13 Computer Security: Why you should never trust a computer system

Goal: Prevent Misuse of computers

- Definitions
- Authentication
- Private and Public Key Encryption
- Access Control and Capabilities
- Enforcement of security policies
- Examples of Security Problems

13.1 Definitions

Types of Misuse

- Accidental
- Intentional

Protection is to prevent either accidental or intentional misuse
Security is to prevent intentional misuse
Three pieces to security

Authentication Who user is

Authorisation Who is allowed to do what

Enforcement Ensure that people only do what they are allowed to do

A loophole in any of these can cause problem eg

1. Log in as super-user
2. Log in as anyone, do anything
3. Can you trust software to make decisions about 1 and 2?

13.2 Authentication

Common approach: Passwords. Shared secret between two parties. Since only I know password, machine can assume it is me.

Problem 1 system must keep copy of secret, to check against password. What if malicious user gains access to this list of passwords?

Encryption Transformation on data that is difficult to reverse - in particular, secure digest functions.

13.2.1 Secure Digest Functions

A secure digest function $h = H(M)$ has the following properties:

1. Given M, is is easy to compute h.
2. Given h, it is hard to compute M.
3. Given M, it is hard to compute M' such that $H(M) = H(M')$

For example: Unix `/etc/passwd` file
Password → one way transform → encrypted password
System stores only encrypted version, so ok if someone reads the file. When you type in password, system encrypts password and compares against the stored encrypted versions.

Over the years, password protection has evolved from DES, using a well-known string as the input data, and the password as the key, through to MD5 to SHA-1.

13.2.2 Passwords as Human Factors Problem

Passwords must be long and obscure.

Paradox: short passwords are easy to crack, *but* long ones, people write down

Improving technology means we have to use longer passwords. Consider that unix initially required only lowercase 5 letter passwords

How long for an exhaustive search? $26^5 = 10,000,000$

- In 1975, 10 ms to check a password → 1 day
- In 2003, 0.00001 ms to check a password → 0.1 second

Most people choose even simpler passwords such as English words - it takes even less time to check for all words in a dictionary.

13.2.3 Some solutions

1. Extend everyone's password with a unique number (stored in passwd file). Can't crack multiple passwords at a time.
2. Require more complex passwords, eg 7 letter with lower, upper, number and special $70^7 \approx 8000$ billion, or 1 day. But people pick common patterns eg 6 lower case plus number.
3. Make it take a long time to check each password. For example, delay every login attempt by 1 second.
4. Assign long passwords. Give everyone a calculator or smart card to carry around to remember password, with PIN to activate. Need physical theft to steal card.

Problem 3 Can you trust the encryption algorithm? Recent example: techniques thought to be safe such as DES, have back doors (intentional or unintentional). If there is a back door, you don't need to do complete exhaustive search.

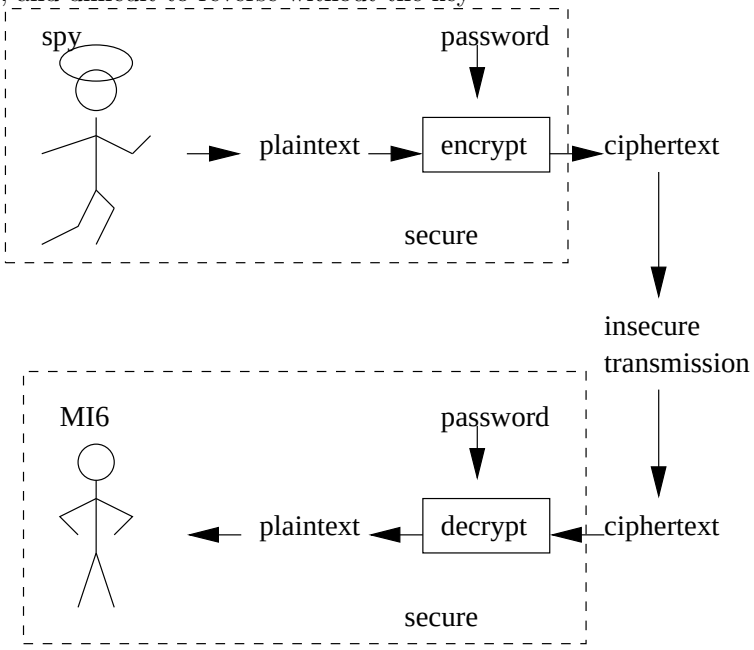
13.3 Authentication in distributed systems: Private Key Encryption

In distributed systems, the network between the machine on which the password is typed and the machine the password is authenticating on is accessible to everyone.

