Access Control and Privacy Policies (2)

Email: christian.urban at kcl.ac.uk
Office: S1.27 (1st floor Strand Building)
Slides: KEATS (also home work is there)

Homework

... I have a question about the homework.

Is it required to submit the homework before the next lecture?

Thank you!
Anonymous

SmartWater

- seems helpful for preventing cable theft
- wouldn't be helpful to make your property safe, because of possible abuse
- security is always a tradeoff

Plaintext Passwords from IEEE

On 25 September 2012, a report on a data breach at IEEE:

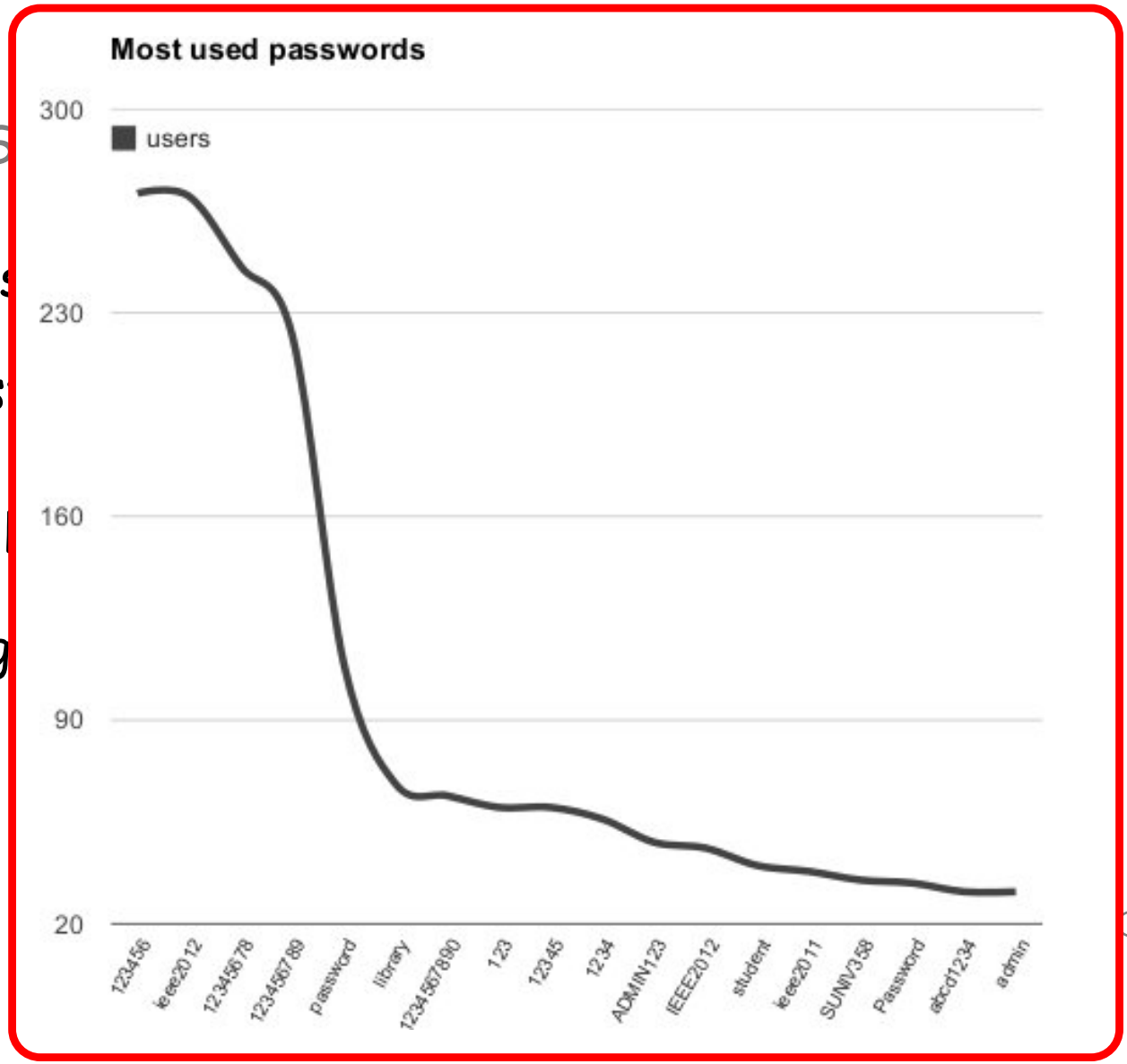
- IEEE is a standards organisation (not for profit)
- many standards in CS are by IEEE
- 100k plain-text passwords were recorded in logs
- the logs were openly accessible on their FTP server

http://ieeelog.com

Plaintext Passwords from IEEE

On 25 S IEEE:

