

INFOSEC GIRLS TRAINING IN COLLABORATION WITH HACKCUMMINS

\$WHOIS INFOSEC GIRLS

- Community for women in Information Security.

[Website](#)

[Twitter](#)

[Facebook](#)

[Linkedin](#)

[YouTube](#)

[Mailing List](#)



WHY INFOSEC GIRLS?

- We aim is to bring in more women in security and help them to grow in the area.

TRAINERS

- Ishaq Mohammed, Appsec @ Qualys
- Shrutirupa Banerjee, @WAF Research @ Qualys
- Komal Armarkar, Spotlight Engineer @Crowdstrike

AGENDA:

Basics of Web Application

Architecture

Client Server Communication

HTTP/S

HTTP Methods

Status Codes

The WHY factor

Test Cases

Vulnerabilities

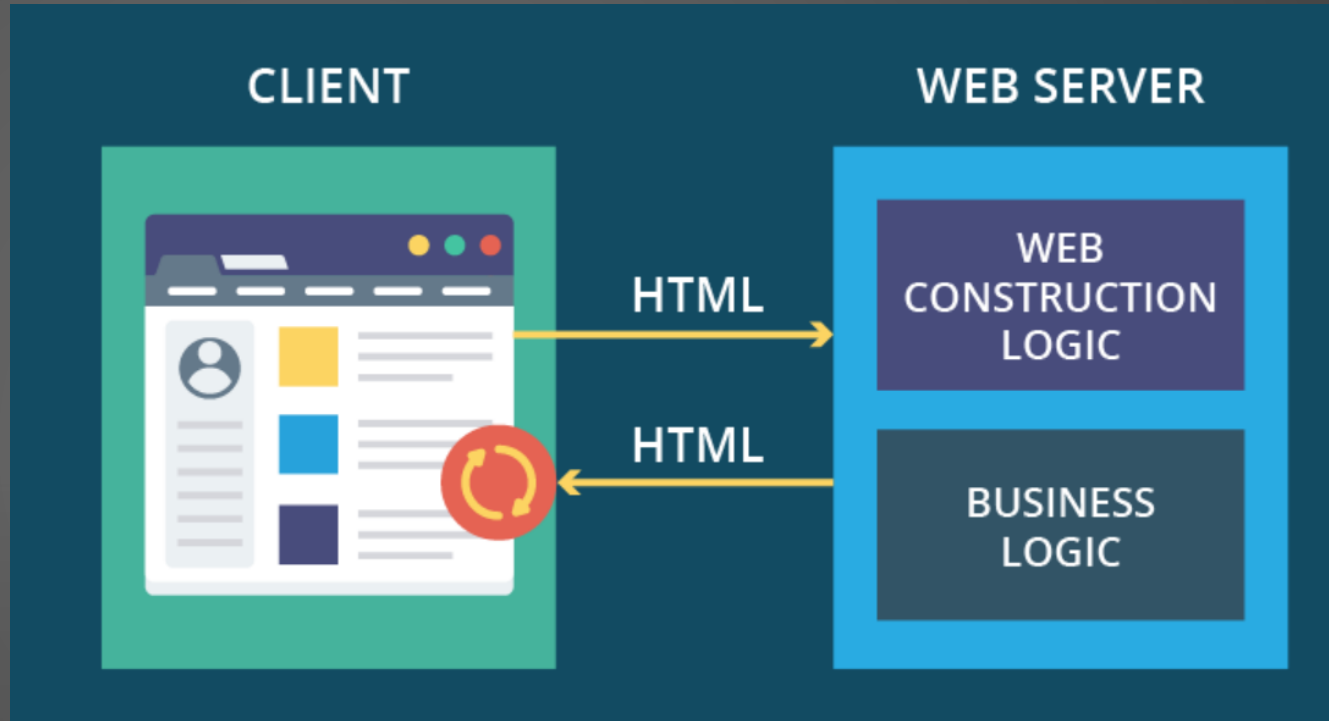
Demo



WEB APPLICATION



ARCHITECTURE



HTTP/HTTPS

- HyperText Transfer Protocol (Secure)
- used by the World Wide Web
- defines how messages are formatted and transmitted
- what actions Web servers and browsers should take in response to various commands

METHODS

- GET
- POST
- PUT
- OPTIONS
- AND MANY MORE...

