Guide to Developing a Secure Website

Bryan Wood
June 9, 2004

Problem Statement

The winter of 2004 brought with it a new opportunity for me. I began learning and programming in PHP and MySQL as an independent study and also for my job with both the admissions department and financial aid department of Eastern Washington University. My first bit of programming got off to a great start and it was when I was put in charge of developing a secure website that I began researching the topic of website security. I knew very little about how people hacked into websites to do various illegal activities, so I decided that some of the basic entry points into a website and its server needed to be addressed.

Secure website development is of the utmost importance as the number of websites that provide personalized accounts and utilize cross site information. “When you deploy your application on the Web, it becomes available to everyone” (Kabir 737). In this paper I plan to bring to light some dangerous practices that are being used to attack websites. I will present an example of a practice, present a practical and secure way to handle it and if I have encountered the example in my own code, I will discuss my experiences with it. To differentiate between a normal user and a hacker I will use hacker to mean a user whose intentions are to disrupt the normal running of a website and/or server.

“A supposed group of international hackers declares digital war against the United States directly from their headquarters in Russia” begins the article written by Paulo Rebêlo about a group of hackers called Hackers Against America. The goal of the group is the “stealing [of] classified documents and launching mass virus attacks against government servers”. This sounds like the plot of a great new sci-fi movie. Nevertheless, it’s the real thing.

A security expert helped bring down this hacker threat with his own set of security and hacking tools. The hackers also threatened to deploy viruses onto government servers and to do so they would have to gain access to the system as if they are right there at the machine. This is called root access.

Root access is the same as accessing a computer as if you were at the computer yourself. When a hacker gains this level of access they have every right that the server has been granted (i.e. create, modify, read or delete files and/or system settings). Usually when a user accesses a computer at this level they will place viruses or create a backdoor so they can return. Gaining root access to a machine opens the doors to accessing other parts of the system such as the database and the web site’s code.

Database access is usually accompanied by root access and allows a user to do as they wish. If a hacker’s goal is to access a database then the most likely target of that database will be the user information.

Code access allows the hacker to peruse the code and even allow them to make changes. This is often seen where a site has been defaced. This is commonly accompanied by the hacker placing a backdoor so they can return. Code access is a simple way to leave a calling card to show off his accomplishments.

User access levels are how computers determine what a user can or cannot do. This includes creating, modifying, viewing or deleting files. These settings are also designed to allow applications to function as a user would. A web server requires the ability to read files, edit database files, and run various programs to help service the client user. A web server should be set to read only the website files, denying the privilege to view other system files, run specific services such as the PHP preprocessor and to access the database to run various queries on it.

Many companies are looking for the cheaper faster solution for developing their websites. To develop a website using Microsoft products you have to buy a set of expensive licenses and software. The alternatives to Microsoft products are those of other big name companies such as Sun, Oracle and IBM. These products can be expensive beyond many development budgets; the free and inexpensive solution that remains is that of open source products. These products mainly require a commercial license to be used.

One of the leading dynamic web languages that compete on the market is called PHP. This is an open source language that can be used freely under an Apache-Style license.

> “PHP, which stands for "PHP: Hypertext Preprocessor" is a widely-used Open Source general-purpose scripting language that is especially suited for Web development and can be embedded into HTML. Its syntax draws upon C, Java, and Perl, and is easy to learn. The main goal of the language is to allow web developers to write dynamically generated web pages quickly, but you can do much more with PHP” (PHP Documentation Group).

As it stands now, PHP is one of the most widely used web languages. Developers use PHP to cut the cost of development since PHP is a free open source product and it can run on most popular operating system. PHP also contains an entire set of features that are specialized to work with databases.

