PowerLoaderV2 Analysis Report

By: Mustafa Hussien

Aug 2020

TABLE OF CONTENTS

SUMMARY

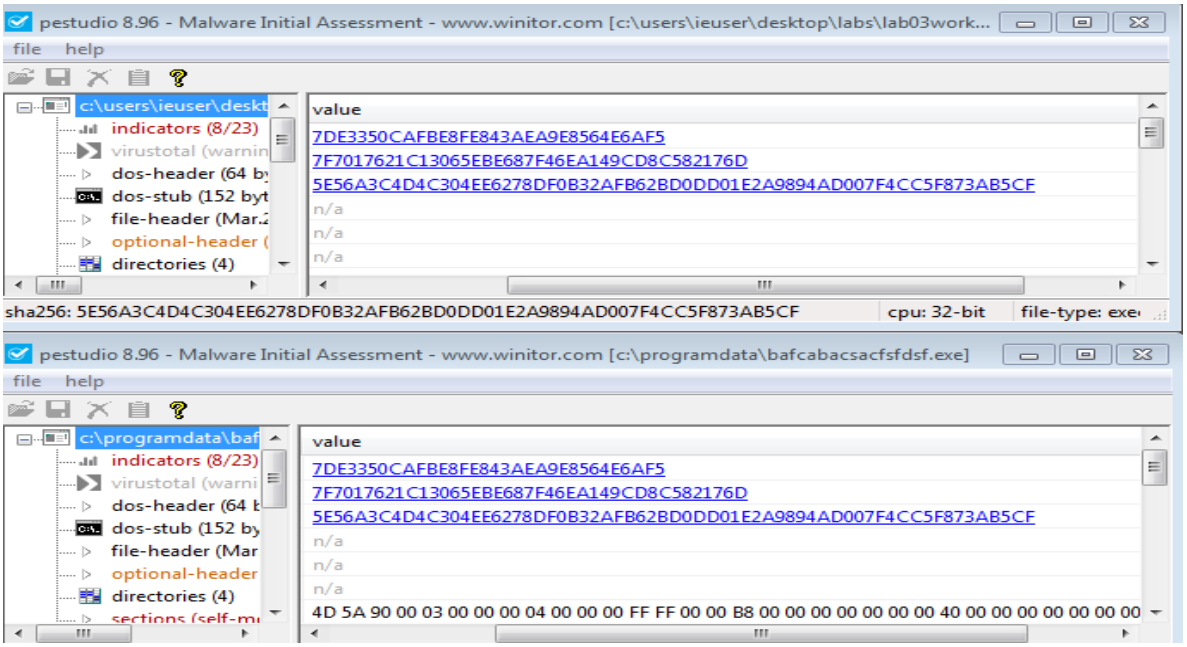
- PowerloaderV2.exe malware is a dropper with backdoor functionality copies itself into a different path with a different name inside this path C:\Program Data.
- It working on process injection and searching for explorer.exe to be injected by its code.
- It checks at first if this process is 32 bit or x64 bit and inject accordingly.
- It communicates with malicious domains trying to get updates from C&C servers or to download other malwares.

TECHNICAL ANALYSIS

In this section we're going to have all related technical details about PowerloaderV2.exe malware and its behavior.

Filename	MD5	PE Timestamp	Size (in Bytes)	Description
PowerLoaderV2.exe	7DE3350CAFBE8FE843AEA9E8564E6AF5	0x5141C969(Thu Mar 14 05:58:17 2013)	58880	Type of the malware: Backdoor, 32 bits
bafcabacsacfsfdsf.exe	7DE3350CAFBE8FE843AEA9E8564E6AF5	0x5141C969 (Thu Mar 14 05:58:17 2013)	58880	Type of the malware: Backdoor, 32 bits

As you see here that PowerLoaderV2.exe and bafcabacsacfsfdsf.exe have the same hash which indicate to the malware copies itself with a different name.



INTRO

- PowerLoaderV2.exe malware is a backdoor copies itself into a different path with a different name C:\Program Data\ bafcabacsacfsfdsf.exe.
- Also, it has dropper routines trying to connect with malicious domains to download other related samples.

