

11 – Security

What is computer security?

As computers are more and more interconnected, the problem of keeping them *secure* becomes more and more complex

... but what does *security* mean, in this field?

■ The C.I.A. Paradigm

- Confidentiality (my data are unknown to others, if they are not authorized to access them)
- Integrity (data remain the same if I don't change them)
- Availability (if I am authorized to access some data, they are accessible in a convenient format and in a reasonable time)

What is computer security?

- The main concepts of the C.I.A. paradigm are strongly connected with the following ones:
 - Identification (who do you say you are?)
 - Authentication (how do I know it is really you?)
 - Authorization (what are you allowed to do?)
 - Accountability (who did what?)

Università della Svizzera italiana	Facoltà di scienze della comunicazione	I
		4

... what does it mean?

- How can we translate these concept into real life terms?
 - Confidentiality
 - my email account is safe, nobody else can read it
 - Integrity
 - my blog is safe, no unauthorized people can delete my posts
 - Availability
 - I can always connect to the services I rely on for my work or my leisure time

Università della Svizzera italiana	Facoltà di scienze della comunicazione	I
		5

Are we secure?

- The purpose of this lesson is to show you how much we are (not) secure
 - ... of course, I will depict things in a slightly negative way, however...

Example 1: email spoofing

- Play with your own SMTP email server
- ... and learn to understand who actually sent you an email (hint: do not trust the "From" field ;-))

```
MAIL FROM: whoever.youwant@wherever.youlike
250 2.1.0 Ok
RCPT TO: blablah@gmail.com
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: this is a spoofed message

Assistenza esami
Hi there,

I am not who you think I am
:-)

.
250 2.0.0 Ok: queued as D3E7840131
```

Example 2: sniffing data

- Always choose encrypted channels when possible, otherwise...

The screenshot shows the Wireshark interface with a list of captured packets. Packet 78 is highlighted in red, indicating it is selected. The packet list shows the following details:

No.	Time	Source	Destination	Protocol	Info
77	38.134048	192.168.1.101	88.149.156.170	TCP	37908 > http [ACK] Seq=1 Ack=1 Win=588
78	38.134251	192.168.1.101	88.149.156.170	HTTP	POST /main.php3 HTTP/1.1 (application/x-www-form-urlencoded)
79	38.187077	88.149.156.170	192.168.1.101	TCP	http > 37908 [ACK] Seq=1 Ack=698 Win=7
80	38.319071	88.149.156.170	192.168.1.101	HTTP	HTTP/1.1 200 OK (text/html)
81	38.319108	192.168.1.101	88.149.156.170	TCP	37908 > http [ACK] Seq=698 Ack=1163 Win=...

The detailed view of Frame 78 shows the following structure:

- Frame 78 (763 bytes on wire, 763 bytes captured)
- Ethernet II, Src: Intel_63:19:ba (00:16:6f:63:19:ba), Dst: Netgear_6d:d2:5a (00:14:6c:6d:d2:5a)
- Internet Protocol, Src: 192.168.1.101 (192.168.1.101), Dst: 88.149.156.170 (88.149.156.170)
- Transmission Control Protocol, Src Port: 37908 (37908), Dst Port: http (80), Seq: 1, Ack: 1, Len: 697
- Hypertext Transfer Protocol
- Line-based text data: application/x-www-form-urlencoded

The raw data of the POST request body is shown in hexadecimal and ASCII:

```

0220 00 70 02 02 03 01 03 0c 00 77 00 01 73 00 03 00  npbb3_3nwmj_k=
0230 20 70 68 70 62 62 33 5f 33 6e 6d 77 6a 5f 6b 3d  dde9cc68 fabd0699
0240 64 64 65 39 63 63 36 38 66 61 62 64 30 36 39 39  ; npbb3_3nwmj_s
0250 3b 20 70 68 70 62 62 33 5f 33 6e 6d 77 6a 5f 73  id=6elf3 50ba5233
0260 69 64 3d 36 65 31 66 33 35 30 62 61 35 32 33 33  f8125e0c e0d7a411
0270 66 38 31 32 35 65 30 63 65 30 64 37 61 34 31 31  eef..Content-Typ
0280 65 65 66 0d 0a 43 6f 6e 74 65 6e 74 2d 54 79 70  e: application/x
0290 65 3a 20 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78  -www-form-urlencoded..Content-Le
02a0 2d 77 77 77 2d 66 6f 72 6d 2d 75 72 6c 65 6e 63  ngth: 47 ...login=
02b0 6f 64 65 64 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 65  n=aittal am&pass=
02c0 6e 67 74 68 3a 20 34 37 0d 0a 0d 0a 6c 6f 67 69
02d0 6e 3d 61 69 74 74 61 6c 61 6d 26 70 61 73 73 3d
    
```

Example 3: SQL injection

- A rather old, but still valid technique (see [here](#) and [here](#))
- Avoid mistakes such as:

```
// get the login and pass and check if they are correct
$login  = $_REQUEST['login'];
$pass   = $_REQUEST['pass'];
$query  = "select * from user where login='$login' and pass='$pass'";
$result = mysql_query($query);
```

... as the result might be:

```
The query is: select * from user where login='test01' and pass="" or '1'
Welcome, user id 1
```


Example 4: code reversing

- Do not trust your code to be secure just because it is compiled.

The screenshot shows OllyDbg debugging Crackme1.exe. The assembly window displays instructions from 004011F7 to 0040127F, including PUSH, CALL, JMP, and CMP instructions. The registers window shows EAX at 0000541E and ESI at 00402183. A dialog box titled 'Good work!' with a yellow warning icon and the text 'Great work, mate! Now try the next CrackMe!' is overlaid on the assembly view. The taskbar at the bottom shows the Start button, taskbar icons for ollydbg and Crackme v1.0, and the system tray with the date 0.21.

Are we secure?

- ... from the examples we have seen we can learn some important lessons:
 - there is no “out of the box” secure solution, as security depends on many different factors
 - “security by obscurity” is NOT security
 - most of the times, security is just related to knowing (well!) how things work. The more you know the more chances you have to make your system secure... or at least know what you risk!

- Tools used for this lesson:
 - Wireshark: <http://www.wireshark.org/>
 - OlllyDBG: <http://www.ollydbg.de/>