Kryptose

Personnel: Jonathan Shi (js2845), Antonio Marcedone (am2623), Alexander Guziel (asg252), Jeff Tian (yt336)

System Purpose: It is often infeasible for users to memorize large sets of passwords that they use for their different accounts. This gets especially difficult due to a variety of factors, such as the fact that to optimize safety, passwords should be long and randomly chosen strings that are different for each of the user's accounts, and that each password should changed occasionally to maintain secrecy. With the goal of solving this problem, our vision for this project is to build a secure application whereby the user authenticates to a remote server using a single master password in order to retrieve a set of passwords.

On a high level, we will achieve our goal using the following mechanism. The client will establish a TLS connection and authenticate the server by verifying its certificate (which will be embedded in the client itself). The server will authenticate the client through a key which is derived from a master password chosen by the user at the time of registration (and possibly modified later). The server will never see the original master password (the key derivation happens on the client side).

After a successful authentication, the server will allow the user to modify or retrieve the encrypted passwords. Such passwords will be encrypted by another symmetric key derived from the same master password, and all the encryption and decryption operation happen on the client (thus protecting confidentiality against a possibly malicious or compromised server).

The user can then select an account to perform an action on from read, update, or delete or they can create a new account to store credentials on.

User Types:

1. Application User: This is the person using the service to create, read, update, and delete their passwords. They can log in and out and change their master password and (possibly) other ways of authentication.
2. Administrator: They can look at logs of all actions performed on the system by all users, as a form of audit. They also have the ability to delete user accounts, such as accounts which have been appropriately deemed inactive and useless or

those where the user has lost the master password and requests for their account to be deleted. However, they expressly do not have the ability to read the credentials stored by the users of the system.

Assets:

1. Handle (for associating people with accounts)
2. Master passwords (for authenticating people with their accounts)
3. Stored credentials (domain, username, password)
4. User logs
    a. Credential retrieval times and device info
    b. Credential modification times and device info
    c. Connection and decryption and integrity-checking errors
    d. List of recognized devices (possible feature)
5. User account (includes all of the above)
6. Admin logs
    a. Information on each connection made, including remote address
    b. Connection and decryption and integrity-checking errors
    c. Event logs of each action requested by a client
    d. Unexpected exceptions and errors

System Backlog -- To do:

User type	Assets	Imp	User story
App user	User account - Handle; User account - Master password	M	User can create an account to allow them to use the service, associating a handle and a user master password to themselves
App user	User account - Master password	S	Authenticated user can change their master password, for security purposes and for personal preference
App user	User account - Master password	M	User can authenticate themselves using a master password, to gain access to the other services
App user	User account - Master password	W	User can employ some alternative methods of authentication (eg biometric, image-based)
App user	User account - Stored	W	User can have the system

	credential		automatically generate part or all of an attack-resistant random set of credentials
App user	User account - Stored credential	C	User may access a cached (and possibly out-of-date) copy of their own credentials on their own local devices, in case of lack of server availability
App user	User account	C	Users can set how long it takes for their authentication to expire
App user	User account	M	Users who are authenticated can end their authentication session
App User	User account - Stored credential	M	User can delete a previously stored login credential
App User	User account - Stored credential	C	User can see when a login credential was last changed
App User	User account - Stored credential	S	User can copy their credentials into their OS clipboard for convenient input into other forms
App User	User account - Stored credential	W	User can indicate a UI form to be automatically populated with their stored credentials
App User	User account	C	User can completely delete the account from the server, except for the administrators' logs related to the account.
App User	User account - Log	C	User can see when the previous access to the system has occurred, as well as the source IP address and device info of that access.
App User	User account	W	User can use a second device as a second form of authentication
App user	User account	W	User can tell the system to require two-factor authentication for particular actions, for increased

			security
App user	User logs - Recognized devices	W	User may choose to enable two-factor authentication for registration of a new recognized device.
App user	User logs - Recognized devices	W	User may choose to deregister existing recognized devices.

System Backlog -- Completed:

User type	Assets	Imp	User story
App user	User account - Master password	M	User can authenticate themselves using a master password, to gain access to the other services
App user	User account - Stored credential	M	Authenticated user can associate a new set of login credentials to their account, for later retrieval
App User	User account - Stored credential	M	User can change a previously stored login credential
App User	User account - Stored credential	C	User can view the plaintext of their stored login credentials
Admin User	Logs	M	Admin can view logs for troubleshooting and attack detection and recovery
Admin User	User account	W	Admin can completely delete any user's account from the server, except for the administrators' logs related to the account.

Threat Analysis:

We consider several categories of threats. We categorize these threats based on their discriminancy in their targets (discriminate threats seek to compromise a particular user, indiscriminate threats do not have particular target users in mind), and based on their

capabilities (they may have read/write access to communications, external web content, or the server itself).