FUNCTIONALITY

- One of its main functionalities is process injection, trying to inject itself into a trusted running process explorer.exe to evade antiviruses.
- It makes a http connection with malicious domains hopefully to get other samples to be dropped.
- The malware achieves persistence inside the system creating itself into registry HKCU\SOFTWARE\Wow6432Node\Microsoft\Windows\Current Version\Run

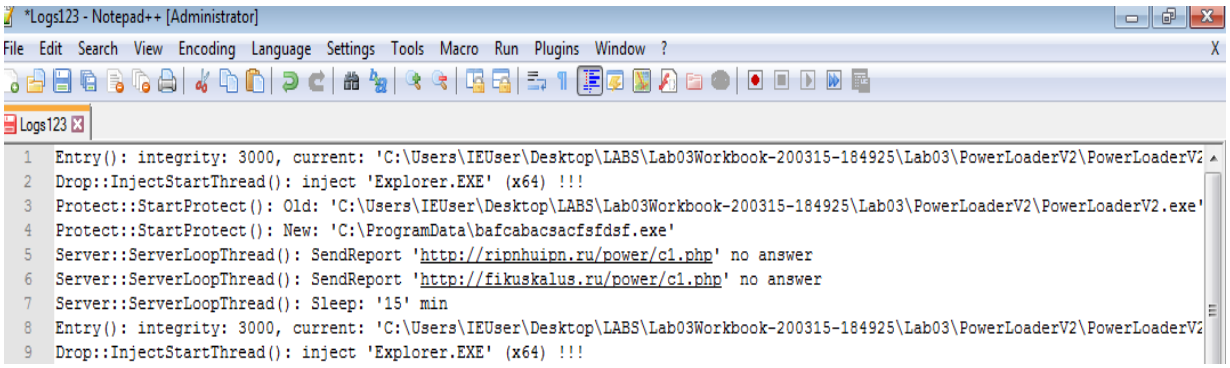
FILESYSTEM CHANGES

Filename	Change Type	Description
Logs123.txt	WRITE	Contains all logs about the traffic of malware connection and the API name responsible for injecting into Explorer.exe process.
.CFG	Created	Created by powerloader.exe contains all configurations.
bafcabacsacfsfdsf.exe	Write	It's a copy of the malware with a different name using it in the path of persistence.

PROCESSES AND MEMORY CHANGES

Process	Change Type	Description
PowerloaderV2.exe	Created/Service	Main malware
bafcabacsacfsfdsf.exe	Created/Service	Used in maintaining persistence
Explorer.exe	Injected	trusted process that will be injected by the malware

- As you see in the above tables PowerloaderV2 malware made changes to specific paths in windows with creation of many files.
- It created Logs123.txt file contains main function of the malware:
  - Communication with the malicious domains.
  - Trusted process which will be injected by its code.



- This section for all changes that happened in the registry by powerloaderV2.exe

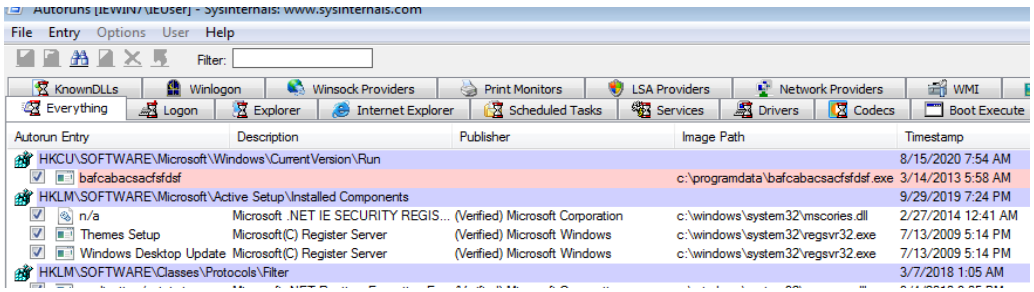
REGISTRY CHANGES

Registry Key	Value Name	Type	Value Data	Description
HKCU\Software\Microsoft\Windows\CurrentVersion\Run	bafcabacsacfsfdsf.exe	REG_SZ	"C:\ProgramData\bafcabacsacfsfdsf.exe"	Persistence/Autorun
HKLM\System\CurrentControlSet\Control\SafeBoot\Option	authenticodeenabled	REG_DWORD	0	Malware trying to query a value.
HKCU\SOFTWARE\bafcabacsacfsfdsf\CurrentPath111	CurrentPath111	REG_SZ	C:\ProgramData\bafcabacsacfsfdsf.exe	Adding and entry for itself under HKCU\Software
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options				Using it as a way of maintaining persistence

