Dissecting Malware 101

Andrea Continella

acontinella@iseclab.org

https://github.com/conand/dissecting-malware-101

Malware???

- Malicious Software intentionally written to **violate** one or more **security policy**
- Different categories:
    - Virus: Infect hosts and files reproducing itself
    - Trojans: mislead users of its true intent
    - Ransomware: encrypt victim's files and ask for a ransom

How to analyze it?

Static Analysis

Dynamic Analysis

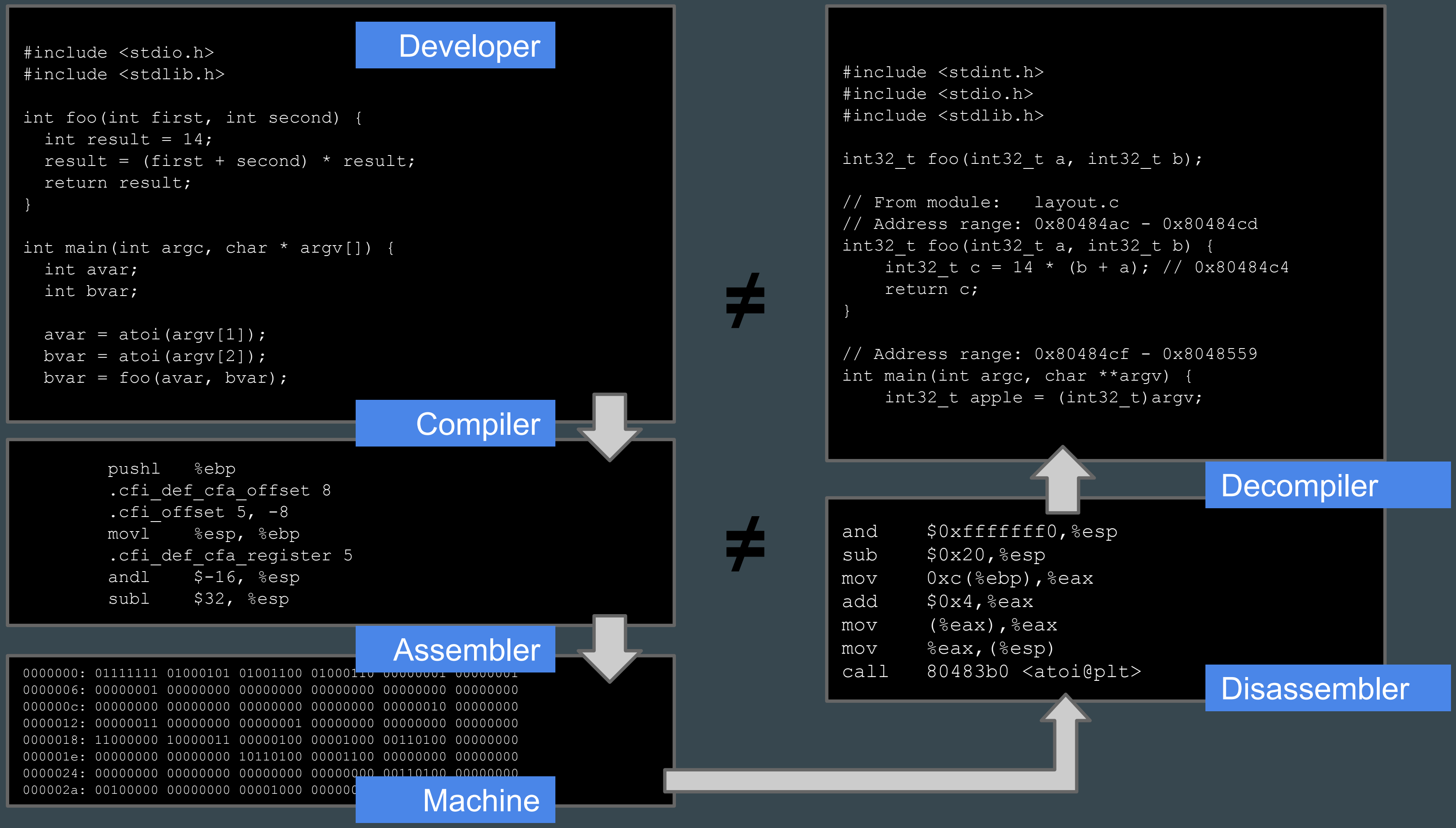
Static Analysis

- Understand the functionalities of a binary looking at its code.
- Disassemble instructions

