Team: Minal Kondawar
Rosy Agarwal

Final Report

User Verification System

Abstract

Authentication is a vital part of a network's security scheme. It is the mechanism for ensuring that the identity of a user, computer, or service is valid. There are a number of ways that authentication can be accomplished, depending on network operating system and connection type.

The user verification system is a system which securely store the password of the system and verifies the user's identity and remind the user with forgotten password and username. The fact that plain password authentication is still by far the most widely used form of authentication, gives credence to the seriousness of the lack of security on both the Internet and within private networks. The user verification system uses the hash algorithm ensuring that the operation cannot be reverse engineered to obtain the original password from the hash results. The only hashing algorithm can make it insecure. So the passwords are hashed along with salt and attack is made slow using key stretching. Further encrypting the processed password makes it more secure. Thus making the user account secure. The user is verified checking the credentials against the stored data. One of the major aspect with the user account is forgetfulness of username and password. Providing the username or password on request makes the system vulnerable. Therefore, the reminding mechanism such as security/challenge questions along with CAPTCHA gives access only to authorized user of the system. The security questions plays a very important role in verifying the user. The security question which are personal as well as long enough to guess makes the attack less effective. The password reset link for setting up the new password in user's email adds the one more security level.

Background

Many web applications do not properly protect sensitive data, such as credit cards, SSNs, and authentication credentials, with appropriate encryption or hashing. Attackers may steal or modify such weakly protected data to conduct identity theft, credit card fraud, or other crimes. Applications help users to store and recover passwords. It is very important to keep passwords safe in the database as databases are prone to various attacks. A good approach is to apply password hashing and then encrypt the password. Various cryptographic algorithms are available to perform this task e.g MD5, SHA1 etc

The MD5 message-digest algorithm is been widely used cryptographic hash function producing a 128-bit (16-byte) hash value. It is expressed in text format as a 32 digit hexadecimal number. MD5 has been utilized in a wide variety of cryptographic applications, and is also commonly used to verify data integrity. The algorithm suffers from several drawbacks. The size of the hash value (128 bits) is small enough to contemplate a birthday attack. If there is a collision attack it can find collisions within seconds on a computer with a 2.6 GHz Pentium 4 processor. Further, there is also a chosen-prefix collision attack that can produce a collision for two inputs with specified prefixes within hours, using off-the-shelf computing hardware.

Collision Vulnerabilities: It means that there exist two (or more) inputs m1 and m2 such that MD5 will generate the same output from these distinct and different inputs i.e. hash (m1) = hash(m2).

Pre-image Vulnerability: MD5 is broken because it fails being a cryptographic hash. It does not any longer have second pre-image resistance. Suppose there exist: h = MD5(d)

where d is a document and h is the MD5-hash of it. There can be a d1 such that MD5(d1) = h. This means there exists a second preimage that claims to have the same hash. That means one can make a document and alter hidden parts of it until the same hash

value is obtained. That means a false document can be created which can be hash-checked to make it original. Thus MD5 is not safe to use and key derivation for passwords PBKDF2, bcrypt and salt must be used. Another solution is using HMAC algorithms.

Other Vulnerabilities: A number of projects have published MD5 rainbow tables online, which can be used to reverse many MD5 hashes into strings that collide with the original input, usually for the purposes of password cracking. The use of MD5 in some websites' URLs means that search engines such as Google can also sometimes function as a limited tool for reverse lookup of MD5 hashes.

Overview

There are several physical means by which you can provide your authentication credentials to the system. Today's competitive business environment demands options that offer more protection when network resources include highly sensitive data. Existing authentication system are:

Password authentication: There are many system which store the password in plain text which is vulnerable to SQL injections, also if the backup is not taken properly or any other slipups my lead the customers password in public domain. The average chance that all their passwords for all their other accounts on totally independent systems are in the public domain.

Smart card authentication: Smart cards are credit card-sized devices that hold a small computer chip, which is used to store public and private keys and other personal information used to identify a person and authenticate him or her to the system. Logging onto the network with a smart card requires that you physically insert the card into (or slide it through) a reader and then enter a Personal Identification Number. Smart cards use cryptography-based authentication and provide stronger security than a password because in order to gain access, **but** the user must be in physical possession of the card and must know the PIN.

