2016 IEEE 36th International Conference on Distributed Computing Systems

Amnesia: A Bilateral Generative Password Manager

Luren Wang*, Yue Li*, Kun Sun
Department of Computer Science
College of William and Mary
lurenwang@gmail.com, {yli,ksun}@cs.wm.edu

***Abstract*—While numerous flaws have been recognized in using passwords as a method of authentication, passwords still remain the de-facto authentication standard in use today. Though password managers can ameliorate password fatigue, the vast majority of password managers require the user to choose and maintain a strong master password while offering little to no recourse in the event that the master password is compromised. The wide-application of cloud-based password managers congregate passwords in an encrypted database, which becomes an attractive target for attackers and also represents a single point of failure. In this paper, we propose Amnesia, a bilateral generative password manager that requires both the knowledge of the master password and the possession of the user's smartphone to generate website passwords for the user. Our generative password manager is not vulnerable to the password database leakage, since it generates the requested password on demand using both the master password and the secret information on the smartphone. An attacker wishing to steal the user's website passwords has to compromise both the user's smartphone and the master password. Amnesia also has strong recovery capability when either the master password is compromised or the smartphone is lost/stolen. By using an Amnesia server, a user can have the access to the password manager on multiple computers without installing any software on those computers. We implemented an Amnesia system prototype using Android and Cherrypy web framework and evaluated it in terms of security, usability, and overhead. A user study of 31 testers shows that Amnesia increases password security while maintaining reasonable user convenience.**

I. Introduction

Despite consensus among security researchers on the numerous problems password based authentication impose, passwords have nevertheless been the main form of electronic authentication for over 50 years [1]. The main issue regarding passwords as a form of authentication is the limitation of human memory. It is proven users have selected very weak passwords that are vulnerable to various attacks [2]–[4]. Furthermore, it is not realistic to depend on users to follow the most secure practice [1], [5]. For example, a password is shown to be shared across 3.9 different sites even though password reuse is known to be an insecure practice [6].

A number of password managers exist which help mitigate password management issues. The two main types of password managers are *retrieval* and *generative*. Retrieval password managers are characterised by an encrypted vault that stores the user's passwords online or offline. The user needs to input the master password in order to access the passwords in the vault. Despite security precautions, online retrieval password databases are attractive targets to attackers and are at risk to data breaches [7]. Meanwhile, offline retrieval password managers incur the danger of leaving multiple copies of encrypted database across several devices. Generative password managers generate user passwords based on information such as site URL, account username, and a master password. Though generative password managers can avoid the data breach problems of retrieval password managers, they are often unwieldy. For instance, some generative password managers force the user to set and memorize a counter that specifies how many times they have changed a password to an account [8]. Both retrieval and generative password managers rely on a single master password as a security anchor. Thus, the master password is a single point of failure in these systems.

In this paper, we develop *Amnesia*, a bilateral generative password manager that enhances the password security by generating a requested password with both the knowledge of a master password and the possession of the user's smartphone. When the user wishes to obtain a password to a particular website account, it first uses its master password to log into an Amnesia web server. After the user clicks the targeted account it wants to use, a password request is sent from the Amnesia web server to the user's smartphone. With the confirmation from the user, the smartphone sends a token of sensitive password data to the server. Then, the server combines the token with its server-side secrets to generate the requested password. Finally, the server sends the generated password to the user's computer.

Amnesia is a bilateral password manager since it splits password generative information between the Amnesia server and the user's smartphone. Since a desktop without wireless communication devices cannot directly contact the smartphone, we provide an Amnesia server to communicate with the smartphone by using a rendezvous server such as Google Cloud Messaging [9]. Note since the Amnesia server usually has static IP address, the smartphone can directly communicate with the Amnesia server without the packet forwarding through the rendezvous server. Moreover, with the deployment of the Amnesia server, a user can have access to the password manager on multiple computers without installing any software on those computers. To access the secrets in the Amnesia server, the user only needs to authenticate itself to the Amnesia web server using its master password. As a

* The first two authors contribute equally to this paper. The idea and most of the paper writing are from Luren Wang, who is mentored by Yue Li. The developments are equally contributed between Luren and Yue.

1063-6927/16 $31.00 © 2016 IEEE
DOI 10.1109/ICDCS.2016.90

IEEE computer society

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 15,2024 at 20:41:19 UTC from IEEE Xplore. Restrictions apply.

generative password manager, Amnesia is not vulnerable to database breaches.

An attacker wishing to steal the user's passwords has to either compromise both the user's smartphone and master password or both the user's smartphone and the Amnesia web server. Amnesia also provides strong recovery capability when either the master password is compromised or the smartphone is lost/stolen. First, when the smartphone is stolen, the user can use the master password and backup data to reset a new smartphone and update the password information on the Amnesia server. Second, when the master password is compromised, the user can use the secret information in the smartphone to change the master password. To prevent the attacker from misusing the stolen smartphone to reset the master password, the user should first input the current master password to authenticate itself.

We implement an Amnesia system prototype using Android to develop the Amnesia mobile application on the smartphone and CherryPy [10] web framework to build up the Amnesia server. The experimental results show that Amnesia only introduces small time latency. We compare Amnesia with other password managers using the comparative evaluation framework developed by Bonneau, et al. [11]. Moreover, we perform a user study consisting of 31 testers to measure the security and usability of Amnesia. The study results show that Amnesia increases password security while maintaining reasonable user convenience.

In summary, we make the following contributions:

1) We propose a bilateral generative password manager architecture that both eliminates password breach vulnerabilities and provides an additional layer of security to the master password by utilizing the user's smartphone.
2) We implement a prototype of Amnesia that includes the Amnesia server and the Amnesia mobile application. We also conduct experiments to show that the time latency of password generation is small.
3) We conduct a user study with 31 participants. The study explores participants' password management habits and measures their disposition towards Amnesia's usability. It shows that Amnesia does increase password security with minimal user involvement during the authentication process.

The remainder of the paper is organized as follows. In section II, we discuss the threat model. Section III describes Amnesia's system architecture while section IV performs security analysis on Amnesia. Section V illustrates the prototype implementation. Section VI includes Amnesia evaluation while section VII details our user study. Section VIII discusses limitations of Amnesia and section IX surveys related works. Finally, section X concludes this paper.