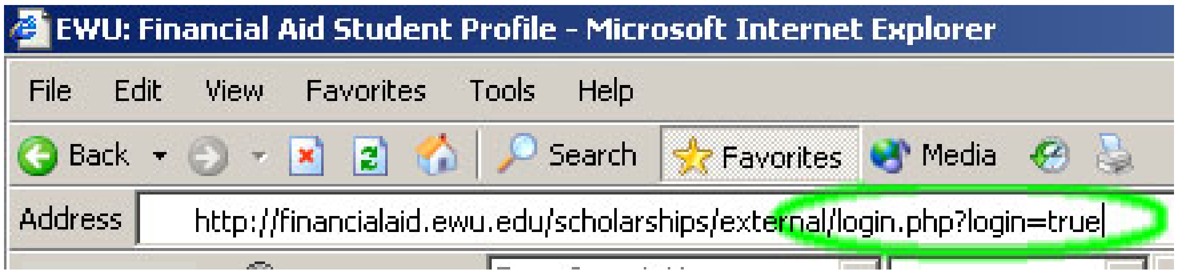
“MySQL is the world's most popular Open Source SQL (Structured Query Language) database management system” (Bloch). MySQL is released using a dual-licensing method. This database is free under the General Public License (LGPL) and companies can purchase a commercial license that cost hundreds less then that of the competition. MySQL is quickly becoming a mainstream database as businesses look towards the open source market for the less expensive products to support their business solutions.

Throughout my research I found four issues that continued to be a point of focus. These are query string manipulation, SQL injection, poor variable management and an unclear separation of the client and server functionality. I have also provided some code samples in the appendix for reference to these possible solutions.

Query String Manipulation

<u>What it is:</u>

The query string is a set of values that are passed through the address bar of a browser. They usually appear on sites that provide a service of some sort. Query String Manipulation can be very dangerous to a poorly programmed website. It provides an access point for a hacker to the database via a method known as SQL Injection, root access to the machine and access to various parts of the websites code.


EWU: Financial Aid Student Profile - Microsoft Internet Explorer
File
Edit
View
Favorites
Tools
Help
Back
Search
Favorites
Media
Address

http://financialaid.ewu.edu/scholarships/external/login.php?login=true

The image above depicts an attempt to manipulate a webpage by providing a variable called login to the page. This attempt will login a user if the page is does not defend against this attack by some means.

<u>What this means:</u>

Root access means that a hacker has access to the entire machine and is limited only by the access level provided to the web server, which is commonly an administrative level. This allows the hacker to modify or delete files currently on the machine. This hacker can also gain access to the database and all the data in the database even if the web server and database server are not on the same machine.

With a hacker that has access to the website's code, site defacement is usually the main purpose. However, interacting with a websites code can reveal the design of the database. Using this information one could obtain access to the database, and they could even change and view the database's contents.

<u>**Why this happens:**</u>

This kind of attack on a website is due to an improper use and handling of the query string. The most common mistake is, trusting the input from the query string to be valid. This allows a hacker to tamper with the query string to get the information they wants. The query string should be validated and verified every time you create and access the data. Query string data should be tested for its existence and validity.

Root level access with the administrator privileges can be gained because the server wasn't set up properly. Normally a server should be set to run with the least amount of privileges required by the server to perform its duties. One should determine the appropriate level of access a server will have before a website is developed.

<u>**Example:**</u>

You create a link with a query string variable to be passed to the next page. This variable is called ID. If you are using this variable to access information from a database then you need to test for it first.

Using the isset() function is the simplest method. This allows you to move the user back to the previous page where the value becomes present. Then if the variable does exist you should test to see if its value has been tampered with. We know that ID is supposed to be a numeric field only so we should run an examination on the value with a regular expression.

PHP provides an excellent regular expression evaluator function called: ereg(). "Regular expressions are the key to powerful, flexible, and efficient text processing" (Friedl 1). Using the regular expression, ‘^[0-9]+$’, to test the value will insure that there is at least a single digit 0-9 in the variable. The anatomy of regular expressions will not be covered in this paper, but can be found covered in the book “Regular Expressions” written by Jeffery E. F. Friedl and published by O’Reilly. The expression evaluates the value of a string as beginning with a number, containing at least one number and ending in a number.

After verifying that the value of ID, the use of this variable is more secure, and you will have weeded out most of the invalid values of ID. To provide more security, if this value is being used to access a file or a database field, test those situations for its existence before using the value.

