Exploit Vulnerability Report

Research: Octuber 2017

Targets: Audacious 3.9, QQPlayer 3.9, VLC 2.2.6

VerSprite

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Alejandro Parodi** | | |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Audacious 3.9 – StackOverflow** | | |
|  |  |  |

**. Finding:**

This vulnerability was found through fuzzing .aac files with CERT BFF FUZZING FRAMEWORK after a deeper analysis of file parser libraries used by the program.

**. Impact:**

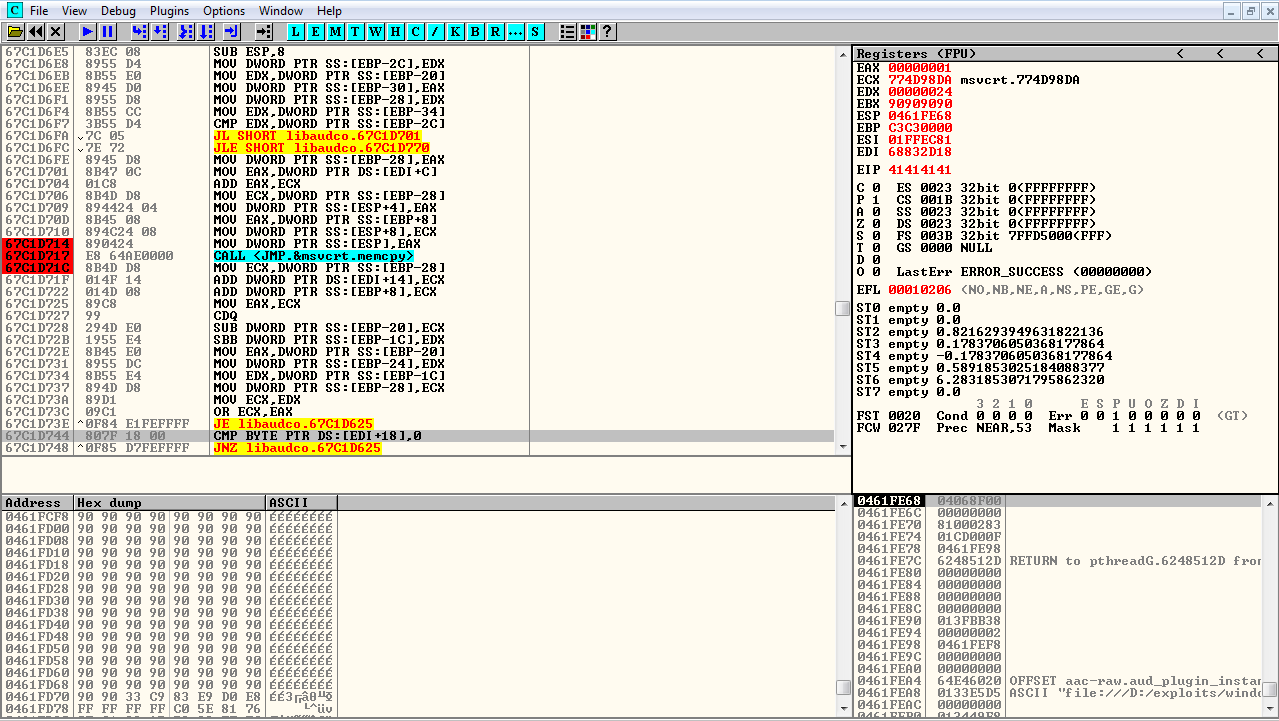
The vulnerability found in this application deal in code execution with the privileges of the user that run Audacious Player via local file or remot http file server.

**. Requeriments:**

The vulnerability can be triggered in all Windows versions and probably in Linux environments too (The PoC was build to run over Windows 8 and Windows 10 to a working exploit in Win7 only is necessary change some gadget offsets).

**. Details:**

The vulnerability exist in the library libaudco where a recursive call of memcpy derive in EIP overwriting.  
This happen due that the memcpy function is called inside a loop, this loop try to find an especific delimiter to break the recursive function, the malformed file does not contain this delimiter so the execution cannot escape of the loop and after a serie of execution with an especific file length is posible controll EIP, EDI, ESI, EBP without overwrite SEH Chain and get control of the execution flow of the application.

