</tr>
</tbody>
</table>
This little code snippet was most likely implemented to have a little more operational security present from the malware developer's perspective: delete the original program so it is harder to figure out what happened on the infected system and implement a silent execution, so the command prompt gets suppressed. The 3 seconds timeout delay may also fool some sandboxes resulting in halting execution.

We have previously seen this method in other families like GrandStealer. Crooks seem to operate on the "if it works, don't change it" philosophy.

**Baldr's malware family connections**

Baldr is not merely a credential thief. The malware serves as a distribution method for other types of malware, capable of downloading a payload executable from its C2. And Baldr has been served up as a payload by other malware. Baldr's malware relationship status is: Complicated.
For example, we recently observed ransomware loading Baldr onto a victim's machine, executing the stealer to glean data of value from a victim’s computer before starting the encryption routine. We've also logged instances of Arkei or Megumin dropping Baldr during an infection, and Baldr dropping Megumin. That makes sense, as Baldr's primary distributor also sells Megumin.

Functions used by Baldr to grab FileZilla and Telegram data mimic the subroutines in another credential thief, GrandStealer. The cookies converter function on Baldr panel is a precise match from Azorult's panel: the same Javascript code is responsible for the conversion. Baldr is a Frankenstein's monster of code bits from other malware.
The same NetscapeToJSON cookie converter function seen in Azorult
Baldr C2 and the administration panel

Specific type of malware families (stealers, bots, RATs) often are sold with their own administrative panel for the ease of management of bots or victim logs. Some of Baldr’s customers were careless with their operational security, and left the C2 package accessible in an open directory on the C2 server, so we downloaded a few to take a closer look.

As it turns out, having a copy of the C2 server code is akin to stealing the other team’s playbook, because it gives you a lot of insight into the goals of the malware author, and the strategies they chose to achieve those goals.

The malicious Baldr stealer - in all cases - connects to this panel via a "gate", which sorts the incoming data into a specific folder based on a few parameters: these could be date, location or IP address. Then the victim logs are easily manageable via the GUI.

Panel structure

Panel overview (v2.2 vs v3.0)

The server-side panel is a very simple administrative board where the operator can initiate various actions. It consists of five pages and each page has its own distinctive function:

- Dashboard
- Loader
- Cookies Converter
- Search
- Settings
The dashboard provides general statistics about the collected victim logs. An administrator can also look at individual logs, take actions on them, download specific parts of the log file, or tag unique logs with comments.

![The main dashboard](image)

**Loader**

The loader page probably has the greatest impact, as an operator can designate the server to push down a file payload to the victim. The malware pulls the file down on its next check-in with the C2 server.
Loader can insert further payloads onto the victim machines (v2.2)
New loader options implemented (v3.0)

Cookies converter

The cookies converter page is a simple NetscapeToJSON converter script which basically beautifies the Browser cookies into a readable format.
**Built-in NetscapeToJSON converter**

### Search

A quick way for admins to look for specific log files, the search feature finds different strings inside cookies, wallets, credentials/passwords. The v3.0 panel brought some additional filters too.
The operator can search for specific entries amongst all victim logs (v2.2)
The settings pane allows the administrator to handle victim logs at mass (v2.2)

Buyers can selectively choose which type of logs they would want to extract from victims.
Significant upgrade in granularity (v3.0)

Note the Grabber rules section: it enables the potential buyer to add specific locations which are then evaluated at malware runtime grabbing any files that match the given criteria.

Usually this section is full of entries of Desktop\Download\Temp folder paths. The following environmental variables also work as a specified location to grab from:

- `%ALLUSERSPROFILE%`
- `%APPDATA%`
- `%HOMEPATH%`
- `%USERPROFILE%`
- `%USERPROFILE%\Desktop`
- `%USERPROFILE%\Documents`
- `%USERPROFILE%\Downloads`

We can see the implemented *SelfDelete* function in the v3.0 version of the panel which was detailed previously. Also, here’s the execution delay function which was also dissected in the functionalities section.
Baldr's third iteration brought some changes to the login panel: authentication is done now via the `http-basic-auth` protocol, which sends the credentials base64-encoded inside the Authorization header of the HTTP response.