Two roles for encryption:

1. Authentication - do we share the same secret?
2. Secrecy - I don't want anyone to know this data (eg medical records)

Use an encryption algorithm that can easily be reversed given the correct key, and difficult to reverse without the key



- From cipher text, can't decode without password.
- From plain text and cipher text, can't derive password.
- As long as the password stays secret, we get both secrecy and authentication.

13.3.1 Symmetric Encryption

Symmetric encryptions use exclusive ors, addition, multiplication, shifts and transpositions, all of which are fast on modern processors.

DES The Data Encryption Standard (DES) was released in 1977. The 56 bit key is too weak for most uses now, and instead, 3DES or *triple DES* is used, which has 128 bit keys.

IDEA The International Data Encryption Algorithm has 128 bit keys, and has proven strong against a large body of analysis.

AES The US based NIST has defined a new encryption algorithm based on Rijndael, which offers 128, 192 or 256 bit keys.

13.3.2 Authentication Server - Kerberos Operation

The server keeps a list of passwords, provides a way for two parties, A, B to talk to each other, as long as they trust the server.

Notation: Kxy is a key for talking between x and y. (..)∧K means encrypt message (..) with key K.

Simplistic overview:

- A asks server for key:

  A → S (Hi! I'd like a key for A,B)

- Server gives back special "session" key encrypted in A's key, along with ticket to give to B:

  S → A (Use Kab (This is A! Use Kab)∧Ksb)∧Ksa

- A gives B the ticket:

  A → B ((This is A! Use Kab)∧Ksb

Details: add in timestamps to limit how long each key exists, use single use authenticators so that clients are sure of server identities, and to prevent machine replaying messages later.

Also have to include encrypted checksums to prevent malicious user inserting garbage into message.

13.4 Public Key Encryption

Public key encryption is a much slower alternative to private key; separates authentication from secrecy.

Each key is a pair K,K-1

With private key: (text)∧K∧K = text

With public key:

- (text)∧K∧K-1 = text, but (text)∧K∧K ≠ text
- (text)∧K-1∧K = text, but (text)∧K-1∧K-1 ≠ text

Can't derive K from K-1, or vice versa

13.4.1 Public key directory

Idea: K is kept secret, K-1 made public, such as public directory

For example: (I'm Ian)∧K Everyone can read it, but only I could have sent it (authentication).

(Hi!)∧K-1 Anyone can send it but only I can read it (secrecy).

((I'm Ian)∧K Hi!)∧K'-1

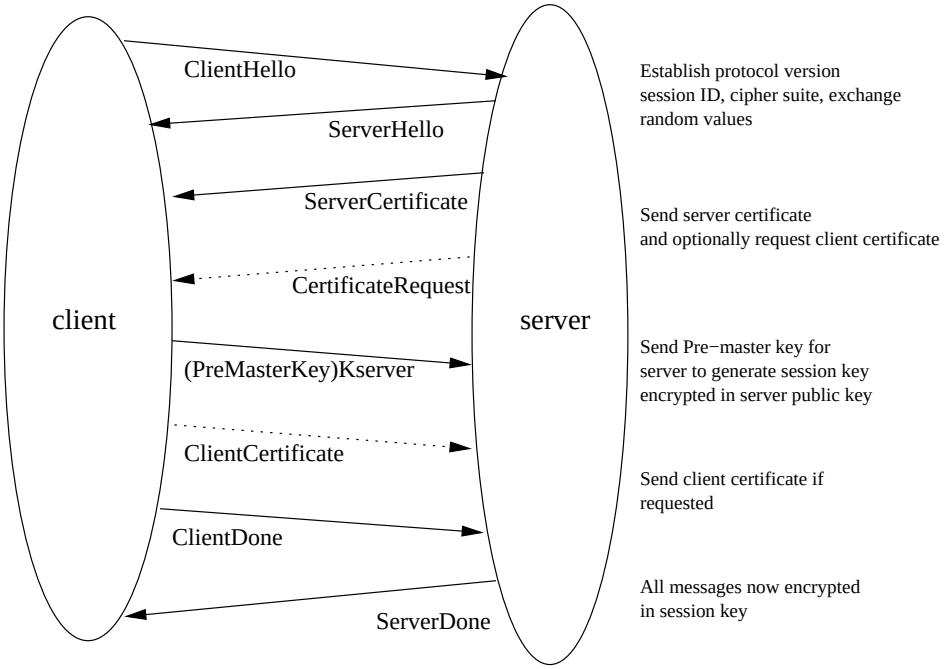
On first glance, only I can send it, only you can read it. *What's wrong with this assumption?*

Problem: How do you trust the dictionary of public keys? Maybe somebody lied to you in giving you a key.

