Writing More Secure Code

Ynon Perek

www.tocode.co.il
ynon@tocode.co.il

Agenda

- IT Security Today
- Secure C/C++
- Secure Networking
- Secure Password Storage
- Secure Cryptography
- Secure Web & Mobile Apps

Security:

Why Bother?

Reasons for Security

- Security of a system = Security of the weakest part

Examples

- For each of the following examples see if you can find:
  - **Who** could have acted differently to avoid the damage?
  - **What** could have they done differently?

DECEMBER 5, 2017

Popular third-party keyboard AI.type leaks personal data of over 31 million users

Chance Miller - Dec. 5th 2017 3:43 pm PT @ChanceHMiller

רשלנות משמעותית במשרד הפנים חושפת את פרטי האזרחים המוציאים תיעוד ביומטרי

ינואר 8, 2018 תגיות : אבטחת מידע חדשות אינטרנט בקטגוריה: חדשות אינטרנט

המאמר נכתב על ידי רן בר-זיק, מתכנת מנוסה וכותב כל המאמרים באתר אינטרנט ישראל.

`event-stream` dependency attack steals wallets from users of copay #9346

Closed **deanveloper** opened this issue on Nov 26, 2018 · 29 comments

האקר "נעל" מסמכים של עיריית נצרת עילית ודורש כופר

פרסום ראשון: גורם בכיר בעירייה טוען כי "אם החומרים ייעלמו לנו, אנחנו נישאר בלי ידיים ובלי רגליים". בעירייה מתלבטים אם להיענות לבקשת הכופר של 10,000 שקלים שביקש ההאקר, או להסתכן באיבוד הקבצים החיוניים לתפקוד העיר

יאיר קראוס וינון בן שושן | 17:36 28/11/2016

LOUISE MATSAKIS AND ISSIE LAPOWSKY SECURITY 09.28.18 03:03 PM

EVERYTHING WE KNOW ABOUT FACEBOOK'S MASSIVE SECURITY BREACH

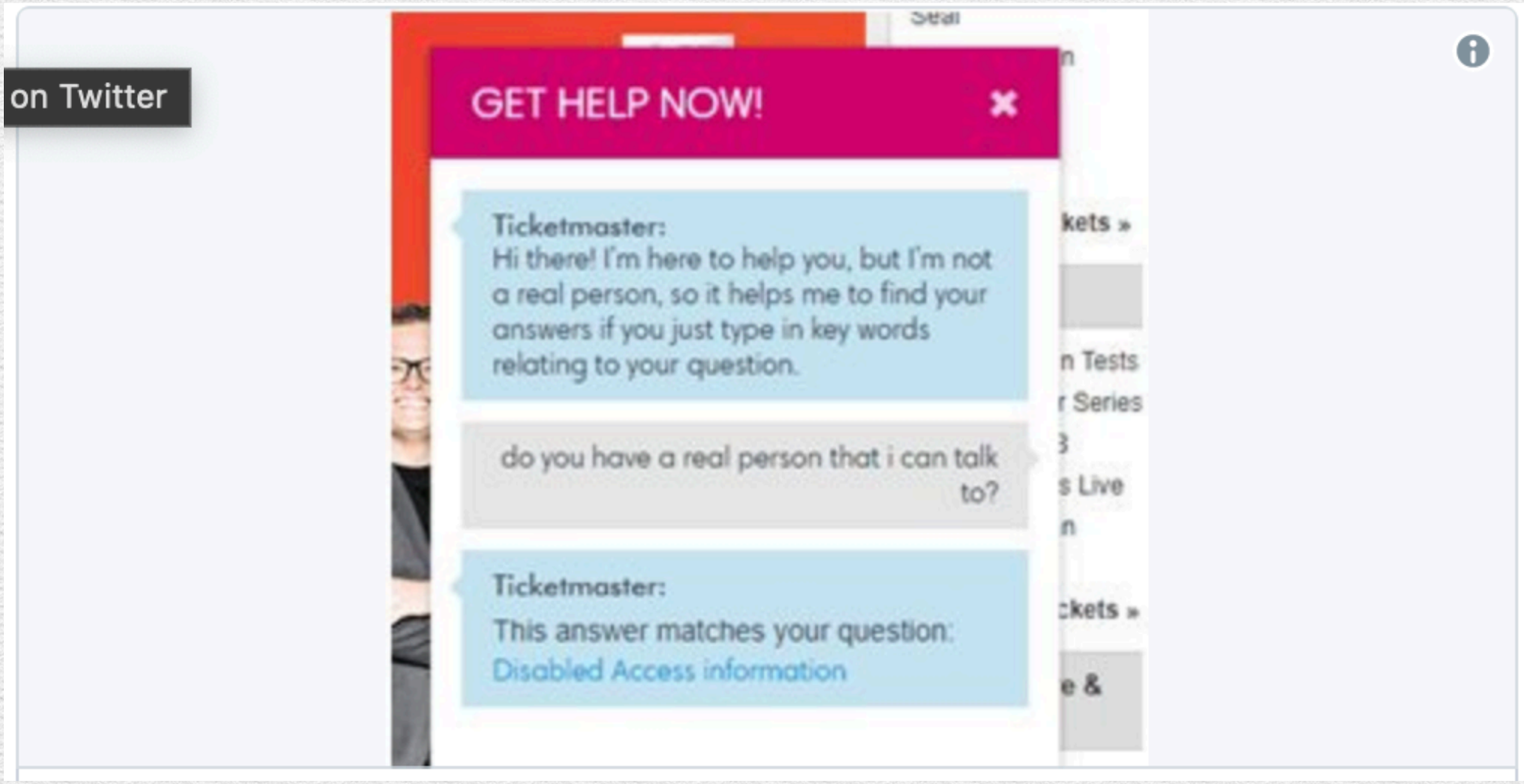
https://www.wired.com/story/facebook-security-breach-50-million-accounts/

INSIDER NEWS

Raising awareness quickly: The eBay data breach

Here's a quick breakdown on the situation that can be shared in-house

https://www.csoonline.com/article/2157782/security-awareness-raising-awareness-quickly-the-ebay-database-compromise.html



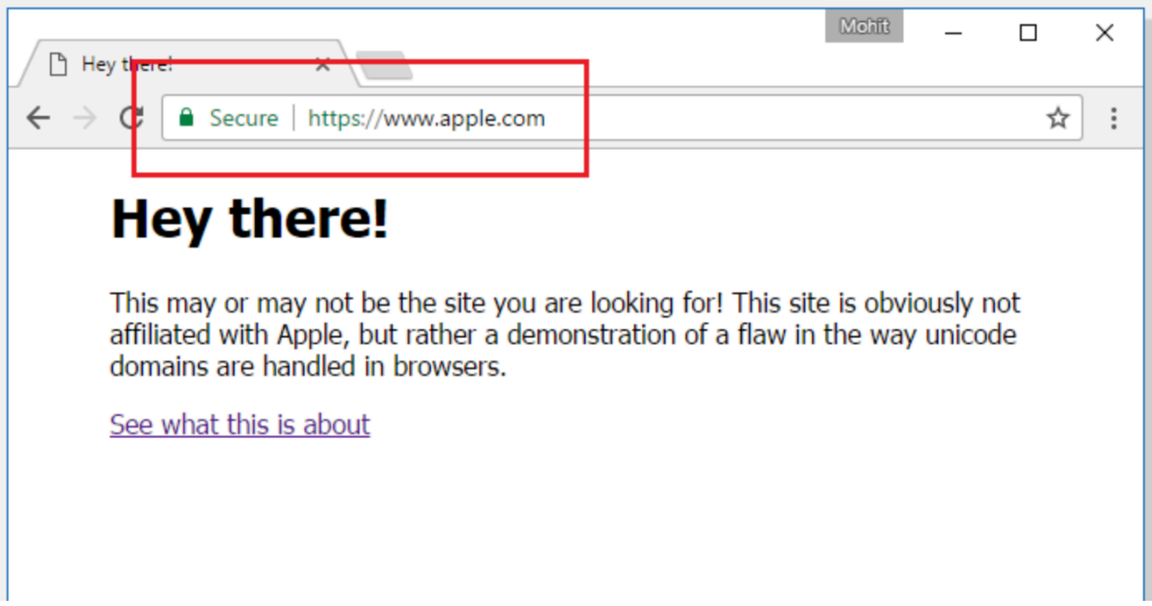
https://www.bankinfosecurity.com/ticketmaster-breach-traces-to-embedded-chatbot-software-a-11144

שניות בודדות של עבודה היו מונעות דליפת מידע רגיש

20,000 חשבוניות עם פרטי לקוחות דלפו משרת הרץ ישראל. כלי הפריצה: דפדפן וחיפוש בגוגל; לזכותה של החברה ייאמר שלאחר הפנייה לדובר, העניין טופל במהירות ובמקצועיות

[This Phishing Attack is Almost Impossible to Detect On Chrome, Firefox and Opera]

Monday, April 17, 2017 Mohit Kumar



CVE-2018-15981

Flash Player versions 31.0.0.148 and earlier have a type confusion vulnerability. Successful exploitation could lead to arbitrary code execution.

CVE-2019-0565 Detail

Current Description

A remote code execution vulnerability exists when Microsoft Edge improperly accesses objects in memory, aka "Microsoft Edge Memory Corruption Vulnerability." This affects Microsoft Edge.

Cellcom 21:56 58%

Profile

Follow

סטטוסים מצייצים @StatusHunter 9m

Hack by palestine :$:)

Othman ' abad ' yazan

Fuck isral fb.me/472NuZB2w

סטטוסים מצייצים @StatusHunter 25m

ממהרים ין ישראל תעשה גשם טילים

Ĺĺĺĺĺĺĺĺĺĺĺĺĺ

Aedm האבהי

آن آمرنآ نفعل سرآئيل سوف تمطر صوآريخ

ههههههههههههه

القآم آعضم

7 10

Timelines Notifications Messages Me

20:30

لحيدوش مرسم התרופות של כללית בעלות
[redacted] של 2.90 ש"ח

SMS

PayPal

Notice of Policy Updates

Dear Customer,

Some information on your account appears to be missing or incorrect.

Your account will be suspended within 27 days ,update your account as soon as possible.

Please update your information promptly so that you can continue

to enjoy all the benefits of your PayPal account.

Update

If you need help logging in, go to our Help Center by clicking the Help link located in the upper right-hand corner of any PayPal page..

Sincerely,

PayPal

Please do not reply to this email. We are unable to respond to inquiries sent to this address. For immediate answers to your questions, visit our Help Center by clicking "Help" at the top of any PayPal page.

כניסה ללקוחות רשומים

עדיין לא נרשמתם לאזור האישי? הירשמו עכשיו

שם משתמש ?

סיסמה ?