**v2.2 server-side code**

Starting with `auth.php`, we see the emergence of Baldr using a simple authentication method that works like this.

Once Baldr dispatches its credentials, the script grabs the POST parameters `login` (username) and `password` and cross-references them with the ones stored in the backend mysql database.

If it matches and the login credentials are correct the page gets redirected to the dashboard with a HTTP 301 follow-up response.
gate.php

This script file is the one responsible for C2 communication. Right off the bat we can identify the panel's version and its associated xorKey that is used for encrypting the exfiltrated data, which is not set at the start. Later down we learn that the xorKey is generated via a random function.

Next on we see that the REMOTE_ADDR (potential victim IP) is grabbed from a variety of HTTP Headers that might contain the real victim IP behind potential firewalls, proxies.

After that we find 2 important functions. "myxor" implements a simple xor operation that does encryption on the C2 channel. The generateRandomString is the randomSeed for generating a specific XOR key.
This next section specifies what the C2 response should look like when the gate.php resource is requested. We can see the exfiltrated C2 data is received through `php://input`.

In the next `if` branching, the panel is checking whether data was already received from the given victim IP. Once a request has been received by `gate.php` it stores a fingerprint file in `/tmp` with a 4-byte long `xorKey`, that gets generated with `generateRandomString` function.
IPs visiting gate.php on one of the Baldr's C2 server

The reason we see two different values in size is that the ones with 4 bytes content had only visited gate.php directly: the generateRandomString will execute even if there is no data sent through php://input as seen above in the source code.

The two IPs we see with 117 bytes long content are real victims as the content inside them matches up with the Baldr's C2 parameters.

The developer implemented a separate category for CIS (Commonwealth of Independent States) related countries, these include Russia, Kazakhstan, Ukraine and Belarus. If enabled, these logs will populate the logs/cislogs folder path on the C2 server.

If we consider the Russian underground forum origins of this malware, this is surprising: most Russian-made malware isn't used in attacks against the motherland, but maybe the developer wants to keep their options open. It could be that, in future versions of the malware, Baldr would handle infections in CIS countries differently, but there isn't anything like that now.
The code shown below tries to determine the country of origin of the victim. The C2 server initiates a request to \textit{ip-api.com/json} and gets a json response, which contains the requestor’s IP, country, ISP, region, ZIP code, etc. The json response gets flattened and stored in variable $\textit{loc}$, and then the country gets passed to $\textit{country}$ variable.

If there is no response from \textit{ip-api.com}, or the country field contains nothing, an ERROR label will be shown on the panel.

Countries are being identified by a lookup to \textit{ip-api.com}

The json response is stored in $\textit{loc}$ and then country gets selected and stored in $\textit{country}$

\textbf{v3.0 server-side code}

A quick glance from file size perspective: we can tell that there have been quite a few modifications. \textit{Auth.php} size was cut, \textit{gate.php} also got modified a bit and we can spot an overall refinement.
auth.php

In the previous version of Baldr, the authenticating credential was pulled from the local mysql database and it was compared against the one the user put in. v3.0 brought some advancements in this area too, as only the MD5 hash of the password gets evaluated now. Another advancement is the use of Cache-Control HTTP header that enables for cache policing.

It is a method used to specify browser caching policies in both client requests and server responses. It can include how the resource is cached, where it is being cached and the maximum age before cache expires. We can see three directives have been specified:

- no-cache
- must-revalidate
- max-age=0
database.php

In v3.0, login credentials are now defined in auth.php instead of the mysql backend. Notice that from this version on only the MD5 hash of the password is hard-coded in the php file: the password the user put in will get hashed and checked against this MD5 hash.

The two variables that are being evaluated are login and md5Password.

```
<?php

//math
$login = "admin";
$md5Password = "";

```

```
<?php

//error_reporting(0);
//ini_set("display_errors",0);

```

Hard-coded credentials in Baldr v3.0 panel vs. the mysql method used in v2.2

gate.php

The gate.php file - in version 3.0 - now includes more HTTP headers for catching real source IPs in case victims were connecting through VPNs or proxies. We can also see some new PHP environmental settings which were probably implemented for the event that the uploaded logs were far too big and the connection times out.