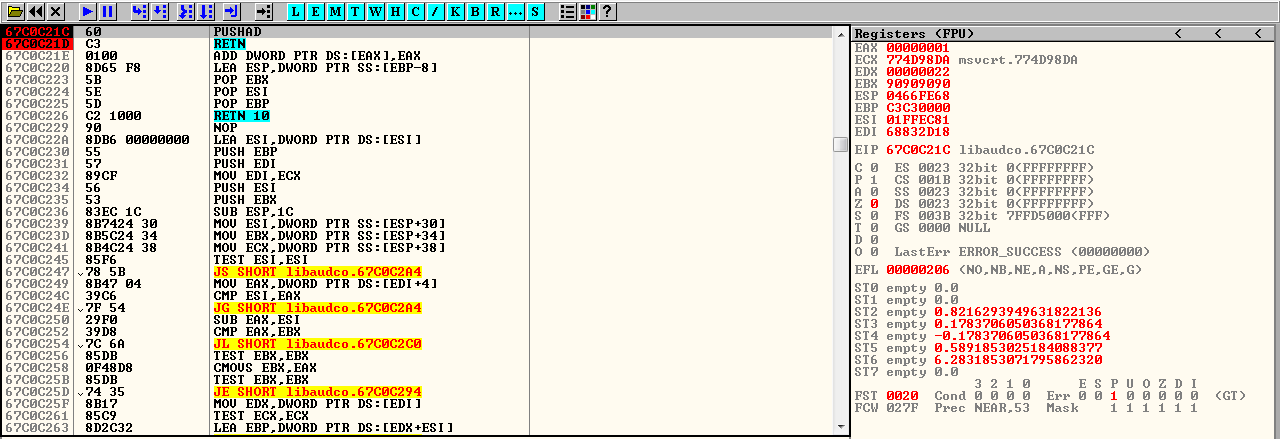
**

Vulnerable Memcpy Function call.

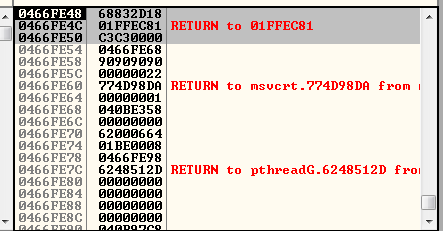
It is important to highlight that different kinds of modules in the application including the Main module are compiled without ASLR and without DEP, this allow an attacker to write the nopsleed+shellcode inside of the stack but the application control can be manipulated out of the main application thread because of this is imposible jump directly over a stack address.

Other important complication is that the last 4 bytes in the malformed file are the ones that overwrite EIP and 1 byte more of the necessary to overwrite EIP end in a SEH Overwriting and make imposible the execution in Windows 8 and later. Also the shellcode is sitting above of the EIP overwriting so is necesario a negative stack pivot in order to achive code execution.

To overcome this problem is necessary build an appropiate ROP Chain using as first gadgen a **pushad ret**, this gadget allow to push in the stack the controlled values of EBP, ESI, EDI and use those 3 address to start our ROP Chain witouth overwrite SHE Chain.