שחזור פרטי משתמש

כניסה לאזור האישי

Cellcom 13:46 71%

1 +43 1 962325699301
last seen today at 13:34

ברוך הבא המשתמש היקר, לאחר מבט על יומני השרת שלנו מצאנו שהחשבון שלך דיווח על זה מאדם לא ידוע שטוען שזה החשבון שלו, אבל שלחנו קוד אימות למספר שלך וודא שאתה מחדש הבעלים האמיתי של החשבון, אנא שלח את הקוד לשיחה זו.
כדי לאבטח את חשבונך ולאשר זאת, פרט לכך שיינעל את חשבונך לאחר פרק זמן של 24 שעות כדי לבצע את ההליכים הדרושים כדי להשיג את תנאי השימוש. ותודה לך

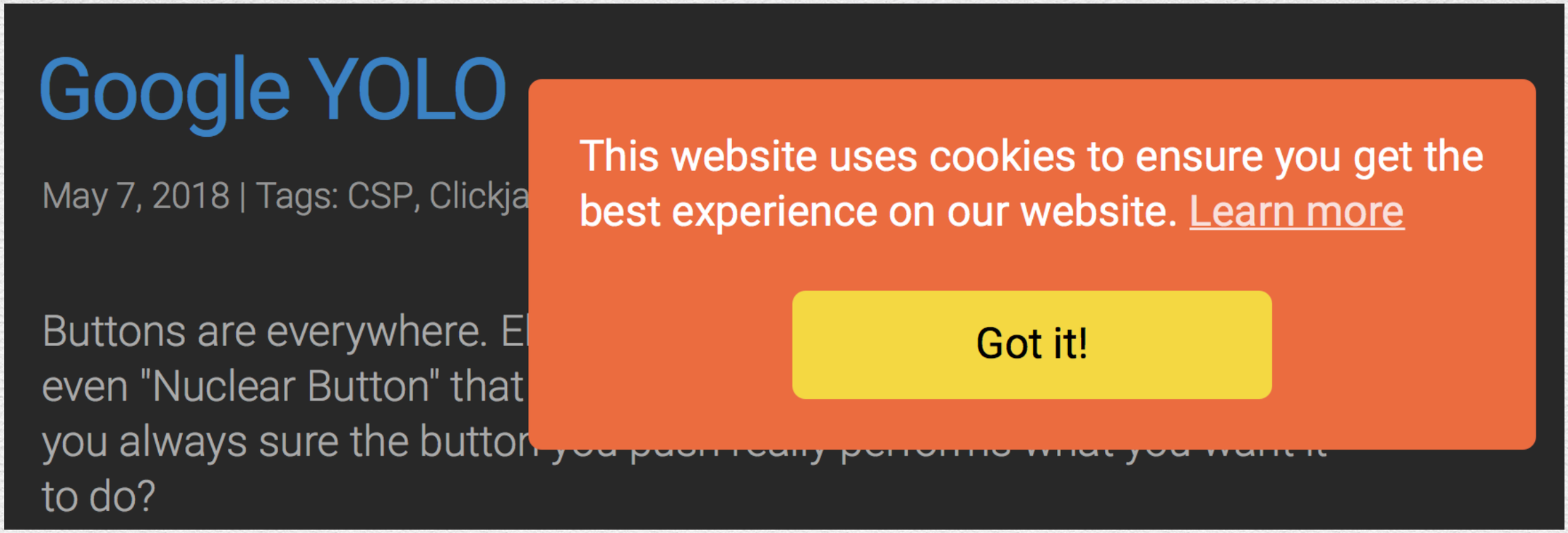
צוות תמיכה WhatsApp

13:36

עליך לשלוח את הקוד שקיבלת באמצעות הודעת טקסט ואם לא תגיב, חשבונך יקפא

13:44

This sender is not in your contacts



An SEO Expert Has Shown How Chrome's Back Button can be Hijacked to Spy on Users

https://www.privateinternetaccess.com/blog/2018/09/an-seo-expert-has-shown-how-chromes-back-button-can-be-hijacked-to-spy-on-users/

Backdoor Account Found in D-Link DIR-620 Routers

By **Catalin Cimpanu**

May 23, 2018 09:11 AM 2

Cisco fixes hard-coded password 'backdoor' flaw in Wi-Fi access points

Two of the vulnerabilities were considered 'critical," allowing an attacker to take complete over an affected device.

By Zack Whittaker for Zero Day | January 14, 2016 -- 14:31 GMT (14:31 GMT) | Topic: Security

Hardcoded Password Found in Cisco Enterprise Software, Again

By **Catalin Cimpanu**

May 17, 2018 12:50 AM 2

מטריד: מאות מצלמות אבטחה הפכו לכלי הנשלט מרחוק

הילה חיימוביץ' | לפני 3 שעות 28 דקות 0 אבטחת מידע

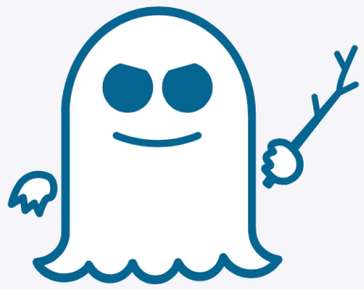
חברת האבטחה הישראלית Incapsula פרסמה דו"ח לפיו מאות מצלמות אבטחה הפכו לרשת גדולה הנשלטת מרחוק ובקלות. האם הן באמת מגינות עלינו או אולי דווקא להפך?

The Dyn report: What we know so far about the world's biggest DDoS attack

The Internet of Things has been proven to be just as dangerous as we feared, with an assault from tens of millions of internet addresses.

By Steven J. Vaughan-Nichols | October 25, 2016 -- 15:28 GMT (16:28 BST) | Topic: Networking

Meltdown



Spectre

Microsoft Security Bulletin MS10-070 - Important

Vulnerability in ASP.NET Could Allow Information Disclosure (2418042)

Published: September 28, 2010 | Updated: October 26, 2011

NEWS

465,000 Abbott pacemakers vulnerable to hacking, need a firmware fix

The FDA and Homeland Security issued alerts about vulnerabilities in Abbott (formerly St. Jude Medical) pacemakers and a firmware update to close those security holes.

'Panty Buster' Toy Left Private Sex Lives Of 50,000 Exposed

Thomas Brewster Forbes Staff

Security

I cover crime, privacy and security in digital and physical forms.

קפטן אינטרנט ברשת

פרצת אבטחה חמורה חשפה פרטי מזמיני כרטיסים בבארבי, בצוותא ובמועדונים אחרים

חוקרי אבטחה גילו כי מערכת ניהול ההזמנות של חברת Eventgo לא היתה מוגנת בסיסמה, וכל גולש יכול היה לגשת אליה; החברה טוענת: לא דלף מידע אישי

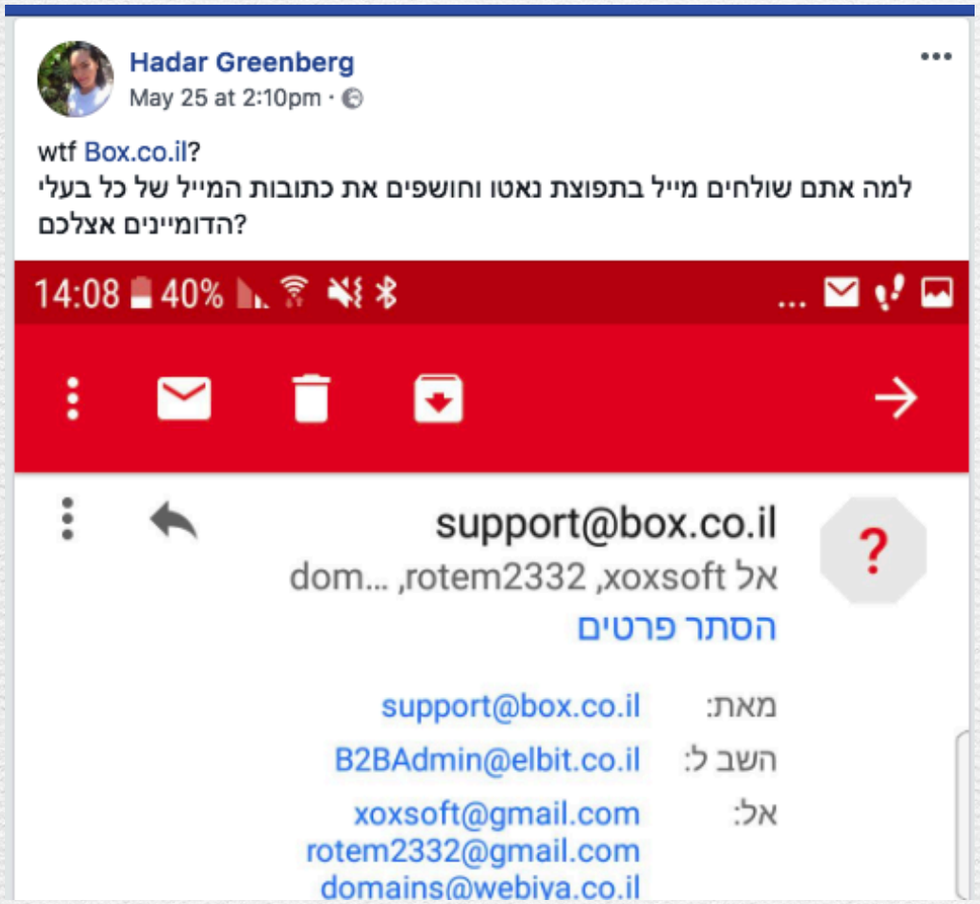
עודד ירון |
15:16 18.01.2015

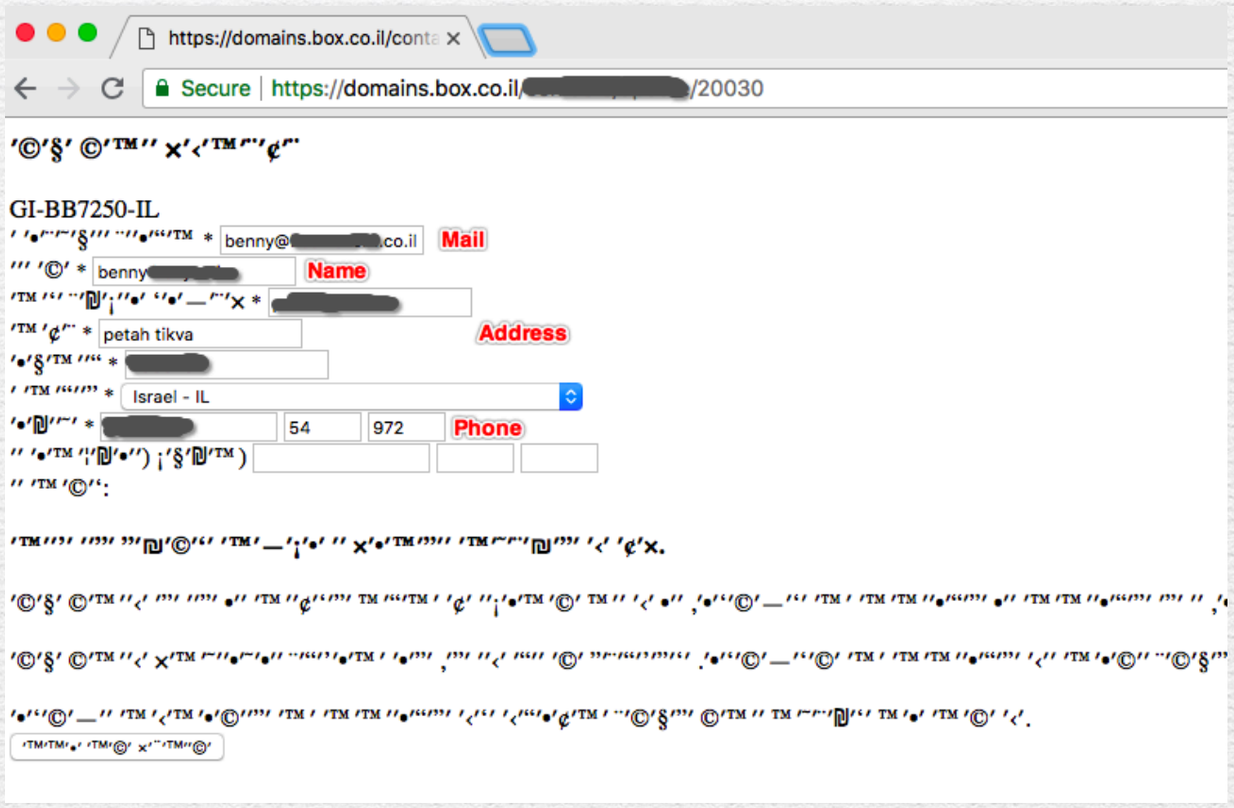
המשטרה חושדת: מאגר המידע של בתי המשפט נפרץ

אלה לוי-וינריב
16:06 ,10/09/2012

נושאים למעקב >>

הותר לפרסום: משטרת ישראל חשפה פרשת פריצה למאגרי המידע של בתי המשפט, ופרסום מסמכים חסויים. החשוד המרכזי בפרשה משה הלוי, בן 42 מעכו, נעצר לאחרונה בחשד כי פרץ למערכת בתי המשפט וחשף חומרים סודיים ורגישים. החקירה הסמויה בפרשה נפתחה ב-28.6, בעקבות תלונה שהגישה





https://internet-israel.com/%D7%97%D7%93%D7%A9%D7%95%D7%AA-%D7%90%D7%99%D7%A0%D7%98%D7%A8%D7%A0%D7%98/%D7%9E%D7%90%D7%92%D7%A8-%D7%94%D7%9C%D7%A7%D7%95%D7%97%D7%95%D7%AA-%D7%A9%D7%9C-%D7%91%D7%95%D7%A7%D7%A1-/%D7%94%D7%99%D7%94-%D7%92%D7%9C%D7%95%D7%99-%D7%9C%D7%97%D7%9C%D7%95%D7%98%D7%99%D7%9F

Let’s Recap

- Operations
- Human Factor (user)
- Past Mines
- Impersonation
- Broken Tools
- You’re Using It Wrong

Attack Ingredients

Buggy technology

Human Mistakes

Surprise

What To Expect

- If someone can use it -> they'll misuse it
- It's not always in our hands
- Best Attacks: technical + human factor

Why Is It Hard ?