On the other hand, these settings might also put some additional strain on the hosting server in rare cases resulting in web-server crashes.
In the new version, `gate.php` code has changed quite a bit. We see the new delimiter: `~;~`, which job is to divide sections of the C2 response. The php function `explode` is used to do the separation.

In this new version, the panel operator can specify which threads of the stealer should run: these are the server settings. The operator can turn off fetching Telegram, Jabber logs, etc. features one by one. The variable `$statusSettings` will hold this information.

In v3.0 the C2 response includes grabber settings, which specifies what additional files the stealer should grab from the victim’s system. The variable `$grabber` will take this information.
The structure of the C2 response (v3.0):

And this is how it looks like decrypted:
In v3.0, we can also spot a new addition: in case the victim does not match any country (tries to look up country via a \textit{ip-api.com} lookup), the victim’s country is getting fetched from a local geo IP library that is included in the new version of the panel.

\begin{verbatim}
For ($crashes = 0; $crashes < 5; $crashes++) {
    try {
        $loc = json_decode(file_get_contents("http://ip-api.com/json/" . $ip), true);
        $country = $loc["country"];  
        $countryCode = $loc["countryCode"];
    } catch (Exception $e) {
        $country = "\text{ERROR}";
        $countryCode = "\text{ERROR}";
    }
    if ($country == "\text{ERROR}"{ 
        require_once("assets/GeoIP/geoloc.php");
        $country = ip_name($ip);
        $countryCode = ip_code($ip);
        $geoLocationString = GeoLocationString . "$\text{Country Code} : " . $countryCode . "$\text{\n}";
        $geoLocationString = GeoLocationString . "$\text{Country} : " . $country . "$\text{\n}";
    }
}
\end{verbatim}

\textit{New feature to identify victim’s geo-location}

\section*{Miscellaneous C2 components}

The \textit{viewer.php} did not change much in terms of the source code with the switch to 3.0, the only difference we see is the use of \textit{require_once} instead of the if clause seen in v2.2.

This further optimization allows for better performance on the panel: php will make sure that \textit{auth.php} is only included once and decides not to reload it once it is already included.
Panel inception (vulnerabilities)

The administrative panels lack effective security, and are vulnerable to a number of attacks themselves. We observed Baldr C2 servers that had been repeatedly taken over with web shells by other threat actors, who have been investigating and taking advantage of these panels.

![List of webshells found in Baldr panels](image)

We noticed at least one dark web forum post that reported a vulnerability in control panel code.
A customer conversing with Baldr's developer about a potential vulnerability in Baldr's panel code, who confirms it's fixed now

The vulnerability report linked to this image
The developer seems to pay attention to these reports, and continuously make their "product" as infallible as possible.

These vulnerabilities often stem from:

- misconfigured web-server settings, open directories
- default login credentials
- too open file permissions, browsable directories
- misusing other open protocols on the web-server
- local or remote file inclusion (LFI/RFI)

Mass-tracking Baldr panels

Before v3.0, it was relatively easy to find new panels in the wild: a well configured shodan.io search did the trick.

HTML code: `<title>Administrator log in | Baldr</title>` - [Shodan.io search](https://www.shodan.io)

Html tags specific to Baldr allowed us to track these panels at mass, but now with Basic Authentication getting introduced from v3.0, this method is not viable anymore.

This just goes to show that it is again a constant "cat-and-mouse" game between actors and security practitioners. Actors also evolve and learn from their previous mistakes, upping their game in the cybersecurity scene.

We do not currently know why the developer had changed the panel source code. It could be that the developer naturally had evolved the right operational security mindset and realized the potential for identifying Baldr panels via this method or it could even be that he was tipped off. Anyhow, we should all be aware of responsible disclosure. We should all refrain from releasing content that directly helps criminals and malware authors to fix bugs in their malware.