SQL Injection

<u>What it is:</u>

“With SQL injection, it is possible […] to send crafted user name and/or password field that will change the SQL query and thus grant us something else” (SecuriTeam.com). This is an off-shoot of query string manipulation because of its nature. A hacker can actually attack a database by either changing values in the query string that has been determined to be part of an SQL string or by placing the malicious SQL into form fields.

<u>What this means:</u>

This allows a hacker to gain access to various parts of the server and even possibly root access. The most common goal is to acquire data that is contained in the database. Usually this data contains monetary or personal information such as emails or entire profiles for identity theft. A hacker could even set up a backdoor to the system and monitor and change the data if they chose to.

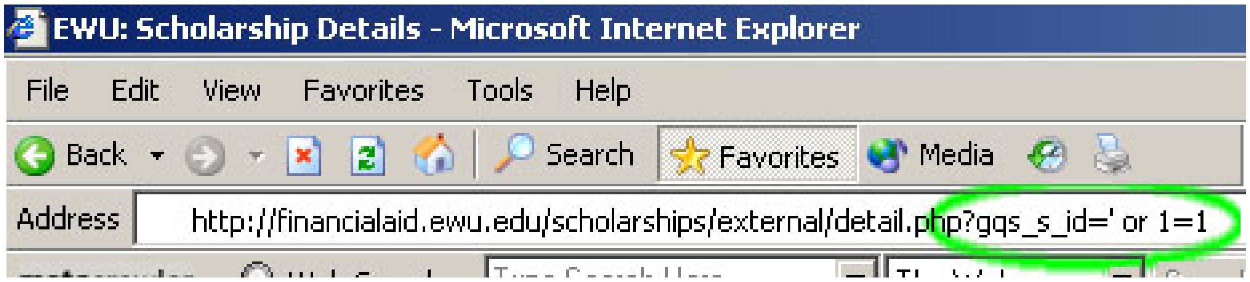
<u>Why this happens:</u>

SQL injections happen when a programmer does not screen any input that is used to interact with a database system. If unchecked data can be inserted into a database, then that information can be made to exploit the entire system. JavaScript tags can be placed to redirect a user to an insecure mock-up of a site to acquire that user’s data.

<u>Example:</u>

“… You should ensure that direct user input is not used in SQL Statements” (Kabir 747). To prevent this, a programmer must always verify the validity of the data being received by their application. PHP provides the perfect tools to test your data that will be stored in or interact with your database. One recommended tool is the regular expression function. This allows you to define the content of any string variable, since all data coming from a form is generally posted as string data and then manipulated there-after. “The best way to filter your data is with a default-deny regular expression” (Spett 22). The use if ereg() provides access to the evaluation of regular expressions in PHP.

There are cases where you would have to allow for special characters, called meta-characters, which would cause the database to react to them. This is because a database uses specific characters called operators which are used to signify some operation that the system is to take. One such character is the apostrophe. This character signifies the beginning and ending of a string. If a hacker were to include a statement such as “‘ or 1=1” and this was to be inserted at the end of a SQL statement, would cause the SQL statement to return data that it was not meant to return.



http://financialaid.ewu.edu/scholarships/external/detail.php?gqs_s_id=' or 1=1

The image above depicts an SQL injection through the use of manipulating the query string.

The solution when using a MySQL database would be to escape the string using the mysql_escape_string() function. This escapes all meta-characters that would usually be identified as operators in the database; however, using this function makes the string more or less benign going into the database. When this function is used to insert something into a database then the value of it is stored as string data in the database will need to be repaired. What this means to a programmer is that you will need to un-escape the string using stripslashes(). This returns the string to the original form and can then be utilized for its intended purpose.

Data that is allowed to be stored in a database in this fashion can also be used to trick the user. This is because HTML tags were not cleared or screened for. If a table is to store some HTML value and you are allowing a user to store just anything in the table, a hacker can exploit that storage by inserting script values that can cause adverse behaviors. There are many ways you can secure your site from hidden embedded HTML. The best way would be to utilize the ereg() function along with the mysql_escape_string()/stripslashes() functions and screen for the nasty tags of <script> or only allow pure html tags.

