

UNIX security



- Ulf Larson (modified by Erland Jonsson/Magnus Almgren)
- Computer security group
- Dept. of Computer Science and Engineering
- Chalmers University of Technology, Sweden

Outline

- UNIX security ideas
- Users and groups
- File protection
- Setting temporary privileges
 - Permission bits
 - Program language components
- Examples

UNIX security ideas

- Memory protection for processes
 - Processes have own virtual address space
 - Communication with hardware is done through the operating system
- Files are protected between users
 - The “everything is a file” concept implies that same mechanisms apply for all objects
- Maintenance is carried out by a “reliable” system administrator
 - Also known as root, or superuser

Users and groups

- A user name is internally represented by a user identifier, or UID
 - Special user names are used for system functions, such as root, guest and apache
 - UNIX *id* command show UID
 - UIDs are stored in file */etc/passwd* with username, preferred shell
 - Your system privileges depends on UID
- Group id, or GID is used to identify *groups* of users
 - UNIX *groups* *<username>* shows the groups that *<username>* belongs to

Users and groups (2)

- /etc/passwd file entry for user root:
 - root:AAencryptedpw:0:0:root:/root:/bin/bash
- Special user names
 - UNIX comes with special users for administrative purposes: the superuser, or root.
 - As root you can log users out and in, shutdown the computer, start and run network services, run all programs, view all files for all users
 - As root: ***most security restrictions are bypassed.***
 - “Hacking root” provides an attacker with unrestricted privileges to a system...**BAD!**

Users and groups (3)

- Sometimes a user need to perform actions as *another* user
 - UNIX **su command** (substitute user / switch user)
 - User enter username and *password* for account. User *becomes* the other user until log out
 - UNIX **sudo command**
 - Run a **single** command usually limited for root (perm. in sudoers file)
 - Users enter their own password (typically) and the access is logged.
 - Executable **files with SUID bit** set
 - Operating system lets user perform the desired operation with the privileges of the **owner** of the object. When execution finished, user assumes ordinary privileges.
 - Using the **setuid()** function call

Users and groups (4)

- Real and effective UIDs
 - Each user has at any given point in time two (sometimes three) different UIDs
 - Real UID, or **RUID** is assigned to user when logging in. Used to identify unique user and *remain unchanged*
 - Effective UID, or **EUID** is initially same as RUID, but changes to *owner* of file during execution of files with the SUID flag set (SUID files).
EUID changes back to RUID after execution

File Protection

- UNIX file system controls which users can access what items and how
- Simply put: *Everything visible to a user can be represented as a “file”*
 - Each “file” has at least one name, an owner and access rights
 - Running UNIX **ls** command reveals information about files and directories

File Protection (2): Example

```
>>ls -l /home/ulf/example.txt
```

```
-rw-r--r- 1 ulf ulfgrp 1024 Sep 1 11:00 example.txt
```

-	file type
rw-r--r--	file permissions (owner, group, other)
1	# names of the file
ulf	owner
ulfgrp	group
1024	file size
Sep 1 11:00	modification date and time
example.txt	name

File Protection (3)

- File permissions indicate ***who*** that can do ***what*** on a specified object.
- 9 characters grouped in 3 *classes* and 3 *kinds* of permissions
- Classes:

Owner	=	The file's owner
Group	=	Users in the file's group
Other	=	Everybody else (except the superuser)
- Kinds:

r	=	Class has <i>read</i> access to file,
w	=	Class has <i>write</i> access to file,
x	=	Class has <i>execute</i> access to file

File Protection (4)

- Example:
 - Who can access file a.txt, and in what way:
 - `rwX r-- --- usrOne grpTwo a.txt`

Answer:

usrOne has read, write and execute access to a.txt

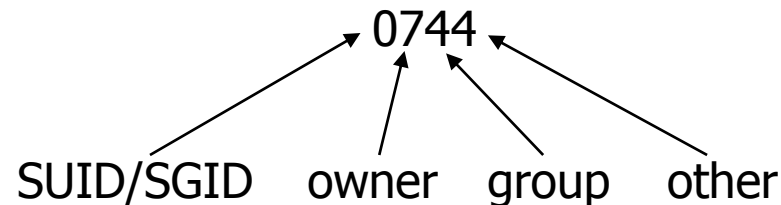
grpTwo has read access to a.txt

other has *no* access to a.txt

(superuser has full access to a.txt)

File Protection (5)

- UNIX **chmod** command is used to change file access permissions – 2 different modes
 - Octal file permissions
 - Four octal numbers are used as follows:



When calculating: r adds 4, w adds 2 and x adds 1 to total.

Example: What is the result (in octal) of setting r,w,x for owner, r for group and r for other for non SUID file?

File Protection (6)

- Combining kinds r, w and x and s with '+', '=' and '-' and classes u, g and o
 - To add write permissions for group: g+w
 - To remove read permissions for other: o-r
 - To set read access for user: u=r
- Example:

Assuming file.txt has permissions 0744, the following two operations achieve the same result

 - >> chmod 0764 file.txt
 - >> chmod g+w file.txt

Setting temporary permissions: SUID program

- A **SUID program** is a program for which the “s” bit is set
- Used to grant temporary privileges during execution to unprivileged user
 - Example: change the /etc/passwd file
 - What programs are SUID on your system, run
find / -perm -4000 -print
- There are two main methods for changing the s flag through the use of permission bits and **chmod**
 - SUID bits in file permission.
 - SUID = `chmod 4755 file.txt`, or `chmod u+s file`
 - Result: **rws** r-x r-x

Setting temporary permissions: SUID example

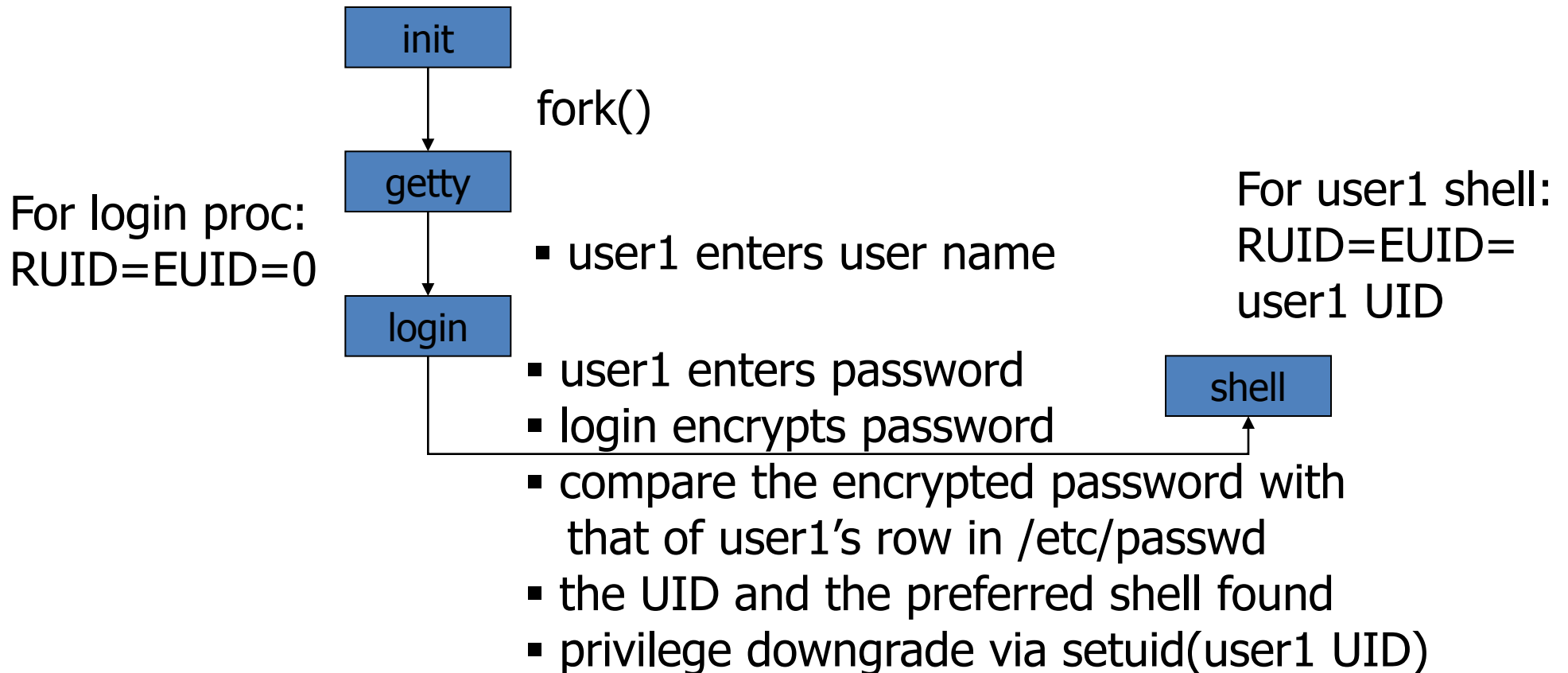
- Impact on RUID and EUID from the use of SUID
 - Repeat slide “Users and Groups (4)”
 - During execution of a SUID file, EUID changes to that of the *owner* of the SUID file. The RUID does **not** change.