II. Threat Model

We assume the user's smartphone may be stolen, and then the attacker can have full access to the sensitive data in the Amnesia mobile application and the communication between the smartphone and the Amnesia server. Also, we assume the user's master password may be compromised. Though the attackers are able to compromise either the user's smartphone or the master password of the password manager, we assume they cannot compromise both smartphone and the master password without the user noticing and taking reactive measures. Moreover, we assume that the attacker is able to compromise the Amnesia web server. Here, the attacker may gain full access to the data at rest including the server side secrets. However, we do not assume the attacker can read the memory of other processes on the same server.

Since we focus on the security of password manager, we assume the computer that is used to access websites can be trusted. Moreover, the web browser can be trusted. Otherwise, it can see all the passwords in plaintext. We assume all the communication between the browser, the Amnesia server, and the Amnesia mobile application is protected. We also assume the a third-party cloud provider can be trusted and its connection to Amnesia application is secure during the recovery process when a smart phone is lost.

III. System Architecture

Figure 1 shows the system architecture of Amnesia, which consists of three main components, namely, *a web browser on user's computer*, *an Amnesia web server on a remote server*, and *an Amnesia mobile app on the user's smart phone*, which collaborate password management with six steps. First, when the user uses the computer to access a website that requests password for user authentication, the web browser forwards the web domain information such as the URL to the Amnesia server, to which the user has been successfully authenticated. In the third step, the Amnesia server generates a password request to the user's smartphone. However, since the Amnesia server may not be able to directly communicate with the smartphone, it has to use a rendezvous server to help forward the request. Next, the smartphone calculates a token based on the password request and its phone-side secret. Since the Amnesia server's address is fixed, the smartphone can directly send the token to the Amnesia server. In the fifth step, after generating the password using the token and its server-side secret, the Amnesia server will send the password to the browser, which can automatically fill the password into the website authentication page. The process is the same for a user using a mobile browser. In this case, the phone would also take on the role of the PC.

For clarification purposes, we list important notations used in our system as follows.

- *Server-side secret* K_s: Server-side secret is a collection of secrets that are only stored on the Amnesia server. It is used in the computation of a requested password along with a requested token from the phone.
- *Phone-side secret* K_p: Phone-side secret is a collection of secrets that are only stored on the Amnesia application. It is used in the computation of a token which is then sent to the Amnesia server for password generation.

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 15,2024 at 20:41:19 UTC from IEEE Xplore. Restrictions apply.

Fig. 1: Amnesia System Architecture

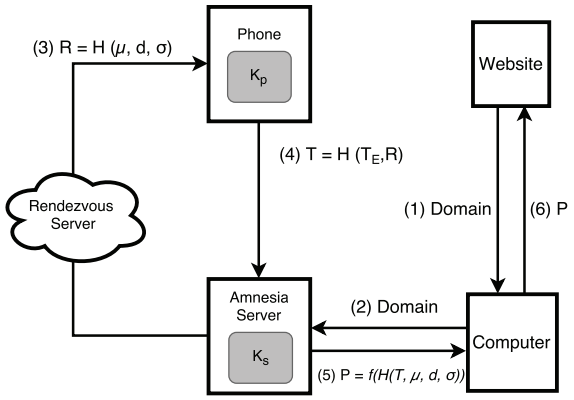


TABLE I: Server Side Data

Data	Value
Username	$UserAlice$
Registration ID	$fl1K4VTKHcc : AP \ldots$
$H(MP + salt)$	$0x6fa2514 \ldots$
$H(P_{id} + salt)$	$0x4a3f321 \ldots$
Salt	$0x19ad452 \ldots$
$(\mu, d, \sigma)_1$	(Alice, mail.google.com, $0xff4323a \ldots$)
$(\mu, d, \sigma)_2$	(Alice2, www.facebook.com, $0xe12341f \ldots$)
$(\mu, d, \sigma)_3$	(Bob, www.yahoo.com, $0xe58ae34 \ldots$)

μ is the account username, d is the domain ID, and σ is the seed.

- *Password request* R: A value that is derived from the server-side secret K_s and the website's domain information.
- *Token* T: A value that is calculated by the smartphone using the phone-side secret K_p and the password request R.
- *Generated password* P: A password that is generated for a particular user web account. The password is computed using both K_s and K_p.
- *Master password* MP: The password the user uses to authenticate to the Amnesia server.

A. Core Components

We elaborate in detail on the three core components - the Amnesia server, mobile application, and user computer - particularly on the variables and secrets stored in each component.

1) User Computer: The user computer itself does not store any variables necessary to generate particular passwords. Rather, the user computer can authenticate to and interact with the Amnesia server with the master password MP, usually in the form of a web browser. MP is the only password that should be remembered by the user. The communication between the computer and the server is protected by HTTPS.

2) Amnesia Server: Amnesia server stores two types of secrets, K_s for generating passwords and functional variables V_f for other functionalities such as authentication and account recovery. Upon registration to the Amnesia server, the user will be associated with a user-chosen username. When the user adds one website password account, an entry is created. The entry is composed of three values, namely, the account domain d, the account username u, and a 256-bit seed σ. The account domain can be anything (for example a URL) that identifies a website or entity that the user has an account on. The account username is simply the username of a particular account on domain d. In brief, the pair (d, u) is used to uniquely identify the user's accounts. σ plays two roles. First, if the user wishes to change its password to account A, it only needs to change σ_i in order to regenerate a new password. Second, σ also provides security when the server generates and sends request R. This aspect is elaborated on in Section III-B. Besides K_s, The server also stores several other functional variables. These are the hash of the salted master password MP for user authentication and *registration id (R_{id})*, which is required for the communication through a rendezvous server. Furthermore, the server also stores a hashed and salted P_{id} to associate a physical mobile phone for recovery. In brief, the server data can be represented as

$$K_s = (\mu, d, \sigma_{dom}), V_f = (H(MP, salt), R_{id}, (H(P_{id}, salt))$$

We summarize the server side data in Table I.