Biometric authentication: The more secure type of authentication than smart cards is biometric authentication that involves the use of biological statistics. It shows that the probability of two people having identical biological characteristics such as fingerprints is infinitesimally small. In addition to fingerprints, voice, retinal, and iris patterns are virtually unique to each individual and can be used for authentication purposes. But it requires expensive equipment to input the fingerprint, voice sample, or eye

Authentication methods and protocols: There are a large number of authentication methods and protocols that can be used, depending on the application and security requirements. In the following sections, we will discuss:

- Kerberos
- SSL
- PAP and SPAP
- CHAP

Kerberos: Kerberos was developed at MIT to provide secure authentication for UNIX networks. It has become an Internet standard and is supported by Microsoft's latest network operating system, Windows 2000. Kerberos uses temporary certificates called tickets, which contain the credentials that identify the user to the servers on the network. In the current version of Kerberos, v5, the data contained in the tickets is encrypted, including the user's password. A Key Distribution Center (KDC) is a service that runs on a network server, which issues a ticket called a Ticket Granting Ticket (TGT) to the clients that authenticates to the Ticket Granting Service (TGS). The client uses this TGT to access the TGS (which can run on the same computer as the KDC). The TGS issues a service or session ticket, which is used to access a network service or resource.

Drawbacks:

- Password guessing: no authentication is required to request a ticket, hence attacker can gather equivalent of /etc/passwd by requesting many tickets.
- Kerberos 5 CRC-32 checksum is not collision proof (as MD4 is thought to be)
- K5 still uses timestamps to authentication KRB_SAFE and KRB_PRIV messages; should use sequence numbers.

Secure Sockets Layer (SSL): The SSL protocol is another Internet standard, often used to provide secure access to Web sites, using a combination of public key technology and secret key technology. Secret key encryption (also called symmetric encryption) is faster, but asymmetric public key encryption provides for better authentication, so SSL is designed to benefit from the advantages of both. SSL authentication is based on digital certificates that allow Web servers and clients to verify each other's identities before they establish a connection. (This is called mutual authentication.)

Drawbacks: Performance tests have revealed that using SSL between the agent and the runtime server, for example, increases network traffic three-fold, and can reduce the speed of response of the runtime server to agent requests by from two to ten times, depending largely on the agent platform.

PAP: PAP is used for authenticating a user over a remote access control. It sends user passwords across the network to the authenticating server in plain text. This poses a significant security risk, as an unauthorized user could capture the data packets using a protocol analyzer (sniffer) and obtain the password. The advantage of PAP is that it is compatible with many server types running different operating systems. PAP should be used only when necessary for compatibility purposes.

SPAP: SPAP is an improvement over PAP in terms of the security level, as it uses an encryption method. The client sends the user name along with the encrypted password, and the remote server decrypts the password. If the username and password match the information in the server's database, the remote server sends an Acknowledgment (ACK) message and allows the connection. If not, a Negative Acknowledgment (NAK) is sent, and the connection is refused.

CHAP and MS-CHAP: CHAP is another authentication protocol used for remote access security. It is an Internet standard that uses MD5, a one-way encryption method, which performs a hash operation on the password and transmits the hash result, instead of the password itself over the network.

MS-CHAP is Microsoft's version of CHAP. MS-CHAPv2 uses two-way authentication so that the identity of the server, as well as the client, is verified. This protects against server impersonation. MS-CHAP also increases security by using separate cryptographic keys for transmitted and received data.

Security Questions: The security question are the mechanism through which the user is verified whether he is a valid user. It is the mechanism through which user can reset his password. Thus, play an important role in increasing the security. For many existing system the question are too obvious to guess such as What is the color of sky or so general what is name of ur mother or teacher. Making it easy to crack. So, after guessing it right, system allows to reset the link without providing any layer of security

Many existing system, on forgot password, send the password on unsecure channel to users email. This is very vulnerable. The original password should never be stored and moreover password should not be provided rather should facilitate to reset new password.

Our work

Module1: Account creation:

Username: The user should provide unique username

Password: After reading papers and articles, we studied the points which could make the password strong from user-side.

User should create password which is strong and should have at least with following requirement:

1. The password should be at least 8 character long, but restriction on exact 8 character gives the attacker the length of the password which is bad.
2. The password should can contain one upper-case character, one lower-case character, one number and one special character.
3. The user should use bad grammar which is not the word in dictionary.
4. The user should use words that are not complete.
5. The user should use a paraphrase-include acronym
6. The user can make length of password long by padding with the symbols e.g. c-@T--9---,easy to remember but not in dictionary.