- Secure code problems:
  - Lack of knowledge
  - Carelessness

Security Concepts

- Threats
- Security Policy
- Security Flaw
- Vulnerability
- Exploit

Security Threats

- Denial of Service
- Gain information
- Change data

Security Policy

A set of rules and practices that specify how a system protects sensitive resources

Security Flaw

A software defect that poses a potential security risk

Vulnerability

A set of conditions that allows an attacker to violate a security policy

https://web.nvd.nist.gov/view/vuln/search

Exploit

A technique for using a security vulnerability to violate the security policy

Our Goal

- Reduce security flaws
- Don’t wait for an exploitable vulnerability

Myth-Busting

- Myth:
  Attacker can’t see my source code

Myth-Busting

- Myth:
  Attacker can't see my source code

- Reality:
  Source code leaks

Myth-Busting

- Myth:
  Some flaws are un-exploitable

Myth-Busting

- Myth:
  Some flaws are un-exploitable

- Reality:
  Nobody found the exploit yet

Myth-Busting

- Myth:
  Who’d want to hack into my system ?

Myth-Busting

- Myth:
  Who'd want to hack into my system ?
- Reality:
  Anyone from script kidz, through competitors and organized crime.

Q & A

Types Of Vulnerabilities

Human Factor

Technical

Human Factor

My $50,000 Twitter Username Was Stolen Thanks to PayPal and GoDaddy

I had a rare Twitter username, @N. Yep, just one letter. I've been offered as much as $50,000 for it. People have tried to steal it. Password reset instructions are a regular sight in my email inbox. As of today, I no longer control @N. I was extorted into giving it up.

Application Level Vulnerabilities

http://www.youtube.com/watch?v=39urRvFFZH0

Application Level Vulnerabilities

- In multi player games, a player can send malicious data to other players

- Cause vulnerable apps to misbehave

Platform Level Vulnerability

- In multi player games, a player can send malicious data to other players

- Only now - everyone is vulnerable

Server Level Vulnerability

- Attacking the server usually has devastating consequences, for it has all the data

Famous Server Level Vulnerabilities

- (2002) guess.com revealed 200K customer names and credit cards
- (2007) Microsoft UK Defacement
- (2009) RockYou DB hacked for 30Mil users
- (2011) MySql.com hacked
- (2012) Yahoo lost 450K login credentials

Server-Platform Vulnerabilities

- Same as before, but now it’s not in your hands…

Server-Platform Vulnerabilities

Microsoft Security Bulletin MS10-070 – Important

Vulnerability in ASP.NET Could Allow Information Disclosure (2418042)

Published: Tuesday, September 28, 2010 | Updated: Wednesday, October 26, 2011

Version: 4.2

Windows Server 2008 for x64-based Systems	Microsoft .NET Framework 1.1 Service Pack 1** (KB2416447) Microsoft .NET Framework 3.5** (KB2418240) Microsoft .NET Framework 3.5 Service Pack 1** (KB2416473) Microsoft .NET Framework 4.0**[1] (KB2416472)	Information Disclosure	Important

Secure Design Takeaways

- When analysing a system's security, must take into account all aspects of the system
- Security of the system = Security of its weakest link
- Plan for security failures

Q & A

Implementation Risks

- Under Java/C# lies C/C++ chaos
- Vulnerabilities love chaos

Agenda

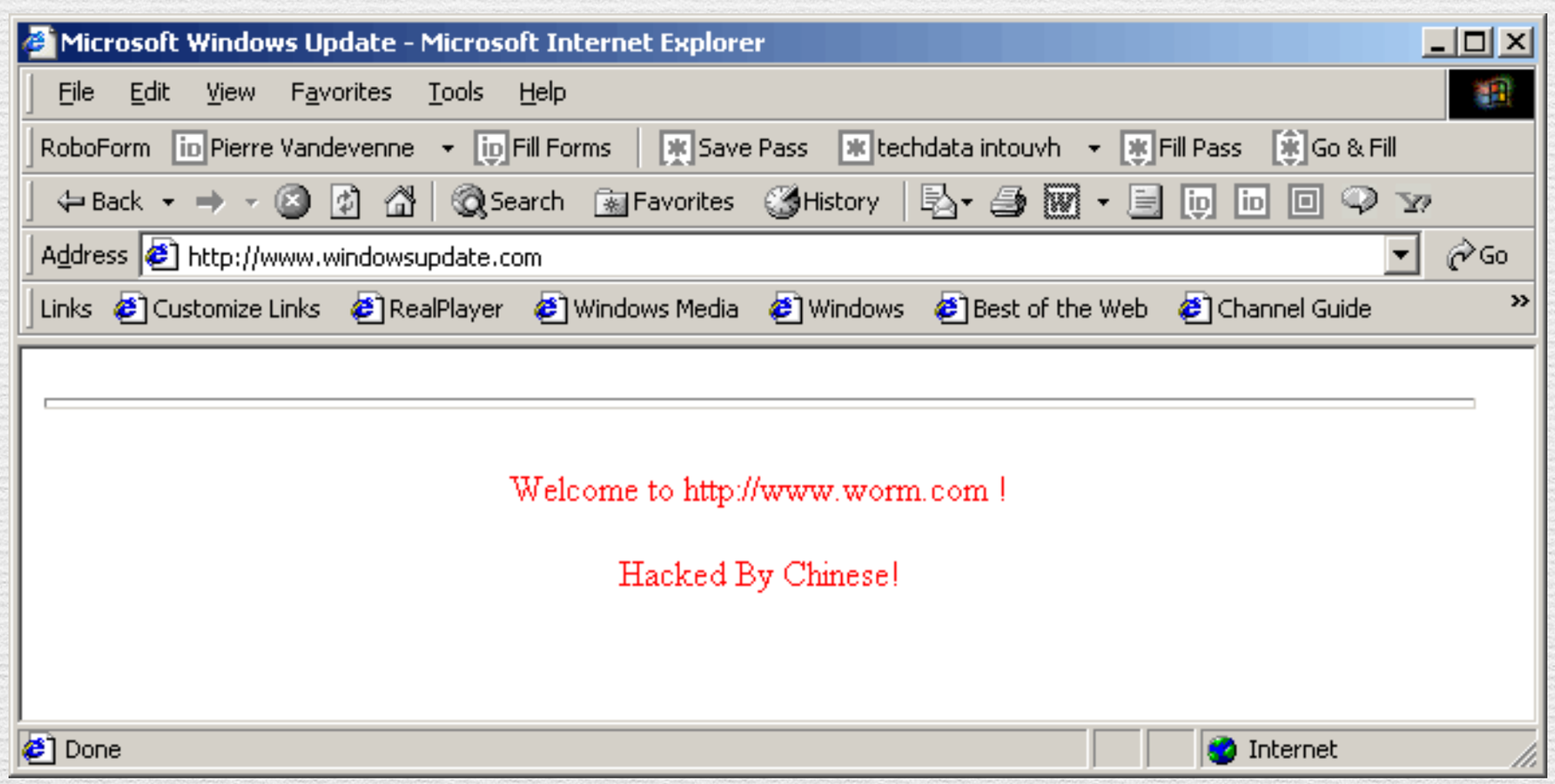
- Buffer Overflows
- Bad Arithmetics
- Format Strings
- Bitten By The Framework