3) Amnesia Mobile Application: The phone-side secret K_p consists of a static and unique value called phone ID, which is 512-bits in length and denoted P_{id}. The mobile application also provides an entry table E containing N 256-bit random entry values and each of the entry is denoted as e, which is used to generate the Token.

$$K_p = (P_{id}, E), E = \{e_i\}$$

We choose N to be 5000 and an application side data example in shown in Table II.

TABLE II: Application Side Data

Data	Value
P_{id}	$0xff32241 \ldots$
e_1	$0x312ae44 \ldots$
e_2	$0x988ffe1 \ldots$
$\ldots$	$\ldots$
e_{4999}	$0x2034af4 \ldots$

e_i is the i_{th} entry value.

B. Core Functions

Now we dissect the core functions involved in both application registration and password generation processes, particularly the functions on generating password request R, generating token T, and generating the requested password P.

1) Registering mobile application with the Amnesia server.: After the user signs up with Amnesia server, it needs to download and register the Amnesia mobile application. Each application instance is uniquely identified with both P_{id} and registration id. A new P_{id} is generated each time the application is installed. Likewise, the registration id is assigned to the application by the rendezvous server. Upon registration, the application provides the server with both the registration

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 15,2024 at 20:41:19 UTC from IEEE Xplore. Restrictions apply.

id and P_{id}. For instance, we can simply make the user type in a generated CAPTCHA code on their application that is displayed on the Amnesia webpage. If the captcha codes match, the server will accept the application's P_{id} and registration id. Next, the server stores the registration id in plaintext and hash and salt the P_{id}. It is important that the server does not store the P_{id} in plaintext as the P_{id} is considered a phone-side secret.

2) Generating password request R: Assume that the user wishes to generate their password for account A. Account A is identified by (μ_A, d_A), which is signaled by the user to Amnesia server through standard HTTPS request. On the server, R is then generated as

$$R = H(\mu_A||d_A||\sigma_A)$$

σ_A is included as a preventative measure. Consider the following scenario, a passive attacker is eavesdropping on Amnesia server to rendezvous server communication. Without σ_A, the attacker could easily verify that the user is sending request R to their Amnesia application by computing $H(u_A||d_A)$ since u_A and d_A are predictable secrets.

3) Generating token T: Because R was generated with SHA-256, R is 64 hex-digit long. In order to generate T_E, we first split R into 16 segments of length 4. Each segment is denoted as s_i and $s_i = R[4i : 4i + 4]$. Note that the length of each segment should be able to cover the entry table size N, namely $16^l \geq N$. Each segment is modded against the entry table size N to obtain an index which is then used to get a particular e_i ($i = s_i\%N$). Next, the set of e_i will be concatenated and hashed with SHA-256. Thus, T_E can be expressed as

$$T_E = H(e_0||e_1||...||e_{15})$$

We express this process in Algorithm 1 for clearer presentation. Additionally, we observe that since $N = 5000$ and each request R yields 16 e_i, there are 5000^{16} or 1.53×10^{59} unique T_E. Finally, the token T is generated via hashing T_E along with R.

4) Generating requested password P: Password generation begins as soon as the token T arrives to the server. First, the server obtains the account's domain, username, and σ. These values are concatenated with T and hashed using SHA-512. Thus, the hashed intermediate value, which we designate as p, for account A can be expressed as

$$p_A = H(T_A||\mu_A||d_A||\sigma_A)$$

The intermediate value is then passed through a template function, which aims to map the intermediate value to a real password. The template function parallels the token generating function in the previous subsection. Here, the server contains a character table set. The size N_c of the character table set T_c is 94 and this includes lowercase letters, uppercase letters, numbers, and special characters. However, the character set on the table can be adjusted per account by the user to adapt to various website password policy. For instance, if an account does not allow the user to use special characters such as exclamation marks, the user can exclude them. Similar to generating a token T, p is then split into 32 4 hex-digit long segments g_i expressed as $g_i = p[4i : 4i + 4]$. Each segment is then modded against the table size and the value is used as an index to retrieve a particular character $c_i = T_c[g_i\%N_c]$. The retrieved characters are then concatenated to obtain P as

$$P = c_0||c_1||c_2 \ldots ||c_{31}$$

Thus, the maximum password length the user can generate is 32 characters. Of course, the user also has the option to limit the password length if situation demands it. In this case, the remaining characters that exceed the defined length are simply discarded. Note that Amnesia can support longer passwords. However, we believe a 32-character-long password is sufficiently secure.

Algorithm 1 generateT_E function

```
procedure GENERATE T_E(R)        ▷ Generates T)E from R
    c ← 0
    segmentList ← ∅
    entrySet ← ∅
    concatenated ← ∅
    while c + 4 ≤ R.length do
        segmentList.add(R.subString[c, c + 4])
        c ← c + 4
    end while
    for each segment in segmentList do
        index ← segment mod tableSize
        entryValue ← table[index]
        entrySet.add(entryValue)
    end for
    for each value in entrySet do
        concatenated.append(value)
    end for
    T_E ← SHA-256(concatenated)
    return T_E
end procedure
```

C. Recovery Protocols

Amnesia is a robust password manager that allows the user to recover when the user's phone is compromised/lost or when the user's master password is compromised.

1) Phone compromise recovery: Upon Amnesia application install and registration, the user is prompted to perform a one-time backup of K_p data onto a third-party cloud provider such as Google Drive or Dropbox. This process can be fully automated by utilizing the respective APIs. The backup process occurs between the mobile phone and the cloud provider. Specifically, this includes P_{id} and the entry table T_E. Even if the phone is compromised, the users are able to reset their passwords in order to recover Amnesia's two-factor security. We assume that both the third-party storage and the HTTPS connection between the phone and the storage are secure. The user will first log into the Amnesia server using MP. Next, the user will initiate phone recovery by uploading backup

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 15,2024 at 20:41:19 UTC from IEEE Xplore. Restrictions apply.

information from their cloud storage. The server then verifies the user by hashing the uploaded P_{id} value and matching it with the value stored in its database. After verification, the server will use the uploaded entry table and regenerate all of the user's passwords. The regenerated passwords can then be downloaded by the user for password recovery. The server then purges information related to the old phone such as the old hashed P_{id} value and registration id. From here, the user would then need to reinstall the Amnesia application on the new phone and re-register the application with the Amnesia server.