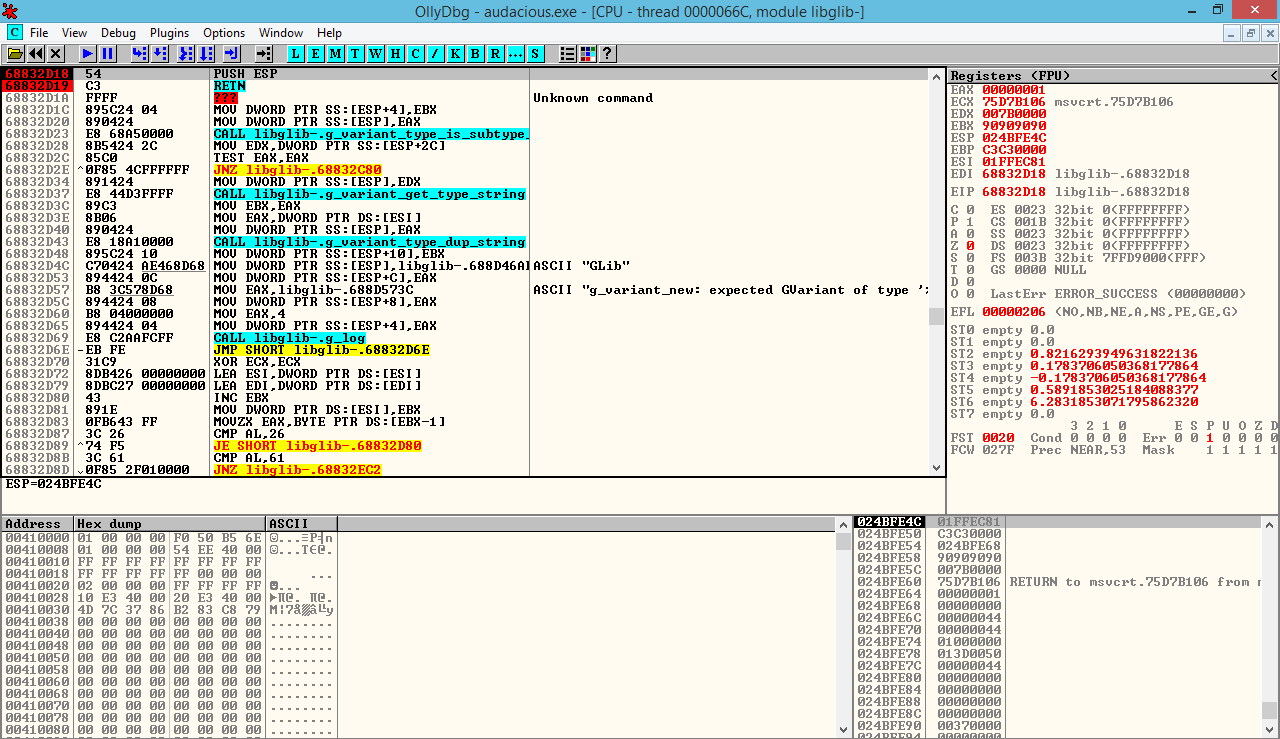


First Gadget to Rop.

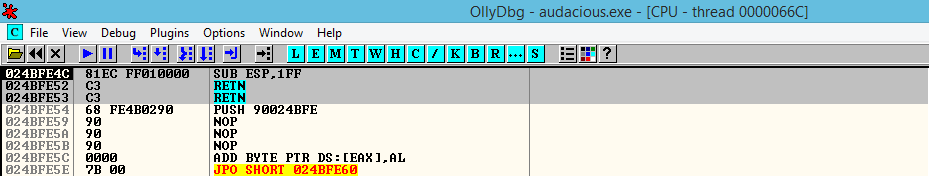


Stack manipulation with EDI, ESI and EBP (ESI+EBP = sub esp, 0x1ff).

The second gadget of the ROP chain **(0x68832D18)** execute **push esp, ret** to set the execution context in the current stack and using the lack of DEP is posible harcode the negative stack pivot (**sub esp, 0x1ff**).