Setting temporary permissions:

`setuid fcn call`

- The `setuid()` function call
 - `>>man setuid` (for help – used in Lab 1)
 - Changes the UID of the user to that of the argument of the function call
 - Non-root users can only `setuid` to their own UID.
 - If the caller of the `setuid()` function is non-root, then EUID is changed.
 - If the caller of the `setuid()` function is root, then RUID and EUID is changed. This is used by root to *downgrade* privileges for a user after the user has logged in.

Example: UNIX login



Summarizing example 1

- Example 1

- User Alice logs in to run the SUID file
/home/ulf/becomeMe.exe owned by
ulf (UID 12345)

```
>>ls -l becomeMe.exe
```

```
rwsr-xr-x ulf ce becomeMe.exe
```

If user Alice has UID=22448, what are the RUID and EUID
before, during and after execution of the file `becomeMe.exe`?

Summarizing example (2)

- Example 2

- User Alice logs in to run the file

- `/home/ulf/dontbecomeMe.exe`

- owned by ulf (UID 12345)**

- `>>ls -l dontbecomeMe.exe`

- `rwxr-xr-x ulf ce dontbecomeMe.exe`

If Alice has the UID 22448, what are the RUID and EUID before, during and after execution of file `dontbecomeMe.exe`?

Solutions to examples

- Example 1
 - Before RUID=EUID=22448
 - During RUID=22448, EUID=12345
 - After RUID=EUID=22448
- Example 2
 - Before RUID=EUID=22448
 - During RUID=EUID=22448
 - After RUID=EUID=22448

Computer Security

Passwords



Erland Jonsson

Department of Computer Science and Engineering

Chalmers University of Technology, Sweden

Bad passwords

- Names (own, wife, child, dog, colleague, car, mistress, etc)
- Numbers that can be related to you (telephone-, car-, birth) or “well-known” numbers, such as e , p , Planck’s constant,....
- Based on any other info that can easily be related to you
- “Popwords” (wizard, gandalf, guatama,...)
- word in dictionary or encyclopedia (Swedish, English, Japanese,...)
- special patters (qwertyui,...)
- none of the above backwards!
- none of the above slightly modified! (i.e. +number, with one big letter, etc)

Good passwords (or at least better!)

1. with small and capital characters
 2. with numbers and special characters
 3. with at least 8 characters (for UNIX)
 4. could be typed easily
-
- 1)-3) to avoid exhaustive search
 - 4) to avoid shoulder surfing
 - preferably: random, ***but: hard to memorize in that case***
 - ***Password manager?***

Password Rules

- never reveal your password to anyone!
- do not write it down (in any interpretable way)!
- change it regularly (or at least every now and then...)!
- could be typed fast!
- memorizing rules:
 - first characters of words in a sentence (Ex. “tiaWcics”)
 - combine two short words + extra character:
(Ex. “end(pagE”)
 - the way to work/auntie Ann/... (Ex. GOPAJOle)

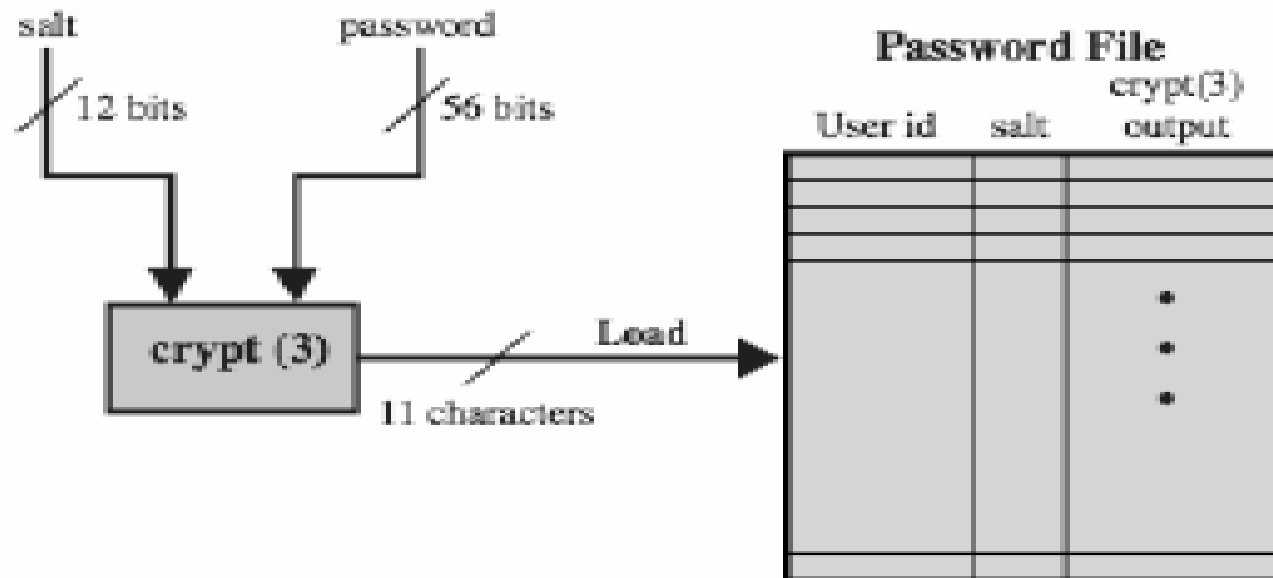
Password Attacks

- There are three different ways to attack a password:

