



PROGRAMMAZIONE SICURA

bWAPP: Bee-Box «HTML Injection»

Candidati:

**Aniello Giugliano
Giacomo Cocozziello**

Docente:

Barbara Masucci

OBIETTIVO DELLA PRESENTAZIONE

L'obiettivo della presentazione è quello di approfondire la tematica della «HTML Injection» attraverso l'utilizzo della macchina *bWAPP*: *bee-box*.

In particolare andremo ad approfondire tre tipi di attacchi:

- *HTML Injection – Reflected (Get)*
- *HTML Injection – Reflected (Post)*
- *HTML Injection – Stored (Blog)*





COS'È BWAPP?

bWAPP è un'applicazione web open source e deliberatamente non sicura, che presenta numerose vulnerabilità al suo interno. Copre tutti i principali bug web noti, comprese le vulnerabilità del progetto OWASP top 10.

bWAPP è un'applicazione PHP che utilizza un database MySQL

INSTALLAZIONE BWAPP

- La macchina *bWAPP: bee-box* può essere scaricata attraverso il sito vulnhub, collegandoci al sito *sourceforge.net*

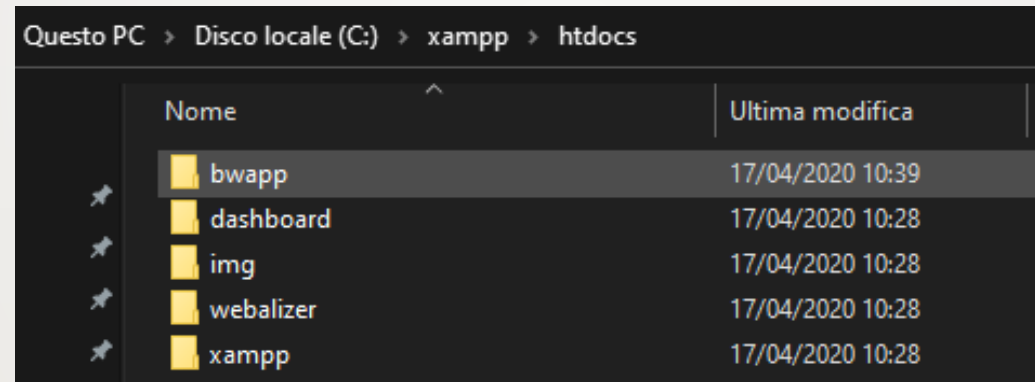
The screenshot shows the VulnHub page for the machine **bWAPP: bee-box (v1.6)**. The page has a navigation bar at the top with links: Back, About Release, Download, Description, File Information, Virtual Machine, Networking, Screenshot(s), and Walkthrough(s). Below the navigation bar, the title **bWAPP: bee-box (v1.6)** is displayed, followed by social media icons for Twitter, Facebook, and Email. The main content area is divided into two sections: **About Release** and **Download**. The **About Release** section contains the following information:

- Name:** bWAPP: bee-box (v1.6)
- Date release:** 2 Nov 2014
- Author:** Malik Mesellem
- Series:** bWAPP
- Web page:** <http://www.itsecgames.com/>

The **Download** section includes a disclaimer: "Please remember that VulnHub is a free community resource so we are unable to check the machines that are provided to us. Before you download, please read our FAQs sections dealing with the dangers of running unknown VMs and our suggestions for 'protecting yourself and your network. If you understand the risks, please download!" and provides the download link: **bee-box_v1.6.7z** (Size: 1.2 GB) with the download URL: http://sourceforge.net/projects/bwapp/files/bee-box/bee-box_v1.6.7z/download.

INSTALLAZIONE BWAPP

- Copiamo la cartella di «bWapp» e la inseriamo all'interno della cartella «xampp/htdocs»