Variable Management

What it is:

"Some people have this utter obsession with temporary variables" (Hughes). The most important aspects about a variable are its memory requirements and processing time. The more variables you have, the more memory and processing time your script is going to need. This can become a burden to program and also allow an adept hacker to see what your scripts are doing and exploit it. A common way to exploit a page that uses too many global variables would be using the query string for several variables at a time. If you develop a page that allows someone to change the values in a variable then the security has been compromised.

What this means:

A hacker can manipulate and edit your page contents and even the functioning of your site. This also opens the door to being able to place HTML in the query string that will be placed into your page and executed. Most of the time a hacker who places HTML into the query string rarely affects another user. In certain aspects, placing malicious HTML/JavaScript into the query string will allow the string to be stored in the database. At a later date, an unsuspecting user will encounter this data by accessing it as part of the website. This is due to the fact that this information is inserted into a webpage that is viewable by anyone, such as a web forum page. The unsuspecting user will encounter this code and interpret it as a new portion of the website and will be lead away from the original site to provide the hacker with their private information.

Why this happens:

"Site defacements are sometimes left as calling cards by a hacker who entered a system by more complicated means" (Converse 820). This is allowed to happen because a developer utilizes the wrong techniques to develop a web page. When developing PHP you should make use of functions that return values instead of storing a value into a variable to be used somewhere else, and encapsulate your use of the global variables so that they are protected from tampering. If a page is requiring a lot of variables perhaps consider using an array to group the similar variables. Arrays can be assessed via an associated name instead of the indexing number. This means that a user can call an array's element by a name such as 'birthday' instead of using a number such as '4' and having to remember that '4' means the value of 'birthday' from the form.

Using an associative array helps group your data and also allows you to access it by the name you would have given the variable itself. This helps prevent getting values from the global scope variables mixed up with your page scope variables. Grouping a set of variables into a single array isn't saving that much more space. However it will provide you with cleaner and easier to read code. This can provide safer and more readable code. Using an array means that you will have to process the values and can be more ideally check for corrupt data.

Example:

In my experience the pages that require the most data variables are form pages. With the intent of allowing a page to validate a variables value that is posted by a form, it is important to remember that value in case not all the values were valid. If a form comes back to the user and says that they entered a value that was not valid in one or more fields, the user would not be happy if they had to reenter all 15 fields. Correcting only the field that was wrong is much easier for the user.

To create this kind of feature the values have to be re-accessed by the page. One could simply reuse the global post values and place them into the fields. To access the POST global is more secure then the GET global, unless someone intercepted the POST values. This means that the POST can potentially have dangerous data in it now. Do not post this information right away.

Instead, while the fields are being evaluated by some method of validation, place the valid items into a new array that has the same fields from your form. This way if an invalid field is considered invalid because of malicious code, the code is not executed on that user's machine.

Also be aware of sensitive data and always run this kind of information over a secure SSH connection with the page, since all form post data is sent as plain text. You should be aware of how you are using a variable, and if you are storing a value to be printed out later that is returned by a function, you should just call the function. An example would be to use the date function if you are going to print out the date to a page once or twice. If you are going to be printing a function's value more then two or three times then consider using a variable instead.

Error Checking

<u>What it is:</u>

All applications encounter errors when they are being developed and also after they are deployed. Many application errors occur when data not of the proper form and cause the application to stop functioning as it was designed. Developers will create websites that require information to be validated, and will only supply events for the valid data that is received. This can become a problem when invalid data is provided to the form by a hacker. Since invalid data is being passed in and is still accepted into a website and its database, this can cause the webpage to act outside of its original design.

<u>What this means:</u>

This allows a hacker to cause the application to react sporadically and in vice of its original design. Applications that are corrupted in this fashion allow this user to create an interface to the machine or machines hosting the application. Hackers could in theory gain root access to the machine and use their access to procure information about other users of the website, modify the data by any means and also provide a reentry point if the administrator regains control over the system.