FIND / GUESS / CRACK

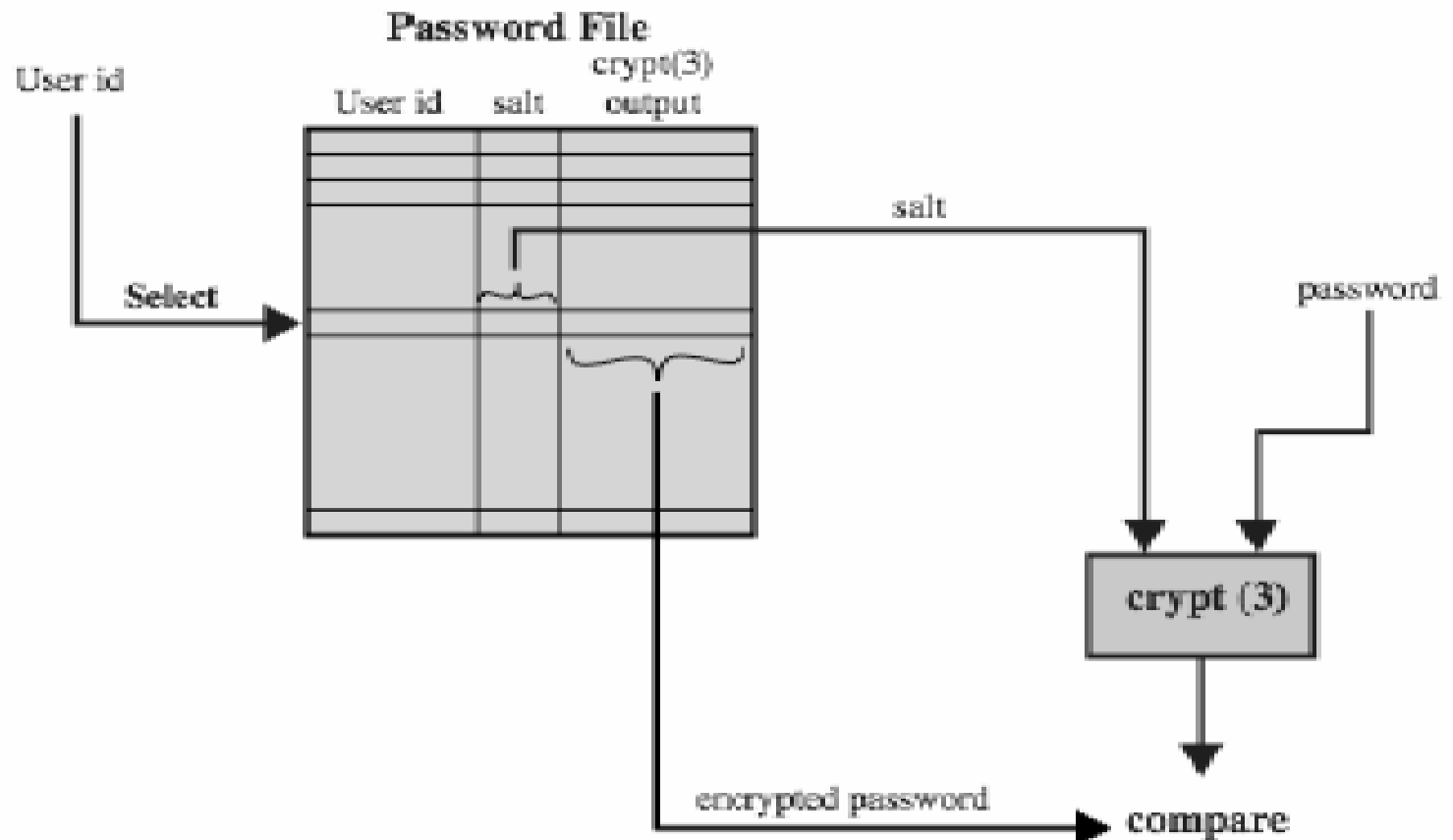
- **Find**: find note, eavesdrop, keyboard snooping, shoulder surfing, asking for it(!)
 - **Guess**: try “probable” cases, “Joe accounts”
 - **Crack**: exhaustive search, dictionary attacks
- Example: the UNIX salt feature:
 - prevents 2 users with the same passwords from knowing it
 - makes exhaustive search for multiple password computationally more expensive

Introducing a new password



(a) Loading a new password

Verifying a password



(b) Verifying a password

UNIX implementation

- original scheme
 - 8 character password form 56-bit key
 - 12-bit salt used to modify DES encryption into a one-way hash function
 - 0 value repeatedly encrypted 25 times
 - output translated to 11 character sequence
- now regarded as woefully insecure
 - e.g. supercomputer, 50 million tests, 80 min
- sometimes still used for compatibility

Improved implementation

- have other, stronger, hash/salt variants
- many systems now use MD5
 - with 48-bit salt
 - password length is unlimited
 - is hashed with 1000 times inner loop
 - produces 128-bit hash
- OpenBSD uses Blowfish block cipher based hash algorithm called Bcrypt
 - uses 128-bit salt to create 192-bit hash value

One-time passwords

- A one-time password is a password that is valid only once
 - Thus, it is resistant to eavesdropping and wire-tapping.
 - One-time passwords can be implemented using special **password generators** (time-dependent passwords, dynamic password generation) or simply as a **list of passwords**.
 - A special type of one-time passwords are those generated by a **challenge-response system**.
 - In this case the system **generates a challenge** (seed, nonce), which is different each time and the **user calculates a response** (=the password) using the challenge. Thus, the password will change every time and can not be re-used.
 - In this case, the secret is the **function** that translates the challenge to the response (or: see book Fig 3.11)

Password guessing/cracking (Bruce Schneier)

- **Password Recovery Toolkit (PRTK)** from Access Data
- Password security depends on:
 1. if you can **slow down the password** testing (in the SW)
 2. **the order of guessing** by the program
- Guesses 350,000 passwords/s (Microsoft OpSys)
- A typical password consists of a root + appendage
- Appendage is a suffix (90%) or prefix (10%)
- PRTK guessing procedure:
 1. dictionary of 1,000 pws (e.g. letmein, 123456, etc)
 2. adds 100 common suffixes (e.g 1, 4u, 69, etc)
- ➔ 24 % of all passwords!

Password guessing/cracking cnt'd (Bruce Schneier)

- Exhaustive search of all 4 character strings:
1) all lowercase, 2) initial uppercase, 3) uppercase
- All with common substitutions (@ for a, 1 for l, etc)
- Collects personal info plus other passwords, which greatly reduces search time
- **Conclusion: Good passwords are those not found by PRTK**
- Forensic Toolkit:
 - **scans harddrive for printable strings** to create dictionary
→ 50% of passwords
 - Windows opsys leaves data (**residues**) all over the place. May be permanently stored on the hard disc!
 - Thus, use opsys insecurity instead of guessing

Computer Security

Authentication and Access Control



Erland Jonsson

Department of Computer Science and Engineering

Chalmers University of Technology, Sweden

User authentication

Authentication – definition

- Authentication is *verifying a user's identity*
- cp: message authentication: is check of message authenticity (Sw. äkthet) and source
- In an OS each account has one **identifier** (e.g. username) and one **authenticator** (e.g. password)
- The identifier tells *who you are*.
- The authenticator *verifies that this is true, i.e. it provides a secure coupling between the user and his account*

User Authentication

- fundamental security building block
 - basis of access control & user accountability
- is the process of verifying an identity claimed by or for a system entity
- has two steps:
 - **identification** - specify identifier
 - **verification** - bind entity (person) and identifier
- distinct from message authentication

Authentication procedure

The **authentication procedure** consists of 4 stages:

1. **identification** of the user (who is it?)
2. provision of some kind of **authentication information**, which is secret and unforgeable.
3. **transmission of the authentication information** to the system through a secure channel.
4. **validation** of the authentication information wrt some reference information (proof of correctness)

Problems (errors, attacks) can occur in all those 4 stages

Authentication information

The authentication information can be of **3 different, generic types**, based on something that is unique for the user:

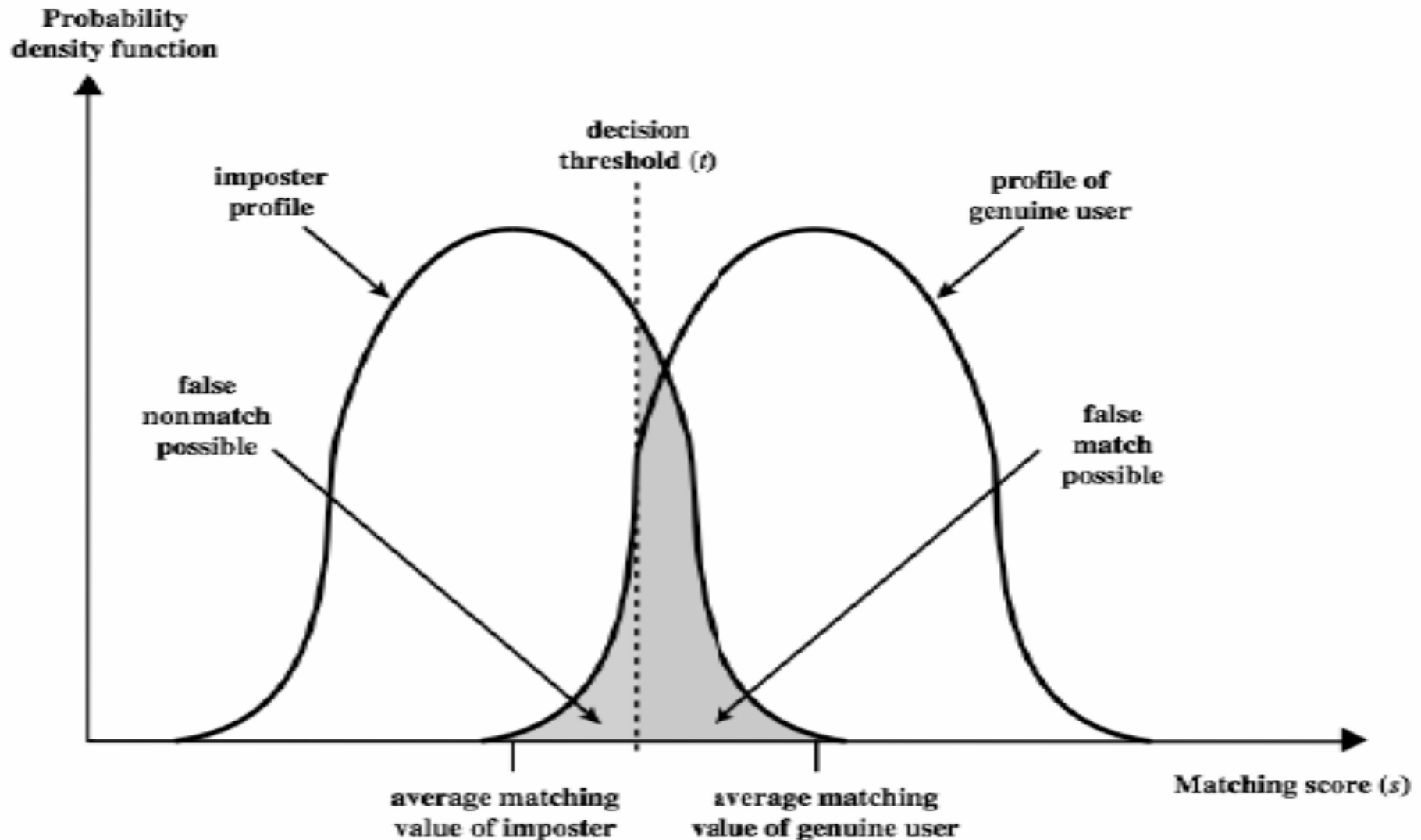
- something you **KNOW** (e.g password, PIN code)
- something you **HAVE** (e.g smartcard)
- something you **ARE (DO)** (e.g fingerprint), (biometrical methods, something characteristic about you)

