

WEB EXPLOITATION WORKSHOP

Δέσποινα Βιδάλη - Γεώργιος Μπούρδαλος



NTUA_H4CK
POWERED BY CS IEEE NTUA SB

Web Intro



How the web works,
almost ...

HTTP Protocol and Request

Hypertext **t**ransfer **p**rotocol is used by web browsers and servers to communicate and transfer web page data, whether that is HTML, Images, Videos, etc (it's not secure because the data are not encrypted like the more secure HTTPS)

There's a set of HTTP request method for different desired actions and interactions. The two most commonly used methods are GET and POST.

- The GET method requests a representation of the specified resource. Requests using GET should only retrieve data. This is usually the method used when visiting a webpage.
- The POST method is used to send data to the server to create or update a resource. A common example of this method is submitting forms.

Other methods: HEAD, PUT, DELETE, CONNECT, OPTIONS, TRACE, PATCH

Http Request & Response content

Using cURL or Burp suite we can see what the request and response looks like

Here we can see our http request with it's method and some Header. The headers give some information about what we want and who we are to the server

- >Http method
- >IP address of the host
- >The Browser we use
- >What kind of data we expect the server to return to us

```
→ ~ curl 192.168.252.129/login.php -v
*   Trying 192.168.252.129:80...
* Connected to 192.168.252.129 (192.168.252.129) port 80
> GET /login.php HTTP/1.1
> Host: 192.168.252.129
> User-Agent: curl/8.4.0
> Accept: */*
```

Http Request & Response content

```
< HTTP/1.1 200 OK
< Date: Thu, 14 Dec 2023 17:11:07 GMT
< Server: Apache/2.2.14 (Unix) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.8l PHP/5.3.3
< X-Powered-By: PHP/5.3.1
< Set-Cookie: PHPSESSID=ilngl3meo02qcop6pejf7r7k92; path=/
< Expires: Tue, 23 Jun 2009 12:00:00 GMT
< Cache-Control: no-cache, must-revalidate
< Pragma: no-cache
< Set-Cookie: security=high
< Content-Length: 1224
< Content-Type: text/html; charset=utf-8
<

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"
<html xmlns="http://www.w3.org/1999/xhtml"
```

In the response we can find the status code that the server and some response header that give useful information about the server to our browser.

The most interesting are:

- Status code: indicates if our request has been successfully completed ex. "404 not found"
- Set-Cookies: returns the cookies that the server has assigned us

At the bottom of the response we can see the html code that we requested.

The cookies Header

HTTP cookies provide the server with a mechanism to store and retrieve state information on the client application's system. This mechanism allows web based applications the ability to store information about selected items, user preferences, registration information, and other information that can be retrieved later.

User sessions: Cookies help associate website activity with a specific user. A session cookie contains a unique string (a combination of letters and numbers) that matches a user session with relevant data and content for that user.

Personalization: Cookies help a website "remember" user actions or user preferences, enabling the website to customize the user's experience.

Tracking: Some cookies record what websites users visit.

Note that the set-cookies value is used as a cookies value to the future requests we make to the same server.

HTTP Status Codes

100-199 : Information Response These are sent to tell the client the first part of their request has been accepted and they should continue sending the rest of their request. These codes are no longer very common.

200-299 : Success This range of status codes is used to tell the client their request was successful.

300-399 : Redirection These are used to redirect the client's request to another resource. This can be either to a different webpage or a different website altogether.

400-499 : Client Errors Used to inform the client that there was an error with their request.

500-599 : Server Errors This is reserved for errors happening on the server side and usually indicate quite a major problem with the server handling the request.

HTTP Status Codes

200 : OK The request was completed successfully.

201 : Created A resource has been created (for example a new user or new blog post).

301 : Permanent Redirect This redirects the client's browser to a new webpage or tells search engines that the page has moved somewhere else and to look there instead.

302 : Temporary Redirect Similar to the above permanent redirect, but as the name suggests, this is only a temporary change and it may change again in the near future.

400 : Bad Request This tells the browser that something was either wrong or missing in their request. This could sometimes be used if the web server resource that is being requested expected a certain parameter that the client didn't send.

401 : Not Authorised You are not currently allowed to view this resource until you have authorised with the web application, most commonly with a username and password.

403 : Forbidden You do not have permission to view this resource whether you are logged in or not.

405 : Method Not Allowed The resource does not allow this method request, for example, you send a GET request to the resource /create account when it was expecting a POST request instead.

404 : Page Not Found The page/resource you requested does not exist.

500 : Internal Service Error The server has encountered some kind of error with your request that it doesn't know how to handle properly.

503 : Service Unavailable This server cannot handle your request as it's either overloaded or down for maintenance.

Cookie Manipulation



Exploiting Cookie Values

Session Hijacking: This attack compromises the session token by stealing or predicting an active session token to gain unauthorized access to the Web Server. With the stolen session cookie, the attacker can access sensitive data and perform actions on behalf of the user without their knowledge.

Some ways that session cookies can be compromised include predicting valid session ID values, Session Sniffing, Cross-Site Scripting (XSS), Social Engineering.