Static Analysis

- Understand the functionalities of a binary looking at its code.
- Disassemble instructions

⚠ Malware's code is often encrypted or obfuscated



Developer

```
#include <stdio.h>
#include <stdlib.h>

int foo(int first, int second) {
  int result = 14;
  result = (first + second) * result;
  return result;
}

int main(int argc, char * argv[]) {
  int avar;
  int bvar;

  avar = atoi(argv[1]);
  bvar = atoi(argv[2]);
  bvar = foo(avar, bvar);
```

Compiler

```
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        andl    $-16, %esp
        subl    $32, %esp
```

Assembler

```
0000000: 01111111 01000101 01001100 01000110 00000001 00000001
0000006: 00000001 00000000 00000000 00000000 00000000 00000000
000000c: 00000000 00000000 00000000 00000000 00000010 00000000
0000012: 00000011 00000000 00000001 00000000 00000000 00000000
0000018: 11000000 10000011 00000100 00001000 00110100 00000000
000001e: 00000000 00000000 10110100 00001100 00000000 00000000
0000024: 00000000 00000000 00000000 00000000 00110100 00000000
000002a: 00100000 00000000 00001000 000000
```

Machine

Disassembler

```
and     $0xfffffff0,%esp
sub     $0x20,%esp
mov     0xc(%ebp),%eax
add     $0x4,%eax
mov     (%eax),%eax
mov     %eax,(%esp)
call    80483b0 <atoi@plt>
```

Decompiler

```
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>

int32_t foo(int32_t a, int32_t b);

// From module:   layout.c
// Address range: 0x80484ac - 0x80484cd
int32_t foo(int32_t a, int32_t b) {
    int32_t c = 14 * (b + a); // 0x80484c4
    return c;
}

// Address range: 0x80484cf - 0x8048559
int main(int argc, char **argv) {
    int32_t apple = (int32_t)argv;
```

≠

≠

Static Analysis

- Understand the functionalities of a binary looking at its code.
- Disassemble instructions

⚠ Malware’s code is often encrypted or obfuscated

Dynamic Analysis

- Execute the binary in a controlled environment and monitor its activity
- Look at interactions with the environment

Static Analysis

- Understand the functionalities of a binary looking at its code.
- Disassemble instructions

⚠ Malware's code is often encrypted or obfuscated

Dynamic Analysis

- Execute the binary in a controlled environment and monitor its activity
- Look at interactions with the environment

⚠ Evasive Malware can recognize analysis environment and hide its malicious behavior

Can we analyze one?

What if we create one…
…and then analyze it? :-)

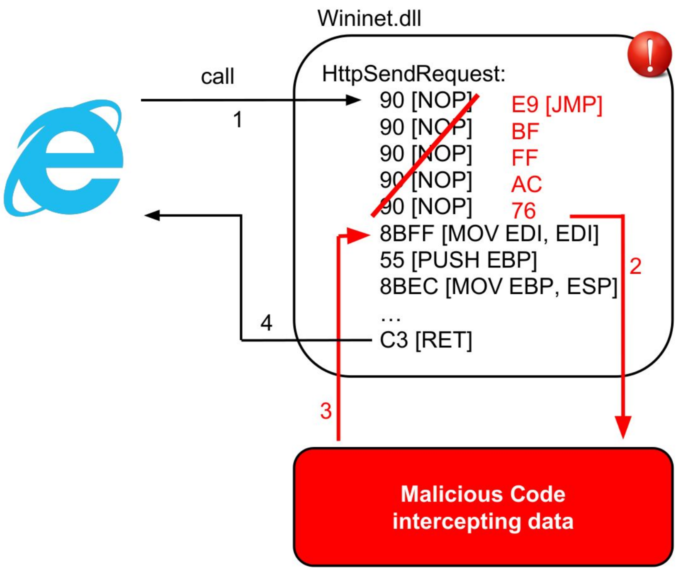
Who is ZeuS?

