Malware Analysis Report

ImageExfil Malware

Oct 2024 | Ra-Sec | v1.0

Table of Contents

Executive Summary

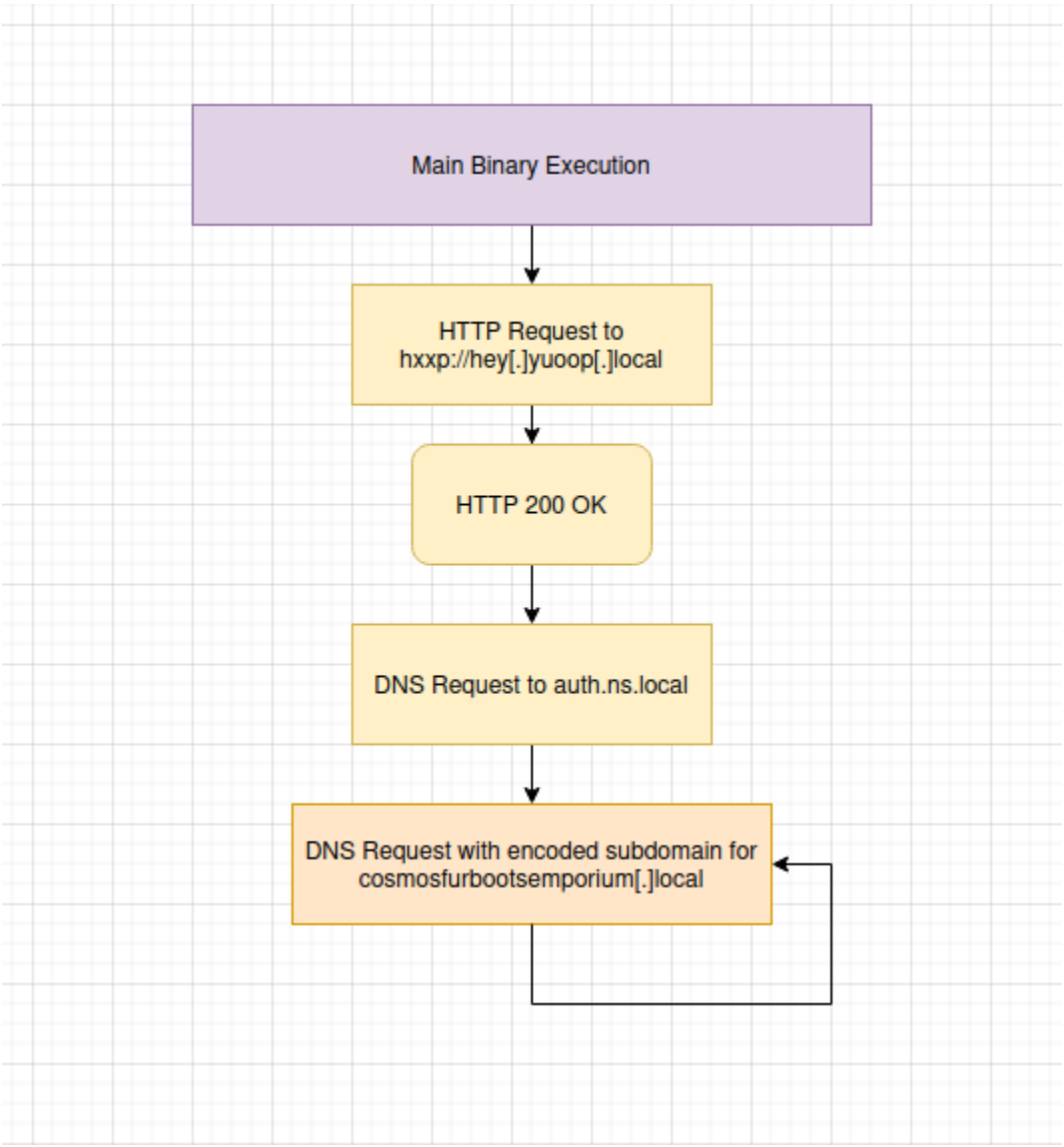
SHA256 hash	81A10784AE60A58A969E858C9C4A2AE0D4EBE46E9BD6776992461C062F70099D

ImageExfil Malware is a exfiltration malware sample first identified on 19th Oct 2024. It is a Nim compiled binary that run on 32/64 bit Windows operating system. It consists of one payload only which when is able to ping to a specific domain, starts with the exfiltration of cosmo.jpeg that is stored on the desktop of the system. The URLs are listed in Appendix B. A lot of encoded DNS requests are done from the endpoint and it seems to be in a constant loop.

YARA signature rules are attached in Appendix A.