(**WHERE** you are can also be used in some situations)

In general, something that you *have* is called a **token**. i.e. something that is used for authentication

A **capability** is an **unforgeable token** that gives the possessor certain **rights** (to an object) - **authorization**

Biometric accuracy – threshold selection



The transmission channel

- The transmission channel is **often the weakest link**, especially when long distances are involved
- The transmission channel may be very short and still be vulnerable
- The “usual” transmission threats and problems apply, such as:
 - eavesdropping
 - manipulation of routers, gateways
 - replay attacks
- Consequently, the “usual” remedies also apply



Roll over image to zoom in

KeyGrabber USB KeyLogger 8MB Black

by DataLogger
★★★★★ 13 customer reviews

Price: \$47.90 & FREE Shipping. Details

In Stock.
Sold by Vengeance Gaming and Fulfilled by Amazon. Gift-wrap available.

Want it tomorrow, Jan. 28? Order within 8 hrs 11 mins and choose One-Day Shipping at checkout. Details

- Saves Over 4000 Pages Of Text!
- 100% Undetectable
- Works Instantly! No Installation Needed!

4 new from \$47.85

Qty: 1

☐ Yes, I want FREE Two-Day Shipping with Amazon Prime

Add to Cart

or

Sign in to turn on 1-Click ordering.

Add to Wish List

More buying choices

Best Niche Product Add to Cart

\$47.85 + \$4.99 shipping

4 new from \$47.85

Have one to sell?

Sell on Amazon

Share

Customers viewing this page may be interested in these sponsored links (What's this?)

- Spector Pro Key Logger PC Magazine Editors' Choice for Key Logger software. 40% Off
www.spectorsoft.com/
- USB sticka med tryck USB sticka med tryck, 3dgr leverans från 25 ex, 10 års fabriksgaranti
www.attentionmedia.se/
- USB från ELFA USB direkt från lagret. 24 timmars leverans. Beställ nu!
www.elfa.se/

See a problem with these advertisements? Let us know

Frequently Bought Together



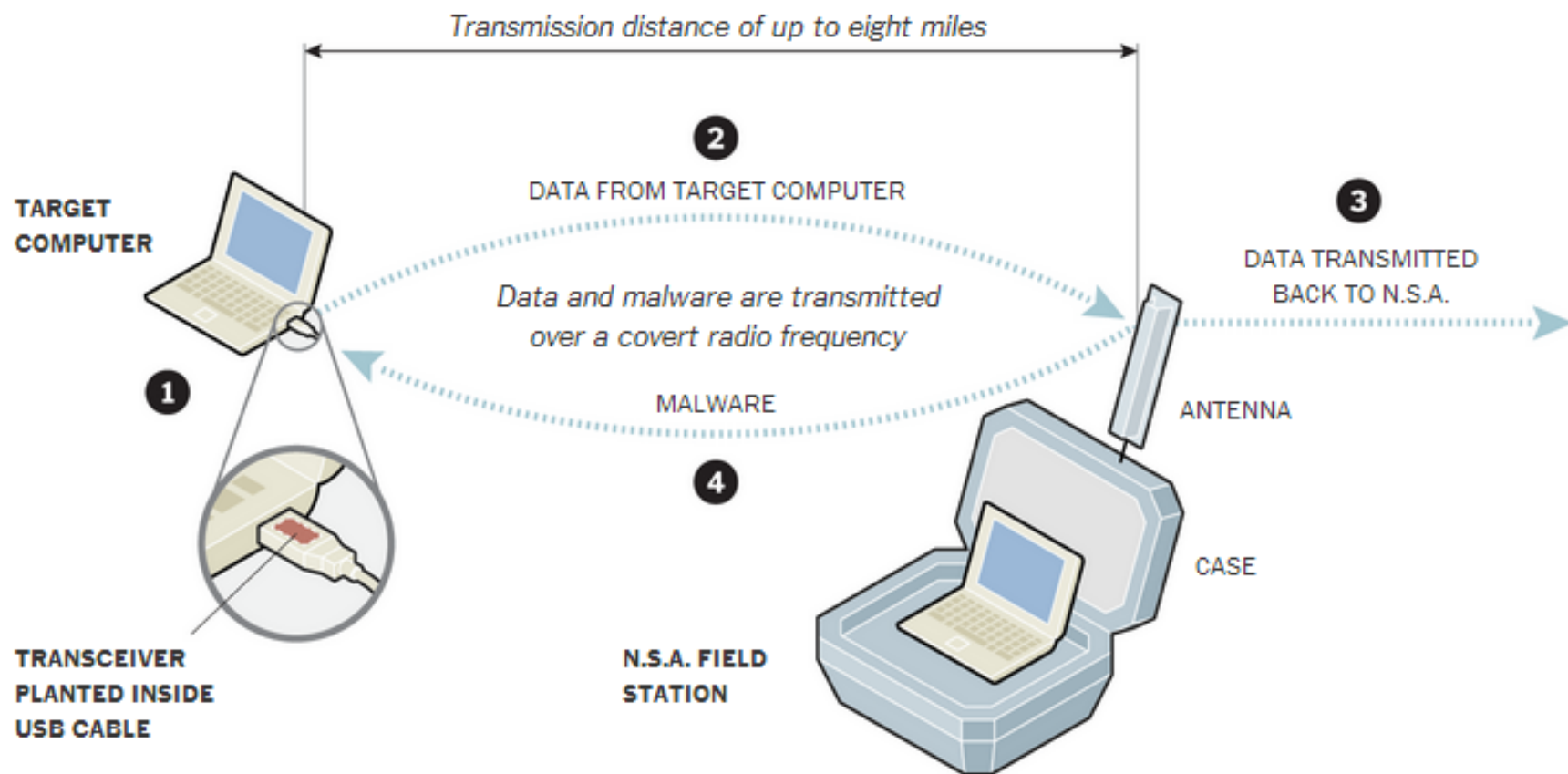
Price for all three: \$127.84

Add all three to Cart Add all three to Wish List

Show availability and shipping details

How the N.S.A. Uses Radio Frequencies to Penetrate Computers

The N.S.A. and the Pentagon's Cyber Command have implanted nearly 100,000 "computer network exploits" around the world, but the hardest problem is getting inside machines isolated from outside communications.



1. Tiny transceivers are built into USB plugs and inserted into target computers. Small circuit boards may be placed in the computers themselves.

2. The transceivers communicate with a briefcase-size N.S.A. field station, or hidden relay station, up to eight miles away.