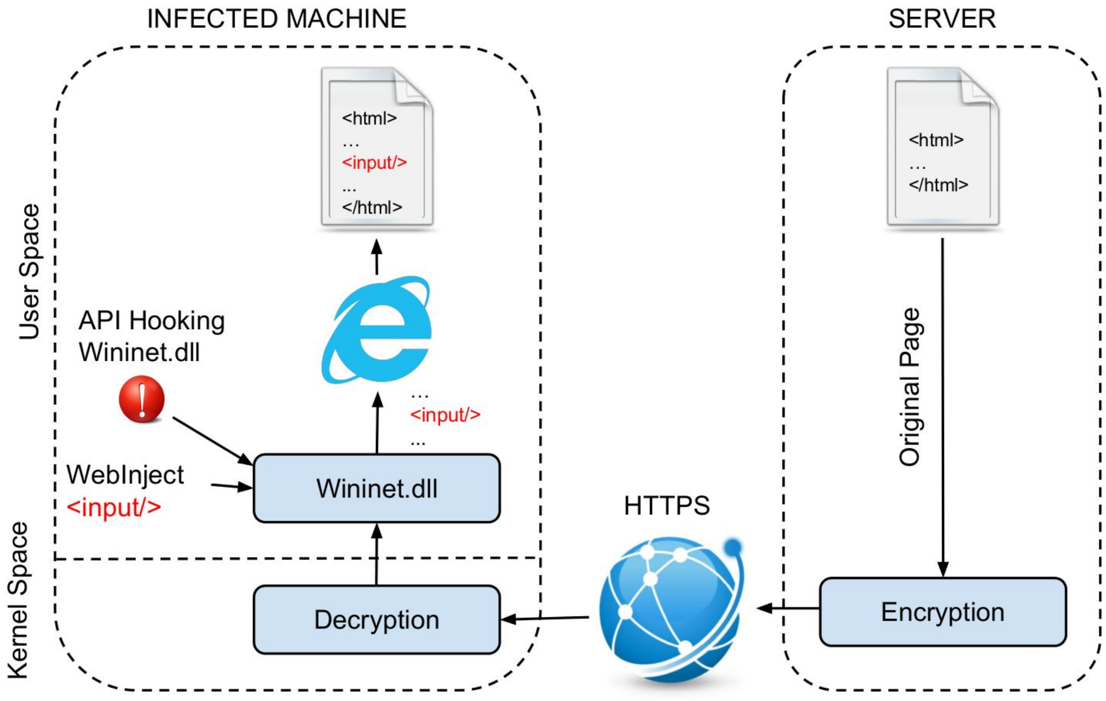
- One of the most famous “**Banking Trojans**”
- Perform “**Man in the Browser**” attacks to **steal** credentials and perform financial frauds
- Steal info submitted to web-forms
- Keylogger
- Record screenshots
- **Botnet** architecture