Security Question: We read many research papers regarding the good security question to increase the security of the system. Depending upon study, the strategy we implemented for security question

The questions should be such that:

1. The question are very personal to user such that answer are known to them only decreasing the probability to guess.
2. Easy to remember, even 5 or 10 yrs. from now
3. At least thousands of possible answers which will increase the guessing space and make attack slow
4. Not a question you would answer on Facebook, Myspace, in a "Fun Questions to Ask" survey, or in an article or interview making guessing difficult
5. The questions are framed the way the answers are not one word. They are forced to answer in a paraphrase. Longer the length, slower the attack
6. The questions are fixed because if the user is given option to frame their own question, there is a probability they end up with simple question for example: what is my favorite color?. Thus making the guessing of password easy.

Security Answers: The important aspect for security answers is that it should be long in length, more than a word, a paraphrase and user should use bad grammar making the dictionary attack ineffective.

Module 2: Security Measures on Server Side: Hashing with Salt

As user accounts are hacked frequently it is important to store user information in database in a secure way. It is absolutely necessary to safeguard the users' passwords if website is ever breached. The best way to protect passwords is to employ salted password hashing. Hash algorithms are one way functions. They can convert any amount of data into a fixed-length "fingerprint" that cannot be reversed. Also they have the quality if the input changes by even a tiny bit, the resulting hash is completely different. This is a great feature for protecting passwords, because we want to store passwords in an encrypted form that's impossible to decrypt, but at the same time, we are also able to verify that a user's password is correct.

The general workflow for account registration and authentication in a hash-based account system is as follows:

- The user creates an account.
- Their password is hashed and stored in the database. At no point is the plain-text (unencrypted) password stored in the backend.

- When the user tries to login, the hash of the password entered is verified against the hash of their real password (retrieved from the database).
- If the hashes match, the user is granted access. If not, the user is prompted for invalid login credentials.
- Steps 3 and 4 is repeated every time someone tries to login to their account.

Cryptographic hash functions like SHA256, SHA512, Ripe MD, and WHIRLPOOL may be used to implement password hashing.

Adding salt

Salt allows to randomize the hashes by appending or prepending a random string to the password before hashing. Salt is required to check if the password is correct. As a result it is usually stored in the user account database along with the hash, or as part of the hash string itself.

Salt should be generated using a **Cryptographically Secure Pseudo-Random Number Generator** (CSPRNG). CSPRNGs are cryptographically secure as they provide high level of randomness and are completely unpredictable.

Algorithm:

1. To Store a Password

• Generate a long random salt using a CSPRNG.
• Prepend the salt to the password and hash it with a standard cryptographic hash function such as SHA256.
• Save both the salt and the hash in the user's database record.

2. To Validate a Password

• Retrieve the user's salt and hash from the database.
• Prepend the salt to the given password and hash it using the same hash function.
• Compare the hash of the given password with the hash from the database. If they match, the password is correct. Otherwise, the password is incorrect.

2. Security measures on Client Side: Key Stretching

As high-end graphics cards (GPUs) and custom hardware can make malicious attacks very effective by computing billions of hashes per second, we use a technique known as key stretching. The goal is to make the hash function slow enough to impede attacks, but still fast enough to not cause a noticeable delay for the user. Key stretching is implemented using standard algorithm like PBKDF2 or bcrypt wich is a special type of CPU-intensive hash function. These algorithms take a security factor or iteration count as an argument. This value determines how slow the hash function will be.

To the hash As long as an attacker can use a hash to check whether a password guess is right or wrong, the attacker can run a dictionary or brute-force attack on the hash. So we added a secret key to hash so that only someone who knows the key can use the hash to validate a password. We used keyed hash algorithm: HMAC

Module 3: Forgot Username

• User provides email address which is validated in database
• Username is sent to email address

Module 4: Forgot password

1st level of security: User needs to provide email address which is used to authenticate the user.
2nd level of security: User needs to provide answer to security question which is validated in database
3rd level of security: Access code is sent to user email id. Without access code, the user cannot create new password
4th level of security: User enters new password and displayed CAPTCHA image.