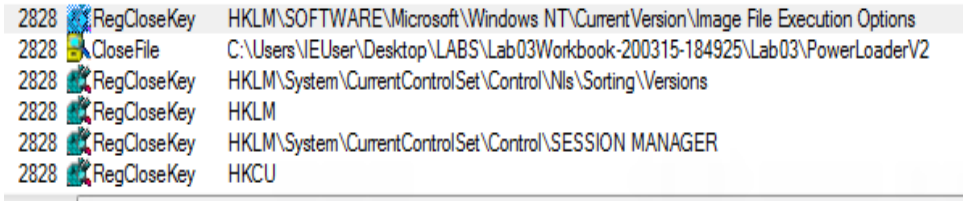
MAINTAINING PERSISTENCE

This section is talking about all required entries to maintain persistence including

- Autorun registry keys
  - HKCU\Software\Microsoft\Windows\CurrentVersion\Run



- Image File Execution Options
  - HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options



COMMAND & CONTROL (C&C)

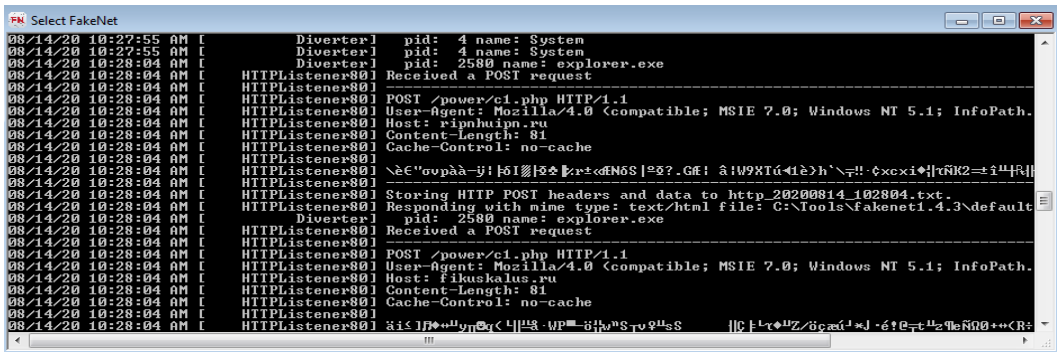
Here, we will present all information about PowerLoaderV2.exe communications with C&C servers

- Communication to multiple malicious domains trying to get C&C commands and other related samples to be downloaded.

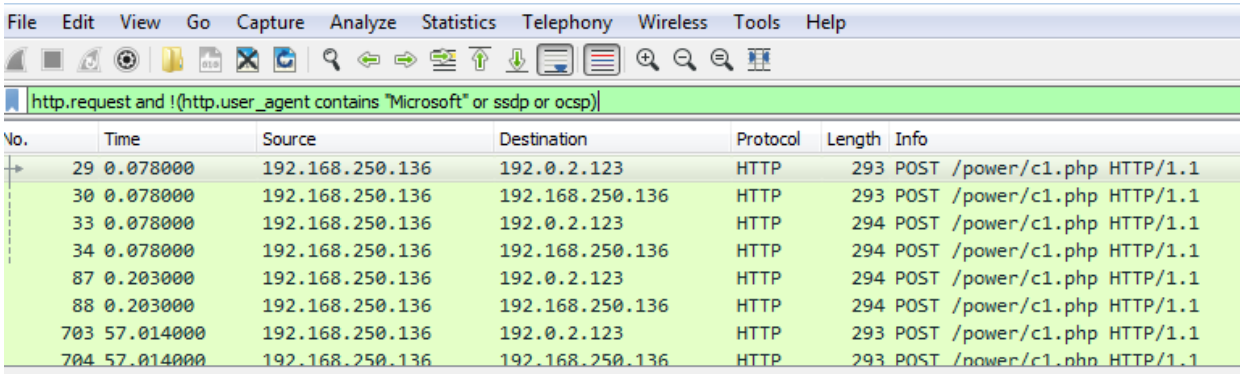
NETWORK IOCS

In this section, we will mention all related IPs, malicious domains that the malware uses to communicate with.

- Domains:
  - It communicates with ripnhuipn.ru and fikuskalus.run to get commands from C&C servers and updates also to download other malwares.