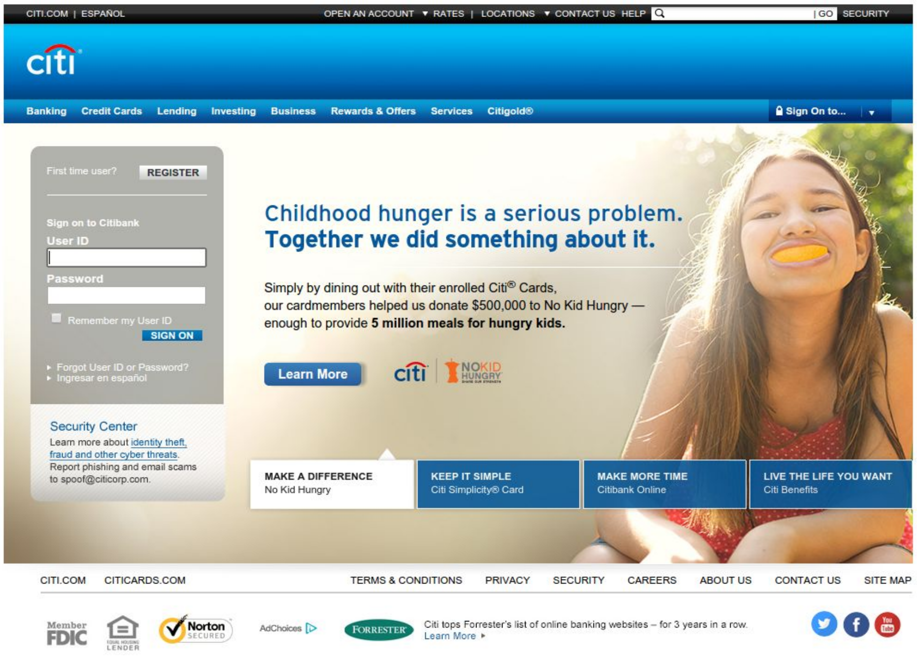
Leaked sources ~> https://github.com/Visgean/Zeus

How it works

- **API hooking:** intercept data flowing into the browser, even when the connection is encrypted (HTTPS)!
- **WebInject:** manipulate and modify web-pages locally
- **Goal:** modify web-pages to add further fields in forms and steal further information


Wininet.dll
call
1
HttpSendRequest:
~~90 [NOP]~~ E9 [JMP]
~~90 [NOP]~~ BF
~~90 [NOP]~~ FF
~~90 [NOP]~~ AC
~~90 [NOP]~~ 76
8BFF [MOV EDI, EDI]
55 [PUSH EBP]
8BEC [MOV EBP, ESP]
…
C3 [RET]
2
3
4
Malicious Code
intercepting data


INFECTED MACHINE
SERVER
User Space
Kernel Space
<html>
…
<input/>
...
</html>
API Hooking
Wininet.dll
…
<input/>
...
WebInject
<input/>
Wininet.dll
Decryption
HTTPS
<html>
…
</html>
Original Page
Encryption



CITI.COM | ESPAÑOL OPEN AN ACCOUNT ▾ RATES | LOCATIONS ▾ CONTACT US HELP | GO SECURITY

Banking Credit Cards Lending Investing Business Rewards & Offers Services Citigold®

Sign On to... ▾

First time user? REGISTER

Sign on to Citibank

User ID

Password

□ Remember my User ID

SIGN ON

- Forgot User ID or Password?
- Ingresar en español

Security Center

Learn more about identity theft, fraud and other cyber threats.

Report phishing and email scams to spoof@citicorp.com.

Childhood hunger is a serious problem.

Together we did something about it.

Simply by dining out with their enrolled Citi® Cards, our cardmembers helped us donate $500,000 to No Kid Hungry — enough to provide **5 million meals for hungry kids.**

