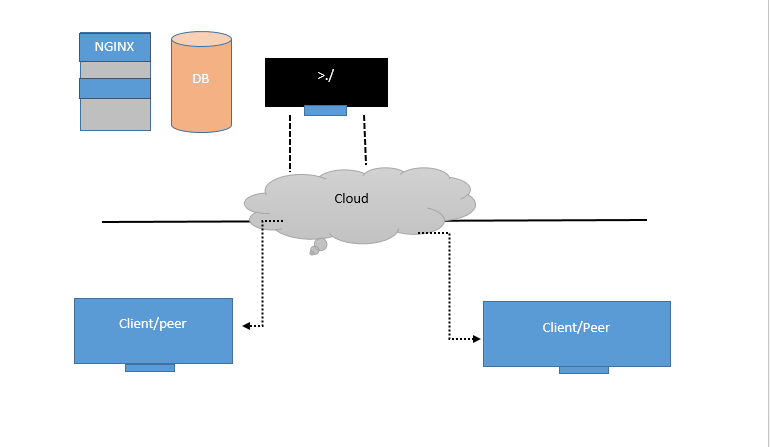
**PHP Security Flaws and Preventive Actions**

**NAME**

**COLLEGE NUMBER**

**Abstract**

The internet as we all know it runs on a number of very powerful tools. Some of these tools are well known to us whereas some are not. Different servers process thousands of information globally located in different parts of the globe. Below is a sample overview of how this architecture looks like:

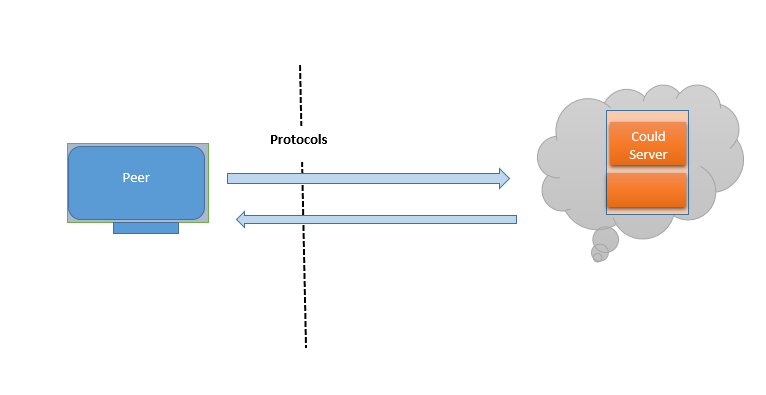
****

So what happens here is that the web as we know it is an interconnection of computers running applications that are hosted on several if not different applications? Every computer has got a memory discs or drive that is stored inside it, the memory disk stores information that is relevant and needed by any computer relevant connected to this computer Fielding et al (2002). This is what gives this particular computer the name sever. Its serves information based on what it has and is known to it, as shall be discussed in the next topic.

The database is the backbone of every application holding the data together. This is gold. For every web app to run, become dynamic and user centric, the data component of the application system has to be well deigned and implemented. When hackers are interested in gaining access to any software or system, the core interest is usually at the database layer, with major emphasis being laid on the location of such useful information. Valade (2004). Further, every internet interaction is methodized as some kind of web storage. At the back of the mind, of the kiddie user, the theory they have about the internet is that the computer, data, software is literally stored in the atmospheric cloud.

Well, research into these kinds of possibilities are currently ongoing, with the capabilities and abilities of storing user information and data in gaseous data, which then would mean that data is literally in the cloud. Currently, it’s possible to store data in molecular form. The next stage of data storage would combine a physics and chemistry interdisciplinary implementation of now converting the data from its current molecular data into gaseous form that may then not need, in the architecture above, the cloud is also symbolic of the remote nurture in which application servers are stored and located remotely. Between the servers and the client applications is a thin line connecting these infrastructures, these infrastructures are called protocols.

Protocols define how the information travels and is consumed by very machine Tanenbaum (2017). The host machine has a defined rule that is restricted and identified by what one might decide to call as protocols. These protocols shall define where, the request information is coming from, and who is sending the request and what should the target server return type be like. Look at the sample below:



From the above illustration, the protocols shall define how the information is shred and transmitted over the networks. Some of the protocols discussed and used in web communication include:

* DNS
* TCP
* IPV4
* IPV6
* SSL
* TLS
* HTPPS
* HTTPS

The discussion on the above protocols shall be handy in establishing the security related features of PHP as a framework. Focus however should be drawn to three components, the language (PHP), the database systems and then the programming paradigms used by various developers in coming up with most of the applications.

**Introduction:**

There are close to 700 programming languages in the world today. Every programming language is designed to solve a particular problem in the world today. Earlier programming languages were more of processed based and statically and loosely typed to solve the various problems at that time, for instance the Common Business Oriented Language, COBOL, develop in the early 1960s by Watsons IBM, focused more on developing better computers at that time.