- IPs:
  - It tries to communicate with 192.0.2.123



File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

http.request and !(http.user_agent contains "Microsoft" or ssdp or ocsp)

No.	Time	Source	Destination	Protocol	Length	Info
29	0.078000	192.168.250.136	192.0.2.123	HTTP	293	POST /power/c1.php HTTP/1.1
30	0.078000	192.168.250.136	192.168.250.136	HTTP	293	POST /power/c1.php HTTP/1.1
33	0.078000	192.168.250.136	192.0.2.123	HTTP	294	POST /power/c1.php HTTP/1.1
34	0.078000	192.168.250.136	192.168.250.136	HTTP	294	POST /power/c1.php HTTP/1.1
87	0.203000	192.168.250.136	192.0.2.123	HTTP	294	POST /power/c1.php HTTP/1.1
88	0.203000	192.168.250.136	192.168.250.136	HTTP	294	POST /power/c1.php HTTP/1.1
703	57.014000	192.168.250.136	192.0.2.123	HTTP	293	POST /power/c1.php HTTP/1.1
704	57.014000	192.168.250.136	192.168.250.136	HTTP	293	POST /power/c1.php HTTP/1.1

STATIC CODE ANALYSIS

- In this section we will present all related static analysis about the main functionality of PowerLoaderV2 malware which Process injection of into Explorer.exe.
- We will analyze the function sub_4054F0, It's like the base of all process injection.
  - As you see here's a list of called functions by Sub_4054F0

```
.text:004054F0
.text:004054F0 sub_4054F0      proc near               ; CODE XREF: start+FE↓p
.text:004054F0
.text:004054F0 var_8           = dword ptr -8
.text:004054F0 var_4           = dword ptr -4
.text:004054F0
.text:004054F0                 push    ebp
.text:004054F1                 mov     ebp, esp
.text:004054F3                 sub     esp, 8
.text:004054F6                 push    ebx
.text:004054F7                 push    esi
.text:004054F8                 xor     bl, bl
.text:004054FA                 call    sub_405480
.text:004054FF                 push    offset aExplorerExe ; "explorer.exe"
.text:00405504                 call    sub_4023F0
.text:00405509                 mov     esi, eax
.text:0040550B                 test    esi, esi
.text:0040550D                 jz      short loc_40555D
.text:0040550F                 call    ds:GetCurrentProcess
.text:00405515                 push    eax
.text:00405516                 call    sub_402D20
.text:0040551B                 test    al, al
.text:0040551D                 jz      short loc_405544
.text:0040551F                 lea     eax, [ebp+var_4]
.text:00405522                 push    eax
.text:00405523                 lea     ecx, [ebp+var_8]
.text:00405526                 push    ecx
.text:00405527                 call    sub_402130
.text:0040552C                 test    al, al
```

- The 1st function it calls is a function at 0x00405480 which will call different APIs like LookupPrivilegeValueA and AdjustTokenPrivileges and openprocesstoken. These APIs are used to set some privileges to the process including SeDebugPrivilege which allows the process to write data into another process and execute them. Any malicious code injector will need to get these debugging privileges to inject its code.also It will open a handle to the access token associated with a process

```
push    eax             ; lpLuid
push    offset aSedebugprivile ; "SeDebugPrivilege"
push    0               ; lpSystemName
call    ds:LookupPrivilegeValueA
mov     ecx, [ebp+Luid.LowPart]
mov     edx, [ebp+Luid.HighPart]
lea     eax, [ebp+TokenHandle]
push    eax             ; TokenHandle
push    0F01FFh         ; DesiredAccess
mov     [ebp+NewState.Privileges.Luid.LowPart], ecx
mov     [ebp+NewState.Privileges.Luid.HighPart], edx
mov     [ebp+NewState.Privileges.Attributes], 2
mov     [ebp+NewState.PrivilegeCount], 1
call    ds:GetCurrentProcess
push    eax             ; ProcessHandle
call    ds:OpenProcessToken
mov     edx, [ebp+TokenHandle]
push    0               ; ReturnLength
push    0               ; PreviousState
push    10h             ; BufferLength
lea     ecx, [ebp+NewState]
push    ecx             ; NewState
push    0               ; DisableAllPrivileges
push    edx             ; TokenHandle
call    ds:AdjustTokenPrivileges
mov     esp, ebp
pop     ebp
retn
```