Cookie Tampering: Manipulating the stored data on a web browser. This method can be used to change user privileges, access sensitive information from users, or other session-related data. For example, it's possible to alter a user's role from regular user to administrator by manipulating the cookie values.

Practical Example (Root me challenge)

Email

[Saved email addresses](#)
You need to be admin

Validation password : ml-SYMPA

Elements Console Sources Application >> 1 ⚙️ ⋮ ✕

Filter

Name	Value	D	P	E	S	H	S	S	P	P
ga...	GS1.1.17025...	/	2..	5..					M.
_ga	GA1.1.12588...	/	2..	2..					M.
ch7	visiteur	c...	/...	S..	1..					M.

IndexedDB
Web SQL
Cookies
http://challenge01.root
Private state tokens
Interest groups
Shared storage
Cache storage

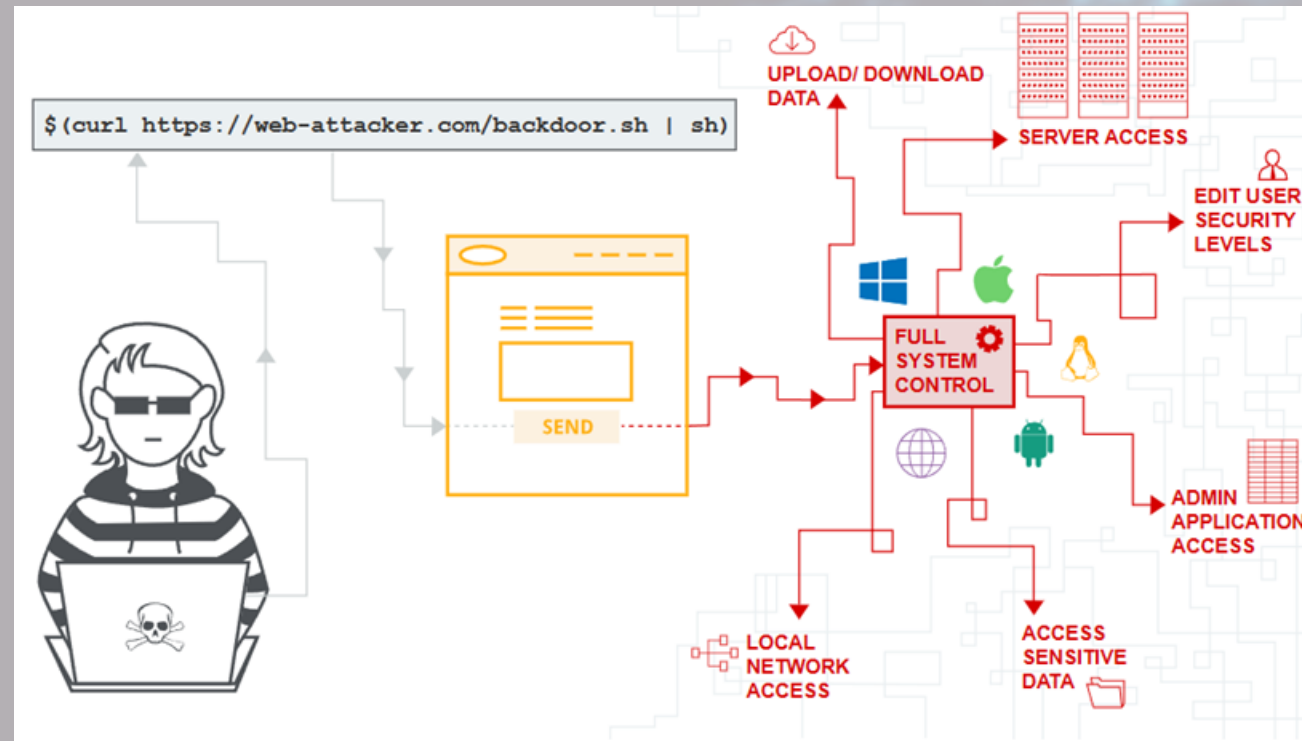
Elements Console Sources Application >> 1 ⚙️ ⋮ ✕

Filter

Na...	Value	D	P	E	S	H	S	S	P	P
_g...	GS1.1.17...	/	2.	5.					M
_ga	GA1.1.12...	/	2.	2.					M
ch7	admin	c..	/..	S.	8.					M

IndexedDB
Web SQL
Cookies
http://challenge01.root
Private state tokens
Interest groups
Shared storage
Cache storage

Command Injection



What is a shell command?

A shell is a program that allow to the user to talk with the operating system with commands.

In the old days, it was the only user interface available on a Unix-like system such as Linux.