Our focus will remain at getting ahead of cybercrime actors.
**C2 communication**

v2.1

As of v2.1, the initial C2 reporting is done via a POST request with client information attached to the `http_uri` parameter. This serves as a fingerprinting mechanism. This also allows for easy detection through NSM, IDS are well equipped to catch this type of traffic. The POST request also contains a zip file which consists of the victim logs:

- information.log
- passwords.log

![TCP stream of the C2 communication for Baldr v2.1](image)

v2.2

v2.2 changed the game a bit, and the C2 communication is now encrypted with a XOR key issued from the C2 server. An initial beacon is received after sample contacts the C2 server with a simple GET request. The 4-byte long XOR key is then used to encrypt the system fingerprint that is sent along to the server, which has the same parameters as in v2.1. The malware then pushes the victim logs onto the server.
v3.0
We've seen further improvements with v3.0. There is the new separator: "\~;\~", which divides the initial beacon into certain sections. Since v3.0 the malware gets instructions from the C2 server itself regarding what to grab from the victim's machine whereas in v2.x it was hardcoded.
v3.0 also encrypts the victim logs with the 4-byte XOR key into a file called Encrypted.zip which is then uploaded to the C2 server via a POST request.

Assume one doesn't know XOR key. How do you figure it out? One unique characteristic of the XOR operation is that, once it hits 0x00 dwords and encrypts it with the key, you always get the actual XOR key back. So we did that.

The following python script extracts the plain text settings and the victim fingerprint using the 4-byte XOR key received from the server:
Baldr's use of bulletproof hosting

Contrary to the popular belief, these administrative panels usually stay online for weeks. The operators of these panels often come back and take down their panels only if they feel the heat.

In the time SophosLabs has observed the Baldr C2 traffic, we've seen certain C2 servers being taken down to upgrade panel version, or even enhance the overall operational security of their server, addressing vulnerabilities, patching certain holes. Or even changing tactics and exchanging a Baldr panel to Azorult for example.

However, all these countermeasures would be for nothing if it weren't for bulletproof hosting services. Usually we refer to hosting firms as bulletproof when the service they provide allows their customers for considerable tolerance in terms of hosted content. They achieve this tolerance by hosting clones of the malware on dozens of different IP addresses, and switch the domain's DNS to point at different IPs frequently.

The services of these bulletproof hosting companies are then being exploited by spammers, script kiddies and cybercrime operators for the hosting and the dissemination of adult pornography, black-hat content, carding, or in this case: hosting malware panels.

We are currently tracking just around ~250 Baldr panel installations. We broke down the stats and showcased the Top 3 most used bulletproof hosting and the distribution between those three.
The top 3 observed hosting providers for Baldr panels and their unique distribution

The second figure illustrates panel installations by upper-tier ISPs. The most offending ASN seems to be 200313, where about 1/5 of all Baldr panels are currently hosted.

It is not a bit unusual that the operators behind Baldr also recommend hosting providers. Without the bulletproof hosting, panels would fall very quickly.

For safe operations the distributor claims that the following 3 services are safe to use to do Baldr hosting, since bans are rare on these providers. These line up nicely with the observation we made about live Baldr panels.
There are several other hosting services listed to warn potential buyers not to use there: we can assume banning is a lot more frequent on these providers hence the disapproval.

Thank you for purchasing a license - BALDR Stealer & Loader. It will be useful to you.

This product is not recommended to be installed on such Host services:

- timeweb
- ooowebhost
- zzz com ua
- sprinthost
- hostinger
- hostia

We recommend such host services as: ******** | ******** | ********, at a moment, rare bans.

You will be able to reach your agreement. Remember, bans of course arrive very rarely, but once a year the cue-ball grows ...

Hosting services recommended by the Baldr developer/distributor

Tracking Baldr's buyer

In rare circumstances, the malware buyers also become victim to their own stealer, either by mistake or for testing purposes they execute the malware sample on their own machine. These rare occasions give some insight into how these crooks operate daily.