Famous Overflows

- Born in 1988
- The Morris Worm
- Used a bug in finger

Famous Overflows

- 2001 Code Red Worm
- Bug in MS IIS

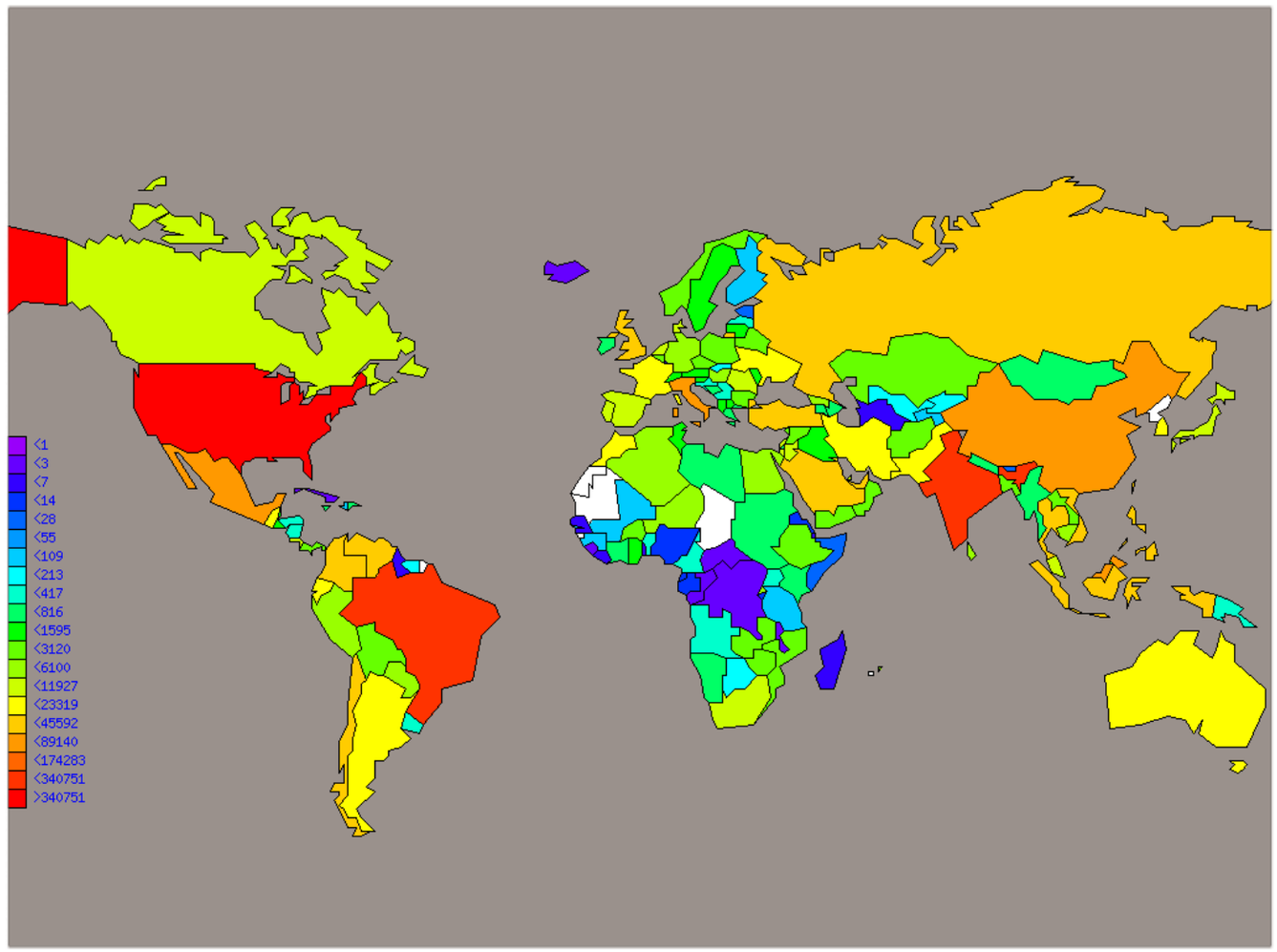


Famous Overflows

- 2004 Sasser worm
- Bug in Windows LSASS
- Created by Sven Jaschan

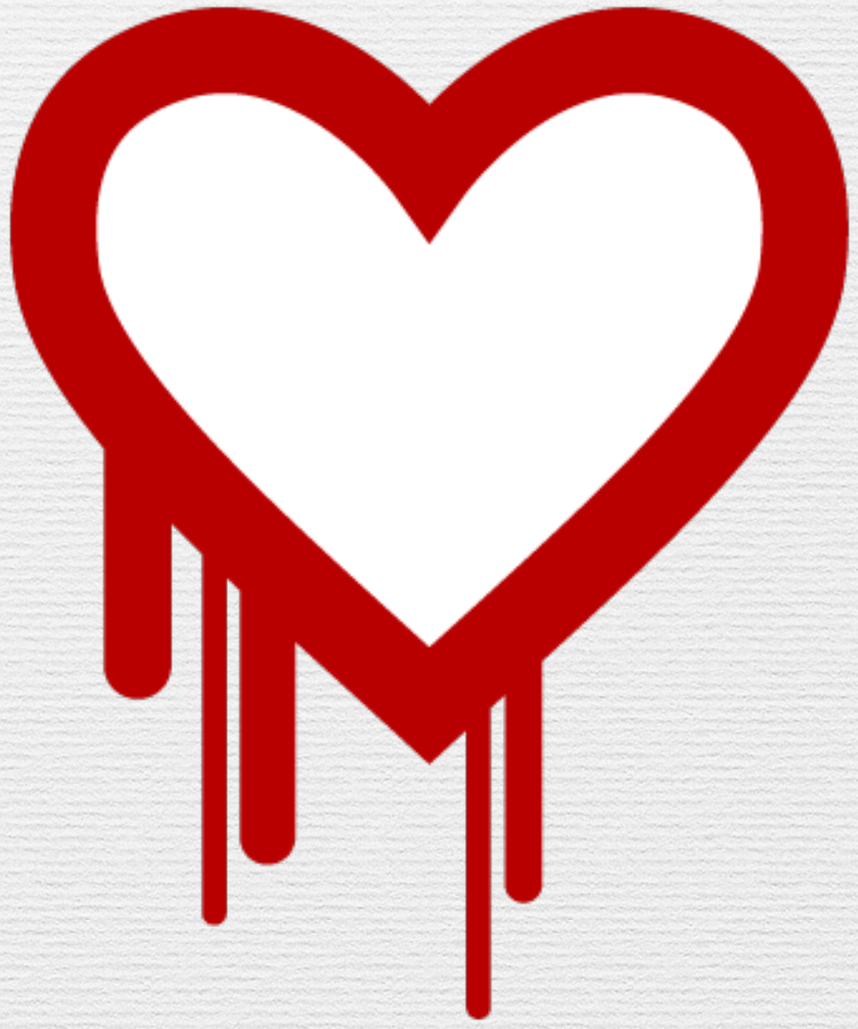
Famous Overflows

- (2008) Conficker
- Heat map as of May 2009
- Millions of computers, over 200 countries



Famous Overflow

- The Heartbleed Bug
- Missing bounds check in OpenSSL broke the internet (for 2 years)
- Xkcd: http://xkcd.com/1354



Heartbleed (t1_lib.c)

```
2412        /* Read type and payload length first */
2413        hbtype = *p++;
2414        n2s(p, payload);
2415        pl = p;
2416
2417        if (s->msg_callback)
2418                s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
2419                        &s->s3->rrec.data[0], s->s3->rrec.length,
2420                        s, s->msg_callback_arg);
2421
2422        if (hbtype == TLS1_HB_REQUEST)
2423                {
2424                unsigned char *buffer, *bp;
2425                int r;
2426
2427                /* Allocate memory for the response, size is 1 bytes
2428                 * message type, plus 2 bytes payload length, plus
2429                 * payload, plus padding
2430                 */
2431                buffer = OPENSSL_malloc(1 + 2 + payload + padding);
2432                bp = buffer;
2433
2434                /* Enter response type, length and copy payload */
2435                *bp++ = TLS1_HB_RESPONSE;
2436                s2n(payload, bp);
2437                memcpy(bp, pl, payload);
2438
2439                r = ssl3_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload + padding);
```

Other Overflows

- (CVE-2016-1541) Heap-based buffer overflow in libarchive before 3.2.0 allows remote attackers to execute arbitrary code via crafted entry-size values in a ZIP archive.

- (CVE-2016-2105) Integer overflow in OpenSSL before 1.0.1t and 1.0.2 before 1.0.2h allows remote attackers to cause a denial of service

- (CVE-2016-2814) Heap-based buffer overflow in Mozilla Firefox before 46.0 allows remote attackers to execute arbitrary code

- (CVE-2016-1018) Stack-based buffer overflow in Adobe Flash Player allows attackers to execute arbitrary code via crafted JPEG-XR data

Infected Languages

- Directly: C/C++
- Indirectly: Almost all

Overflow In Action

```c
#include <stdio.h>
#include <string.h>

int main(int argc, char **argv) {
    char password[16];

    printf("Hello. What's the password ?");
    gets(password);

    if ( strncmp(password,
                 "letmein",
                 sizeof(password) / sizeof(char) ) == 0 ) {
        printf("Welcome, Master\n");
    } else {
        printf("INTRUDER ALERT\n");
    }
}
```

Overflow In Action

```c
#include <stdio.h>
#include <string.h>

int main(int argc, char **argv) {
    char password[16];

    printf("Hello. What's the password ?");
    gets(password);

    if ( strncmp(password,
                 "letmein",
                 sizeof(password) / sizeof(char) ) == 0 ) {
        printf("Welcome, Master\n");
    } else {
        printf("INTRUDER ALERT\n");
    }
}
```

