Security Research and Development Framework

By Amr Thabet
Malware Researcher
@Q-CERT

About The Author

- Malware Researcher at Q-CERT
- Wrote a Stuxnet Malware Analysis Paper
- Author of Pokas x86 Emulator
- Author of SRDF (what we will talk about)

Introduction

- Development Framework (Library)
- Contains many security classes/tools
- Created For:
  - Malware Analysis
  - Packet Analysis
  - Antivirus and Firewall Tools
- Free and Open Source

Why SRDF?

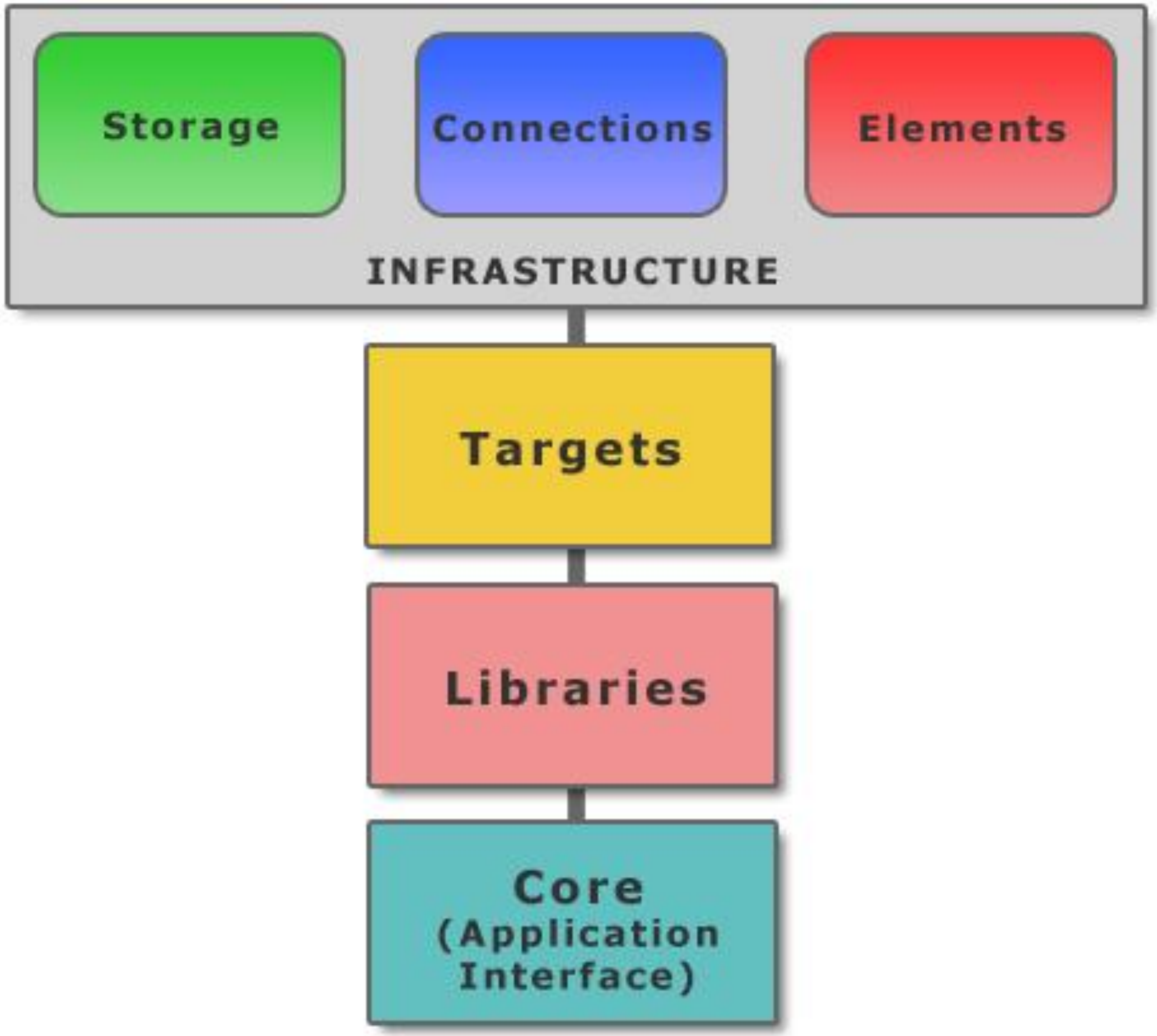
For This Guy !!

Why SRDF?

- Implement your Inovative Idea
- Don’t re-invent The Wheel
- Don’t waste your time
- Flexible Design
- Production Quality
- Community Based Development and Beta-testing

Contents

User-Mode Design



User-Mode Design

- Infrastructure:

  The Common Part at any Framework … not related to security
- Targets:

  What you will secure or secure from
- Libraries:

  The Security Tools are here ☺ … it’s divided into Malware and Network
- Core:

  The interface and the managment

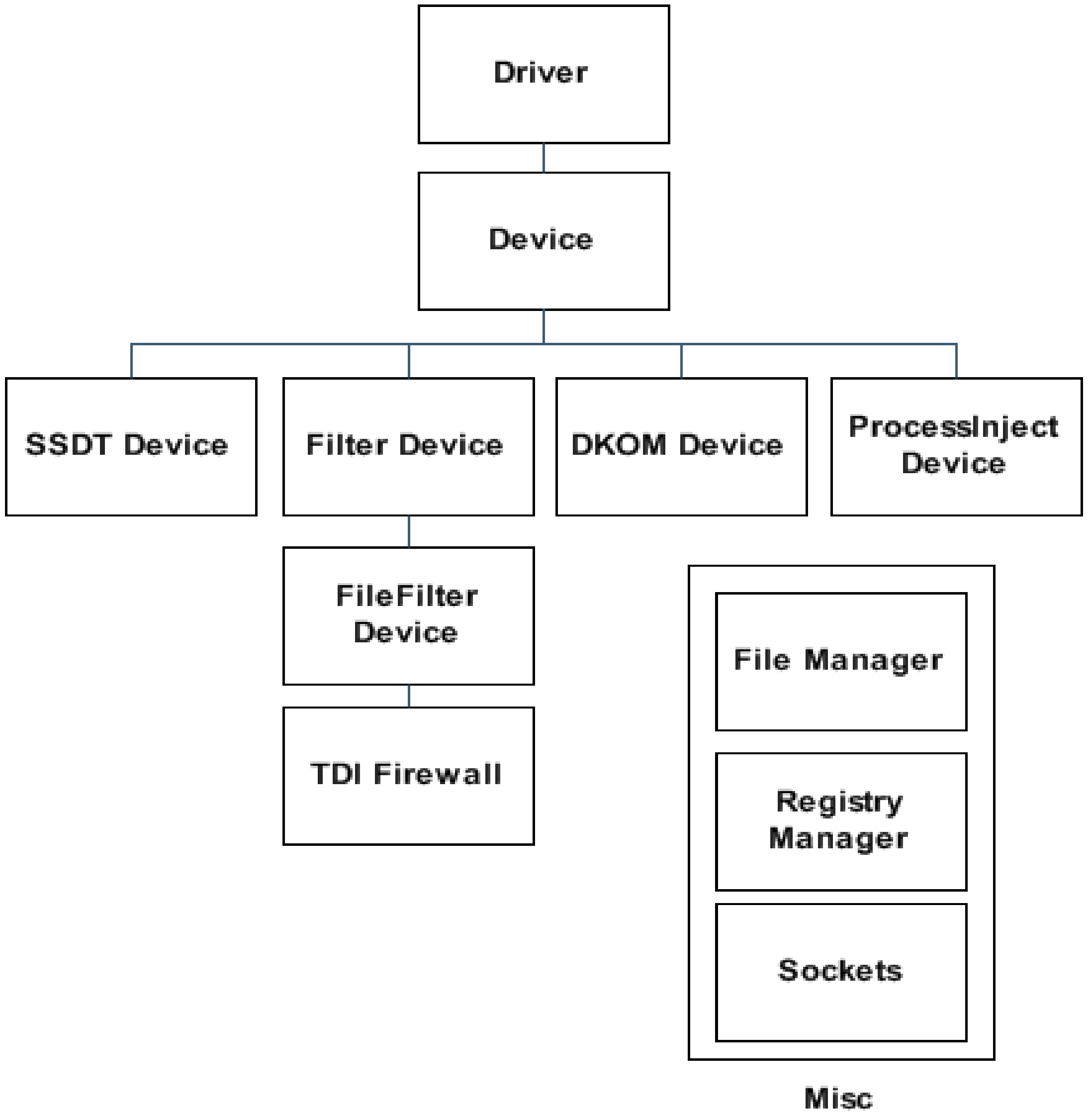
Features

- Full OOP
- PE, ELF, PDF and Andorid File Parsers
- x86 Disassembler, Debugger and Emulator
- API Hooking
- Packet, Protocol and Network Flow Analysis
- Production Quality
- FREE and Open Source

Kernel-Mode

- Support x32 Bits until now
- Little bit old
- Should be extended to x64
- Under Construction

