Lab: Passwords

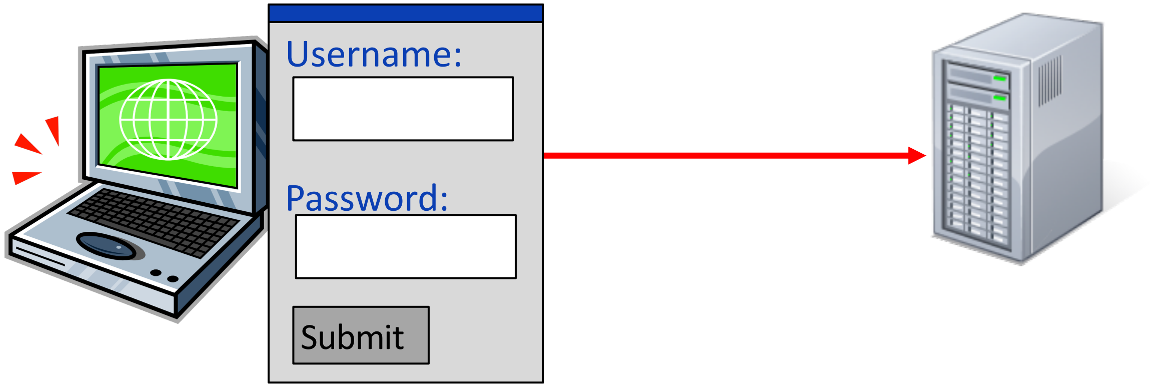
Mariano Ceccato

mariano.ceccato@univr.it

1. Offline password attack

Password-based authentication

- Each end-user is assigned a <username,password> pair
  - The *username* is used to identify the end-user (typically public)
  - The *password* is used to authenticate the end-user (private)



- A user is granted access only if the provided pair matches the one in the password database

Naïve implementation

- Username/password is stored in the r/w protected file

- Problem: an attacker who access this database would know all the passwords for all the end-users

- Solution: cypher a password before storing it in the database (do not store in in cleartext)

Implementation

- Instead of storing cyphered passwords, they are *hashed*

- **One-way**: it is difficult to obtain the cleartext starting from the hash
- **Fast**: hash values are easy to compute
- **No key** is needed to compute a hash value

```
<username, hash(password)>
```

Authentication scenario

- The end-user inserts the username
- The end-user inserts the password
- The typed password is hashed
- The hash value is compared to the has value in the database
- If the two values are the same, the user is authenticated

Operative system	Hash function
Microsoft windows	MD4
Unix	Des (Modified)
FreeBSD	Blowfish

Brute force attacks

- Idea: enumerate all the passwords and check them
  - Pro: effective and exhaustive
  - Con: inefficient with very long passwords

Dictionary attacks

- Automated attacks based on list of likely passwords
- It works when an end-user selects passwords from a small set of possible values
    - Words in a (set of) language(s)
    - Words that make sense for the end-users
    - Their variations

- Pro: efficient (fast)
- Con: might fail

Unix passwords

- Mainly two kinds of accounts:
  - *User* account: for normal end-users, with limited access to the system resources
  - *Superuser* accounts: used for administration purposes, they have access to all the system resources and privileges
    - E.g., administrator, admin, root

/etc/passwd

- This file is the password database
- Used to verify the password value typed at login

- It contains an entry for each user in the system
  - Username
  - Hashed password
  - UID: User Identification Number
  - GID: Group Identification Number
  - Full name (optional)
  - Home dir
  - Shell to be used at login

```
mariano:$1$UbS7$yJxgdFCyCbxAQ:1001:1001:Mariano Ceccato:/home/mariano:/bin/bash
```

Salting

- Salt: 12-bit string used to *perturbate* the password
  - 2^{12}=4096 different perturbations
  - It is selected at random when the password is created

- It is saved in cleartext in `/etc/password` before the password
- This makes attacks harder
  - For each word in a dictionary, there are 4096 possible hashes (precomputed values are useless)
  - Two users with the same passwords will have different salts and, so, different entries in `/etc/password`

Salting

```
mariano:$1$UbS7$yJxgdFCyCbxAQ:1001:1001:Mariano Ceccato:/home/mariano:/bin/bash
```

```
$id$salt$encrypted
```

ID	Digest method
1	MD5
2	Blowfish
5	SHA-256
6	SHA-512

Shadow password

- `/etc/passwd` is changed
  - "x" instead of the password (or a random string)

- `/etc/shadow` contains the actual cyphered password
- This file can be read only by an administrator

```
kali:x:1000:1000:Kali linux,,,:/home/kali:/bin/bash
```