3. The field station communicates back to the N.S.A.'s Remote Operations Center.

4. It can also transmit malware, including the kind used in attacks against Iran's nuclear facilities.

Validation of authentication

- The system must have some kind of **reference information** in order to **validate** the authentication information
- An **attack** can be launched **against the reference info**, e.g.:
 - read stored password
 - change the reference info
- Protection of password reference info:
 - a) store in a file with strong and limited Access Control
 - b) Encryption
 - c) (a + b)
- Pros and Cons:
 - a) - cleartext storage and comparison is in cleartext
- back-up tapes, memory dumps reveals password
 - b) + could be stored in readable files (?)
- open for brute force attacks

Access Control

Access Control

- Definition of Access Control:

The prevention of unauthorized use of a resource

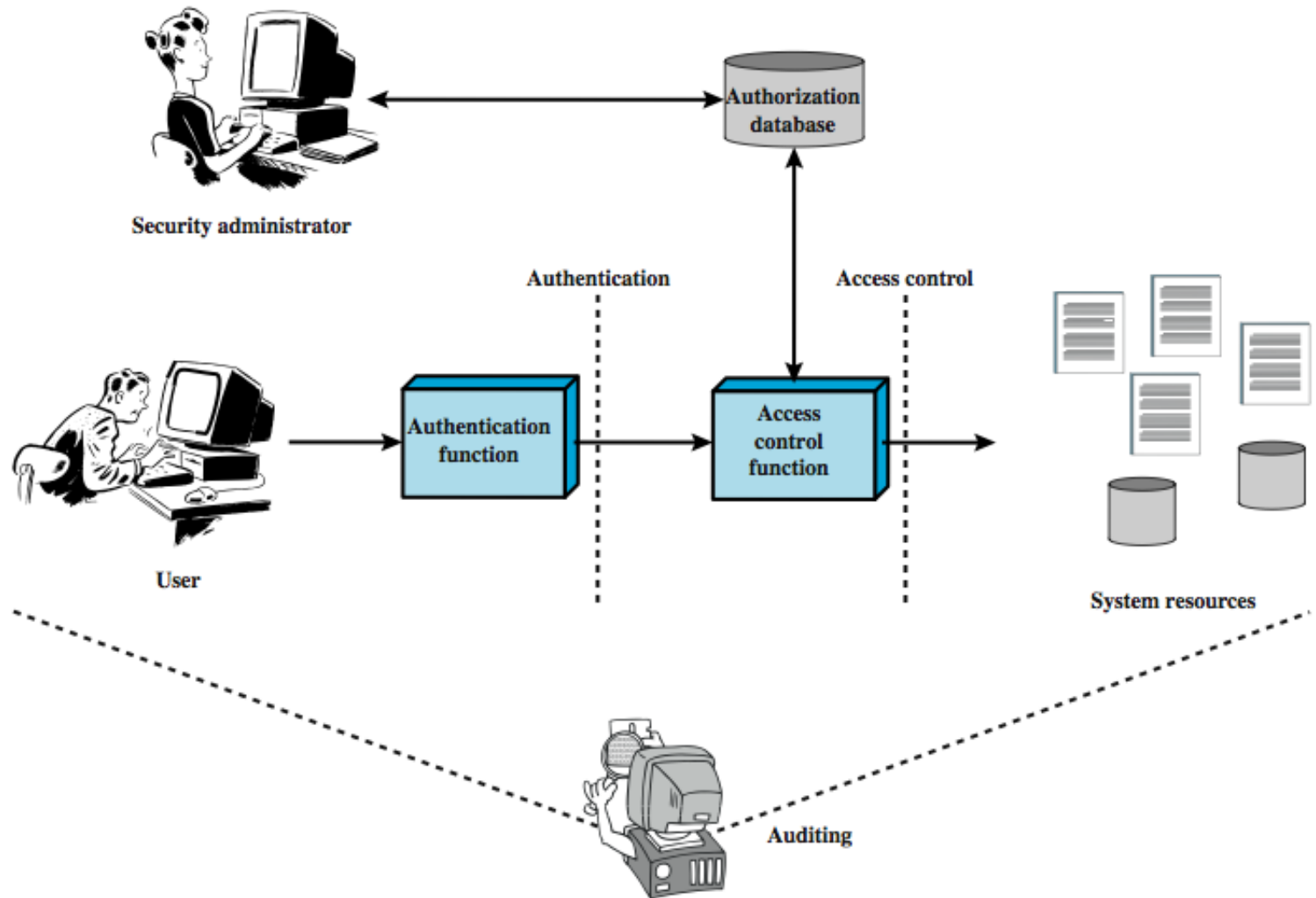
(including the prevention of use of a resource in an unauthorized manner)

- central element of computer security, used for boundary protection
- access control permits users and groups
 - to authenticate to system
 - to be assigned access rights to certain resources in the system i.e. authorized

Access Control Elements

- **subject** - entity that can access objects
 - a process representing user/application
 - often have 3 classes: owner, group, world
- **object** - access controlled resource
 - e.g. files, directories, records, programs etc
 - number/type depend on environment
- **access right** - way in which subject accesses an object
 - e.g. read, write, execute, delete, create, search

Access Control Usage



Access Control

- provided using an **access control matrix**
 - **lists of subjects** in one dimension (rows)
 - **lists of objects** in the other dimension (columns)
 - each entry specifies access rights of the specified subject to that object
- access control matrix is often sparse
- can decompose by either column, leading to an **access control list** (ACL) or by row, leading to **capability tickets**

Access Control Matrix

		OBJECTS								
		subjects			files		processes		disk drives	
		S ₁	S ₂	S ₃	F ₁	F ₁	P ₁	P ₂	D ₁	D ₂
		SUBJECTS								
	S ₁	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
	S ₂		control		write *	execute			owner	seek *
	S ₃			control		write	stop			

* - copy flag set

Access Control 2

- The **access control list** provides a list of subjects, who can access a single object (one list “per file” or object)¹
- The **capability ticket approach** presents a list of objects accessible by a single subject (one list “per user” or subject)¹
- A **capability ticket** is an unforgeable token that gives the possessor certain rights to an object, i.e. it **specifies the authorization** for a particular user

1. See book fig. 4.3

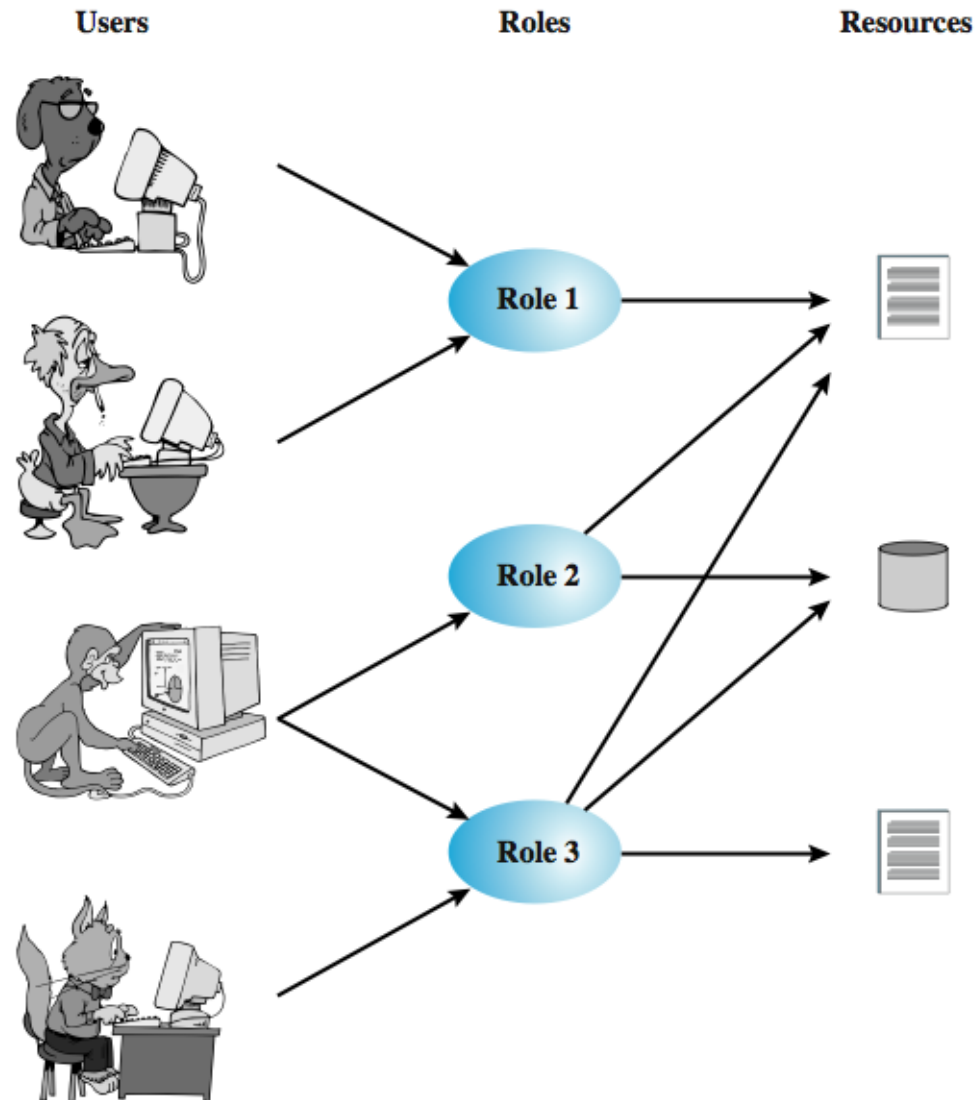
Mandatory and Discretionary Access Control