STATUS CODES

- 1xx - informational
- 2xx - success
- 3xx - redirection
- 4xx - client error
- 5xx - server error

HTTP HEADERS

```
GET / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:71.0) Gecko/20100101 Firefox/71.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Age: 110010
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Fri, 17 Jan 2020 19:44:04 GMT
Etag: "3147526947+ident"
Expires: Fri, 24 Jan 2020 19:44:04 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: ECS (nyb/1D1F)
Vary: Accept-Encoding
X-Cache: HIT
Content-Length: 1256
Connection: close
```

```
POST /DVWA/vulnerabilities/sqli/ HTTP/1.1
Host: localhost.
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:71.0)
Gecko/20100101 Firefox/71.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 18
Origin: http://localhost.
Connection: close
Referer: http://localhost./DVWA/vulnerabilities/sqli/
Cookie: security=medium; PHPSESSID=9b405610uc02i6laem0a3kn7qq
Upgrade-Insecure-Requests: 1

id=1&submit=submit
```

```
HTTP/1.1 200 OK
Date: Fri, 17 Jan 2020 19:44:44 GMT
Server: Apache/2.4.38 (Ubuntu)
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 4682
Connection: close
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">

  <head>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

    <title>Vulnerability: SQL Injection :: Damn Vulnerable Web Application (DVWA) v1.10 *Development*</title>

    <link rel="stylesheet" type="text/css" href="../../dvwa/css/main.css" />

    <link rel="icon" type="image/ico" href="../../favicon.ico" />

    <script type="text/javascript" src="../../dvwa/js/dvwaPage.js"></script>

  </head>

  <body class="home">
    <div id="container">

      <div id="header">

      </div>

      <div id="main_menu">

        <div id="main_menu_padded">
          <ul class="menuBlocks"><li class=""><a href="../../">Home</a></li>
          <li class=""><a href="../../instructions.php">Instructions</a></li>
          <li class=""><a href="../../setup.php">Setup / Reset DB</a></li>
          </ul><ul class="menuBlocks"><li class=""><a href="../../vulnerabilities/brute/">Brute Force</a></li>
          <li class=""><a href="../../vulnerabilities/exec/">Command Injection</a></li>
          <li class=""><a href="../../vulnerabilities/csrf/">CSRF</a></li>
          <li class=""><a href="../../vulnerabilities/fi/?page=include.php">File Inclusion</a></li>
          <li class=""><a href="../../vulnerabilities/upload/">File Upload</a></li>
          <li class=""><a href="../../vulnerabilities/captcha/">Insecure CAPTCHA</a></li>
          <li class="selected"><a href="../../vulnerabilities/sqli/">SQL Injection</a></li>
          <li class=""><a href="../../vulnerabilities/sqli_blind/">SQL Injection (Blind)</a></li>
          <li class=""><a href="../../vulnerabilities/weak_id/">Weak Session IDs</a></li>
          <li class=""><a href="../../vulnerabilities/xss_d/">XSS (DOM)</a></li>
```


COOKIE

```
GET /[REDACTED]/other/[REDACTED] HTTP/1.1
Host: net.[REDACTED].com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.5) Gecko
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq120
Pragma: no-cache
Cache-Control: no-cache
```


HOW DO WE VIEW
THE SOURCE
CODE??