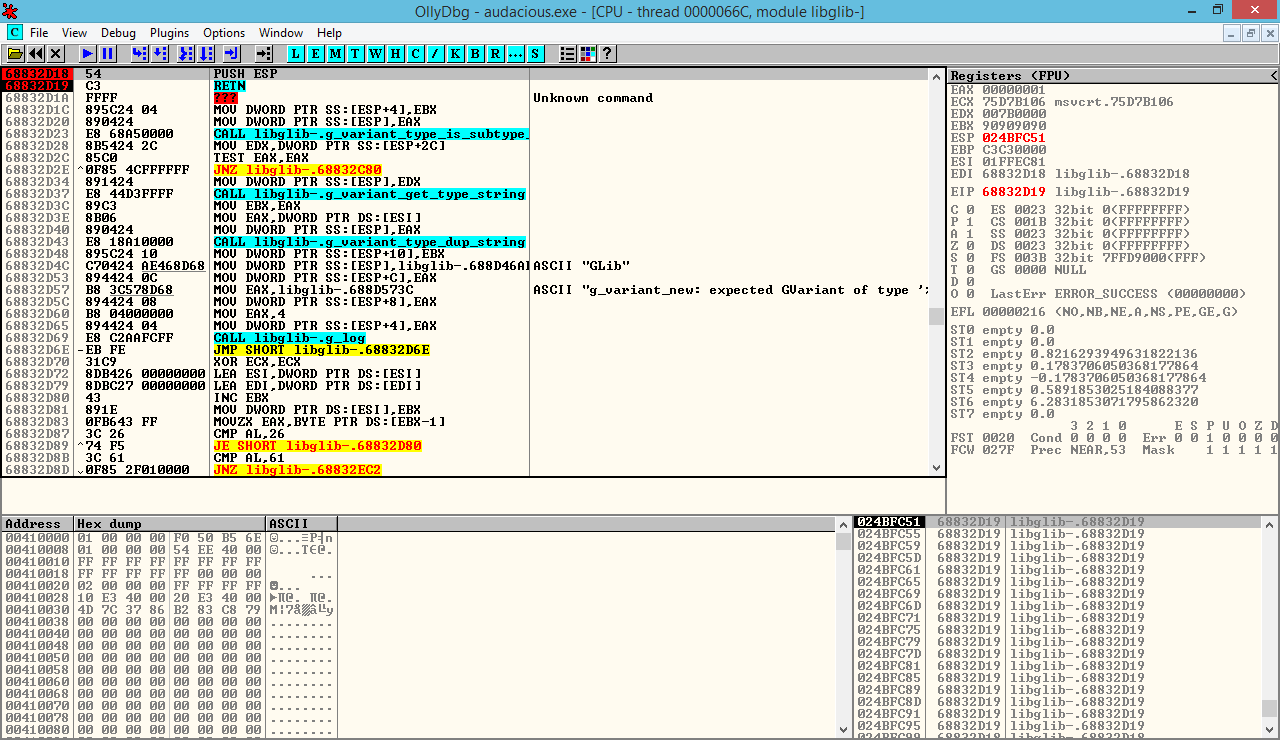


Push ESP to set EIP over the Stack.

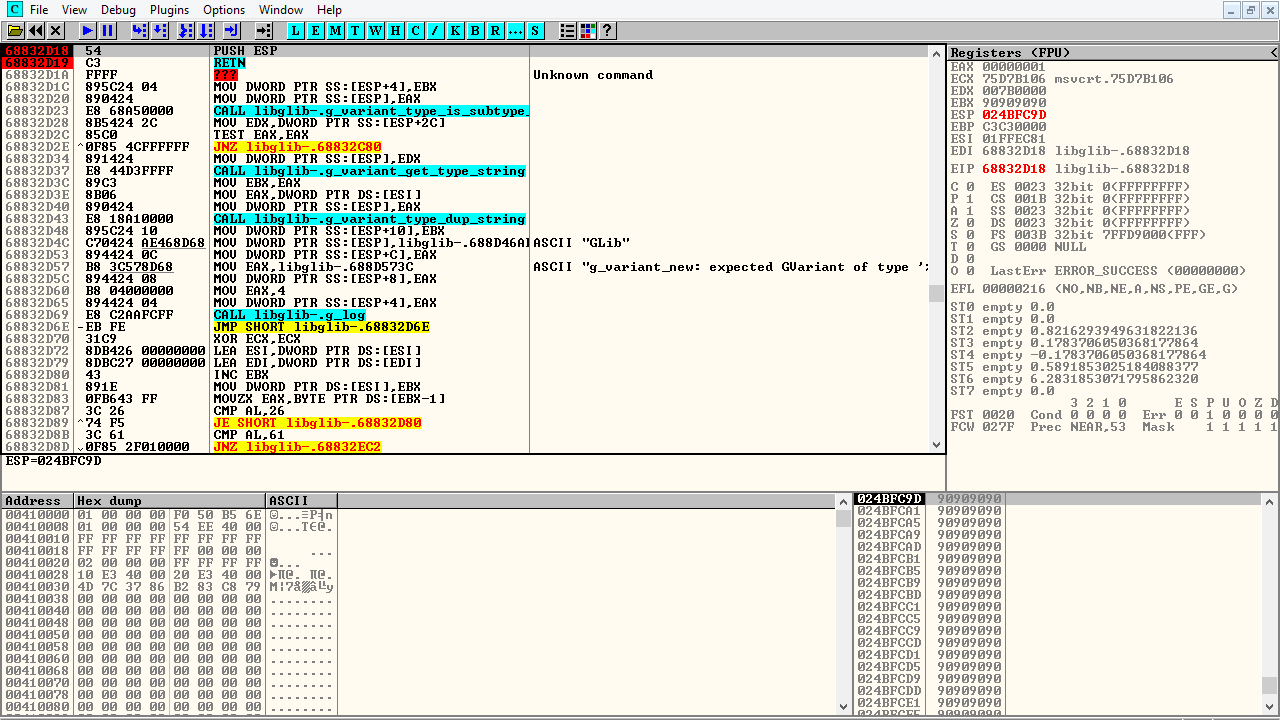


Execution Hardcoded Negative Stack Pivot from the Stack (no DEP).

