Čudesan svijet reverznog inženjeringa

Branko Spasojević, Symantec

CIS u suradnji s IEEE Hrvatska sekcija Croatia Section

AMAC-FER Hrvatska udruga diplomiranih inženjera Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu

Sadržaj

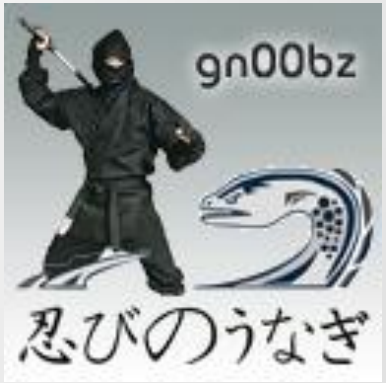
- $whoami
- Motivacija
- Povijesni razvoj
- RE Mete
- Prvi koraci u RE-u
- Metodologije
- RE u znanosti
- Pitanja?

NAPOMENA

- Ovo može izazvati ovisnost!!!
- "And finally, I just wanted to have fun, 'cause you know all those cool kids going out, going to clubs, they just haven't discovered reverse engineering yet." Natalie Silvanovich - 29c3

WHOAMI

- LSS (ZESOI)
- Infigo IS
- Symantec
- CTF član: gn00bz, BAZINGA



Motivacija

- Reverzni inženjering (RE) je proces otkrivanja tehnoloških principa uređaja, objekata ili sustava kroz analizu njegove strukture, funkcije i operacije.
  - Analiza zlućudnog (eng. malware) koda
  - Analiza sklopovlja i embedded uređaja
  - Pronalaženje/iskorištavanje sigurnosnih propusta
  - Zaobilaženje programskih zaštita i DRM sustava (eng. cracking)

- Fun and profit!

Povijesni razvoj RE-a

- 80e
  - ATARI, Comodore64
  - Zaobilaženje zaštita računalnih igara
- 90e
  - Zaobilaženje zaštita računalnih igara i programa
  - Iskorištavanje sigurnosnih propusta
  - Virusi
- 21. st. ☺
  - Virusi
  - Iskorištavanje sigurnosnih propusta
  - Zaobilaženje programskih zaštita
  - Hardware hacking

Mete za RE

- Native izvršni kod:
  - Intel x86 (PE, ELF), ARM (ELF, firmware), PPC (ELF, firmware)
- VM Bytecode:
  - Java/JVM, Android/Dalvik, Python, VisualBasic (P-CODE), .NET
- Izvorni kod:
  - JavaScript, C/C++, Java
- Strukture datoteka/podataka:
  - Baze podataka, serijalizirani objekti
- Mrežni protokoli:
  - Internet bankarstvo, malware C&C

Analiza izvršnog koda

- Komplicirana i vremenski intenzivna
- Pristupi analizi:
  - Duboko razumijevanje: Statička analiza
  - Specifični problem: Dinamička analiza
- Alati koji se koriste:
  - Debugger: OllyDbg, Immunity Debugger, Windbg, gdb
  - Disassembler: IDA Pro, Hopper
  - API monitoring: Procmon, APIMonitor
  - Decompiler: HexRays, Hopper

DEMO

- Analiza izvršnog koda

Analiza bytecodea

- U pravilu jednostavnije nego analiza izvršnog koda
- Debugging nešto otežan zbog korištenja VM interpretera
- Analiza olakšana s dotupnošću dekompajlera
- Pristupi analizi:
  - Statička analiza
  - Dinamička analiza
- Alati:
  - Decompiler: dotPeek, Reflector, VB-decompiler, JD-GUI...
  - Disassembler: IDA Pro, javap, dis...