<u>Why this happens:</u>

This happens due to the poor programming practices that beginners develop while reading mediocre tutorials and utilizing introductory materials [cannot say this]. These developers will not understand, or just plain don't, check for data based errors.

Through my experiences in the Computer Science Department at Eastern Washington University, I have been taught to explore the more unobvious use instances of a program and anticipate program errors. Developers "… that do not take the time to plan their script and identify all the possible places where something can go wrong." (Hughes) can cause fatal errors in their website and leave holes for hackers to attack.

Example:

A developer designs and programs a webpage that will take user input and place it into a database field. The developer creates the table and the columns that will be used to store the data from the page. Then the HTML portion of the page is developed to insure that all data is collected. In the end, the developer will code in a system that will handle data from the form. This will check to see if the submit button was pressed and then check if all the fields contain data since a developer wants to be sure that there are no empty fields. However, if a developer doesn't use any functions that verify the integrity of the data that the user will be providing, this will leave an entry point for a hacker to attack the system.

This is a classic design flaw seen throughout the web, which Sterling Hughes reports as one of his top 21 textbook mistakes. The developer should be analyzing the data from each and every field, insuring that the data is what it is supposed to be and that there are no malicious sections to it. This goes back to using the ereg() function, escaping the data going into MySQL and also filtering out any other parts that are not wanted in the database.

A developer should take an active position to understand what is being presented to the user and what the user is being allowed to do. Analysis of the code can reveal that certain coding practices where omitted and opened the flood gates for many unscrupulous actions to be allowed. Without properly testing the data that is coming in and preventing errors from occurring, a site is very vulnerable to attacks that exploit the errors printed to the client.

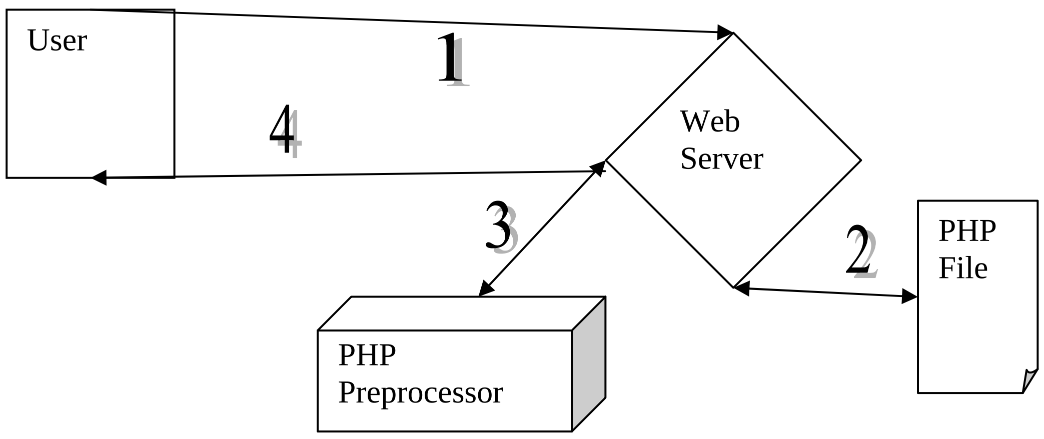
Separation of client and server

<u>What it is:</u>

"Sensitive data is often made available unintentionally by programs to people who should not have any access to the information" (Kabir 25). Web applications run in two locations: the server and the client. This means that both locations need to be developed to provide the best security for the user. The server needs to be developed in such a way as the information being stored is not compromised; while the client needs to be developed to present and retrieve only the required information. A client that divulges too much information is not likely to be secure.

<u>What this means:</u>

The server is where all the application's actions are taken place. The separation is mostly transparent to the user since they are seeing everything happening as the server and client interact. This is because PHP operates on a transitive level for the web page between the client and the server. The user will request a page(1) and the server responds by finding that file(2) in its directories and then processes(3) it and finally delivers it back to the user(4). The application event of PHP processing was not seen by the user since the browser announced that the page was being accessed and received.



