



## Overview

**BluStealer** is a crypto stealer, keylogger, and C# .NET hack tool to steal credentials. This malware was first discovered in 2019 and referred to as [a310logger](#). In fact, [a310logger](#) is just one of the components that appeared in the string artifacts of this malware, and recently it is mentioned as "fresh malware", and recently it is mentioned to go with the BluStealer naming while preserving its inner workings.

[a310logger](#) is just one of the components that appeared in the string artifacts of this malware, and recently it is mentioned to go with the BluStealer naming while preserving its inner workings.

BluStealer is primarily spread through malicious emails that come from a particular campaign that is currently active. The analysis of this loader is provided in this post. The first sample is a fake DHL invoice in English. The second sample is a fake invoice from a metal company, in Spanish. Both samples contain a lure message that claims the user has won a prize. The attachments contain the malware executable.

In the graph below, we can see a significant increase in the number of samples starting around October 10-11, 2021.

The daily amount of samples is shown in the graph below.

## BluStealer Analysis

As mentioned, BluStealer consists of a combination of several components. Both components vary greatly among the samples, so I will analyze each component separately. The VB component is the most interesting, hence the inclusion of "SpyEx" strings. It has added the capabilities to steal credentials from the clipboard, find and upload document files, and use various anti-analysis/anti-VM tactics. On the other hand, the VB component is a stealer that is patched together from a combination of existing malware samples, such as [ThunderFox](#), [ChromeRecovery](#), [StormKitty](#), and [DarkSide](#), all of which are available in a single sample.

## Obfuscation

## Menu

- Mobile
- Network
- PC
- IoT
- Careers

Type here to search...



## Categories

( [EVENTS](#) ) ( [IOT](#) ) ( [MOBILE](#) ) ( [NETWORK](#) ) ( [PC](#) ) ( [UNCATEGORIZED](#) )

## Tags

ALT-STORE ANALYSIS ANDROID APT BACKDOOR BANKER  
CRYPTER CRYPTOMINING CRYSTAL BALL CSRF CVE DIRTYMOE  
GOOGLE PLAY STORE HIDING HW IOT MALWARE MISC  
PHISHING PREDICTIONS RANSOMWARE RAT RESEARCH REV  
SONARDNS SPYWARE STEALER THREED  
THREE-DIMENSIONAL

## Recent Posts

[BluStealer: from SpyEx to ThunderFox](#)

[DirtyMoe: Code Signing Certificate](#)

[Research shows over 10% of sampled Firebase instances open to the public](#)

[DirtyMoe: Rootkit Driver](#)

[Magnitude Exploit Kit: Still Alive and Kicking](#)

## Archive

September 2021  
August 2021  
July 2021  
June 2021  
May 2021  
April 2021  
March 2021  
February 2021  
December 2020  
November 2020  
October 2020  
September 2020  
August 2020

Each string is encrypted with a unique key  
the xor cipher, RC4, or the [WinZip AES](#) im-  
tion of the custom AES algorithm:

A utility to help decrypt all strings in IDA is:

## Anti-VM Tactics

BluStealer checks the following conditions

If property **Model** of **Win32\_ComputerSystem**:

**VIRTUA** (without L), **VMware Virtual Platform**,  
**vmw**

If property **SerialNumber** of **Win32\_BaseBoard**:

If the following files exist:

- C:\Windows\System32\drivers\vhci.dll
- C:\Windows\System32\drivers\vmhgfs.dll
- C:\Windows\System32\drivers\vmnet.dll
- C:\Windows\System32\drivers\vmrd.dll
- C:\Windows\System32\drivers\VBoxGuest.dll
- C:\Windows\System32\drivers\VBoxNetAdp.dll
- C:\Windows\System32\drivers\VBoxNetFlt.dll
- C:\Windows\System32\drivers\VBoxVirt.dll

If any of these conditions are satisfied, Blu

## .NET Component

The BluStealer retrieves the .NET payload  
using the WinZip AES algorithm using a hardcoded  
utility to launch the .NET executable(s):

- C:\Windows\Microsoft.NET\Microsoft.WindowsAPICodePack
- C:\Windows\Microsoft.NET\Framework\v4.0.30319\mscor2.dll

Examples of two .NET executables loaded by Blu

The .NET component does not communicate with  
popular browsers and applications then writes the  
filename (i.e credentials.txt). The VB core  
is better explained in the next [section](#).