At this point, the user should have completely new passwords for each account since a new T_E is generated. Using the downloaded old passwords, the user should then access and reset the old passwords of each account to the newly generated ones. Note that while the attacker who compromised the phone now has access to K_p, the password is still safe due to lack of K_s. However, sole K_s may not be able to withhold a second attack as K_p is already lost in the first attack. Amnesia requires the user to reset their passwords to maintain 2 factor security. We believe that the user losing their phone is infrequent enough for this recovery to become too troublesome.

2) Master password compromise recovery: Compared to the phone recovery protocol, master password recovery is painless and easy. The user would first log into the server using their compromised master password. The user then initiates the process to change the master password. The server will display a text to the user. Next, the user types the text on their phone and sends P_{id} and text from the phone to the server for verification. The P_{id} is hashed and matched with the hashed P_{id} stored on the server with the text used to locate the user's account. If the verification passes, the user is allowed to change the master password. Note, however, that there is no protocol available if the user forgets their master password.

IV. Security Analysis

Although Amnesia is similar to 2FA, they are still different. The key point is that Amnesia uses 2FA to generate passwords. However, the websites themselves are unaltered, which means they still rely on a single-password (generated by Amnesia) for authentication. Although Amnesia provides strong protection for the password generation process, unlike 2FA, Amnesia does not provide client-side security. For instance, a keylogger is able to capture the passwords retreived by the client. Nevertheless, Amnesia is highly compatible and deployable since it does not require any modification on existing websites.

There are five potential attack vectors against the Amnesia architecture, including three connections and two components. The connections include the HTTPS between the user's computer and the Amnesia server, the rendezvous server routing from the Amnesia server to the user's application, and the HTTPS connection from the application to the Amnesia server. On the other hand, the attacker can also attack the server or compromise the user's phone. We do not include an attack on the user's computer since the consequences are similar to the case when the attacker compromises the HTTPS connection between the computer and the server.

A. Broken HTTPS

HTTPS is used to protect communication between the user's computer and Amnesia server. Likewise, HTTPS is also used for protecting phone to server communication. Assuming that the attacker is somehow able to compromise the connection and snoop the traffic, the severity of the consequences would depend on where the connection is. If the attacker is able to eavesdrop on the communication from the phone to the server, he would be able to retrieve T. However, having T alone is useless since the attacker does not have enough information to use it. The case where the attacker compromises the HTTPS connection between the user's computer and Amnesia server would represent a far greater threat. In this situation, the attacker can eavesdrop on password P that the victim has generated for some account. Over a period of time, it is possible for the attacker to collect a large set of P from the user.

B. Rendezvous Server Eavesdropping

The rendezvous server is used to transmit request R from the Amnesia server to the user's Amnesia application on their smartphone. Assuming that the attacker is able to eavesdrop on rendezvous messages, he will be able to obtain R. However, $R = H(u_A||d_A||\sigma_A)$ for some account A. As briefly mentioned in section-III, the σ_A helps prevent the attacker from deriving any information from R. Without σ_A, the attacker can simply compute $H(u_A||d_A)$ in order to verify the fact that the user is making a request for account A. Since σ_A is a 256-bit value, it is very unlikely for the attacker to make use of this information by itself.

C. Server Breach

The Amnesia server stores the K_s of each Amnesia account. As detailed in section III, K_s is comprised of d, u, and σ for each account that the user manages through Amnesia. Furthermore, the server also stores hashed and salted P_{id} along with the Amnesia application's registration id. Recall that $p_A = H(T_A||\mu_A||d_A||\sigma_A)$ for some account A. Here, the attacker who has compromised the server would have access to μ_A, d_A, and σ_A. However, he is still missing T_A if he wished to steal the password for account A. Since T_A is a 256-bit value, it is very unlikely for the attacker to derive anything password related from K_s alone. Furthermore, current password cracking techniques such as brute-force and dictionary attacks will not work against Amnesia's generative method. In order to derive the 256-bit T_A value, the attacker would need to brute-force 2^{256} possible combinations. Assuming only 50 percent needs to be exhausted to yield the correct T_A value, this still results in 2^{255} combinations. Additionally, the attacker would not have any feedback whether a particular guessed T_A value is correct since the real password P_A would not resemble anything created by a human due to its generative origins.

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 15,2024 at 20:41:19 UTC from IEEE Xplore. Restrictions apply.

However, a server breach will nevertheless still afford the attacker information. Because Amnesia server stores u and d for each account, the attacker would know the accounts and usernames that the victims are managing under Amnesia. Additionally, it is conceivable that the attacker may leverage the registration id to their advantage. For instance, the attacker may abscond with the victim's K_s and then send a request R from his own malicious server using the victim's registration id. Although it would appear suspicious to the victim that a request R came in despite the victim never requesting anything, nevertheless the possibility is there that a naive user may simply press accept and give away their password.

D. Phone Compromise

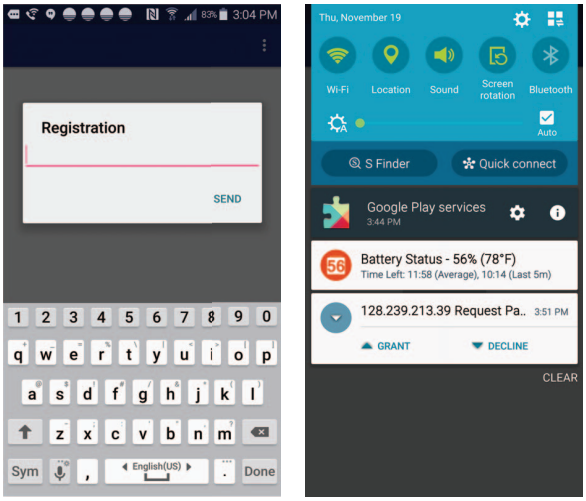
The Amnesia application stores K_p which includes the P_{id} and entry table. Additionally, the server's certificate is also stored here. As with the threat model, we assume that the user's smartphone is an insecure environment. This suggests that an attacker who has compromised the user's smartphone not only has the ability to view K_p but also peek into application memory. Therefore, the attacker would be able to view the computation of T from R which we recall is $T = H(T_E, R)$. Despite this advantage however, there are two key information deprived from the attacker. Specifically, the attacker does not know K_s and the attacker does not know which account R is for. As stated before, the reason the attacker does not know details regarding R is due to the inclusion of σ. Therefore, unless the attacker either compromises the user's master password or breach the Amnesia server, he will be unable to derive anything useful from the user through their phone.

