



AUTHENTICATION SCHEMES FOR SESSION PASSWORDS USING COLOR AND IMAGES

¹Mr.S.Janakiraman, ²Sanjana A

¹Assistant Professor, II MCA

^{1,2}Department of Master of Computer Applications,

^{1,2}Er.Perumal Manimekalai College of Engineering, Hosur,

Abstract: Most people still rely on typing passwords to authenticate themselves online. Although they are popular such devices can be hacked with eavesdropping, dictionary attacks, social engineering and shoulder surfing. Having these vulnerabilities means using static textual passwords for security could no longer be trusted. A solution to these problems is the use of graphical passwords. Improved usability thanks to using visual memory does not stop shoulder surfing attacks which can allow an attacker to easily read someone's password. Therefore such encryption is not fully adequate in secure situations, especially where others may be watching. Two approaches are presented in this paper that blend text with images or colors to create session passwords. With every attempt to login, the passwords are different and meant to block people who try to see your session details. Since the screen space and standard way of using lock screens is limited on PDAs, the new methods are especially suitable for them. Carrying out these techniques helps security while still providing users with an easy way to work. Using a different session password for every session blocks reuse and decreases the chances of standard cyber attacks. Having colors and text in your passwords gives attackers more reasons to struggle, even if they spy on your logging in.

Keywords: PHP, MySQL, Apache Server, HTML/CSS & Bootstrap

INTRODUCTION

By using authentication, digital systems are protected from the first attempt to access them unlawfully. In the past, passwords made from text met basic security requirements because they were easy, cheaper and known to everyone. As digital worlds grow and threats become more advanced, textual passwords have revealed important weaknesses. These systems have various risks such as being attacked by eavesdroppers, dictionaries or brute forces as well as shoulder surfers. Often, people choose simple and easy-to-guess passwords and then use the same ones on more than one device which greatly increases their risk. Consequently, people and computers are both easy targets for those looking to compromise accounts and systems.

In order to improve on textual passwords, scientists have suggested new techniques such as graphical password schemes. With these, users resolve verification by preceding a path of pictures, designs or elements rather than entering text or numbers. Even though this design style helps users remember and use the site more easily, it has its shortcomings. Graphical passwords, in many cases, are as easily susceptible to shoulder surfing as textual ones, as their symbols are more visible to attackers than letter combinations. Moreover, these approaches could need more screen area or more advanced computing, so they may not fit on mobile or embedded machines.

PROPOSED WORK

With this proposed system, users authenticate through colors and shown text when they receive one-time passwords. Two methods are involved: the Pair-Based Authentication Scheme and the Hybrid Textual Authentication Scheme and both sets produce a fresh password for each login. As a result, security is increased and the system becomes hard to observe or replay.

2. Following the Pair-Based Scheme, a user generates a secret phrase that is harvested using randomly created 6x6 grids. Under the Hybrid Scheme, a user is required to sort the colors given which curve and stretch along with changing digit grid patterns to form the session passwords. Both ways make sure you do not enter static credentials which stops people from shoulder surfing.

MODULES

1. User Registration Module

Collects user details, secret password, and color preferences.

2.Pair-Based Authentication Scheme Module

Generates session password using character pairs and a dynamic 6x6 grid.

3.Hybrid Textual Authentication Scheme Module

Generates session password using user-defined color ratings and an 8x8 color-number grid.

4.Login & Verification Module

Displays interface (grid/color), collects session password, and verifies it.

5.Session Password Generation Module

Core logic for generating one-time session passwords.

6.Database Management Module

Handles secure storage of user credentials, passwords, and session data using MySQL.

7.User Interface Module

Front-end design for registration, login, and authentication pages (built using PHP/HTML/CSS).

SOFTWARE DESCRIPTION**1. Overview of PHP****History**

In 1995, RasmusLerdorf, an independent software development contractor, made a Perl/CGI script to keep track of those visiting his résumé online. We used his script to collect information about site visitors and show the number of people visiting the Web page. Since Leadoff's scripts were popular in the early Web, I began to receive emails asking how to use them. So, Lerdorf started freely sharing his toolkit, referred to as Personal Home Page (PHP).

Because many people were interested in the PHP toolset, Lerdorf decided to add to it by creating a function that would turn the data entered in an HTML form into symbols that let users share that data elsewhere. Therefore, in order to accomplish that, he selected C over Perl for programming. By November 1997, various updates to the PHP toolkit had

resulted in the arrival of PHP 2.0, later called Personal Home Page — Form Interpreter (PHP-FI). Consequently, the 2.0 upgrade included extensive enhancements made by many worldwide programmers.

The new PHP version became very popular and not long after, a main development team joined Lerdorf. PHP 3.0 came about when its creators continued to include code with HTML and simplified the way the program was processed. At the time the 3.0 update was released in June 1998, more than 50,000 people were including PHP in their Web pages.