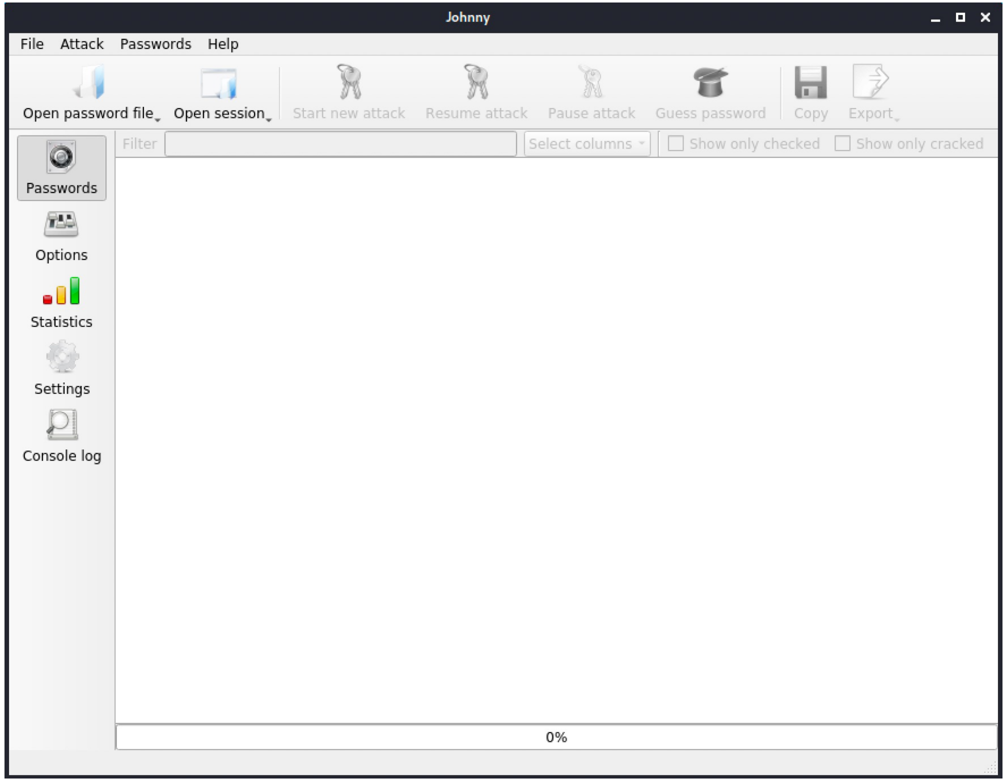
```
kali:$6$jLA.1OwWM1uGyWTJ$xMETR7yrEky/pfF7bSpQ0i36A910R3JrE5c6uiuIQjQFF0gVCO7
Hum.zI1lDsEZcjM07syG7B1ggxhtdAW9xN1:18288:0:99999:7:::
```

John the ripper

- https://www.openwall.com/john/
- Open source password security auditing
  - Unix, macOS, Windows
  - Popular web apps, e.g. WordPress
  - Database servers
  - Network traffic, e.g. wifi WPA-PSK
  - Filesystems, e.g. macOS .dmg files
  - Archives, e.g. zip, rar
  - Documents, e.g. PDF, Microsoft Office

Johnny – a GUI frontend for john

- sudo apt-get install johnny
- sudo johnny



Settings: attack mode

- **Single crack**
  - guess password using *username* and other <u>fields</u> from the `passwd` file + mangling <u>rules</u>
- **Word list**
  - Uses a resource of <u>passwords</u> + <u>rules</u>
  - In kali linux: /usr/share/wordlists
- **Incremental**
  - Combinations according to <u>rules</u>
- **Default**
  - Single crack → word list → incremental
- **External**
  - Specify your own <u>custom code</u> (in C) to enumerate passwords to try

Single crack mode

- Use information available in the password file (e.g., username)

- Example: for user "Hacker" these passwords will be attempted
  - hacker
  - HACKER
  - hacker1
  - h-acker
  - hacker=

- Usage:
  - `john --single single-mode.pwd`
  - `john --show single-mode.pwd`

Wordlist Crack Mode

- Use a wordlist (aka Dictionary)
- Compares the hashes of the words in the Dictionary with the hashes of the passwords to guess.
- We can use any desired wordlist.
- John comes with a password.lst which contains most of the common passwords

- Usage:
  - `john --wordlist=password.lst wordlist-mode.pwd`
  - `john --wordlist=password.lst --stdout`
  - `john --show wordlist-mode.pwd`

Wordlist Crack Mode + rules

- End-users often apply small changes to old passwords, in order not to remember a completely new password
- Rule: enable word mangling rules for wordlist mode
  - A wordlist represents only the starting point for cracking a password
  - It is more flexible to specify how to change words from the list to guess small deviations from them