Unlike recently when computer programs have been developed to solve very specific business and user needs. Lately, frameworks have been developed on top of the existing programming languages to help developers write code much easily, reduce time during code compilation and render applications very fast, without the need to write code from scratch.

**Problem statement**

Information security is a key concern to every organization and company. Malicious programmers and black hat are constantly developing tools, algorithms and tools that can bypass security levels used in most applications, the effect of which has seen most organizations lose large amounts of finances, reputation and patented confidential data and information. However, it should be known and understood that security flaws can happen in any given system, since so particular system is 100% effective and secure proof. Continuous research should still be put in place to ensure that the infrastructure running most of the applications are up to date.

Even though some of these innovations by developers have focused more on rendering business need and reducing the delivery times, some have also gone out of the way to develop secure PHP applications. The purpose of this research shall therefore examine the use of PHP at four major layers:

* At programming/development level
* At application use level
* At data transmission level
* At database level.

Also, it’s worth noting that some of these flaws associated with the language are either due to user negligence, knowledge gap or simply in-adequate due diligence in managing applications that have been developed using the language.

**Introducing Hypertext preprocessor (PHP)**

The language was initially introduced as a general web programming language. PHP suites well server side programming, i.e. writing and rewarding requests from and to the server. The language can as well be embedded inside the normal HTML documents as per below:

<!DOCTYPE html>

<html>

    <head>

        <title>PHP ARENA ⛺ </title>

    </head>

    <body>

        <?php

            echo "Php is great 🐘 ";

        ?>

    </body>

</html>

There are a number of frameworks that have been built on top of PHP as a programming language. These frameworks include Laravel, Yii, and Codeignitor. Other CMS applications have also been developed and are open source for developers to use and scaffold to product of choice.

**Php Security flaws:**

There are a number of security issues associated with the use of PHP and the development of applications running on the PHP frameworks.

**Cross site scripting**

This kind of an attack is also known as XSS. It enables the attacker to simulate and have another view as almost what the original user has. Usually, the attacker sends active that contains some malicious code that executes behind the back gourd when the target computer or hosts clicks it. The code will run behind the scenes, duping the user to have the same site as the original, now when the user enters sensitive information like their username and bank passwords, the same are fingerprinted by the attackers and can later be used by the attacker to conduct criminal activity in other sites and systems as well.

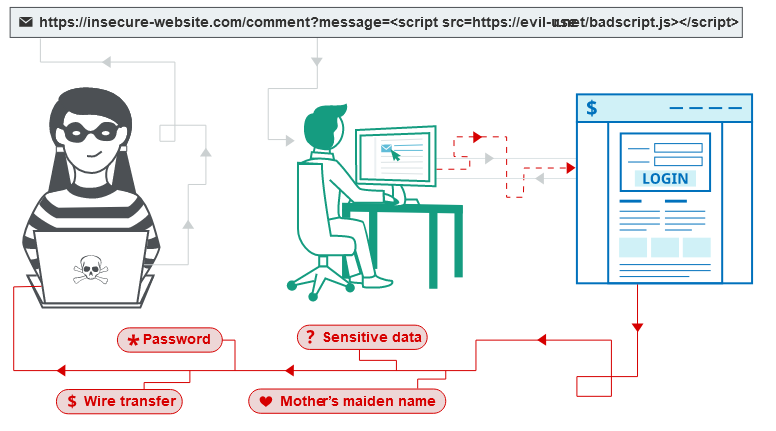
****

Figure 1 https://porstwigger.net/web-secuirty/cross-site-scripting

There are three types of XSS attacks and they include:

* Reflected XSS
* Stored XSS
* DOM based XSS

According to Stein (1998), usually the reflected XSS attacks happen when the user gets some kind of information on an insecure networks that doesn’t run the secure protocols as previously discussed in the above security protocols of data transmissions. Instead if securing the PHP application using the secure HTTPS protocol, its left open and the attacker can easily manipulate the request as below:

https://unsecuresite.com/status?message=<script>/\*+Malcicious+code+insert...+\*/</script>

<p>Status: <script>/\* Malicious code here... \*/</script></p>

This is usually the most common types of attack on PHP applications and if the user goes on and clicks the link, the link will executes the above code and the malicious program will again access to the same content that the user has, this can include passwords and other sensitive information.

Stored XSS happens usually when the applications receive and store some kind of information that seems safe but has unsecure content within it. These kinds of contents can be stored in the application databases, message boards, chat rooms and blog pots comments. All the attacker has to do is to store the XSS scripts and masks it as a safe comment or chat message. The target user computer may be tempted to open it and once they click this unsafe content malicious XSS code executes faithfully behind the scenes and begins to gather useful information about the user and their usernames and passwords.