- Then it pushes the explorer.exe string to be queried and searched by the Sub_4023F0 function which will loop on all loaded processes in a moment/snapshot captured by CreateToolhelp32Snapshot. It uses Process32First and Process32Next to loop through all the processes searching for the process “Explorer.exe” and it returns its PID.

```
push    edi             ; th32ProcessID
push    2               ; dwFlags
call    ds:CreateToolhelp32Snapshot
mov     esi, eax
cmp     esi, 0FFFFFFFFh
jz      short loc_402473
lea     eax, [ebp+pe]
push    eax             ; lppe
push    esi             ; hSnapshot
mov     [ebp+pe.dwSize], 22Ch
call    ds:Process32FirstW
test    eax, eax
jz      short loc_40246C
push    ebx
mov     ebx, ds:lstrcmpiW

                        ; CODE XREF: sub_4023F0+61↓j
mov     edx, [ebp+lpString1]
lea     ecx, [ebp+pe.szExeFile]
push    ecx             ; lpString2
push    edx             ; lpString1
call    ebx ; lstrcmpiW
test    eax, eax
jz      short loc_402465
lea     eax, [ebp+pe]
push    eax             ; lppe
push    esi             ; hSnapshot
call    ds:Process32NextW
```

- Then it calls the 3rd function which is Sub_402D20, this function searches for IsWow64Process. This API is only found in Windows 64-bit only (that's why it has to load this API dynamically to stay compatible with 32-bit operating systems). It calls to this API to check if it's running as a 32-bit process inside a 64-bit operating system If a process is a Wow64 Process that means it's a 32-bit process running inside Wow64 emulator.

```
push    ebp
mov     ebp, esp
push    ecx
push    offset ProcName ; "IsWow64Process"
push    offset ModuleName ; "kernel32"
mov     [ebp+var_4], 0
call    ds:GetModuleHandleA
push    eax             ; hModule
call    ds:GetProcAddress
test    eax, eax
jz      short loc_402D50
mov     edx, [ebp+arg_0]
lea     ecx, [ebp+var_4]
push    ecx
push    edx
call    eax
```

- you will notice that it calls the function at 004053F0 (Inject_Function) in the picture below given lpBuffer and nSize variables.
- Inside this function, you will notice that it opens the process Explorer.exe that what it got and it allocates a memory space inside It (with nSize).

```
                        ; CODE XREF: sub_4054F0+2D↑j
mov     eax, lpBuffer
mov     ecx, nSize      ; nSize
push    eax             ; lpBuffer
mov     eax, esi
call    Inject_Function
pop     esi
pop     ebx
mov     esp, ebp
pop     ebp
retn
---------------------------------------------------------
```

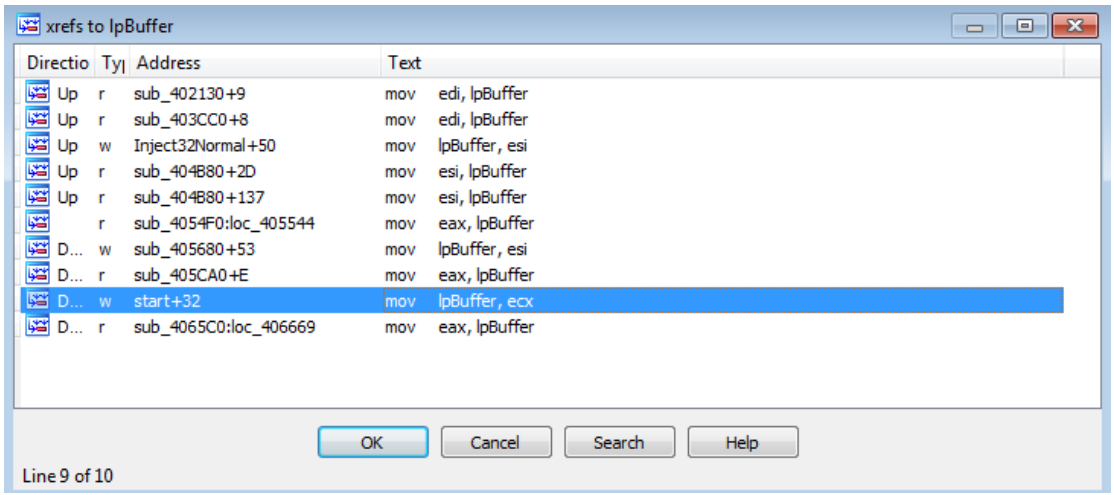
- Inside memory allocation, it uses **0x40** parameter as Protection (PAGE_EXECUTE_READWRITE) which Enables execute, read-only, or read/write access to the committed region of pages, and **0x3000** as Allocation type (MEM_COMMIT & MEM_RESERVE).
  So it's getting ready for code injection (EXECUTE protection) and then it writes the data pointed by lpBuffer global variable using WriteProcessMemory.

```
call    ds:OpenProcess
mov     edi, eax
test    edi, edi
jz      short loc_40546F
push    40h             ; flProtect
push    3000h           ; flAllocationType
push    esi             ; dwSize
push    0               ; lpAddress
push    edi             ; hProcess
call    ds:VirtualAllocEx
mov     ebx, eax
test    ebx, ebx
jz      short loc_405465
lea     ecx, [ebp+NumberOfBytesWritten]
push    ecx             ; lpNumberOfBytesWritten
push    esi             ; nSize
mov     esi, [ebp+lpBuffer]
push    esi             ; lpBuffer
push    ebx             ; lpBaseAddress
push    edi             ; hProcess
call    ds:WriteProcessMemory
```