View-source



Inspect element

VIEW-SOURCE

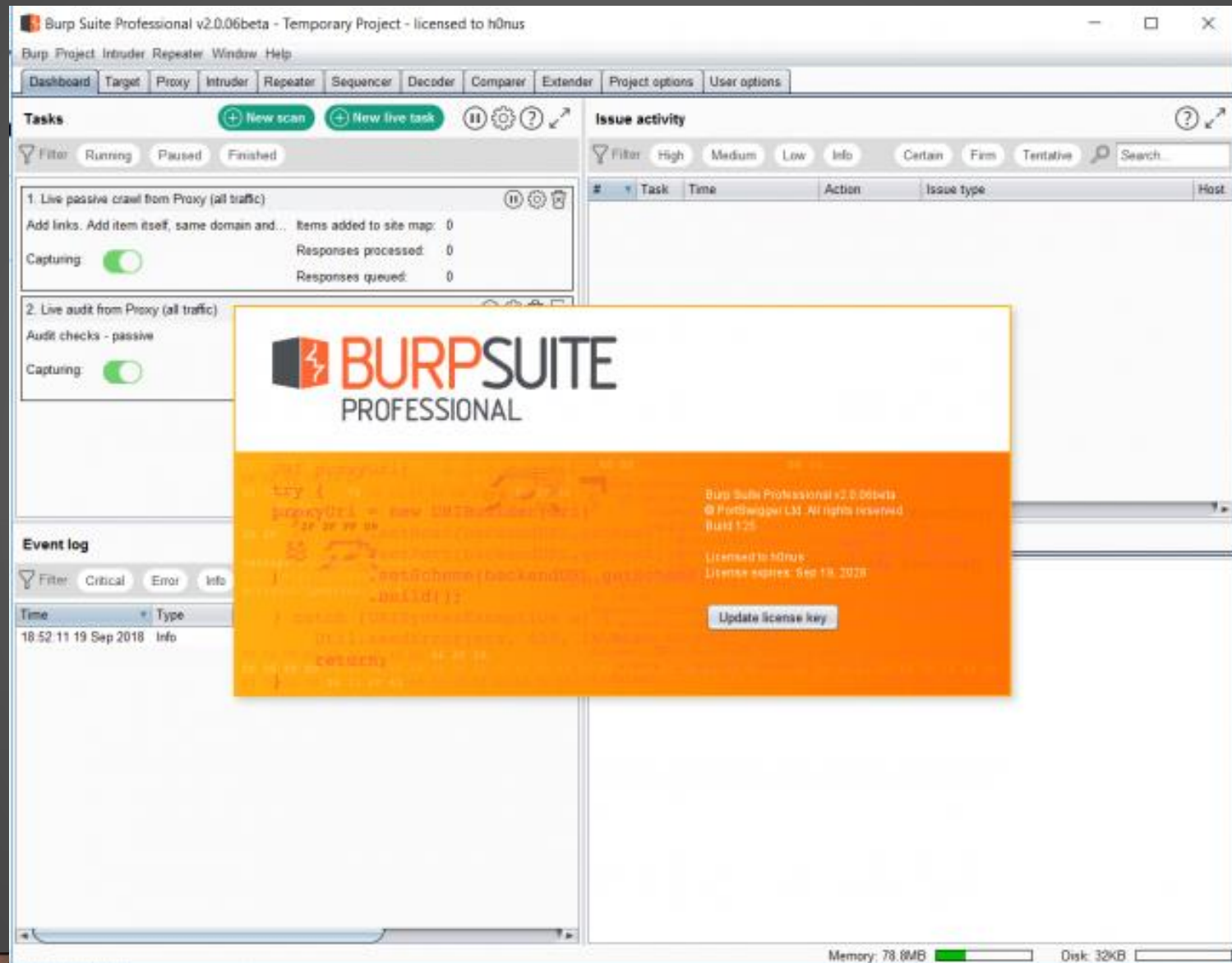
```
← → ↻ ↗ ⓘ Not secure | view-source:192.168.43.178/DVWA/vulnerabilities/upload/#
1
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3
4 <html xmlns="http://www.w3.org/1999/xhtml">
5
6   <head>
7     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
8
9     <title>Vulnerability: File Upload :: Damn Vulnerable Web Application (DVWA) v1.10 *Development*</title>
10
11     <link rel="stylesheet" type="text/css" href="../../dvwa/css/main.css" />
12
13     <link rel="icon" type="image/ico" href="../../favicon.ico" />
14
15     <script type="text/javascript" src="../../dvwa/js/dvwaPage.js"></script>
16
17   </head>
18
19   <body class="home">
20     <div id="container">
21
22       <div id="header">
23
24         
25
26       </div>
27
28       <div id="main_menu">
29
30         <div id="main_menu_padded">
31           <ul class="menuBlocks"><li class=""><a href="..">Home</a></li>
32 <li class=""><a href="../../instructions.php">Instructions</a></li>
33 <li class=""><a href="../../setup.php">Setup / Reset DB</a></li>
34 </ul><ul class="menuBlocks"><li class=""><a href="../../vulnerabilities/brute/">Brute Force</a></li>
35 <li class=""><a href="../../vulnerabilities/exec/">Command Injection</a></li>
36 <li class=""><a href="../../vulnerabilities/csrf/">CSRF</a></li>
37 <li class=""><a href="../../vulnerabilities/fi/?page=include.php">File Inclusion</a></li>
38 <li class="selected"><a href="../../vulnerabilities/upload/">File Upload</a></li>
39 <li class=""><a href="../../vulnerabilities/captcha/">Insecure CAPTCHA</a></li>
40 <li class=""><a href="../../vulnerabilities/sqli/">SQL Injection</a></li>
41 <li class=""><a href="../../vulnerabilities/sqli_blind/">SQL Injection (Blind)</a></li>
42 <li class=""><a href="../../vulnerabilities/weak_id/">Weak Session IDs</a></li>
43 <li class=""><a href="../../vulnerabilities/xss_d/">XSS (DOM)</a></li>
```