High-Level Technical Summary

The binary consists of one payload. That is designed in such a way that it will exfiltrate cosmo.jpeg file that is stored on the desktop of the endpoint. If it is unable to find the file, it will simply not work. Initially it attempts to connect to a URL hey[.]youup[.]com. Once the response from the server is acquired, it sends a DNS query for auth[.]ns[.]local. Immediately after the query, it starts sending encoded DNS queries for the domain encodedtext[.]cosmosfurempprorium[.]local.



Malware Composition

DemoWare consists of the following components:

File Name	SHA256 Hash
ImageExfil.exe	81A10784AE60A58A969E858C9C4A2AE0D4EBE46E9BD6776992461C062F70099D

ImageExfil.exe

The initial executable that runs all the operations after the initial execution on the endpoint

Basic Static Analysis

property	value
file	
file > sha256	81A10784AE60A58A969E858C9C4A2AE0D4EBE46E9BD6776992461C062F70099D
file > first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
file > first-bytes-text	M Z . . . . . @ . . . . .
size	123392 bytes
entropy	6.402
file > type	executable
cpu	32-bit
subsystem	GUI
version	n/a
description	n/a
entry-point > first-bytes-hex	83 EC 0C C7 05 D4 22 42 00 01 00 00 00 E8 CE 7F 01 00 83 C4 0C E9 A6 FC FF FF 8D B6 00 00 00 00 83
entry-point > location	0x000014A0 (section[.text])
signature | tooling	Nim Compiler

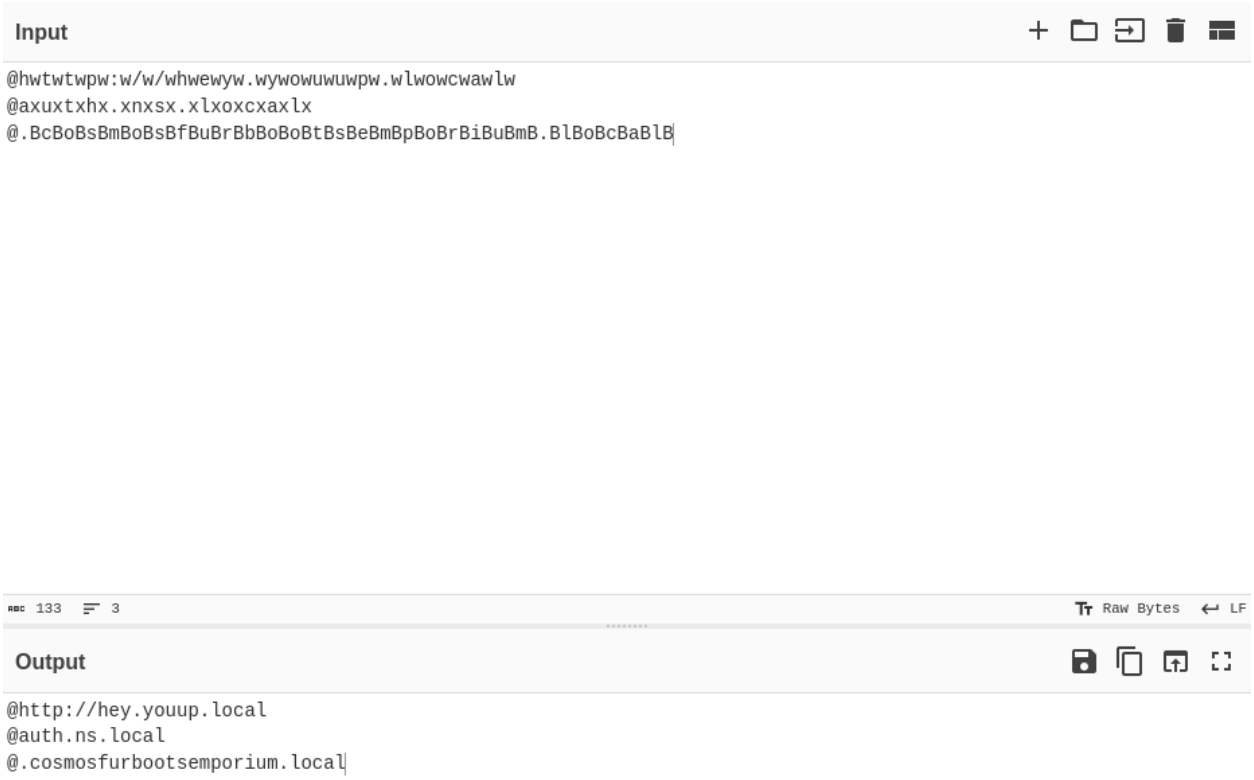
This shows that the PE is a 32-bit executable. And the MZ magic byte shows that it is indeed a PE file.