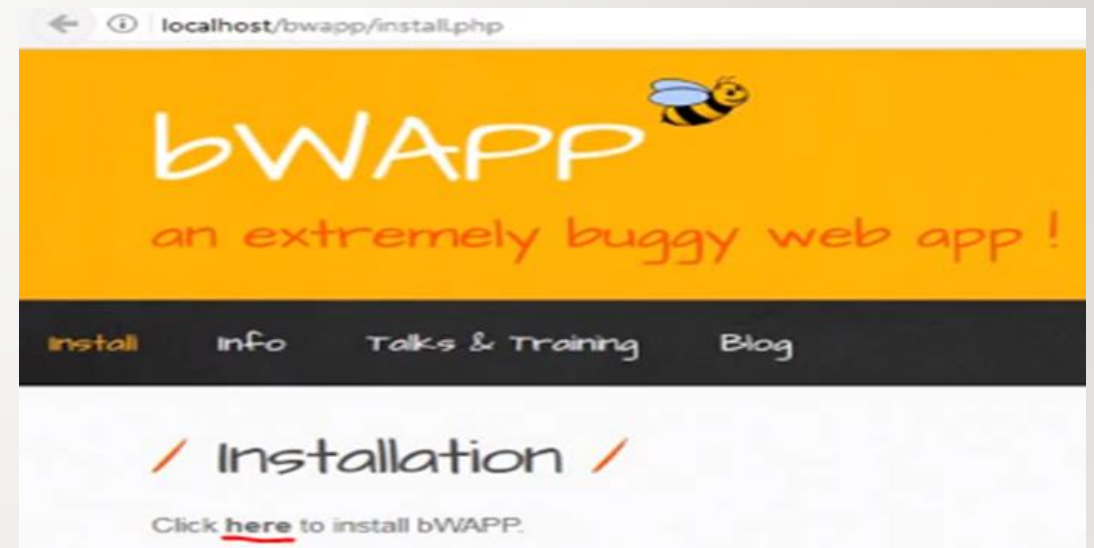
INSTALLAZIONE BWAPP

- Apriamo XAMPP ed attiviamo il servizio di MYSQL ed il sever APACHE.



INSTALLAZIONE BWAPP

- Accediamo al sito
localhost/bwapp/install.php
- Clicchiamo sul link di installazione
«[here](#)»

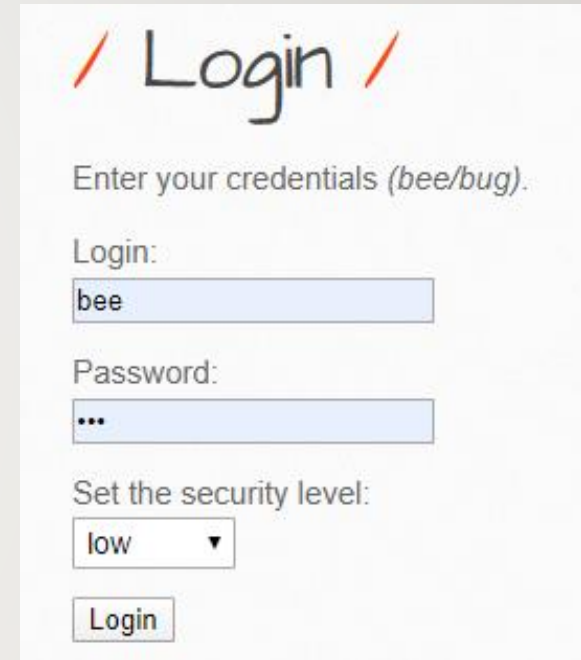


LOGIN

Per loggarci all'interno di bWAPP si utilizzano le credenziali:

- *Nome utente: bee*
- *Password: bug*

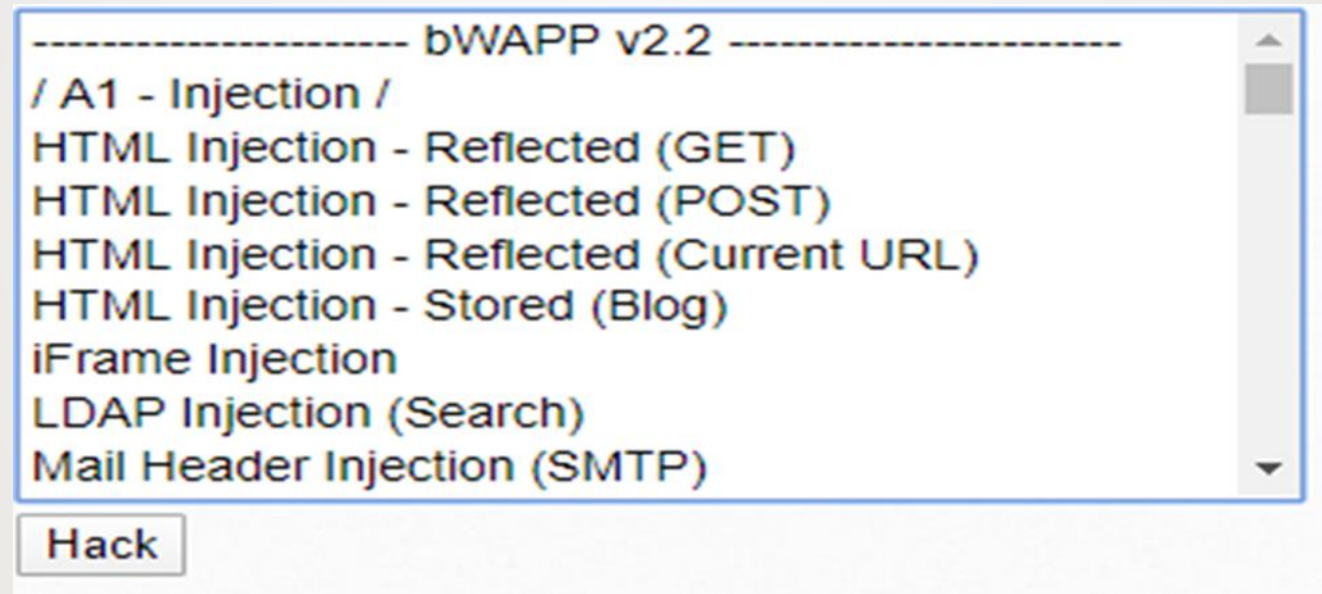
Possiamo, inoltre, settare il livello di sicurezza del sito, scegliendo tra tre valori disponibili: *low* / *medium* / *high*.



