Security

AFNOG 3 Workshop

1

Main Security Concerns

- Confidentiality
  - Keeping our data safe from prying eyes
- Integrity
  - Protecting our data from loss or unauthorised alteration
- Authentication and Authorisation
  - Is this person who they claim to be?
  - Is this person allowed to do this?
- Availability
  - Are our systems working when we need them? (Denial of Service)

Security Implications of connecting to the Internet

- The Internet lets you connect to millions of hosts
  - but they can also connect to you!
- Many points of access (e.g. telephone, X25)
  - even if you can trace an attack to a point on the Internet, the real source may be untraceable
- Your host runs many Internet services
  - many potential points of vulnerability
  - many servers run as "root" !

3

Network-based attacks

- Passive attacks
  - e.g. packet sniffers, traffic analysis
- Active attacks
  - e.g. connection hijacking, IP source spoofing, exploitation of weaknesses in IP stack or applications
- Denial of Service attacks
  - e.g. synflood
- Attacks against the network itself
  - e.g. smurf

Other common attacks

- Brute-force and Dictionary attacks (password guessing)
- Viruses
- Trojan horses
- Humans are often the weakest link
  - "Hi, this is Bob, what's the root password?"

5

Authentication: Passwords

- Can be guessed
- If too complex, users tend to write them down
- If sent unencrypted, can be "sniffed" from the network and re-used

Choosing good passwords

- Combinations of upper and lower-case letters, numbers and symbols
  - 'brute force' attacker has to try many more combinations
- Not in any dictionary, including hackers dictionaries

$40&yc4f
"Money for nothing and your chicks for free"

wsR!vst?
"workshop students aRe not very sleepy today ?"

Authentication: Source IP address

- Not verified by the network (since not used in datagram delivery)
- Datagrams are easily forged
- TCP 3-way handshake gives some degree of protection, as long as you can't guess TCP sequence numbers
  - Legitimate example: controlling SMTP relaying by source IP address
- Any UDP protocol is completely vulnerable
  - e.g. NFS

Authentication: Host name

- Very weak
- DNS is easily attacked (e.g. by loading false information into cache)
- Slight protection by ensuring that reverse and forward DNS matches
  - e.g. Connection received from 80.248.72.254
  - Lookup 80.248.72.254 -> noc.ws.afnog.org
  - Lookup noc.ws.afnog.org -> 80.248.72.254
- This is why many sites won't let you connect unless your forward and reverse matches

Cryptographic methods

- Can provide REALLY SECURE solutions to authentication, privacy and integrity
- Some are hard to implement, many different tools, usually requires special clients
- Export and usage restrictions (less of a problem these days)
- Take care to understand where the weaknesses lie

Simple combinations

- The lock on your front door can be picked
- Two locks are better than one
- The thief is more likely to try somewhere else

IP source address AND password authentication

- You can use "tcp wrappers" (/etc/hosts.allow) to add IP source authentication to any service run from inetd
  - For info and examples: man 5 hosts_access
- The application also typically has password authentication

Exercise

- Enable telnet (note: bad idea!)
  - Uncomment telnet ... tcp line in /etc/inetd.conf
  - killall -1 inetd
  - Check other people can telnet to your machine
- Now restrict access to only yourself and your neighbour
  - Add two lines to top of /etc/hosts.allow
  - telnetd : 80.248.72.12, 80.248.72.11 : allow
  - telnetd : ALL : deny
- Get someone on a different row to try to telnet to you. What happens if you telnet to 127.0.0.1 ?

UNDERSTAND what you're doing

- A bad security solution is worse than no security at all
- Know what you're doing
  - Read all the documentation
  - Read sample configurations
  - Build test machines
  - Ask questions
  - Join the announcements mailing list for your O/S and applications
- Test what you've done
  - Try connecting from outside your network
  - Try circumventing your own rules

Summary

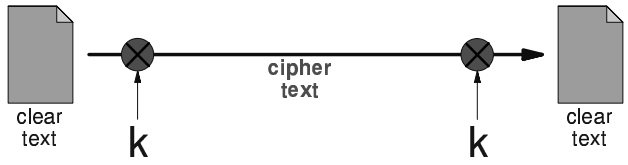
- Disable all services which are not needed
- Apply security patches promptly; join the announcement mailing lists
- Good password management
- Combine passwords with IP access controls where possible
- Use cryptographic methods where possible

Cryptographic methods:

Three important components of cryptographic systems

Recommended reading:
"Applied Cryptography", Bruce Schnier

1. "Private key" or "symmetric" ciphers



The same key is used to encrypt the document before sending and decrypt it at the far end

We assume an eavesdropper is able to intercept the ciphertext

- How can they recover the cleartext?