CAPTCHA is an identification measure to identify is one is human or a robot. The intention to use it is to avoid the automated submission of forms which could be used as an attempt to breach security. When a CAPTCHA is used, it means that it can't be brute-forced either to spam an individual or to attempt to identify the existence of accounts.

Module 5: Forgot username and password both: First needs to get username through Module 3 above followed by Module 4.

We learnt about:

1. **Security concepts** such as Encryption/Decryption, Hashing, Salt, Key stretching.
2. **Attacks**: dictionary attacks, brute-force attacks, lookup table attacks, rainbow table attack.
3. **Security Algorithms** (includes advantages and disadvantages): MD5, SHA1, SHA256, bcrypt, PBKDF2, HMAC.
4. We also learnt about existing secure system, their comparison, advantages and disadvantages.
5. **Technology**: Java, JSP, Servlet, MVC Model, MySQL,
6. **Platform**: Eclipse, WAMP

Evaluation:

The goals of the project that were set out to achieve:

Module 1: User Account Creation: User creates an account in User Verification System by providing personal information such as: Username, Password, Email address and Security Questions and Answers

Module 2: Security Measures on Server Side: Hashing with Salt and Key Stretching

Module 3: Forgot Username: User provides email address which is validated in database and username is sent to email address

Module 5: Forgot password
- User provides email address which is validated in database
- User provides answer to security question which is validated in database
- Reset link is sent to email address which redirects to reset page (Alternate solution: Access code was implemented)
- User enters new password and displayed CAPTCHA image.

We successfully implemented all the above goals except the one below:
Reset link is sent to email address which redirects to reset page: We were able to successfully send the reset link to user but were not able to connect back to our system. We tried a lot but due to technology issues we were not able to implement it.
We found the alternative to it i.e**. Access code.** When user is verified against the database, the access code is sent to his email id. Without this access code, the user cannot reset his password. Thus increasing the security.

1. User Verification system prevents the attackers from enumerating valid usernames without knowing their password as we never tell the user if it was the username or password they got wrong by displaying the generic message “Invalid username or password”
2. When the user registers, the password is stored in database in an encrypted format (using Hashing) i.e it is not a plain text. So even if someone gets an unauthorized access to DB he will not be able to login to the application.
3. Our system allows two or more users to have same password. This is because passwords are hashed with salt. Salt is be generated using a Cryptographically Secure Pseudo-Random Number Generator (CSPRNG) which randomize the hashes by appending or prepending to password.
4. The system avoids hash collision.
5. The system makes it impossible for an attacker to create a lookup table for every possible salt, size of salt is same as the output of the hash function.

1. As system uses salt, it ensures that attackers can't use specialized attacks like lookup tables, rainbow tables, reverse lookup tables.
   Lookup table attack: The general idea is to pre-compute the hashes of the passwords in a password dictionary and store them, and their corresponding password, in a lookup table data structure. A good implementation of a lookup table can process hundreds of hash lookups per second, even when they contain many billions of hashes.

```
Searching:          5f4dcc3b5aa765d61d8327deb882cf99:          FOUND:          password5
Searching:      6cbe615c106f422d23669b610b564800:          not      in      database
Searching:          630bf032efe4507f2c57b280995925a9:          FOUND:          letMEin12
Searching:          386f43fab5d096a7a66d67c8f213e5ec:          FOUND:          mcd0nalds
Searching: d5ec75d5fe70d428685510fae36492d9: FOUND: p@ssw0rd!
```

2. Reverse lookup tables: First, the attacker creates a lookup table that maps each password hash from the compromised user account database to a list of users who had that hash. The attacker then hashes each password guess and uses the lookup table to get a list of users whose password was the attacker's guess. This attack is especially effective because it is common for many users to have the same password.

```
Searching for hash(apple) in users' hash list...      : Matches [alice3, 0bob0, charles8]
Searching for hash(blueberry) in users' hash list... : Matches [usr10101, timmy, john91]
Searching for hash(letmein) in users' hash list...    : Matches [wilson10, dragonslayerX,
joe1984]
Searching for hash(s3cr3t) in users' hash list...    : Matches [bruce19, knuth1337, john87]
Searching for hash(z@29hjja) in users' hash list...  : No users used this password
```

3. Rainbow attack: Rainbow tables are a time-memory trade-off technique. They are like lookup tables, except that they sacrifice hash cracking speed to make the lookup tables smaller. Because they are smaller, the solutions to more hashes can be stored in the same amount of space, making them more effective. Rainbow tables that can crack any md5 hash of a password up to 8 characters long exist.