E. Generated Password Strength

By default, Amnesia will generate a 32 character length password. The users, however, have the option to curtail this length if they wish as well as modify the character set. However, assuming that the hashed intermediate value p is random and the character set and its length are set to default, the average generated password would comprise of roughly 9 lowercase characters, 9 uppercase characters, 3 numerals, and 11 special characters. Additionally, the password space is 94^{32} or 1.38×10^{63} possible combinations. Since Amnesia's passwords are generated, attackers are unable to employ dictionary-based attacks. Furthermore, attackers would lack any confirmation whether the password they cracked is legitimate due to its generative nature.

V. Implementation

The Amnesia system requires two major components, specifically the Amnesia server and the Amnesia smartphone application. The Amnesia web server is implemented on Amazon's EC2 platform while the Amnesia application is developed on Android. Additionally, we use Google's GCM [9] as our rendezvous server to forward messages from the Amnsia server to the smartphone.



(a) Registering application (b) Receiving password request

Fig. 2: Amnesia application screenshots

A. Amnesia Server

We build a prototype of Amnesia server use Cherrypy [10], a lightweight python-based web framework. Additionally, we generated a self-signed HTTPS certificate to protect traffic between the user's smartphone as well as the user's computer. Because the prototype is at most used for latency tests (section VI) and our user study (section VII), we only allocated a maximum of 10 threads in our Cherrypy thread-pool.

The server is comprised of three components, namely, a component that manages and handles user interaction and sessions, a cryptography component, and a database handler. Information such as K_s, hashed and salted master password, registration id, etc. are stored in a SQLite database. Our cryptography component is implemented using PyCrypto [12].

B. Amnesia Mobile Application

The Amnesia application is implemented with Java using Android Studio targeted at SDK version 23 with a minimum SDK version of 16. If the user has not registered, the mobile phone will be prompted a screen as shown in Figure 2(a) for phone registration. The application is comprised of three components, specifically a GCM service listener, cryptography service, and database handler. The GCM listener is responsible for handling password requests from the Amnesia server. Once a password request arrives, the GCM service listener will notify the user via Android's notification action. Additionally, the GCM data bundle includes information such as the IP address of the originating request. This information is captured on the server and bundled together along with R for transit. We capture the password request in Figure 2(b). After verification, the cryptography service is responsible for generating T based on R. Cryptography on Android, specifically the hashing operation, is implemented using *java.security* package. Similar to the Amnesia server, we store K_p data using a SQLite database. Furthermore, the application also stores the server's

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 15,2024 at 20:41:19 UTC from IEEE Xplore. Restrictions apply.

TABLE III: Amnesia Comparative Evaluation

Scheme	Usability								Deployability						Security										
	Memorywise-Effortless	Scalable-for-Users	Nothing-to-Carry	Physically-Effortless	Easy-to-Learn	Efficient-to-Use	Infrequent-Errors	Easy-Recovery-from-Loss	Accessible	Negligible-Cost-per-User	Server-Compatible	Browser-Compatible	Mature	Non-Proprietary	Resilient-to-Physical-Observation	Resilient-to-Targeted-Impersonation	Resilient-to-Throttled-Guessing	Resilient-to-Unthrottled-Guessing	Resilient-to-Internal-Observation	Resilient-to-Leaks-from-Other-Verifiers	Resilient-to-Phishing	Resilient-to-Theft	No-Trusted-Third-Party	Requiring-Explicit-Consent	Unlinkable
Password			●		●	●	❍	●	●	●	●	●	●	●		❍						●	●	●	●
Firefox (MP)	❍	●	❍	❍	●	●	●		●	●	●		●	●	❍	❍					●	●	●	●	●
LastPass	❍	●	❍	❍	●	●	●	❍	●	❍	●		●		❍	❍	❍	❍		❍	●	●		●	●
Tapas	●	●	❍	❍	●	❍	●		❍	●	●			●	●	❍					●	●	●	●	●
Amnesia	❍	●	❍	❍	●	❍	●		●	●	●	●		●		●	●	●		●	●	●	●		●

● indicates that password manager fully fulfills the property and ❍ indicates that the password manager semi-fulfills the property.

self-signed certificate for HTTPS communication. We tested the Amnesia application on Samsung Galaxy Note 4 running Android Lollipop version 5.1.1.

VI. Evaluation

We conducted a comparative evaluation of Amnesia with other password managers using the framework developed by Bonneau et al. [11]. Moreover, we measured Amnesia's latency during a password generation process. This measurement is performed on both wifi and 4G connections.

A. Comparative Evaluation Framework

Bonneau's framework is a comparative platform to gauge three aspects of various authentication schemes, specifically usability, deployability, and security. Bonneau argues that the list has been refined and categorized into these three areas and thus best highlights the multifaceted nature of modern authentication. We compare Amnesia with other related authentication schemes, which are shown in Table III. We compared Amnesia with Tapas, which is similar to Amnesia because it also separates information necessary for password retrieval between two mediums, in this case the desktop and smartphone. For the other comparisons, we chose examples that we think best exemplify the most common forms of password managers: built-in browser password managers (Firefox) and cloud-based password managers (LastPass).

In regards to other password managers, we see that Amnesia does comparatively well in both security and deployability. Specifically, except for the mature property, Amnesia fulfills all deployability requirements. The reason for this is because Amnesia, currently as a prototype, implements account management and password generation all within its web server. Thus, the user is not required to download a client application on their computer, such as a browser plugin, in order to use Amnesia. However, Amnesia does require the user to download and install the Amnesia application on their smartphone. As of right now, the Amnesia application is only available for Android OS, although it would not take much effort to port it to other platforms such as iOS.