The size of the file is rather small so it is highly likely that it is not compiled in GO

```
@https
@No uri scheme supplied.
@Desktop\cosmo.jpeg
@200 OK
@Authorization
@Host
@httpclient.nim(1144, 15) `false`
@Transfer-Encoding
@Content-Type
@Content-Length
@httpclient.nim(1082, 13) `not url.contains({'\r', '\n'})` url shouldn't contain any newline characters
@Nim httpclient/1.6.2
@hwtwtwpw:w/w/whwewyw.wywowuwuwpw.wlwowcwawlw
@axuxtxhx.xnxsx.xlxoxcxaxlx
@.BcBoBsBmBoBsBfBuBrBbBoBoBtBsBeBmBpBoBrBiBuBmB.BlBoBcBaBlB
Unknown error
_matherr(): %s in %s(%g, %g)  (retval=%g)
Argument domain error (DOMAIN)
Argument singularity (SIGN)
```

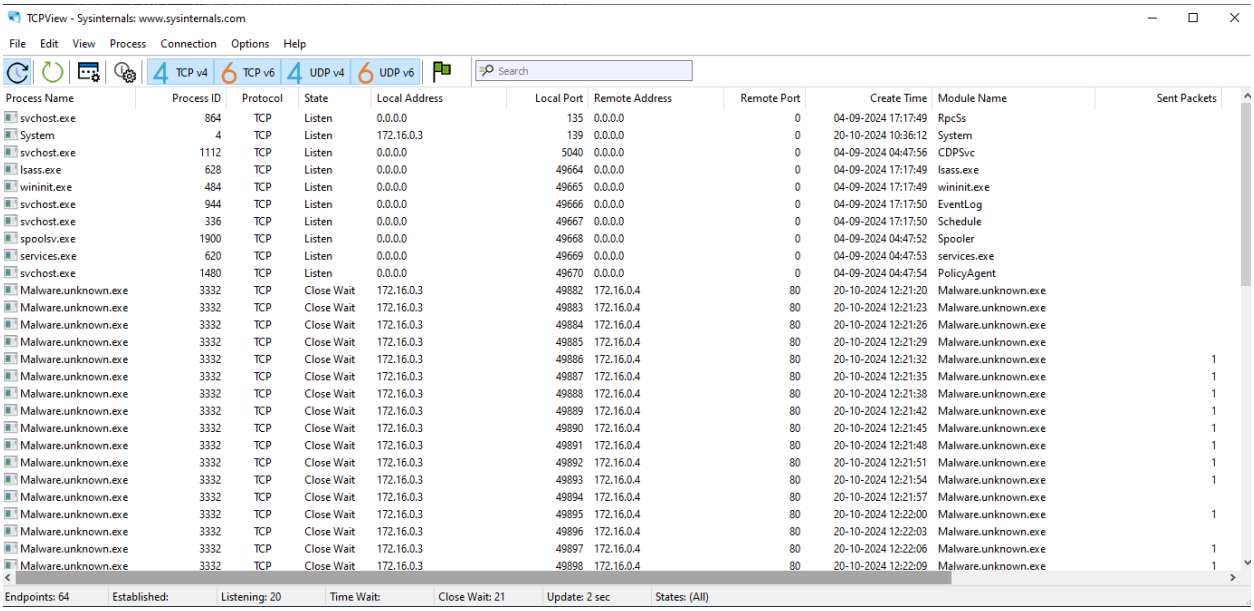
The strings output gives out a good overview of the activity of the binary. The highlighted section gives a lot of confidence to the fact that we will be seeing some HTTP traffic. In the third row, we can see that the specific file is also mentioned. In this case it is cosmo.jpeg.

The URLs have been contaminated with a few characters which can be easily fixed.