Kernel- Mode Design



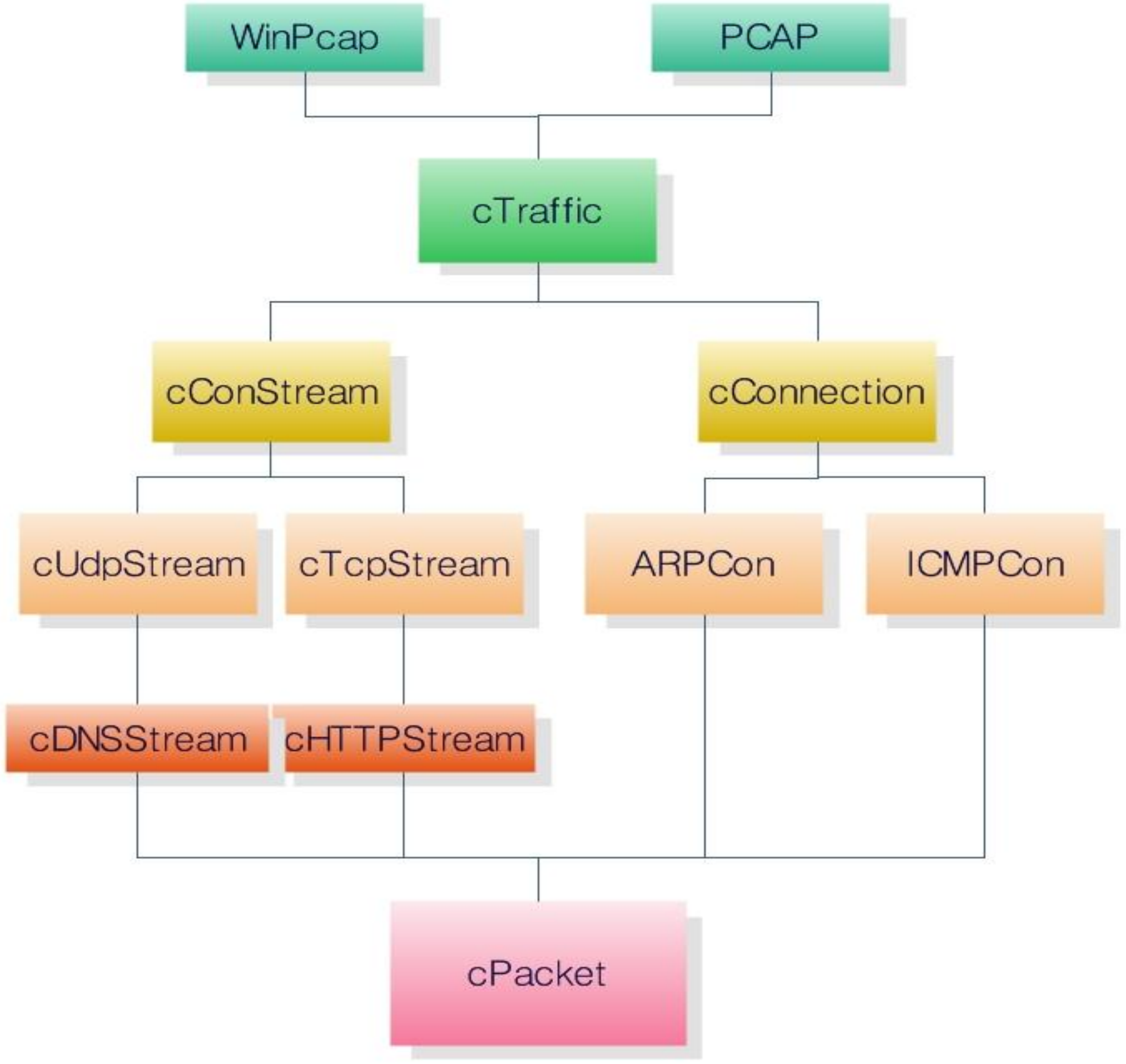
Major Projects

- Packetyzer
- x86 Emulator

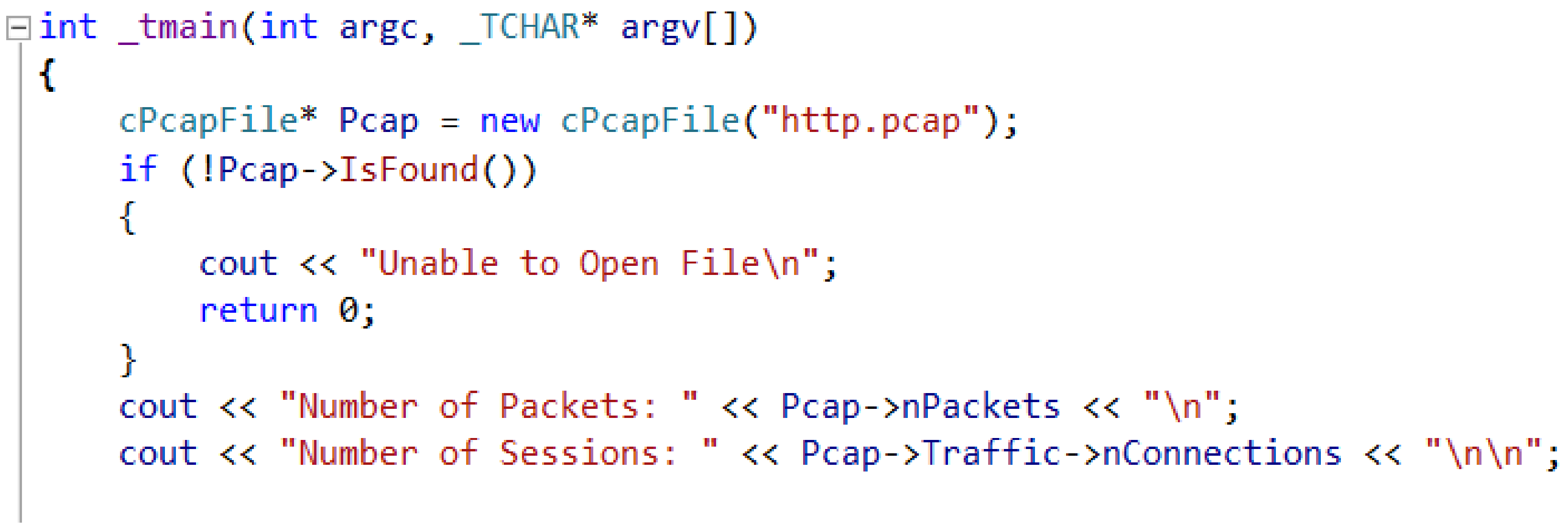
Packetyzer

- Created By Anwar Mohamed
- Packet Analysis Tool
- Session Separation
- Generating Packets and Send (Winpcap)
- Decodes:
  - ARP,ICMP,TCP,UDP
  - HTTP, DNS
- Parse PCAP Files
- Reassemble Packets

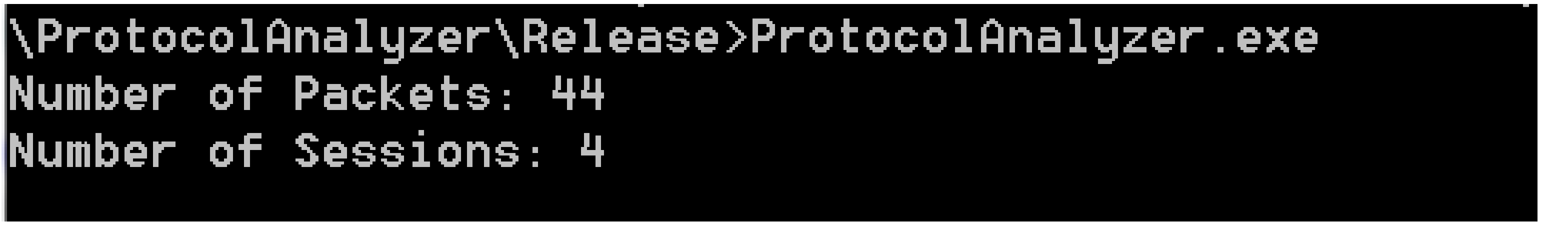
Design


WinPcap
PCAP
cTraffic
cConStream
cConnection
cUdpStream
cTcpStream
ARPCon
ICMPCon
cDNSStream
cHTTPStream
cPacket

Simple Demo – Read Pcap File



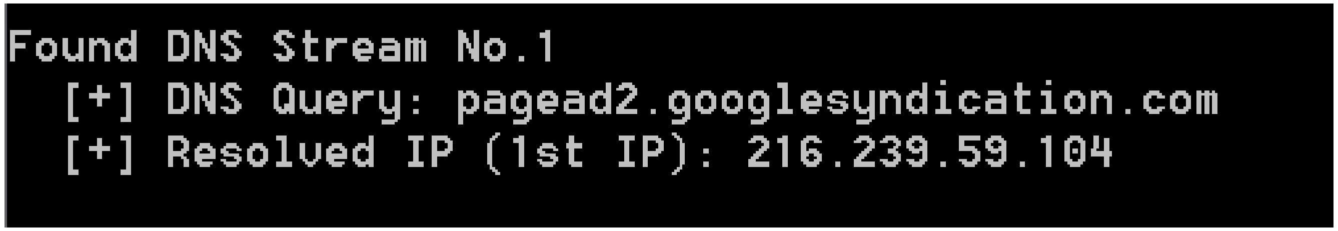
```
int _tmain(int argc, _TCHAR* argv[])
{
    cPcapFile* Pcap = new cPcapFile("http.pcap");
    if (!Pcap->IsFound())
    {
        cout << "Unable to Open File\n";
        return 0;
    }
    cout << "Number of Packets: " << Pcap->nPackets << "\n";
    cout << "Number of Sessions: " << Pcap->Traffic->nConnections << "\n\n";
```



```
\ProtocolAnalyzer\Release>ProtocolAnalyzer.exe
Number of Packets: 44
Number of Sessions: 4
```

Simple Demo – DNS Streams

```cpp
for (int i = 0; i < Pcap->Traffic->nConnections;i++)
{
    if (Pcap->Traffic->Connections[i]->ApplicationType == CONN_APPLICATION_DNS)
    {
        cDNSStream* DNS = (cDNSStream*)Pcap->Traffic->Connections[i];
        cout << "Found DNS Stream No." << i << "\n";
        cout << "   [+] DNS Query: " << DNS->RequestedDomain << "\n";
        cout << "   [+] Resolved IP (1st IP): " << PrintIP(DNS->ResolvedIPs[0]) << "\n";
        cout << "\n";
    }
```

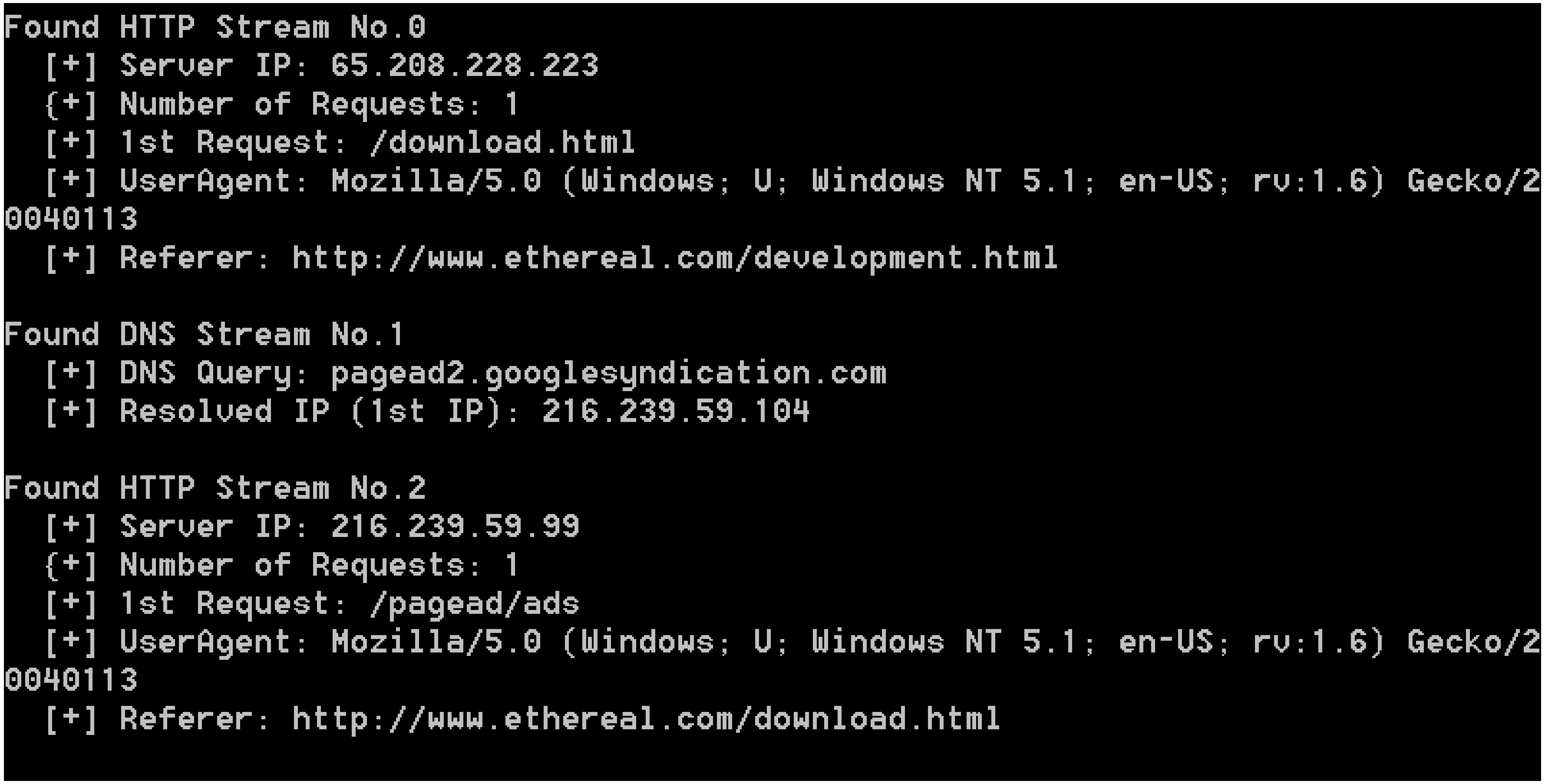


```
Found DNS Stream No.1
  [+] DNS Query: pagead2.googlesyndication.com
  [+] Resolved IP (1st IP): 216.239.59.104
```

Simple Demo – HTTP Streams

```
if (Pcap->Traffic->Connections[i]->ApplicationType == CONN_APPLICATION_HTTP)
{
    cHTTPStream* HTTP = (cHTTPStream*)Pcap->Traffic->Connections[i];
    cout << "Found HTTP Stream No." << i << "\n";
    cout << "  [+] Server IP: " << PrintIP(HTTP->ServerIP) << "\n";
    cout << "  {+] Number of Requests: " << HTTP->nRequests << "\n";
    if (HTTP->nRequests != 0)
        cout << "  [+] 1st Request: " << (char*)HTTP->Requests[0].Address->GetChar() << "\n";
    cout << "  [+] UserAgent: " << HTTP->UserAgent->GetChar() << "\n";
    cout << "  [+] Referer: " << (char*)HTTP->Referer->GetChar() << "\n";
    cout << "\n";

}
```