The .NET component is just a copy/pasta of  
information on their respective Github pages:

- ThunderFox: [github.com/V1V1/ThunderFox](https://github.com/V1V1/ThunderFox)
- ChromeRecovery: [github.com/ElysianHQ/ChromeRecovery](https://github.com/ElysianHQ/ChromeRecovery)
- StormKitty: [github.com/swagkarna/StormKitty](https://github.com/swagkarna/StormKitty)

August 2020

June 2020

May 2020

April 2020

December 2019

September 2019

August 2019

July 2019

April 2019

March 2019

February 2019

January 2019

August 2018

January 2018

October 2017

## Meta

[Log in](#)

[Entries feed](#)

[Comments feed](#)

[WordPress.org](#)

- Firepwd:github.com/lclevy/firepwd

## Information Stealer

Both the VB core and the .NET component plates folder. Each type of stolen data is core sets up different timers to watch over size increases, the VB core will send it to th

Handler	Arbitrary file
.NET component	credentials
.NET component	Cookies.z
VB Core	CryptoWall
VB Core	FilesGra&ber\\Files.z
VB Core	Others

BluStealer VB core also detects the crypto the attacker's predefined ones. Collective cash, Ethereum, Monero, Litecoin.

## Data Exfiltration

BluStealer exfiltrates stolen data via SMT of server-side code. The Telegram token sendDocument and sendMessage as shov

- [https://api.telegram.org/bot\[BOT\]\[MY\\_MESSAGE\\_TEXT\]](https://api.telegram.org/bot[BOT][MY_MESSAGE_TEXT])
- [https://api.telegram.org/bot\[BOT\] TO \[MY\\_CAPTION\]](https://api.telegram.org/bot[BOT] TO [MY_CAPTION])

## .NET Loader Walkthrough

This .NET Loader has been used by families such as Stealer, RedLine, as well as BluStealer.

Demo sample: 19595e11dbccfbfeb9560e1

## First Stage

The first stage of the .NET loader has a generic name: 'Microsoft .NET Obfuscator'. However, one resource contains a decrypted module in the resource:

By looking for this module's reference within the resources, we can see it has been loaded into memory as shown below:

Prior to loading the next stage, the loader will register itself in the Windows registry through the Startup folder and registry run keys:

- C:\Users\\*\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup
- HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Run
- C:\Users\\*\AppData\Roaming\Microsoft\Windows\Run
- HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Run
- HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce
- HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce
- C:\Users\\*\AppData\Roaming\Microsoft\Windows\RunOnce

In the samples we looked at closely, the 'Run' key is obfuscated by a string provider function that performs a tail jump that resides within the same narrow cases, it usually is the call to the external file examples:

## Second stage

Inside the Data() function of the second stage:

The second stage has the function calls a lot of assembly code that is not very helpful. However, there are two resources that are useful. Their getter functions can be easily found by setting a breakpoint on the Ehiuvbfrnprkuyuxc and it is gzip decompressed revealing a PE file.

Ehiuvbfrnprkuyuxc

On the other hand, the breakpoint on the

leaves us in the middle of the Data() function, which is part of the CompareComparator function that we will analyze.

#### Comparing Method Tokens

In order to understand what is going on, we can look at the experience working with [MassLogger](#) in the resource section, which in this case, is the same.

Note that it is important to find out whether the obfuscation process gets us directly to the designated function or through the field-method binding process. So when the solver.GetMethodInfo() will be called to build the method token map file, the modification only consists of replacing some tokens. Since the obfuscation is the same, it is sufficient enough to obfuscate the method token map file.

#### Method Token Map File

The search of the "Dic Attr" string leads us to the method token map file. This value represents the method token that we are looking for. For the method tokens start with 0x4A, just two are the chosen ones to be modified for obfuscation.

With all the function calls revealed, we can start the analysis. First, it loads a new assembly that is the SmartAssembly assembly. An instance of an object named SmartAssembly is created. A method called "RegisterSerializer" is invoked on this object. At this point, we can assume that the purpose is to register the assembly file and execute it.

Heading back to the newly loaded assembly (assembly token: 0x13649d15f5c7d36b69274a76efdac), we can see the application features labeled as shown below.

#### Overview of the decompressed Ehiuvbfrnprkuyuxqv.0x13649d15f5c7d36b69274a76efdac

The unpacking process will not be much different from the previous code virtualization. From static analysis, we can see that the SmartAssembly.Queues class is instantiated.

#### Instantiating the Queue Class

The function content after instantiated.

#### Fast forward to where the function is called

Lucky for us, the code looks fairly straightforward. The token 0x00000000 is passed to a function that RulesListener::GetManifestResourceStream. This function takes the field token to method token map file we created earlier and searching for the GetManifestResourceStream function, where the map is established:

#### The resource file Params.h

RulesListener.IncludeState has token 0x00000000. In this function, the decryption algorithm is run.