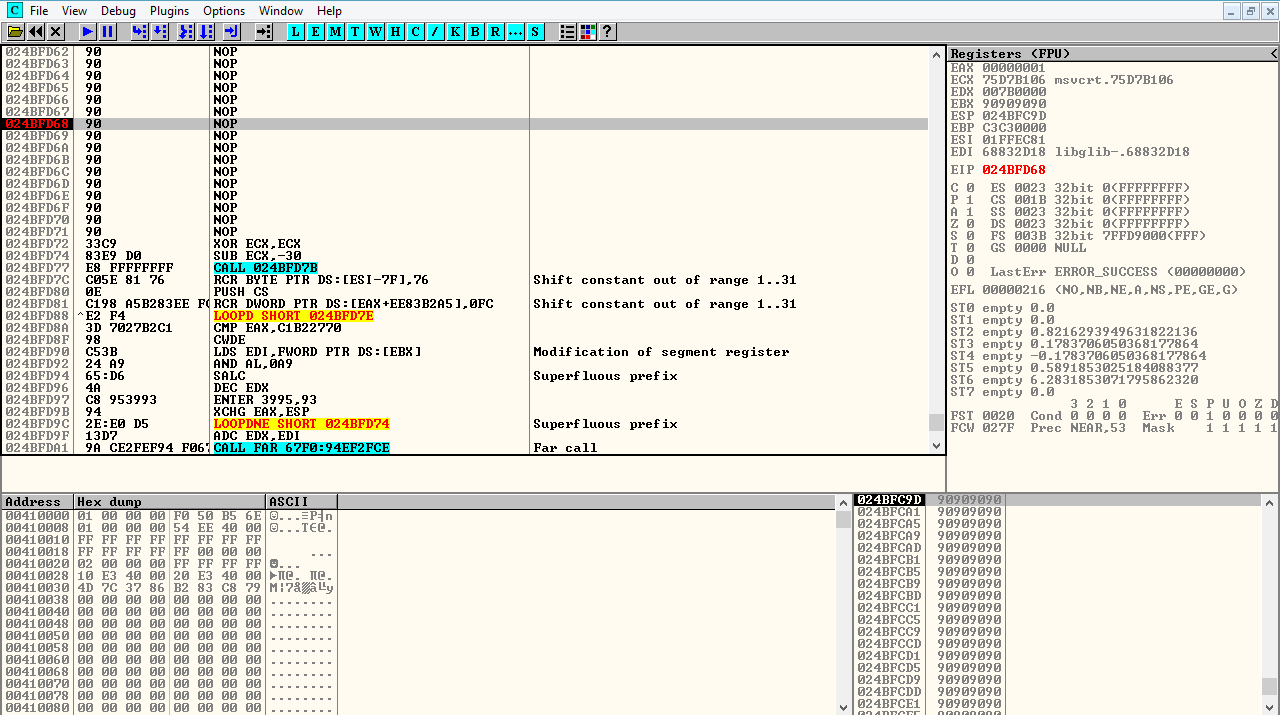
The **sub esp, 0x1ff** instruction jump over our stack spray of **RET** (to add more realibility between windows versions) and after this a new **push esp, ret** allow the payload to change the execution context to the nopsleed before the shellcode and achive the code execution.



