User security in a PHP-driven website

How should you properly do it?

Jonathan Sundqvist

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden

ABSTRACT

User security (or just general website security) is an important part of any website. If not properly implemented it could have serious repercussions if any sensitive information were to leak.

In this report we investigate some common fault factors as well as common key areas for a website and how to implement security for them so as to increase the security of the website and the users

Three main areas are found and discussed as well as several sub-areas and an explanation for the main and sub areas is constructed.

We conclude that there is a surprising amount of information available for solving these various problems. We also conclude how to implement protections of several key areas and that, given how easy it was to access this information it's surprising how many websites don't have the protections in place. Moreover, we conclude that there is a need for further, more detailed and advanced research into this topic as well, to cover several other points not covered by this report.

Keywords: PHP; Security; Web

CONTENTS

1 INTRODUCTION

When creating an interactive website, specifically one driven by the users. Where you'd potentially make accounts or save various parts of information about oneself, like a simple forum. It's important to think in a secure way.

Far too many big and small services, big and small, are victims of hacking[1][2]. Where large parts of private and/or secret information is saved in plain text or in such a way that you can easily extract the information based on the remaining other parts of data. Because there are so many people creating similar kinds of websites, an important part to learn, is to learn how to create with the mindset of system security on the edge of your mind.

You have to know how to properly handle a user's password, Social Security Number (SSN), credit card numbers etc. You also have to know how to properly create and handle various processes. Such as resetting a user's password or updating a user's information in such a way that an unauthorized user can't do it for them.
If this information is missing in the developer, or if the information is old or incorrect or incomplete then the chances of there being a problem with hacking and user's information being out in the open are greatly increased.

The problem is of course very important for developers to start to handle. But it's equally important, if not more important for the end user's, since it's their information being leaked. The way chosen to examine this subject in this report is to describe how to handle some common processes as well as show some things one might easily miss or do if they're not properly taught.

[1] https://www.forbes.com/sites/robertberger/2017/09/10/equifax-hack-how-to-protect-your-credit-and-identity-if-your-data-was-compromised/#549ee1cc27ea Retrieved 2017-09-11
[2] https://www.identityforce.com/blog/2017-data-breaches Retrieved 2017-09-11

2 RESEARCH QUESTIONS

The main questions in this report consists of four main questions and one minor, but still important, fifth question.

4.1. How can you save a user's sensitive data such as the user's password in a secure way?
4.2. How can you properly handle the process of creating an account in a secure way?
4.3. How can you properly handle the process of restoring a user's password in a secure way?
4.4. How can you properly handle the process of saving or altering a user's information in such a way that an unauthorized user can't save or alter said information?
4.5. What are some potential factors that can add to a decrease in the security of the system?[3][4][5][6][7]

These questions, although aimed at one area per question, will show the developer how to perform basic actions for a website. These actions are in turn very versatile and the same way you handle a user's password could be used later to handle some other piece of information and so forth.
The purpose of these questions, and as a result, the report. Is to show how to properly perform certain common actions in a website. And by gaining an understanding of these basic problems and factors that can decrease quality the aim is to help improve the way websites are made and in turn decrease data hacks.

[3] http://php.net/manual/en/faq.passwords.php Retrieved 2017-09-11
[4] https://stackoverflow.com/questions/1624846/php-best-practices-for-user-authentication-and-password-security Retrieved 2017-09-11
[5] https://www.owasp.org/index.php/PHP_Security_Cheat_Sheet Retrieved 2017-09-11
[6] https://www.sitepoint.com/php-security-blunders/ Retrieved 2017-09-11
[7] https://stackoverflow.com/questions/6585649/php-forgot-password-function Retrieved 2017-09-11

3 METHOD

Since this report concerns the way to handle information and certain processes on a website, the method used won't be the normal method one might use where you'd most likely refer to book sources. Since this concerns a subject that's continually changing the method of gathering sources will be primarily through google to ensure that the information gathered will be most up to date.