Simple Demo – HTTP Output



```
Found HTTP Stream No.0
  [+] Server IP: 65.208.228.223
  {+] Number of Requests: 1
  [+] 1st Request: /download.html
  [+] UserAgent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.6) Gecko/2
0040113
  [+] Referer: http://www.ethereal.com/development.html

Found DNS Stream No.1
  [+] DNS Query: pagead2.googlesyndication.com
  [+] Resolved IP (1st IP): 216.239.59.104

Found HTTP Stream No.2
  [+] Server IP: 216.239.59.99
  {+] Number of Requests: 1
  [+] 1st Request: /pagead/ads
  [+] UserAgent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.6) Gecko/2
0040113
  [+] Referer: http://www.ethereal.com/download.html
```

Packetyzer

- Reach it at:

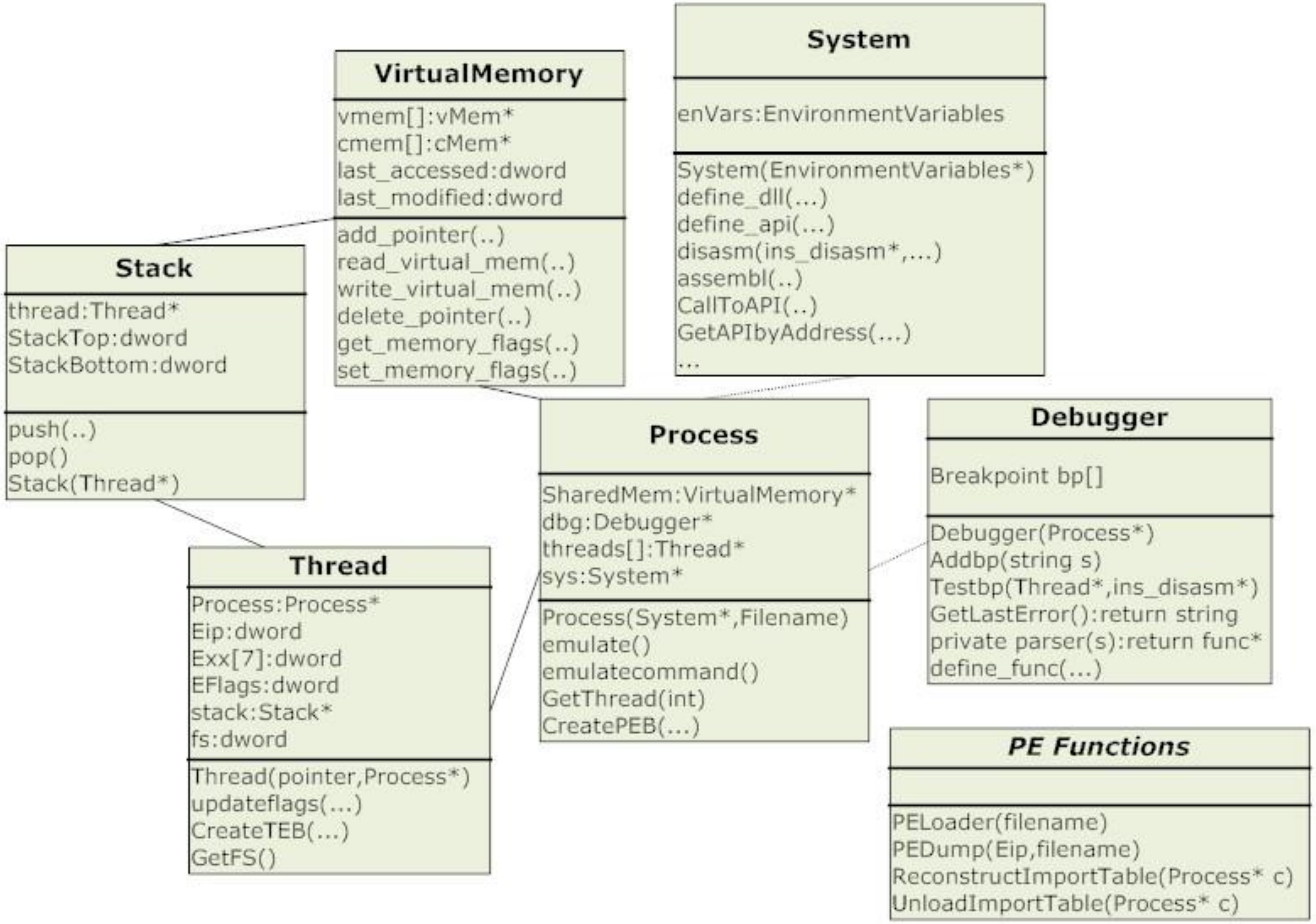
  https://github.com/AnwarMohamed/Packetyzer

- It's also a Part of SRDF

Pokas Emulator

- For win32 Applications
- very powerful debugger
- Monitor Memory Writes
- Emulate PE Files and Shellcode
- Dump The Process
- Reconstruct Import Table
- SRDF has a Wrapper Class for it

Design


VirtualMemory
vmem[]:vMem*
cmem[]:cMem*
last_accessed:dword
last_modified:dword
add_pointer(..)
read_virtual_mem(..)
write_virtual_mem(..)
delete_pointer(..)
get_memory_flags(..)
set_memory_flags(..)
System
enVars:EnvironmentVariables
System(EnvironmentVariables*)
define_dll(...)
define_api(...)
disasm(ins_disasm*,...)
assembl(..)
CallToAPI(..)
GetAPIbyAddress(...)
...
Stack
thread:Thread*
StackTop:dword
StackBottom:dword
push(..)
pop()
Stack(Thread*)
Process
SharedMem:VirtualMemory*
dbg:Debugger*
threads[]:Thread*
sys:System*
Process(System*,Filename)
emulate()
emulatecommand()
GetThread(int)
CreatePEB(...)
Debugger
Breakpoint bp[]
Debugger(Process*)
Addbp(string s)
Testbp(Thread*,ins_disasm*)
GetLastError():return string
private parser(s):return func*
define_func(...)
Thread
Process:Process*
Eip:dword
Exx[7]:dword
EFlags:dword
stack:Stack*
fs:dword
Thread(pointer,Process*)
updateflags(...)
CreateTEB(...)
GetFS()
PE Functions
PELoader(filename)
PEDump(Eip,filename)
ReconstructImportTable(Process* c)
UnloadImportTable(Process* c)

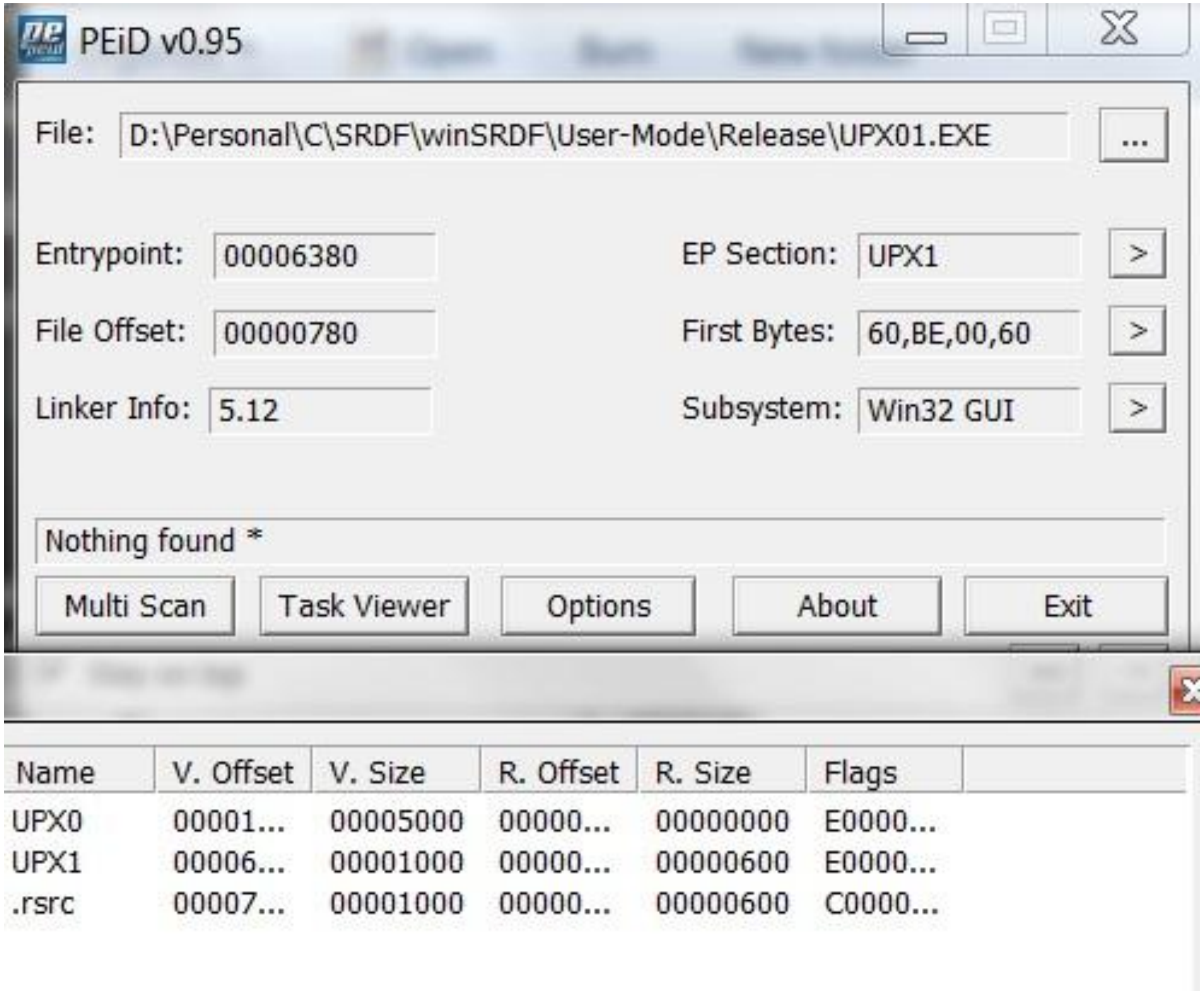
The Emulator’s Debugger

- Take String Condition
- Convert it into Native Code
- Very Fast
- Easy to Customize
- Have Predefined Functions
- Allow to Add Function

Examples

- "__isdirty(eip)"
- "__disp() >=0x00401000 &&ecx>10"
- "(eax& 0xff)> 5*(edx& 0xff) || __read(0x401000)==0x500"
- "__isapiequal('getprocaddress') || __isapiequal('loadlibraryA')"

Demo: Unpack UPX - PEid


PEiD v0.95

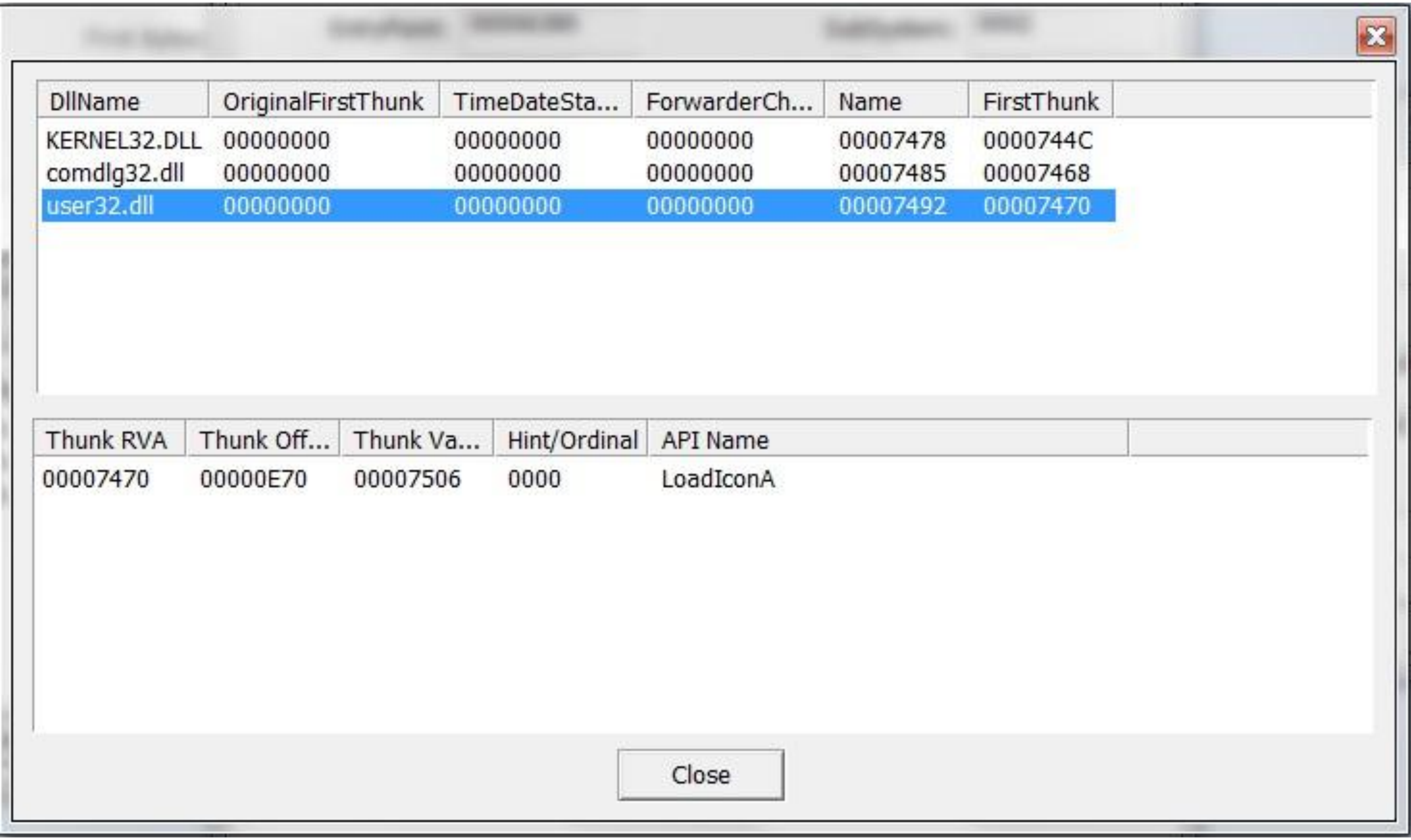
File: D:\Personal\C\SRDF\winSRDF\User-Mode\Release\UPX01.EXE

