

Software analyzing

Реверс Інжиніринг

- Реверс інжиніринг - процес "дослідження", відновлення вихідників з кінцевого продукту інженерної або наукової діяльності, інтуїтивно конструюючи внутрішню механіку за принципом "а які процеси повинні викликати таку зовнішню поведінку цього продукту?»
- Навіщо потрібний реверс інжиніринг?
 - Відновлення принципів, ідей, алгоритмів роботи програми для дослідження та/або створення аналогічного ПЗ
 - Аналіз вірусів, троянів, хробаків з метою створення засобів захисту
 - Аналіз роботи пропрієтарного програмного забезпечення, драйверів для створення відкритих, загальнодоступних аналогів
 - Пошук дірок у пропрієтарному або відкритому ПЗ з метою створення вірусів, троянів, сплойтів

Що перешкоджає реверс інжинірингу

- "поганий код"
- Антиналагодження
- Обфускація
- Пакувальники
- Протектори
- Віртуальні машини
- Звернення до “зломників” усередині виконуваного файлу у вигляді ASCII-рядків

Утиліти Реверс Інженірингу (C/C++)

- Дизасемблери
- Компілятори
- Обфускатори
- Відладчики
- PE-редактори
- HEX-редактори

PE TOOLS

Тип: PE-редактор

Інтерфейс: GUI

Ліцензія: MIT

Операционные системы: Windows

BOOMERANG

Тип: Декомпілятор

Інтерфейс: CLI, GUI

Ліцензія: MIT

Операционные системы: Linux, BSD, OS X, Windows

Поддерживаемые архитектуры: x86 (IA-32), SPARC (V8/V9), PPC, ST20

Утиліти Реверс Інженірингу(C/C++)

x64DBG

Тип: Отладчик
Интерфейс: GUI
Лицензия: GPL 3.0
ОС: Windows

RECSTUDIO

Тип: Декомпилятор
Интерфейс: GUI
ОС : Linux, BSD, OS X, Windows
Поддерживаемые архитектуры: x86 (IA-32), x86-64, MIPS, PPC, mc68k

GHIDRA

Тип: Дизассемблер и фреймворк для реверс-инжиниринга
Авторы: Агентство национальной безопасности США
Интерфейс: GUI
Лицензия: Apache License 2.0
ОС : Linux, BSD, OS X, Windows

IDA + HEX RAYS

Тип: Дизассемблер и фреймворк для реверс-инжиниринга
Интерфейс: GUI
Лицензия: Shareware
ОС : Linux, BSD, OS X, Windows

Інші Утиліти Реверс Інженірингу

- Delphi
 - Delphi Decompiler: DeDe
- Java
 - Jdgui
 - Fernflower (IntelliJ)
- C#
 - dotPeek

