# *PHPoor Security*: A Vulnerable Web Application for Beginner Programmers

Adam Decker
CEN4914 Senior Project
Department of CISE
University of Florida

Advisor: Dr. Joseph N. Wilson, *email:* jnw@cise.ufl.edu
Department of CISE
University of Florida, Gainesville, FL 32611

Date of Talk: December 9, 2014

**Abstract**:

It has become apparent to me throughout my four years at the University of Florida that web security is a current and rising issue for beginner programmers. This could be caused by any number of reasons: programming shortcuts are often taken, some programmers are inexperienced in coding secure programs, and sometimes it's just a simple mistake. My goal in this project was to help mitigate these causes by offering a hands on application made specifically for programmers

This web application describes the three most common vulnerabilities of 2013 based on the Open Web Application Security Project (OWASP) Top Ten report: Injections, Session Management, and Cross Site Scripting (XSS). The web application includes information regarding each of these vulnerabilities and how they may occur. The user is also able to access a series of challenges testing a basic knowledge of web programming in the PHP/HTML/JavaScript languages. Lastly, there is a section that describes programming practices to fix the exploited vulnerabilities. A basic knowledge of programming is required to get the most out of this application.

The following includes an outline of the current security problem, the steps I took to create this web application, the virtual server it runs on, the proper programming methodologies used to fix these common vulnerabilities, and references used throughout the project.

**Introduction:**

Although they may have not known it at the time, the early pioneers of computing technology truly changed the world. However, their inventions also led to some unanticipated problems. With the discovery of the Internet that connected people and their computers to others all over the world, a new issue arose, computer security: a serious issue during that time, and still a major issue now because of the inherent properties of a computer. A computer simply interprets the signals that are given to it. It cannot distinguish between good and bad or right and wrong.

**Problem Domain:**

Ever since the Internet's inception, sometime during the 1980's, there have been hackers trying to take advantage of the many vulnerabilities present in computers and Internet protocols. Since that time, things have gotten a lot more secure. Protocols have been revised to be encrypted, hardware has been improved, computer software has been developed to stop any malicious attacks, and computer Operating Systems are more developed, but there is always a risk simply because of the nature of the Internet and computing.

Every year, hackers steal billions and billions of Internet dollars [1]. During this day and age, there is a rising need for web security given the growing popularity of smartphones and the convenience of online transactions. The OWASP Top Ten Project explains in detail, the top ten Internet vulnerabilities of 2013. My application demonstrates the top three: Injection, Session Management, and Cross Site Scripting [2].

**Literature Search:**

The Open Web Application Security Project (OWASP) is an online community dedicated to spreading knowledge of web application security. Their website includes an abundance of projects, articles, and videos made for both users and programmers that are designed to help educate and defend against security attacks [3]. Some popular and successful projects include: OWASP Web Testing Environment Project, Zed Framework Project, OWASP CSRFGuard Project, OWASP Java Project, and many more. OWASP is at the forefront of the security world and has become a central hub for many security companies.

Enigma Group is a website that uses a hands-on approach to help programmers understand and prevent security flaws [4]. Enigma Group is self-sufficient and offers many security challenges that deal with Cryptography, Injection, Cross Site Scripting, Software Cracking, Programming, and much more. Once a challenge has been completed, the user gets a certain number of points, which gives extra motivation to continue to complete every challenge. The only flaw

that I can find with this site is that most of the challenges are don't explain the why the vulnerabilities exist specifically in code.

*The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws* is a book written by Dafydd Stuttard and Marcus Pinto that also uses a hands-on approach to demonstrate web security and where vulnerabilities may exist in code [5]. The book includes coding examples that demonstrate Injections, Session Management, Session Authentication, and more. There are also many other organizations and publications that have become a part of the fight against web application insecurity. I encourage everyone reading this to check out all of these resources to become more educated in Computer Security.

**Solution:**

The PHPoor Security application is the solution I designed to help in the fight for a more secure online world. It is designed for a very specific usage in comparison the aforementioned security organizations. It deals mostly with PHP (as the title suggests), it is made specifically for beginner programmers (coding details), and it is used with a virtual machine (security purposes/potentially demonstrative). As previously mentioned, it demonstrates the causes of the top three vulnerabilities of 2013 and provides an explanation as to how they occur and the best ways to prevent them in PHP.