Character classes

Rule	Meaning
?v	vowels: "aeiouAEIOU"
?c	consonants: "bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ"
?l	lowercase letters [a-z]
?u	uppercase letters [A-Z]
?d	digits [0-9]
?a	letters [a-zA-Z]
?x	letters and digits [a-zA-Z0-9]
?w	whitespace: space and horizontal tabulation characters
?p	punctuation: ".,:;'?!`"
?s	symbols "$%^&*()-_+=|\<>[]{}#@/~"

Rules

Rule	Meaning
:	no-op: do nothing to the input word
l	convert to lowercase
u	convert to uppercase
c	capitalize
C	lowercase the first character, and uppercase the rest
t	toggle case of all characters in the word
TN	toggle case of the character in position N
r	reverse: "Fred" -> "derF"
d	duplicate: "Fred" -> "FredFred"
f	reflect: "Fred" -> "FredderF"
{	rotate the word left: "jsmith" -> "smithj"
}	rotate the word right: "smithj" -> "jsmith"
$X	append character X to the word
^X	prefix the word with character X

Reject rules

Rule	Meaning
-:	no-op: don't reject
-c	reject this rule unless current hash type is case-sensitive
-8	reject this rule unless current hash type uses 8-bit characters
-s	reject this rule unless some password hashes were split at loading
-p	reject this rule unless word pair commands are currently allowed

Wordlist Crack Mode + predefined rules

- Usage:
  - `john --wordlist=password.lst --rules wordlist-rules-mode.pwd`
  - `john --show wordlist-rules-mode.pwd`

Incremental mode

- Brute force attack by enumeration all the possible passwords (possibly according to rules)
  - Passwords with 0-13 characters, chosen from 95 printable ASCII characters
  - 8^{95} (10^{85}) passwords
  - Try earlier with sequences with higher probability

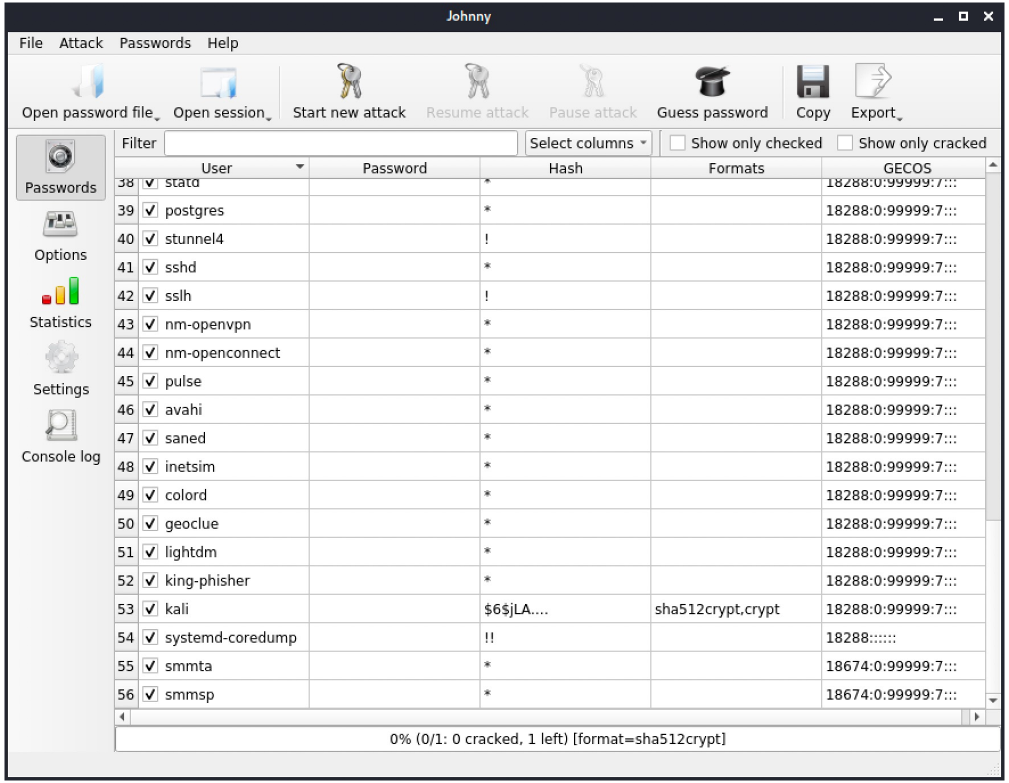
- Usage:
  - `john --incremental incremental-mode.pwd`

Incremental mode – character limits

- Unless all the passwords are weak and get cracked, incremental mode can take a long time to complete
- Limited character set crack simpler passwords faster
  - ASCII: all 95 printable ASCII characters (default),
  - LM_ASCII: for use on Windows LM hashes
  - Alnum: all 62 alphanumeric characters
  - Alpha: all 52 letters
  - LowerNum: lowercase letters plus digits, for 36 total
  - UpperNum: uppercase letters plus digits, for 36 total
  - LowerSpace: lowercase letters plus space, for 27 total
  - Lower: lowercase letters
  - Upper: uppercase letters
  - Digits: digits only
- Usage:
  - `john --incremental=digits incremental-mode.pwd`