In the two years that followed, development moved fast, new features were added in huge numbers and the number of users surged. According to Net craft, a reliable estimate showed that in 1999, PHP had already become the most popular scripting language, with over 1,000,000 users. First, it pleased developers a lot by quickly becoming known as a server-side solution and then it amazed them when users showed that they planned to use it for applications of even larger size. Because of the efforts of ZeevSuraski and Andi Gutmans, PHP core developers decided to rethink the language and built a brand new PHP parser known as the Zend scripting engine. The aim was achieved with the introduction of PHP 4.

PHP is often well summarized by this statement. Still, there are many technical terms you might not be familiar with. You can also view PHP as a strong scripting language that works without your visitors noticing.

When people load your PHP webpage, the web server runs and executes its code behind the scenes. It selects which pieces of the program to display to visitors and hides the rest and then changes your PHP code into HTML. After being translated, the HTML output is sent on to the web browser displayed on your visitor's computer.

How PHP is put into use.

In the year 2000, less than one thousand websites were built with PHP. In 2007, the number went over two million and this number keeps getting bigger. At present, the most visited and used websites are built with PHP code.

PHP is a scripting language that is designed for web applications. A web server is where PHP is generally found. The requested PHP file is processed by PHP runtime which

usually creates the content of the web page dynamically. It is also suitable for both creating scripts at the command-line and for building GUI applications. You can find PHP on most servers, a variety of operating systems and platforms and it works well with relational database management systems. It is free to download and those interested can receive the complete code from the PHP Group to customize and extend as they wish.

PHP Security:

The National Vulnerability Database stores all vulnerabilities found in computer software. The overall proportion of PHP-related vulnerabilities on the database amounted to: 20% in 2004, 28% in 2005, 43% in 2006, 36% in 2007, and 35% in 2008. Most of these PHP-related vulnerabilities can be exploited remotely: they allow hackers to steal or destroy data from data sources linked to the web server (such as an SQL database), send spam or contribute to DOS attacks using malware, which itself can be installed on the vulnerable servers.

These vulnerabilities are caused mostly by not following best practice programming rules: technical security flaws of the language itself or of its core libraries are not frequent (23 in 2008, about 1% of the total). Recognizing that programmers cannot be trusted, some languages include taint checking to detect automatically the lack of input validation which induces many issues. Such a feature is being developed for PHP, but its inclusion in a release has been rejected several times in the past.

5.2 and earlier:

Functions are not first-class functions and can only be referenced by their name, directly or dynamically by a variable containing the name of the function. User-defined functions can be created at any time without being prototyped. Functions can be defined inside code blocks, permitting a run-time decision as to whether or not a function should be defined. Function calls must use parentheses, with the exception of zero argument class constructor functions called with the PHP new operator, where parentheses are optional. PHP supports quasi-anonymous functions through the create function () function, although they are not true anonymous functions because anonymous functions are

nameless, but functions can only be referenced by name, or indirectly through a variable \$function name (); in PHP.

Speed optimization:

As with any interpreted language, PHP scripts are stored as human-readable source code and are compiled on-the-fly by the PHP engine. In order to speed up execution time and not have to compile the PHP source code every time the webpage is accessed, PHP scripts can also be stored in binary format using PHP compilers such as phc and road send.

Code optimizers aim to reduce the runtime of the compiled code by reducing its size and making other changes that can reduce the execution time with the goal of improving performance. The nature of the PHP compiler is such that there are often opportunities for code optimization, and an example of a code optimizer is the eAccelerator PHP extension.

Another approach for reducing overhead for high load PHP servers is using an Opcode cache. Opcode caches work by caching the compiled form of a PHP script (Opcode) in shared memory to avoid the overhead of parsing and compiling the code every time the script runs. An Opcode cache, APC, will be built into PHP 6. Opcode caching is also available in Zend Server Community Edition.

2. MySQL:

The Structured Query Language (SQL) is a very popular database language, and its standardization makes it quite easy to store, update and access data. One of the most powerful SQL servers out there is called MySQL and surprisingly enough, its free.

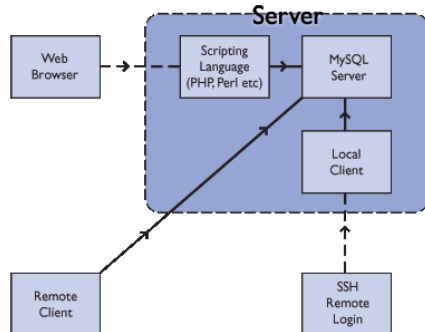
Some of the features of MySQL Include:

- Handles large databases, in the area of 50,000,000+ records.
- No memory leaks. Tested with a commercial memory leakage detector (purify).
- A privilege and password system which is very flexible and secure, and which allows host-based verification.
- Passwords are secure since all password traffic when connecting to a server is encrypted.

Even several corporate web sites have started using MySQL

How does MySQL work?

MySQL is a database server program and as such is installed on one machine, but can 'serve' the database to a variety of locations. To explain look at the following diagram.



The MySQL Server is installed on a Server and can be accessed **directly** via various client interfaces, which send SQL statements to the server and then display the results to a user. Some of these are:

A Local Client – is a program on the same machine as the server. An example of this is the command line MySQL client software we will be using in the rest of the MySQL workshops (although there are other programs including graphical interfaces).