13.5 Secure Socket Layer

- Provides a techniques for data sent over a TCP connection to be encrypted.
- Uses public key technology to agree upon a key, then 3DES or whatever to encrypt the session.
- Data encrypted in blocks, optionally with compression first.
- Used in http as **https**, and for telnet, ftp etc.

13.5.1 SSL handshake protocol



13.6 Authorisation

Who can do what...

Access control matrix: formalisation of all the permissions in the system

objects	file1	file2	file3	...
users				
A	rw	r		
B		rw		
C			r	
...				

For example, one box represents C can read file3

Potentially huge number of users and objects, so impractical to store all of these.

13.6.1 Approaches to authorisation

1. Access control list - store all permissions for all users with each object

   Still might be large number of users. Unix addresses by having rwx for user group world. Recent systems provide way of specifying groups of users and permissions for each group

2. Capability list - each process stores tickets for the objects it can operate on.

13.6.2 Digital Rights Management

- Digital content is produced by people earning a living, and they wish to protect their investment
- Digital Rights Management is the catchall term for the technologies controlling the use of digital content.
- Two main approaches:

  Containment in which the content is wrapped is an encrypted shell, so that users have to prove their *capability* of using the content through knowledge of the key.

  Watermarking where the content is marked so that devices know that the data is protected.

- The problem is how to *enforce* the checking of capabilities and *enforce* the no circumvention requirement.

13.7 Enforcement

Enforcer is the program that checks passwords, access control lists, etc

Any bug in enforcer means way for malicious user to gain ability to do anything

In Unix, superuser has all the powers of the Unix kernel - can do anything. Because of coarse-grained access control, lots of stuff has to run as superuser to work. If bug in any program, you're in trouble.

Paradox:

1. make enforcer as small as possible - easier to get correct, but simple minded protection model (more programs have high privilege).
2. fancy protection - only minimal privilege, but hard to get right.

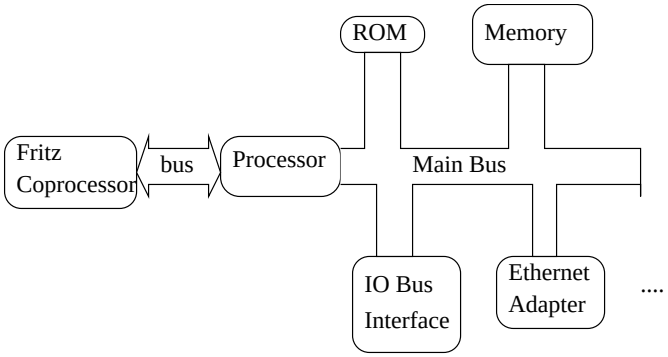
13.8 Trusted Computing Platform

Question: How do we ensure no software transgresses digital rights?

Answer: By only allowing approved software to have access to the data.

- The basic computer and operating system must be running validated software.
- There must be an untamperable part of the computer/OS that can hold keys and hand over only to validated software.
- The untamperable part of the computer is the Fritz chip, a co-processor which holds a unique certificate that it is running some validated code.

13.8.1 Startup Sequence



1. Fritz checks boot rom has been signed, executes, validates the state of the machine.
2. Fritz checks the first section of the OS has been signed, executes, loads and validates state of the machine.
3. As hardware comes up, the serial number of the hardware is checked to see if it is valid, and then the associated driver had been signed and validates the state.
4. If new hardware is added that is not valid, then the machine must be validated.

5. Once hardware is fully up, control is handed over to the validated OS.

13.8.2 Trusted Operating Systems

- Initial design came from mobile code thoughts in how to protect programmbel network components.
- Digital content can require that only validated software can be used to play it and that it can only be played on a specific machine.
- The Fritz chip uses the key it holds and with the inner kernel can unpack data, check its being passed to the certified software and is still on the correct machine.
- *Palladium*[1] is Microsoft's name for their secure OS.
- Ethical issues? Many...
- Economic issues? Many...

13.9 Firewalls

- TCP/IP has port numbers which indicate services that packets should go to.
- Firewalls inspect each packet going in and out of the network, applying pattern matches on port numbers (and other fields).
- Firewalls can prevent packets entering or leaving networks for some services.
- Filters can be triggered and stateful eg if a Telnet connection is ongoing to a host, then allow X connections from that host to the telnet source, otherwise disallow X connections.
- Genrally routed through static routed installed at ingress points for networks, although can be more distributed eg through tunneling.

13.10 Classes of security problems

Abuse of privilege If the superuser is evil, then we're all in trouble - nothing can be done

Imposter Break into system by pretending to be someone else.

For example in Unix, can set up a .rhosts file to allow logins from one machine to another, without having to retype password

Also allows `rsh` - command to do an operation on a remote node.