Guess kali linux password

- Open password file
  - /etc/passwd
- Start new attack



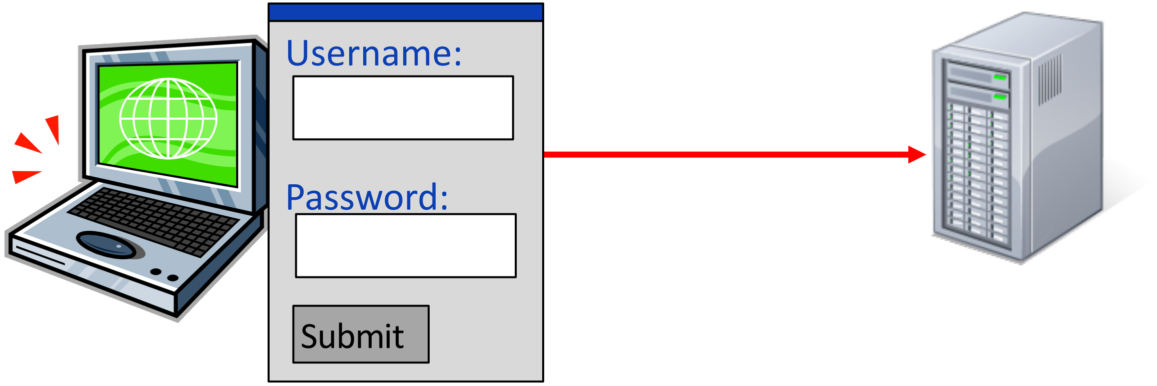
		User	Password	Hash	Formats	GECOS
38	✓	statd		*		18288:0:99999:7:::
39	✓	postgres		*		18288:0:99999:7:::
40	✓	stunnel4		!		18288:0:99999:7:::
41	✓	sshd		*		18288:0:99999:7:::
42	✓	sslh		!		18288:0:99999:7:::
43	✓	nm-openvpn		*		18288:0:99999:7:::
44	✓	nm-openconnect		*		18288:0:99999:7:::
45	✓	pulse		*		18288:0:99999:7:::
46	✓	avahi		*		18288:0:99999:7:::
47	✓	saned		*		18288:0:99999:7:::
48	✓	inetsim		*		18288:0:99999:7:::
49	✓	colord		*		18288:0:99999:7:::
50	✓	geoclue		*		18288:0:99999:7:::
51	✓	lightdm		*		18288:0:99999:7:::
52	✓	king-phisher		*		18288:0:99999:7:::
53	✓	kali		6jLA....	sha512crypt,crypt	18288:0:99999:7:::
54	✓	systemd-coredump		!!		18288::::::
55	✓	smmta		*		18674:0:99999:7:::
56	✓	smmsp		*		18674:0:99999:7:::

```
0% (0/1: 0 cracked, 1 left) [format=sha512crypt]
```

2. Online password attack

Password-based authentication

- Each end-user is assigned a <username,password> pair
  - The *username* is used to identify the end-user (typically public)
  - The *password* is used to authenticate the end-user (private)



- A user is granted access only if the provided pair matches the one in the password database

Black box approach

- The password file is not available
  - No possibility to compare with password hash

- The only option is to submit username/password to the authentication system and obtain an answer
  - Response time can be long
  - Guessing can be revealed in case of too many (or too frequent) attempts

TCP ports

- A service accepting incoming connections is listening on a TPC port (identified by a port number)
  - 25: SMTP Simple Mail Transfer Protocol.
  - 143: IMAP Internet Message Access Protocol
  - 80: HTTP Hypertext Transfer Protocol. ...
  - 443: HTTPS secure HTTP
  - 20-21: FTP File Transfer Protocol
  - 23: TELNET to establish connections between remote computers
  - 22: SSH Secure shell login
  - 53: DNS Domain Name System

Probing for for ports

- `nmap 192.168.56.102`

```
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-11 17:06 CET
Nmap scan report for 192.168.56.102
Host is up (0.00071s latency).
Not shown: 997 filtered ports
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
111/tcp open  rpcbind
```

Hydra

- Fast network logon cracker that supports many protocols
- https://github.com/vanhauser-thc/thc-hydra
  - Cisco AAA, Cisco auth, Cisco enable,
  - CVS, Subversion,
  - POP3, IMAP, SMTP, SMTP Enum, SNMP v1+v2+v3, SIP,
  - FTP, HTTP(S)-FORM-GET, HTTP(S)-FORM-POST, HTTP(S)-GET, HTTP(S)-HEAD, HTTP-Proxy,
  - ICQ, IRC, VNC and XMPP,
  - MS-SQL, MySQL, NNTP, Oracle Listener, Oracle SID, PC-Anywhere, PC-NFS, PostgreSQL,
  - LDAP, RDP, Rexec, Rlogin, Rsh, SSH (v1 and v2), SMB(NT), SSHKEY, SOCKS5, Teamspeak (TS2), Telnet, VMware-Auth.