- After the malware injected its code it will get the address of InjectNormRoutine exported function and it will execute this Function inside Explorer.exe Process.

```
push    1
push    offset aInjectnormrout_0 ; "InjectNormRoutine"
call    sub_4017A0
push    0               ; lpThreadId
push    0               ; dwCreationFlags
push    ebx             ; lpParameter
add     eax, ebx
push    eax             ; lpStartAddress
push    0               ; dwStackSize
push    0               ; lpThreadAttributes
push    edi             ; hProcess
call    ds:CreateRemoteThread
mov     bl, 1
test    eax, eax
jnz     short loc_405468
```

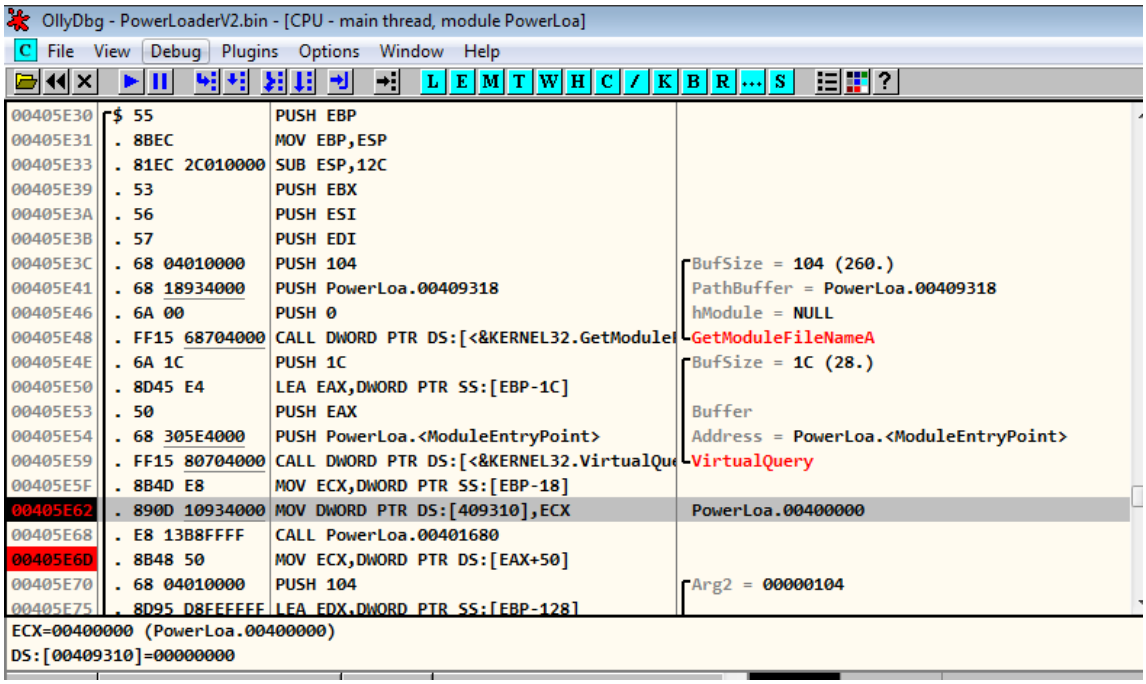
- Now to ensure that this malware has changed the Ip Buffer value, we can go back to the function Sub_4054F0 and check the Xref to see if IP Buffer as destination and got a value changed.
- As you notice in the entrypoint the IP Buffer is a destination with changed value of ecx.