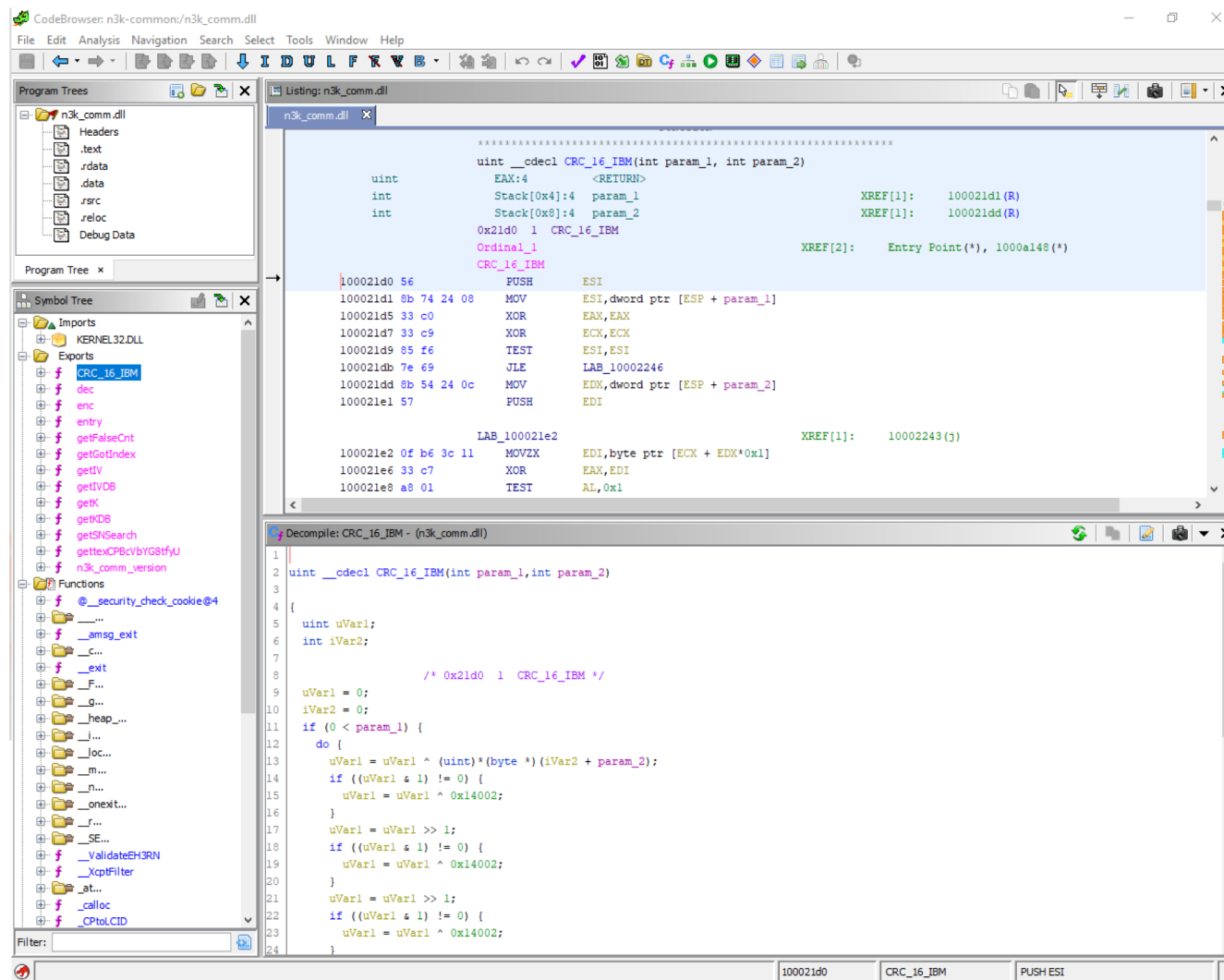
Реверс Інжиніринг С/С++ бібліотек.

Перелік export функцій

- “\$VSTOOLS_PATH\dumpbin.exe” /exports n3k_comm.dll
 - ...
 - 1 0 000021D0 CRC_16_IBM
 - 2 1 00002310 dec
 - 3 2 00002250 enc
 - 4 3 00002690 getFalseCnt
 - 5 4 00002570 getGotIndex
 - 6 5 00002420 getIV
 - 7 6 00002480 getIVDB
 - 8 7 000023F0 getK
 - 9 8 00002450 getKDB
 - 10 9 000024B0 getSNSearch
 - 11 A 000023E0 gettexCPBcVbYG8tfyU
 - 12 B 000021C0 n3k_comm_version

Реверс Інжиніринг С/С++ бібліотек.

Перелік export функцій



The screenshot displays the CodeBrowser application interface for the file `n3k-common/n3k_comm.dll`. The **Program Tree** on the left shows the file structure, including `Headers`, `.text`, `.rdata`, `.data`, `.rsrc`, `.reloc`, and `Debug Data`. The **Symbol Tree** on the left lists various symbols, including `CRC_16_IBM`, `dec`, `enc`, `entry`, `getFalseCnt`, `getGotIndex`, `getIV`, `getIVDB`, `getK`, `getKDB`, `getSearch`, `gettexCPBcbvYG8tfyU`, `n3k_comm_version`, and several functions like `@_security_check_cookie@4`, `__amsg_exit`, `__C...`, `__exit`, `__F...`, `__g...`, `__heap...`, `__j...`, `__loc...`, `__m...`, `__n...`, `__onexit...`, `__r...`, `__SE...`, `__ValidateEH3RN`, `__XcptFilter`, `__at...`, `__calloc`, and `__CptolCID`.