DEMO

- Analiza VM bytecodea

Prvi koraci u RE-u

Prvi koraci u RE-u

- Korak 1: Odabir područja interesa
  - Analiza malwarea
  - Pronalaženje sigurnosnih propusta
  - Iskorištavanje sigurnosnih propusta
  - Analiza programskih zaštita
  - Hardware hacking

Prvi koraci u RE-u

- Korak 2: Literatura i resursi
  - RE Reddit
  - IRC (##re na freenodeu)
  - Knjige (Secrets of Reverse Engineering, ...)
  - E-zines (Phrack, #29A, ARTeam, ...)
  - Blogovi
  - CPU reference
  - Google.com ☺

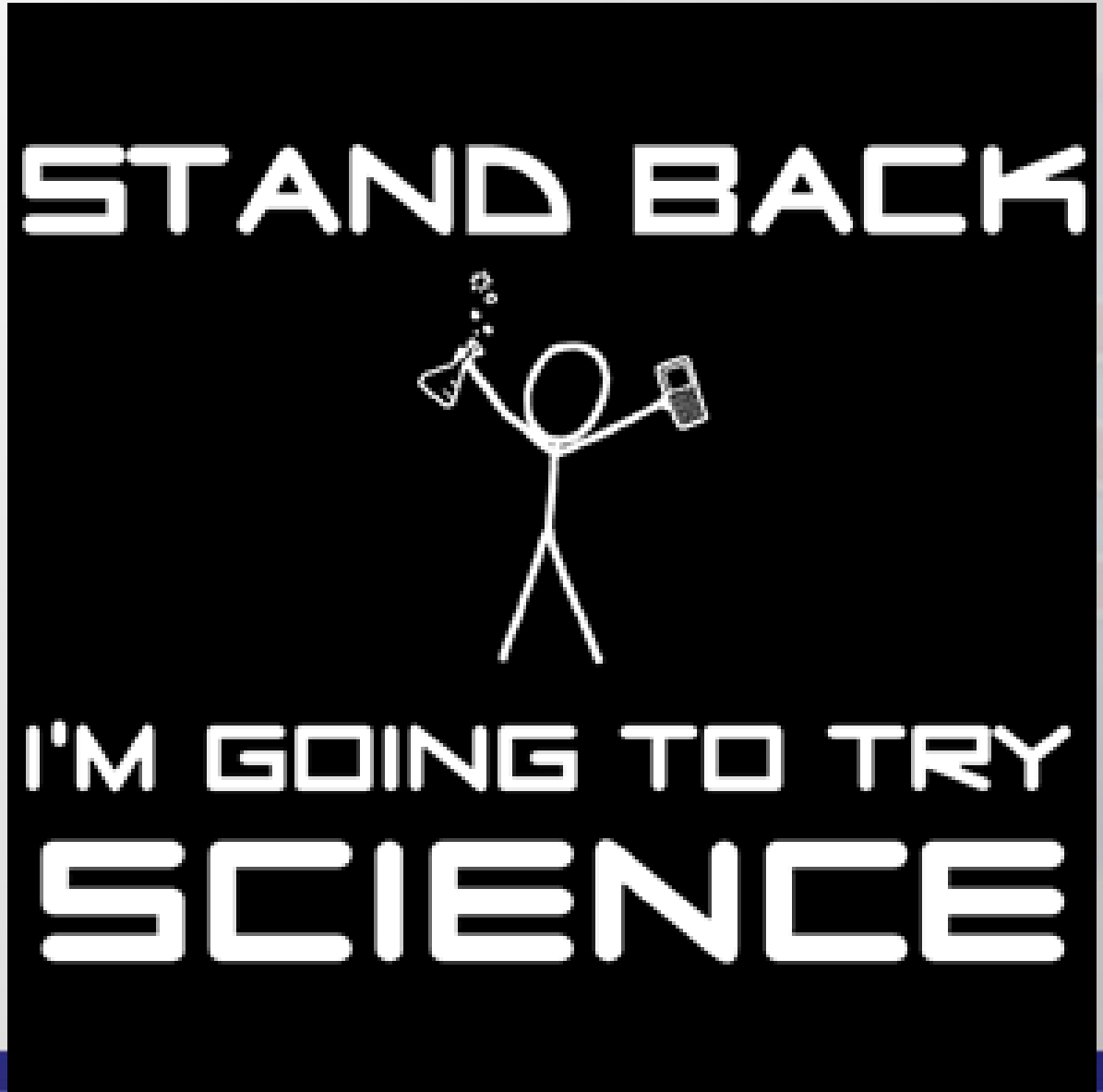
Prvi koraci u RE-u

- Korak 3: Praksa
  - CTF-ovi (www.ctftime.org)
    - Pokreni svoj tim!
  - Wargames (OverTheWire, SmashTheStack)
  - CrackMe's (www.crackmes.de, www.crackmes.us)
  - Analiza ranjivosti (1-day ili patch diffing)