Most PHP words press developers have fallen for this kind s of attacks amongst its users and lost lots of valuable information. Below is an example of an XSS script targeted to a PHP message board website:

<p>Hi, am glad to join this new chatroom!</p>

<p><script>/\* some malicous code come here... \*/</script></p>

With the DOM based XSS scripting, the attacker targets the applications that contains some client side JS with the DOM elements. If the JavaScript code contains some DOM element whose value can be manipulated then, the attacker easily switches that value with some malicious code scripts that is called and executed at JavaScript browser run time. The effects of which are the same as those of a stored XSS. Within no time, the user shall have gained access to sensitive personal data and information belonging to the user. Below shows a sample JS and HTML code illustrating this;

 <script>

        var my\_search = document.getElementById('my\_search ').value;

        var show\_results = document.getElementById('show\_results');

         show\_results.innerHTML = 'Here is your search: ' + my\_search;

      //bad code comes here:

     <h2> Here is your search: <img src=1 onerror='/\* Malicious code goes here... \*/'>  </h2>

    </script>

It’s also worth noting that the dangers of an XSS attack just ho beyond then normal use and password stealing, it also include complete web defacement and injection of Trojan virus into the application.

**Preventing an XSS attack on PHP application**

The most recommended way of doing this is by ensuring that:

1. The inputs coming in from PHP forms have a specific array of allowed characters only
2. Php developers should employ the use of certain type hints and type casts during development.
3. HTMLentities and ENT\_QUOTES should be used to escape certain outputs within HTML contexts

**PHP SQL injection attacks**

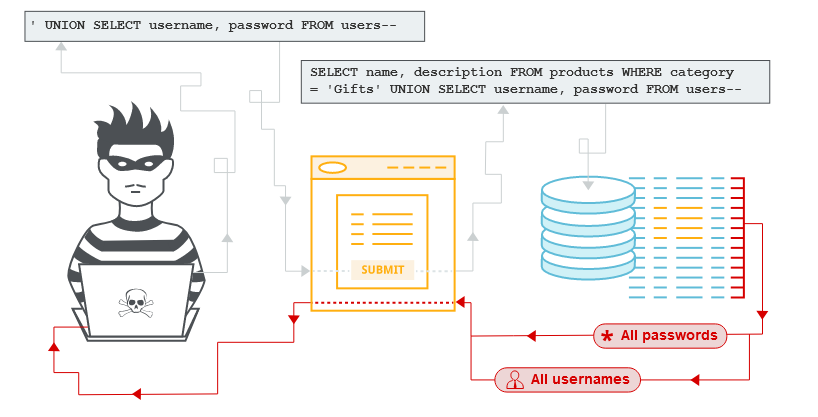
****

Figure 2 <https://portswigger.com/web-security/sql-injection>

SQL injection attacks happen on applications and target the database that host these applications. The attacker is able to re-write the logic that executes queries in php input forms and get access to privileged user list and credentials that a normal user would not have or that which is readily available to the application itself. The result of which can make the admin gain super control of the database. A logged in attacker can view the database, make changes to user privileges and worse still, drop or encrypt certain databases within the application. In some case scenarios, the attacker develops a backdoor within the PHP app that can go for days in end without being noticed easily.

There are several types of SQL injection attacks that focus on:

**Retrieving data:** An attacker modifies part of the queries that gets an item lit from the database. We can consider a situation where a normal user goes to shop and selects a particular product in the category of computers, the resultant query is expected to only return that particular product and the category to which it belongs, may be with price and description details. The sample query looks like the one below:

https://unsafe-site.com/products?category=Computer

SELECT \* FROM products WHERE category = 'Gifts' AND released = 1

//the attacker implements an SQL injection attack 💉

https://unsafe-site.com/products?category=Computer'--

SELECT \* FROM products WHERE category = 'Gifts'--' AND released = 1

https://unsafe-site.com/products?category=Computer'+OR+1=1--

SELECT \* FROM Computers WHERE category = 'computer' OR 1=1--' AND released = 1

From the above site, the attacker manipulates the original query with the extension of a double – which in SQL query means a comment, so anything that has been utilized after the – is not interpreted by the database, what does is that it now give the attacker an opportunity to get and view all products in the database including those that have not yet been applied on the search query. This can allow the attacker to get access to other tables in the database including those of admin and also those of other users, within the PHP app.

When subverting an allocation logic, the attacker will simply take control of the SQL query by passing extra - - to the database that lets the user login without the SQL requiring a password to do so.

SELECT \* FROM users WHERE username = 'admin'--' AND password = ''

Suppose one goes to a php website and enters the above commands for an unsecure php site, the databases will simply log the person in. Further SQL injections can also be used in other parts of the database to get other records from the database by union joining them to another table, the result of which lets the attacker to have several viewed from other tables.