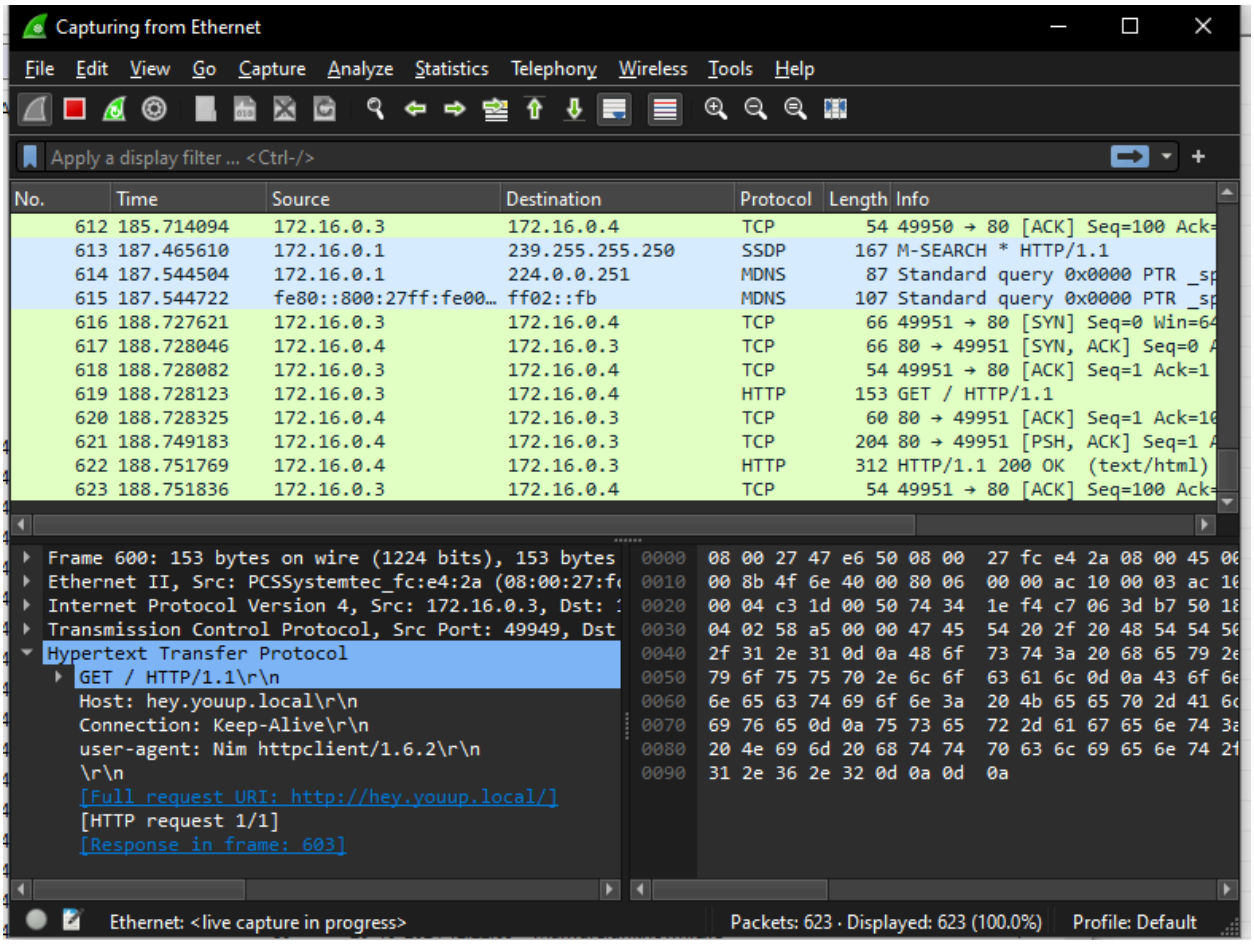


Here we can see the URLs in clean sight.