Uses:

Many web applications use MySQL as the database component of a LAMP software stack. Its popularity for use with web applications is closely tied to the popularity of PHP, which is often combined with MySQL. Several high-traffic web sites (including Flickr, Facebook, Wikipedia, Google. (though not for searches), Nokia and You Tube) use MySQL for data storage and logging of user data.

Features:

As of April 2009, MySQL offers MySQL 5.1 in two different variants: the MySQL Community Server and Enterprise Server. They have a common code base and include the following features:

- A broad subset of ANSI SQL 99, as well as extensions
- Cross-platform support

- Stored procedures
- Triggers
- Cursors
- Updatable Views
- True Varchar support
- INFORMATION_SCHEMA
- Strict mode
- X/Open XA distributed transaction processing (DTP) support; two phase commit as part of this, using Oracle's InnoDB engine
- Independent storage engines (MyISAM for read speed, InnoDB for transactions and referential integrity, MySQL Archive for storing historical data in little space)
- Transactions with the InnoDB, BDB and Cluster storage engines; save points with InnoDB
- SSL support
- Query caching
- Sub-Selects (i.e. nested Selects)
- Replication support (i.e. Master-Master Replication & Master-Slave Replication) with one master per slave, many slaves per master, no automatic support for multiple masters per slave.

MySQL Architecture

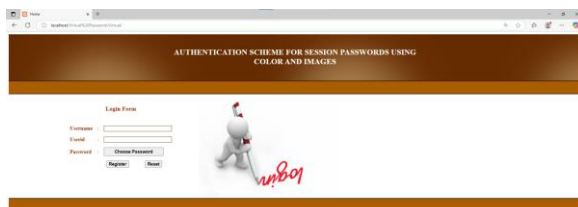
It will greatly aid your thinking about storage engines and the capabilities they bring to MySQL if you have a good mental picture of where they fit. Figure 2-1 provides a logical view of MySQL. It doesn't necessarily reflect the low-level implementation, which is bound to be more complicated and less clear cut. However, it does serve as a guide that will help you understand how storage engines fit in to MySQL. (The NDB storage engine was added to MySQL just before this book was printed. Watch for it in the second edition.)

The topmost layer is composed of the services that aren't unique to MySQL. They're services most network-based client/server tools or servers need: connection handling, authentication, security, etc.

The second layer is where things get interesting. Much of the brains inside MySQL live here, including query parsing, analysis, optimization, caching, and all the built-in functions (dates, times, math, encryption, etc.).

RESULTS

HOME PAGE



SIGN IN PAGE



PASSWORD CREATE PAGE



CONCLUSION

In today's digital world, securing user authentication has become increasingly critical due to the rise in cyberattacks, including dictionary attacks, brute force, and shoulder surfing. Traditional textual password systems are vulnerable to such attacks as users often choose simple or predictable passwords. This project proposes an innovative approach to enhance authentication security by generating session-based one-time passwords using a combination of text, colors, and graphical interfaces.

The implemented system introduces two novel authentication methods—Pair-Based Authentication and Hybrid Textual Authentication. Both techniques focus on generating unique passwords for each session, reducing the possibility of password theft. The Pair-Based scheme uses a character grid to derive session passwords from character pairs, while the Hybrid method leverages user-assigned color priorities to generate dynamic passwords based on a colored grid. These methods are intuitive yet highly secure, making them resistant to common password-related vulnerabilities.

ACKNOWLEDGMENT

The authors declare that they have no reports of acknowledgments for this

REFERENCES

1. Sobrado, L., & Birget, J.-C. (2002). *Graphical passwords*. The Rutgers Scholar, 4, 1–8.
2. Wiedenbeck, S., Waters, J., Birget, J. C., Brodskiy, A., & Memon, N. (2005). *PassPoints: Design and longitudinal evaluation of a graphical password system*. International Journal of Human-Computer Studies, 63(1–2), 102–127.
3. Davis, D., Monroe, F., & Reiter, M. K. (2004). *On user choice in graphical password schemes*. USENIX Security Symposium.
4. Yan, J., Blackwell, A., Anderson, R., & Grant, A. (2004). *Password memorability and security: Empirical results*. IEEE Security & Privacy, 2(5), 25–31.

5. Zhao, H., & Li, X. (2007). *S3PAS: A scalable shoulder-surfing resistant text graphical password authentication scheme*. 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW).
6. Gao, H., Liu, X., Wang, S., & Liu, R. (2009). *Design and analysis of a graphical password scheme*. 2009 International Conference on Innovative Computing, Information and Control (ICICIC).
7. Tari, F., Ozok, A. A., & Holden, S. H. (2006). *A comparison of perceived and real shoulder-surfing risks between alphanumeric and graphical passwords*. Proceedings of the second symposium on Usable privacy and security.
8. Al-Ameen, M. N., & Wright, M. (2014). *A comprehensive study of the usability of graphical passwords*. Journal of Information Security and Applications, 19(4–5), 224–239.
9. Chiasson, S., Forget, A., Biddle, R., & van Oorschot, P. C. (2008). *Influencing users towards better passwords: Persuasive cued click-points*. British HCI Group Annual Conference on People and Computers.