- **MANDATORY ACCESS CONTROL (MAC)** means that some central authority (e.g. the security officer) determines what information is accessible to whom
- **DISCRETIONARY ACCESS CONTROL (DAC)** means that the owner of the file (i.e. the user) determines what information is accessible to whom
- MAC and DAC can both be applied at the same time
- MAC is most commonly used in the multi-level security mechanism (MLS) in the Military Security Policy
- DAC is used in many operating systems, e.g. UNIX.

Role-Based Access Control

- In **ROLE-BASED ACCESS CONTROL (RBAC)** the rights are assigned to roles rather than to the users.
 - *For example in a hospital:* surgeon, medical practitioner, nurse, janitor, etc
- RBAC employs MAC and has been developed to meet the needs from commercial and societal systems.
 - Procedure:
identification - authentication - selection of role - access to information (according to role).
- Advantages:
 - easy to enforce enterprise-specific security policies
 - security management is simplified
- Other policies exist, e.g Team-Based Access Control, etc

Role-Based Access Control



Role-Based Access Control

User to Role:

	R_1	R_2	...	R_n
U_1	×			
U_2	×			
U_3		×		×
U_4				×
U_5				×
U_6				×
•				
•				
•				
U_m	×			

Role to Access
Right:

		OBJECTS								
		R ₁	R ₂	R _n	F ₁	F ₁	P ₁	P ₂	D ₁	D ₂
ROLES	R ₁	control	owner	owner control	read *	read owner	wakeup	wakeup	seek	owner
	R ₂		control		write *	execute			owner	seek *
	•									
	•									
	R _n			control		write	stop			

Smartphone Malware

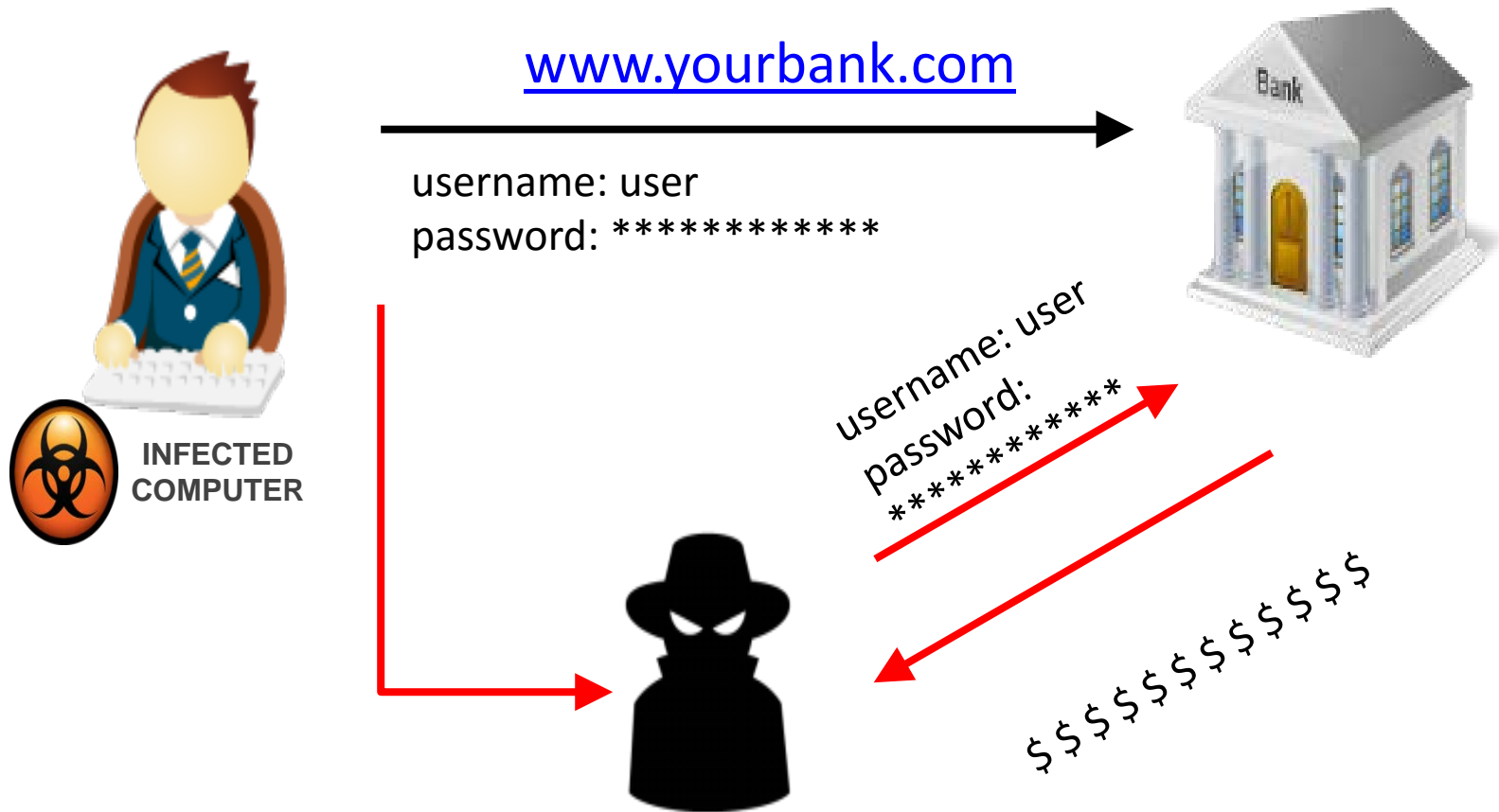
Slides by Federico Maggi

<http://maggi.cc/>

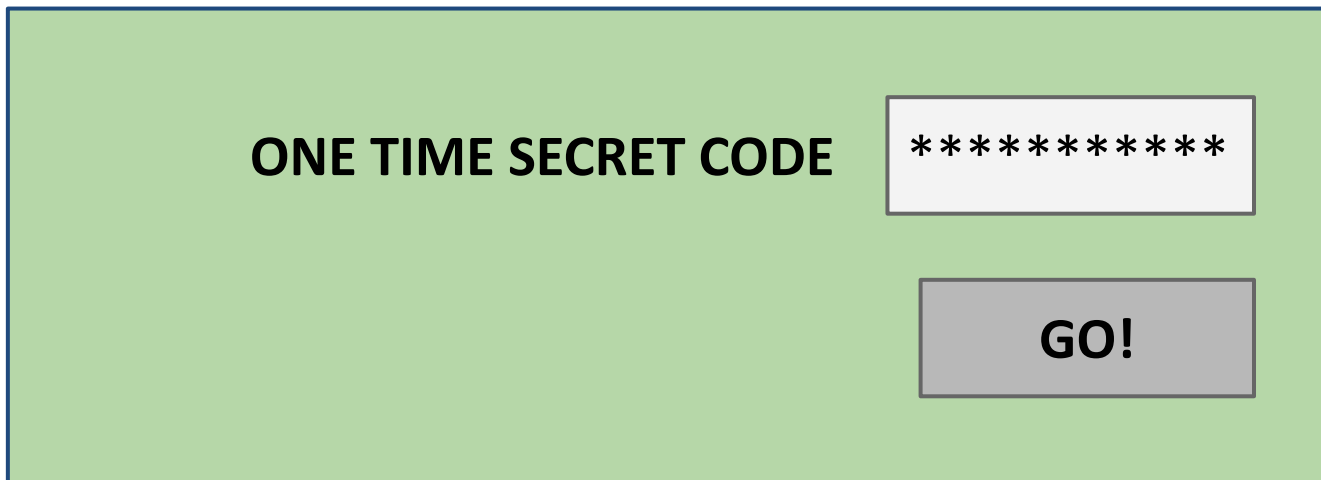
ZitMo & SpitMo (2011)

- *Companion* of the famous Zeus and SpyEye trojans.
- Steal the *mTAN* or *SMS* used for 2-factor authentication.