The screenshot shows the login interface of bWAPP. At the top, the word "Login" is displayed in a large, stylized font, flanked by two orange diagonal slashes. Below this, a prompt reads "Enter your credentials (bee/bug)". There are two input fields: the first is labeled "Login:" and contains the text "bee"; the second is labeled "Password:" and contains three dots. Below the password field, there is a section titled "Set the security level:" with a dropdown menu currently set to "low". At the bottom of the form is a button labeled "Login".

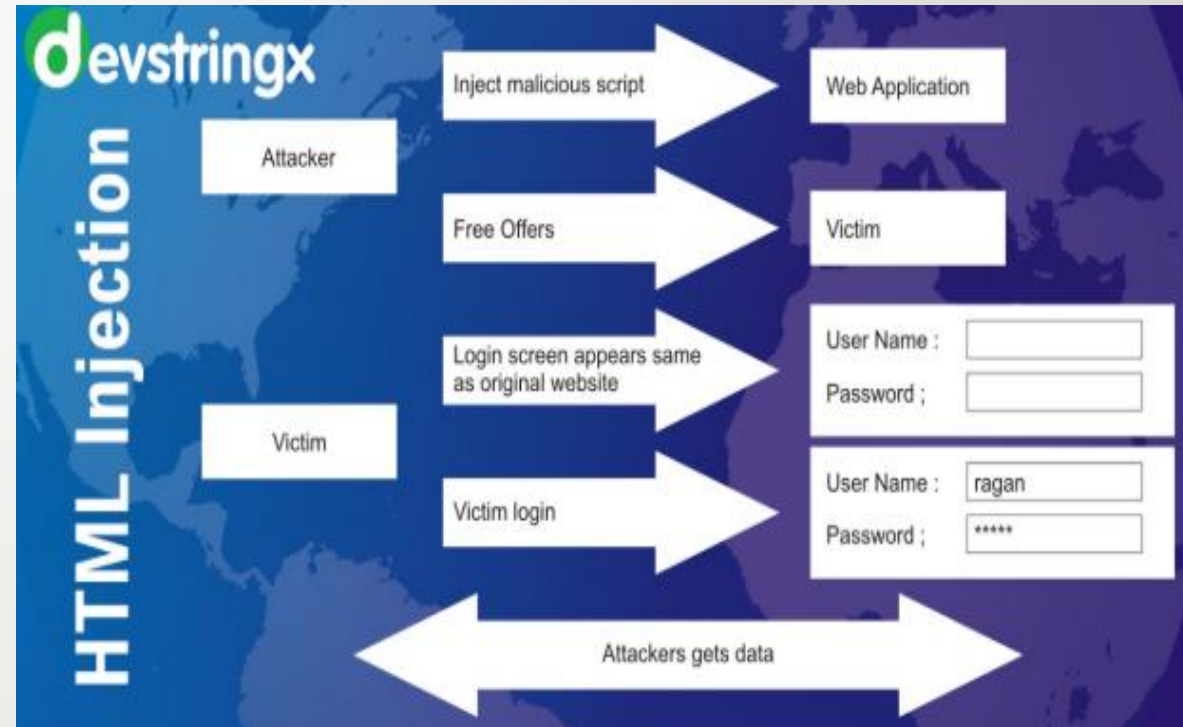
WHICH BUG DO YOU WANT TO HACK TODAY?

- Per la nostra sfida andremo ad approfondire l'attacco «HTML INJECTION»



HTML INJECTION

HTML Injection è una vulnerabilità simile a Cross-site Scripting(XSS) che si verifica quando l'input dell'utente non viene correttamente sanificato oppure quando l'output non viene correttamente codificato e l'attaccante è in grado di iniettare codice html valido all'interno di una pagina web vulnerabile.