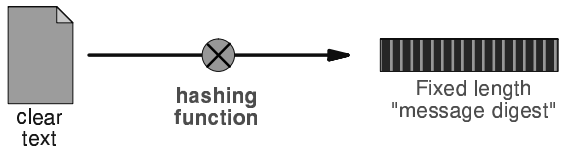
Examples of symmetric ciphers

- DES - 56 bit key length, designed by US security service
- 3DES - effective key length 112 bits
- AES (Advanced Encryption Standard) - 128 to 256 bit key length
- Blowfish - 128 bits, optimised for fast operation on 32-bit microprocessors
- IDEA - 128 bits, patented (requires a licence for commercial use)

Features of symmetric ciphers

- Fast to encrypt and decrypt, suitable for large volumes of data
- A well-designed cipher is only subject to brute-force attack; the strength is therefore directly related to the key length
- Current recommendation is a key length of at least 90 bits
  - i.e. to be fairly sure that your data will be safe for at least 20 years
- Problem - how do you distribute the keys?

2. "Hashing" - one-way encryption



Munging the document gives a short "message digest" (checksum). Not possible to go back from the digest to the original document.

Examples

- Unix crypt() function
- MD5 (Message Digest 5) - 128 bit hash
- SHA1 (Secure Hash Algorithm) - 160 bits
- No two documents have yet been discovered which have the same MD5 digest!
- No feasible method to create any document which has a given MD5 digest

So what use is that? a. Integrity checks

- You can run many megabytes of data through MD5 and still get only 128 bits to check
- An attacker cannot feasibly modify your file and leave it with the same MD5 checksum
- Gives your document a unique "fingerprint"

Exercise

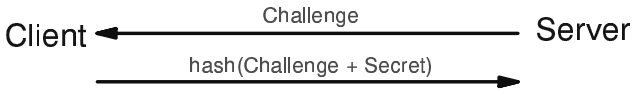
- Exercise: on your machine type
  - cat /etc/motd
- Look at your neighbour's machine. Is their file *exactly* the same as yours? Can you be sure?
- md5 /etc/motd
  - Compare the result with your neighbour
- Now change ONE character in /etc/motd and repeat the md5 test
- Under Linux the command is "md5sum"

So what use is that?
b. Encrypted password storage

- We don't want to keep cleartext passwords if possible; the password file would be far too attractive a target
- Store hash(passwd) in /etc/master.passwd
- When user logs in, calculate the hash of the password they have given, and compare it to the hash in the password file
- If the two hashes match, the user must have entered the correct password

So what use is that?
c. Shared secret authentication

- Using hashing, a user can prove that they possess a password, without actually sending it over the wire
- Usually called a "shared secret" in this case



Server recalculates the hash using the challenge it sent and its local copy of the secret. OK if both hashes match.

Notes on shared secret authentication

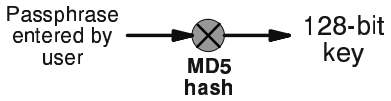
- Examples: APOP, CRAM-MD5
- Sniffer cannot see the secret - but they *can* see the challenge and hash of (challenge + secret). This will allow them to try dictionary and brute-force attacks to recover the secret.
- The secret must be stored in PLAIN TEXT on the server for this method to work.

So what use is that?
d. Generating encryption keys

- Users cannot remember 128 bit binary encryption keys
- However they can remember "passphrases"
- A hash can be used to convert a passphrase into a fixed-length encryption key
- The longer the passphrase, the more "randomness" it contains and the harder to break. English text is typically only 1.3 bits of randomness per character.

http://www.cranfield.ac.uk/docs/email/pgp/pgp-attack-faq.txt
http://www.counterpane.com/personal-entropy.html

Generating encryption keys



Every passphrase generates a different 128-bit key

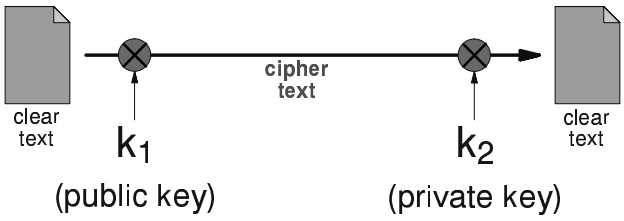
So what use is that?
e. one-time passwords

