OWASP

Today web services gives way to communication, sharing and connection of people via the network. As according to Al-Fedaghi (2011), web applications are applications that can be accessed via Web browser over a network and was developed using browser-supported languages such as HTML and JavaScript. Through how web applications are made we cannot deny the vulnerability of applications security risk or threat.

Some of the common security risks are Injection, Broken Authentication and Broken Access Control.

**I. INJECTION**

Everyday people around the world rely on the internet, applications that can be useful to them. Also, the internet has vast internet platforms that users can interact with, give personal information like their name, birthdays, and sometimes confidential information. According to Time, 74% of Americans care about the person or organization that takes care of their information. What if, the websites they are using and have given their personal information to it is vulnerable and not well protected making it prone to have cybersecurity issues without the users knowing it. One of those issues are the injection attacks. An Injection attack is allowing an attacker to insert a malicious code inside an existing source code or commands and sometimes in queries, and while the interpreter reads it as a part of the program that can change on how the program works when executed. Sometimes injection attack attempts to download a certain file into the victim's device or they can have unauthorized access to the user’s data resulting to corruption, stealing or rejection to access their data. Injection attacks are the oldest and at the same time one of the most dangerous and threatening attack. When it comes to web security, injection attacks are one of their major problem. On the data that was given by OWASP (Open Web Application Project) top 10 critical web application security risk, injection is on top 1 or A1 from the year of 2010, 2013, and 2017. This means that injection attacks are prevalent. Since injection attacks are widespread the common attacks are happening on the legacy codes.

In order to to protect and secure the safety of the web application and its users the developers must know if their web application is vulnerable or not. The developer must examine the source code of their application and search for the codes that can access the external resources or can execute external commands. As much as possible the application must avoid using external interpreters, developers can instead use APIs that are safe and can execute the equivalent function as the system call and shell commands. Use appropriate and relevant privileges of users in performing some functions of the application for example when accessing the database you should not be login as the admin. All the data supplied by the user and put in the command should always be filtered, sanitized and validated by the application. There are no other ways of preventing the injection attack, the only solution is by fixing the source code of their web application making it secure and protected.

**Example of Injection**

One common example of Injection is the SQL injection. It is an attack that's happening in the database of the application. It can control the datas of the database, for example deleting, updating or even dropping of table that can result to data loss.

The query above is prone to sql injection. The attacker can add some malicious code that can fetch the data and view the entire table of the database, for example:



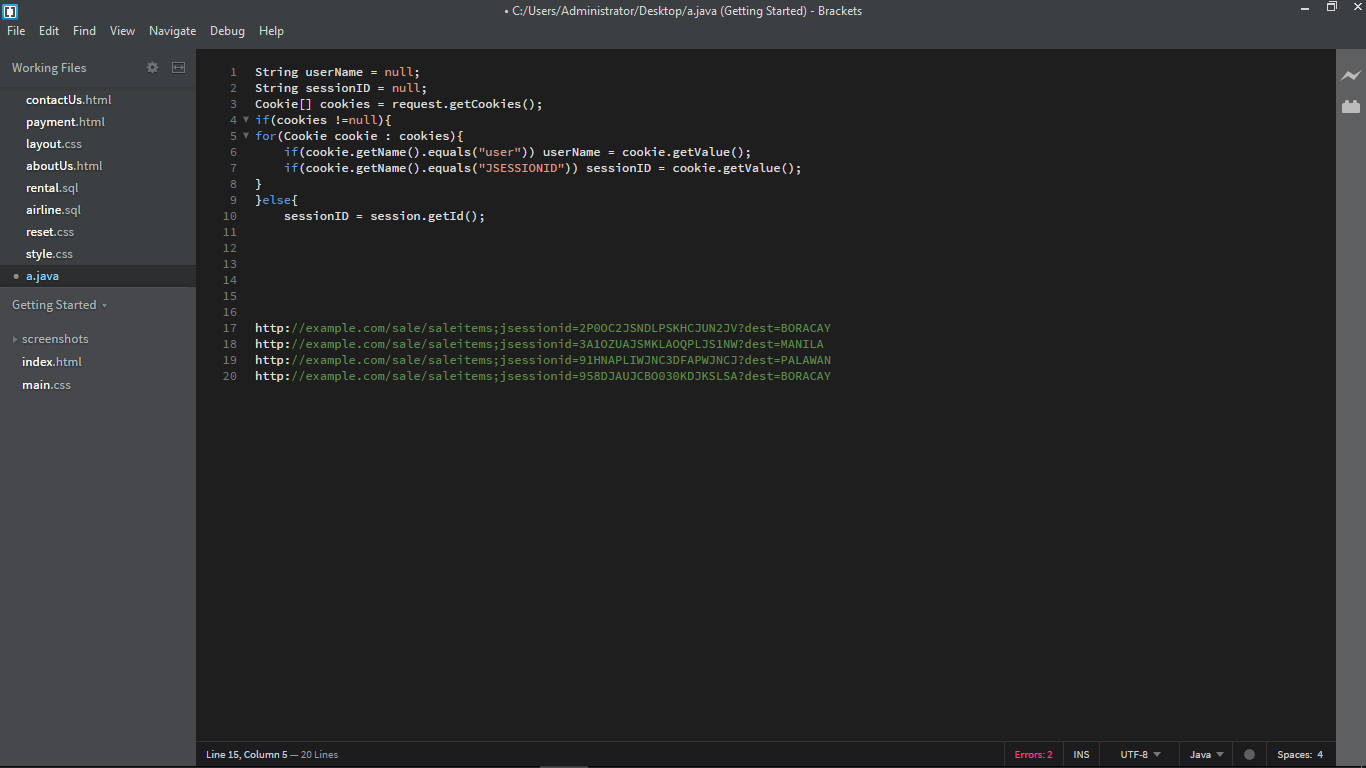
Since 1=1 is true and will always be true, the query above will return all rows of the account table. To avoid this scenario the database must used parameterized queries or stored procedures, whitelist input validation, minimize the privileges of users, and lastly escape all the inputs supplied by the users.