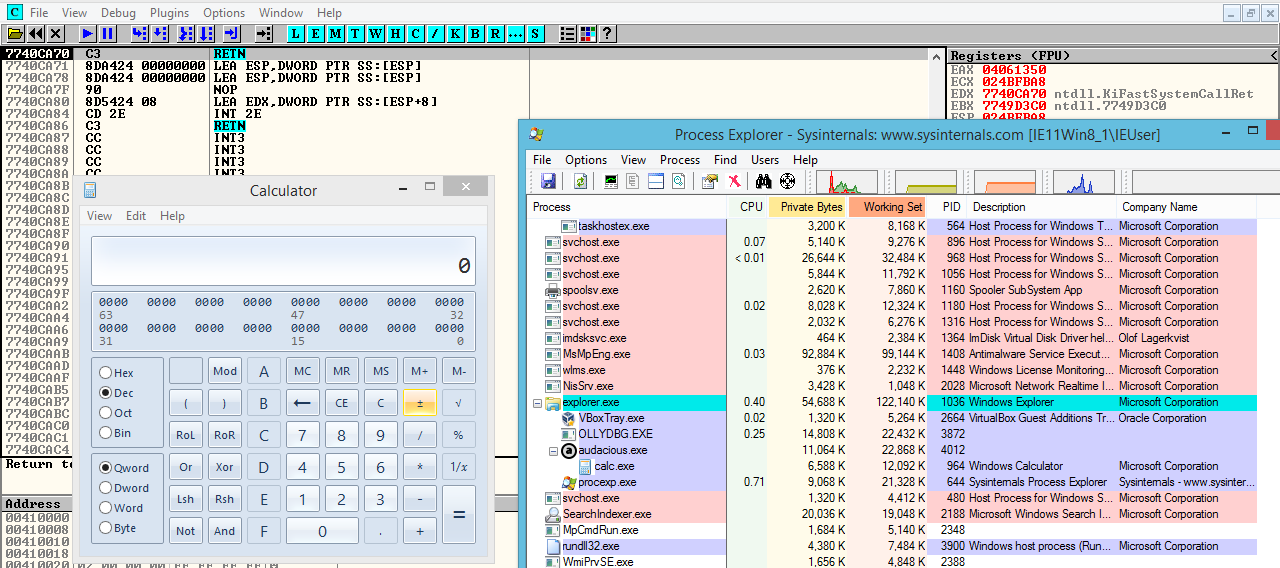
RETSleed call multiple times 0x68832D19 to add more reliability.



Gadget to change the execution context over the Stack (nopsleed + shellcode)



Nopsleed + Shellcode Execution.



Successful code execution.

|  |  |  |
| --- | --- | --- |
| **QQPlayer 3.9 – HeapOverflow** | | |
|  |  |  |

**. Finding:**

This vulnerability was found through fuzzing Matroska file format (.webm) files with WinAFL Framework after a deeper analysis of file parser libraries used by the program.

**. Impact:**

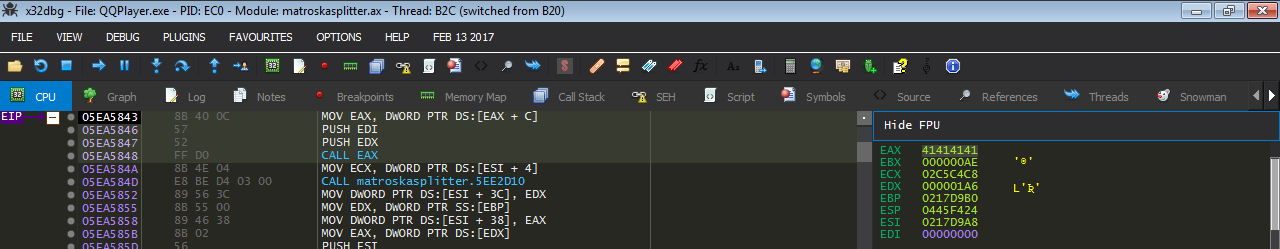
The vulnerability found in this application deal in code execution with the privileges of the user that run QQPlayer via local file.

**. Requeriments:**

The vulnerability can be triggered in all Windows (The PoC works in all platforms).

**. Details:**

The vulnerability is an unidentified Heap Overflow in the library matroskasplitter.ax the malformed file trigger several **call r32** across the program, the pointer used in the **call r32** is setted by a controlled register in a memory mov instructions in all the cases.



Call EAX trigger the bug, eax is set by move eax, dword [eax+C] where EAX value is controlled by the payload.

With an appropiate Heap Spray an attacker can controll all the **mov r32, [memory]; call r32** instances and trigger the bug without import where the crash happen in the application (the application crash in minimun 3 parts of the program in the same way).