The main window displays the assembly code for the `CRC_16_IBM` function. The assembly code is as follows:

```
uint __cdecl CRC_16_IBM(int param_1, int param_2)
{
    EAX:4 <RETURN>
    Stack[0x4]:4 param_1 XREF[1]: 100021d1(R)
    Stack[0x8]:4 param_2 XREF[1]: 100021dd(R)
    0x21d0 1 CRC_16_IBM XREF[2]: Entry Point(*), 1000a148(*)
    Ordinal_1
    CRC_16_IBM
    100021d0 56 PUSH ESI
    100021d1 8b 74 24 08 MOV ESI,dword ptr [ESP + param_1]
    100021d5 33 c0 XOR EAX,EAX
    100021d7 33 c9 XOR ECX,ECX
    100021d9 85 f6 TEST ESI,ESI
    100021db 7e 69 JLE LAB_10002246
    100021dd 8b 54 24 0c MOV EDI,dword ptr [ESP + param_2]
    100021e1 57 PUSH EDI
    LAB_10002246
    100021e2 0f b6 3c 11 MOVZX EDI,byte ptr [ECX + EDI*0x1]
    100021e6 33 c7 XOR EAX,EDI
    100021e8 a8 01 TEST AL,0x1
    XREF[1]: 10002243(j)
}
```

The bottom window shows the decompiled C code for the `CRC_16_IBM` function:

```
1 |
2 | uint __cdecl CRC_16_IBM(int param_1, int param_2)
3 |
4 | {
5 |     uint uVar1;
6 |     int iVar2;
7 |
8 |     /* 0x21d0 1 CRC_16_IBM */
9 |     uVar1 = 0;
10 |    iVar2 = 0;
11 |    if (0 < param_1) {
12 |        do {
13 |            uVar1 = uVar1 ^ (uint)(byte *) (iVar2 + param_2);
14 |            if ((uVar1 & 1) != 0) {
15 |                uVar1 = uVar1 ^ 0x14002;
16 |            }
17 |            uVar1 = uVar1 >> 1;
18 |            if ((uVar1 & 1) != 0) {
19 |                uVar1 = uVar1 ^ 0x14002;
20 |            }
21 |            uVar1 = uVar1 >> 1;
22 |            if ((uVar1 & 1) != 0) {
23 |                uVar1 = uVar1 ^ 0x14002;
24 |            }
25 |        } while (param_1 > 0);
26 |    }
27 |    return uVar1;
28 | }
```


Реверс Інжиніринг С/С++ бібліотек. Hex Dump функції CRC_16_IBM

- ЗСУВ НЕХ-ДАМП
- 100021C0 B8 B6 9C 00 00 C3 CC CC CC CC CC CC CC CC CC
- 100021D0 56 8B 74 24 08 33 C0 33 C9 85 F6 7E 69 8B 54 24
- 100021E0 0C 57 0F B6 3C 11 33 C7 A8 01 74 05 35 02 40 01
- 100021F0 00 D1 E8 A8 01 74 05 35 02 40 01 00 D1 E8 A8 01
- 10002200 74 05 35 02 40 01 00 D1 E8 A8 01 74 05 35 02 40
- 10002210 01 00 D1 E8 A8 01 74 05 35 02 40 01 00 D1 E8 A8
- 10002220 01 74 05 35 02 40 01 00 D1 E8 A8 01 74 05 35 02
- 10002230 40 01 00 D1 E8 A8 01 74 05 35 02 40 01 00 D1 E8
- 10002240 41 3B CE 7C 9D 5F 25 FF FF 00 00 5E C3 CC CC CC

Реверс Інжиніринг С/С++ бібліотек.