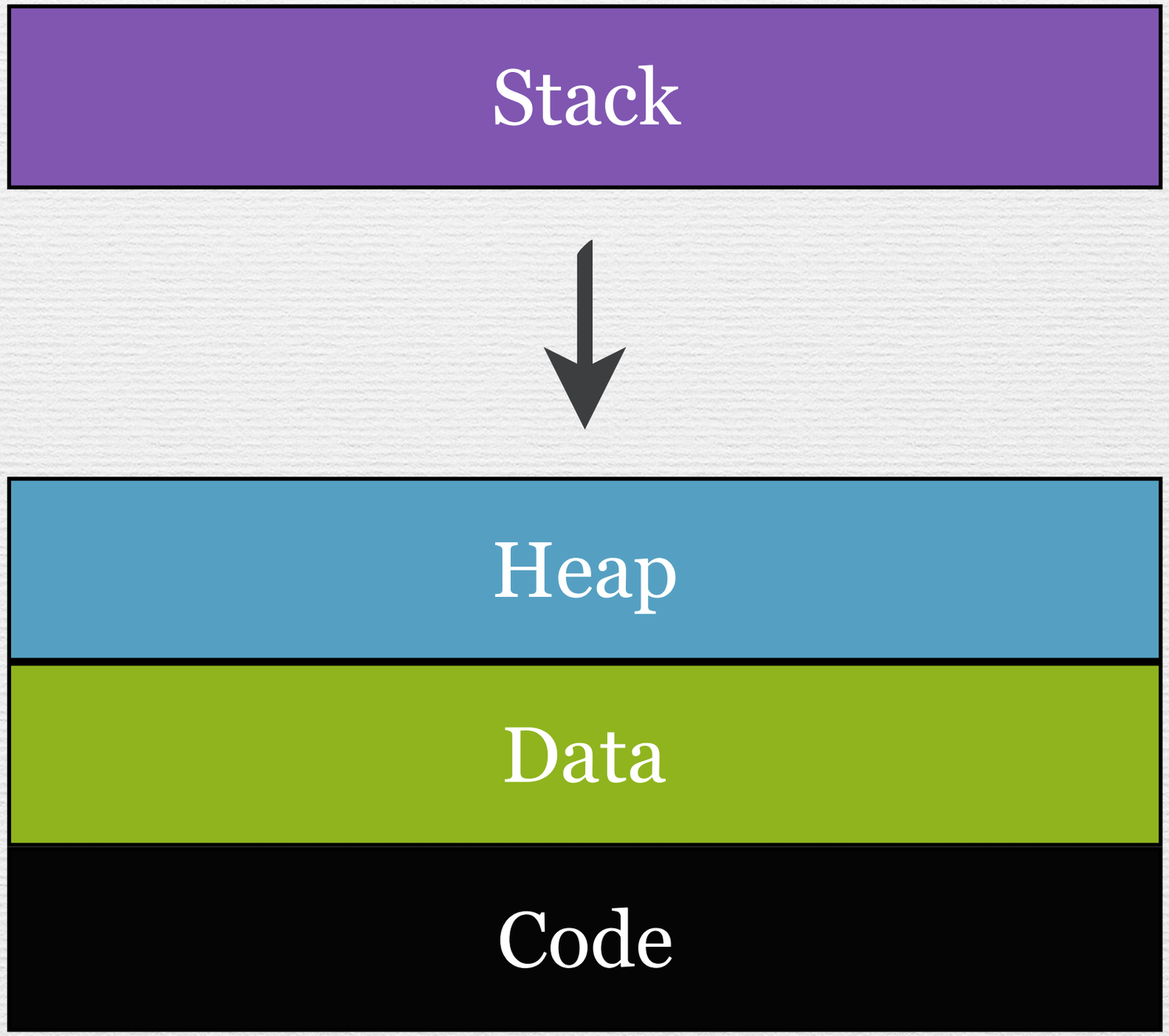
May write beyond buffer bounds

Demo

01_implementation/overflow1.c

Process Memory

Mem End



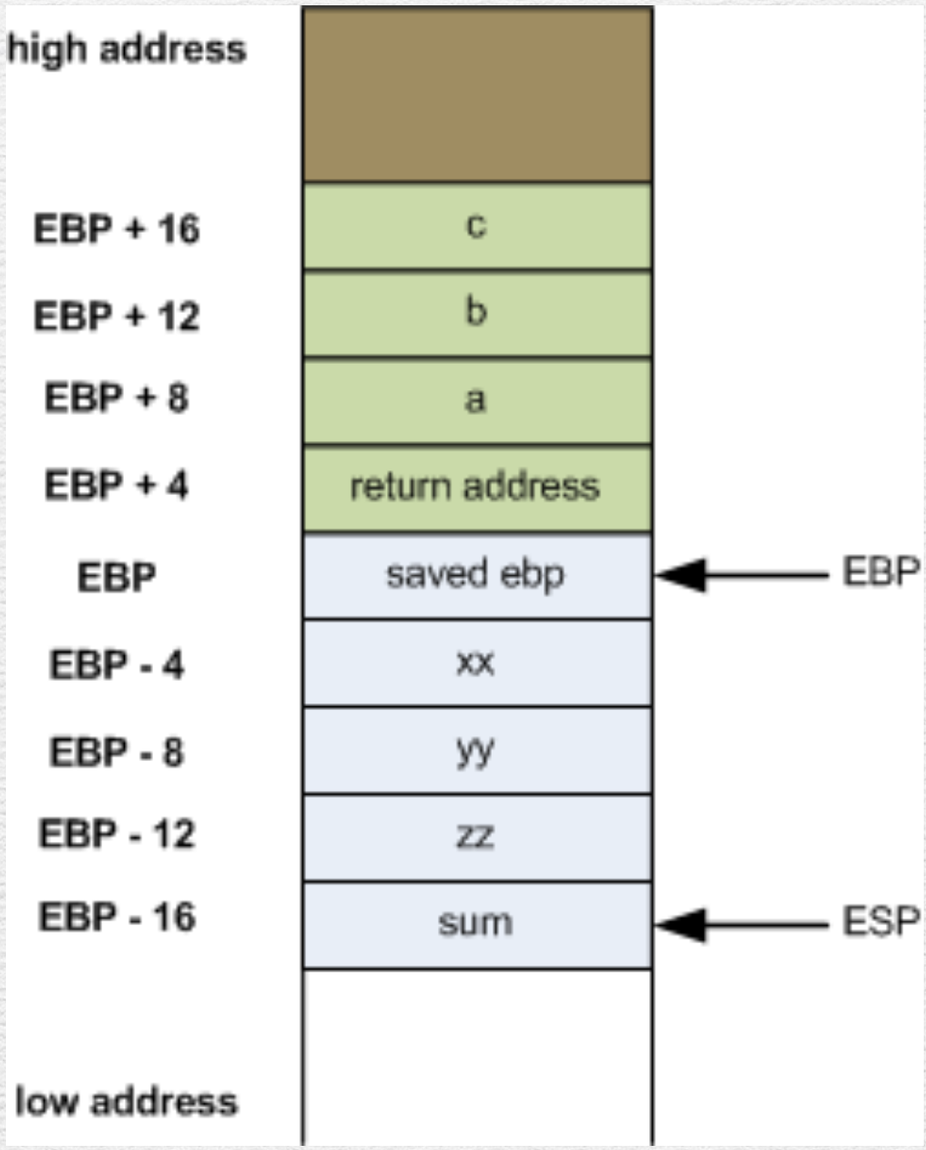
Mem start

Stack Frame

```
int foobar(int a, int b, int c)
{
   int xx = a + 2;
   int yy = b + 3;
   int zz = c + 4;
   int sum = xx + yy + zz;

   return xx * yy * zz + sum;
}

int main()
{
   return foobar(77, 88, 99);
}
```



http://eli.thegreenplace.net/2011/02/04/where-the-top-of-the-stack-is-on-x86/

The Risks

- return address is overwritten by an accidentally long write to a local buffer

Demo: Breaking In

- The function `check_password` has a memory leak
- Let's use it to modify the return address to land in the "then" block

overflow2.c

```
int main(int argc, char **argv) {

    printf("Hello. What's the
password ?");

    int master = check_password();

    if ( master ) {
        printf("Welcome, Master\n");
    } else {
        printf("INTRUDER ALERT\n");
    }
}
```

Simple Overflows

- Changing program flow through buffer overflows is called arc injection
- Compiler protection in Linux/GCC:
  `-fno-stack-protector / -fstack-protector`
- Compiler protection in Windows:
  `#pragma string_gs_check(on)`
  `compile with /GS`

Stack Canaries

Note About Protection

- Compiler protection is not as effective as you think
- Sophisticated hacks can break protection
- From MSDN:
  /GS does not protect against all buffer overrun security attacks.

/GS Won't Help For:

- Functions that do not contain a buffer.
- If optimizations are not enabled.
- Functions with a variable argument list
- Functions containing inline assembly code in the first statement
- And more

gcc has

`-fstack-protector`

default `-fstact-protector-strong`

`-fstack-protector-all`

`-fstack-protector-explicit`

Example

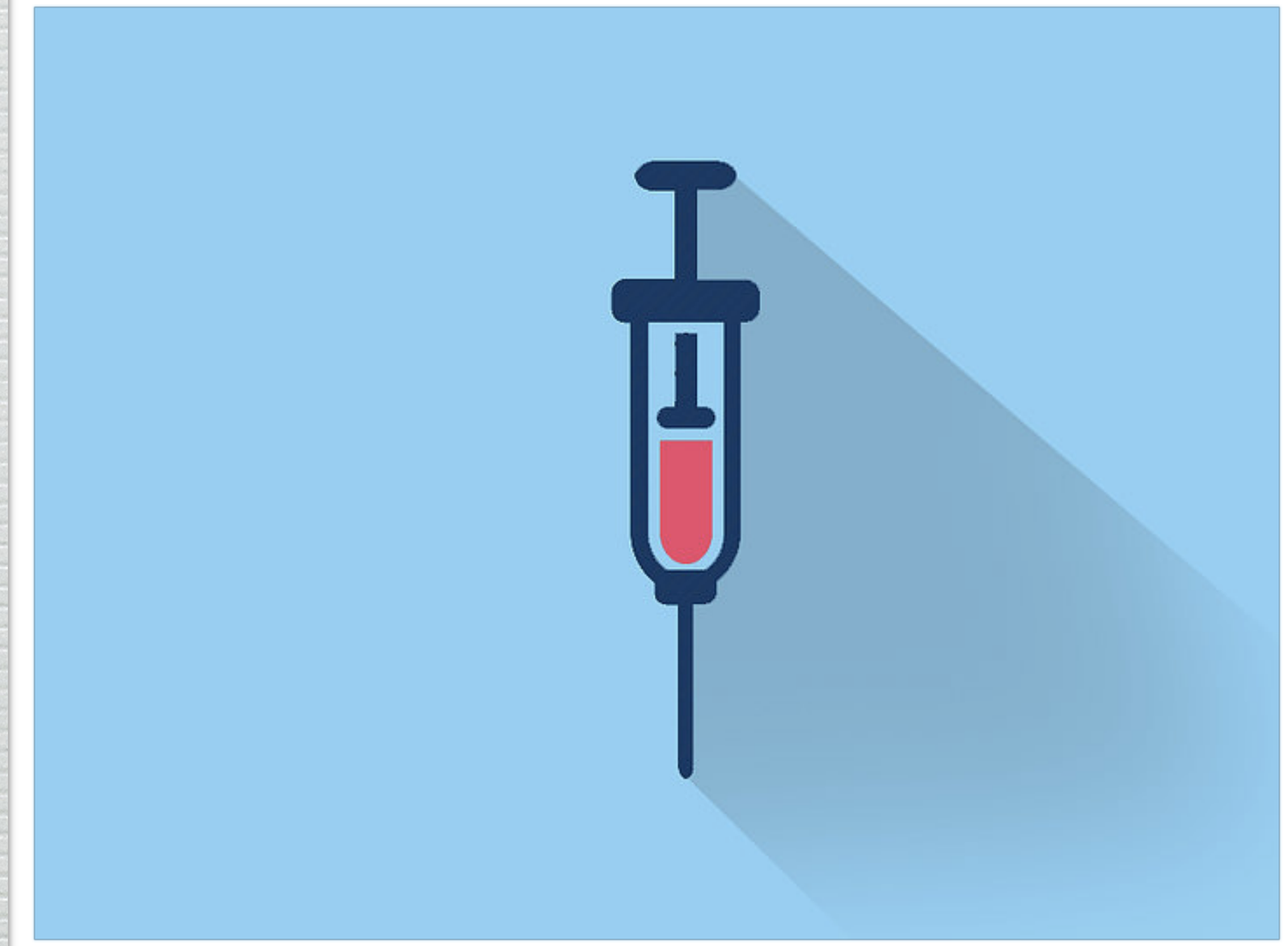
- Stack smashing protection in the following function is generated by -fstack-protector-strong but not by -fstack-protector

```
int not_protected(char *word, int size) {
  int x = 10;
  memcpy(&x, word, size);

  return x;
}
```

