Access Control and Privacy Policies (4)

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

two weeks ago: buffer overflow attacks

Buffer Overflows

```
As a proof-of-concept, the following URL allows
attackers to control the return value saved on
the stack (the vulnerability is triggered when
executing ”/usr/sbin/widget”):

 curl http://<target ip>/post_login.xml?hash=AAA...AAABBBB

The value of the ”hash” HTTP GET parameter consists in
292 occurrences of the ’A’ character, followed by four
occurrences of character ’B’. In our lab setup, characters
’B’ overwrite the saved program counter (%ra).

Discovery date: 06/03/2013
Release date:   02/08/2013
```

http://pastebin.com/vbiG42VD

Backdoors

D-Link router flaw lets anyone login through "Joel's Backdoor":

If you tell your browser to identify itself as Joel's backdoor, instead of (say) as Mozilla/5.0 AppleWebKit/536.30.1 Version/6.0.5, you're in without authentication.

"What is this string," I hear you ask?
You will laugh: it is

```
xmlset_roodkcableoj28840ybtide
```

October 15, 2013
http://www.devttys0.com/2013/10/reverse-engineering-a-d-link-backdoor/

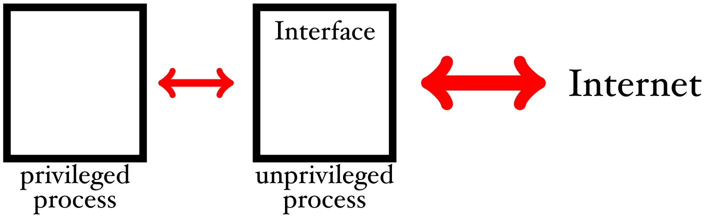
Access Control in Unix

- access control provided by the OS
- authenticate principals (login)
- mediate access to files, ports, processes according to roles (user ids)
- roles get attached with privileges

> principle of least privilege:
> programs should only have as much privilege as they need

Access Control in Unix (2)

- the idea is to restrict access to files and therefore lower the consequences of an attack



Access Control

- Discretionary Access Control:

  Access to objects (files, directories, devices, etc.) is permitted based on user identity. Each object is owned by a user. Owners can specify freely (at their discretion) how they want to share their objects with other users, by specifying which other users can have which form of access to their objects.

  Discretionary access control is implemented on any multi-user OS (Unix, Windows NT, etc.).

Access Control

- Mandatory Access Control:

  Access to objects is controlled by a system-wide policy, for example to prevent certain flows of information. In some forms, the system maintains security labels for both objects and subjects (processes, users), based on which access is granted or denied. Labels can change as the result of an access. Security policies are enforced without the cooperation of users or application programs.

  This is implemented today in special military operating system versions (SELinux).

Discretionary Access Control

In its most generic form usually given by an Access Control Matrix of the form

	/mail/jane	edit.exe	sendmail
jane	r, w	r, x	r, x
john	$\emptyset$	r, w, x	r, x
sendmail	a	$\emptyset$	r, x

access privileges: **r**ead, **w**rite, e**x**ecute, **a**ppend

Mandatory Access Control

- Restrictions to allowed information flows are not decided at the user's discretion (as with Unix chmod), but instead enforced by system policies.
- Mandatory access control mechanisms are aimed in particular at preventing policy violations by untrusted application software, which typically have at least the same access privileges as the invoking user.

  Simple example: Air Gap Security. Uses completely separate network and computer hardware for different application classes.

The Bell/LaPadula Model

- Formal policy model for mandatory access control in a military multi-level security environment. All subjects (processes, users, terminals) and data objects (files, directories, windows, connections) are labeled with a confidentiality level, e.g.

  unclassified < confidential < secret < top secret.

- The system policy automatically prevents the flow of information from high-level objects to lower levels. A process that reads top secret data becomes tagged as top secret by the operating system, as will be all files into which it writes afterwards.

Bell-LaPadula

- Read Rule: A principal P can read an object O if and only if P's security level is at least as high as O's.
- Write Rule: A principal P can write an object O if and only if O's security level is at least as high as P's.
- Meta-Rule: All principals in a system should have a sufficiently high security level in order to access an object.

This restricts information flow $\Rightarrow$ military

Bell-LaPadula

- Read Rule: A principal P can read an object O if and only if P's security level is at least as high as O's.
- Write Rule: A principal P can write an object O if and only if O's security level is at least as high as P's.
- Meta-Rule: All principals in a system should have a sufficiently high security level in order to access an object.