The study will be performed by searching relevant terms connected to the question at hand. Examples of search terms used are "security in PHP" or "SSL free encryption". The search terms used will be some variation of various keywords pertaining to the current area of research, whether it be passwords, SSL traffic encryption or PHP password saving.
If a piece of information then occurs on three or more different entities, the information will be deemed valid. However, if the source is of code or a certain way to perform an action, some testing could be done as well. To test its authenticity.
To gather the information required for each question, several different sources, as well as the official documentation page for PHP[8] will be used for maximum coverage of the topic at hand. The sources chosen also just won't be randomly selected. They'll be under scrutiny as well, to determine if the source itself normally writes about similar topics and if what they've previously written is credible. The domain will be checked to find out their interests to ensure that they aren't tilting the article in a certain direction. Combine this with the previously mentioned authenticity test, the goal is to only have credible sources.

The analysis will be done on a question to question basis, with some referral to previous questions possibly occurring. Each question will be described in detail, with common pitfalls documented and a detailed solution. In the end some common analysis relevant to all of the questions could be mentioned as well.

[8] http://php.net/docs.php Retrieved 2017-10-03

4 LITERATURE REVIEW, ANALYSIS, AND DISCUSSION

4.1. Protecting a user's information

4.1.1 Short source review

On this topic there is of course a plethora of information to choose from. These sources vary in the form that some say how a developer should protect the user's information, and how a user should use its information to minimize damage in case of a leak. However, the sources, whether they be Google[9], regular people online[10] or the developers of a CMS (Content Management System)[11], all agree more or less how to achieve it.

Google writes that in order to secure a user's information all of the traffic is encrypted and the user's stored information such as the password is hashed[12] as well. The other sources agree as well. In this area there aren't that many different points, the main ones are to just encrypt the websites traffic as well as hash the user's passwords etc.

In short, there isn't really that much to the basics of this topic, and the majority of people agree on the same method of methods.

4.1.2 Analysis and discussion

Protecting a user's information is the base of every website. As mentioned before, you should start with encrypting the traffic to the website by using SSL[13].

Most webhosts offer SSL either as a free addition or with a charge[14][15]. But there is also the way to go if you want a free SSL certificate; certain services – such as Let's Encrypt – exist to give anyone access to an SSL certificate for free in order to better promote security.[16]

After you've secured your connections to your website, you then need to make sure that when the user creates its account the information is saved securely, in this case the password. To achieve that you hash the password. And although there are several hashing algorithms, such as MD5, SHA1 and SHA512.[17] You need to be careful of which one you use, since MD5 for example has the high chance of creating collisions.

You should use an algorithm that produces few if any collisions that also isn't superfast like most algorithms tend to be. A great example of this is bcrypt[18], which is a hashing algorithm designed for security in the sense that it should never produce collisions, it uses a unique

[9] https://privacy.google.com/your-security.html?categories_activeEl=sign-in Retrieved 2017-10-20
[10] http://www.creativebloq.com/web-design/website-security-tips-protect-your-site-7122853 Retrieved 2017-10-20
[11] http://opensolution.org/how-to-protect-your-website-and-your-customers-data-,en,202.html Retrieved 2017-10-20
[12] https://en.wikipedia.org/wiki/Hash_function Retrieved 2017-10-22
[13] https://en.wikipedia.org/wiki/Transport_Layer_Security Retrieved 2017-10-21
[14] https://www.one.com/en/hosting Retrieved 2017-10-21
[15] https://www.godaddy.com/web-security/ssl-certificate Retrieved 2017-10-21
[16] https://letsencrypt.org/ Retrieved 2017-10-21
[17] https://en.wikipedia.org/wiki/Secure_Hash_Algorithms Retrieved 2017-10-22
[18] https://en.wikipedia.org/wiki/Bcrypt Retrieved 2017-10-22

salt[19] - a unique key used to alter the output to produce unique output – and also has an option for scalability allowing you to alter how much computing should go into creating the salt and password hash.
When checking if the user entered the correct password you simply use the salt for the password associated with the alleged correct username provided and create the password again using the same salt. If the password is the same in the end then you know that the user entered the correct password.

4.2. Handling sensitive processes such as restoring passwords

4.2.1 Short source review

In this topic, depending on what level of developer you are, the sources you are looking at are of course going to be different, since you know more or less. Because of this problem there are of course websites that advertise both very insecure ways and others that advertise very secure ways.

The website codingcyber, a blog-like website. Shows one way of creating accounts and restoring passwords.[20] This author has allegedly several years of experience and shows an – although correct – way of restoring a password, the implementation being insecure but the principles of it being good. Simply enter your username and get your password sent to you.