Entrypoint:	00006380	EP Section:	UPX1
File Offset:	00000780	First Bytes:	60,BE,00,60
Linker Info:	5.12	Subsystem:	Win32 GUI

Nothing found *

Name	V. Offset	V. Size	R. Offset	R. Size	Flags
UPX0	00001...	00005000	00000...	00000000	E0000...
UPX1	00006...	00001000	00000...	00000600	E0000...
.rsrc	00007...	00001000	00000...	00000600	C0000...

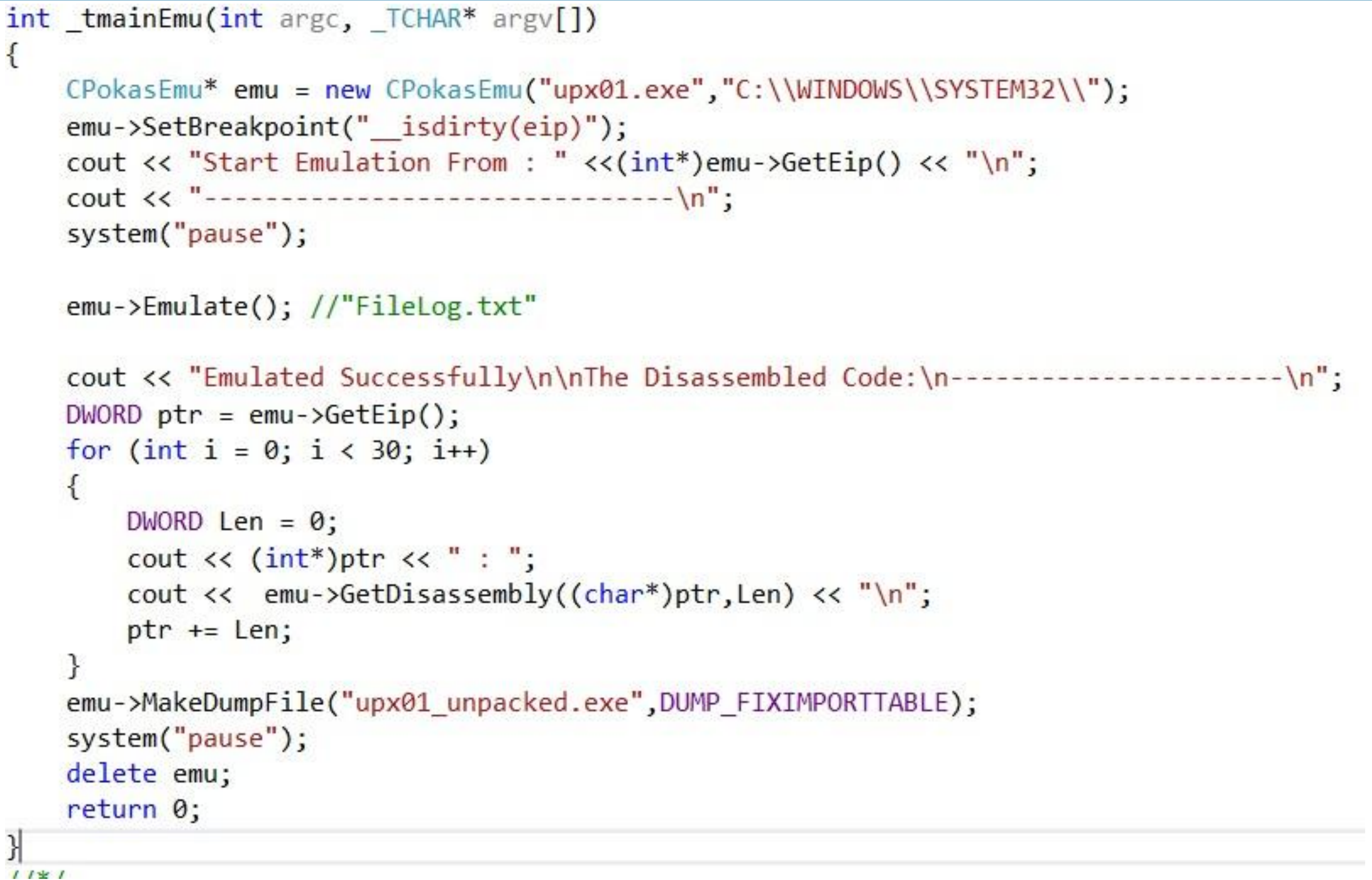
Demo: Unpack UPX – ImportTable



DllName	OriginalFirstThunk	TimeDateSta...	ForwarderCh...	Name	FirstThunk
KERNEL32.DLL	00000000	00000000	00000000	00007478	0000744C
comdlg32.dll	00000000	00000000	00000000	00007485	00007468
user32.dll	00000000	00000000	00000000	00007492	00007470

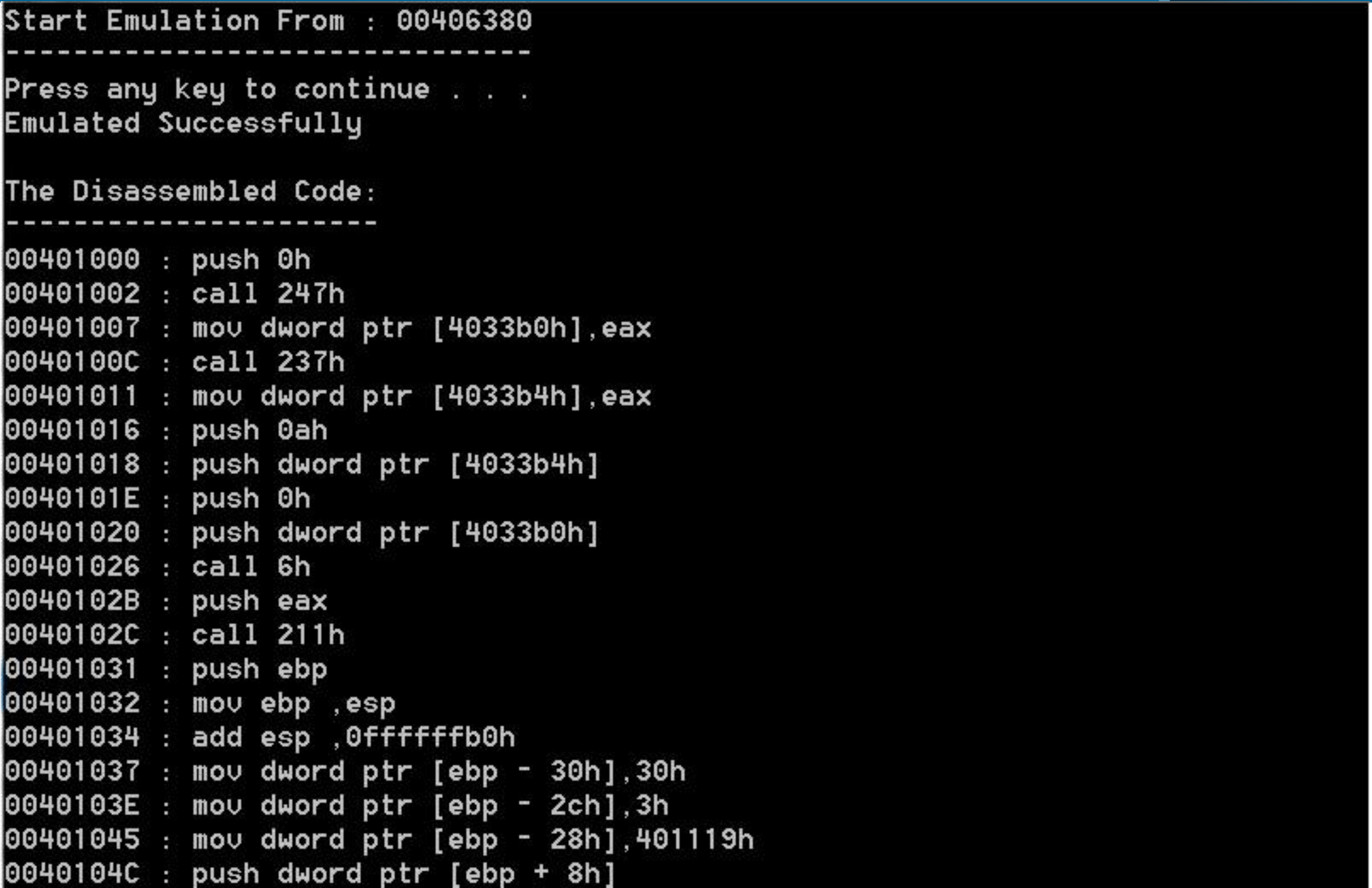
Thunk RVA	Thunk Off...	Thunk Va...	Hint/Ordinal	API Name
00007470	00000E70	00007506	0000	LoadIconA