Developing for the client means that the data and actions the user can take must be limited. For instance you would not present the user with a form that updates a news forum without the proper authorization.

A web page should not allow the user to interact directly with a database. If a user is providing information the client will only collect and submit that data to the server. If a developer wishes to screen content with JavaScript, then do so being aware that a client does not have to have JavaScript enabled or the page submitting the data may have been altered not to validate the data.

Why this happens:

When developing a web site a developer will have to manage two kinds of data. These data types are sensitive data which is usually user information and business data, and insensitive which is mainly application level data used to display images and included files. Nevertheless, all data is very delicate in its nature. I am making the division of data to illustrate that some data needs to receive special handling conditions.

The client of a web application can always be decoded and viewed by anyone and this allows the client portion to be the beginning point of accessing the sensitive data. This exposes insensitive data for the most part. The cannibalized version of the client can be used to attack the site or fool other unsuspecting users into divulging vital information.

A website is delivered to a browser in HTML. This HTML is then interpreted by the browser and can be viewed as plain text. Since the HTML is available most browsers allow its user to view the source of a web page. Please take note that the code delivered to a browser is the final pure HTML/JavaScript and not PHP or any other language even though the address may contain something other then htm or html.

A hacker will analyze the source of a web page and deduce some of the functionalities that the page contains. A form and its included fields all contains values that are posted when the form is submitted. This content can be manipulated to post as something else which can cause the server to think it is receiving something other then what the page was suppose to submit. Overloading the form like this may cause sensitive data to be displayed back to the hacker.

Example:

The server portion of the application should be designed to handle input that is meant to be malicious. Upon receiving a client's request and posting variables, the server needs to validate and verify this data and request. A simple example to prevent malicious clients from interacting with a server side portion would to use a global variable provided by PHP. This variable would store the server details such as the client's internet protocol (IP) address, the location of the requesting page and many more. If security is a top issue then validating these values should be a priority in a developer's code. Securing the connection between the client and server is a must.

Conclusion

The World Wide Web is a place of great software evolution which opens new doors to how information can be presented to a user and how we do business. However, as these new opportunities become available, so do new ways to get at the sensitive data that is being stored in a database website. These ways of acquiring sensitive data are changing daily and while new methods are implemented to protect this information; new ways to get around this protection are also being developed.

Throughout the development of a website a developer needs to be aware of what the user is being allowed to do. The topics covered above are just a few of the situations that a developer needs to take into consideration when developing a database website. These topics are perhaps the most elementary of a website's security and are complimented by server security settings and system security levels. Developing securely written PHP and MySQL code will help provide adequate defense. Remember "when you deploy your application on the Web, it becomes available to everyone" (Kabir 737).

Terminology

Client: A program designed to request information from a source and display it to a user. A web browser is a client.

Database: A collection of related information stored in a structured manor. Relational databases are becoming trendier as they allow the relation of information to be developed so that quicker access to appropriate data is achieved.

Open Source: Programming code that is developed openly free from any restraining license agreements or costs. Open source code is usually published under a general public use license.

Query String: The information that a web browser provides to the server by appending the location of a webpage. For example: “http://www.itanexmedia.com/index.php?date=20040501” the address is appended with a date of 20040501 (May 1, 2004).

Server: A program designed to process file and data requests from its client. A server application usually runs on a designated machine also called a server.

SSH: “Secure Shell is a program to log into another computer over a network, to execute commands in a remote machine, and to move files from one machine to another. It provides strong authentication and secure communications over insecure channels” (Jupitermedia Corporation).

Code Sample Solutions

This code is just a sample and should be placed in the appropriate location in the file. All the following solutions are in PHP and SQL statements compatible with MySQL.

Query String Manipulation

Situation: A developer needs to pass one variable, called ‘error’, to the next page via a link. Its value will hold a number that will determine what error to present. The receiving page needs to evaluate the variable then print out the appropriate error message.