This restricts information flow $\Rightarrow$ military

Bell-LaPadula: **'no read up'** - **'no write down'**

Principle of Least Privilege

A principal should have as few privileges as possible to access a resource.

- Bob (*TS*) and Alice (*S*) want to communicate

  ⇒ Bob should lower his security level

Biba Policy

Data Integrity (rather than data confidentiality)

- Biba: **'no read down' - 'no write up'**
- Read Rule: A principal P can read an object O if and only if P's security level is lower or equal than O's.
- Write Rule: A principal P can write an object O if and only if O's security level is lower or equal than P's.

Biba Policy

Data Integrity (rather than data confidentiality)

- Biba: **'no read down'** - **'no write up'**
- Read Rule: A principal P can read an object O if and only if P's security level is lower or equal than O's.
- Write Rule: A principal P can write an object O if and only if O's security level is lower or equal than P's.

E.g. Firewalls: you can read from inside the firewall, but not from outside
Phishing: you can look at an approved PDF, but not one from a random email

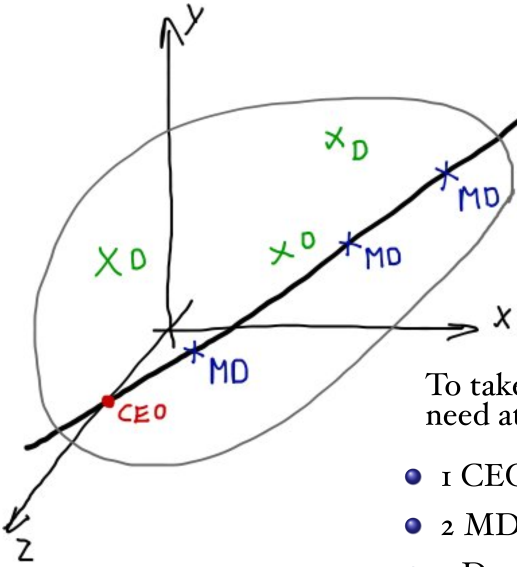
Security Levels (2)

- Bell — La Padula preserves data secrecy, but not data integrity

Security Levels (2)

- Bell — La Padula preserves data secrecy, but not data integrity
- Biba model is for data integrity
  - read: your own level and above
  - write: your own level and below

Shared Access Control



To take an action you need at least either:

- 1 CEO
- 2 MDs, or
- 3 Ds

Lessons from Access Control

Not just restricted to Unix:

- if you have too many roles (i.e. too finegrained AC), then hierarchy is too complex
  you invite situations like...let's be root
- you can still abuse the system...

Protocols

$$A \;\rightarrow\; B : \ldots$$

- by convention A, B are named principals Alice...
  but most likely they are programs, which just
  follow some instructions (they are more like roles)

Protocols

$$\begin{array}{lcl} A & \rightarrow & B : \ldots \\ B & \rightarrow & A : \ldots \\ & \vdots & \end{array}$$

- by convention A, B are named principals Alice... but most likely they are programs, which just follow some instructions (they are more like roles)

- indicates one "protocol run", or session, which specifies some order in the communication
- there can be several sessions in parallel (think of wifi routers)

A mutual authentication protocol

$$
\begin{aligned}
A \to B&: \quad N_a \\
B \to A&: \quad \{N_a, N_b\}_{K_{ab}} \\
A \to B&: \quad N_b
\end{aligned}
$$

A mutual authentication protocol

$$\begin{aligned} A \rightarrow B&: \quad N_a \\ B \rightarrow A&: \quad \{N_a, N_b\}_{K_{ab}} \\ A \rightarrow B&: \quad N_b \end{aligned}$$

An attacker E can launch an impersonation attack by intercepting all messages for B and make A decrypt her own challenges.