Prvi koraci u RE-u

- Korak 4: Osobni projekt
  - Razvoj alata
  - Pisanje bloga/članaka
  - Traženje ranjivosti
  - Threat intelligence (malware hunting)

RE i znanost



RE i znanost

- Iznimno aktivno područje
- Analiza programskog koda kompleksno i znanstveno izazovno područje
- Sva veća sveučilišta imaju jake security/RE grupe:
  - Berkeley, CMU, UCSB, ETH, TU Wien (iseclab)...
- Istraživanja se mogu svrstati u 2 grupe:
  - Zaštita programskog koda
  - Analiza i zaključivanje o programskom kodu

RE i znanost

- Zaštita programskog koda
- Problem:
  - Kako otežati analizu koda i zaključivanje o funkcionalnosti programa
  - „On the (Im)possibility of Obfuscating Programs”
- Rješenja:
  - CFG flattening obfuskacije
  - VM translacija
  - White box cryptography

RE i znanost

- Analiza i zaključivanje o programskom kodu
- Formalne metode
  - Symbolic execution (sympy, smiasm)
  - Abstract interpretation (Bitblaze, BAP)
  - Theorem provers (SMT, SAT)
  - Korištenje tehnika kompajlera (Optimice, Metasm)
    - Optimizacija, translacija u IR
  - Tehnike raspoznavanja uzoraka i klasifikacije
  - Teorija grafova (Bindiff, TurboDiff, PatchDiff2)
    - CFG analiza

RE i znanost

- SAT/SMT
  - Using SAT and SMT to defeat simple hashing algorithms
  - Semi-Automated Input Crafting by Symbolic Execution, with an Application to Automatic Key Generator Generation
  - Finding Bugs in VMs with a Theorem Prover
  - SMT Solvers for Software Security
- Tools:
  - Z3, Yices, CryptoMiniSat

SMT primjer #1 – Code auditing

- 3 ujutro, brain-dead, oči se magle

```
char x,y,z;
...
if ((x + y < 10 || x > 9) && (y + z < 5))
    RANJIVOST();     // :D
```

SMT primjer #1 – Code auditing

- Zabrinjavajuće trivijalno

```
from z3 import *

a = BitVec('a', 8)
b = BitVec('b', 8)
c = BitVec('c', 8)

solve( (a + b < 10 or a > 9) and ((b + c) < 5) )
```

SMT primjer #2 – Kripto trivia

```
XOR_TABLE = [230374058,434659955,1700721058,3536163043]

def do_hash(a,b,c,d):
    nn = 0

    for i, n in enumerate((a, b, c, d)):
        nn ^= XOR_TABLE[i] ^ n

    return nn

o = 0xdeadbeef
x = 0xdeafc0de
y = 0xcafebabe
z = 0xd00dd00d

csum = do_hash(o, x, y, z)
if csum == 0xb9392d1a:
    print ”Program cracked!”
```

SMT primjer #2 – Kripto trivia

```
from z3 import *
XOR_TABLE = [230374058,434659955,1700721058,3536163043]

def hash(a, b, c, d):
    nn = BitVecVal(0, 32)
    for i, n in enumerate((a, b, c, d)):
        nn ^= XOR_TABLE[i] ^ n
    return nn

s = Solver()
a = BitVec('a', 32)
b = BitVec('b', 32)
c = BitVec('c', 32)
d = BitVec('d', 32)

csum2 = hash(a, b, c, d)
solve(csum2 == 0xb9392d1a, a>0, b>0, c>0, d>0)
```

The End

- Hvala, brankospa@gmail.com

- Pitanja?