First off, there are injections. PHP code, as well as many other scripting languages, is executed at runtime. This allows malicious users to potentially inject their own code along with the benign PHP/HTML code. The ability to inject their own code allows hackers to do things like log in without sufficient authentication, get data from the server database, and even hijack the server. The challenges for the "Injection" portion of the application test the users ability to do just that, and allow the user to see the effects first hand.

Tutorials are available to help the user understand each of the challenges vulnerabilities and how to fix them. For example, injections generally occur when the server does not check to see if user inputted data is benign or not. In PHP, you can use regular expressions to check this. If a user enters a "SELECT * FROM…" in a

“Username Login” form field to submit, the server should be able to recognize a SQL script and not allow that input.

There is also an inherent danger that comes when user input data is displayed like in a web forum. The user can once again enter in code and have it run on the page when it is displayed. There are a few solutions to this problem, you could do the same as above and not allow those types of inputs, but this greatly limits the users ability to post useful information. The best solution to this is to display everything the user inputs as a literal string and have it not execute on the web server. This can be done in PHP by using String filters and escape keys on the user input.

The next vulnerability is Session Management. Session management attacks occur when a hacker is able to hijack a user’s session token, and gain access to that users session. This allows the hacker to potentially gain private data about the user, perform actions as that user, and potentially gain special website privileges. Session management vulnerabilities generally fall into two categories: weakness in the generation of session tokens and weakness in the handling of session tokens throughout their lifecycle.

Due to the nature of Session Management, the “challenges” of the application are more like tutorials rather than hands-on like the others. They still offer coding examples and convey important, specific information regarding the generation of session tokens, the handling of these tokens, and also some potential alternatives to using sessions like HTTP authentication and session-less state mechanisms.

If an attacker is able to create several different accounts and log in using each of these accounts, he/she may be able to find out the way the tokens are being generated and how they are being used to validate certain actions on the website. They could then generate and submit their own tokens to try to gain administrative access. The solution is to use a better form of token generation, which makes guessing a token much more difficult. Using a truly random number generator algorithm along with different types of encoding/cryptography can produce a good session token.

An equally important factor in session management is the handling of generated tokens. Transmitting tokens over a network in an unencrypted form can still allow any session to be hijacked. Some applications use an encrypted transfer of data when logging in, but then change to an unencrypted protocol like HTTP. Some applications have very persistent user session, like Facebook, which allows anyone to access a user's session if that user forgets to log out of a public computer. Always encrypting session tokens and being aware of the way an application is being used can prevent most of these issues.

Lastly, PHPoor Security demonstrates a special form of injection called Cross Site Scripting (XSS). XSS differs from most form of injection in the sense that it deals more with attacking the users of an application rather than the server. It can be used in conjunction with other malicious behaviors to hijack sessions, gain personal data, and potentially execute arbitrary commands on a user's computer.

XSS can be used in many different ways. In my application, I supply some challenges to test a fundamental understanding of how XSS works and how it can be dangerous. For example, if user input is not verified, a user can enter in a script that can redirect all other future users to a webpage that is hosted by the malicious user and get session information without them even knowing. The solution to this is similar to the solution for many forms of injection: don't trust user data, always verify.

Creating an application that deals with Internet security inherently introduces a real danger of being hacked when using the application. Therefore, the application was created in a sandbox virtual machine that does not have an Internet connection. The set up of this server includes configuration of a VirtualBox Ubuntu Server and installation and configuration of Apache, PHP, and MySQL. Details regarding the installation and set up of this server can be found in the references portion of this report [6].

Screen shots of the PHPoor Security application can be found under Appendix A.

**Conclusions:**

PHPoor Security was developed in PHP, HTML, and CSS. It was designed to help educate programmers who were not comfortable with coding in PHP. The whole application can and should be run on a virtual machine after installing and configuring Apache, PHP, and MySQL. Each vulnerability being demonstrated has several web pages associated with it that allow the user to practice their coding skills and see immediate results.

PHPoor Security is a good starting point for beginner programmers looking to learn a little bit more about application security. It is by no means a comprehensive security application, but it demonstrates useful coding practices and allows for a user to practice their programming techniques in a virtual machine sandbox where the risks aren't nearly as great.

In the future I would like to expand my application to be more comprehensive and in depth. I would also like to expand to cover more of the vulnerabilities that were in the OWASP Top Ten list.