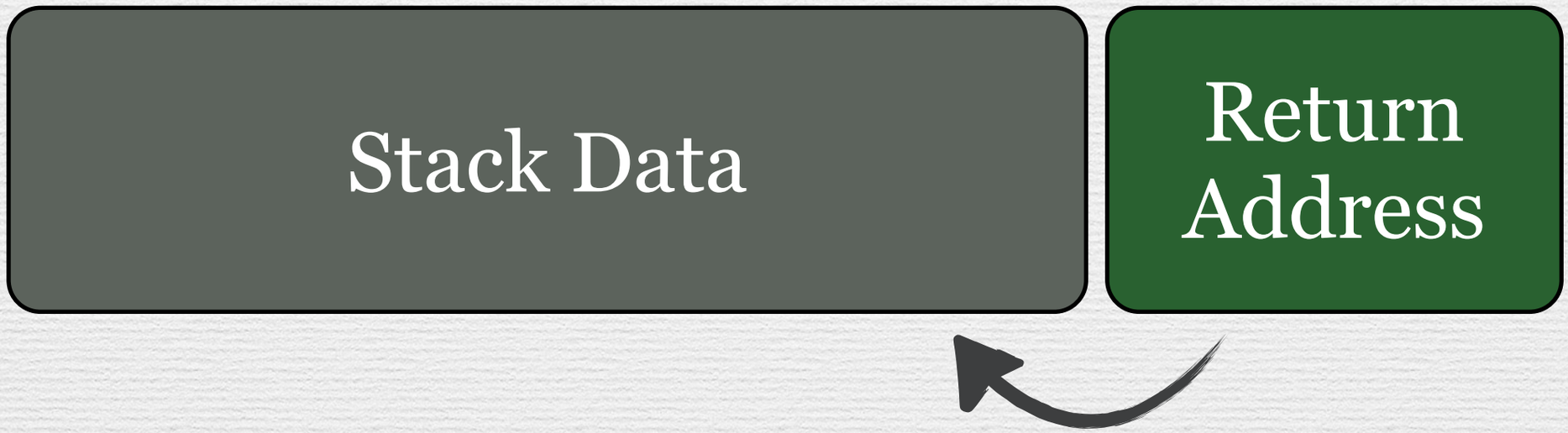
Q & A

Arc Injections



Shell Code Injection

Injecting ShellCode



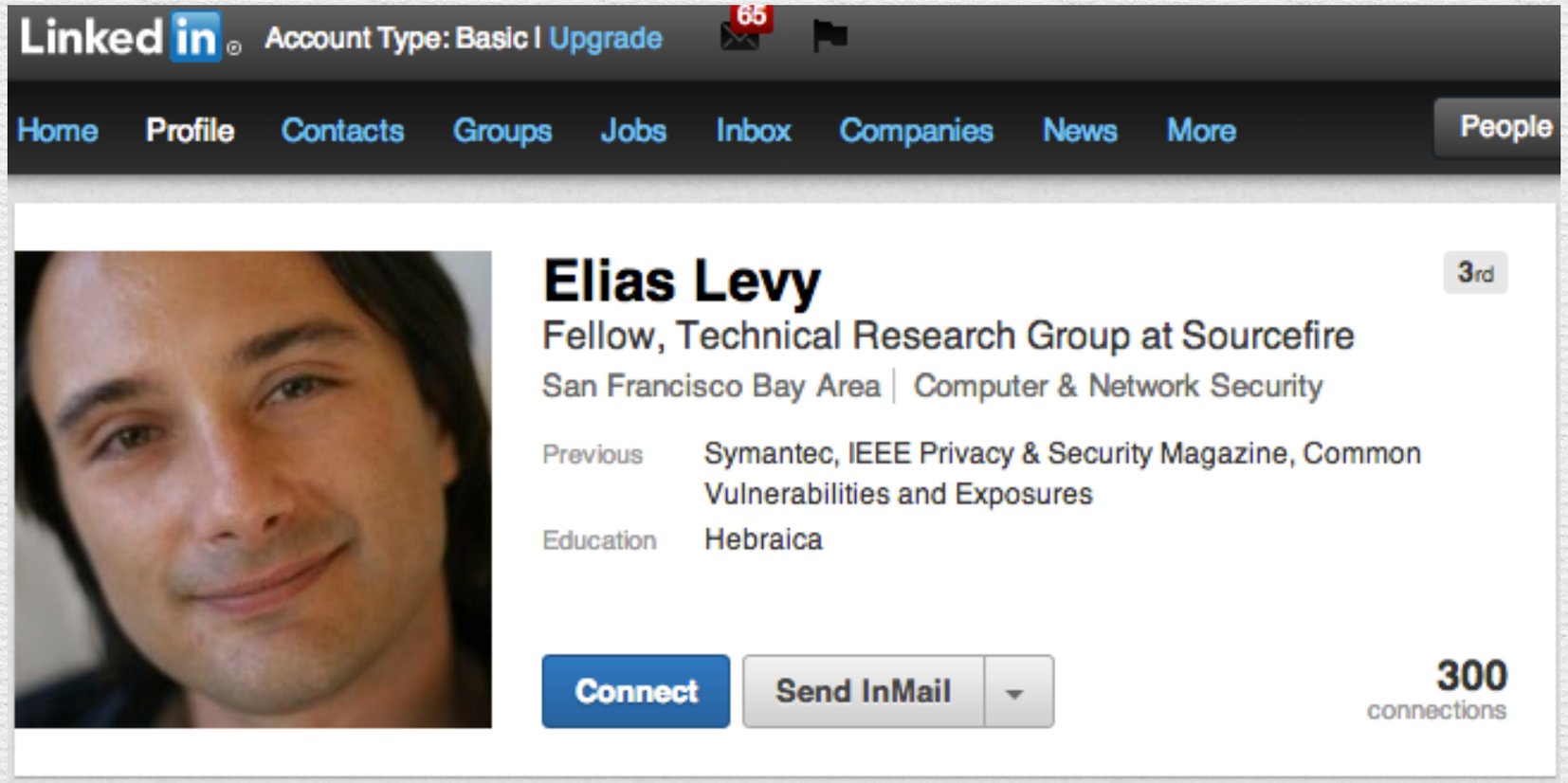
Injecting ShellCode

- Small compiled program that is injected onto the stack
- The Hack: Inject shell code, than set the stack as the return address
- Result: program runs shellcode

```
"\xeb\x18\x5e\x31\xc0\x88\x46
\x09\x89\x76\x0a"
"\x89\x46\x0e\xb0\x0b\x89\xf3
\x8d\x4e\x0a\x8d"
"\x56\x0e\xcd\x80\xe8\xe3\xff
\xff\xff\x2f"
"\x62\x69\x6e\x2f\x64\x61\x73
\x68\x41\x42\x42"
"\x42\x42\x43\x43\x43\x43";
```

History

- (1996) Smashing The Stack for Fun and Profit
- Author: Aleph One (aka Elias Levy)



Challenges

- Find the start address of the buffer
- Workarounds: pad with NOPs and guess

Demo

- Use shell code to run a shell from vulnerable program

NX Bit

- Marks areas as non-executable
- Useful to prevent injecting code on stack
- Not useful against arc or return-to-libc injections
- On linux disable with:
  `execstack -s a.out`

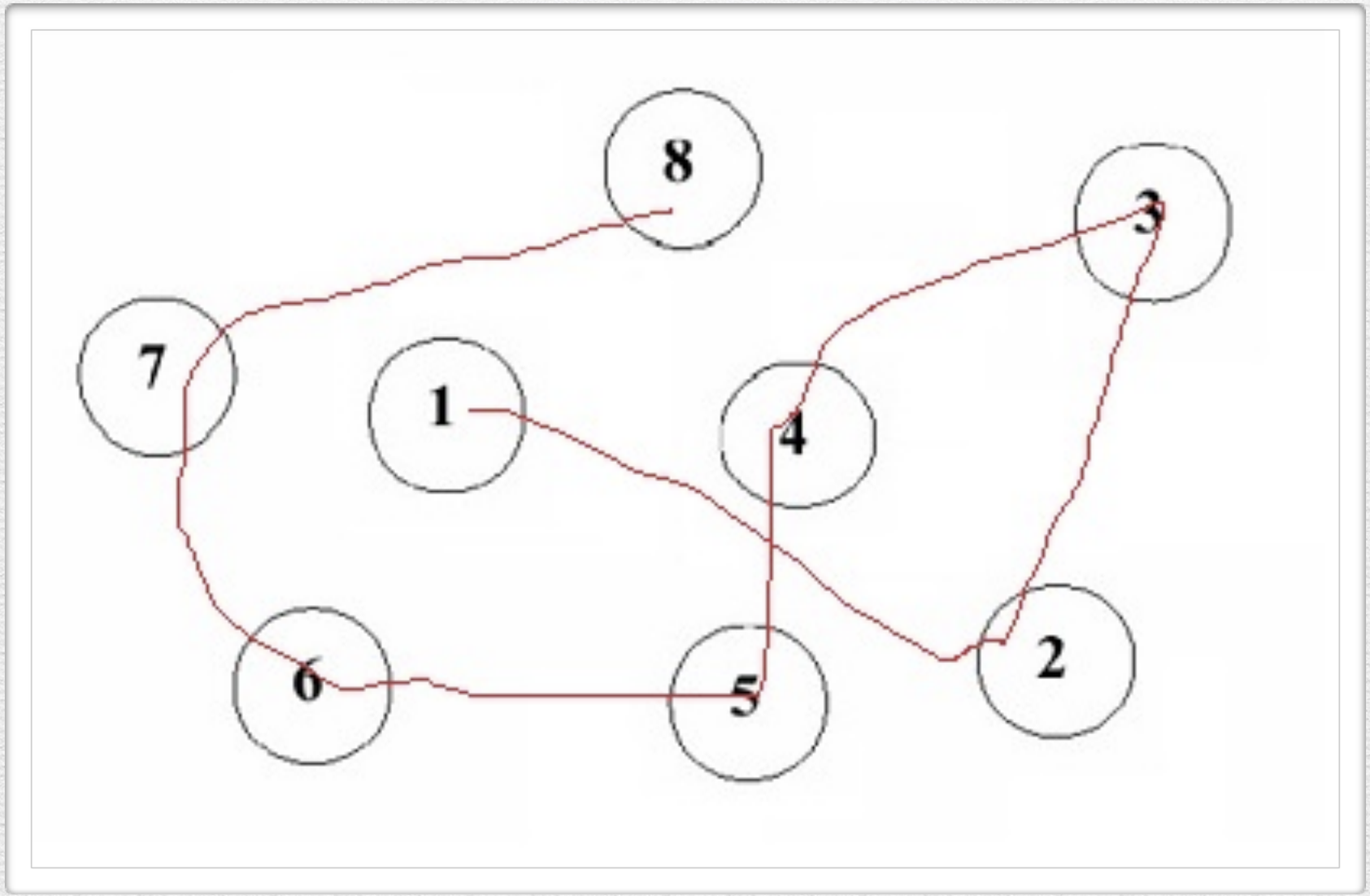
Windows NX

- Linker flag `/NXCOMPAT` on by default
- Can also be set from inside process with `SetProcessDEPPolicy`
- `SetProcessDEPPolicy` can only be called once, so it's recommended to use with: `PROCESS_DEP_ENABLE`

Back To Arcs

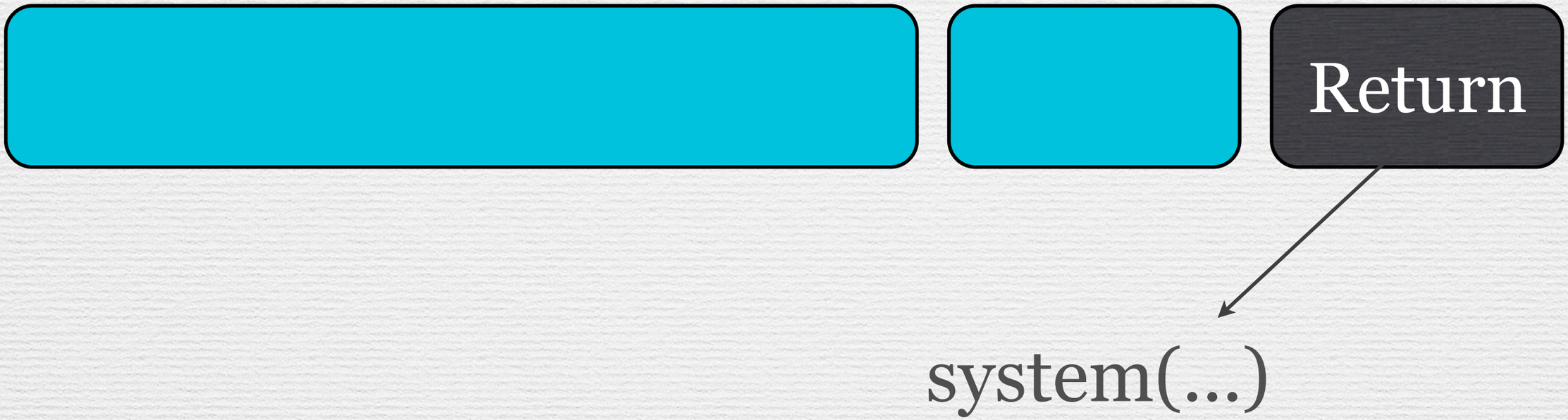
(but smarter…)

Return Oriented Programming



Before: Shell-code injection

ROP Attack



ROP Attack


```
pop ecx; pop eax; ret
“/bin/“
0x08049704
mov [ecx],eax; ret
pop ecx ; pop eax; ret
“//sh”
0x08049708
```

Demo

Finding gadgets in libc

ROP Challenges

- Hard to find the function's addresses
- Hard to find strings addresses
- AND - it only gets harder

ASLR

- Modern OS use ASLR (Address space layout randomization)
- On linux control with:
  `sysctl -w kernel.randomize_va_space = 0 / 1 / 2`
- For windows ASLR is determined by a registry key

  `KLM\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management\MoveImages`

ASLR

- Not a silver bullet
- Around since 2001
- Usually involves mechanisms to prevent brute forcing
- Easy to guess on 32 bit systems