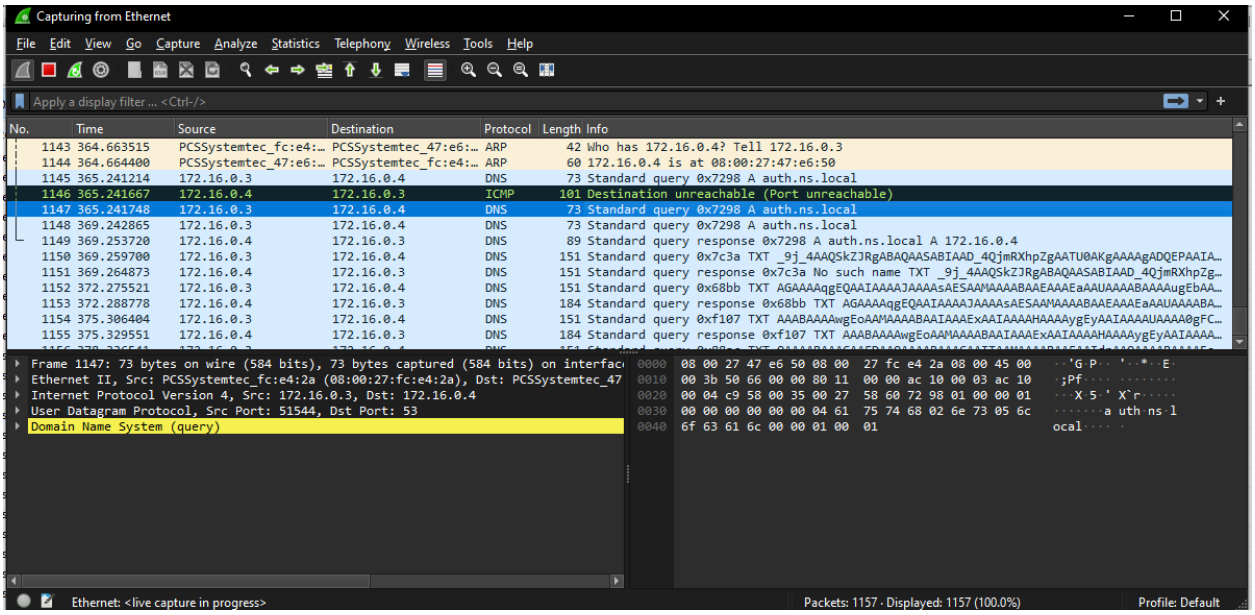
Basic Dynamic Analysis



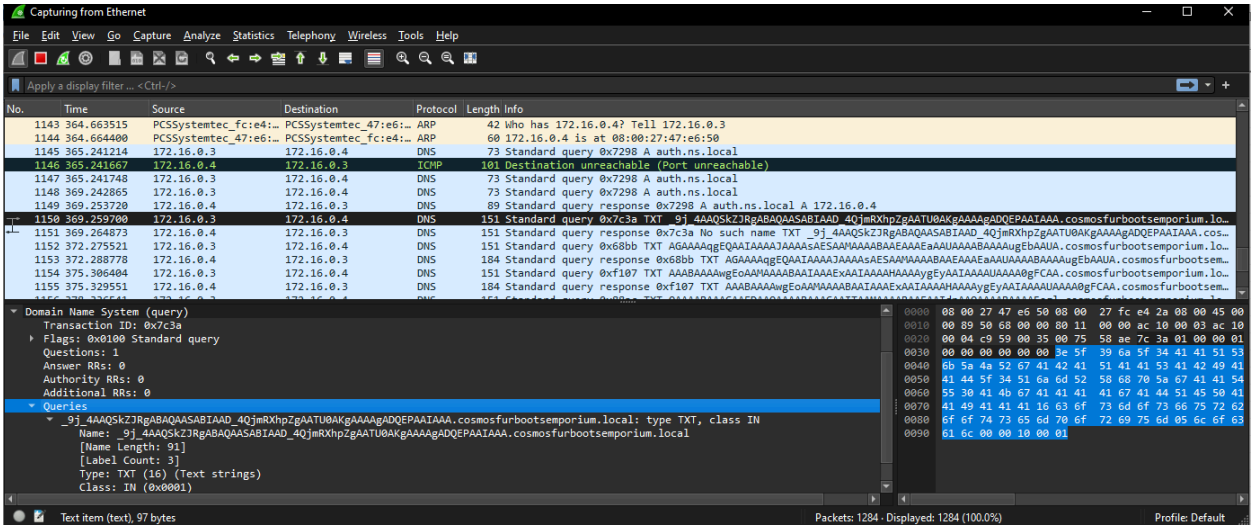
Upon execution initially we do not see any window and we can see it create requests to port 80 on the remote server. It continues to do so for each subsequent port.



On wireshark, we can see it repeatedly sending HTTP requests to the initial domain. To move it to the next stage, we have to somehow disrupt the outbound HTTP requests. It is as simple as disconnecting it from the internet for a few seconds.



After a simple disconnect and reconnect, we can see it query the second domain. And right after the response from the DNS server, we can see that it begins to exfiltrate the data using DNS queries.