Дизасемблювання функції CRC_16_IBM (1/3)

```
1      align 10h
2      public CRC_16_IBM
3 CRC_16_IBM    proc near          ; DATA XREF: .rdata:off_1000A148↓o
5 arg_0        = dword ptr 4
6 arg_4        = dword ptr 8
8      push    esi
9      mov     esi, [esp+4+arg_0]
10     xor     eax, eax
11     xor     ecx, ecx
12     test    esi, esi
13     jle     short loc_10002246
14     mov     edx, [esp+4+arg_4]
15     push    edi
17 loc_100021E2:                ; CODE XREF: CRC_16_IBM+73↓j
18     movzx   edi, byte ptr [ecx+edx]
```

Реверс Інжиніринг С/С++ бібліотек.

Дизасемблювання функції CRC_16_IBM (2/3)

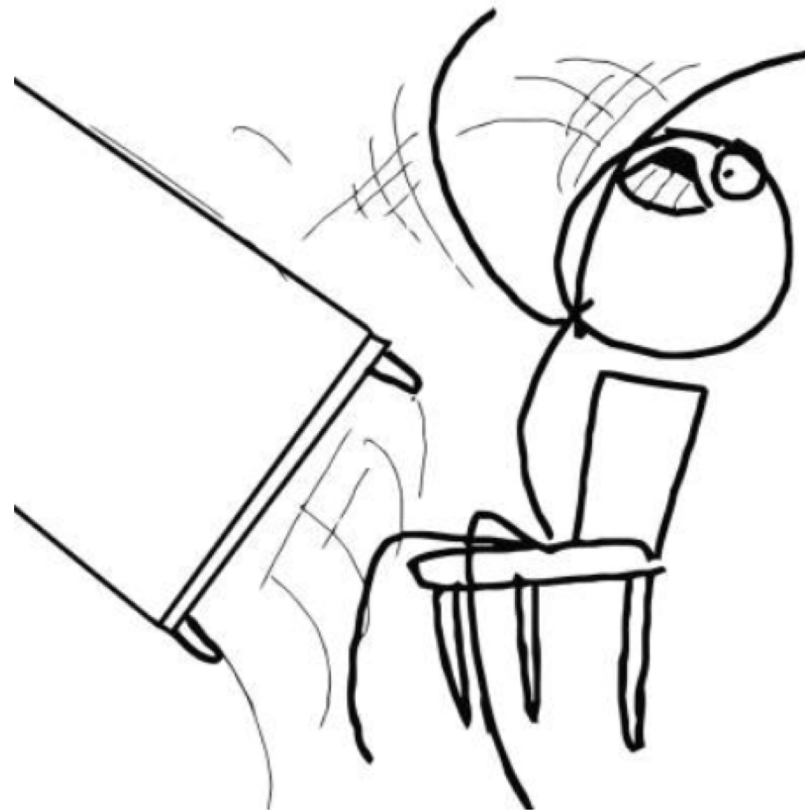
```
19      xor    eax, edi
20      test   al, 1
21      jz     short loc_100021F1
22      xor    eax, 14002h
24 loc_100021F1:                ; CODE XREF: CRC_16_IBM+1A↑j
25      shr    eax, 1
26      test   al, 1
27      jz     short loc_100021FC
28      xor    eax, 14002h
30 loc_100021FC:                ; CODE XREF: CRC_16_IBM+25↑j
31      shr    eax, 1
32      test   al, 1
33      jz     short loc_10002207
34      xor    eax, 14002h
36 loc_10002207:                ; CODE XREF: CRC_16_IBM+30↑j
```

x 3

Реверс Інжиніринг С/С++ бібліотек. Дизасемблювання функції CRC_16_IBM (3/3)

```
72 loc_10002246:                ; CODE XREF: CRC_16_IBM+B↑j
73         and     eax, 0FFFFh
74         pop     esi
75         retn
76 CRC_16_IBM     endp
```

Полезность полученного
исходного кода
сомнительная, так как
требуется значительных усилий
для понимания
происходящего



Реверс Інжиніринг С/С++ бібліотек.