The Best Protection

- Mitigate buffer overflows in your code
- Not as easy as you think

Lab

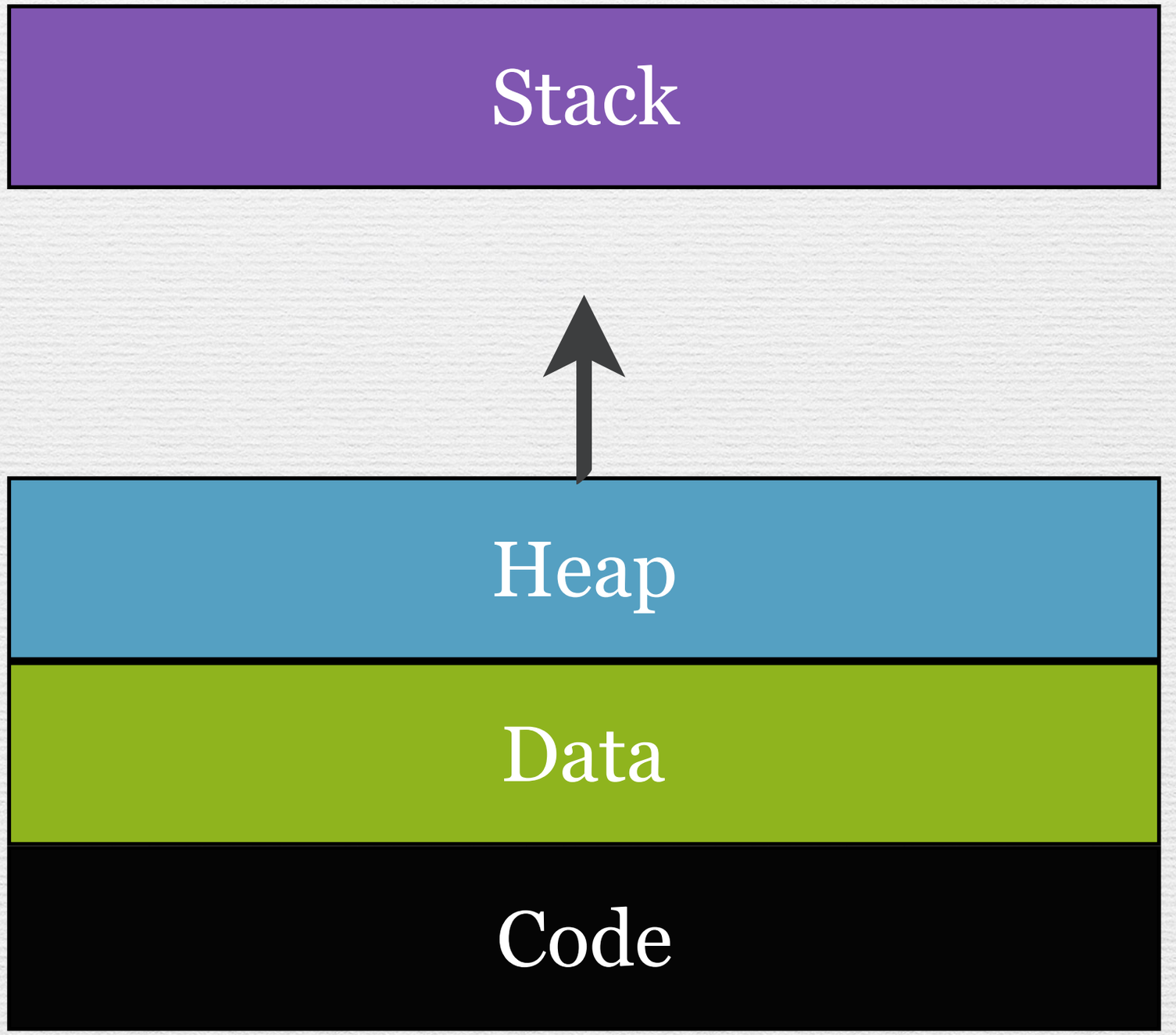
- Find memory errors and follow lab instructions in the following programs
  - labs/lab1.c
  - labs/lab2.c
  - labs/lab3.c

Other Overflows

- Heap Overflow
- Objects vtable Overflow

Process Memory

Mem End



Mem start

Process Memory

- Heap grows towards higher memory addresses
- Keeps only data (no control)

Abusing Heap

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

int main(int argc, char **argv) {
    int MAX_SIZE = 10;

    char *buf = (char *)malloc(sizeof(char) * MAX_SIZE);

    gets(buf);
    printf("Got: %s\n", buf);
}
```

Abusing Heap

- Program didn’t crash
- No return address written there

Abusing Heap

- Attacker can overwrite other heap variables
- Attacker can execute code by changing function pointers
- Attacker can abuse heap manager

Changing Memory

```
class Person {
    private:
        char buf[16];
        int m_age;

    public:
    void overflow(char *data, int len) {
        memcpy( buf, data, len );
    }
    bool canVote() { return m_age > 18; }
    bool growUp() { m_age += 1; }

    Person(): m_age(5) { }
};
```

Changing Memory

Writing after end of buf will change m_age

```
class Person {
    private:
        char buf[16];
        int m_age;

    public:
    void overflow(char *data, int len) {
        memcpy( buf, data, len );
    }
    bool canVote() { return m_age > 18; }
    bool growUp() { m_age += 1; }

    Person(): m_age(5) { }
};
```

Abusing Heap

Vulnerable Buffer

```
bool isLoggedIn() {
    return this.logged_in;
}
```

Object Person

method: isLoggedIn
method: getAge

Abusing Heap

Vulnerable Buffer

Object Person

method: isLoggedIn
method: getAge

Abusing heap memory,
An attacker can write her own address in the Person vtable

Heap Failures

- (CVE-2012-3969) Firefox < 15: Attacker can execute arbitrary code via a crafted SVG filter
- (CVE-2013-0604) Adobe Reader: Heap-based buffer overflow
- (CVE-2012-4447) LibTIFF : Heap-based buffer overflow

Heap Failures

JPEG Vulnerability - CAN-2004-0200:

A buffer overrun vulnerability exists in the processing of JPEG image formats that could allow remote code execution on an affected system. Any program that processes JPEG images on the affected systems could be vulnerable to this attack, and any system that uses the affected programs or components could be vulnerable to this attack. An attacker who successfully exploited this vulnerability could take complete control of an affected system.

Bug Spotting

- Un-sanitized / taint input (network, files, command line, environment variables)
- Use of unsafe string functions

Bug Spotting

- Runtime analyzers
- Windows: MS Application Verifier
- Linux: Valgrind

Bug Spotting

- Static Analyzers
- MS PREfast
- flawfinder

Demo

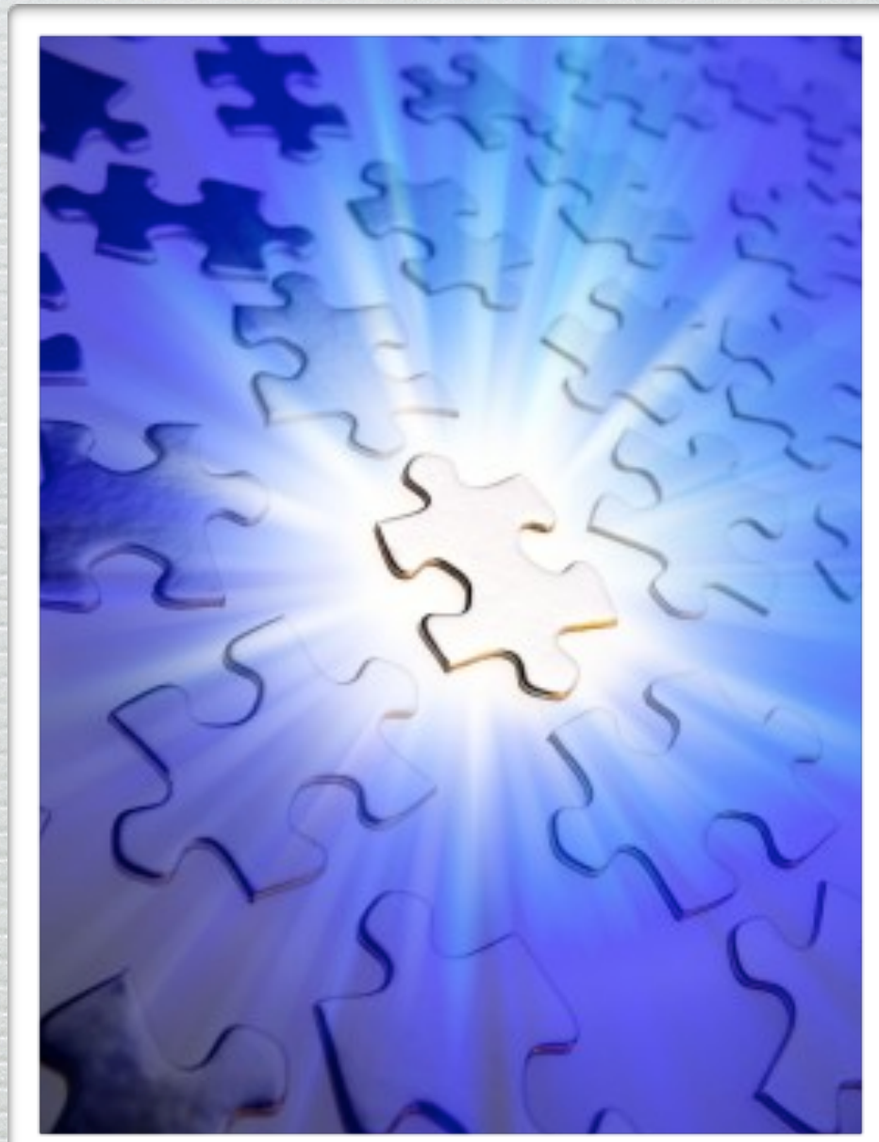
- Valgrind
- Flawfinder

How To Avoid

- Use C++ Strings
- Use STL Containers
- Use a framework (Qt)

Q & A

Buffer Overflows



Abusing Format Strings

Format Strings

- C/C++ can write formatted output using `printf( format, data )`
- Using the wrong format is a security risk

Format Strings Demo

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

int main(int argc, char **argv) {
  int x = 5;
  char name[] = "Bob";
  double PI = 3.14;

  printf("Hello ! I'm %s and I need %d PIs
          (each worth $%g )\n",
           name, x, PI);
}
```

Variadic Functions

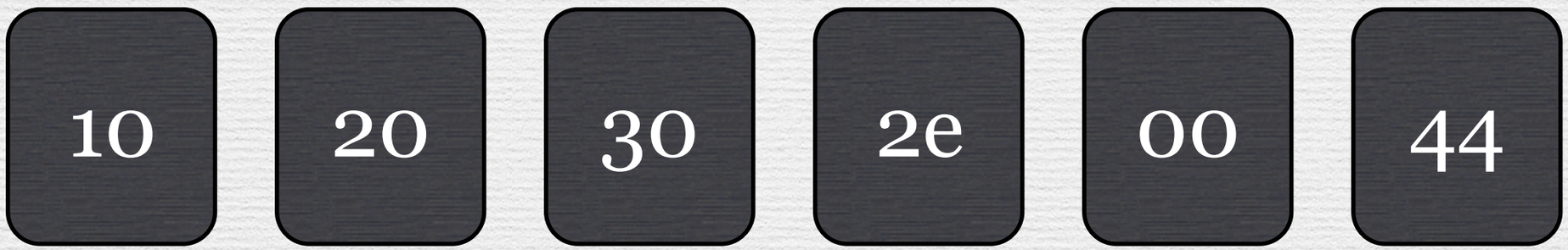
- Take a variable number of arguments
- Function can't tell how many arguments were actually passed in
- Must use other arguments as "hints"

Variadic Functions

```
double average(int count, ...)
{
    va_list ap;
    int j;
    double tot = 0;
    va_start(ap, count);
    for(j=0; j<count; j++)
        tot+=va_arg(ap, double);
    va_end(ap);
    return tot/count;
}
```

Variadic Functions

- Args are located on the stack




```
average(3, 10, 20, 30);
```

Variadic Bugs

- Reading more data than passed in
- Reading different types
- Variadic Functions don’t know arguments count or types

Story of Printf

- Formatted output can take any number of arguments - depending on the format:

```
printf("Got: %s [%d]\n", "Hello", 10);

sprintf(buf, "Got: %s [%d]\n", "Hello", 10);
```

Reading Stack Data

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


int main(int argc, char **argv) {
    if ( argc > 1 ) {
        printf(argv[1]);
    }
    return 0;
}
```

```
./a.out "0x%x 0x%x 0x%x"
```

Reading Stack Data

```
(gdb) r "%x - %x - %x - %x"
Starting program: /home/ynon/secure-code/a.out "%x - %x - %x - %x"
0 - 8048459 - b7fbdff4 - 8048450

Breakpoint 1, main (argc=2, argv=0xbfffeff4) at heap.c:10
10          return 0;
(gdb) x/20x $esp
0xbfffef40:     0x0000000a      0x00000000      0x08048459      0xb7fbdff4
0xbfffef50:     0x08048450      0x00000000      0x00000000      0xb7e314d3
0xbfffef60:     0x00000002      0xbfffeff4      0xbffff000      0xb7fdc858
```

Abusing Format Bugs

- Bypassing ASLR
- Writing random memory with “%n”

Famous Exploits

- (CVE-2012-0809) Format string vulnerability in sudo 1.8.0 allows local users to execute arbitrary code
- (CVE-2012-3569) Format string vulnerability in VMware
- (CVE-2012-3569) Format string vulnerability in exim
- (CVE-2000-0573) The lreply function in wu-ftpd 2.6.0 and earlier does not properly cleanse an untrusted format string

CVE-2012-2369

otr-plugin.c (lines 288-291)