SCENARIO DI ATTACCO

- L'attaccante scopre la vulnerabilità dell'iniezione e decide di effettuare un HTML Injection.
- L'attaccante genera un form e attende che l'utente (vittima) inserisca i dati.
- L'utente si collega alla pagina e inserisce i dati.
- Il codice HTML viene inviato all'attaccante.

TIPI DI INIEZIONE

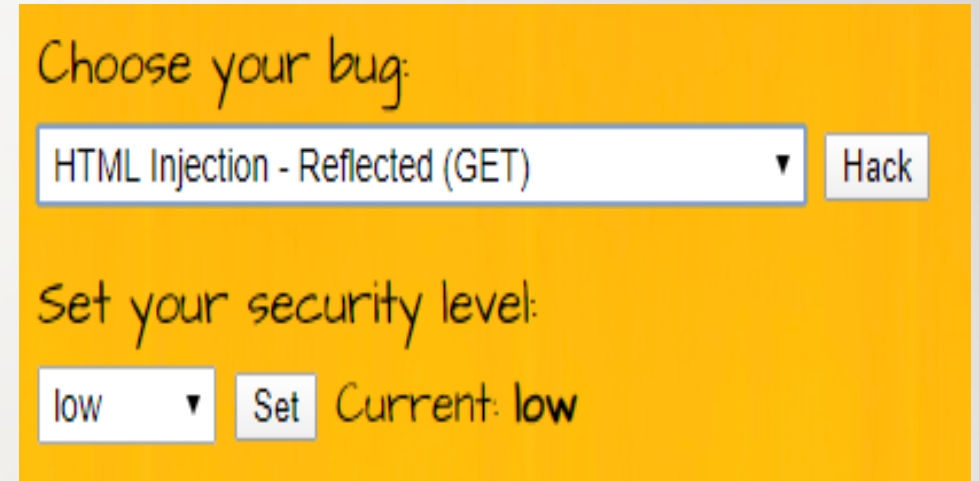
- **Reflected Html Injection:** il codice html non viene permanentemente salvato all'interno di un web server. L'attacco avviene quando il sito web risponde ad un input malevolo.
- **Stored Html Injection:** l'attacco avviene quando il codice html malevolo viene salvato all'interno di un web server e viene eseguito ogni volta che l'utente (vittima) chiama una funzione appropriata.

HTML INJECTION – REFLECTED (GET)

REFLECTED GET

Il metodo *GET* salva all'interno dell'URL i parametri che sono stati inseriti nel form.

- Selezioniamo all'interno del menù a tendina *HTML Injection – Reflected (GET)*.
- Selezioniamo il livello di sicurezza: *low*.



Choose your bug:

HTML Injection - Reflected (GET) ▼ Hack

Set your security level:

low ▼ Set Current: low



REFLECTED GET

Immettiamo l'input seguente nel form

Firstname: giacomo

Lastname: cocozziello

Quale risposta otteniamo?

REFLECTED GET

/ HTML Injection - Reflected (GET) /

Enter your first and last name:

First name:

Last name:

Welcome giacomo cocoziello

Nell'URL
compaiono i
parametri inseriti
nel form

`localhost/bwapp/htmli_get.php?firstname=giacomo&lastname=cocoziello&form=submit`

REFLECTED GET

Immettiamo l'input seguente nel form:

- *Firstname: jackcoco*
- *Lastname: <h1>Prova HTML</h1>*

Ottenendo il seguente risultato:

/ HTML Injection - Reflected (GET) /

Enter your first and last name:

First name:

Last name:

Welcome jackcoco

/ Prova HTML /

REFLECTED GET

Proviamo ad iniettare all'interno del form codice javascript e vediamo cosa restituisce la web page.

/ HTML Injection - Reflected (GET) /

Enter your first and last name:

First name:

Last name:

localhost dice
la pagina riflette gli input

OK

REFLECTED GET

Proviamo ad iniettare all'interno del form un codice javascript che ci stampi a video i cookie dell'utente.

/ HTML Injection - Reflected (GET) /

Enter your first and last name:

First name:

Last name:

localhost dice

PHPSESSID=sdv2sfaij8p615kqpdag3eps2m; security_level=0

OK

COSA ABBIAMO SCOPERTO?

Scegliendo *low* come livello di sicurezza abbiamo scoperto che possiamo inserire qualsiasi dato all'interno del form e la web page elabora i dati e li restituisce in output senza applicare alcun filtro. Ispezioniamo il codice sorgente «htmli_get.php»

```
function htmli($data)
{
    switch($_COOKIE["security_level"])
    {
        case "0" :
            $data = no_check($data);
            break;

        case "1" :
            $data = xss_check_1($data);
            break;

        case "2" :
            $data = xss_check_3($data);
            break;

        default :
            $data = no_check($data);
            break;
    }
}
```

In level low
non avviene
il controllo
sui dati

REFLECTED GET

Impostiamo il livello *Medium* e inseriamo nel form gli stessi input utilizzati in precedenza ed osserviamone i cambiamenti.