- S/Key (OPIE)
- Cryptocard

So what use is that?
f. Registering copyright

- By giving someone the MD5 digest of a document, I can prove that I possessed the document at that time, without having to reveal its contents until later
- Lots of other uses

3. "Public key" ciphers

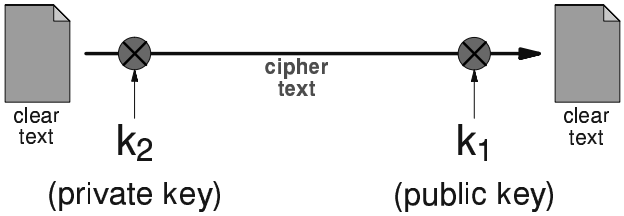


One key is used to encrypt the document, a different key is used to decrypt it

Public key and Private key

- The Public key and Private key are mathematically related (generated as a pair)
- It is easy to convert the Private key into the Public key. It is not easy to do the reverse.
- Key distribution problem is solved: you can post your public key anywhere. People can use it to encrypt messages to you, but only the holder of the private key can decrypt them.
- Examples: RSA, Elgamal (DSA)

Use for authentication: reverse the roles of the keys

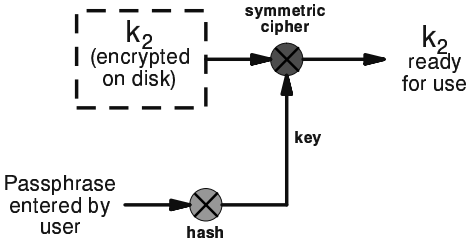


If you can decrypt the document with the public key, it proves it was written by the owner of the private key (and was not changed)

Protecting the private key

- The security of the private key is paramount: keep it safe!
- Keep it on a floppy or a smartcard?
- Prefer to keep it *encrypted* if on a hard drive
- That means you have to decrypt it (using a passphrase) each time you use it
- An attacker would need to steal the file containing the private key, AND know or guess the passphrase

Protecting the private key



Key lengths

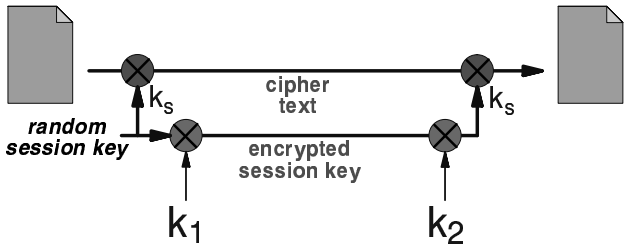
- Attacks on public key systems involve mathematical attempts to convert the public key into the private key. This is more efficient than brute force.
- Recent developments suggest that 1024-bit keys might not be secure for long
- Recommend using 2048-bit keys

Public key cryptosystems require a lot of computation

- So we use some tricks to minimise the amount of data which is encrypted

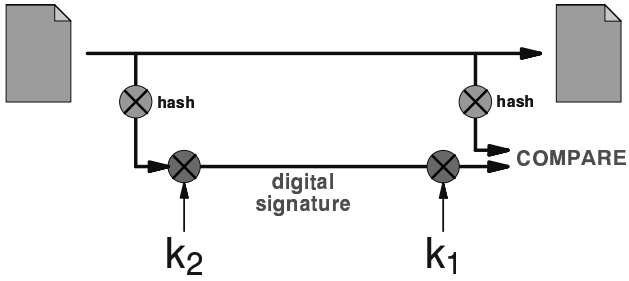
When encrypting:

- Use a symmetric cipher with a random key (the "session key"). Use a public key cipher to encrypt the session key and send it along with the encrypted document.



When authenticating:

- Take a hash of the document and encrypt only that. An encrypted hash is called a "digital signature"



Digital Signatures have many uses...

- E-commerce. An instruction to your bank to transfer money can be authenticated with a digital signature.
  - Legislative regimes are slow to catch up
- A trusted third party can issue declarations such as "the holder of this key is a person who is legally known as Alice Hacker"
  - like a passport binds your identity to your face
- Such a declaration is called a "certificate"
- You only need the third-party's public key to check the signature

Where can you apply these cryptgraphic methods?

- At the link layer
  - PPP encryption
- At the network layer
  - IPSEC
- At the transport layer
  - TLS (SSL)
- At the application layer
  - SSH, PGP/GPG