Nonces

1. I generate a nonce (random number) and send it to you encrypted with a key we share
2. you increase it by one, encrypt it under a key I know and send it back to me

I can infer:

- you must have received my message
- you could only have generated your answer after I send you my initial message
- if only you and me know the key, the message must have come from you

$$
\begin{aligned}
A \rightarrow B &: \quad N_a \\
B \rightarrow A &: \quad \{N_a, N_b\}_{K_{ab}} \\
A \rightarrow B &: \quad N_b
\end{aligned}
$$

The attack:

$$
\begin{aligned}
A \rightarrow E &: \quad N_a \\
E \rightarrow A &: \quad N_a \\
A \rightarrow E &: \quad \{N_a, N_a\}_{K_{ab}} \\
E \rightarrow A &: \quad \{N_a, N_a\}_{K_{ab}} \\
A \rightarrow E &: \quad N_a \quad (= N_b)
\end{aligned}
$$

$$
\begin{aligned}
A \rightarrow B&: \quad N_a \\
B \rightarrow A&: \quad \{N_a, N_b\}_{K_{ab}} \\
A \rightarrow B&: \quad N_b
\end{aligned}
$$

The attack:

$$
\begin{aligned}
A \rightarrow E&: \quad N_a \\
E \rightarrow A&: \quad N_a \\
A \rightarrow E&: \quad \{N_a, N_a\}_{K_{ab}} \\
E \rightarrow A&: \quad \{N_a, N_a\}_{K_{ab}} \\
A \rightarrow E&: \quad N_a \quad (= N_b)
\end{aligned}
$$

Solutions: $K_{ab} \neq K_{ba}$ or include an id in the second message

Identify Friend or Foe

Identify Friend or Foe

198?: war between Angola (supported by Cuba) and Namibia (supported by SA)

Identify Friend or Foe

198?: war between Angola (supported by Cuba) and Namibia (supported by SA)

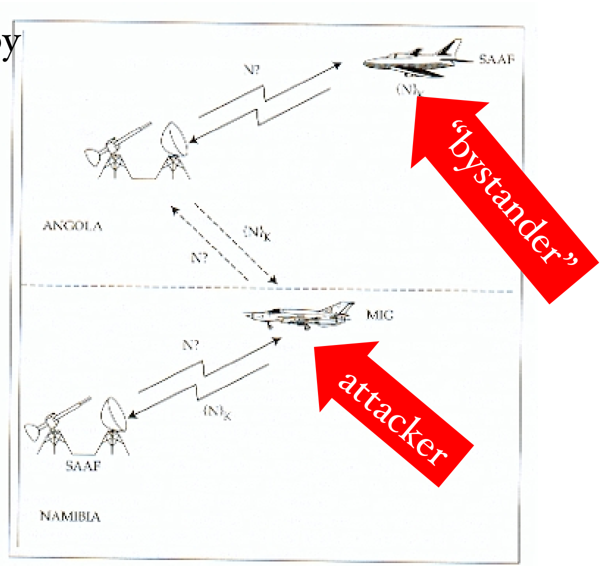


Figure 2.2 The MIG-in-the-middle attack.

Identify Friend or Foe

198?: war between Angola (supported by Cuba) and Namibia (supported by SA)

being outsmarted by Angola/Cuba ended SA involvement (?)

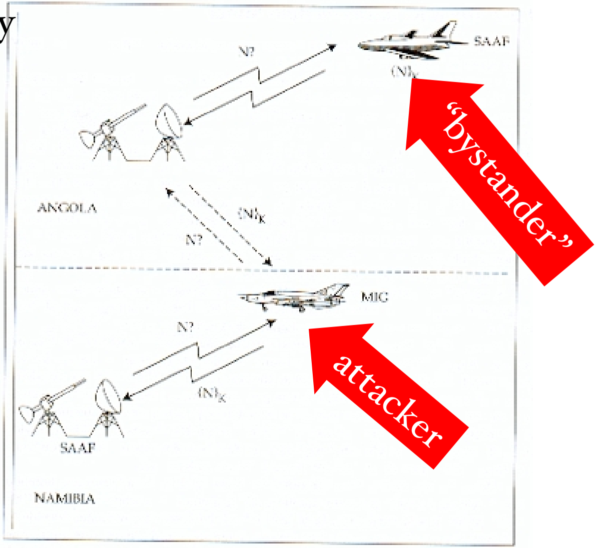


Figure 2.2 The MIG-in-the middle attack.

Identify Friend or Foe

198?: war between Angola (supported by Cuba) and Namibia (supported by SA)

being outsmarted by Angola/Cuba ended SA involvement (?)