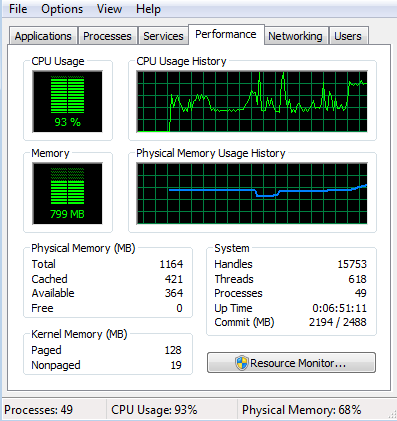
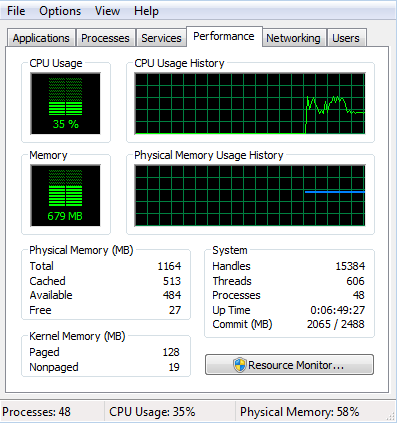
The only problem here was found a way to do the Heap Spray from the same file, this because QQPlayer does not have a Browser Plugin so is imposible Spray using JavaScript.

After several reversing process and deeper study of Matroska file format a way to Heap Spray appear due that the application not free the memory allocated by Matroska MultiTrack file structure.

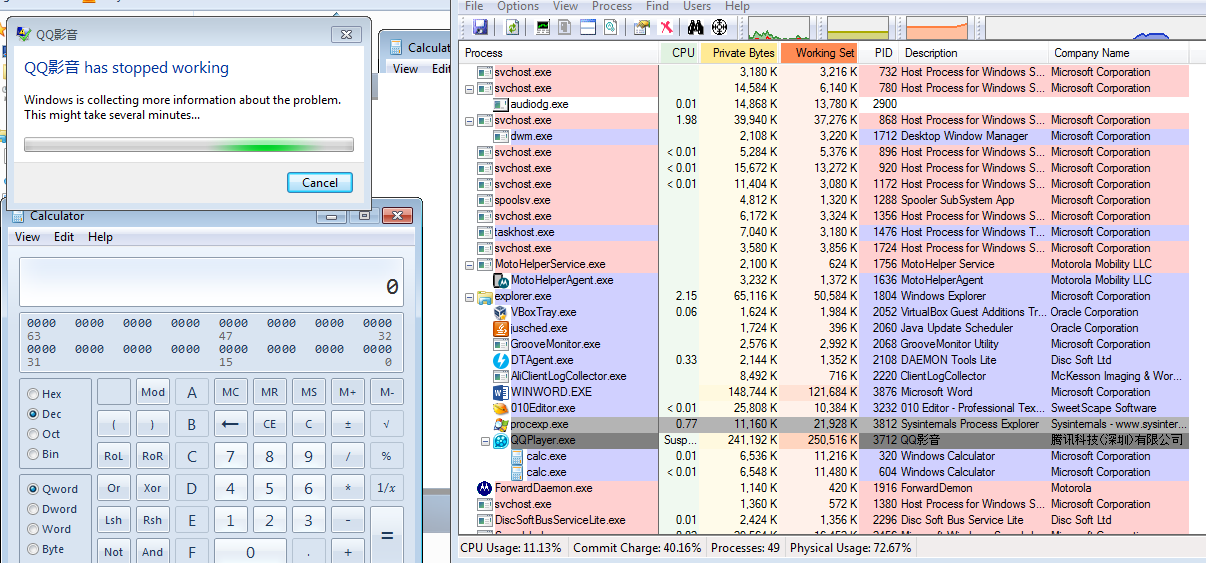
The vulnerability can be exploited crafting a malformed Matroska file that Spray the Heap using Multi-Track feature and in a last single track trigger the Heap Overflow, doing this, an attacker is able to control all the vulnerables **call r32** instances and trigger the code execution.

The process to derive in a full functional exploit after found the way to Heap Spray the application was in essence a try and error approach playing arround with the Matroska file specifications.

The result is the following:



Memory before and after the Heap Spray.



Successful code execution through Heap Spray.

|  |  |  |
| --- | --- | --- |
| **VLC 2.2.6 – StackOverflow** | | |
|  |  |  |

**. Finding:**

This vulnerability was found through fuzzing Video Object file format (.vob) files with WinAFL Framework after a deeper analysis of file parser libraries used by the program.

**. Impact:**

The vulnerability found in this application deal in code execution with the privileges of the user that run VLC via browsing an html file with Internet Explorer.