Another source shows a secure and better way of handling it[21], although no code is shown it describes a very secure way of handling the process, and since it's on a public forum anyone can add or remove details to correct each other for a better response. That's the process we will focus on in this report.

In short, the literature on this topic varies, from the common person describing and insecure, simple yet correct way with the basic principle conserved. And the more knowledgeable source giving a more secure way.

4.2.2 Analysis and discussion

When handling sensitive processes such as creating an account, restoring a password or even just sending form data, there are several was to protect one self, many of the ways people don't think of.

For this we assume that you already have a functioning system for logging in and out, so as to be available to store information in user-unique sessions.

The first step is of course when sending form data, for example when creating an account or logging in, for securing the forms from being accessed by third parties you should have a functioning CSRF[22]-protection system. An example implementation of this can be seen here https://github.com/krysvac/csrf/tree/master. By using a protection system you ensure that the action sent to the server was indeed made by the user and not a malicious third party.

[19] https://en.wikipedia.org/wiki/Salt_(cryptography) Retrieved 2017-10-22
[20] http://codingcyber.org/send-forgotten-password-by-mail-using-php-and-mysql-35/ Retrieved 2017-10-21
[21] https://stackoverflow.com/questions/6585649/php-forgot-password-function Retrieved 2017-10-21
[22] https://en.wikipedia.org/wiki/Cross-site_request_forgery Retrieved 2017-10-22

You can also set the names of the various components of a form to random strings, to further secure the form so that no third party can refer to the field named "password" or "delete" etc.

After that we come to the part of resetting a user's password. Some people would just send the password in plain text to the user, but since the passwords should be illegible because of being hashed you can't do it that way. What you instead do is have the user input its username and then create a unique link and send that to the user, you set a time limit on the link for any amount of time for example 24 hours.

By doing this you ensure that the only user that can change the password is the user with access to the provided e-mail and making sure that you don't alter anything unless the user explicitly want's it. When the user then decides to click the link they simply input a new password and log in with the new credentials at the same time invalidating the link. A more detailed workflow can be seen here https://stackoverflow.com/a/6585668/2766226.

4.3. Potential factors for security flaws

4.3.1 Short source review

On this topic the literature is quite abundant.[23][24][25]
However, they mostly say the same information, to not trust the users input and never blindly print it or use it before escaping it and be very aware of how you send information and what it contains.[26][27]

When having any service that outputs user input it's always important to never trust the input to as to not produce any security flaws, and although worded a bit differently, the sources all talk about mostly the same topics. They also mention how important it is to always be aware of how you send data and how you show it, although again worded a bit differently between them. But they all more or less agree on the different argumentative points.

4.3.2 Analysis and discussion

One of the biggest things that can go wrong, that a lot of people might not even think about, is blindly trusting the user's input. If you create any type of website or program and don't properly handle the user's input you open yourself up to several issues right there.

In the case of a website you've immediately opened holes to SQL injections – a malicious action whereby a third party has SQL code, a very common language used for communicating to your database – where sensitive information could be leaked.
You also have the issue of printing unchecked input which could be used to show user's malicious code that could potentially harm them.

[23] https://www.sitepoint.com/php-security-blunders/ Retrieved 2017-10-21
[24] https://www.toptal.com/security/10-most-common-web-security-vulnerabilities Retrieved 2017-10-21
[25] https://www.phpclasses.org/blog/post/338-6-Common-PHP-Security-Issues-And-Their-Remedies.html Retrieved 2017-10-21
[26] https://stackoverflow.com/questions/129677/whats-the-best-method-for-sanitizing-user-input-with-php Retrieved 2017-10-21
[27] http://php.net/manual/en/function.htmlentities.php Retrieved 2017-10-21

In short, a large majority of security flaws can be stopped right away if you simply have the issue of checking the user’s input at the front of your mind.

5 RESULTS

Websites these days exist in the millions, and there are obviously going to be issues bubbling up to the top when so many people make websites. However, by simply following the few guidelines previously mentioned you can turn a website with flawed security into a secure way.

These practices, once properly implemented and learned by the developer can, aside from increasing the security of the website, then later be implemented in part on something else thus increasing the security of that website as well.