[1] Next Generation Secure Computing Base

Combination means: send rsh request, pretending to be from the trusted user to install a .rhosts file granting imposter full access.

Similarly, if have open X windows connectionover the network, can send message appearing to be key strokes from window but really is commands to allow imposter access. If no encryption, no way to stop this

Trojan Horse Greeks present Troy with present of wooden horse, but army hidden inside.

Trojan Horse appears helpful, but really does something harmful

Salami Attack Richard Prior in Superman 3

Idea was to build up a small bit at a time. What happens to partial pennies from interest on bank and mortgage accounts?

Bank keeps it. Re-program so that partial pennies go to programmers account. Millions of customers adds up quickly. See Internet Worm later...

Eavesdropping Listener - tap into serial line on back of terminal or onto Ethernet. See everything typed in, as almost everything goes over network unencrypted. On telnet, password goes over network unencrypted.

How can these be prevented? Hard to build system that is both useful and prevents misuse.

13.10.1 Tenex

Popular operating system at universities in early seventies before Unix

Thought to be secure. To demonstrate, created a team to find holes. Given source and documentation (wanted source to be given away, as Unix was), gave them normal account.

In 48 Hours had every password on system.

Code for password check

```
for(i=0;i<8;i++)
   if(userPasswd[i] != realPasswd[i])
      go to error
```

Appears that you'd have to try all 256^8 combinations,

But, Tenex used virtual memory, and it can interact badly with above code.

Key idea: Force page break at inopportune times. Difference in timing can reveal how far check has proceeded.

Arrange first character in string to be as last character in page, rest to be on next page. Arrange that page with the first character is in memory, page with remainder on disk, eg by referencing lots of other pages then referencing the first page.

```
a|bcdefgh
```

where a in memory, bcedefgh on disk)

By timing how long the password check takes, can figure out whether the first character is correct.

if fast, then first character is wrong.
if slow, then first character is correct, page fault, other character incorrect.

Try all first characters till one is slow. Then put first two characters on memory, remainder on disk. Try all second characters till one is slow.

Mean takes 128×8 to crack passwords.

Fix easy: don't stop till you look at all characters. But how is this known in advance?

13.10.2 Internet Worm

1988 - Worm infected thousands of machines over Internet.

Three attacks

- Dictionary lookup on passwd file
- Sendmail - Debug mode, if configured wrongly will execute commands as root.
- fingerd - finger ianw@csrp.crn

Fingerd didn't check for length of string, only allocated a fixed size array on stack. By passing a carefully crafted really long string, could overwrite stack, get the program to call arbitrary code.

Got caught because idea was to launch attacks onto other machines from whatever systems were broken into. Ended up cutting salami too thickly when machines were attacked multiple times; dragged CPU down so much that people noticed.

Variant attack: kernel checks arguments to call before using them. If multi-threaded, could co-ordinate non-calling thread to change args after kernel check.

13.10.3 Thompson's self-replicating program

Bury Trojan Horses in binaries, so no evidence in source,

Replicates to every Unix system in the world, and even onto new unixes on different platforms with no visible sign.

Would give Ken Thompson, one of original Unix designers, ability to logon to any system in world.

1. Make it possible (easy)
2. Hide it (difficult)

Step 1 Modify login.c

```
A:
  if(name == "ken")
    don't check password
    log in as root
```

Idea is to then hide change so no one can see it

Step 2 Modify the compiler

Instead of code in login.c, have code in compiler

```
B:
  if(see trigger)
    insert A into input stream
```

Whenever the compiler sees a trigger (/*gobbledygook*/), puts A into source stream into compiler.

Now, don't need A in login.c, just need trigger.

Need to get rid of problem in compiler

step 3 Modify compiler to have

```
C:
  if(see trigger2)
     insert B and C into input stream
```

Note that this can now be self-replicating code.

step 4 compile the compiler with C present

now in the binary for the compiler

step 5 Replace code for C in compiler with trigger2

Result - everything is in binary. As long as trigger2 remains in compiler source, will replicate C, with B, with A, to every compiler built.

Every time login.c recompiled, backdoor created. Every time compiler recompiled, the compiler re-inserts backdoor inserter.

When porting to a new machine, Compiler used to generate new cross compiler, and then new compiler from source code written in c.

13.11 Lessons

1. Hard to resecure system after penetration

   How do you remove backdoor? Remove triggers?

   But what if another trigger in editor? If observer trigger being removed, re-insert trigger on saving file.

2. Hard to detect when system has been penetrated. Easy to make system forget

3. Any system has loopholes, and every system has bugs.

4. The more complex the system, the more bugs - KISS