- IEEE is
- many s
- 100k pl
- the log



og.com

Virgin Mobile (USA)

http://arstechnica.com/security/2012/09/virgin-mobile-password-crack-risk/

- for online accounts passwords must be 6 digits
- you must cycle through 1M combinations (online)

Virgin Mobile (USA)

http://arstechnica.com/security/2012/09/virgin-mobile-password-crack-risk/

- for online accounts passwords must be 6 digits
- you must cycle through 1M combinations (online)

- he limited the attack on his own account to 1 guess per second, **and**
- wrote a script that cleared the cookie set after each guess

Virgin Mobile (USA)

http://arstechnica.com/security/2012/09/virgin-mobile-password-crack-risk/

- for online accounts passwords must be 6 digits
- you must cycle through 1M combinations (online)

- he limited the attack on his own account to 1 guess per second, **and**
- wrote a script that cleared the cookie set after each guess
- has been fixed now

Smash the Stack for Fun ...

- "smashing the stack attacks" or "buffer overflow attacks"
- one of the most popular attacks;
  attack of the (last) decade
  (> 50% of security incidents reported at CERT are related to buffer overflows)

  http://www.kb.cert.org/vuls

- made popular in an article by Elias Levy
  (also known as Aleph One):

  "Smashing The Stack For Fun and Profit"

  http://www.phrack.org, Issue 49, Article 14

The Problem

- The basic problem is that library routines in C look as follows:

```
1  void strcpy(char *src, char *dst) {
2    int i = 0;
3    while (src[i] != "\0") {
4      dst[i] = src[i];
5      i = i + 1;
6    }
7  }
```

- the resulting problems are often remotely exploitable
- can be used to circumvents all access control (botnets for further attacks)

Variants

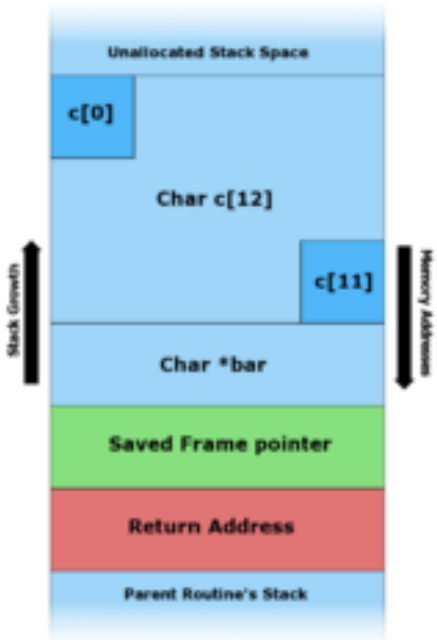
There are many variants:

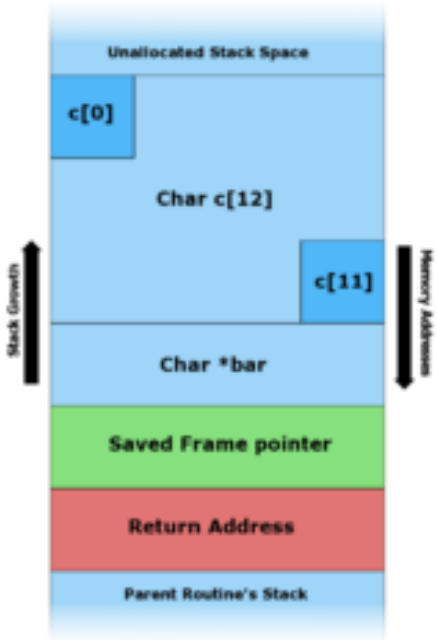
- return-to-lib-C attacks
- heap-smashing attacks
  (Slammer Worm in 2003 infected 90% of vulnerable systems within 10 minutes)

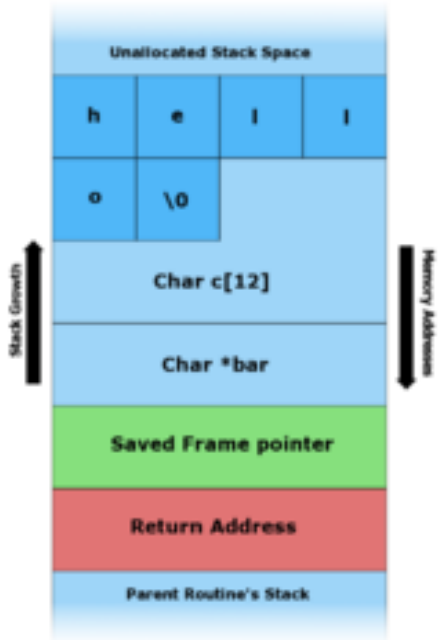
- "zero-days-attacks" (new unknown vulnerability)