INSPECT ELEMENT

The screenshot shows a web browser window with the address bar displaying `192.168.43.178/DVWA/vulnerabilities/upload/#`. The page title is "Vulnerability: File Upload". On the left, a sidebar lists various security modules, with "File Upload" highlighted in green. The main content area contains a file upload form with a "Choose File" button and an "Upload" button. Below the form, there is a "More Information" section with three links: https://www.owasp.org/index.php/Security_Controls/Security_Controls_Web_Security/Security_Controls_Web_Security_File_Upload, <https://blogs.securiteam.com/index.php/archives/144>, and <https://www.acunetix.com/web/vulnerabilities/file-upload/>. The DevTools window is open on the "Elements" tab, showing the DOM tree. The selected element is the `<html>` tag, which has an `xmlns` attribute pointing to `http://www.w3.org/1999/xhtml`. The DOM tree shows the following structure:

```
<html xmlns="http://www.w3.org/1999/xhtml" => $0
  <head>...</head>
  <body class="home">
    <div id="container">
      <div id="header">...</div>
      <div id="main_menu">...</div>
      <div id="main_body">
        <div class="body_padded">
          <h1>Vulnerability: File Upload</h1>
          <div class="vulnerable_code_area">
            <form enctype="multipart/form-data" action="#" method="POST">...</form>
          </div>
        </div>
      </div>
    </div>
  </body>
</html>
```

The breadcrumb at the bottom of the DOM tree is: `html > body.home > div#container > div#main_body > div.body_padded > div.vulnerable_code_area > form`.



WHY ARE WE TALKING
ABOUT IT???

THE WHY FACTOR



OWASP Top 10 - 2017

A1:2017-Injection

A2:2017-Broken Authentication

A3:2017-Sensitive Data Exposure

A4:2017-XML External Entities (XXE)

A5:2017-Broken Access Control

A6:2017-Security Misconfiguration

A7:2017-Cross-Site Scripting (XSS)

A8:2017-Insecure Deserialization

A9:2017-Using Components with Known Vulnerabilities

A10:2017-Insufficient Logging & Monitoring

VULNERABILITIES TO BE COVERED TODAY



File inclusion



File upload



Xss reflected



Xss stored



SQL Injection

DEMO



Username

Password

Login

LET'S BEGIN



makeameme.org

FILE INCLUSION

- Local File Inclusion
- Remote File Inclusion

FILE UPLOAD

- Uploading a malicious file
- Can lead to Remote Code Execution

XSS REFLECTED

- malicious script bounces off of another website to the victim's browser
- You can trick a user to click on a malicious link to steal user's cookies

XSS STORED

- malicious script is injected directly into a vulnerable web application
- The script is stored in the web application

SQL INJECTION

- Executing malicious SQL statements in the web application
- To gain unauthorized access to the sensitive data in the database