There are different types of shell like powershell, command shell, bash ... etc

Purpose of command

	Linux	Windows
Name of current user	<code>whoami</code>	<code>whoami</code>
Operating system	<code>uname -a</code>	<code>ver</code>
Network configuration	<code>ifconfig</code>	<code>ipconfig /all</code>
Network connections	<code>netstat -an</code>	<code>netstat -an</code>
Running processes	<code>ps -ef</code>	<code>tasklist</code>

```
(p13rr3@kali)-[~]  
$ ifconfig  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.252.131 netmask 255.255.255.0 broadcast 192.168.252.255  
    inet6 fe80::20c:29ff:febe:8b30 prefixlen 64 scopeid 0x20<link>  
    ether 00:0c:29:be:8b:30 txqueuelen 1000 (Ethernet)  
    RX packets 99224 bytes 62967318 (60.0 MiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 72001 bytes 14435671 (13.7 MiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```


What is Command Injection?

Command injection is another injection type where the attacker injects malicious shell commands into input fields of a web application.

This typically involves executing commands in a system shell or other parts of the environment. The attacker extends the default functionality of a vulnerable application, causing it to pass commands to the system shell, without needing to inject malicious code. In many cases, command injection gives the attacker greater control over the target system.

The **&** character is a shell **command separator** for the bash shell and is quite useful because it help us execute two commands with one like.

Practical Example



Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

DVWA Security

PHP Info

About

Logout

Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

submit

More info

<http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
<http://www.ss64.com/bash/>
<http://www.ss64.com/nt/>

Username: admin


Security Level: low

PHPIDS: disabled

View Source

View Help

Damn Vulnerable Web Application (DVWA) v1.0.7



Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

DVWA Security

PHP Info

About

Logout

Vulnerability: Command Execution

Ping for FREE

Enter an IP address below:

submit

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
syslog:x:101:103::/home/syslog:/bin/false
dvwa:x:1000:1000:dvwa,,,:/home/dvwa:/bin/bash
sshd:x:102:65534:./var/run/sshd:/usr/sbin/nologin
messagebus:x:103:110:./var/run/dbus:/bin/false
usbmux:x:104:46:usbmux daemon,,,:/home/usbmux:/bin/false
pulse:x:105:111:PulseAudio daemon,,,:/var/run/pulse:/bin/false
rtkit:x:106:113:RealtimeKit,,,:/proc:/bin/false
```

More info

Username: admin

Security Level: low

PHPIDS: disabled

View Source

View Help

Damn Vulnerable Web Application (DVWA) v1.0.7

SQL Injection



the basics όμως :)

What is SQL?

SQL, which stands for Structured Query Language, is the language used to communicate with databases allowing us to retrieve, update, insert, and delete data. (Databases are organized collections of structured information).

In web applications, SQL plays a crucial role. It helps websites interact with their storage systems, like saving user information, handling login credentials, or managing product inventories. Whenever you fill out a form online, like entering your username and password, the web application often uses SQL to store that information securely in its database. (Usually not plaintext)

For example, this query is used in a typical login system where a user enters their username and password. The query checks if the entered username and password match any entry in the "users" table.

```
SELECT * FROM users WHERE username = 'input_username' AND password = 'input_password';
```


What is SQL Injection?

SQL injection is a hacking technique where attackers insert malicious SQL code into input fields of a web application.

This technique exploits vulnerable input validation mechanisms by inserting carefully crafted SQL commands into these input fields.

The attackers can trick the application into executing unintended SQL commands to potentially gain unauthorized access to sensitive data that was not intended to be displayed, modify data, or even execute administrative operations on the database.

Example of SQL Injection

For the previous example query:


```
SELECT * FROM users WHERE username = 'input_username' AND password = 'input_password';
```

If we enter '**OR 1=1 --**' as the username, the executing query becomes:

```
SELECT * FROM users WHERE username = ' ' OR '1'='1' -- ' AND password = 'input_password';
```

That means it could potentially manipulate the query to always return a valid login, bypassing the need for a correct username or password.

Practical Example in Damn Vulnerable Web Application



[Home](#)
[Instructions](#)
[Setup](#)

[Brute Force](#)
[Command Execution](#)
[CSRF](#)
[File Inclusion](#)
[SQL Injection](#)
[SQL Injection \(Blind\)](#)
[Upload](#)
[XSS reflected](#)
[XSS stored](#)

[DVWA Security](#)
[PHP Info](#)
[About](#)

[Logout](#)

Vulnerability: SQL Injection

User ID:

More info

<http://www.securiteam.com/securityreviews/SDP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

Username: admin
Security Level: low
PHPIDS: disabled

[View Source](#) [View Help](#)

Damn Vulnerable Web Application (DVWA) v1.0.7



[Home](#)
[Instructions](#)
[Setup](#)

[Brute Force](#)
[Command Execution](#)
[CSRF](#)
[File Inclusion](#)
[SQL Injection](#)
[SQL Injection \(Blind\)](#)
[Upload](#)
[XSS reflected](#)
[XSS stored](#)

[DVWA Security](#)
[PHP Info](#)
[About](#)

[Logout](#)

Vulnerability: SQL Injection

User ID:

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

More info

<http://www.securiteam.com/securityreviews/SDP0N1P76E.html>
http://en.wikipedia.org/wiki/SQL_injection
<http://www.unixwiz.net/techtips/sql-injection.html>

Username: admin
Security Level: low
PHPIDS: disabled

[View Source](#) [View Help](#)