Декомпіляція функції CRC_16_IBM

```
1 uint __cdecl CRC_16_IBM(int param_1, int param_2) {
2     uint uVar1 = 0;
3     int iVar2 = 0;
4
5     if (0 < param_1) {
6         do {
7             uVar1 = uVar1 ^ (uint) * (byte*)(iVar2 + param_2);
8             if ((uVar1 & 1) != 0) {
9                 uVar1 = uVar1 ^ 0x14002;
10            }
11            uVar1 = uVar1 >> 1;
12            if ((uVar1 & 1) != 0) {
13                uVar1 = uVar1 ^ 0x14002;
14            }
15            uVar1 = uVar1 >> 1;
16            if ((uVar1 & 1) != 0) {
17                uVar1 = uVar1 ^ 0x14002;
18            }
19            uVar1 = uVar1 >> 1;
20            if ((uVar1 & 1) != 0) {
21                uVar1 = uVar1 ^ 0x14002;
22            }
```

```
23         uVar1 = uVar1 >> 1;
24         if ((uVar1 & 1) != 0) {
25             uVar1 = uVar1 ^ 0x14002;
26         }
27         // ... (2 blocks)
28         uVar1 = uVar1 >> 1;
29         if ((uVar1 & 1) != 0) {
30             uVar1 = uVar1 ^ 0x14002;
31         }
32         uVar1 = uVar1 >> 1;
33         iVar2 = iVar2 + 1;
34     } while (iVar2 < param_1);
35     return uVar1;
36 }
```

Реверс Інжиніринг C/C++ бібліотек.

Що далі?

- Ми не вміємо зважати на очевидні речі. Це алгоритм CRC-16 IBM (кеп?), вихідний код якого можна знайти в інтернеті (точніше, нам важливе значення полінома - 0xA001)
- Щоб дізнатися, що робить кнопка – натиснути на неї та подивитися, що станеться (с)
 - Не завжди треба реверс інженірувати код функції, щоб зрозуміти, що він робить - достатньо запустити його і знайти закономірність
- Щоб переконатися, що ваш код правильно досліджено (переписано), вам необхідно підключити dll файл до вашого проекту та переконатися, що на ряді наборів даних результати збігаються

HTTP Traffic Analyzer (Wireshark)

Time	Source	Destination	Protocol	Info
2.656833	192.168.1.5	188.184.21.108	HTTP	GET /hypertext/DataSources/Top.html HTTP/1.1
2.725116	188.184.21.108	192.168.1.5	HTTP	HTTP/1.1 200 OK (text/html)
0.162882	149.154.167.51	192.168.1.5	SSL	Continuation Data
0.512397	149.154.167.51	192.168.1.5	SSL	Continuation Data
0.761861	149.154.167.51	192.168.1.5	SSL	Continuation Data
1.814484	149.154.167.51	192.168.1.5	SSL	Continuation Data
2.717634	149.154.167.51	192.168.1.5	SSL	Continuation Data
0.011135	192.168.1.5	20.114.195.241	TCP	60922 → 443 [ACK] Seq=1 Ack=39 Win=6488 Len=0 TSval=330035...
0.057313	52.113.194.132	192.168.1.5	TCP	443 → 62173 [ACK] Seq=1 Ack=1 Win=2049 Len=0
0.142581	192.168.1.5	193.123.151.37	TCP	62702 → 443 [ACK] Seq=51 Ack=67 Win=2048 Len=0 TSval=10150...
0.163086	192.168.1.5	149.154.167.51	TCP	60935 → 443 [ACK] Seq=1 Ack=106 Win=2046 Len=0 TSval=33980...
0.217410	13.83.65.43	192.168.1.5	TCP	443 → 62108 [ACK] Seq=1 Ack=1 Win=2053 Len=0
0.263155	192.168.1.5	20.114.195.241	TCP	60922 → 443 [ACK] Seq=43 Ack=77 Win=6488 Len=0 TSval=33003...
0.380821	20.114.195.241	192.168.1.5	TCP	[TCP Spurious Retransmission] 443 → 60922 [PSH, ACK] Seq=3...
> Frame 125: 962 bytes on wire (7696 bits), 962 bytes captured (7696 bits) on interface en0, id 0 > Ethernet II, Src: Tp-LinkT_86:50:c0 (d4:6e:0e:86:50:c0), Dst: Apple_35:75:dd (50:ed:3c:35:75:dd) > Internet Protocol Version 4, Src: 188.184.21.108, Dst: 192.168.1.5 > Transmission Control Protocol, Src Port: 80, Dst Port: 62841, Seq: 1, Ack: 566, Len: 896 > Hypertext Transfer Protocol > Line-based text data: text/html (17 lines) <title>Overview of the Web</title>\n <h1>General Overview</h1>\n There is no "top" to the World-Wide Web.\n You can look at it from many points of view.\n If you have no other bias, here are some ways of looking for\n information.\n				

```
sudo tcpdump -i en0 -G 3600 -W 6 -w result.pcap 'port 80'
```