Learn More

MAKE A DIFFERENCE No Kid Hungry

KEEP IT SIMPLE Citi Simplicity® Card

MAKE MORE TIME Citibank Online

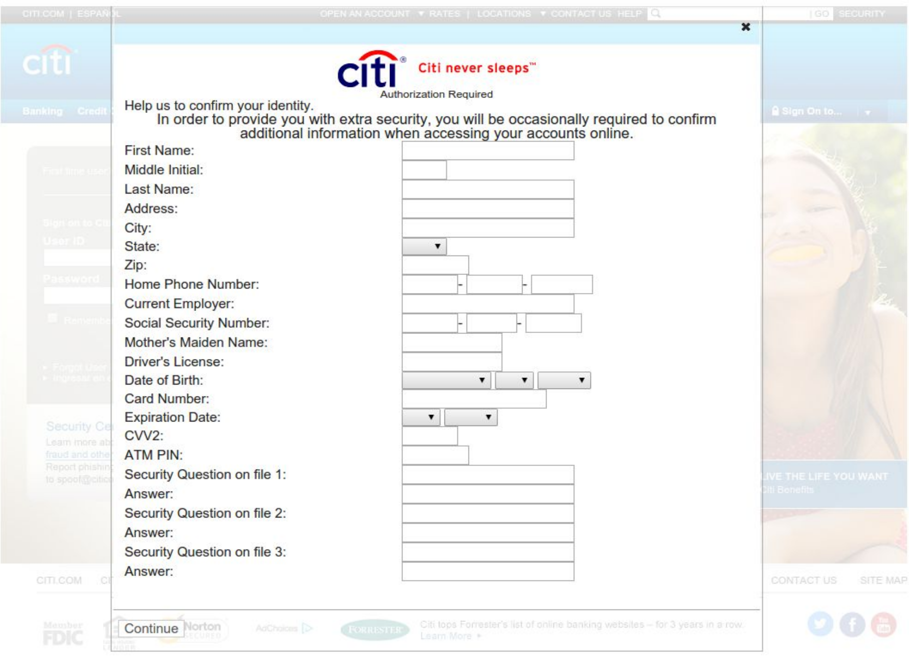
LIVE THE LIFE YOU WANT Citi Benefits

CITI.COM CITICARDS.COM TERMS & CONDITIONS PRIVACY SECURITY CAREERS ABOUT US CONTACT US SITE MAP

Member FDIC AdChoices

Citi tops Forrester's list of online banking websites – for 3 years in a row.

Learn More ▸



Citi never sleeps™

Authorization Required

Help us to confirm your identity.

In order to provide you with extra security, you will be occasionally required to confirm additional information when accessing your accounts online.

First Name:

Middle Initial:

Last Name:

Address:

City:

State: ▾

Zip:

Home Phone Number: - -

Current Employer:

Social Security Number: - -

Mother's Maiden Name:

Driver's License:

Date of Birth: ▾ ▾ ▾

Card Number:

Expiration Date: ▾ ▾

CVV2:

ATM PIN:

Security Question on file 1:

Answer:

Security Question on file 2:

Answer:

Security Question on file 3:

Answer:

Continue

Hands-on: Build a ZeuS sample!

Static Analysis Tools

- file
- readelf
- strings
- Disassembler: objdump, binary ninja, Radare2, IDA
- Decompiler: IDA

Hands-on: Static Analysis

Static Analysis

- file, strings
- Disassemble `bot.exe`
  - `objdump`
  - https://binary.ninja/
  - https://www.hex-rays.com/products/ida/
- Look for code injection techniques:
  - `CreateRemoteThread?`
- More on code injection: https://github.com/peperunas/injectopi

Dynamic Analysis Tools

- strace, ltrace
- debuggers: gdb, OllyDbg, WinDbg...
- emulators: QEMU
- sandboxes: Cuckoo http://www.cuckoosandbox.org/

Hands-on: Dynamic Analysis

Dynamic Analysis

- Install and set-up cuckoo: http://www.cuckoosandbox.org/
- Analyze `bot.exe`
- Read cuckoo's report
- Dump the memory
- Inspect the memory dump
  - Install and use volatility: https://github.com/volatilityfoundation/volatility
  - Have a look at Yara: http://virustotal.github.io/yara/
- Analyze Network Traffic

Task: Custom Analysis

Automate the extraction of the WebInject targets given a sample

1. Execute the sample
2. Open the browser
   - Interesting info are allocated into the browser's address space!
3. Dump the memory
4. Look for interesting stuff! ;-)

Too Simple?

Task: Analyze `fun.exe`

More Stuff

- https://github.com/necst/arancino
- https://github.com/rshipp/awesome-malware-analysis
- https://github.com/CheckPointSW/InviZzzible
- http://www.kernelmode.info/forum/viewtopic.php?f=11&t=3478
- https://github.com/AlicanAkyol/sems
- https://github.com/angr/angr

Analysis Completed!

•••

Andrea Continella

🐦 @_conand

https://conand.me

acontinella@iseclab.org