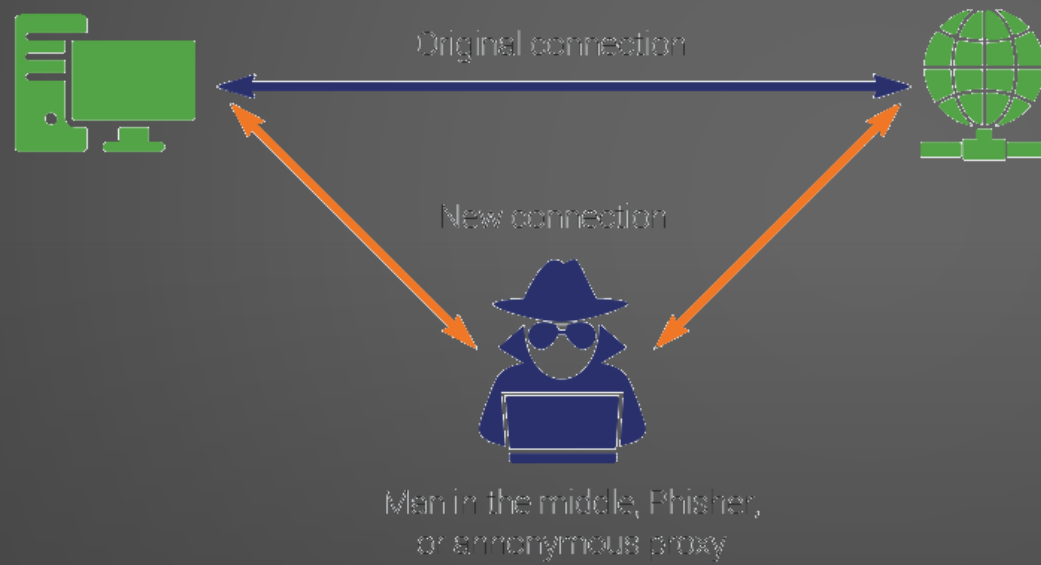
COMMAND INJECTION


- Attacker can execute commands directly
- Can lead to remote code execution, hence, getting the shell of the server

LET'S DEEP
DIVE A BIT
MORE!!!



WHAT IS PROXY??