Fiddler (with enabled HTTPS decrypt, trial)

Stub Server

- stubby4j
- Sub Server configuration (stub.yml file)
 - request:
 - method: GET
 - url: /verify
 - response:
 - status: 200
 - headers:
 - content-type: application/json
 - body: Valid
- Changes: /etc/hosts
 - 127.0.0.1 verification.site
- Run Stub server
 - java -jar stubby4j-7.5.2.jar -d stub.yml
- Test
 - Request:
 - GET <http://verification.site:8882/verify>
 - (e.g. in browser)
 - curl <http://verification.site:8882/verify>
 - Response: Valid

Redirect from ip (linux):

- `sysctl -w net.ipv4.conf.eth0.route_localnet=1` # to enable redirecting to localhost
- `sudo iptables -t nat -A OUTPUT -d 123.123.123.123 -j DNAT --to-destination 127.0.0.1`

Java Agents

- Java agents є особливим типом класу, який, використовуючи Java Instrumentation API, може інтерпретувати додатки, що запускаються на JVM, змінюючи їх байт-код.
- Instrumentation, в контексті програмного забезпечення, є технологією, що використовується для зміни коду існуючого додатку, фактично не редагуючи файл source коду.

How to use Java Agents

- Dependency:
 - org.javassist:javassist:3.21.0-GA
- Manifest.mf
 - PreMain-Class: <our agent's Main class full qualified name>
 - Optional (if you want to modify victim's code)
 - Can-Redefine-Classes: true
 - Can-Transform-Class: true
- Main Class

```
public static void premain(String args, Instrumentation instrumentation) throws
IOException {
    ClassPool cp = ClassPool.getDefault();
    instrumentation.addTransformer(new CustomTransformer(cp), true);
}
```

How to use Java Agents

- New CustomTransformer
 - CustomTransformer implements ClassFileTransformer
 - `byte[] transform(ClassLoader, String, Class<?>, ProtectionDomain, byte[])`
 - Метод викликається для кожного класу, що завантажується. Ми повинні переконатися, що «зараз» ми опрацьовуємо цільовий клас (його ім'я вказано у другому аргументі)
 - Отримати інформацію про цільовий метод можна за допомогою

```
CtClass cc = cp.get(loadClassName);
CtMethod m = cc.getDeclaredMethod("checkLicenseStatus");
```
 - Змінювати частину методу ми можемо, тому необхідно переписати весь метод повністю, і, за необхідності, його можна спростити

```
String code = "{ return \"License is OK\"; }";
m.setBody(code);
classfileBuffer = cc.toBytecode();
cc.detach();
```
- Make fat jar (with dependencies)
- Run victim with:
 - `-javaagent:agent.jar`