Protecting a user's information

In order to secure a website you first need to ensure that the website's traffic is secure. This can be achieved using SSL, either included as a free or paid service from your web host, or by using a service like https://letsencrypt.org/ which offers free SSL certificates.

You then need to hash the user's password by using something like bcrypt in PHP to secure it from being read in plain text.

Handling sensitive processes

Once you've secured the traffic using SSL and the user's password by hashing it, you should secure the sending of information from CSRF. A good way to achieve this is by using something like https://github.com/krysvac/csrf which is a simple CSRF class which provides a unique token for each request to prevent forgery. It also provides unique names for form components so as to make it harder to access by a third party. By using this on all forms and checking if the request is valid before making anything more you can stop a potential attacker right there.

Now that the user's information is secure and the forms used to send information secured to prevent a third party from sending requests posing as someone else you need to make a way to reset a user's password.

This can be achieved by having the user input its username and sending a unique link to the user, the link also being tied to the user. You'd set it to only work for 24 hours or some other time, and once the user clicks the link it is invalidated, so that if someone else were to click the link at the same time only one could change the password. After the user's updated its password the user just logs in using the new credentials.

Potential factors for security flaws

There are several potential factors for security flaws, such as lacking knowledge. But the biggest one is trusting the users' input. By trusting that the user inputs something correct and not validating it you immediately have a potential for a security flaw. Never trust the user.

6 CONCLUSION

It’s surprising that, given how easy it is to find solutions to these problems, they even exist. The questions asked, though specific, belong to key areas of a website. The security of the users stored and transmitted information, as well as preserving the integrity of the user’s actions by stopping third parties.

Protecting a user’s traffic and/or information is a very important thing and properly implementing security – at least if done from the beginning- isn’t very hard, it just requires some thought behind the code.
There are a multitude of people out there on the internet offering advice on how to increase your websites security, in the form of news reports, blogs, YouTube videos etc. You just have to be willing to learn.

To conclude, security is necessary and by following the steps shown in this report you can greatly increase the quality of the security. Save the passwords by hashing them to prevent anyone reading them in plain text. Encrypt the traffic, use CSRF to increase the security even more. And never trust the user’s information.

7 FUTURE WORK

In this report, there is an overview on how to protect a website on a basic level. A future work could be to investigate more in depth, i.e. implementing “smart” systems to prevent bots access.
One could also investigate better security solutions such as two-factor authorization or warning a user of a new login. In short, just a more advanced in depth report that really accurately describes what you’d need to think of if you’re a website such as google and how to implement those solutions.

Since large websites such as google probably has some very advanced systems in place for protecting its website and user’s.

