Advantages and disadvantages of optimisation techniques

Technology, Computer

We have to use different techniques to optimise our website. Our main focus will be on how to reduce the size of the webpage and script execution time. This, in turn, will definitely improve the user experience as well as reduce the load on the servers. The images and graphical interfaces occupy the maximum storage size on the webpage. Therefore we need to compress the images and the scripts in an effective manner that won't affect functionality and quality of the website. First we need to simplify the design of the website, reducing the number of HTTP requests. JavaScript and CSS can be minified by removing comments and space characters from the code. Furthermore we can make java script and CSS external which can be cached by the browser. Post Load and Pre Load components can be set, which will make the user experience better by initially loading the important content and then loading the remaining content. Ultimately we can optimise loops which will reduce the script execution time.

Specific Optimisation Techniques to be used for the Website

The different optimisation techniques are:

Minimise HTTP requests

Add an expiry or a cache control header

Put scripts at the bottom

Remove duplicate scripts

Make Ajax cacheable

Post load components

Pre load components

Make JavaScript and CSS external

Reduce DNS Lookups

Minimise HTTP requests

The majority of the time taken when a page is loading is a result of HTTP requests. Reducing the number of components will thus reduce the number of HTTP requests required to render the page. This is the key to faster loading pages.

Combined files are a way to reduce the number of HTTP requests by combining all scripts into a single script, and similarly combining all CSS into a single stylesheet.

Discussing the decisions for each page

We will use some techniques which will be applicable on all the pages. Style sheets will be placed at the top and scripts at the bottom of the webpage. AJAX, JavaScript and CSS made cacheable. CDN servers will be used and redirects, 404 errors, and empty image source avoided.

GET will be used for AJAX requests and also make efficient use of pre and post load components. The complete JavaScript code and CSS will be minified.

The Home page is the one which should be fast, attractive and interactive. The Home page will make up the rapport of your website. In order to make it attractive we need to use different images, CSS, graphical interfaces and flash content. Images and graphical interfaces are bigger in size so we will scale them and try to convert them into a more suitable format such as Portable Network Graphics (. png).

The user’s login page, discussion page, upload page and modify page will be much simpler than home page but they will contain lots of videos and images with the description. Thus, we can just remove the white space characters from their code reducing the size of the webpage and optimising the images. Videos are too big in size so we will not upload it on our server. We will use YouTube or other free video upload sites and embed the link on our webpage.

Product Catalogue will contain all the images of the product that can be searched and sorted by category and brands. The loops will be optimised for faster search and sorting.

Business information, current employees and product information will contain the detailed textual description with images of what we do, what we sell and who works in our company. All this information is static, so we will cache the information and optimise the images.

Q2. Client side security issues are an extremely important component of any web-based application.

Introduction

Client side security is one of the most important topics in internet security. All the information which has been downloaded from servers is stored on the client’s machine. All the site preferences as well as your login details are stored as cookies on the local machine and we need to keep those files safe from hackers. We use different antiviruses and firewalls on the local machine, however they are not as efficient as they should be.

JavaScript and AJAX are the most vulnerable languages for most of the current web-based exploits like Trojans, viruses, etc. [1]

(Uta Priss, 2012, Advanced Client-Side Security: What many users do not know, From http://www. upriss. org. uk/awt/lec4a. pdf)

In this document we will look at the major threats, type of client side attacks and some strategies for minimising those risks.

Problem Domain

Nowadays internet is a basic necessity of day to day life. We are so dependent on internet these days. Everything from paying our electricity bills to international business meetings we do online. All our bank details are on our local machine which is vulnerable to hackers. For this reason do we need a secure system to work safely online. Whenever we use internet our local system stores the information from the server which contains your preferences, form data and the history of the webpages you viewed.

How do hackers operate..?

Hackers try to bypass the firewall and gain inappropriate access to local host resources. There are more chances to breach the security of the firewall when the hacker and the host are on the same network because request for resources originating within the network can be trusted more than request originating from outside the network. (Bidgoli Hossein , Wiley John & Sons, (2006) Hanew Jersey, John Wiley & Sons)

Issues

One of the most widely used languages for website development, JavaScript, is not secure. JavaScript is an open scripting language which means anyone can manipulate it and change its function. The JavaScript security model attempts to protect the user from websites that may be malicious and is not designed to protect the website owner. It can’t protect data sent from the browser to the server and there are limits on what the page author can control via JavaScript whilst it is being executed within the browser. The success of JavaScript is also however the reason why attackers have targeted and leveraged the technology as a means to compromise the systems and reek untold grief for clients. JavaScript has been used to perform attacks that involve redirects, downloading of content, or even revealing details about a victim’s system.[2] Now we will discuss some attack strategies such as XSS (Cross Site Scripting), CSRF (Cross Site Request Forgery) and introduce some prevention measures to improve the security of the website.

XSS

Cross-Site Scripting attacks are a type of injection problem, in which malicious scripts are injected into the otherwise benign and trusted websites. Cross-site scripting (XSS) attacks occur when an attacker uses a web application to send a malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user in the output it generates without validating or encoding it.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user’s browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by your browser and used within that site. These scripts can even rewrite the content of the HTML page.

Prevention measures:

Validate, filter, and sanitise all input

Process output response stream data through encoding

Many modern browsers will attempt to detect an XSS attack and notify the user

CSRF

CSRF (Cross-Site Request Forgery) is an attack which forces an end user to execute unwanted actions on a web application in which he/she is currently

authenticated. With a little help of social engineering (like sending a link via email/chat), an attacker may force the users of a web application to execute actions of the attacker’s choosing. A successful CSRF exploit can compromise end user data and operation in the case of a normal user. If the targeted end user is the administrator account, this can compromise the entire web application.

Prevention measures:

Implement strong XSS mitigations

Use Tokens to verify expected user actions

Hidden form value fields

E. g., RoR & ASP. Net MVC provide framework support

Use POST for any actions that alter data on server side

Is the idempotent web paradigm for HTTP GET compromised?

Check HTTP Referrer

Most modern browsers include features to palliate the following attacks:

Phishing Attacks

Spyware

Malicious websites

Adware

Destruction or corruption of data or configuration

Theft of configuration information

Installation of malware

Theft of information and identification

(Oriyano Sean-Philip and Shimonsk Robert,(2012)Client Side Attacks and Defense, USA, Elsevier, Pg 130)

Conclusion

In summary, we discussed the major client side scripting attacks, preventive measures and the most vulnerable languages. Client side scripting attacks are effective in taking the personal information of the user. However, if we play smart we can extenuate and avoid those attacks in the first place by making some changes in our firewall settings and not clicking on suspicious links. Prevention is always preferred over a cure for the problems being faced for a normal user in this unprotected web environment. As programmers, we should make appropriate use of the AJAX commands and code the website in such a way as to make it more reliable and harder to alter. Lastly, JavaScript is the most popular language and will remain so in coming years. In light of this, we should make efficient use of the primary functions and the libraries to make it less vulnerable to attacks.