/ HTML Injection - Reflected (GET) /

Enter your first and last name:

First name:

Last name:

Welcome beep <h1>Prova HTML</h1>

In questo caso
l'input digitato non
viene elaborato e
visualizzato dalla
web page

MITIGAZIONE LEVEL LOW

La funzione `xss_check_1($data)` converte i caratteri `<` e `>` in un altro carattere, impedendo l'utilizzo dei tag html.

```
function xss_check_1($data)
{
    // Converts only "<" and ">" to HTML entities
    $input = str_replace("<", "&lt;", $data);
    $input = str_replace(">", "&gt;", $input);

    // Failure is an option
    // Bypasses double encoding attacks
    // <script>alert(0)</script>
    // %3Cscript%3Ealert%280%29%3C%2Fscript%3E
    // %253Cscript%253Ealert%25280%2529%253C%252Fscript%253E
    $input = urldecode($input);

    return $input;
}
```

REFLECTED GET

Poichè la funzione disabilita i tag proviamo ad utilizzare URL Encoding.

Encode to URL encoded format

Simply enter your data then push the encode button.

```
<h1>Prova HTML</h1>
```

> ENCODE <

Encodes your data into the textarea below.

```
%3Ch1%3EProva%20HTML%3C%2Fh1%3E%0A
```


/ HTML Injection - Reflected (GET) /

Enter your first and last name:

First name:

Last name:

Welcome jackcoco

/ Prova HTML /

MITIGAZIONE LEVEL MEDIUM

La funzione `xss_check_3` converte i caratteri `<`, `>` ed `&` (utilizzati negli esempi precedenti) in altri caratteri.

Come effetto della mitigazione non possiamo utilizzare l'URL Encoding.

```
function xss_check_3($data, $encoding = "UTF-8")
{
    // htmlspecialchars - converts special characters to HTML entities
    // '&' (ampersand) becomes '&amp;'
    // '"' (double quote) becomes '&quot;' when ENT_NOQUOTES is not set
    // "'" (single quote) becomes '&#039;' (or &apos;) only when ENT_QUOTES is
    // '<' (less than) becomes '&lt;'
    // '>' (greater than) becomes '&gt;'

    return htmlspecialchars($data, ENT_QUOTES, $encoding);
}
```

/ HTML Injection - Reflected (GET) /

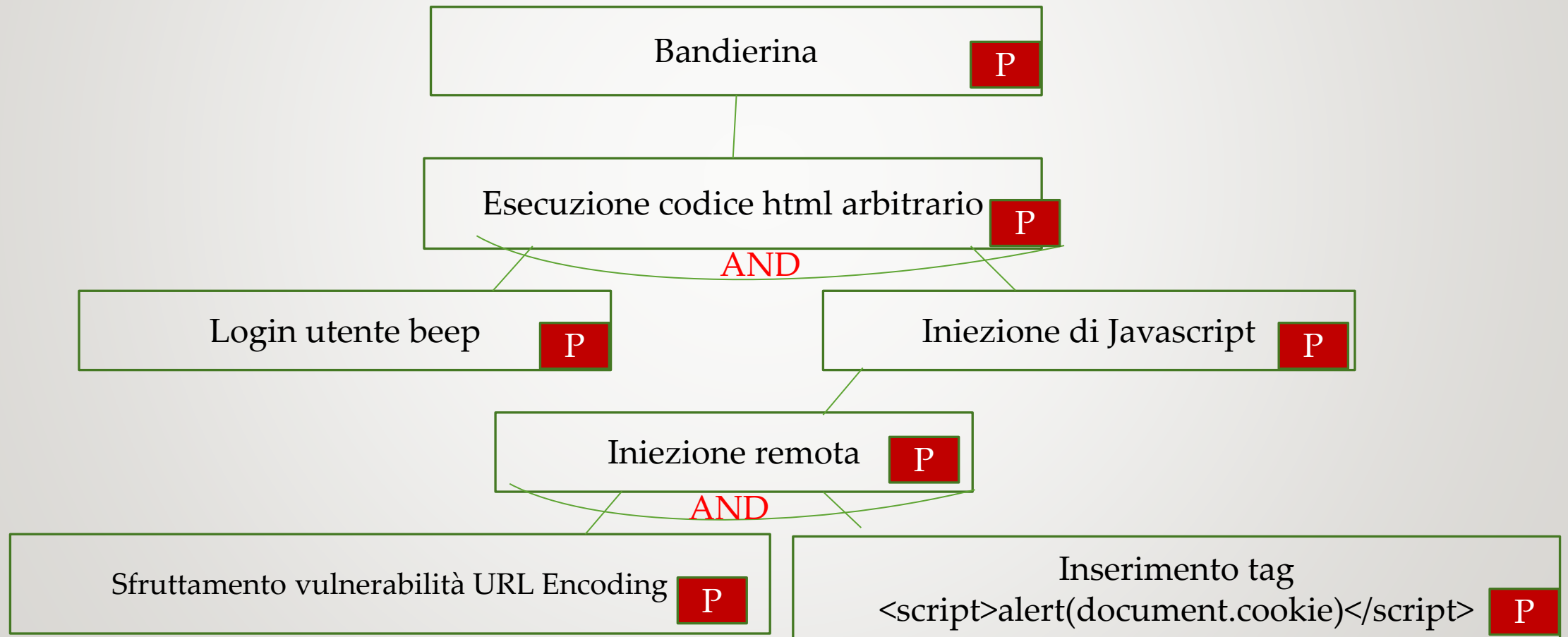
Enter your first and last name:

First name:

Last name:

Welcome jackcoco %3Ch1%3EProva%20HTML%3C%2Fh1%3E%0A

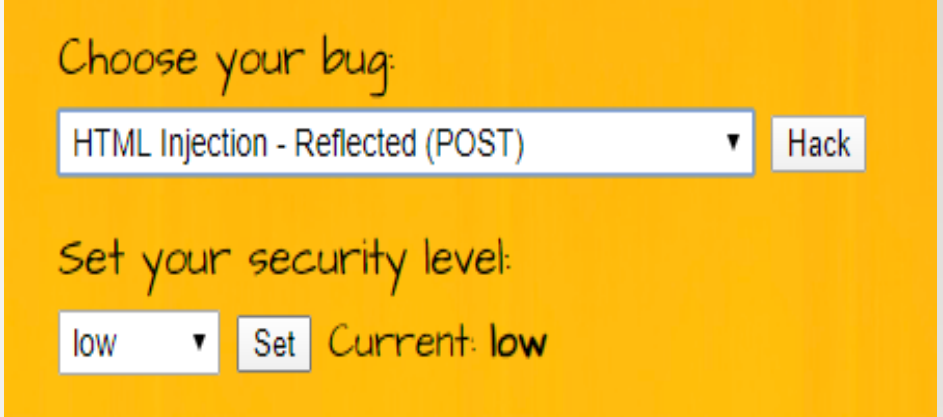
HTML INJECTION REFLECTED GET ALBERO DI ATTACCO



HTML INJECTION – REFLECTED (POST)

REFLECTED POST

- All'interno di una richiesta *POST* i parametri non vengono visualizzati all'interno dell'URL e quindi non possono essere tracciati negli access log dei web server.
- Ad esempio, viene utilizzato in un form che contiene dati personali, come username e password.



Choose your bug:

HTML Injection - Reflected (POST) ▼ Hack

Set your security level:

low ▼ Set Current: low

REFLECTED POST

Immettiamo l'input nel seguente form:

- *First name: beep*
- *Lastname:*

```
<html>
<head>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
<p>Benvenuti ragazzi</p>
</body>
</html>
```

/ HTML Injection - Reflected (POST) /

Enter your first and last name:

First name:

Last name:

Welcome beep

Benvenuti ragazzi

Notiamo che
nell'URL non
compaiono i dati
inseriti nel form.

localhost/bwapp/htmli_post.php

REFLECTED POST

La sfida HTML Injection – Reflected (post) presenta le stesse vulnerabilità e mitigazioni della sfida HTML Injection –Reflected (get).

HTML INJECTION – STORED (BLOG)

REFLECTED STORED (BLOG)

Con questa iniezione possiamo iniettare codice html all'interno del database di una web page e recuperarlo in seguito quando è necessario.

Choose your bug:

HTML Injection - Stored (Blog) ▼ Hack

Set your security level:

low ▼ Set Current: low

/ HTML Injection - Stored (Blog) /

Submit Add: ☒ Show all: ☐ Delete: ☐

#	Owner	Date	
---	-------	------	--

REFLECTED STORED (BLOG)

/ HTML Injection - Stored (Blog) /

Add: ☒ Show all: ☐ Delete: ☐ Your entry was added to our blog!