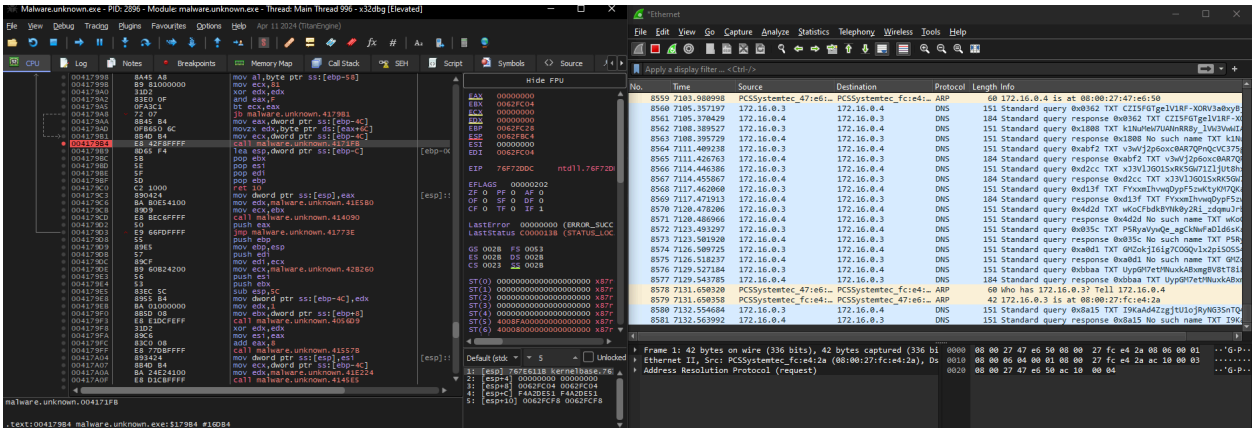


Advanced Static Analysis

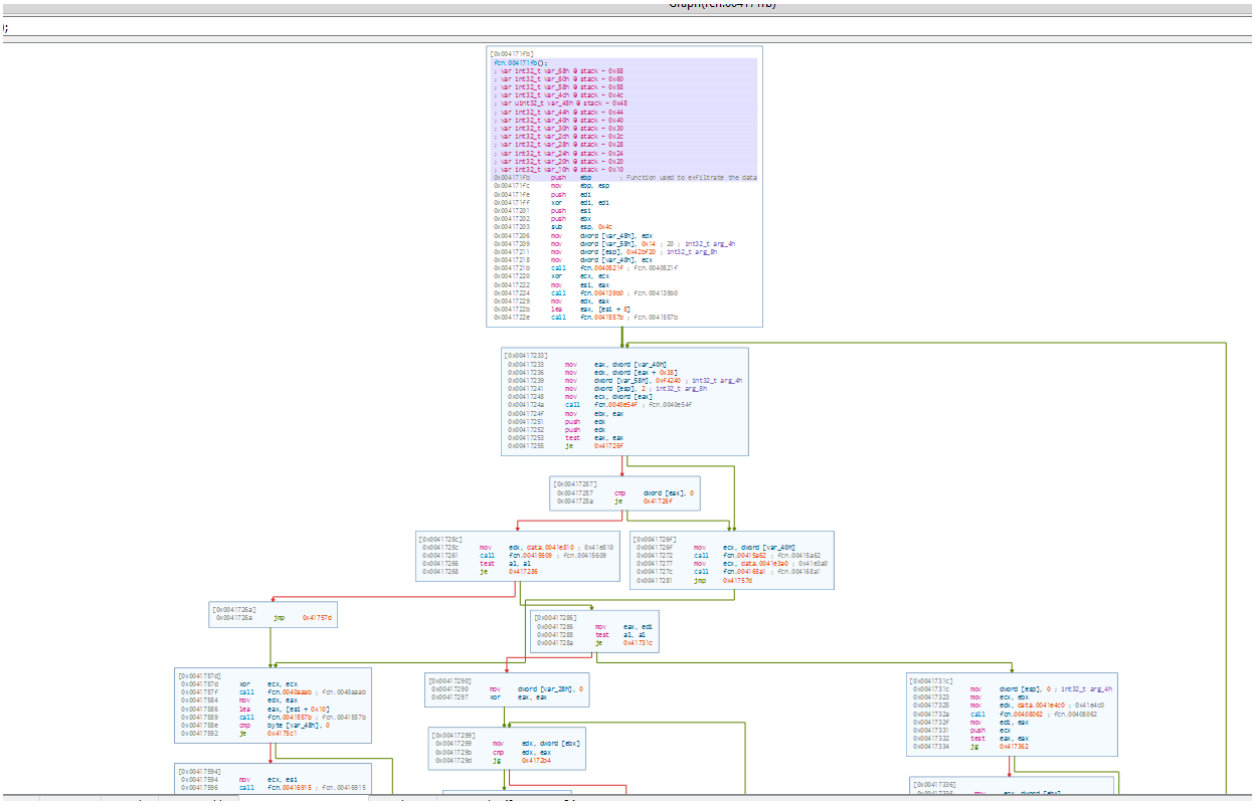
```
[0x0041aaa0]
int main(int argc, char **argv, char **envp);
; var int32_t var_1ch @ stack - 0x1c
; var int32_t var_18h @ stack - 0x18
; var int32_t var_14h @ stack - 0x14
; var int32_t var_ch @ stack - 0xc
; arg int argc @ stack + 0x4
0x0041aaa0      lea     ecx, [argc]
0x0041aaa4      and     esp, 0xfffffff0
0x0041aaa7      push    dword [ecx - 4]
0x0041aaaa      push    ebp
0x0041aaab      mov     ebp, esp
0x0041aaad      push    ecx
0x0041aaae      sub     esp, 0x14
0x0041aab1      call    fcn.00419440 ; fcn.00419440 ;  fcn.00419440(void)
0x0041aab6      mov     eax, dword [section..data] ; 0x41b000
0x0041aabb      mov     dword [var_1ch], 0
0x0041aac3      mov     dword [var_14h], eax
0x0041aac7      mov     eax, dword [0x422318]
0x0041aacc      mov     dword [var_18h], eax
0x0041aad0      mov     eax, dword [0x42231c]
0x0041aad5      mov     dword [esp], eax
0x0041aad8      call    fcn.00419128 ; fcn.00419128
0x0041aadd      mov     ecx, dword [var_ch]
0x0041aae0      sub     esp, 0x10
0x0041aae3      leave
0x0041aae4      lea     esp, [ecx - 4]
0x0041aae7      ret
```