In terms of usability, we see that Amnesia lags a bit behind other password managers. Because Amnesia is a bilateral password manager, the user would need to carry around their smartphone in order to use Amnesia. Therefore, unlike password managers such as LastPass or Firefox's built-in password manager, the user would need to interact with the smartphone in order to authorize password related transactions. Similarly, Tapas [13] is also a bilateral password manager. Thus, we see similar scores between Amnesia and Tapas in the usability section.

Security-wise, Amnesia performs comparatively well. Currently, the Amnesia prototype is not resistant to physical observations. This is because the generated password is displayed to the user in text form. However, this issue can be solved with the implementation of an auto-filler. Additionally, Amnesia is not resilient to internal observation. This property requires a given scheme to be resistant to impersonation in the event an attacker is able to eavesdrop on internal information. While Amnesia communications are done securely through HTTPS and Google's GCM service, if an attacker were to break through, as mentioned in section IV, he can easily intercept generated passwords. While impersonation would be difficult, since the attacker would need to breach either the server or compromise the phone in order to forge legitimate R and T, we still consider this property to be unfulfilled.

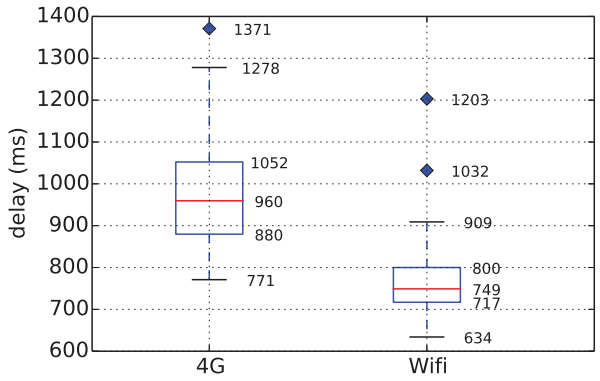
B. Latency Evaluation

A major usability concern with Amnesia is the latency associated with generating a password. Recall that password generation involves three steps: the server sending request R to the user's application through Google GCM service, the computation of T from R and K_p, and finally the transmission of T back to the server along with the final password computation using K_s and T. Thus, we wish to measure the total time it takes to generate a single password.

We used our existing prototype for the experiment. We created several modifications. First, we added timestamp t_{start} to the message sent to the application. t_{start} is the time that

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 15,2024 at 20:41:19 UTC from IEEE Xplore. Restrictions apply.

Fig. 3: Amnesia Latency



R is being sent from the server to Google GCM. Next, we removed the user verification notification from the application and instead made the phone automatically compute T from any request R. After the computation of T is finished, the phone will include t_{start} in its message along with T and send it to the server. Once the server receives the message from the application, it will compute password P. After this operation, it will take the current timestamp t_{end}. The latency is simply computed by the following: $latency = t_{end} - t_{start}$.

We conducted this test over both Wifi and 4G using Samsung Galaxy Note running 5.1.1 Lollipop. The Wifi service provider is Cox Communications with a download speed of 30 mbps and an upload speed of 10 mbps. Additionally, the 4G provider is T-Mobile. Both tests were conducted in a suburban environment. Both Wifi and 4G were tested 100 times.

The latency results under the two settings are shown in Figure 3. The mean $\bar{x}$ for the Wifi test is 785.3 ms while the standard deviation σ is 171.5 ms. $\bar{x}$ for 4G test is 978.7 ms and σ is 137.9 ms. The sample size for each is 100 test trials. Hence, under our testing conditions, we see that Wifi has less latency than 4G. With an average delay of 785.3 ms for Wifi and 978.7 ms for 4G, we can conclude that latency is not a big issue.

VII. User Study

While we tried to make Amnesia as secure as possible, one of the main reasons for password managers is to ease the burden of memorizing multiple strong passwords for the user. Therefore, we needed to design Amnesia in a way that is both reasonably easy and convenient to use as well as being secure. With this in mind, we had two main purposes for our user study. Our first goal for the user study was to determine their current password behaviors. Factors such as password length, entropy level, and password uniqueness are all major contributors to password security. Amnesia's generative properties satisfies these requirements. However, we wanted to compare the security of Amnesia's generated passwords with the users' current passwords. As such, the first portion of the survey will deal with how secure the users' current passwords are. Our second goal for the user study was to determine the usability of Amnesia. As password managers are meant to relieve user stress, it is important for them to be user friendly. If a password manager becomes too cumbersome to user, the user might slip into poor password habits such as reusing passwords rather than continue using the password manager. As such, the second portion of the survey will deal with how convenient users find the Amnesia application. Our user study cannot provide conclusive evidence on Amnesia's usability due to its small scale. Sources of potential bias may result from the study's size, crowd-sourcing platform, and the novelty effect. Instead, this is a pilot-study meant to evaluate the feasibility of Amnesia's two-factor mechanism in everyday use. The study allowed us to obtain immediate feedback on any deficiencies in the user experience which we can improve upon.

A. Setup

To start testing Amnesia, we created a dummy site so users can practice adding accounts to Amnesia. While the dummy site did emulate a lot of functionality of a real website, we did not wish for users to be creating throwaway accounts on real sites for the purposes of this testing. We also created a short video that explains to the user how to use the Amnesia application. Our video provides instructions for registering on the Amnesia website, downloading the Amnesia application, adding and managing accounts on Amnesia, and generating a password from Amnesia. Once we set up the video and the dummy site, we asked for participants from Amazon Mechanical Turk. We requested that the users possess their own Android phones in order to run our application. When the users were finished testing out Amnesia with the dummy website, they were asked to fill out a short post-test questionnaire about their previous password habits and their experiences using Amnesia. The survey itself was online for seven days and users were allocated 2 hours and 30 minutes to complete the testing.