Conditions: The variable can only be an integer and a string should display a special error. If there is no error then do nothing and ignore all other values.

```
if(isset($_GET[‘error’])){
      if(ereg(“^[0-9]+$”, $_GET[‘error’])){
            //display the appropriate error message for the
            //value of $_GET[‘error’]
      }else{
            //display a message about the illegal string
            //value for $_GET[‘error’]
      }
}
```

SQL Injection

Situation: A form will post two field values, ‘username’ and ‘password’, to a destination page. These field need to be validated by a database SQL statement.

Conditions: Both fields can only contain alphanumeric characters. The receiving page will validate the field values and then check the values in the database.

```
//check to see if the fields have values first
if(isset($_POST[‘username’] and (isset($_POST[‘password’])){

      //check if the username contains alpha numeric characters
      if(ereg(“^[a-zA-Z0-9]+$”, $_POST[‘username’])){

            //check if the password contains alpha numeric characters
            if(ereg(“^[a-zA-Z0-9]+$”, $_POST[‘password’])){

                  //run a select statement on the database with the
                  //username and password that is passed in

                  $result = mysql_query(“SELECT uid FROM clients WHERE
                        user=’”.$_POST[‘username’].”’ AND
                        pass=’”.$_POST[‘username’].”’”);

                  if(mysql_num_rows($result) == 1){
                        //log user in
                  }else{
                        //display a message that the login failed
                  }
            }else{
                  //display a message about illegal character values
                  //in $_POST[‘username’]
            }
      }else{
            //display a message about illegal character values
            //in $_POST[‘username’]
      }
}
```

Research Sources

Materials of Reference:

Converse, Tim, and Park, Joyce. PHP Bible, 2nd Edition. Indianapolis, IN: Wiley Publishing, Inc., 2002.

Howard, Michal, and LeBlanc, David. Writing Secure Code. Redmond, WA: Microsoft Press, 2003.

McGrath, Mike. PHP in easy steps. Southam Warwickshire CV47 0FB, UK: Computer Step. 2003.

Materials of Citations:

Bloch, Michael. "PHP/MySQL Tutorial - Introduction." ThinkHost. 6/6/2004 <http://www.thinkhost.com/services/kb/php-mysql-tutorial.shtml>.

Friedl, Jeffery E. F. Mastering Regular Expressions, Second Edition. Sebastopol, CA: O'Reilly & Associates Inc., 2002.

Hughes, Sterling."Top 21 PHP Programming mistakes – Part I: Seven Textbook Mistakes" 8/13/2000. Zend.com. 3/18/2004 <http://www.zend.com/zend/art/mistake.php>.

Hughes, Sterling."Top 21 PHP Programming mistakes – Part II: Seven Textbook Mistakes" 9/7/2000. Zend.com. 3/18/2004 <http://www.zend.com/zend/art/mistake.php>.

Kabir, Mohammed J. Secure PHP Development: Building 50 Practical Applications. Indianapolis, IN: Wiley Publishing, Inc., 2003.

Leyden, John."Mossad website 'hacker' walks free." 3/1/2004. The Register. 5/26/2004 <http://www.theregister.co.uk/2004/03/01/mossad_website_hacker_walks_free/>.

PHP Documentation Group, The. PHP Manual. Zend Technologies, Inc. 2003.

Rebêlo, Paulo."More emerges about Brazilian hacking hacker." 5/21/2004. Breakthrough Publishing Ltd. 5/26/2004 <http://www.theinquirer.net/?article=16073>.

SecuriTeam.com. "SQL Injection Walkthrough". 5/26/2002. Beyond Security Ltd. 5/10/2004 <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>.

Spett, Kevin. "SQL injection." 2002. SPI Dynamics, Inc. 5/10/2004 <http://www.spidynamics.com/whitepapers/WhitepaperSQLInjection.pdf>.

Jupitermedia Corporation. "SSH." 2004. Jupitermedia Corporation. 6/6/2004 <http://isp.webopedia.com/TERM/S/SSH.html>.