Main Function

Advanced Dynamic Analysis



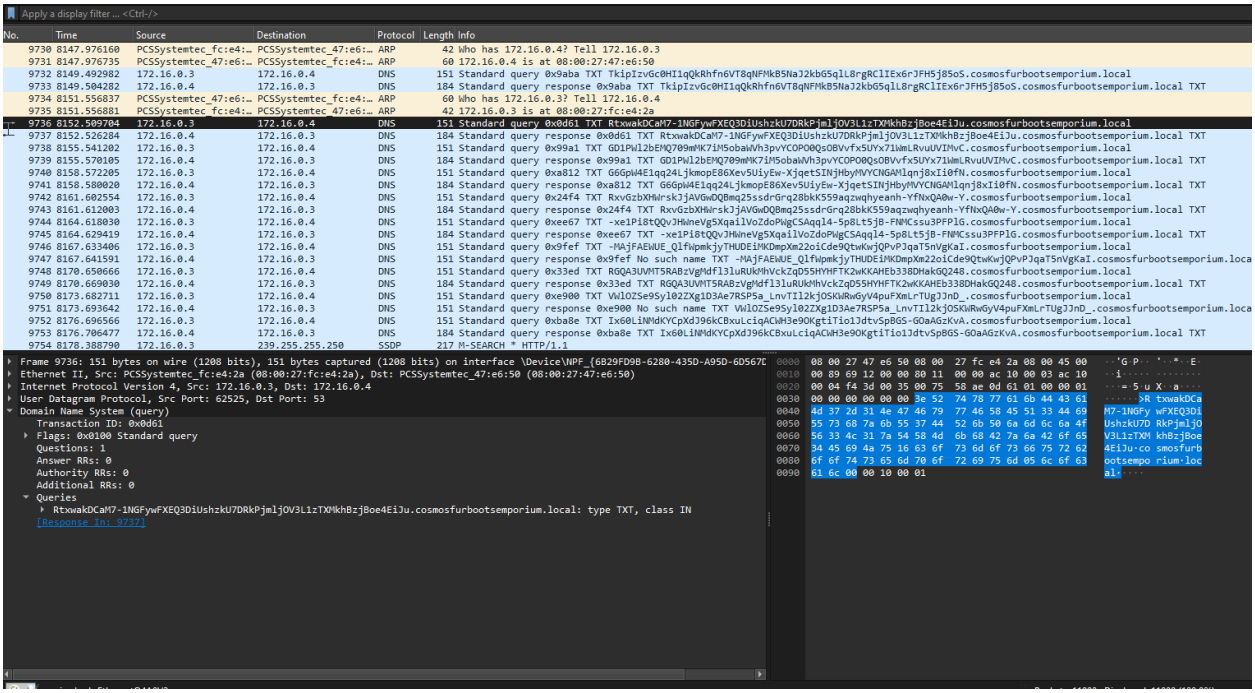
The particular function that started exfiltrating the data as can be seen from the wireshark output and a debugger.



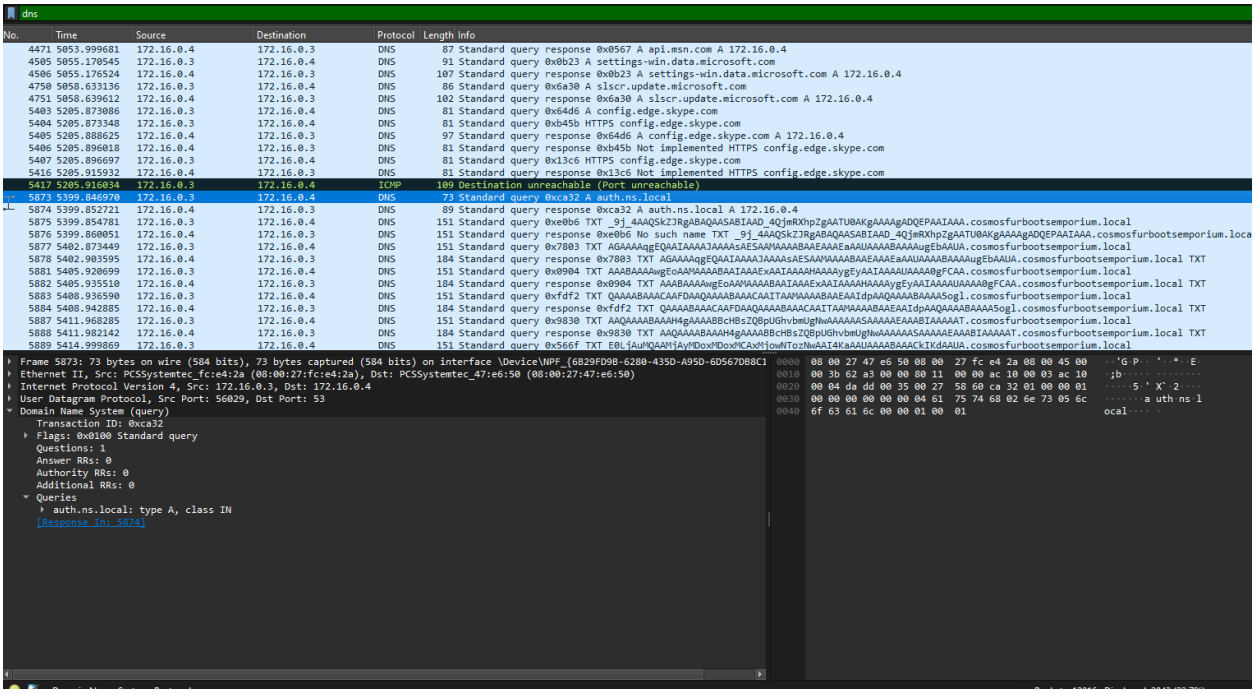
Indicators of Compromise

The full list of IOCs can be found in the Appendices.