**Preventing SQL injections in PHP**

Since the most common type of sql injection arises from query and string parameterization, it’s essential to ensure that queries written in PHP are hardcoded and none contains the variables related in any way to the application. The avoidance of direct input fields by a user is an effective way of ensuring that the query only gets executed on the?’ command and this cannot be manipulated.

**Cross Site Request Forgery in PHP**

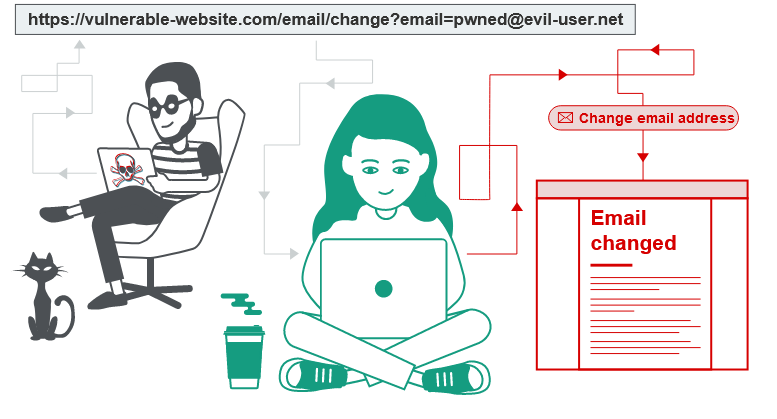


Figure 3 https://portswigger.com/web-security/csrf

Also known as CSRF have been designed to lure innocent application users to perform actions against their will. The attack happens by developing a malicious code in HTML that resembles the original web page of the user, usually a tool like Burp Suite is used to determine the contents of the web page in question and the type of response that will be made to the target host site when the request is made to the unsuspecting user like below:

<html>

  <body>

    <form action="https://unsafe-site.com/email/change" method="POST">

      <input type="hidden" name="email" value="pwned@evil-user.net" />

    </form>

    <script>

      document.forms[0].submit();

    </script>

  </body>

</html>

The actions performed by this site lures the user to reset their email account and if this is effectively done, the attacker keeps track of the cookie session storage, gets access to the username and password associated with use, the whole process cheats the bowser as if request is original and come from the legitimate source. Once the attack is successful, the attacker can deliver these links either by mail or chat groups, the result of which have worse implications if the victim is a privileged user.

**Preventing CSRF attacks in PHP**

The commonest way and valid one of stopping these attacks is by using CSRF tokens which have the following options:

* Is tied to the current user session
* Is hard to predict and has very high entropy
* Is strongly validated before any action is called

Laravel which happends to be a PHP framework has inbuilt out of the box CSRF implementation that automatically generates a user token for every validation session user as below:

use Illuminate\Http\Request;

Route::get('/token', function (Request $request) {

    $token = $request->session()->token();

    $token = csrf\_token();

    // ...

});

<form method="POST" action="/profile">

    @csrf

    <!-- Equivalent to... -->

    <input type="hidden" name="\_token" value="{{ csrf\_token() }}" />

</form>

**Techniques and evaluation**

Most of the security vulnerabilities associated with PHP flaws can be reviewed at based on a case by case evaluation. Whereas other factors can be a contributing factor to the effectiveness of these attacks, the following tables summarizes some of the reasons as to why these attacks can occur.

|  |  |
| --- | --- |
| Index | Threat cause |
| 1 | User negligence |
| 2 | Outdated software/ lack of maintenance |
| 3 | Poor programming paradigms |
| 4 | Inadequate information security training |
| 5 | Inadequate software and antivirus protection tools |

**Summary and conclusion**

Even though no software application program is considered 100% safe, a lot still needs to be put in place to prevent the flaws that generate from the use of these applications. PHP on its stance as a strong language hat has been used to power the web for several years and the focus now shifts to application developers and users, to develop more secure tested and validated web products that are up to fate with the latest industry standards and reputations.

The language in itself has been vulnerable to CSRF, SQL and cross site forgery attacks. On the other hand, users need to verify and have the adequate background information on handling these applications, the threats that might come with clicking phishing links and opening suspicious attachments. Additionally, installing antivirus and operating system detection software goes along way into protecting these flaws from happening.

**REFERENCES**

<https://www.php.net/manual/en/intro-whatis.php>

<https://portswigger.net/web-security/>

Fielding, R. T., & Taylor, R. N. (2002). Principled design of the modern web architecture. *ACM Transactions on Internet Technology (TOIT)*, *2*(2), 115-150.

Stein, L. D. (1998). Web security. *Addison-Wesley, Massachusetts*, *26*, 1-4.

Valade, J. (2004). *PHP and MySQL for Dummies*. John Wiley & Sons.

Tanenbaum, A. S. (1981). Network protocols. *ACM Computing Surveys (CSUR)*, *13*(4), 453-489.