Demo: Unpack UPX - Code



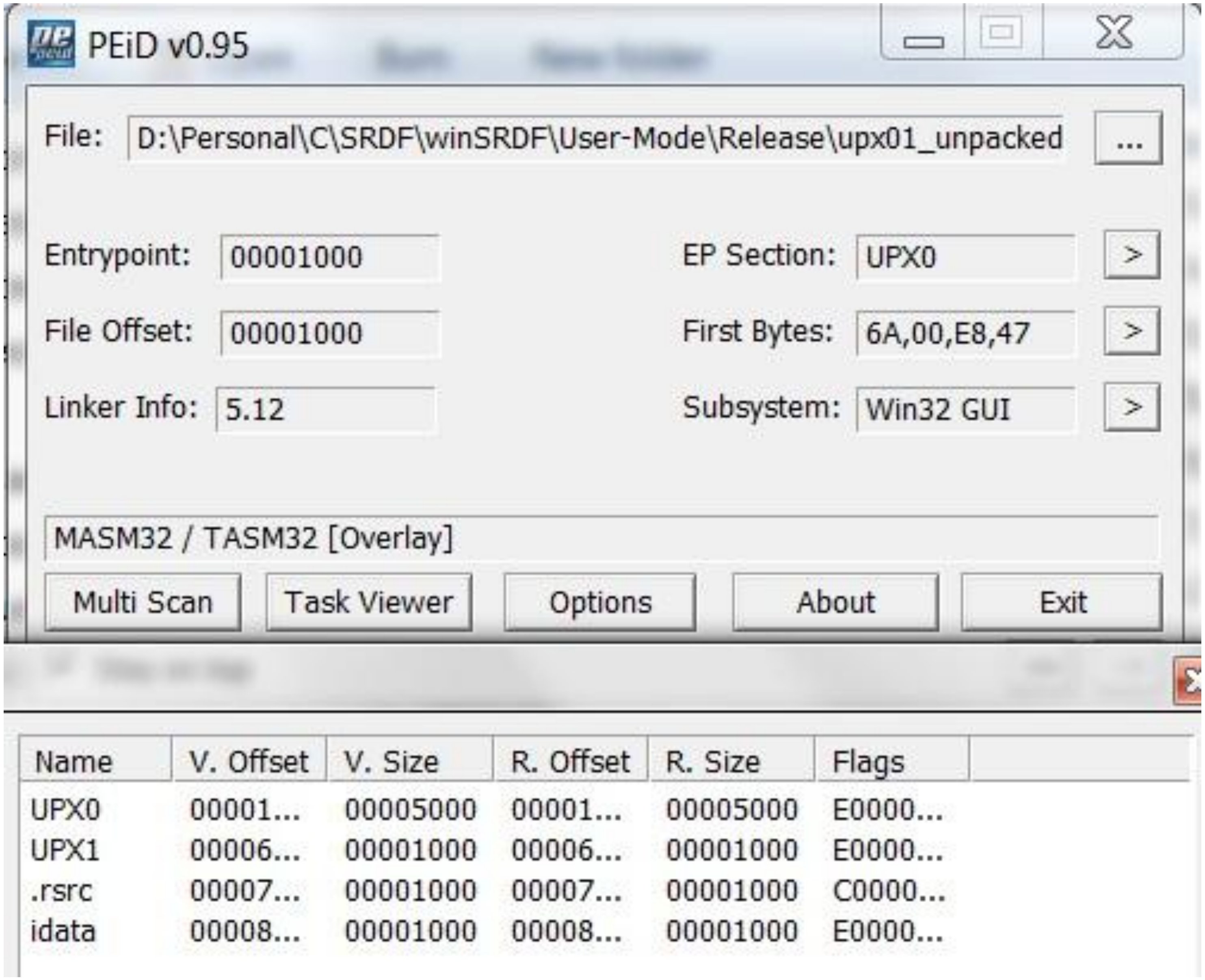
```
int _tmainEmu(int argc, _TCHAR* argv[])
{
    CPokasEmu* emu = new CPokasEmu("upx01.exe","C:\\WINDOWS\\SYSTEM32\\");
    emu->SetBreakpoint("__isdirty(eip)");
    cout << "Start Emulation From : " <<(int*)emu->GetEip() << "\n";
    cout << "-------------------------------\n";
    system("pause");

    emu->Emulate(); //"FileLog.txt"

    cout << "Emulated Successfully\n\nThe Disassembled Code:\n----------------------\n";
    DWORD ptr = emu->GetEip();
    for (int i = 0; i < 30; i++)
    {
        DWORD Len = 0;
        cout << (int*)ptr << " : ";
        cout <<  emu->GetDisassembly((char*)ptr,Len) << "\n";
        ptr += Len;
    }
    emu->MakeDumpFile("upx01_unpacked.exe",DUMP_FIXIMPORTTABLE);
    system("pause");
    delete emu;
    return 0;
}
//*/
```

Demo: Unpack UPX – Run Code



```
Start Emulation From : 00406380
-------------------------------
Press any key to continue . . .
Emulated Successfully

The Disassembled Code:
----------------------
00401000 : push 0h
00401002 : call 247h
00401007 : mov dword ptr [4033b0h],eax
0040100C : call 237h
00401011 : mov dword ptr [4033b4h],eax
00401016 : push 0ah
00401018 : push dword ptr [4033b4h]
0040101E : push 0h
00401020 : push dword ptr [4033b0h]
00401026 : call 6h
0040102B : push eax
0040102C : call 211h
00401031 : push ebp
00401032 : mov ebp ,esp
00401034 : add esp ,0ffffffb0h
00401037 : mov dword ptr [ebp - 30h],30h
0040103E : mov dword ptr [ebp - 2ch],3h
00401045 : mov dword ptr [ebp - 28h],401119h
0040104C : push dword ptr [ebp + 8h]
```

Demo: Unpack UPX - Unpacked



PEiD v0.95

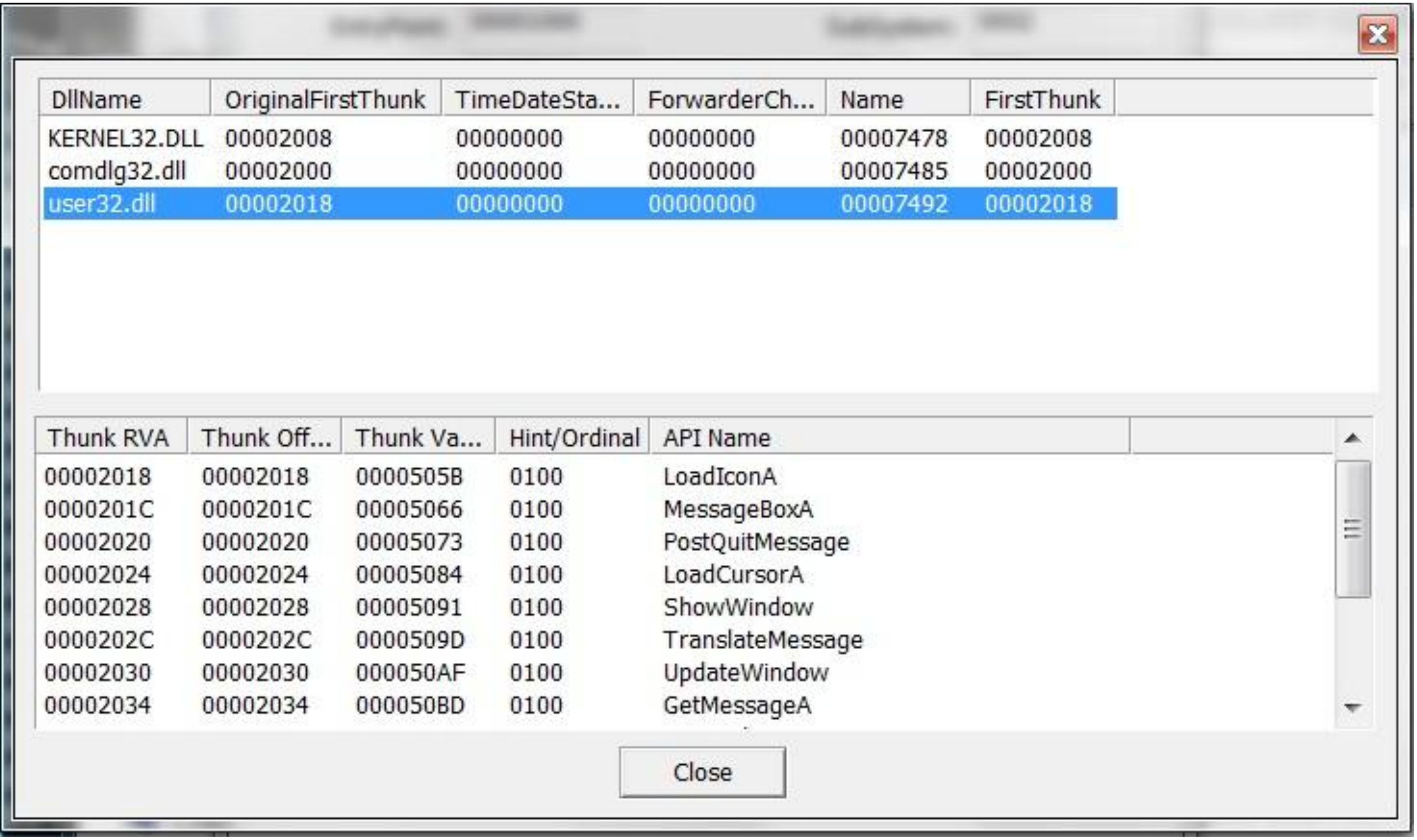
File: D:\Personal\C\SRDF\winSRDF\User-Mode\Release\upx01_unpacked

Entrypoint:	00001000	EP Section:	UPX0
File Offset:	00001000	First Bytes:	6A,00,E8,47
Linker Info:	5.12	Subsystem:	Win32 GUI

MASM32 / TASM32 [Overlay]

Name	V. Offset	V. Size	R. Offset	R. Size	Flags
UPX0	00001...	00005000	00001...	00005000	E0000...
UPX1	00006...	00001000	00006...	00001000	E0000...
.rsrc	00007...	00001000	00007...	00001000	C0000...
idata	00008...	00001000	00008...	00001000	E0000...

Demo: Unpack UPX - ImportTable



DllName	OriginalFirstThunk	TimeDateSta...	ForwarderCh...	Name	FirstThunk
KERNEL32.DLL	00002008	00000000	00000000	00007478	00002008
comdlg32.dll	00002000	00000000	00000000	00007485	00002000
user32.dll	00002018	00000000	00000000	00007492	00002018

Thunk RVA	Thunk Off...	Thunk Va...	Hint/Ordinal	API Name
00002018	00002018	0000505B	0100	LoadIconA
0000201C	0000201C	00005066	0100	MessageBoxA
00002020	00002020	00005073	0100	PostQuitMessage
00002024	00002024	00005084	0100	LoadCursorA
00002028	00002028	00005091	0100	ShowWindow
0000202C	0000202C	0000509D	0100	TranslateMessage
00002030	00002030	000050AF	0100	UpdateWindow
00002034	00002034	000050BD	0100	GetMessageA

x86 Emulator

- Reach it at:

  https://github.com/AmrThabet/x86Emulator

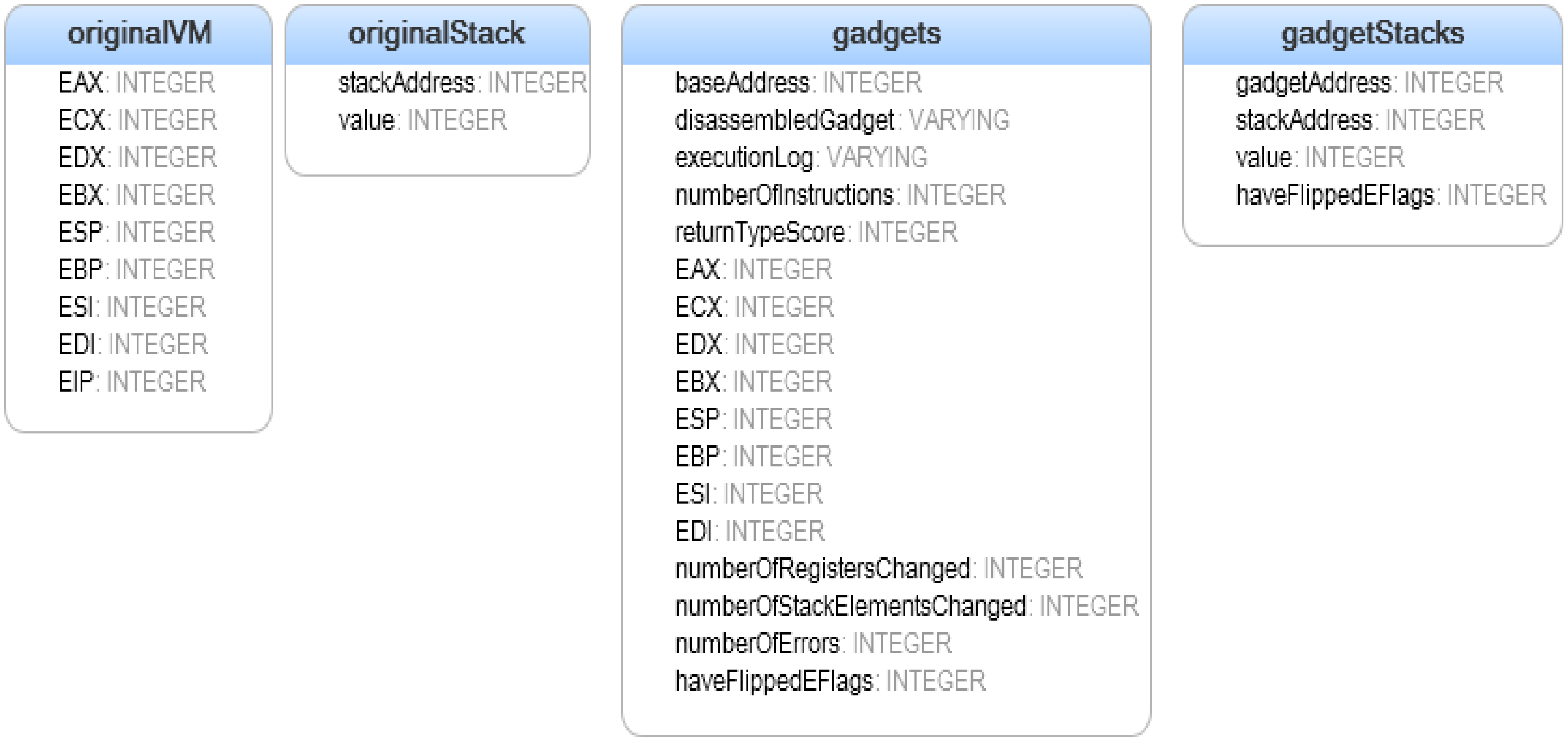
Projects Based on SRDF

- Inspector’s Gadget
- Exploitation Detection System

Inspector’s Gadget

- Created by Jonas lykkegaard
- ROP gadget indexing and searching tool.
- Emulating Gadgets
- Scoring and Categorizing
- Flexible Search