 Request to http://192.168.0.104:80

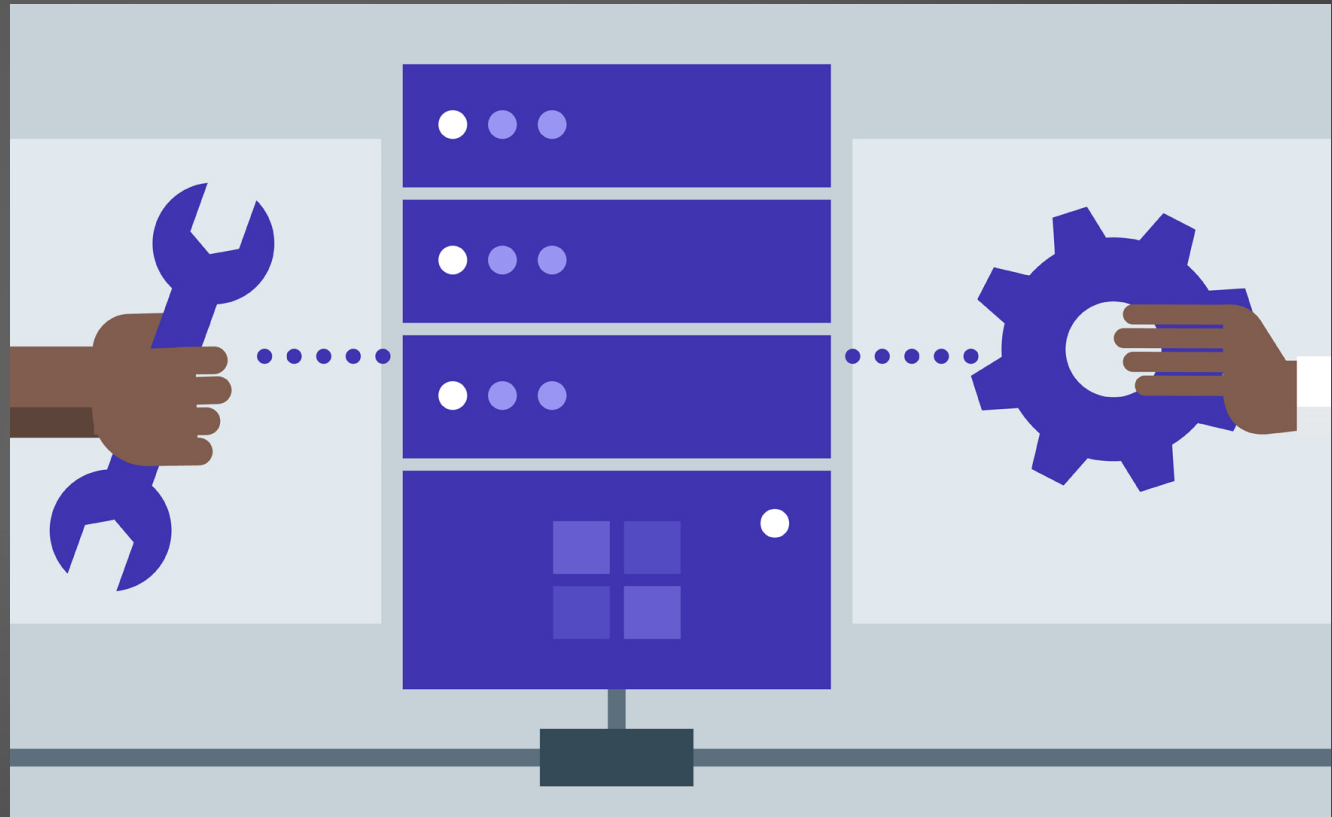
Forward Drop Intercept is on Action

Raw Params Headers Hex

```
GET /DVWA/index.php HTTP/1.1
Host: 192.168.0.104
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:73.0) Gecko/20100101 Firefox/73.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.0.104/DVWA/login.php
Connection: close
Cookie: security=impossible; PHPSESSID=lpj8ig72iaug5iqge75d46ekoi
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

BURPSUITE

INITIAL SETUP AND CONFIGURATION



A stylized, light gray silhouette of a spider is centered on a dark red background. The spider has a rounded body and eight long, thin legs. The word "SPIDER" is written in a bold, orange, serif font, positioned to the right of the spider's body.

SPIDER



SCANNER

Send

Cancel

< | ▾

> | ▾

Request

Raw

Params

Headers

Hex

```
GET /DVWA/vulnerabilities/csrf/?password_new=dead&password_conf=dead&Change=Change
HTTP/1.1
Host: 192.168.0.104
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:72.0) Gecko/20100101
Firefox/72.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://192.168.0.104/DVWA/vulnerabilities/csrf/
Cookie: security=low; PHPSESSID=rp5noh0lu8l9aeqnzn03t9tsri
Upgrade-Insecure-Requests: 1
```

Response

Raw

Headers

Hex

HTML

Render

```
<li class=""><a href="../../../phpinfo.php">PHP Info
<li class=""><a href="../../../about.php">About</a><
</li><div class="menuBlocks"><li class=""><a href=
</ul>
</div>
</div>
<div id="main_body">
<div class="body_padded">
  <h1>Vulnerability: Cross Site Request For
  <div class="vulnerable_code_area">
    <h3>Change your admin password:</
    <br />
    <form action="#" method="GET">
      New password:<br />
```

REPEATER

INTRUDER



Intruder

Repeater

Sequencer

Decoder

Comparer

Extender

Project options

User options

DECODER

LET'S EXPLOIT SOME VULNERABILITIES USING
BURP!!!





FILE UPLOAD –
MEDIUM



CSRF – LOW

FILE UPLOAD

- Let's bypass the mitigations

CSRF

- Cross Site Request Forgery
- Aims at authenticated users to make them execute unwanted actions



REFERENCES:

- [Damn Vulnerable Web Application \(DVWA\)](#)
- [HTTP](#)
- [Learn web development](#)

RESOURCES

- [So, you want to work in security?](#)
- [Roadmap for Application Security](#)
- [Getting Started in Offensive Security](#)
- [BREAKING INTO INFOSEC: A BEGINNERS CURRICULUM](#)
- [OWASP Foundation](#)
- [OWASP Top 10 – 2017](#)
- [CodePath Web Security Guides](#)
- [Awesome-Hacking](#)
- [High-Level Approaches for Finding Vulnerabilities](#)

TRAINER CONTACT

- Shrutirupa Banerjee - https://twitter.com/freak_crypt
- Komal Armarkar - https://twitter.com/n0th1n3_00X
- Ishaq Mohammed - https://twitter.com/security_prince