IFF opened up a nice side-channel attack

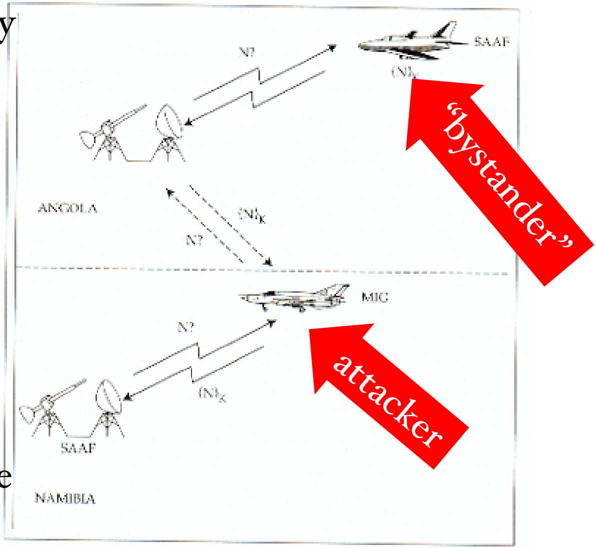


Figure 2.2 The MIG-in-the middle attack.

Encryption to the Rescue?

- $A \to B : \{A, N_A\}_{K_{AB}}$ encrypted
- $B \to A : \{N_A, K'_{AB}\}_{K_{AB}}$
- $A \to B : \{N_A\}_{K'_{AB}}$

Encryption to the Rescue?

- $A \to B : \{A, N_A\}_{K_{AB}}$ encrypted
- $B \to A : \{N_A, K'_{AB}\}_{K_{AB}}$
- $A \to B : \{N_A\}_{K'_{AB}}$

means you need to send separate "Hello" signals (bad), or worse share a single key between many entities

Protocol Attacks

- replay attacks
- reflection attacks
- man-in-the-middle attacks
- timing attacks
- parallel session attacks
- binding attacks (public key protocols)
- changing environment / changing assumptions

- (social engineering attacks)

Public-Key Infrastructure

- the idea is to have a certificate authority (CA)
- you go to the CA to identify yourself
- CA: "I, the CA, have verified that public key P^{pub}_{Bob} belongs to Bob"
- CA must be trusted by everybody
- What happens if CA issues a false certificate? Who pays in case of loss? (VeriSign explicitly limits liability to $100.)

Binding Attacks

with public-private keys it is important that the public key is bound to the right owner (verified by a certification authority CA)

$$A \rightarrow CA : A, B, N_A$$
$$CA \rightarrow A : CA, \{CA, A, N_A, K_B^{pub}\}_{K_A^{pub}}$$

A knows K_A^{priv} and can verify the message came from CA in response to A's message and trusts K_B^{pub} is B's public key

Binding Attacks

$$A \rightarrow I(CA) : A, B, N_A$$
$$I(A) \rightarrow CA : A, I, N_A$$
$$CA \rightarrow I(A) : CA, \{CA, A, N_A, K_I^{pub}\}_{K_A^{pub}}$$
$$I(CA) \rightarrow A : CA, \{CA, A, N_A, K_I^{pub}\}_{K_A^{pub}}$$

Binding Attacks

$$A \to I(CA) : A, B, N_A$$
$$I(A) \to CA : A, I, N_A$$
$$CA \to I(A) : CA, \{CA, A, N_A, K_I^{pub}\}_{K_A^{pub}}$$
$$I(CA) \to A : CA, \{CA, A, N_A, K_I^{pub}\}_{K_A^{pub}}$$

A now encrypts messages for B with the public key of I (which happily decrypts them with its private key)

Replay Attacks

Schroeder-Needham protocol: exchange of a symmetric key with a trusted 3rd-party S:

$$
\begin{aligned}
A \rightarrow S &: A, B, N_A \\
S \rightarrow A &: \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}} \\
A \rightarrow B &: \{K_{AB}, A\}_{K_{BS}} \\
B \rightarrow A &: \{N_B\}_{K_{AB}} \\
A \rightarrow B &: \{N_B - 1\}_{K_{AB}}
\end{aligned}
$$

Replay Attacks

Schroeder-Needham protocol: exchange of a symmetric key with a trusted 3rd-party S:

$$
\begin{array}{l}
A \rightarrow S : A, B, N_A \\
S \rightarrow A : \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}} \\
A \rightarrow B : \{K_{AB}, A\}_{K_{BS}} \\
B \rightarrow A : \{N_B\}_{K_{AB}} \\
A \rightarrow B : \{N_B - 1\}_{K_{AB}}
\end{array}
$$

at the end of the protocol both A and B should be in the possession of the secret key K_{AB} and know that the other principal has the key

$$A \to S : A, B, N_A$$
$$S \to A : \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$$
$$A \to B : \{K_{AB}, A\}_{K_{BS}}$$
$$B \to A : \{N_B\}_{K_{AB}}$$
$$A \to B : \{N_B - 1\}_{K_{AB}}$$

