

By: igal neman

Id: 300369972

Date 26.8.19

Report: DOM XSS (Cross Site Scripting), RFI

Background:

Cross-site scripting or XSS for short is one of today's most common attacks at the application level to attack users, by infiltration into web applications.

The attack causes and affects the privacy of web surfers infected and could lead to a violation of user privacy, the details of which are stolen or treated

Unlike most attacks, which have two sides to them - Validity and website attack XSS involves three parties - valid client (victim) and the Web site of the injury.

Attack XSS is a type of injection where malicious scripts injected into Web sites considered to be reliable. The attack occurs when an attacker manages to upload the victim of malicious code, usually in the form of a script is run on the browser, and this code is dropped and is run in browsers of unsuspecting end users. Flaws allow this attack to succeed are quite common and can occur anywhere a web application uses user input without validation or encoding appropriate, and does not recognize that the uploaded code and data input.

The victim's browser has no way of knowing that the script passed to it is from a trusted source, and the browser will continue to progress and will run the script

Operating, malicious script can access any cookies in your browser, information on contracts, or other sensitive information saved by the browser being used against the targeted site and rewrite the contents of the HTML page.

Difficult to defend against XSS fully and must be aware of their repercussions on the different expressions while writing code, to recognize the attack and the ways in which it operates to write safe code to protect the site and browsing visitors.

Modern Web applications are pages Static HTML.

They are usually full of dynamic content changes

They pull data and information from many different sources. This information is incorporated with the Web page and displayed surfer web pages by the browser, it can contain simple text or images and can include tags Like HTML `<p>` paragraph `` image and `<script>` script.

AttackA typical XSS attacker infecting legitimate Web page by injecting its malicious script input field is not set correctly. Random surfer enters an infected page, the browser downloads and runs the script is injected, causing unwanted behavior. For example, a simple post on your Facebook wall may contain malicious script injected into it. If not filtered by Facebook's servers, while visiting an infected profile of skier, dropped the script and will play the naive visitor's browser. It is important to emphasize that this type of attack the attacker can take actions on the service user by utilizing the protocol restrictionsHTTPAnd stealUser ID.) shadow)

When Vulnerabilities is 'temporary':

1. Typically, the breach of theProvisional we seem XSS Web site addresses when addresses dynamically.
2. Dynamic address would be `Http://mysite.com/index.php&search=hello` - at this we do a search for the string "hello", and it uses a method named GET Sending its data, according to which the parameters are passed through the URL (for example: search is the parameter, indicating that the page index.php some string used for search. In this case, the user decided to look for hello).
3. Option why display to the user when such a search is: "I looked for the stringhello, and here are the results. " We see event output is clearly tied to a string Shznno, and thus play a bit with the address.
4. Change the address to`<Http://mysite.com/index.php&search= <script> alert ("hello") </ script>`. What I put the search parameter is actually a short script in Java Script aims is to squeeze a user.
5. Entered the address, and if the user is shown an error message that says hello, we know for sure that the site is vulnerable.

Many think "Why is vulnerable, it is shown only for us" - and therein lies the problem We can send such a link to another user on the same site as the parameter search can put a script he could pull the cookies and send them directly to our Cookie Stealer. When the user clicks on the link (it's really not such a difficult thing to do with a littleSocial engineering)

- When referring to RFI my intention that it includes the ability to access remote files (RFI- RemoteFile Inclusion):
- The process inclusion of remote files, occurs when a page receives as input the path to the file to be included, but the input is not filtered and verified properly and allows injection of external address and the operation of remote files instead of the original input.

Document Object Model Cross Site Scripting Based: (DOM XSS BASE)

There is usually a client side script that uses the input to produce PageHTML, Without ensuring that the input does not contain malicious code.

For example, code JavaScript that gets URLAs input, and using it to produce HTML page, without confirming that the address itself does not contain code, contain XSS breach locally.

Unlike the previous two, where vulnerability is in-code side server) lack in validation input (here vulnerability is running in the browser code.

Examples:

Link verify banking application created with malicious code integration link and send it by email assailed the

The victim received an email, click on the link, the user sends the request to the script lies in the Bank's website

He did not recognize it and thus sends response with a script and executed by the browser of the victim.

Run the script browser sends the victim-cookie- and info session to attacker

Here's a picture illustration shows us the vulnerability

JavaScript injection (DOM-based)

| | |
|-------------|----------------------------------|
| Issue: | JavaScript injection (DOM-based) |
| Severity: | High |
| Confidence: | Firm |
| Host: | http://172.16.67.136 |
| Path: | /WebGoat/attack |

Issue detail

The application may be vulnerable to DOM-based JavaScript injection. Data is read from **document.URL** and written to **eval()** via the following statements:

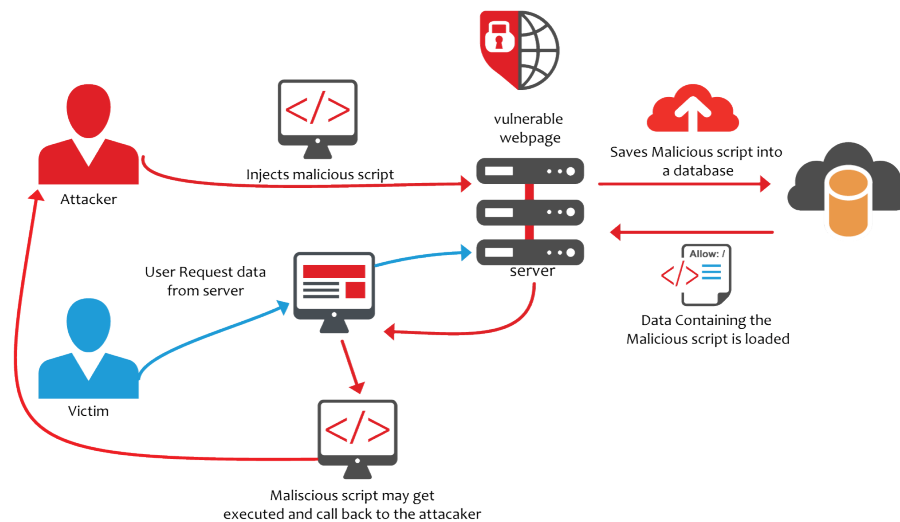
- `ur=document.URL;`
- `pr=ur.substring(x+1,ur.length).split("&");`
- `nv=pr[i].split("=");`
- `mn="menu"+unescape(nv[1]);`
- `eval("trigMenuMagic1("+mn+"",""+opt+"")");`

Objectives and importance of our test

Overall this is an attack with a high level of risk. And if the problem 'loophole' that is not dealt \ will be corrected as soon as possible, force that has the potential likelihood is very high, and almost certainly will be able to \ break and \ or exercise break from your site and the way to steal information from the database of the organization, as well as the attacker could also attack innocent people network thanks to malicious code implanted on the server

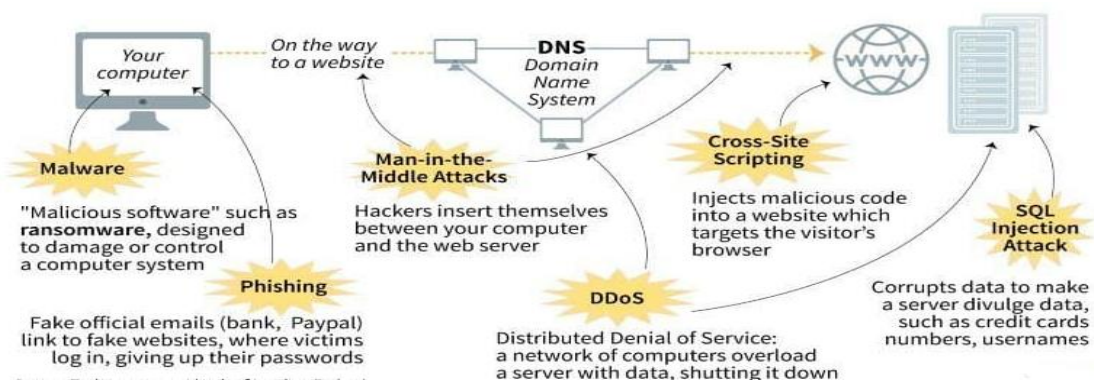
Below is (Design pattern))(Unprotected web site)

Example :to us how and how all this actually happens execution of the attack in terms of realization (Photo illustration)



As we can see, the attacker connects to the server, and plants malicious code. After a user enters the website linked to the innocent and malicious code drops it to a computer automatically without the user's knowledge and performs EXECUTE (running) behind the scenes.

An attacker has control over the DATABASE database as well as the computer of someone infected with the malicious file listing page.



As you can see from the chart above, a potential attacker might realize the security breach \ Wow if he had made it that were not protected, the period can realize through your server attacks PHISHING, DDOS attacks are driving service.

An attacker could also enter queries in SQL and steal information from the database, such as:

Username and passwords or credit card details

Report summary:

- In view of the kind of problem is this, the recommendation is: to consider the problem of this nature very serious and treat it urgently and does not reject it because Helfrich security of this type has the potential for devastating high Whether it is the organization and whether it's on unsuspecting users on the network, failure and that accordingly, the recommendation is not to delay treatment of this topic

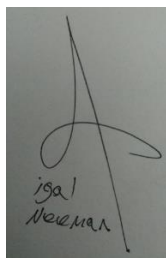
Recommendations for implementation: Whether the site is secure or not should be three circles

1. Prevention - writing code in a secure site, where all input fields are validated and coding

e.g long-term prevention process is to advise the organization to perform input control which means that they can not enter invalid values;

2. Protection - Using protection; such as WAF, IPS, backup site and preparing an action plan for recovery.
3. Testing process usually takes place by - site assessment scanning and vulnerability detection, using software or
4. automated cloud services, or reliance on testing the strength of external factors

Thanks,

A handwritten signature in black ink on a light background. The signature is stylized, with a large loop and a long horizontal stroke. Below the signature, the name "igal neman" is written in a smaller, simpler font.

igal neman.