Design



originalVM

EAX: INTEGER
ECX: INTEGER
EDX: INTEGER
EBX: INTEGER
ESP: INTEGER
EBP: INTEGER
ESI: INTEGER
EDI: INTEGER
EIP: INTEGER

originalStack

stackAddress: INTEGER
value: INTEGER

gadgets

baseAddress: INTEGER
disassembledGadget: VARYING
executionLog: VARYING
numberOfInstructions: INTEGER
returnTypeScore: INTEGER
EAX: INTEGER
ECX: INTEGER
EDX: INTEGER
EBX: INTEGER
ESP: INTEGER
EBP: INTEGER
ESI: INTEGER
EDI: INTEGER
numberOfRegistersChanged: INTEGER
numberOfStackElementsChanged: INTEGER
numberOfErrors: INTEGER
haveFlippedEFlags: INTEGER

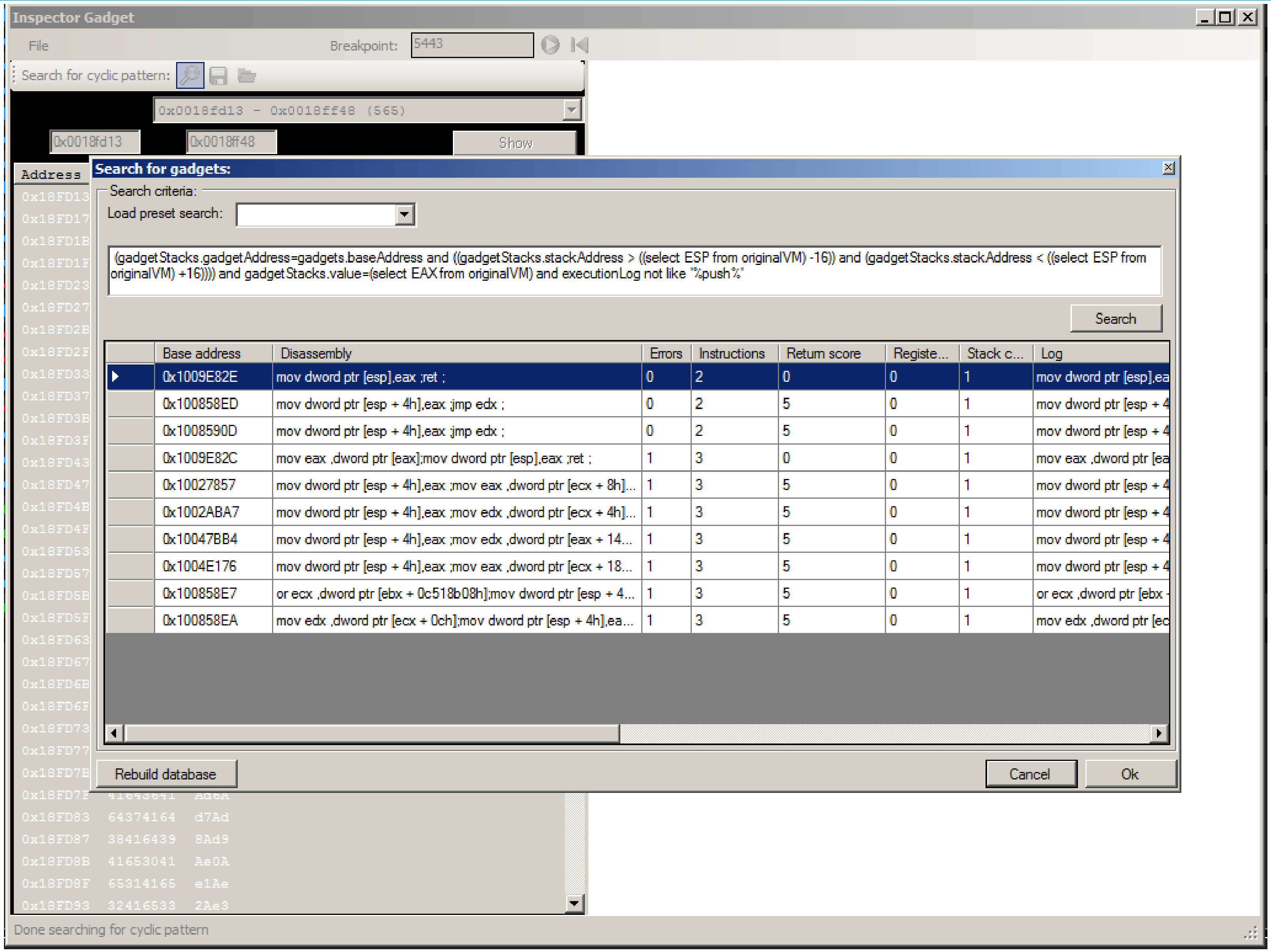
gadgetStacks

gadgetAddress: INTEGER
stackAddress: INTEGER
value: INTEGER
haveFlippedEFlags: INTEGER

Features

- Categorizing by Behavior
- Scoring Gadgets
- Allow *ret, pop/jmp, iret and ret far*
- *Depends on SQLite*
- *SQL Searching*
- *Predefined SQL Queries*
- *GUI Based*

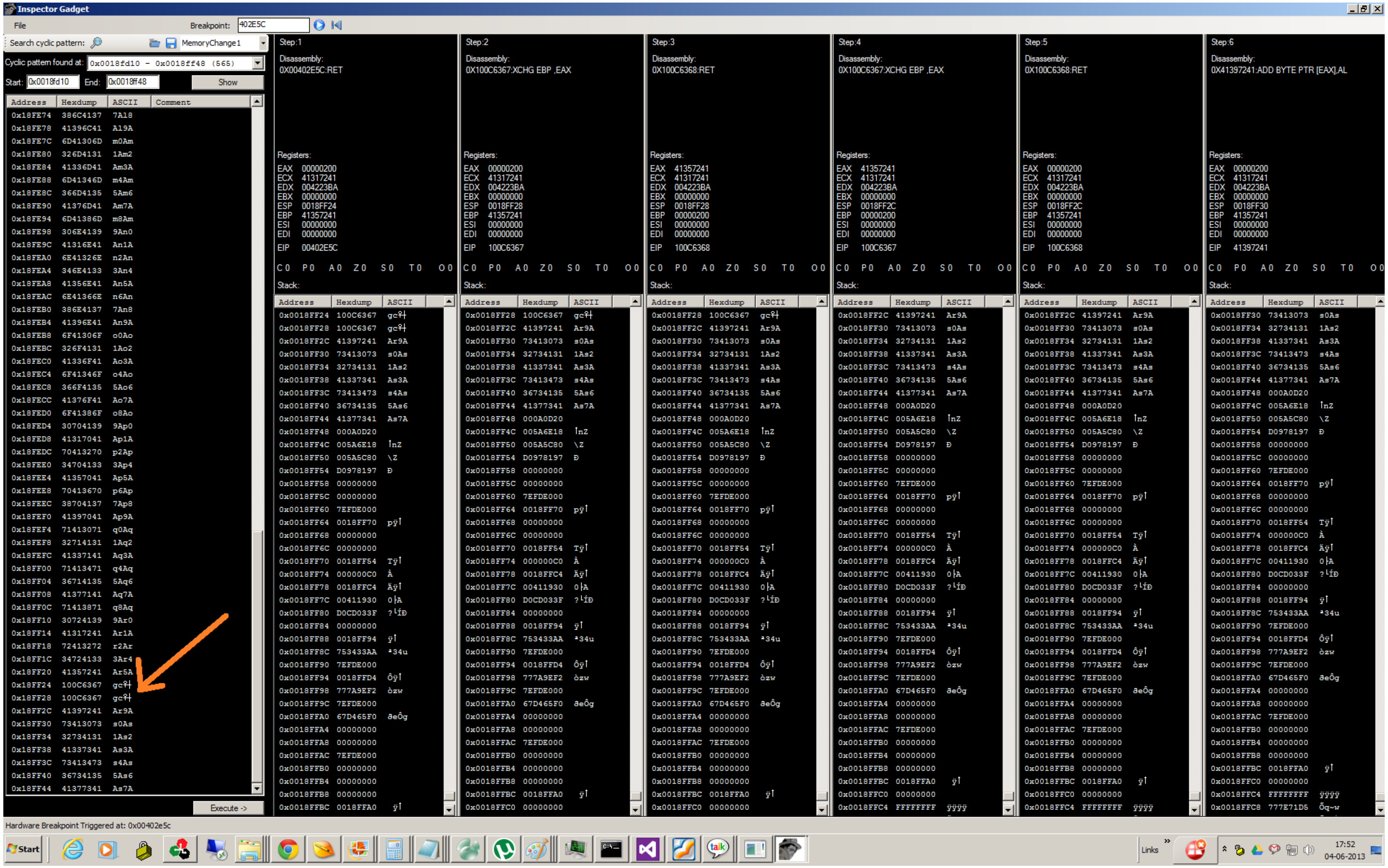
GUI



```
(gadgetStacks.gadgetAddress=gadgets.baseAddress and ((gadgetStacks.stackAddress > ((select ESP from originalVM) -16)) and (gadgetStacks.stackAddress < ((select ESP from originalVM) +16)))) and gadgetStacks.value=(select EAX from originalVM) and executionLog not like '%push%'
```