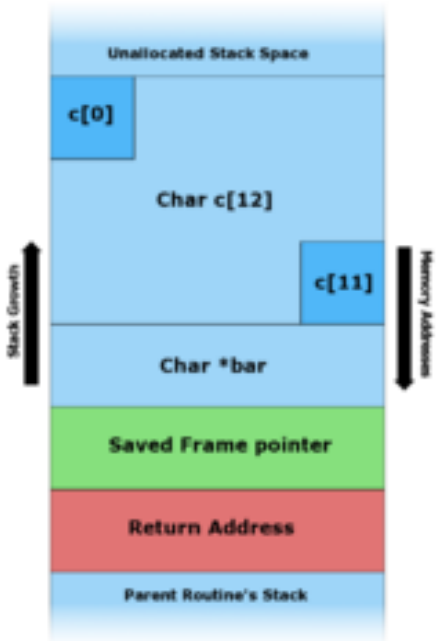
`my_float` is printed twice:

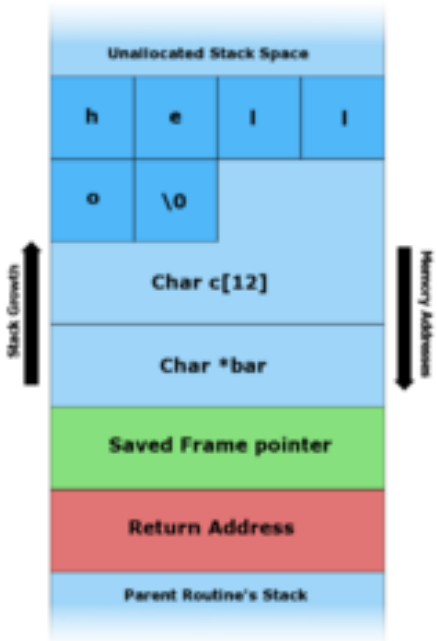
```
1  void foo (char *bar)
2  {
3    float my_float = 10.5;     // in hex: \x41\x28\x00\x00
4    char  buffer[28];
5
6    printf("my float value = %f\n", my_float);
7    strcpy(buffer, bar);
8    printf("my float value = %f\n", my_float);
9  }
10
11 int main (int argc, char **argv)
12 {
13   foo("my string is too long !!!!! ");
14   return 0;
15 }
```

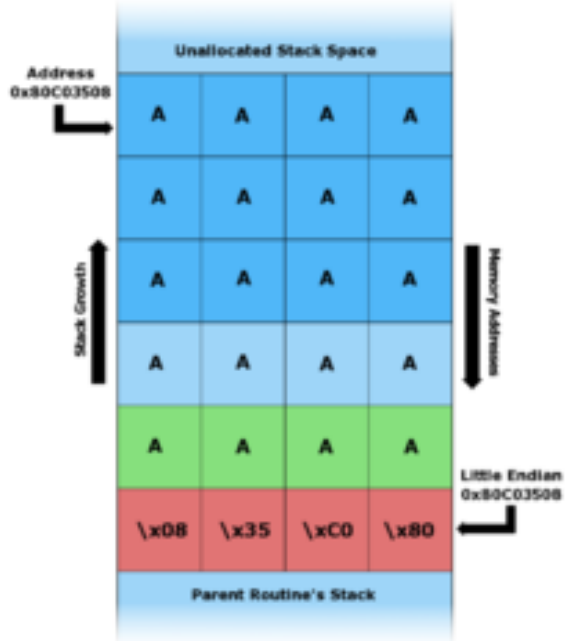

Unallocated Stack Space
c[0]
Char c[12]
c[11]
Char *bar
Saved Frame pointer
Return Address
Parent Routine's Stack
Stack Growth
Memory Addresses


Unallocated Stack Space
c[0]
Char c[12]
c[11]
Char *bar
Saved Frame pointer
Return Address
Parent Routine's Stack
Stack Growth
Memory Addresses


Unallocated Stack Space
h
e
l
l
o
\0
Char c[12]
Char *bar
Saved Frame pointer
Return Address
Parent Routine's Stack
Stack Growth
Memory Addresses


Unallocated Stack Space
c[0]
Char c[12]
c[11]
Char *bar
Saved Frame pointer
Return Address
Parent Routine's Stack
Stack Growth
Memory Addresses