**Acknowledgements:**

First off, I'd like to thank my advisor Dr. Wilson. I always had such a great time in your class and I learned so much and gained a great respect for you and what you do. I was challenged every day to learn new material that I found very interesting and relevant. I haven't had many classes like yours here at UF. Secondly, I would like to give a special thanks to my super-advisor Dr. Schmalz. I had a great time developing this application, and as a side note also really enjoyed taking Introduction to Computer Organization with you. You have a unique teaching style that I always enjoy, and I hope you continue to teach both of these classes.

Lastly, I want to thank the University of Florida and all of the professors that work in the CEN/CISE departments. I've learned so much and I really enjoy what I do, and that comes from enthusiastic professors that love to teach!

**References:**

[1] – Anon. " State and Trends of the Russian Digital Crime Market – 2011", www.group-ib.com/images/media/Group-IB_Report_2011_ENG.pdf Group IB (as-of 2011)

[2] – Anon. “OWASP Top Ten”, www.owasp.org/index.php/Category:OWASP_Top_Ten_Project, OWASP (as-of 12 Jun 2013)

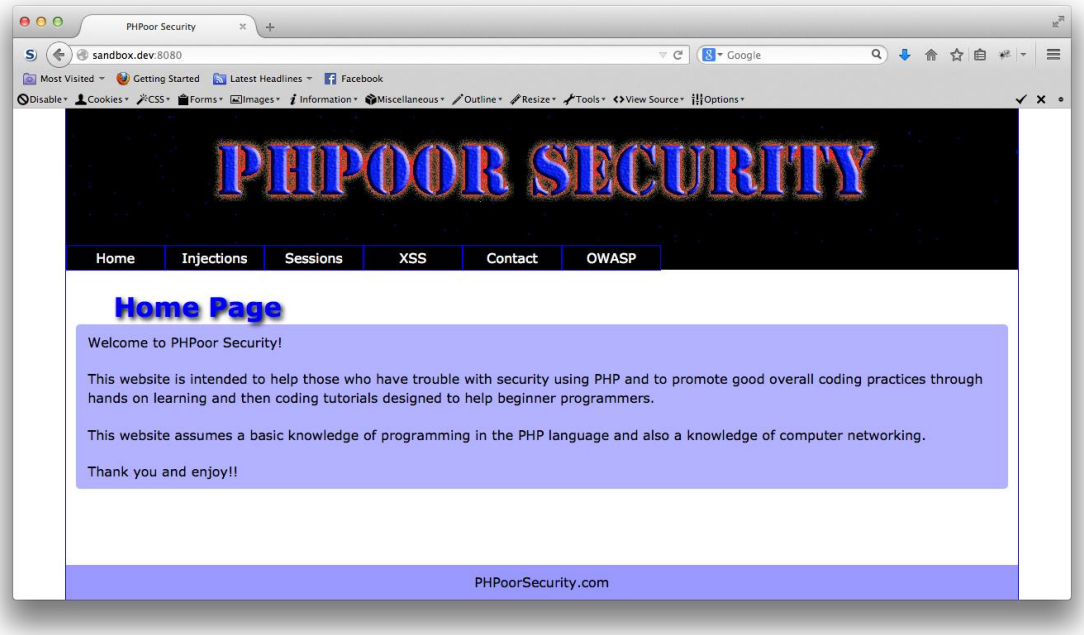
[3] – Anon. “OWASP”, www.owasp.org, OWASP (as-of 2014)

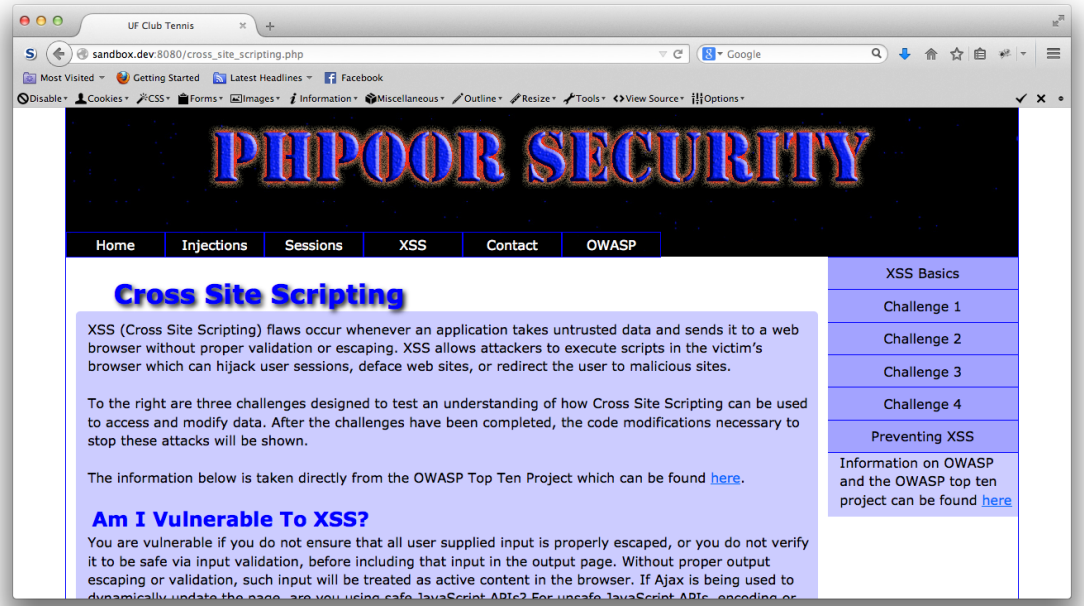
[4] – Anon. “Enigma Group”, http://www.enigmagroup.org/, Enigma Group (as-of 2014)