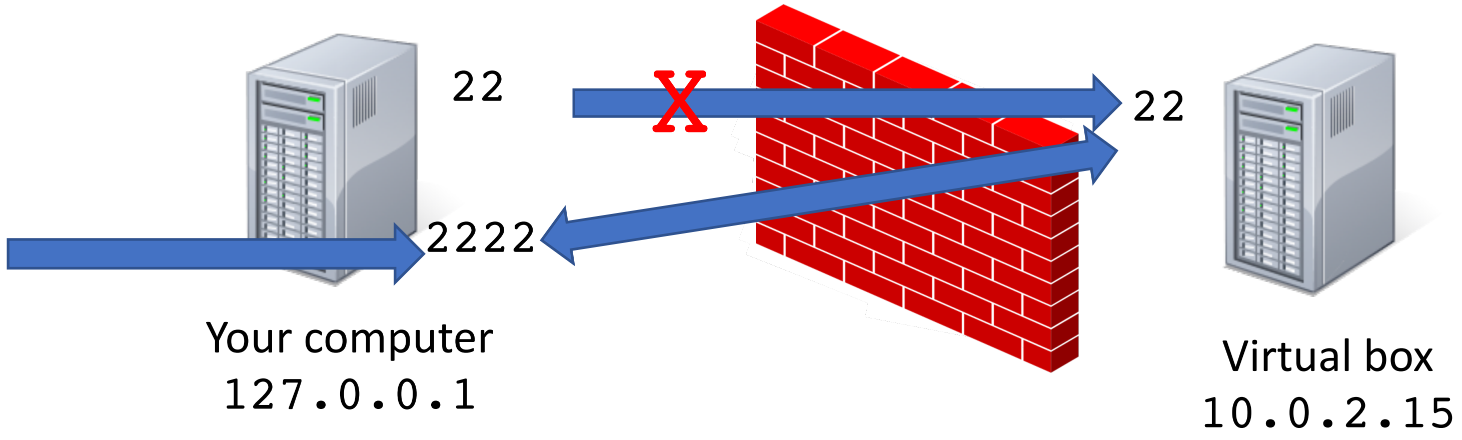
Command line

- hydra
  - -l login
    - or
  - -L logins.txt
  - -p password
    - or
  - -P passwords.txt
  - -f -Vv
  - server service [extra-info]
    - or
  - service://server[:port]

Exercise

- Download and run the virtual-box virtual machine from the course web site
- Use `ssh` to access to the VM
  - Type:
    - `ssh admin@192.168.56.102`
    - `ssh admin@10.0.2.15`
- Does it work?

Firewall and port forwarding



```
ssh –p 2222 117.0.01
```

Exercise

- Guess the password of user `admin` using Hydra
  - User: `admin`
  - Passwords file: `unix_passwords.txt`
  - Service:
    - `ssh://127.0.0.1:2222`
    - `ssh://192.168.56.102`

Cracking Unix passwords with Hydra

```
hydra
-l admin
-P unix_passwords.txt
-f -Vv
ssh://127.0.0.1:2222
```