REFERENCES

Rob Berger, “Equifax Hack -- How To Protect Your Credit And Identity If Your Data Was Compromised”. Forbes, 10/9 2017. [Online] Available: https://www.forbes.com/sites/robertberger/2017/09/10/equifax-hack-how-to-protect-your-credit-and-identity-if-your-data-was-compromised/#549ee1cc27ea [Retrieved 2017-09-11]
Heidi Daitch, “2017 Data Breaches – The worst So Far”. IdentityForce, September 2017. [Online] Available: https://www.identityforce.com/blog/2017-data-breaches [Retrieved 2017-10-03]
PHP, “PHP: Password Hashing”. PHP, Unknown postdate. [Online] Available: http://php.net/manual/en/faq.passwords.php [Retrieved 2017-09-11]
Various users, “PHP best practices for user authentication and password security”. Stackoverflow, October 2009. [Online] Available: https://stackoverflow.com/questions/1624846/php-best-practices-for-user-authentication-and-password-security [Retrieved 2017-09-11]
Unknown, “PHP Security Cheat Sheet”. Owasp (Open Web Application Security Project), Unknown. [Online] Available: https://www.owasp.org/index.php/PHP_Security_Cheat_Sheet [Retrieved 2017-09-11]
Pax Dickinson, “Top 7 PHP Security Blunders”. Sitepoint, 2005. [Online] Available: https://www.sitepoint.com/php-security-blunders/ [Retrieved 2017-09-11]
Various users, “PHP Forgot Password Function”. Stackoverflow, July 2011. [Online] Available: https://stackoverflow.com/questions/6585649/php-forgot-password-function [Retrieved 2017-09-11]
PHP, “PHP: Documentation”. PHP, Unknown. [Online] Available: http://php.net/docs.php [Retrieved 2017-10-03]
Google, “Information Security | How google keeps your data safe”. Google, Unknown postdate. [Online] Available: https://privacy.google.com/your-security.html?categories_activeEl=sign-in [Retrieved 2017-10-22]
Ruald Gerber, Toby Crompton, Tim Perry, “9 security tips to protect your website from hackers”. Creative Bloq, Jan 2017. [Online] Available: http://www.creativebloq.com/web-design/website-security-tips-protect-your-site-7122853 [Retrieved 2017-10-22]
OpenSolution, “How to protect your website and your customers’ data?”. Open Solution, October 2013. [Online] Available: http://opensolution.org/how-to-protect-your-website-and-your-customers-data-,en,202.html [Retrieved 2017-10-22]
Vivek Vengala, “Send Forgotten Password By Mail Using PHP And MySql”. Coding cyber, October 2017. [Online] Available: http://codingcyber.org/send-forgotten-password-by-mail-using-php-and-mysql-35/ [Retrieved 2017-10-22]
Various users, “What’s the best method for sanitizing user input with PHP?” Stackoverflow, September 2008. [Online] Available: https://stackoverflow.com/questions/129677/whats-the-best-method-for-sanitizing-user-input-with-php [Retrieved 2017-10-22]
PHP, “PHP: htmlentities – Manual”. PHP, Unknown. [Online] Available: http://php.net/manual/en/function.htmlentities.php [Retrieved 2017-10-22]
Pax Dickinson, “Top 7 PHP Security Blunders”. Sitepoint, December 2005. [Online] Available: https://www.sitepoint.com/php-security-blunders/ [Retrieved 2017-10-22]
Gergely Kalman, “10 Most Common Web Security Vulnerabilities”. Toptal, Unknown. [Online] Available: https://www.toptal.com/security/10-most-common-web-security-vulnerabilities [Retrieved 2017-10-22]
Atif Shahab Qureshi, “6 Common PHP Security Issues And Their Remedies”. PHPClasses, December 2015. [Online] Available: https://www.phpclasses.org/blog/post/338-6-Common-PHP-Security-Issues-And-Their-Remedies.html [Retrieved 2017-10-22]

Wikipedia, “Transport Layer Security”. Wikipedia, October 2017. [Online] Available: https://en.wikipedia.org/wiki/Transport_Layer_Security [Retrieved 2017-10-22]
One, “Tailor-made for your needs”. One, Unknown. [Online] Avaialble: https://www.one.com/en/hosting [Retrieved 2017-10-22]
Godaddy, “SSL Certificates”. Godaddy, Unknown. [Online] Available: https://www.godaddy.com/web-security/ssl-certificate [Retrieved 2017-10-22]
Let’s Encrypt, “Let’s Encrypt – Free SSL/TLS Certificates”. Let’s Encrypt, Unknown. [Online] Available: https://letsencrypt.org/ [Retrieved 2017-10-22]
Various users, “Fundamental difference between Hashing and Encryption algorithms”. Stackoverflow, Feb 2011. [Online] Available: https://stackoverflow.com/a/4948393/2766226 [Retrieved 2017-10-22]
Wikipedia, “Hash function”. Wikipedia, October 2017. [Online] Available: https://en.wikipedia.org/wiki/Hash_function [Retrieved 2017-10-22]
Wikipedia, “Cross-site request forgery”. Wikipedia, October 2017. [Online] Available: https://en.wikipedia.org/wiki/Cross-site_request_forgery [Retrieved 2017-10-22]
Wikipedia, “Secure Hash Algorithms”. Wikipedia, August 2017. [Online] Available: https://en.wikipedia.org/wiki/Secure_Hash_Algorithms [Retrieved 2017-10-22]
Wikipedia, “bcrypt”. Wikipedia, October 2017. [Online] Available: https://en.wikipedia.org/wiki/Bcrypt [Retrieved 2017-10-22]
Wikipedia, “Salt_(cryptography)”. Wikipedia, October 2017. [Online] Available: https://en.wikipedia.org/wiki/Salt_(cryptography) [Retrieved 2017-10-22]