The attack scheme (1)



2-factors authentication (password + secret code)



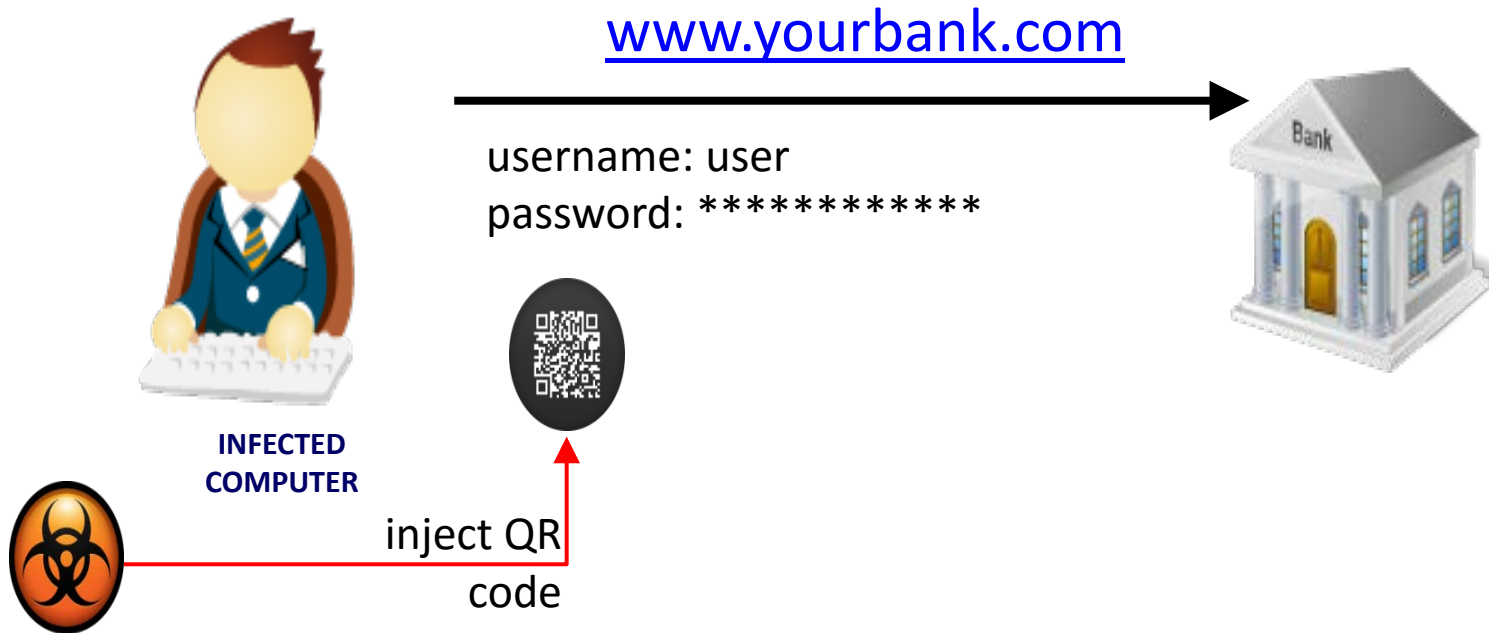
ONE TIME SECRET CODE

GO!

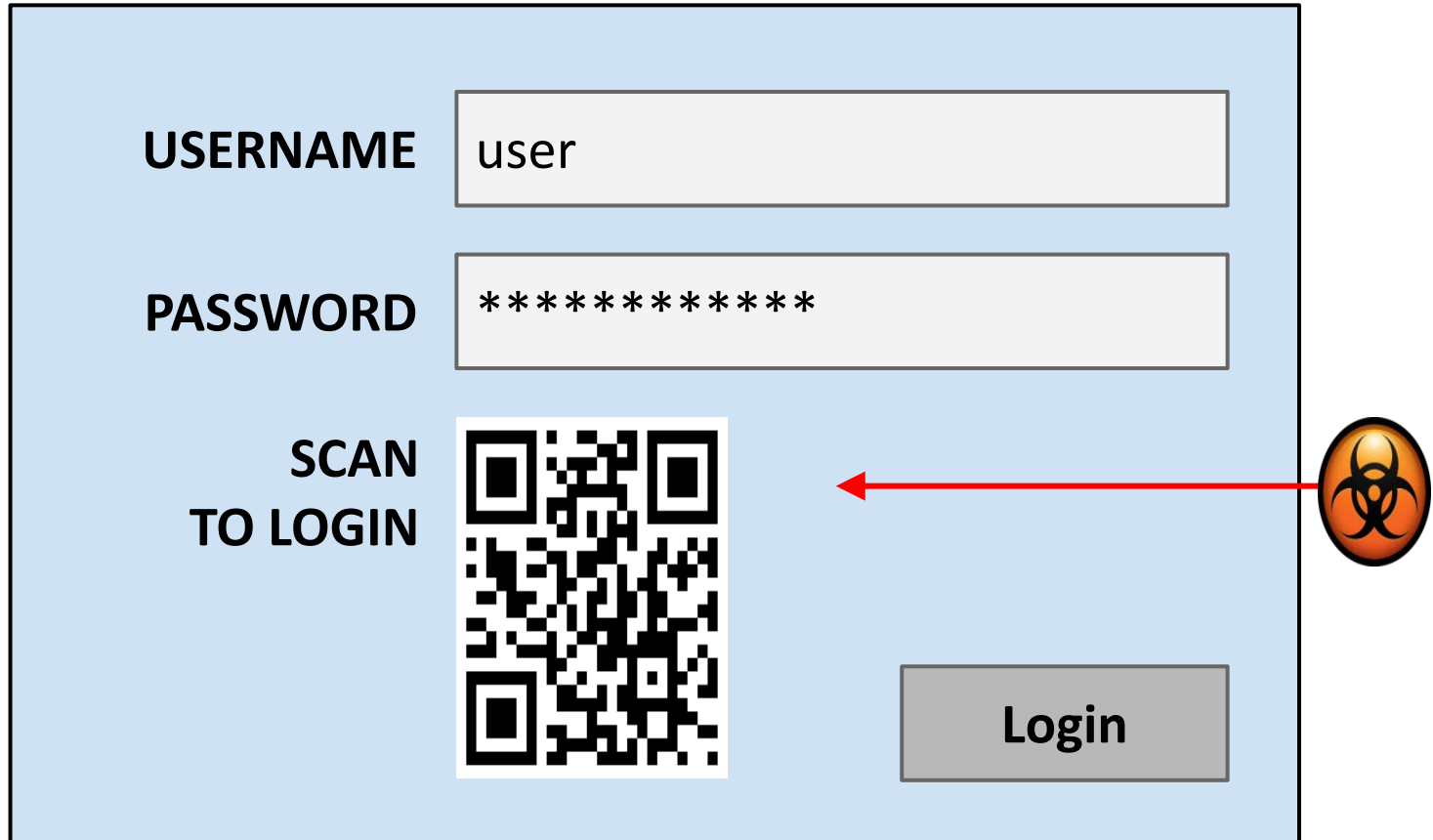
The attack scheme (2)



The attack scheme (2)



Luring Users with a QR Code



The image shows a login interface on a light blue background. It includes a 'USERNAME' field with the text 'user', a 'PASSWORD' field with ten asterisks, and a 'SCAN TO LOGIN' label next to a QR code. A 'Login' button is at the bottom right. A red arrow points from a biohazard icon on the right towards the QR code, indicating a warning or danger.