Username to guess the password for

File with password list

Stop on success, verbose output

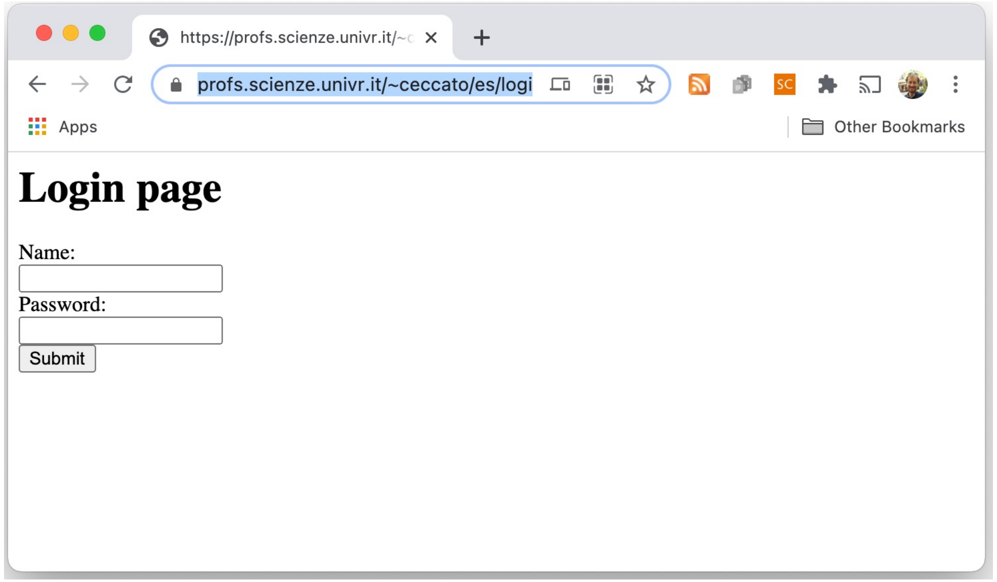
service

Server

Port

Cracking a web-site password

- https://profs.scienze.univr.it/~ceccato/es/login.php

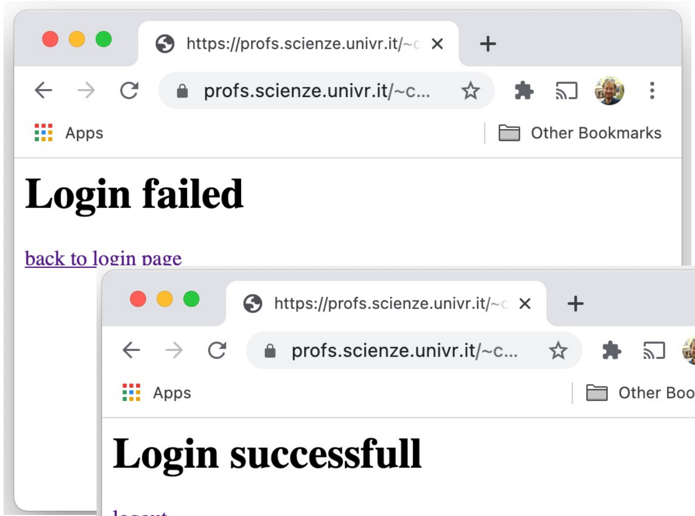

https://profs.scienze.univr.it/~c
profs.scienze.univr.it/~ceccato/es/logi
Apps
Other Bookmarks

Login page

Name:

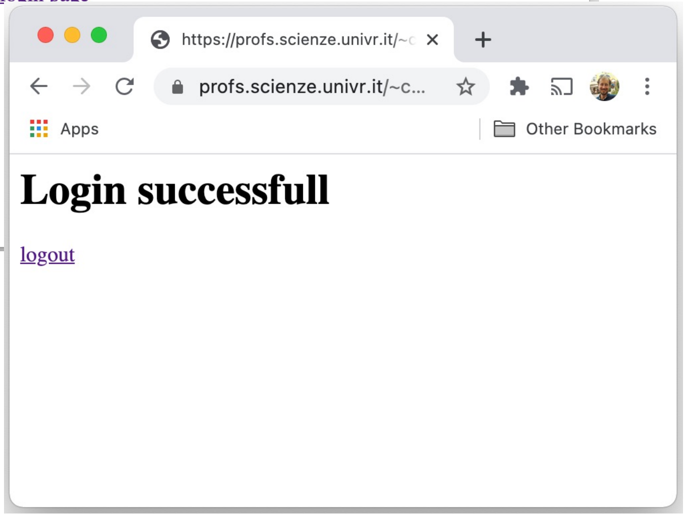
Password:

Submit


https://profs.scienze.univr.it/~c
profs.scienze.univr.it/~c...
Apps
Other Bookmarks

Login failed

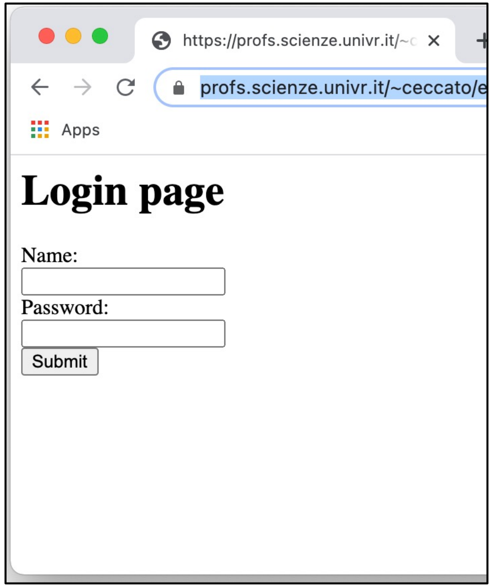
back to login page


https://profs.scienze.univr.it/~c
profs.scienze.univr.it/~c...
Apps
Other Bookmarks