```
1 static void log_message_cb(void *opdata, const char *message)
2 {
3     purple_debug_info("otr", message);
4 }
```

CVE-2012-2369

libpurple/debug.h (external lib)

```
void purple_debug_info
   (const char *category,
    const char *format,...)
```

CVE-2012-2369

```
1 static void log_message_cb(void *opdata, const char *message)
2 {
3     purple_debug_info("otr", message);
4 }
```

```
void purple_debug_info
  (const char *category,
   const char *format,...)
```

Bug Fixing

```c
static void log_message_cb(void *opdata, const char *message)
{
    purple_debug_info("otr", "%s", message);
}
```

Bug Spotting

- Look for printf/sprintf/fprintf
- Use the tools (Flawfinder)
- Careful with formats loaded from files - for i18n and l10n

Testing Techniques

- Use % characters in your test cases
- This applies to higher level languages as well

Avoiding The Bug

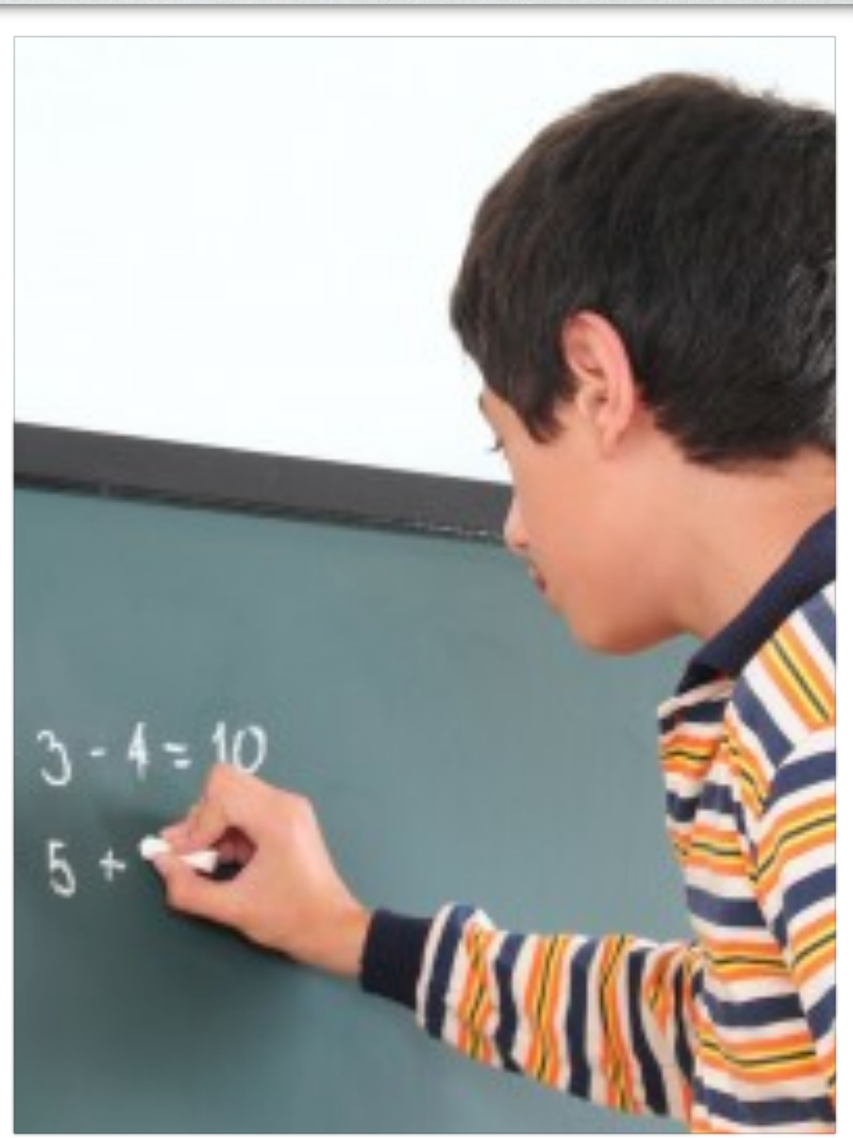
- Use C++ strings and iostream
- Use a framework (Qt / Boost)

Q & A

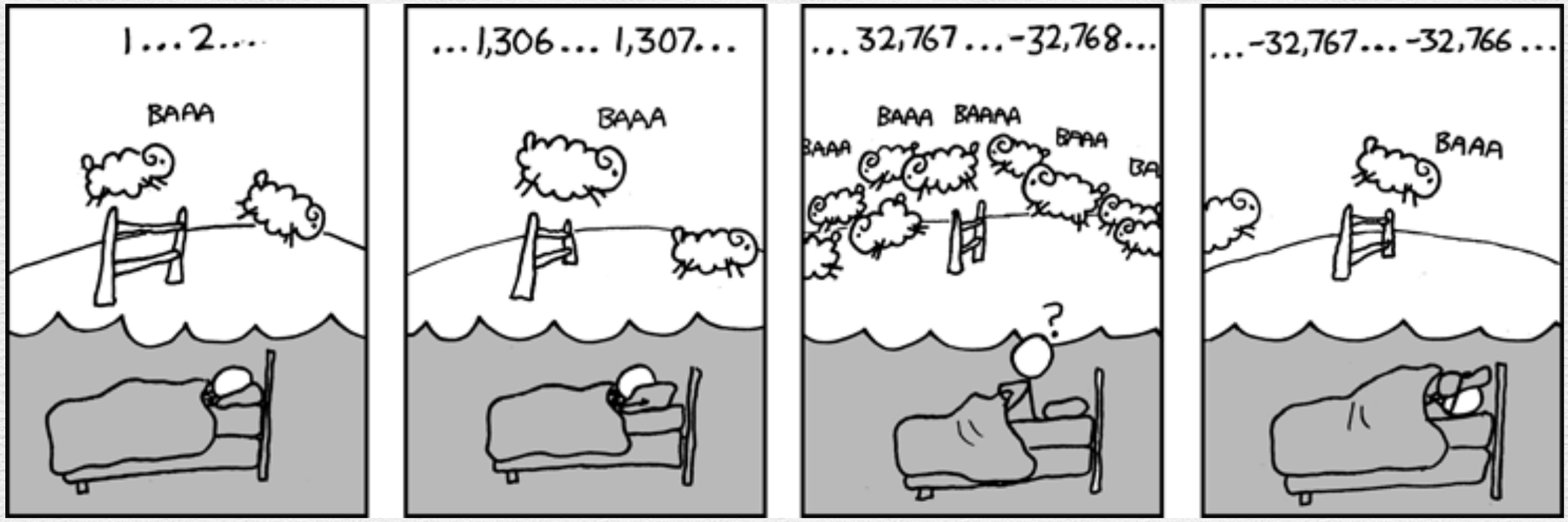
Format Strings

Evil Arithmetics

Avoiding Integer Overflows



Computers Can't Count



Affected Languages

How Computers Count

- Numbers are represented as bits and bytes
- Unsigned use all the byte
- Signed use 2’s complement

Representing Sign

Bits	Unsigned	2's complement
0000 0000	0	0
0000 0001	1	1
0000 0010	2	2
0111 1110	126	126
0111 1111	127	127
1000 0000	128	-128
1000 0001	129	-127
1000 0010	130	-126
1111 1110	254	-2
1111 1111	255	-1

Why You Should Care

```
#define MAX_BUF 256

void bad(char *input) {
    short len;
    char buf[MAX_BUF];

    len = strlen( input );

    if ( len < MAX_BUF ) {
        strcpy( buf, input );
    }
}
```

Why You Should Care

```
#define MAX_BUF 256
void bad(char *input) {
    short len;
    char buf[MAX_BUF];

    len = strlen( input );

    if ( len < MAX_BUF ) {
        strcpy( buf, input );
    }
}
```

For long enough input strlen(...) is really long, making len negative

Why You Should Care

```
#define MAX_BUF 256
void bad(char *input) {
    short len;
    char buf[MAX_BUF];

    len = strlen( input );

    if ( len < MAX_BUF ) {
        strcpy( buf, input );
    }
}
```

The test pass (len < 0) but strcpy overflows

NetBSD Length Bug

```
int off;
int len;

if ( off > len - sizeof(int) )
   goto error;
```

NetBSD Length Bug

- signed - unsigned = unsigned
- small length leads to error

Famous Overflows

- (CVE-2012-5667) Multiple integer overflows in GNU Grep before 2.11
- (CVE-2010-1634) Multiple integer overflows in audioop.c in the audioop module in Python
- (CVE-2011-3026) Integer overflow in libpng

Bug Spotting

- Use `-Wall` warning level
- Careful code using signed that should have used unsigned or `size_t`
- Using `int` for sizes or indexes

Lab

- Find the overflow in the following program
- Test and verify

```
int main(int argc, char **argv) {
    size_t total;

    total = strlen(argv[1]) + strlen(argv[2]);
    char *buf = (char *)malloc(total);

    strcpy(buf, argv[1]);
    strcat(buf, argv[2]);
}
```

Q & A

Format Strings

Related CWEs

- When we say buffer overflow, we actually mean a family of memory related errors
- That family includes:
  - Stack-Based Buffer Overflow (CWE-121)
  - Heap-Based Buffer Overflow (CWE-122)

Related CWEs

- Write-What-Where (CWE-123)
- Boundary Beginning Violation (CWE-124)
- Out-of-bounds Read (CWE-125)

Related CWEs

- Wrap-around Error (CWE-128)
- Unchecked Array Indexing (CWE-129)
- Incorrect Calculation of Buffer Size (CWE-131)

Related CWEs

- Off-By-one Error (CWE-193)
- Buffer Copy without Checking Input Size (CWE-120)

C++11 Safe Functions

- C11 Annex K adds safe functions:
  - `strcpy_s`, `strcat_s`, `strncpy_s`, `strncat_s`
  - `gets_s`
- Not supported in GCC

C++11 Safe Functions

```c
#define __STDC_WANT_LIB_EXT1__ 1
#include <stdio.h>
#include <stdlib.h>

void get_y_or_n(void)
{
  char response[8];
  size_t len = sizeof(response);

  puts("Continue ? [y] n:");
  gets_s( response, len );

  if ( response[0] == 'n' )
    exit(0);
}
```

Beyond C/C++

- Implementation bugs affect higher level languages as well
- Usually through calling library functions

Perl Open Bug

- A string in perl can hold ANY data (including null)
- Dangerous when passed to system functions

Perl Open Bug

```
use strict;
use warnings;
use v5.14;
use autodie;

my $filename = "${external_input}.html";

open my $fh, '<', $filename;
print while (<$fh>);
close $fh;
```

Perl Open Bug

```
use strict;
use warnings;
use v5.14;
use autodie;

# $0 is program name
my $filename = "$0\x00this_is_ignored.txt";

open my $fh, $filename;
print while (<$fh>);
close $fh;
```

Perl Open Bug

- If user controls file name, she can use any extension

```
my $filename = $base . ".jpg";
```

Q & A

Misusing Frameworks

Implementation Takeaways

- Ensure the use of NX, Stack Canaries (GS) and ASLR in your programs

Implementation Takeaways

- Fuzz test your programs

Implementation Takeaways

- Use static and runtime analyzing tools where applicable

Implementation Takeaways

- Use safe functions

Implementation Takeaways

Thanks For Listening

- Ynon Perek
- http://ynonperek.com
- ynon@ynonperek.com