Patryk Tomaszewski

Web 4

# Client side threats

## ClickJacking

Clickjacking is a way of „stealing” the user’s clicks. What that means is, the user voluntarily clicks on an interactive element on a website, such as a button or a link, which in turn redirects the user to some malicious code which can for example, steal their account information or even transfer some money onto the attacker’s account. It’s a medium risk attack, as the victims, mostly those who aren’t technologically savvy in any way need to first get tricked into entering a website that represents the “original” website and using it as if it was the real website. Social Engineering is the term that’s used for it. Although most users know how to recognise a fake website or a possibly malicious email it doesn’t mean that there aren’t people who can’t fall for it.

The simplest way to protect against any clickjacking attempts is to modify the apache configuration to disable rendering iframe and/or object tags. This is the line that needs to be added into your configuration file to prevent clickjacking: (Lam, 2013)

Header always append X-Frame-Options DENY

## HTML Injection

HTML injection, as well as JavaScript injection is one of the simplest attacks a malicious person could perform when not protected against. It’s as simple as entering JavaScript code inside a box in an HTML form. The user could type something like <script> alert(“hi”) </script> in a text box on a website and the browser would execute it. The most vulnerable websites are essentially any websites that allow any comments or simple form usage on them. JavaScript could be sued to extract confidential info from the website.

To prevent those kinds of attacks, try to avoid using client-side validation, and instead use server-side validation as JavaScript might be manipulated using inline JS injections. Do not use cookies to store confidential information and do not use hidden inputs on the webpage to store values as those can be easily found and extracted using JavaScript (Skynet\_Code, 2010)

## Client-Side Cookies

Cookies are used to store data about the user on their machine so that It can be used to enhance their next visit to the website. It’s a good feature however, it can easily be exploited, especially when the data that gets stored reveals personal information or card/bank details. For example, if the cookie stored data about the current session data gets stolen by a hacker, then that hacker can paste the session data into their browser and trick the website into thinking that they were the ones who originally submitted the request, thus giving them access to all user information. Even if data like the user’s name or anything personal gets retrieved by a hacker this gives them the opportunity to social engineer you and possibly making you send them money. (Hallett, 2011)

As a webmaster, to protect your users you should make sure that your website does not gather any personal or session info and stores them into cookies, make sure that information is securely handled by server-side scripts. As a user, you can check your local machine to see what kinds of information gets stored in cookies from websites, if you notice anything suspicious you should stop using that website or disable the usage of cookies in the web browser of your choice.

## Form Autocomplete

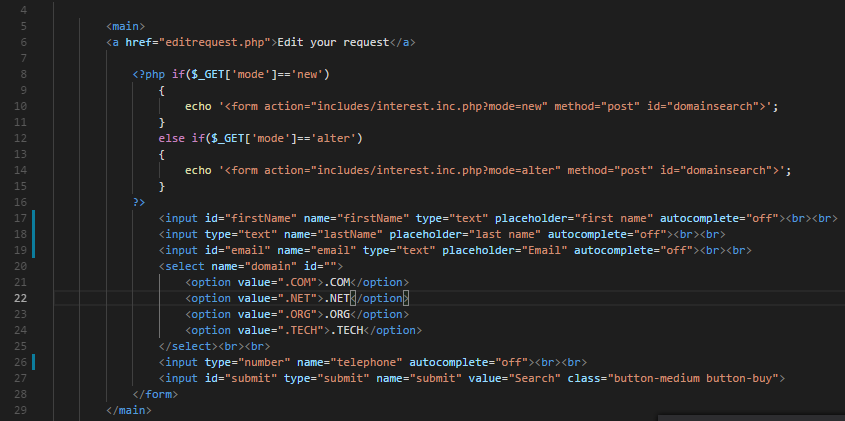
Autocompletion of forms can be harmful as whenever a user inserts some data into a website and submits it, the browser usually asks if the user wishes for the browser to save their login details for later use. Even though that can’t be accessed by hackers remotely, it is still a huge risk as any user who has local access to the person’s computer can just double click the field or simply reload the webpage and the victims login info, including the password, will get pasted into the fields, so all the other person needs to do access their account now, is simply press enter

The solution to prevent users from being able to use autocomplete on your website is to specify so in the HTML code like this:

<input type="password" autocomplete="off">

If you had more inputs, you’d have to make sure that the autocomplete attribute is set to off for every input.