USERNAME user

PASSWORD *****

SCAN TO LOGIN

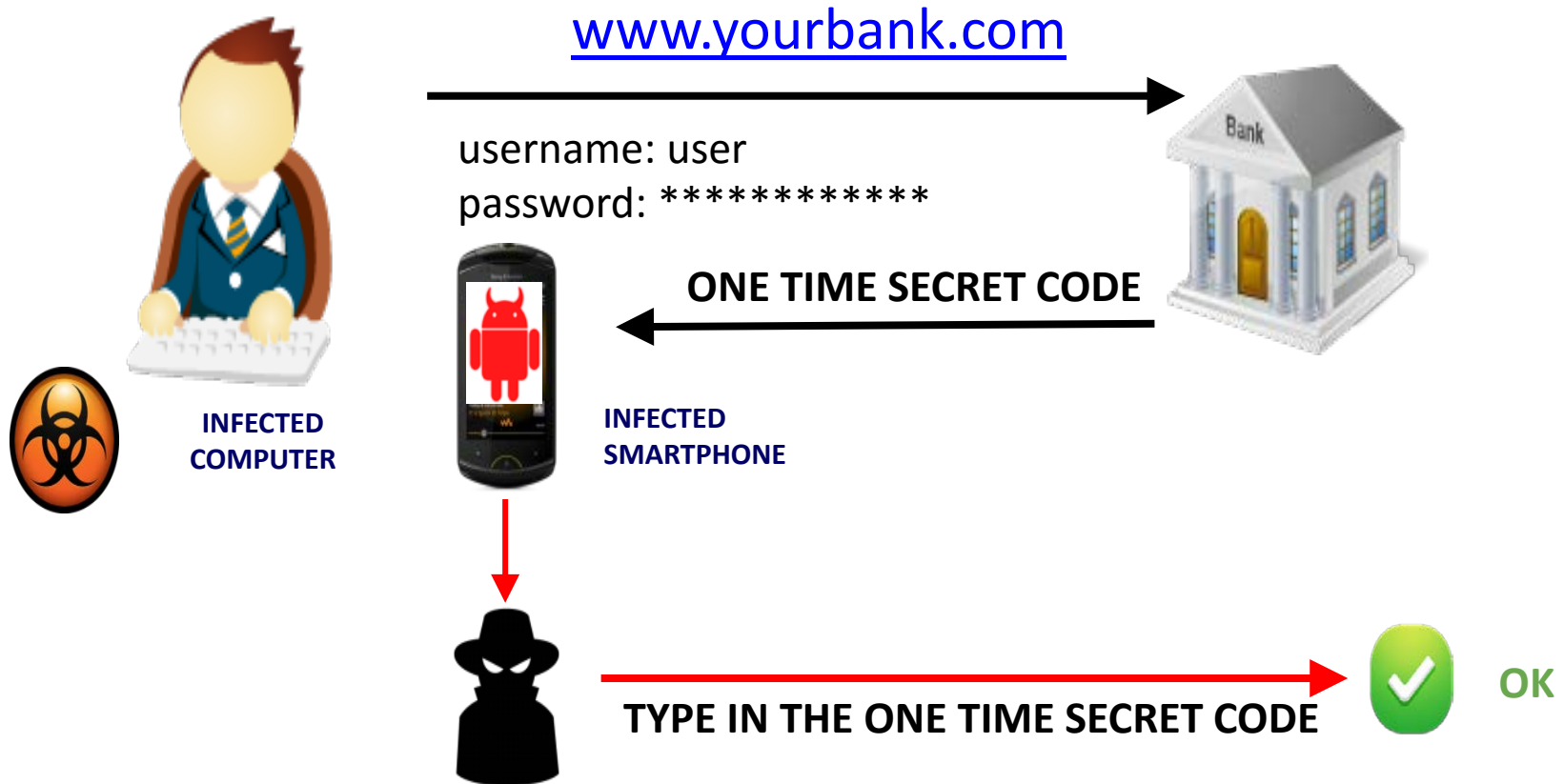
Login

The attack scheme (3)

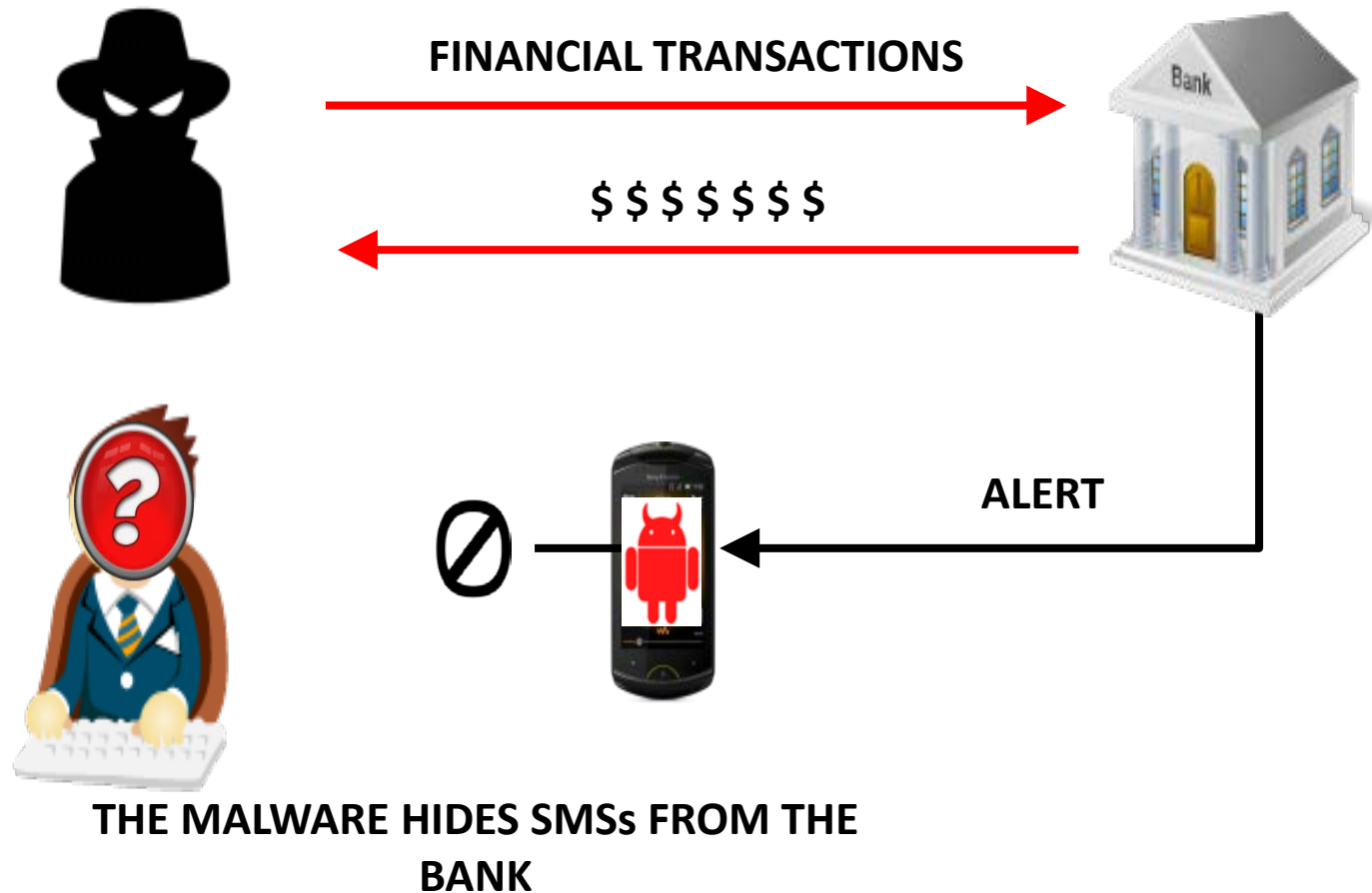
www.evil.org/fake-login-app.apk



The attack scheme (4)



The attack scheme (5)



More information

- http://www.kaspersky.com/about/news/virus/2011/Teamwork_How_the_ZitMo_Trojan_By_passes_Online_Banking_Security