**II. BROKEN AUTHENTICATION**

Today web services gives way to communication, sharing and connection of people via the network. As according to Al-Fedaghi (2011), web applications are applications that can be accessed via Web browser over a network and was developed using browser-supported languages such as HTML and JavaScript. Through how web applications are made we cannot deny the vulnerability of applications security risk or threat. One of which is broken authentication which compromise a system by the ability of the attacker to gain access to few accounts or only the admin’s account. This is just one common scenario when declaring authentication failed. The data captured will be prospect to any of the following cases, money laundering, identity theft and social security fraud. Also, broken authentication together with session management is about handling user authentication and providing a limited session to all users.

The reason why this type of threat belongs to the top 10 most critical web applications security risk by OWASP is the universality of identity and access control design and implementation in which creation of username and password became so conventional. Flaws to secure authentication and manage account’s session leads to failure in protecting user and business credentials. Some of the preventives of this threat to happen are as follows. First*,* eliminate the use of default username and password. It is a good practice to have a strong password and avoid rotational password change. Aside from that, provide a weak password check to ensure that an existing password cannot be easily determined by attackers. Second*,* use a non-derivable or calculated account recovery. Third,whenever you browse confidential information in another device or computer make sure to end the session of authentication. Failure to do so will contribute to the vulnerability of your account to identity theft and social security fraud. These are just some of the prevention tips for users or clients. The developers are recommended to make use of a server – side random session ID generator for the session ID not to be visible to the public**.** It is also advised not to write a revealing URL in your website.

**Example of Broken Authentication**



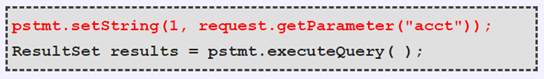
An example of broken authentication is a URL rewriting for an airline reservation that supports putting a session IDs in the URL. Common scenarios, a registered user of the site wishes to share about a sale to his/her friends. Where the user sends out an email containing the link, without realizing they are also sending away their session ID, and other valuable information. This way, an attacker could potentially use it for malicious purposes. Basically, if you notice the “***jsessionid***” value in the image above, it is either from a copy and pasted link or from a screenshot of a debugging tool for HTTP, you will be able to use their login details by just adding the cookie of “***jsessionid***” to the URL. These types of sessions expire naturally within half an hour of inactivity. This is known as the infamous session fixation. To avoid it quickly, (1) you must first disable the URL rewriting so that the jsessionid ceases to exist in the URL, (2) have high regard on to the logins, and your own login data. Let the login and authenticated traffic data go over an “***HTTPS***” instead of an HTTP. Lastly, you must not forget to make the settings of the cookie in your HTTPS to be in a secure mode.

**III. BROKEN ACCESS CONTROL**

Parallel to modernization of the society, comes the sudden increase of security risks that target different users from around the world. OWASP, an organization who publish a yearly list of top ten web security risks mentioned Broken Access Control. To understand further, let us first discuss Access Control or authentication. It is a policy that implies the role of a user and the limit to what a certain role can access. For example, a company website may have three roles, an admin, the boss, and the employees. An admin may read, write, and execute functions, the boss may read and write files, while the employees can only read files. They can only act on certain things because the access control prevents them from acting outside of their given roles. An Access Control policy should be designed carefully with proper research because a hole may lead to Broken Access Control. It is an attack prevalent in the web, wherein the attacker uses a weakness of a website’s access control in order to access unauthorized files and control privileges of a specific role. These attacks may eventually lead to data modification, data destruction and unauthorized information disclosure.

In order to prevent this kind of threat from happening, developers are recommended to properly research and design their access control in a manner such that an attacker won’t find any holes in the policy. Some developers creates their access control as they create the website, doing so may lead to holes and vulnerability to the authorization policy because a developer might miss a branch in the web as it grows and adds up functionalities and resources. Another way is to correctly identify the role of a specific user after authentication. This will prevent a user from acting outside of their given permissions. Next, check the authentication of user every execution of functionalities. This will prevent an attacker from directly accessing a functionality through the urls. Another way is to add access restriction especially for administrative roles. This can be done by adding IP address restriction where in only in a certain network can the user access the files. If a developer finds a hole in the policy, make sure to evaluate other functionalities of a web because one weakness may eventually lead to another weakness if not prevented at its early stage. Create a log to record events when an important web functionalities and files are accessed. This will let the developers investigate and trace out an attacker who tries to breach the website.

**Example of Broken Access Control**

In this example, the attacker can access any user’s account information by merely changing the parameter “acct” from the request.getParameter function. The account information accessed by the attacker may eventually may be used to access unauthorized contents that requires a specific roles and authentication.



In this second example, the attacker targets the url for the administrator access by forcefully typing it in the browser. If an unauthenticated user managed to access the url, it means that there is a hole in the access control policy that requires urgent attention by the developers. If this is not properly fixed, an attacker may use this type of broken access control to use a user's privileges.