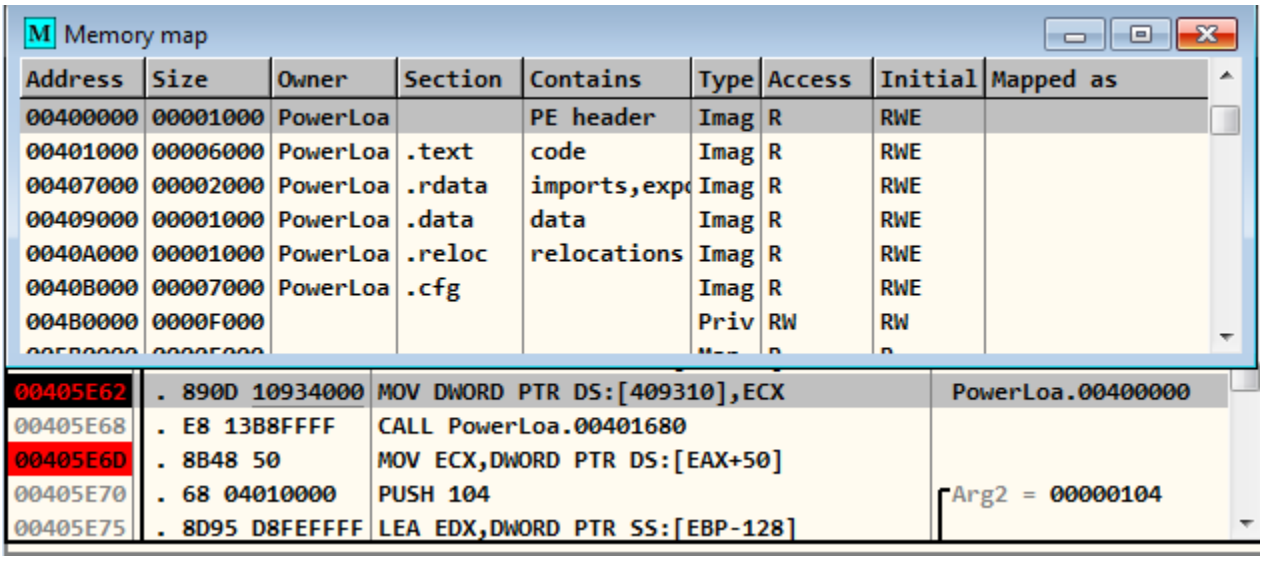
- So, the malware injects the malware DLL into the explorer.exe process. It checks if it's running in 32-bit or 64-bit environment and injects the code regarding that. It executes the exported function "InjectNromRoutine" in the injected process.

DYNAMIC ANLAYSIS

- As you see, it sets the lpBuffer with the malware Imagebase (0x400000) and nSize with the full malware size in memory.



- And as you can notice here in the memory has got the nSize which is (0x400000).



Memory map

Address	Size	Owner	Section	Contains	Type	Access	Initial	Mapped as
00400000	00001000	PowerLoa		PE header	Imag	R	RWE	
00401000	00006000	PowerLoa	.text	code	Imag	R	RWE	
00407000	00002000	PowerLoa	.rdata	imports,expo	Imag	R	RWE	
00409000	00001000	PowerLoa	.data	data	Imag	R	RWE	
0040A000	00001000	PowerLoa	.reloc	relocations	Imag	R	RWE	
0040B000	00007000	PowerLoa	.cfg		Imag	R	RWE	
004B0000	0000F000				Priv	RW	RW	

```
00405E62  . 890D 10934000  MOV DWORD PTR DS:[409310],ECX        PowerLoa.00400000
00405E68  . E8 13B8FFFF    CALL PowerLoa.00401680
00405E6D  . 8B48 50        MOV ECX,DWORD PTR DS:[EAX+50]
00405E70  . 68 04010000    PUSH 104                             Arg2 = 00000104
00405E75  . 8D95 D8FEFFFF  LEA EDX,DWORD PTR SS:[EBP-128]
```

REFERENCES

- ESET Report Analysis:
  - https://www.eset.com/fileadmin/eset/US/resources/docs/white-papers/Hesperbot-Trojan-Warning.pdf
- Trendmicro report analysis:
  - https://www.trendmicro.ae/vinfo/th/threat-encyclopedia/malware/bkdr_liftoh.dlf