$$A \to S : A, B, N_A$$
$$S \to A : \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$$
$$A \to B : \{K_{AB}, A\}_{K_{BS}}$$
$$B \to A : \{N_B\}_{K_{AB}}$$
$$A \to B : \{N_B - 1\}_{K_{AB}}$$

compromise K_{AB}

$$A \rightarrow S : A, B, N_A$$
$$S \rightarrow A : \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$$
$$A \rightarrow B : \{K_{AB}, A\}_{K_{BS}}$$
$$B \rightarrow A : \{N_B\}_{K_{AB}}$$
$$A \rightarrow B : \{N_B - 1\}_{K_{AB}}$$

compromise K_{AB}

$$A \rightarrow S : A, B, N'_A$$
$$S \rightarrow A : \{N'_A, B, K'_{AB}, \{K'_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$$

$$A \to S : A, B, N_A$$
$$S \to A : \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$$
$$A \to B : \{K_{AB}, A\}_{K_{BS}}$$
$$B \to A : \{N_B\}_{K_{AB}}$$
$$A \to B : \{N_B - 1\}_{K_{AB}}$$

compromise K_{AB}

$$A \to S : A, B, N'_A$$
$$S \to A : \{N'_A, B, K'_{AB}, \{K'_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$$
$$I(A) \to B : \{K_{AB}, A\}_{K_{BS}}$$ replay of older run

$$A \to S : A, B, N_A$$
$$S \to A : \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$$
$$A \to B : \{K_{AB}, A\}_{K_{BS}}$$
$$B \to A : \{N_B\}_{K_{AB}}$$
$$A \to B : \{N_B - 1\}_{K_{AB}}$$

compromise K_{AB}

$$A \to S : A, B, N'_A$$
$$S \to A : \{N'_A, B, K'_{AB}, \{K'_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$$
$$I(A) \to B : \{K_{AB}, A\}_{K_{BS}}$$ replay of older run
$$B \to I(A) : \{N'_B\}_{K_{AB}}$$
$$I(A) \to B : \{N'_B - 1\}_{K_{AB}}$$

$$A \to S : A, B, N_A$$
$$S \to A : \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$$
$$A \to B : \{K_{AB}, A\}_{K_{BS}}$$
$$B \to A : \{N_B\}_{K_{AB}}$$
$$A \to B : \{N_B - 1\}_{K_{AB}}$$

compromise K_{AB}

$$A \to S : A, B, N'_A$$
$$S \to A : \{N'_A, B, K'_{AB}, \{K'_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$$
$$I(A) \to B : \{K_{AB}, A\}_{K_{BS}}$$ replay of older run
$$B \to I(A) : \{N'_B\}_{K_{AB}}$$
$$I(A) \to B : \{N'_B - 1\}_{K_{AB}}$$

B believes it is following the correct protocol, intruder I can form the correct response because it knows K_{AB} and talks to B masquerading as A

Time-Stamps

The Schroeder-Needham protocol can be fixed by including a time-stamp (e.g., in Kerberos):

$$
\begin{aligned}
A \rightarrow S &: A, B, N_A \\
S \rightarrow A &: \{N_A, B, K_{AB}, \{K_{AB}, A, T_S\}_{K_{BS}}\}_{K_{AS}} \\
A \rightarrow B &: \{K_{AB}, A, T_S\}_{K_{BS}} \\
B \rightarrow A &: \{N_B\}_{K_{AB}} \\
A \rightarrow B &: \{N_B - 1\}_{K_{AB}}
\end{aligned}
$$

Time-Stamps

The Schroeder-Needham protocol can be fixed by including a time-stamp (e.g., in Kerberos):

$$
\begin{aligned}
A &\rightarrow S : A, B, N_A \\
S &\rightarrow A : \{N_A, B, K_{AB}, \{K_{AB}, A, T_S\}_{K_{BS}}\}_{K_{AS}} \\
A &\rightarrow B : \{K_{AB}, A, T_S\}_{K_{BS}} \\
B &\rightarrow A : \{N_B\}_{K_{AB}} \\
A &\rightarrow B : \{N_B - 1\}_{K_{AB}}
\end{aligned}
$$

but nothing is for free: then you need to synchronise time and possibly become a victim to timing attacks

Changing Environment Attacks

- all protocols rely on some assumptions about the environment (e.g., cryptographic keys cannot be broken)