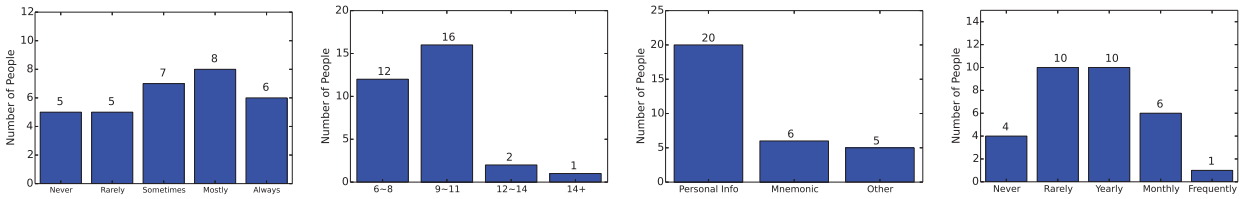
The users are required to perform a number of tasks that are briefly summarized as follows:

1) Create an Amnesia account
2) Download and register the Android application
3) Create an account on Amnesia for the dummy website.
4) Generate a password for the dummy website.
5) Create an account on the dummy website using the generated password.
6) Post a comment on the dummy website containing the generated password.

B. User Demographics

Because Amnesia is designed for a wide audience, we aimed for a more diverse population for our user study. We used the accidental sampling method to obtain a more holistic representation of our audience as opposed to a snowball sampling method which would limit our testing population to those from the college campus. In total, we recruited 31 participants for the user study. 21 of the participants were male and the participants' age ranged between 20 and 61 years old ($\bar{x}$=33.32, σ=9.92). The participants came from a wide variety

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 15,2024 at 20:41:19 UTC from IEEE Xplore. Restrictions apply.



(a) Password Reuse (b) Password Length (c) Password Creation Techniques (d) Password Change Frequency

Fig. 4: Survey Results

of backgrounds including computer science, homemaking, business, medicine, engineering, management and real estate. Additionally, we also asked the users how many hours they spend per day on the internet. 13 out of 31 participants spend 4 to 8 hours a day online while 4 participants spend 1 to 4 hours. 8 participants spend 8 to 12 hour and 6 participants spend over 12 hours online everyday.

C. User Security Habits

The first portion of our survey was the user security analysis. In this part, we asked the users 5 specific questions about their online behavior, the results for which are illustrated in Figure 4. To get some context for the users password habits, we first asked about how many online accounts the user has. 17 (54.8%) of the participants had 10 unique online accounts or less while the remaining 14 (45.2%) had between 11 and 20 unique online accounts. The rest of the questions in this portion dealt different aspects of coming up with and managing passwords. The data for the user's password creation methods is shown in Figure 8. As the graph illustrates, the majority of users have short, personal information based passwords that they reuse for most of their accounts. By using Amnesia, most people (27 out of 31) believe that they would be increasing the security of their passwords.

D. Amnesia Usability

The second portion of our survey was the usability testing. To make the analysis easier, we split Amnesia into 3 distinct parts: creating an account with Amnesia, adding an account to Amnesia, and generating a password with Amnesia. While user response to all three portions were favorable, only 77.4% (24 out of 31) of users found registering with Amnesia to be convenient, compared to the 83.8% (26 out of 31) that found it easy to add an account and to generate a password. The loss of convenience is acceptable because the user only has to register with Amnesia once.

E. Amnesia Preference

Finally, we asked users to reflect on both their own password habits and their experience using Amnesia. We then asked users whether they prefer Amnesia over their current password manager or management method. We found that 70.9% (22 of of 31) of users would use Amnesia over what they are currently using. Specifically, out of 24 subjects who do not utilize password managers, 14 prefer Amnesia over their current methods. Out of the 7 subjects who do use password managers, 6 prefer Amnesia over their current password managers.

From the user feedback, most users felt that Amnesia was very secure. However, the most common complaint was the non-intuitive UI. This aspect in Amnesia can be improved with the inclusion of a browser plugin that has auto-fill capabilities. Currently, the Amnesia prototype was not designed with a focus on usability. However, we plan to publish a fully fledged Amnesia system in the future.

VIII. Limitations

Amnesia has several limitations. First, Amnesia does not provide a protocol for the user in the event that the user forgets their master password. Furthermore, Amnesia's bilateral property may negatively impact its usability factor in comparison to other password managers. Access to the user's accounts thus becomes dependent on the availability of their mobile phone. If the smartphone is powered off or offline, then the user would lose access to their accounts. Additionally, Amnesia's architecture forces the server to compute a hash in order to generate the final password, which may be a bottleneck to the system's performance. As of the current version, users can pick password properties such as password length and the presence of special and/or numerical characters. However, they are unable to store specific chosen passwords. Users would also need to interact with the phone each time they request a password from Amnesia. We plan to address these two issues in the future by including a vault and a session mechanism in a fully fledged Amnesia system.

IX. Related Works

A. Password Analysis

Human-chosen passwords have always been weak and predictable. Morris and Thompson [14] first showed that passwords are vulnerable to dictionary attacks. Additionally, time-memory trade-off [2] and rainbow table [15] further accelerate dictionary attacks. Modern password cracking techniques include using Markov Model [4] and probabilistic context-free grammars [3], or even password semantic information [16]. There have been numerous works focusing on studying the characteristics of passwords [17]–[20], which conclude that passwords are highly predictable.

Besides the inherent predictability and low entropy of human-chosen passwords, users do not always follow the most secure practices, such as frequently changing passwords and not reusing old passwords [1], [6], [21].

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 15,2024 at 20:41:19 UTC from IEEE Xplore. Restrictions apply.

B. Password Manager

Securely managing passwords becomes a challenging endeavour when users acquire an increasing number of accounts. Since replacing text-based passwords does not seem possible in the near future [11], password managers become an important tool for both security and convenience. Password managers have developed a number secure and advanced features. For example, PwdHash [22] hash several seeds to produce strong passwords. Kamouflage [23] proposes to use decoy passwords to defend brute-force attacks on password managers. Chatterjee et al [24] breaks Kamouflage [23] and leverages honey encryption [25] to develop a cracking-resistant password manager. Despite these various features, there are always vulnerabilities or limitations [26]. For instance, Silver et al. [27] investigated the filling policies of password managers and identified their insecure filling practices. Moreover, Chiasson et al. [28] stress that the wrong password manager mental model can significantly hurt the security of user passwords.