12	bee	2020-04-18 17:48:20	/ Hello World! /
----	-----	---------------------	------------------

Il tag inserito nel form viene aggiunto al database della web page

REFLECTED STORED (BLOG)

Creiamo una schermata di accesso duplicata in cui un utente malintenzionato può indurre gli utenti vittima ad inserire i propri dati.

```
<form action="/bWAPP/htmli_post.php" method="POST">

    <p><label>Nome:</label><br>
    <input type="text" id="firstname" nome="firstname"></p>

    <p><label>Cognome:</label><br>
    <input type="text" id="lastname" nome="lastname"></p>

    <button type="submit" name="form" value="submit">GO</button>
</form>
```

REFLECTED STORED (BLOG)

/ HTML Injection - Stored (Blog) /

```
<form action="/bwAPP/htmli_post.php" method="POST">
  <p><label>Nome:</label><br>
```

Submit Add: ☒ Show all: ☐ Delete: ☐ Your entry was added to our blog!

#	Owner	Date	
13	bee	2020-04-18 17:58:54	<div>Nome: <input type="text"/></div> <div>Cognome: <input type="text"/></div> <div>GO</div>

Quando l'utente inserisce i dati nel form l'attaccante li acquisisce.

REFLECTED STORED (BLOG)

Dopo aver compilato il form l'utente passerà alla pagina post:

```
localhost/bWAPP/htmli_post.php
```


REFLECTED STORED (BLOG)

Un utente malintenzionato potrebbe inserire un form per caricare al suo interno file dannosi. Se non viene configurato un firewall (Web Application Firewall) avremo accesso all'intero sistema.

Andando ad inserire un file più grande di 41 MB otteniamo una eccezione.

17	bee	2020-04-19 10:21:56	CAMPO TESTO <input type="text"/>	FILE <input type="button" value="Scegli file"/>	Nessun file selezionato	<input type="button" value="Upload"/>
----	-----	---------------------	----------------------------------	-------------------------------------------------	-------------------------	---------------------------------------

Warning: POST Content-Length of 160497758 bytes exceeds the limit of 41943040 bytes in **Unknown** on line 0



MITIGAZIONE LEVEL LOW

Settando il livello medium e inserendo lo stesso codice inserito in precedenza, notiamo che non è più funzionante.

/ HTML Injection - Stored (Blog) /

```
<form action="/bWAPP/htmli_post.php" method="POST">
  <p><label>Nome:</label><br>
```

Submit

Add: ☒

Show all: ☐

Delete: ☐

Your entry was added to our blog!

19	bee	2020-04-19 10:41:16	<form action="/bWAPP/htmli_post.php" method="POST"> <p><label>Nome:</label> <input type="text" id="firstname" nome="firstname"></p> <p><label>
----	-----	---------------------	----------------------------------------------------------------------------------------------------------------------------------------------------

MITIGAZIONE LEVEL LOW

Proviamo ad utilizzare l'URL Encoding (come eseguito in precedenza) e notiamo che non funziona.

Questo avviene a causa del fatto che già è abilitata la funzione `xss_check_3`, quindi, tutti i caratteri speciali vengono disabilitati.

20	bee	2020-04-19 10:54:08	%3Cform%20action%3D%22%2FbWAPP%2Fhtml_post.php%22%20method%3D%22POST%22%3E%0A%0A%09%3Cp%3E%3Clabel%3ENome%3
----	-----	---------------------	-------------------------------------------------------------------------------------------------------------