Changing Environment Attacks

- all protocols rely on some assumptions about the environment (e.g., cryptographic keys cannot be broken)

- in the "good olden days" (1960/70) rail transport was cheap, so fraud was not worthwhile

Changing Environment Attacks

- all protocols rely on some assumptions about the environment (e.g., cryptographic keys cannot be broken)

- when it got expensive, some people bought cheaper monthly tickets for a suburban station and a nearby one, and one for the destination and a nearby one
- a large investment later all barriers were automatic and tickets could record state

Changing Environment Attacks

- all protocols rely on some assumptions about the environment (e.g., cryptographic keys cannot be broken)

- but suddenly the environment changed: rail transport got privatised creating many competing companies potentially cheating each other
- revenue from monthly tickets was distributed according to a formula involving where the ticket was bought...

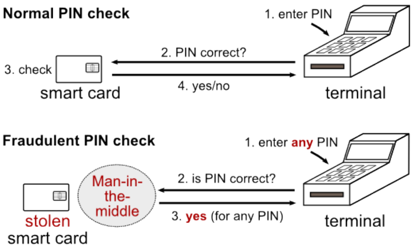
Changing Environment Attacks

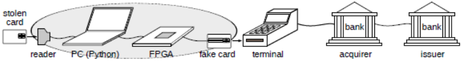
- all protocols rely on some assumptions about the environment (e.g., cryptographic keys cannot be broken)

- apart from bad outsiders (passengers), you also have bad insiders (rail companies)
- chaos and litigation ensued

A Man-in-the-middle attack in real life:

- the card only says yes or no to the terminal if the PIN is correct
- trick the card in thinking transaction is verified by signature
- trick the terminal in thinking the transaction was verified by PIN





Problems with EMV

- it is a wrapper for many protocols
- specification by consensus (resulted unmanageable complexity)
- its specification is 700 pages in English plus 2000+ pages for testing, additionally some further parts are secret
- other attacks have been found
- one solution might be to require always online verification of the PIN with the bank

Problems with WEP (Wifi)

- a standard ratified in 1999
- the protocol was designed by a committee not including cryptographers
- it used the RC4 encryption algorithm which is a stream cipher requiring a unique nonce
- WEP did not allocate enough bits for the nonce
- for authenticating packets it used CRC checksum which can be easily broken
- the network password was used to directly encrypt packages (instead of a key negotiation protocol)

- encryption was turned off by default

Protocols are Difficult

- even the systems designed by experts regularly fail
- try to make everything explicit (you need to authenticate all data you might rely on)
- the one who can fix a system should also be liable for the losses
- cryptography is often not **the** answer

logic is one way protocols are studied in academia (you can use computers to search for attacks)

Schneier: Step 1

What assets are you trying to protect?

This question might seem basic, but a surprising number of people never ask it. The question involves understanding the scope of the problem. For example, securing an airplane, an airport, commercial aviation, the transportation system, and a nation against terrorism are all different security problems, and require different solutions.

Schneier: Step 1

What assets are you trying to protect?

This question might seem basic, but a surprising number of people never ask it. The question involves understanding the scope of the problem. For example, securing an airplane, an airport, commercial aviation, the transportation system, and a nation against terrorism are all different security problems, and require different solutions.

You like to prevent: "It would be terrible if this sort of attack ever happens; we need to do everything in our power to prevent it."

Schneier: Step 2

What are the risks to these assets?

Here we consider the need for security. Answering it involves understanding what is being defended, what the consequences are if it is successfully attacked, who wants to attack it, how they might attack it, and why.

Schneier: Step 3

How well does the security solution mitigate those risks?

Another seemingly obvious question, but one that is frequently ignored. If the security solution doesn't solve the problem, it's no good. This is not as simple as looking at the security solution and seeing how well it works. It involves looking at how the security solution interacts with everything around it, evaluating both its operation and its failures.

Schneier: Step 4

What other risks does the security solution cause?

This question addresses what might be called the problem of unintended consequences. Security solutions have ripple effects, and most cause new security problems. The trick is to understand the new problems and make sure they are smaller than the old ones.

Schneier: Step 5

What costs and trade-offs does the security solution impose?

Every security system has costs and requires trade-offs. Most security costs money, sometimes substantial amounts; but other trade-offs may be more important, ranging from matters of convenience and comfort to issues involving basic freedoms like privacy. Understanding these trade-offs is essential.