Base address	Disassembly	Errors	Instructions	Return score	Registe...	Stack c...	Log
0x1009E82E	mov dword ptr [esp],eax ;ret ;	0	2	0	0	1	mov dword ptr [esp],ea
0x100858ED	mov dword ptr [esp + 4h],eax ;jmp edx ;	0	2	5	0	1	mov dword ptr [esp + 4
0x1008590D	mov dword ptr [esp + 4h],eax ;jmp edx ;	0	2	5	0	1	mov dword ptr [esp + 4
0x1009E82C	mov eax ,dword ptr [eax];mov dword ptr [esp],eax ;ret ;	1	3	0	0	1	mov eax ,dword ptr [ea
0x10027857	mov dword ptr [esp + 4h],eax ;mov eax ,dword ptr [ecx + 8h]...	1	3	5	0	1	mov dword ptr [esp + 4
0x1002ABA7	mov dword ptr [esp + 4h],eax ;mov edx ,dword ptr [ecx + 4h]...	1	3	5	0	1	mov dword ptr [esp + 4
0x10047BB4	mov dword ptr [esp + 4h],eax ;mov edx ,dword ptr [eax + 14...	1	3	5	0	1	mov dword ptr [esp + 4
0x1004E176	mov dword ptr [esp + 4h],eax ;mov eax ,dword ptr [ecx + 18...	1	3	5	0	1	mov dword ptr [esp + 4
0x100858E7	or ecx ,dword ptr [ebx + 0c518b08h];mov dword ptr [esp + 4...	1	3	5	0	1	or ecx ,dword ptr [ebx -
0x100858EA	mov edx ,dword ptr [ecx + 0ch];mov dword ptr [esp + 4h],ea...	1	3	5	0	1	mov edx ,dword ptr [ec

```
Address
0x18FD13
0x18FD17
0x18FD1B
0x18FD1F
0x18FD23
0x18FD27
0x18FD2B
0x18FD2F
0x18FD33
0x18FD37
0x18FD3B
0x18FD3F
0x18FD43
0x18FD47
0x18FD4B
0x18FD4F
0x18FD53
0x18FD57
0x18FD5B
0x18FD5F
0x18FD63
0x18FD67
0x18FD6B
0x18FD6F
0x18FD73
0x18FD77
0x18FD7B
0x18FD7F  41643641  Ad6A
0x18FD83  64374164  d7Ad
0x18FD87  38416439  8Ad9
0x18FD8B  41653041  Ae0A
0x18FD8F  65314165  e1Ae
0x18FD93  32416533  2Ae3
```



```
Inspector Gadget
File                Breakpoint: 402E5C
Search cyclic pattern:            MemoryChange1
Cyclic pattern found at: 0x0018fd10 - 0x0018ff48 (565)
Start: 0x0018fd10   End: 0x0018ff48   Show

Address    Hexdump   ASCII  Comment
0x18FE74   386C4137  7Al8
0x18FE78   41396C41  Al9A
0x18FE7C   6D41306D  m0Am
0x18FE80   326D4131  1Am2
0x18FE84   41336D41  Am3A
0x18FE88   6D41346D  m4Am
0x18FE8C   366D4135  5Am6
0x18FE90   41376D41  Am7A
0x18FE94   6D41386D  m8Am
0x18FE98   306E4139  9An0
0x18FE9C   41316E41  An1A
0x18FEA0   6E41326E  n2An
0x18FEA4   346E4133  3An4
0x18FEA8   41356E41  An5A
0x18FEAC   6E41366E  n6An
0x18FEB0   386E4137  7An8
0x18FEB4   41396E41  An9A
0x18FEB8   6F41306F  o0Ao
0x18FEBC   326F4131  1Ao2
0x18FEC0   41336F41  Ao3A
0x18FEC4   6F41346F  o4Ao
0x18FEC8   366F4135  5Ao6
0x18FECC   41376F41  Ao7A
0x18FED0   6F41386F  o8Ao
0x18FED4   30704139  9Ap0
0x18FED8   41317041  Ap1A
0x18FEDC   70413270  p2Ap
0x18FEE0   34704133  3Ap4
0x18FEE4   41357041  Ap5A
0x18FEE8   70413670  p6Ap
0x18FEEC   38704137  7Ap8
0x18FEF0   41397041  Ap9A
0x18FEF4   71413071  q0Aq
0x18FEF8   32714131  1Aq2
0x18FEFC   41337141  Aq3A
0x18FF00   71413471  q4Aq
0x18FF04   36714135  5Aq6
0x18FF08   41377141  Aq7A
0x18FF0C   71413871  q8Aq
0x18FF10   30724139  9Ar0
0x18FF14   41317241  Ar1A
0x18FF18   72413272  r2Ar
0x18FF1C   34724133  3Ar4
0x18FF20   41357241  Ar5A
0x18FF24   100C6367  gc♀╢
0x18FF28   100C6367  gc♀╢
0x18FF2C   41397241  Ar9A
0x18FF30   73413073  s0As
0x18FF34   32734131  1As2
0x18FF38   41337341  As3A
0x18FF3C   73413473  s4As
0x18FF40   36734135  5As6
0x18FF44   41377341  As7A
                                   Execute ->
```

```
Step:1
Disassembly:
0X00402E5C:RET

Registers:
EAX  00000200
ECX  41317241
EDX  004223BA
EBX  00000000
ESP  0018FF24
EBP  41357241
ESI  00000000
EDI  00000000

EIP  00402E5C

C 0  P 0  A 0  Z 0  S 0  T 0  O 0
Stack:
Address     Hexdump   ASCII
0x0018FF24  100C6367  gc♀╢
0x0018FF28  100C6367  gc♀╢
0x0018FF2C  41397241  Ar9A
0x0018FF30  73413073  s0As
0x0018FF34  32734131  1As2
0x0018FF38  41337341  As3A
0x0018FF3C  73413473  s4As
0x0018FF40  36734135  5As6
0x0018FF44  41377341  As7A
0x0018FF48  000A0D20
0x0018FF4C  005A6E18  ↑nZ
0x0018FF50  005A5C80  \Z
0x0018FF54  D0978197  Ð
0x0018FF58  00000000
0x0018FF5C  00000000
0x0018FF60  7EFDE000
0x0018FF64  0018FF70  pÿ↑
0x0018FF68  00000000
0x0018FF6C  00000000
0x0018FF70  0018FF54  Tÿ↑
0x0018FF74  000000C0  À
0x0018FF78  0018FFC4  Äÿ↑
0x0018FF7C  00411930  0↓A
0x0018FF80  D0CD033F  ?└ÍÐ
0x0018FF84  00000000
0x0018FF88  0018FF94  ÿ↑
0x0018FF8C  753433AA  ª34u
0x0018FF90  7EFDE000
0x0018FF94  0018FFD4  Ôÿ↑
0x0018FF98  777A9EF2  òzw
0x0018FF9C  7EFDE000
0x0018FFA0  67D465F0  ðeÔg
0x0018FFA4  00000000
0x0018FFA8  00000000
0x0018FFAC  7EFDE000
0x0018FFB0  00000000
0x0018FFB4  00000000
0x0018FFB8  00000000
0x0018FFBC  0018FFA0   ÿ↑
```

```
Step:2
Disassembly:
0X100C6367:XCHG EBP ,EAX

Registers:
EAX  00000200
ECX  41317241
EDX  004223BA
EBX  00000000
ESP  0018FF28
EBP  41357241
ESI  00000000
EDI  00000000

EIP  100C6367

C 0  P 0  A 0  Z 0  S 0  T 0  O 0
Stack:
Address     Hexdump   ASCII
0x0018FF28  100C6367  gc♀╢
0x0018FF2C  41397241  Ar9A
0x0018FF30  73413073  s0As
0x0018FF34  32734131  1As2
0x0018FF38  41337341  As3A
0x0018FF3C  73413473  s4As
0x0018FF40  36734135  5As6
0x0018FF44  41377341  As7A
0x0018FF48  000A0D20
0x0018FF4C  005A6E18  ↑nZ
0x0018FF50  005A5C80  \Z
0x0018FF54  D0978197  Ð
0x0018FF58  00000000
0x0018FF5C  00000000
0x0018FF60  7EFDE000
0x0018FF64  0018FF70  pÿ↑
0x0018FF68  00000000
0x0018FF6C  00000000
0x0018FF70  0018FF54  Tÿ↑
0x0018FF74  000000C0  À
0x0018FF78  0018FFC4  Äÿ↑
0x0018FF7C  00411930  0↓A
0x0018FF80  D0CD033F  ?└ÍÐ
0x0018FF84  00000000
0x0018FF88  0018FF94  ÿ↑
0x0018FF8C  753433AA  ª34u
0x0018FF90  7EFDE000
0x0018FF94  0018FFD4  Ôÿ↑
0x0018FF98  777A9EF2  òzw
0x0018FF9C  7EFDE000
0x0018FFA0  67D465F0  ðeÔg
0x0018FFA4  00000000
0x0018FFA8  00000000
0x0018FFAC  7EFDE000
0x0018FFB0  00000000
0x0018FFB4  00000000
0x0018FFB8  00000000
0x0018FFBC  0018FFA0   ÿ↑
0x0018FFC0  00000000
```

```
Step:3
Disassembly:
0X100C6368:RET

Registers:
EAX  41357241
ECX  41317241
EDX  004223BA
EBX  00000000
ESP  0018FF28
EBP  00000200
ESI  00000000
EDI  00000000

EIP  100C6368

C 0  P 0  A 0  Z 0  S 0  T 0  O 0
Stack:
Address     Hexdump   ASCII
0x0018FF28  100C6367  gc♀╢
0x0018FF2C  41397241  Ar9A
0x0018FF30  73413073  s0As
0x0018FF34  32734131  1As2
0x0018FF38  41337341  As3A
0x0018FF3C  73413473  s4As
0x0018FF40  36734135  5As6
0x0018FF44  41377341  As7A
0x0018FF48  000A0D20
0x0018FF4C  005A6E18  ↑nZ
0x0018FF50  005A5C80  \Z
0x0018FF54  D0978197  Ð
0x0018FF58  00000000
0x0018FF5C  00000000
0x0018FF60  7EFDE000
0x0018FF64  0018FF70  pÿ↑
0x0018FF68  00000000
0x0018FF6C  00000000
0x0018FF70  0018FF54  Tÿ↑
0x0018FF74  000000C0  À
0x0018FF78  0018FFC4  Äÿ↑
0x0018FF7C  00411930  0↓A
0x0018FF80  D0CD033F  ?└ÍÐ
0x0018FF84  00000000
0x0018FF88  0018FF94  ÿ↑
0x0018FF8C  753433AA  ª34u
0x0018FF90  7EFDE000
0x0018FF94  0018FFD4  Ôÿ↑
0x0018FF98  777A9EF2  òzw
0x0018FF9C  7EFDE000
0x0018FFA0  67D465F0  ðeÔg
0x0018FFA4  00000000
0x0018FFA8  00000000
0x0018FFAC  7EFDE000
0x0018FFB0  00000000
0x0018FFB4  00000000
0x0018FFB8  00000000
0x0018FFBC  0018FFA0   ÿ↑
0x0018FFC0  00000000
```

```
Step:4
Disassembly:
0X100C6367:XCHG EBP ,EAX

Registers:
EAX  41357241
ECX  41317241
EDX  004223BA
EBX  00000000
ESP  0018FF2C
EBP  00000200
ESI  00000000
EDI  00000000

EIP  100C6367

C 0  P 0  A 0  Z 0  S 0  T 0  O 0
Stack:
Address     Hexdump   ASCII
0x0018FF2C  41397241  Ar9A
0x0018FF30  73413073  s0As
0x0018FF34  32734131  1As2
0x0018FF38  41337341  As3A
0x0018FF3C  73413473  s4As
0x0018FF40  36734135  5As6
0x0018FF44  41377341  As7A
0x0018FF48  000A0D20
0x0018FF4C  005A6E18  ↑nZ
0x0018FF50  005A5C80  \Z
0x0018FF54  D0978197  Ð
0x0018FF58  00000000
0x0018FF5C  00000000
0x0018FF60  7EFDE000
0x0018FF64  0018FF70  pÿ↑
0x0018FF68  00000000
0x0018FF6C  00000000
0x0018FF70  0018FF54  Tÿ↑
0x0018FF74  000000C0  À
0x0018FF78  0018FFC4  Äÿ↑
0x0018FF7C  00411930  0↓A
0x0018FF80  D0CD033F  ?└ÍÐ
0x0018FF84  00000000
0x0018FF88  0018FF94  ÿ↑
0x0018FF8C  753433AA  ª34u
0x0018FF90  7EFDE000
0x0018FF94  0018FFD4  Ôÿ↑
0x0018FF98  777A9EF2  òzw
0x0018FF9C  7EFDE000
0x0018FFA0  67D465F0  ðeÔg
0x0018FFA4  00000000
0x0018FFA8  00000000
0x0018FFAC  7EFDE000
0x0018FFB0  00000000
0x0018FFB4  00000000
0x0018FFB8  00000000
0x0018FFBC  0018FFA0   ÿ↑
0x0018FFC0  00000000
0x0018FFC4  FFFFFFFF  ÿÿÿÿ
```

```
Step:5
Disassembly:
0X100C6368:RET

Registers:
EAX  00000200
ECX  41317241
EDX  004223BA
EBX  00000000
ESP  0018FF2C
EBP  41357241
ESI  00000000
EDI  00000000

EIP  100C6368

C 0  P 0  A 0  Z 0  S 0  T 0  O 0
Stack:
Address     Hexdump   ASCII
0x0018FF2C  41397241  Ar9A
0x0018FF30  73413073  s0As
0x0018FF34  32734131  1As2
0x0018FF38  41337341  As3A
0x0018FF3C  73413473  s4As
0x0018FF40  36734135  5As6
0x0018FF44  41377341  As7A
0x0018FF48  000A0D20
0x0018FF4C  005A6E18  ↑nZ
0x0018FF50  005A5C80  \Z
0x0018FF54  D0978197  Ð
0x0018FF58  00000000
0x0018FF5C  00000000
0x0018FF60  7EFDE000
0x0018FF64  0018FF70  pÿ↑
0x0018FF68  00000000
0x0018FF6C  00000000
0x0018FF70  0018FF54  Tÿ↑
0x0018FF74  000000C0  À
0x0018FF78  0018FFC4  Äÿ↑
0x0018FF7C  00411930  0↓A
0x0018FF80  D0CD033F  ?└ÍÐ
0x0018FF84  00000000
0x0018FF88  0018FF94  ÿ↑
0x0018FF8C  753433AA  ª34u
0x0018FF90  7EFDE000
0x0018FF94  0018FFD4  Ôÿ↑
0x0018FF98  777A9EF2  òzw
0x0018FF9C  7EFDE000
0x0018FFA0  67D465F0  ðeÔg
0x0018FFA4  00000000
0x0018FFA8  00000000
0x0018FFAC  7EFDE000
0x0018FFB0  00000000
0x0018FFB4  00000000
0x0018FFB8  00000000
0x0018FFBC  0018FFA0   ÿ↑
0x0018FFC0  00000000
0x0018FFC4  FFFFFFFF  ÿÿÿÿ
```