MITIGAZIONE LEVEL LOW

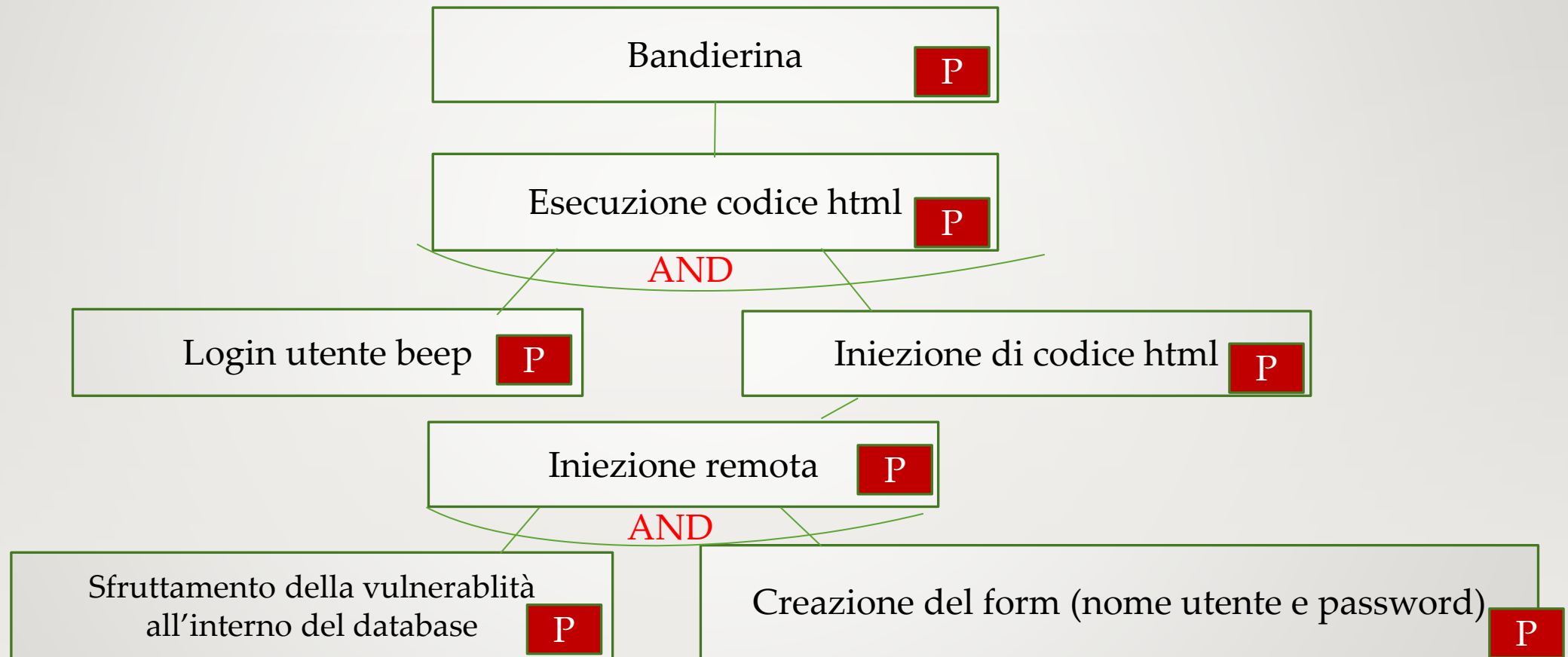
All'interno del codice sorgente «htmli_stored.php» notiamo che viene utilizzata la funzione `xss_check_4`. Tale funzione può essere ispezionata all'interno del file «function_external.php». Notiamo che restituisce una stringa con «\» prima dei caratteri che devono essere quotati all'interno del database.

```
function xss_check_4($data)
{
    // addslashes - returns a string with backslashes before characters that need to be quoted in database queries etc.
    // These characters are single quote ('), double quote ("), backslash (\) and NUL (the NULL byte).
    // Do NOT use this for XSS or HTML validations!!!

    return addslashes($data);
}
```

HTML INJECTION STORED BLOG

ALBERO DI ATTACCO



CWE-79: IMPROPER NEUTRALIZATION OF INPUT DURING WEB PAGE GENERATION

- *Descrizione*: il sito web non neutralizza l'input dell'utente, quindi, un utente malintenzionato potrebbe inserire codice malevolo ed eseguirlo.
- *Conseguenze*: CIA Trade compromesso, in quanto l'attaccante potrebbe eseguire codice arbitrario all'interno del computer della vittima creando problemi.
- *Access Control Confidentiality*: l'attaccante potrebbe rubare dati degli utenti, dai cookie o dal sito web.

CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

Weakness ID: 79

Abstraction: Base

Structure: Simple

Status: Stable

Presentation Filter:

▼ Description

The software does not neutralize or incorrectly neutralizes user-controllable input before it is placed in output that is used as a web page that is served to other users.

CWE-80: IMPROPER NEUTRALIZATION OF SCRIPT-RELATED HTML TAGS IN A WEB PAGE

- Il sito web neutralizza in modo errato caratteri speciali come <, > e & che potrebbero essere interpretati come elementi di web scripting.

Possibili mitigazioni:

- 1) Utilizzo di una White List, che non solo controlla gli input visibili (tag specificati nella form) ma tutti i parametri non visibili nella richiesta, inclusi campi nascosti, cookie, intestazioni e url stesso. Pertanto si consiglia di convalidare tutte le parti della richiesta HTTP.

CWE-80: IMPROPER NEUTRALIZATION OF SCRIPT-RELATED HTML TAGS IN A WEB PAGE

2) *Codifica di output*: utilizzare una codifica di output. Le codifiche comuni includono ISO-8859-1, UTF-7 e UTF-8. Se le codifiche sono incoerenti l'attaccante può essere in grado di sfruttare questa discrepanza e condurre attacchi di iniezione.

Tale problema si presenta spesso nelle pagine web. Se una codifica non è specificata in un'intestazione HTTP, i browser web indovinano quale codifica viene utilizzata.

CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)

Weakness ID: 80

Abstraction: Variant

Structure: Simple

Status: Incomplete

Presentation Filter: ▼

▼ Description

The software receives input from an upstream component, but it does not neutralize or incorrectly neutralizes special characters such as "<", ">", and "&" that could be interpreted as web-scripting elements when they are sent to a downstream component that processes web pages.

[illegible]