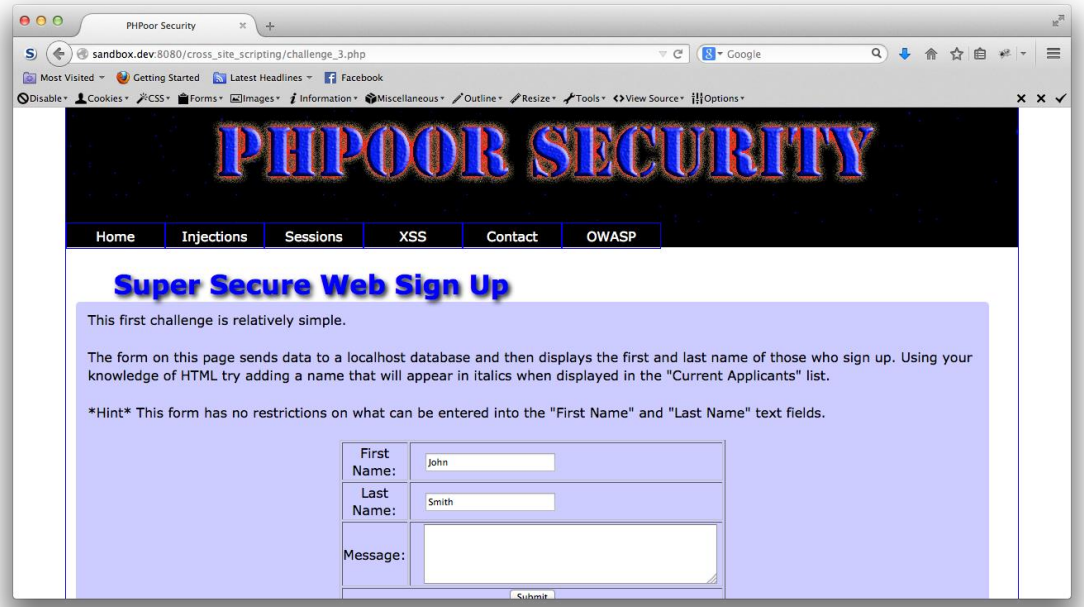
[5] – Dafydd Stuttard and Marcus Pinto. *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws* :Wiley (2011)

[6] – Peck, Jon. “Up and Running with Linux for PHP Developers”,www.lynda.com/sdk/Apache-tutorials/Up-Running-Linux-PHP-Developers/158372-2.html?, Lynda (as-of 13 Jun 2014)

**Appendix A:**






**Biography:**

At the time of this writing, I am a fifth-year senior at the University of Florida. I am about to earn my degree in Computer Engineering with a specialization in software and Cum Laude Honors. I was the UF Club Tennis President for the last two years, and had the task of creating a website for our team. I decided to create it

using PHP/SQL/HTML/CSS. During that time, I was also taking the classes Ethical Hacking and Reverse Malware Engineering with my advisor Dr. Wilson. This is what inspired my senior project.

After Ethical Hacking class I would constantly test what I had learned on my own website. It was this point in time that I realized I needed to learn more about PHP as well as web security. So, this motivated me to create a vulnerable web application so that I could test my own knowledge and help me fix the vulnerabilities I introduced, as well as provide some basic knowledge in secure PHP programming to those who may have a similar background as I do.