Discriminate threats are interested in compromising the assets of a particular user. They may be trying to spy on, blackmail, or publicly shame the user. They may also be attempting to steal assets that are protected by the credentials stored by the user. The assets they might be interested in include:

1. Accessing stored credentials for a specific account
2. Knowing if a particular user has a registered account in the system
3. Denying a user access to their credentials, whether by modifying or deleting them, or impeding availability to the server itself
4. Viewing activity logs for the user
5. Modifying or deleting the logs of the system, to prevent discovery of their attack

Indiscriminate threats do not particularly care which user's assets they obtain. They may be after financial gain, or may simply be in it "for the lulz". Specific assets of interest to them include:

1. Accessing stored credentials, including information identifying what external assets may be secured by those credentials
2. Denying users access to their credentials, whether by modifying or deleting them, or impeding availability to the server itself
3. Viewing activity logs for the users
4. Modifying or deleting the logs of the system, to prevent discovery of their attacks

Communications threats we model as Dolev Yao adversaries, so that they have complete control over the communications channel between the client and the server. These threats may be discriminate or indiscriminate. We are unable to prevent availability outages caused by this type of attacker, but will attempt to mitigate the harms of such an attack.

Web content threats are threats who choose to serve content to client devices, whether via web page or email, without directly attacking the client-server communications of our system. Attacks done by web content threats may include phishing, cross-site scripting, and some limited reading of device state and properties (in particular, some older web browsers such as IE6 enable such attackers unrestricted access to system clipboards). Threats using these capabilities may be looking to take advantage of common user errors. While many of these forms of attack are outside of our control and not in our scope to prevent, we may still take action to mitigate or impede these attacks when our system is involved.

Server access threats are threats who have compromised some or all parts of the server device. These may include administrative users. It is impossible to prevent availability attacks performed by these threats, but confidentiality and integrity of the assets may be protected by appropriate cryptographic "sandboxing". We may also take action to mitigate the harms of an availability attack.

We would like to minimize the amount of trust placed in users with administrative privileges, due to the possible presence of server access threats. Administrative users do, however, have complete read access to the log files, so they are able to see all of the users' actions within the system, ie the times when they log on and the times when they retrieve or change their credentials. This is a type of confidentiality that the user should not expect from our system, however we deem this an appropriate tradeoff as the administrator does not gain any essential passwords from knowing these things but gaining the ability to perform audit and the increased safety it provides outweighs this loss.

Non-threats: Someone who gains physical access or otherwise compromises a client's device in order to gain access to that client's credentials is considered a non-threat under our model. Once compromised, little can be done. For example, if the memory of their computer can be read it is impossible to protect the password since it must be in memory at some point.

Security Goals:

1. The system shall prevent unauthorized principals from accessing the application user's credentials. (Confidentiality)
2. The system shall prevent unauthorized principals from modifying user credentials. (Integrity)
3. The system shall prevent disclosure of any master passwords to any principal. (Confidentiality)
4. The system shall prevent principals other than the server administrator from modifying the logs. (Integrity)
5. The system shall prevent other principals from modifying the user's master password or from deleting the accounts. (Integrity)
6. The system shall prevent unauthorized principals from viewing the logs. (Confidentiality)
7. The system shall never store unencrypted user credentials or master passwords in any part of the server-side's memory at any point in time during the system's operation.

Essential Security Elements:

1. Authentication: In order for any user to obtain access to their stored passwords, the user first has to authenticate using their username and master password. This prevents people other than the user from accessing the user's stored

credentials, while allowing the user themselves to access their credentials. This is obviously important because the point of passwords is that they are secret keys to allow the user to access a set of resources exclusively, and you don't want these secret keys to be obtainable by outside parties.

2. Authorization: Only the user who created an account (whose identity is established through authentication) is authorized to read or write their own stored credentials. Administrative Users are expressly not allowed to see credentials stored in any of the user's accounts, though they are authorized to delete entire accounts if need be. The App User should be able to read (parts of) its own logs, but not to modify them. Administrative Users are allowed to see the logs (but we cannot prevent them from modifying them).
3. Audit: We keep a log and record actions without exposing information that should be kept confidential. This can be useful to recognize and recover from an attack (identifying the vulnerability and possibly the origin of the attack).
4. Confidentiality: Stored credentials should be kept confidential. Compromise of such credentials defeats the whole purpose of using a password manager, as the whole purpose of passwords is that they are secret keys to some resource owned by the owner that should not be accessible by any other unauthorized principal.
5. Integrity: Integrity of the credentials is very important, as unauthorized modification could make the stored credentials useless.