Login successfull

logout

HTTP messages



https://profs.scienze.univr.it/~c

profs.scienze.univr.it/~ceccato/e

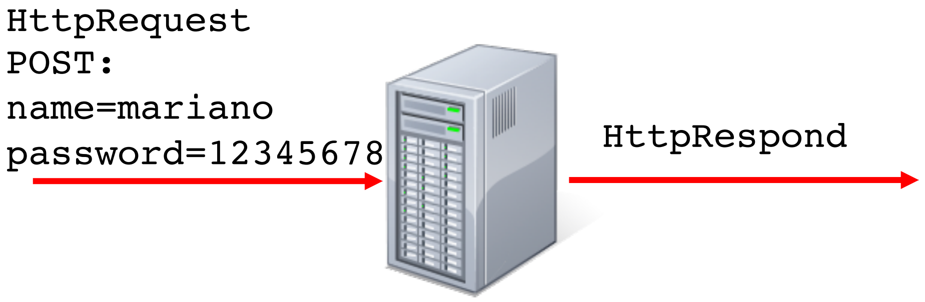
Apps

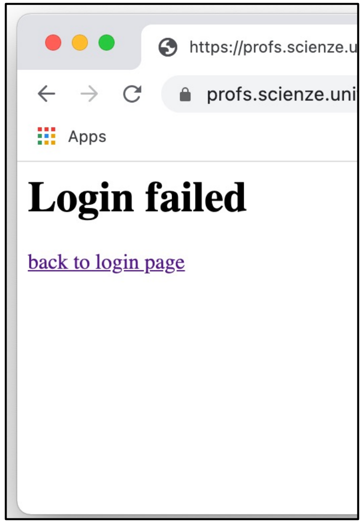
Login page

Name:

Password:

Submit





https://profs.scienze.u

profs.scienze.uni

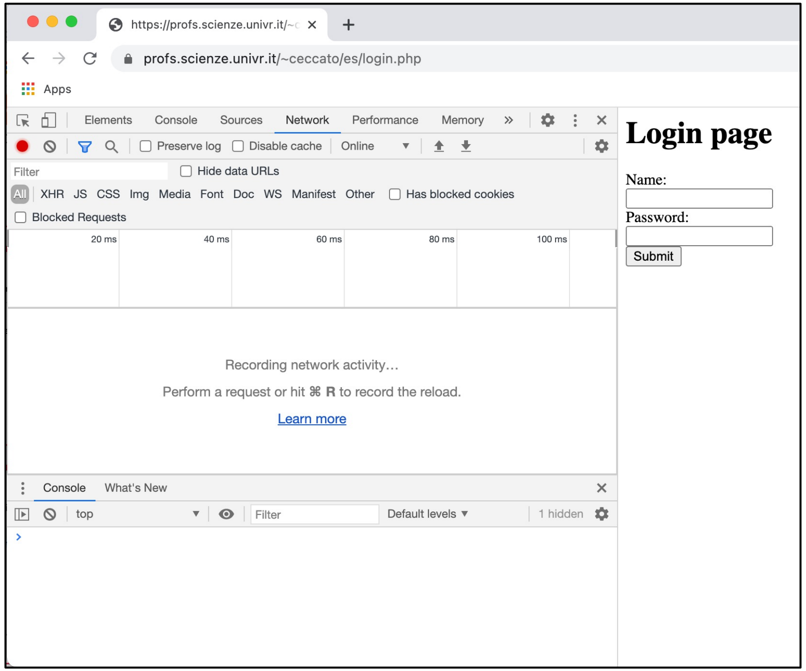
Apps

Login failed

back to login page

Network messages inspection

- Menu →View → Developer → Developer tools: Network



Recording network activity…

Perform a request or hit ⌘ R to record the reload.

Learn more

Login page

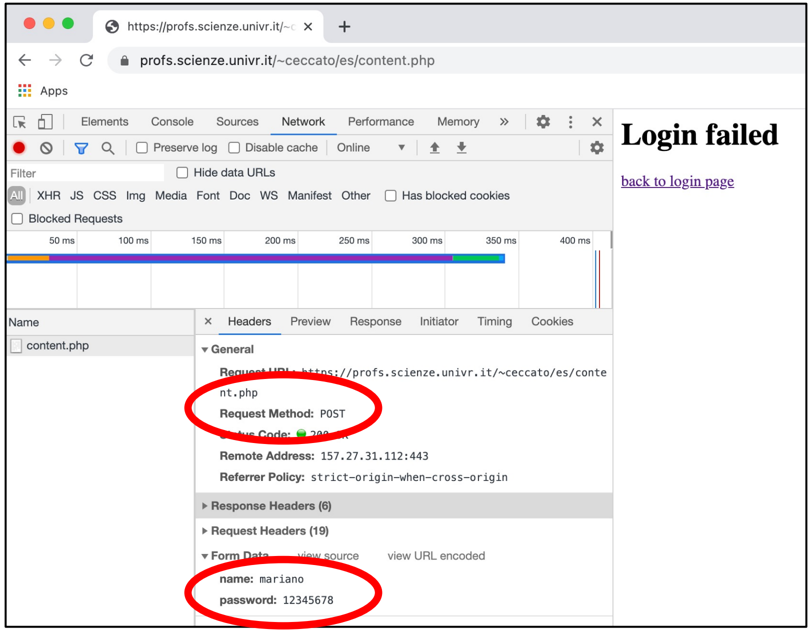
Name:

Password:

Submit

Network messages inspection

- Menu →View → Developer → Developer tools: Network



General

Request URL: https://profs.scienze.univr.it/~ceccato/es/content.php
Request Method: POST
Status Code: 200 OK
Remote Address: 157.27.31.112:443
Referrer Policy: strict-origin-when-cross-origin

Response Headers (6)

Request Headers (19)

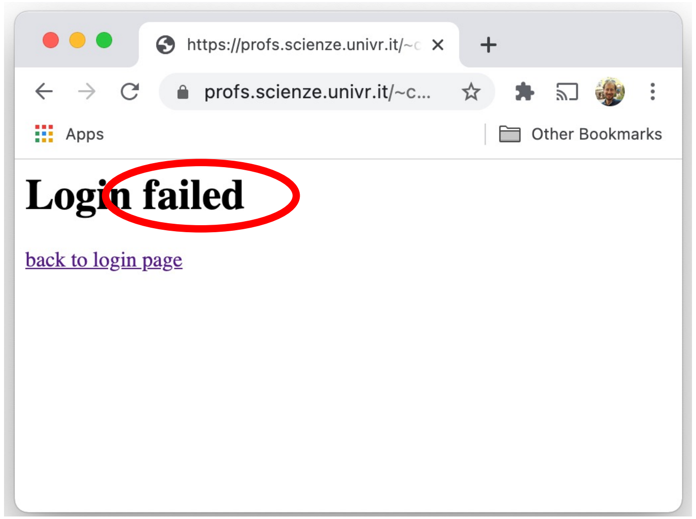
Form Data view source view URL encoded

name: mariano
password: 12345678

Login failed

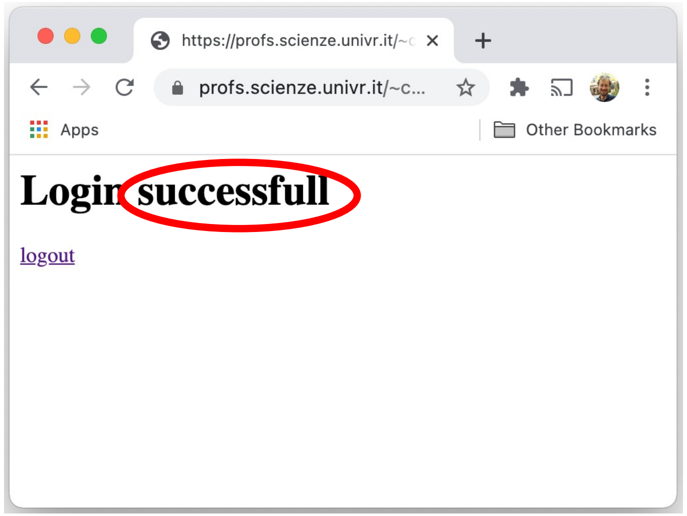
back to login page

Telling if password is correct/wrong


https://profs.scienze.univr.it/~c
profs.scienze.univr.it/~c...
Apps
Other Bookmarks

Login failed

back to login page


https://profs.scienze.univr.it/~c
profs.scienze.univr.it/~c...
Apps
Other Bookmarks

Login successfull

logout

Exercise

- Try to guess the password of user Mariano
  - User name: mariano
  - Passwords list: unix_passwords.txt
  - Service: https-post-form
  - Server: profs.scienze.univr.it
  - Extra info: ”url:paramter1=^USER^&paramter2=^PASS^:S=string”
    - Url: `/~ceccato/es/content.php`
    - Paramenter1: `name`
    - Parameter2: `password`
    - Success string: “`success`”

Running Hydra

```
hydra
-l mariano
-P wordlists/unix_passwords.txt
-f -Vv
profs.scienze.univr.it
https-post-form
"/~ceccato/es/content.php:name=^USER^&password=^PASS^:S=successfull"
```

`-l mariano` Username to guess the password for

`-P wordlists/unix_passwords.txt` File with password list

`-f -Vv` Stop on success, verbose output

`profs.scienze.univr.it` Server

`https-post-form` service

`"/~ceccato/es/content.php:name=^USER^&password=^PASS^:S=successfull"` Extra info

$\underbrace{\texttt{/\~{}ceccato/es/content.php}}_{\text{URL}}\texttt{:}\underbrace{\texttt{name}}_{\text{Username parameter name}}\texttt{=\^{}USER\^{}\&}\underbrace{\texttt{password}}_{\text{Password parameter name}}\texttt{=\^{}PASS\^{}:}\underbrace{\texttt{S=successfull}}_{\text{Decision pattern}}$

Decision pattern
"S=" for success
"F=" for failure

Questions?