They often share samples with each other through Telegram with different cat memes on the side:
Baldr customers sharing Baldr samples between each other and interacting with Telegram chatbot

Their infrastructure usually resides on a bulletproof hosting. They rarely pay for the services, and so if it is an option at the hosting provider, they use the convenience of Trial accounts. This obviously meets their needs because each Baldr campaign usually only spans through a relatively short time, which is just perfectly done and coordinated from even Trial services.
The VPS serves the purpose of a malware playground. One of many activities we found is testing sample detection ratio with VirusTotal or with Specific AV vendor. In one case we found crooks testing a new Baldr sample against Sophos Anti-Virus and HitmanPro 3.8. We investigated the payment methods sections. Our presumption is that these are stolen victim credit cards.

**Telegram preferred for customer contact**

Baldr operation is mainly conducted on Telegram: focused around 3 individuals and 2 chatbots. To gain more information about the distributors, we looked at Telegram profiles too.

![Telegram profiles connected to the Baldr stealer](image)

No surprise that all 3 profiles - tied to Baldr distribution or development - proudly display their affiliation with the stealer. It seems at one point in time the chatbot associated with Baldr was deleted and a new one was created in its place. We do not know its reasons, but it might be possible that they are trying to cover tracks.

We found a second profile that seems to sell the stealer for a slightly lowered price: $140, which is odd considering all the forum posts mention the $150 price tag.

There are a few assumptions for this:

- either it was set up by the original distributor deliberately to gain more sales through a discounted 2nd profile
- there's an individual behind the profile who got a hold of the stealer's builder and is currently creating a sales rivalry for Baldr
- this profile serves as a honeypot

There's a second chatbot, which is strongly tied to Baldr's main distributor. Buyers can also top up their balance in virtual coins, specifically there's option to pay in BTC, LTC, ETH, ETC. The bitcoin addresses are generated on-the-fly at the time of purchase, so we cannot track what the sellers have earned.
**Baldr goes on hiatus?**

On the 31st of May, Baldr’s distributor released a short note both on Telegram and dark web forums.

The announcement came as a surprise for the stealer’s customer base, since they were expecting a rather large feature upgrade on that day (version 4.0).

However, Baldr’s main distributor points new customers to another stealer called Krypton and names it as a successor to follow in Baldr’s footsteps. Since this is relatively fresh news as of writing of this article, we will keep an eye on the outcome of this story and will follow progression.

Sophos will add new detections as different iterations of the malware appears.

Just as Baldr was on the road to take up some space in the cybercrime ring that for example AZORult's demise left behind, it seems Baldr will take the fall now due to internal rivalry.
Appendix: Indicators of Compromise

Files analyzed in this report

<table>
<thead>
<tr>
<th>Type</th>
<th>SHA256:</th>
<th>Detection:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rtf decoy</td>
<td>2290b4ae47189e2505c88435e34c89d427ef578b8e8bcd77ee42a0a006410836</td>
<td>Exp/201711882-A</td>
</tr>
<tr>
<td>Rtf payload</td>
<td>5fa915ad3471a9f0f7532ae034c93c8c5faaf8c73f7c99e7bbdd221c59b78217</td>
<td>Mal/Generic-S</td>
</tr>
<tr>
<td>Ace archive</td>
<td>b7eebb795ee41bd32191b1600653fb26a64c4c31f3cad453fa4d9eb30326acfb</td>
<td>Mal/Generic-S</td>
</tr>
<tr>
<td>Ace payload</td>
<td>3dfe961387852c9d8869fd5fd32aa8000df27387ec2a7e1f2b0742ea4c109a95</td>
<td>Mal/Generic-S</td>
</tr>
<tr>
<td>Unobfuscated</td>
<td>238d4f90580e64b3af1d06db9cb79aae4d3e355fe54727e029fc9a3083b56f4f</td>
<td>Mal/Generic-S</td>
</tr>
<tr>
<td>Unobfuscated</td>
<td>43fccc81880411712570d1a33b52d7fba1df9c2cb1d09fc9b9c72d737a71231db7</td>
<td>Mal/Generic-S</td>
</tr>
</tbody>
</table>

Additional Indicators of Compromise:

A complete list of Baldr IoCs are available through our official Github repository.