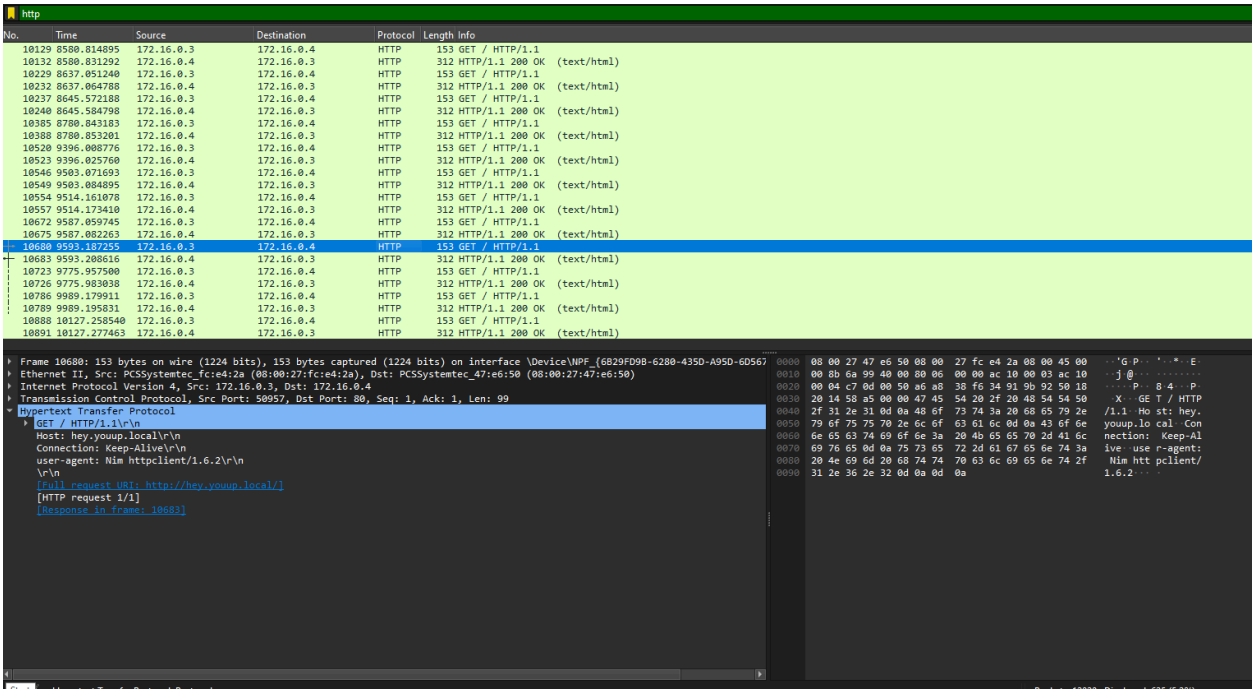
Network Indicators



DNS Requests for cosmosfurbootsemporium.local with encoded data as subdomain



The DNS request to the auth[.]ns[.]local domain after the HTTP traffic was disrupted



The initial HTTP requests to the domain

Rules & Signatures

A full set of YARA rules is included in Appendix A.

Appendices

A. Yara Rules

```
rule Yara_ImageExfil {
    meta:
        last_updated = "2024-11-02"
        author = "RA"
        description = "A Yara rule for ImageExfil Malware"
    strings:
        $string1 = "cosmo.jpeg"
        $string2 = "n1m"
        $string3 = "@hwtwtwpw:w/w/whwewyw.wywowuwuwpw.wlwowcwawlw"
        $string4 = "@axuxtxhx.xnxsx.xlxoxcxaxlx"
        $string5 = "@.BcBoBsBmBoBsBfBuBrBbBoBoBtBsBeBmBpBoBrBiBuBmB.BlBoBcBaBlB"
        $PE_magic_byte = "MZ"
    condition:
        $PE_magic_byte at 0 and
        ($string1 and $string2 and $string3 and $string4 and $string5)
}
```

B. Callback URLs

Domain	Port
hxxps://hey.yuoop.local	80
hxxps://auth.ns.local	53
hxxp://cosmosfuremoprium.local	53