6. The system implements the key stretching which makes dictionary and brute force attack less effective by making the hash function slow using CPU-intensive hash function PBKDF2.
7. The system uses HMAC which prevent an attacker from being able to modify the message and replace it with a different valid hash.
8. The system uses Keyed hashes so that only someone who knows the key can use the hash to validate a password
9. There is no way to prevent dictionary attacks or brute force attacks. They can be made less effective, but there isn't a way to prevent them altogether. Even if password hashing system is secure, the only way to crack the hashes will be to run a dictionary or brute-force attack on each hash.
   Dictionary attack and Brute force attack

```
                    Dictionary Attack

Trying apple        : failed
Trying blueberry    : failed
Trying justinbeiber : failed
                              ...
Trying letmein      : failed
Trying s3cr3t       : success!
                     Brute Force Attack

Trying aaaa : failed
```

```
Trying aaab : failed
Trying aaac : failed
                              ...
Trying acdb : failed
Trying acdc : success!
```

10. The system uses security questions which are too personal to the user. It is very likely that only user know the answers to such questions. Thus even if someone knows the questions he will not be able to guess the right answers to them.
11. The system implements additional layer of security in the form of access code. The access code is sent to user's e-mail address. Only when the correct access code is entered user will be able to proceed to reset his password.
12. The system implements CAPTCHA which helps to verify whether the request to change the password is human or a bot is trying to illegally manipulate the user account

Future Work

The project can be expanded to include One Time Password (OTP) functionality. A one-time password (OTP) is a password that is valid for only one login session or transaction. OTP is better than static passwords as they are not vulnerable to replay attacks. This means that if an adversary tries to intrude into the system by re-using a OTP he will not be able to as they are valid for one time use only. On the downside, OTPs are difficult for human beings to memorize and require additional technology (mobile and e-mail) to work.

OTP generation use of pseudo randomness or randomness. This will make predicting future OTPs difficult (by observing previous ones). Various approaches for the generation of OTPs are:

Based on **time-synchronization** between the authentication server and the client providing the password (OTPs are valid only for a short period of time)

Using a **mathematical algorithm** to generate a new password based on the previous password (OTPs are effectively a chain and must be used in a predefined order).

Using a **mathematical algorithm** where the new password is based on a challenge (e.g., a random number chosen by the authentication server or transaction details) and/or a counter.

Contributions:

Minal Kondawar

1. User account creation (Hashing algorithm)
2. Login verification
3. Username to user's email address

Rosy Agarwal:

1. Implemented CAPTCHA and reset password
2. Security Questions
3. Access Code to user's email address

Conclusion:

We learned various security concepts and cryptographic algorithm and created the system which stores the user information in secured manner, verifies the user identity and provides the mechanism to reset the forgotten password and recover the username. However, the system is secure against attacks such as lookup tables, reverse lookup tables and rainbow table's attacks, there is no way to prevent dictionary attacks or brute force attacks. They can be made less effective, but there isn't a way to prevent them altogether. Even if password hashing system is secure, the only way to crack the hashes will be to run a dictionary or brute-force attack on each hash. Our system implements the security questions for resetting the password which are very personal and more likely known only to user. Also forces the user to provide answers which are more than word thereby increasing the complexity. Further, CAPTCHA protects the systems against bots.

References

http://www.cs.cmu.edu/~agrao/paper/Effect_of_Grammar_on_Security_of_Long_Passwords.pdf

http://www.ijaiem.org/Volume2Issue6/IJAIEM-2013-06-22-060.pdf

https://www.usenix.org/legacy/confadmin/leet08/papers/L12/content.pdf

http://www.cs.utexas.edu/~shmat/shmat_ccs05pwd.pdf

http://home.cyber.ee/ahtbu/CDS2011/PredragTasevskiSlides.pdf

http://security.stackexchange.com/questions/43023/secure-authentication-partial-client-side-key-stretching-please-review-criti

http://tools.ietf.org/html/rfc2104

https://crackstation.net/hashing-security.htm

http://www.lightbluetouchpaper.org/2010/03/04/evaluating-statistical-attacks-on-personal-knowledge-questions/

http://cdn.ly.tl/publications/text-based-captcha-strengths-and-weaknesses.pdf