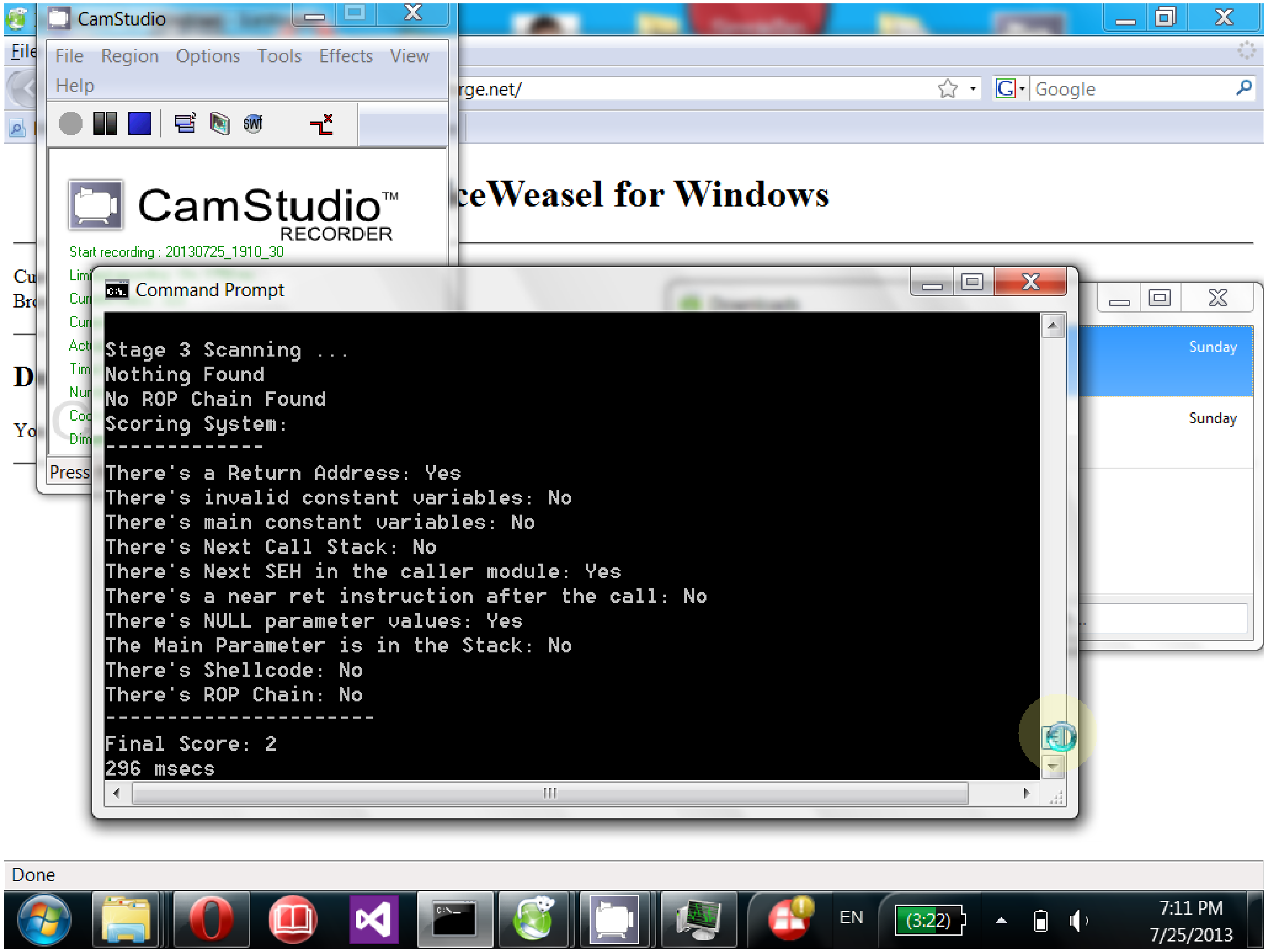
```
Step:6
Disassembly:
0X41397241:ADD BYTE PTR [EAX],AL

Registers:
EAX  00000200
ECX  41317241
EDX  004223BA
EBX  00000000
ESP  0018FF30
EBP  41357241
ESI  00000000
EDI  00000000

EIP  41397241

C 0  P 0  A 0  Z 0  S 0  T 0  O 0
Stack:
Address     Hexdump   ASCII
0x0018FF30  73413073  s0As
0x0018FF34  32734131  1As2
0x0018FF38  41337341  As3A
0x0018FF3C  73413473  s4As
0x0018FF40  36734135  5As6
0x0018FF44  41377341  As7A
0x0018FF48  000A0D20
0x0018FF4C  005A6E18  ↑nZ
0x0018FF50  005A5C80  \Z
0x0018FF54  D0978197  Ð
0x0018FF58  00000000
0x0018FF5C  00000000
0x0018FF60  7EFDE000
0x0018FF64  0018FF70  pÿ↑
0x0018FF68  00000000
0x0018FF6C  00000000
0x0018FF70  0018FF54  Tÿ↑
0x0018FF74  000000C0  À
0x0018FF78  0018FFC4  Äÿ↑
0x0018FF7C  00411930  0↓A
0x0018FF80  D0CD033F  ?└ÍÐ
0x0018FF84  00000000
0x0018FF88  0018FF94  ÿ↑
0x0018FF8C  753433AA  ª34u
0x0018FF90  7EFDE000
0x0018FF94  0018FFD4  Ôÿ↑
0x0018FF98  777A9EF2  òzw
0x0018FF9C  7EFDE000
0x0018FFA0  67D465F0  ðeÔg
0x0018FFA4  00000000
0x0018FFA8  00000000
0x0018FFAC  7EFDE000
0x0018FFB0  00000000
0x0018FFB4  00000000
0x0018FFB8  00000000
0x0018FFBC  0018FFA0   ÿ↑
0x0018FFC0  00000000
0x0018FFC4  FFFFFFFF  ÿÿÿÿ
0x0018FFC8  777E71D5  Õq~w
```

```
Hardware Breakpoint Triggered at: 0x00402e5c
```

Exploitation Detection System

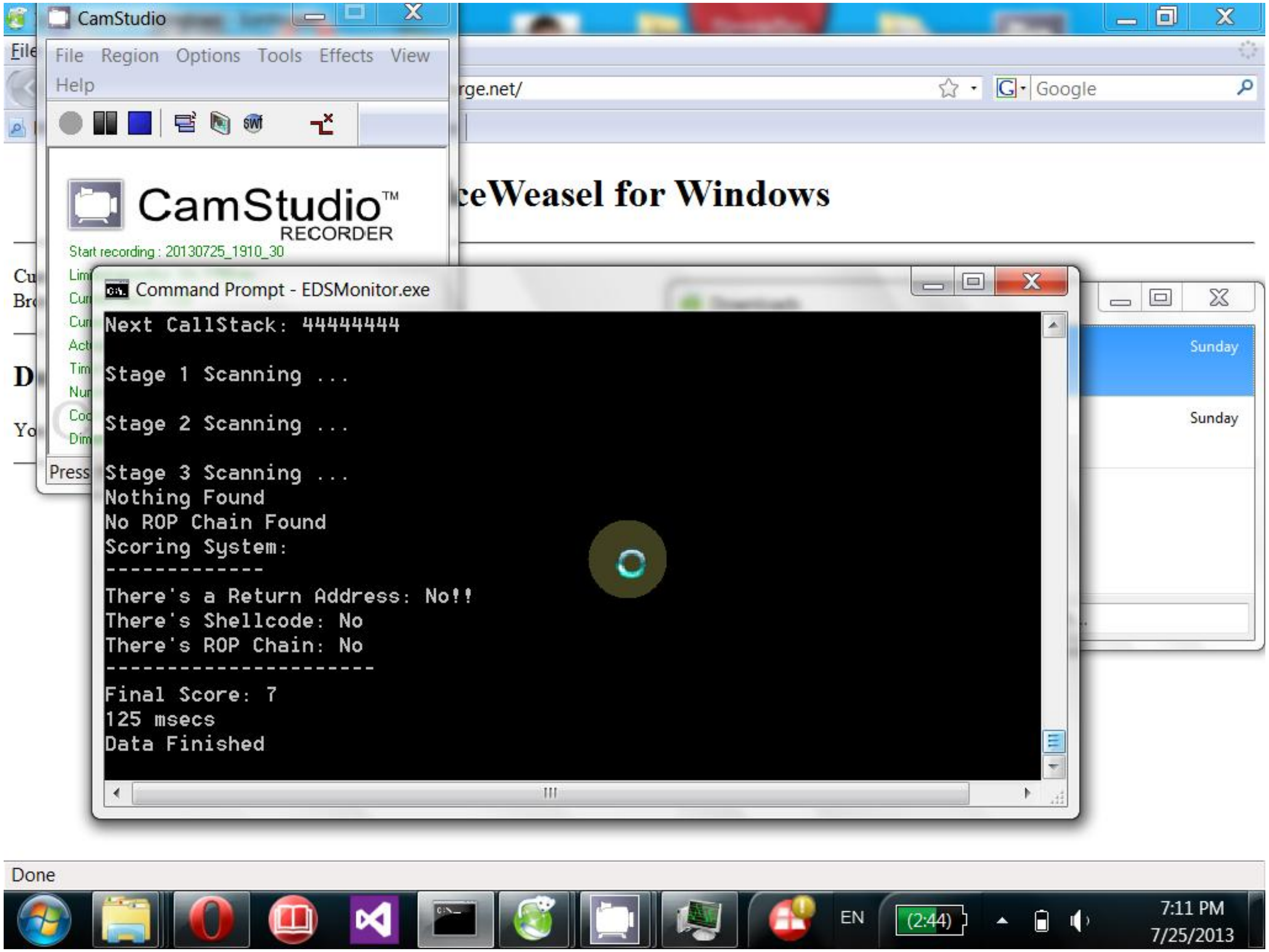
- Security Mitigation Tool
- Detect memory corruption exploits
- Based on SRDF
- Talked about it in DEFCON
- Reach it at:Defcon 21 archive

Normal API call check



```
Stage 3 Scanning ...
Nothing Found
No ROP Chain Found
Scoring System:
-------------
There's a Return Address: Yes
There's invalid constant variables: No
There's main constant variables: No
There's Next Call Stack: No
There's Next SEH in the caller module: Yes
There's a near ret instruction after the call: No
There's NULL parameter values: Yes
The Main Parameter is in the Stack: No
There's Shellcode: No
There's ROP Chain: No
----------------------
Final Score: 2
296 msecs
```

API call with Ret



```
Next CallStack: 44444444

Stage 1 Scanning ...

Stage 2 Scanning ...

Stage 3 Scanning ...
Nothing Found
No ROP Chain Found
Scoring System:
-------------
There's a Return Address: No!!
There's Shellcode: No
There's ROP Chain: No
----------------------
Final Score: 7
125 msecs
Data Finished
```

CamStudio
File Region Options Tools Effects View Help
CamStudio™ RECORDER
Start recording : 20130725_1910_30
Command Prompt - EDSMonitor.exe
ceWeasel for Windows
rge.net/
Google
Sunday
Sunday
Done
EN
(2:44)
7:11 PM
7/25/2013

Reach Us

- SRDF Links:
  - https://github.com/AmrThabet/winSRDF
  - FB: http://www.facebook.com/SecDevelop
  - Twitter: https://twitter.com/winSRDF
  - Website: http://security-framework.com/

Conclusion

- Development Framework for security
- Contains many tools in Malware and Network
- Flexible expandable Design
- Kernel-Mode and User-Mode
- Free and Open Source
- Join Us

Any Question?

Thank You !