Unallocated Stack Space
h
e
l
l
o
\0
Char c[12]
Char *bar
Saved Frame pointer
Return Address
Parent Routine's Stack
Stack Growth
Memory Addresses


Unallocated Stack Space
Address
0x80C03508
A
\x08
\x35
\xC0
\x80
Stack Growth
Memory Addresses
Little Endian
0x80C03508
Parent Routine's Stack

```
1  int match(char *s1, char *s2) {
2    while( *s1 != '\0' && *s2 != '\0' && *s1 == *s2 ){
3      s1++; s2++;
4    }
5    return( *s1 - *s2 );
6  }
7
8  void welcome() { printf("Welcome to the Machine!\n"); exit(0); }
9  void goodbye() { printf("Invalid identity, exiting!\n"); exit(1); }
10
11 main(){
12   char name[8];
13   char pw[8];
14
15   printf("login: ");
16   get_line(name);
17   printf("password: ");
18   get_line(pw);
19
20   if(match(name, pw) == 0)
21     welcome();
22   else
23     goodbye();
24 }
```

A programmer might be careful, but still introducing vulnerabilities:

```
1  // Since gets() is insecure and produces lots of warnings,
2  // I use my own input function instead.
3  char ch;
4  int i;
5
6  void get_line(char *dst) {
7    char buffer[8];
8    i = 0;
9    while ((ch = getchar()) != '\n') {
10     buffer[i++] = ch;
11   }
12   buffer[i] = '\0';
13   strcpy(dst, buffer);
14 }
```

Payloads

- the idea is you store some code as part to the buffer
- you then override the return address to execute this payload
- normally you start a root-shell

Payloads

- the idea is you store some code as part to the buffer
- you then override the return address to execute this payload
- normally you start a root-shell
- difficulty is to guess the place where to "jump"

Payloads (2)

- another difficulty is that the code is not allowed to contain `\x00`:

```
xorl %eax, %eax
```

```
1  void strcpy(char *src, char *dst) {
2    int i = 0;
3    while (src[i] != "\0") {
4      dst[i] = src[i];
5      i = i + 1;
6    }
7  }
```

Format String Vulnerability

`string` is nowhere used:

```
1  #include<stdio.h>
2  #include<string.h>
3
4  main(int argc, char **argv)
5  {
6    char *string = "This is a secret string\n";
7
8    printf(argv[1]);
9  }
```

this vulnerability can be used to read out the stack

Protections against BO Attacks

- use safe library functions
- ensure stack data is not executable (can be defeated)
- address space randomisation (makes one-size-fits-all more difficult)
- choice of programming language (one of the selling points of Java)

Security Goals

- Prevent common vulnerabilities from occurring (e.g. buffer overflows)

Security Goals

- Prevent common vulnerabilities from occurring (e.g. buffer overflows)
- Recover from attacks (traceability and auditing of security-relevant actions)

Security Goals

- Prevent common vulnerabilities from occurring (e.g. buffer overflows)
- Recover from attacks (traceability and auditing of security-relevant actions)
- Monitoring (detect attacks)

Security Goals

- Prevent common vulnerabilities from occurring (e.g. buffer overflows)
- Recover from attacks (traceability and auditing of security-relevant actions)
- Monitoring (detect attacks)
- Privacy, confidentiality, anonymity (to protect secrets)

Security Goals

- Prevent common vulnerabilities from occurring (e.g. buffer overflows)
- Recover from attacks (traceability and auditing of security-relevant actions)
- Monitoring (detect attacks)
- Privacy, confidentiality, anonymity (to protect secrets)
- Authenticity (eeded for access control)

Security Goals

- Prevent common vulnerabilities from occurring (e.g. buffer overflows)
- Recover from attacks (traceability and auditing of security-relevant actions)
- Monitoring (detect attacks)
- Privacy, confidentiality, anonymity (to protect secrets)
- Authenticity (eeded for access control)
- Integrity (prevent unwanted modification or tampering)

Security Goals

- Prevent common vulnerabilities from occurring (e.g. buffer overflows)
- Recover from attacks (traceability and auditing of security-relevant actions)
- Monitoring (detect attacks)
- Privacy, confidentiality, anonymity (to protect secrets)
- Authenticity (eeded for access control)
- Integrity (prevent unwanted modification or tampering)
- Availability and reliability (reduce the risk of DoS attacks)

Homework

- Assume format string attacks allow you to read out the stack. What can you do with this information.
- Assume you can crash a program remotely. Why is this a problem?