#### Data from Local

Then it

In fact, **all** the samples can be unpacked saved **in the second stage**. Hopefully, ever will remain useful at showing you how to c

## Conclusion

In this article, we break down BluStealer file to extract its IOCs. We also highlight its code to save data to disk and without a proper C2 function. In the half of the blog, we show how the BluStealer uses a unique .NET loader. With these insights, we can begin writing and analyzing BluStealer.

### IOCs:

The full list of IOCs is available at [https://github.com/BluStealer/BluStealer/blob/main/BluStealer\\_IOC.md](https://github.com/BluStealer/BluStealer/blob/main/BluStealer_IOC.md)

## BluStealer

### SHA-256

678e9028caccb74ee81779c5dd6627fb6f  
3151ddec325ffc6269e6704d04ef206d621  
49da8145f85c63063230762826aa8d85d  
7abe87a6b675d3601a4014ac6da843924

### Crypto Address List

#### Bitcoin:

1ARtkKzd18Z4QhvHVijrVFTgerYEoopjLP  
1AfFoww2ajt5g1YyrrfNYQfKJAjnRwVUsX  
1MEf31xHgNKqyB7HEeAbcU6BhofMdwl  
38atNsForzrDRhJoVAhyXsQLqWYfYgodd  
bc1qrjl4ks5h7p70jftypr8s6cjpngzd3kerfj:  
bc1qjg3y4d4t6hwg6h22khknlxstevjg2qk  
1KfRWVcShzwE2Atp1njogAqH8qodsf3p  
3P6JnvWtubxbCwgPW7GAAj8u6CLV2h9I  
13vZcoMYRcKrDRDYUyH9Cd4kCRMZVjFl

#### Bitcoincash:

qrej5ltx0sgk5c7aygdsvt2gh7fq04umvusxl  
qrzakt59udz893u2uuwtgrwrjj9dhtk0gc3n

#### Ethereum:

0xd070c48cd3bdeb8a6ca90310249aae9  
0x95d3763546235393B77aC188E5B08d  
0xcfE71c720b7E99e555c0e98b725919B7

#### Monero.address:

46W5WHQG2B1Df9uKrkyuhoLNVtJouMfl  
wYRXhBYVjLRbpDu8CK2UN2xzenr  
43Q4G9CdM3iNbkwujAQJ7TedSLxYQ8h  
caQeoSF86eFE2fL9njU59dQRfPHFnv

#### Litecoin address:

LfADbqTZoQhCPBr39mqQpf9myUiUiFrDE

LY5jmjdFnvgFjJET2wX5fVV6Gv89QdQRv:

## Telegram Tokens:

1901905375:AAFoPAvBxaWxmDiYbdJW  
1989667182:AAFx2Rti45m06IscLpGbHoE

## SMTP

andres.galarraga@sismode.com (smtp.1a...  
info@starkgulf.com (mail.starkgulf.com )  
etopical@bojtai.club (mail.bojtai.club)  
fernando@digitaldirecto.es (smtp.ionos.es)  
baerbelscheibll1809@gmail.com  
dashboard@grandamishabot.ru (shepherd...  
shan@farm-finn.com (mail.farm-finn.com)  
info@starkgulf.com (mail.starkgulf.com)

## .NET Loader SHA-256:

ae29f49fa80c1a4fb2876668aa38c8262d  
35d443578b1eb0708d334d3e1250f6855  
097d0d1119fb73b1beb9738d7e82e1c73c  
c783bdf31d6ee3782d05fde9e87f70e9f3c  
42fe72df91aa852b257cc3227329eb5bf4  
1c29ee414b011a411db774015a98a8970  
81bbcc887017cc47015421c38703c9c261  
6956ea59b4a70d68cd05e6e740598e76e  
6322ebb240ba18119193412e0ed7b325c  
9f2bfedb157a610b8e0b481697bb28123e  
e2dd1be91c6db4b52eab38b5409b39421

Tagged as

Cryptostealer

Malware Analysis

Share:



Furth

( PC )

| DirtyMoe: Code Signing Certificates

26 min read

The Windows kernel allows loading drivers signed with code signing certificates that are moreover widely abused through various mechanisms....

{ PC }

## DirtyMoe: Rootkit Driver

38 min read

The DirtyMoe malware is a complex malicious backdoor that uses a rootkit to hide its presence and avoid detection by security software. One of the more significant safeguards is a rootkit. This allows the malware to modify system files and registry keys without being noticed by standard security tools.

2021 Co