C. Phone-assisted authentication

Smartphones are used as a second factor in many authentication schemes. Examples include Google 2-step verification [29], Phoolproof [30], PhoneAuth [31], etc. However, a severe security drawback of password managers is the single point of failure associated with the master password. Specifically, losing the master password usually means losing every passwords in the password manager. To mitigate such a problem, Tapas [13] isolates the encrypted password wallet in a mobile phone and the decryption key in a computer to ensure that the user must have both the computer and the mobile phone to retrieve any password.

X. Conclusion

We present Amnesia, a bilateral generative password manager that provides strong security while maintaining reasonable usability. Amnesia is a new secure architecture for password managers that avoids the single point of failure associated with congregating sensitive information in one location. Amnesia's bilateral property makes the system immune to critical attacks such as data breaches which retrieval password managers are vulnerable to. Additionally, we created a strong password generation method which provides the users with extremely high entropy passwords without any associated burden. By using Amnesia, users can be effectively ignorant of their account passwords and are only responsible for remembering their master password. Furthermore, users have effective recovery options when their master password or phone is compromised.

Acknowledgment

We thank Dr. Georgios Portokalidis as our shepherd and other anonymous reviewers for their efforts on shaping this paper. We also thank Mr. Zikuan Li for his help in conducting experiments and user study. This work is supported by U.S. Office of Naval Research under grants N00014-15-1-2396 and N00014-15-1-2012.

References

[1] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, "Passwords and the evolution of imperfect authentication," *Communications of the ACM*, 2015.
[2] M. E. Hellman, "A cryptanalytic time-memory trade-off," *IEEE Transactions on Information Theory*, 1980.
[3] A. Narayanan and V. Shmatikov, "Fast dictionary attacks on passwords using time-space tradeoff," in *ACM CCS*, 2005.
[4] M. Weir, S. Aggarwal, B. De Medeiros, and B. Glodek, "Password cracking using probabilistic context-free grammars," in *IEEE Security & Privacy*, 2009.
[5] A. Beautement, M. A. Sasse, and M. Wonham, "The compliance budget: managing security behaviour in organisations," in *NSPW*. ACM, 2008.
[6] D. Florencio and C. Herley, "A large-scale study of web password habits," in *ACM WWW*, 2007.
[7] http://www.csoonline.com/article/2936105/data-breach/lastpass-suffers-data-breach again.html, "Lastpass suffers data breach again."
[8] M. Billemont, "Master password – secure your life, forget your passwords," http://masterpasswordapp.com.
[9] "Google developers, cloud messaging," 2012, https://developers.google.com/cloud-messaging/.
[10] "Cherrypy – a minimalist python web framework," 2015, http://www.cherrypy.org.
[11] J. Bonneau, C. Herley, P. C. Van Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in *IEEE Security & Privacy*, 2012.
[12] "Pycrypto – the python cryptography toolkit," 2015, https://www.dlitz.net/software/pycrypto.
[13] D. McCarney, D. Barrera, J. Clark, S. Chiasson, and P. C. van Oorschot, "Tapas: design, implementation, and usability evaluation of a password manager," in *ACSAC*. ACM, 2012.
[14] R. Morris and K. Thompson, "Password security: A case history," *Communications of the ACM*, 1979.
[15] P. Oechslin, "Making a faster cryptanalytic time-memory trade-off," in *Advances in Cryptology-CRYPTO 2003*, 2003.
[16] R. Veras, C. Collins, and J. Thorpe, "On the semantic patterns of passwords and their security impact," in *NDSS*, 2014.
[17] J. Yan, A. Blackwell, R. Anderson, and A. Grant, "Password memorability and security: Empirical results," *IEEE Security & Privacy Magazine*, 2004.
[18] R. Veras, J. Thorpe, and C. Collins, "Visualizing semantics in passwords: The role of dates," in *IEEE VizSec*, 2012.
[19] J. Bonneau, S. Preibusch, and R. Anderson, "A birthday present every eleven wallets? the security of customer-chosen banking pins," in *Financial Cryptography and Data Security*. Springer, 2012.
[20] Z. Li, W. Han, and W. Xu, "A large-scale empirical analysis of chinese web passwords," in *Proc. USENIX Security*, 2014.
[21] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang, "The tangled web of password reuse," in *NDSS*, 2014.
[22] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell, "Stronger password authentication using browser extensions." in *Usenix security*, 2005.
[23] H. Bojinov, E. Bursztein, X. Boyen, and D. Boneh, "Kamouflage: Loss-resistant password management," in *ESORICS*, 2010.
[24] R. Chatterjee, J. Bonneau, A. Juels, and T. Ristenpart, "Cracking-resistant password vaults using natural language encoders," in *IEEE Security & Privacy*, 2015.
[25] A. Juels and T. Ristenpart, "Honey encryption: Beyond the brute-force barrier," in *EUROCRYPT*, 2014.
[26] Z. Li, W. He, D. Akhawe, and D. Song, "The emperor's new password manager: Security analysis of web-based password managers," in *USENIX Security*, 2014.
[27] D. Silver, S. Jana, E. Chen, C. Jackson, and D. Boneh, "Password managers: Attacks and defenses," in *Usenix Security*, 2014.
[28] S. Chiasson, P. C. van Oorschot, and R. Biddle, "A usability study and critique of two password managers." in *Usenix Security*, 2006.
[29] "Google 2-step verification." [Online]. Available: https://www.google.com/landing/2step/
[30] B. Parno, C. Kuo, and A. Perrig, *Phoolproof phishing prevention*. Springer, 2006.
[31] A. Czeskis, M. Dietz, T. Kohno, D. Wallach, and D. Balfanz, "Strengthening user authentication through opportunistic cryptographic identity assertions," in *ACM CCS*, 2012.

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on May 15,2024 at 20:41:19 UTC from IEEE Xplore. Restrictions apply.