Here is my fasthosts code that shows how I prevent form autocomplete myself:



# Server-Side Threats

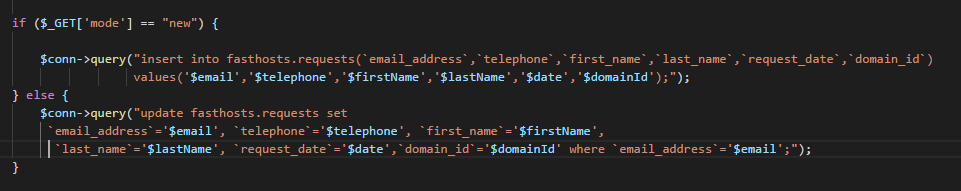
## SQL Injection

An SQL injection attack is quite like the HTML/JavaScript injection attack. The attack works as follows: When a user uses a input box on a form, and that form’s data gets used in an SQL query then the user could try writing SQL code in the field instead of the actual values in order to trick the server into executing a malicious query which could select the admin password, meaning that the user would have access to the admin account and therefore all other user data.

One way of preventing an SQL injection attack, in most cases, would be to prevent the users from entering special characters that can be used as SQL syntax, such as semicolons, quotes etc. Another way of doing it would be to use prepared statements instead of dynamic queries.

Here is my code now vs with prepared statements:

Without prepared statements:



With prepared statements:



## Directory Traversal Attack

Directory Traversal Attack is an HTTP attack which uses vulnerable websites that use dynamic website generation to display content to the user to gain access to files outside of the root directory. For example, if the website used a get parameter in the address bar to show a certain file, then an attacker could exploit this by simply typing “ ../../../filename ” to go down the directory and access files outside of the root directory in order to gather information that can be used to exploit the system.

To prevent Directory Traversal Attacks make sure that you properly secure user input, and make sure that all the system software is updated, as the attackers tend to look for outdated system exploits first. Also make sure you don’t enable navigating around the website using get method parameters. (acunetix, 2018)

## Source Code Disclosure

This kind of attack happens when the attacker can modify the requests sent by the website to download files that normally they wouldn’t be supposed to access. For example, if the website had a get request which had the file name as the parameter, and then that certain file would get downloaded. The attacker could essentially modify the request and insert the name of a website file, such as the index.php file and gain access to the server-side code which the attacker could then use to exploit the website.

To prevent these kinds of attack, avoid dynamically generated content to decide which file to download, and instead use a post request to do so, do not hardcode the filename in the client-side code either as a variety of JavaScript attacks could be executed to still gain access to the files. (BRIGJAJ, 2016)

## Local File Inclusion

This attack is quite like the previous 2 attacks I mentioned. This attack works when the website uses a get request to order the server to dynamically include a script file into another script file. Although this wouldn’t be dangerous if it as was hardcoded in the source code, it is very dangerous as an attacker can use this kind of attack to include and therefore execute a malicious script they could have possibly managed to upload onto the server.

The simplest way to prevent Local file Inclusion attacks is to avoid dynamic file included in general, or specify which files to download in the script. (Muscat, 2017)

# Threat Analysis

Threat Analysis is basically finding out information about various risks, and to be more specific, those risks needs to be related to websites. You will need to Identify all risks, and then consider what impact those would have on your website (cost wise as infrastructure wise as well) and then manage your costs effectively to get rid of the most dangerous risks, if not all.

There are two kinds of risk analysis: quantitative risk analysis and qualitative risk analysis.

Quantitative risk analysis involves assigning numerical values to things as it involves estimating potential costs associated with various losses such as, theft or physical damage to the equipment as well as damage to the stored data itself. It also involves estimating the probability of something happening as well as how often it could happen which leads to estimating the cost of any countermeasures and the expected loss reduction.

Qualitative risk analysis on the other hand, consists of outlining potential threats and outlining the use of countermeasures based on the most harmful risks found. (Shoenfeld, 2000)

In case of the fasthosts company, the most important thing would be the data stored on the server as the business stores not only user information, but also the domains they have requested, and the business needs those to make necessary transactions to fulfil the user needs. Not only the user data is important to protect. The entire server’s filesystem is, as if it got hacked, then a lot of crucial software would be lost.   
The attacks which can cause the above-mentioned damages are SQL Injections, HTML/JS Injections, Directory Traversals and Local File Inclusions. The most dangerous attacks out of these are SQL Injections and HTML/JS injections because they are quite easy to perform when the system isn’t secure and can be used to extract virtually any kind of data. Directory Traversal is also quite dangerous as the hacker will have access to the entire system, not just the root folder and therefore will be able to extract any information they want to exploit the system. Directory Traversal attack is dangerous too, however, developers don’t tend to dynamically include files anyway, hence why it isn’t as important to protect from this attack compared to others.  
The above-mentioned attacks are bad because even a simple breach will cost the company a lot of money. Not only will they break laws by not keeping their users secure, but they will also lose a lot of customers as they might be deemed as “unreliable” due to the system breaches. As soon as that happens, all customer accounts at that moment are vulnerable and the hacker can do with those accounts as they please, for example, steal their payment information, if that was stored on the system. (WebsiteInsurance, 2018)