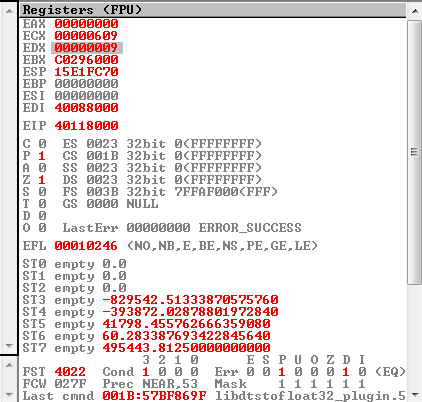
**. Requeriments:**

At the moment, the exploitation of this vulnerability is posible only in a laboratory enviroment due that the research team was unable to find a way to Heap Spray in modern Internet Explorer Browsers and bypass DEP in VLC, so this vulnerability need to be considereted like one gadget in an exploitation chain.

For the Laboratory environment was used a Windows 7 x86 without DEP + IE8.

**. Details:**

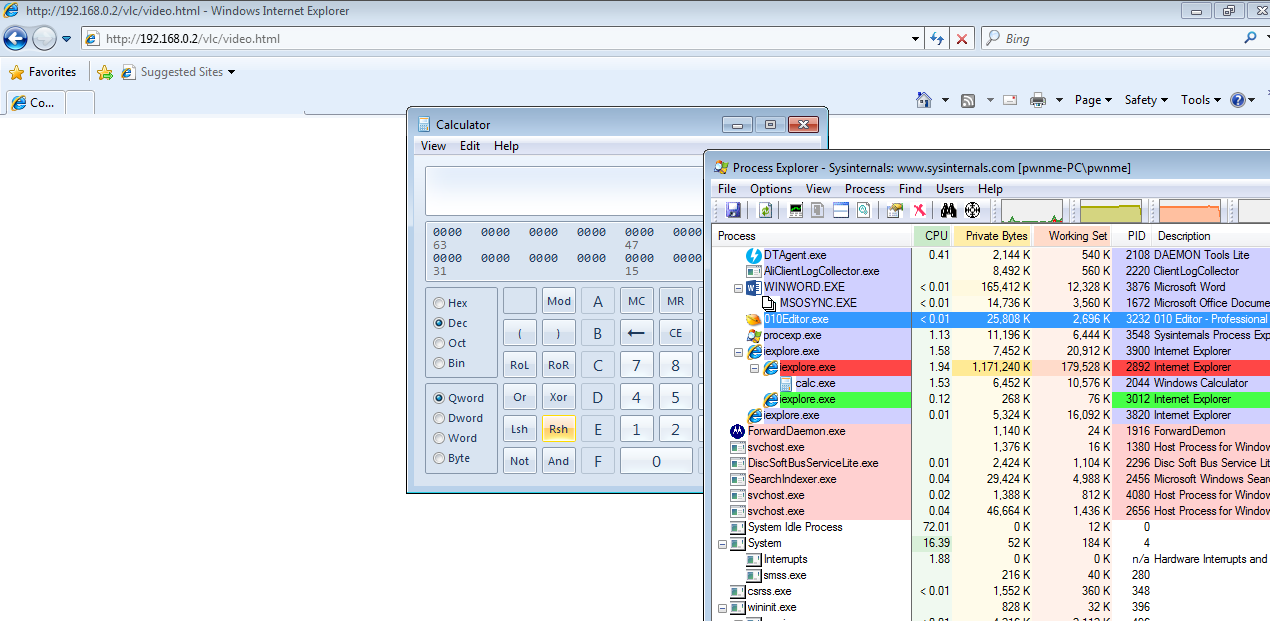
The vulnerability is an unidentified Memory Corruption triggered by a malformed file that derive in an uncontrollabe EIP overwriting.



Uncontrolled EIP Overwrite.

The vulnerabity derive from a bad floating point operation that cannot be controlled at the moment, but the EIP overwrite open the posibility of get code execution with a long Heap Spray that achieve a memory allocation in that zone.

This can be achived by JavaScript due the ActiveX plugin of VLC for Internet Explorer, the Heap Spray in this escenario is trivial only is needed a bit nopsleed and shellcode.  
  
The result of the PoC is the fallowing:



Successful code execution through Heap Spray.