Trust Issues Chat

Thomas Lindblom

System Basics

- Desktop application written with Python 3
- NGINX server running Node.js
- Server running on AWS EC2 instance
- SQLite for the database
- URL for the server is https://donttrustthisgroup.me/

System Components - Authentication

- Users authenticate with email and password

- Users that provide a valid email and password receive a JSON web token from the server

System Components - RSA key exchange

- Users exchange public keys in an out-of-band fashion (using email)
- User sends a friend request to another user and emails their public key separately to the other user
- If the other user accepts, then the system mails their public key to the original sender

3 Main Security Goals

1. Achieve message confidentiality

2. Achieve message integrity

3. Correctly authenticate users

Assets

- Email addresses
- Passwords
- Messages
- AES/HMAC keys
- EC2 instance itself
- Connection between client and server

Analysis - Confidentiality - SSL

- Using SSL prevents anyone from acquiring meaningful data that is being transmitted from client to server

- Any adversary would need the SSL RSA private key

Analysis - Confidentiality - Database data

- Since SSL is secure if an adversary wants any of the data they have to get it from the database directly
- The SQLite database is in persistent storage on the AWS instance!
- Cannot get to database data unless you can compromise the AWS instance

Analysis - Confidentiality - Database data

- AWS instance uses VPC (logically isolated from other cloud networks)
- Adversary would have to SSH in or grab database file via SCP
- Both require the SSH private key which is on my local machine

Analysis - Confidentiality - Database data

- If the adversary was able to get the database file however…

- Passwords, Messages, and AES/HMAC keys all have additional layers of security!

Analysis - Confidentiality

- For the outsider we can reasonably assume that the database data will remain confidential

- If we consider the server to be an insider adversary things change

Analysis - Confidentiality

- There’s nothing stopping the server from directly reading email addresses
- The server can also read passwords because they come in as plaintext
- All other data is still confidential because it is encrypted

Analysis - Integrity

- HMAC ensures that the adversary cannot produce a valid message, tag combo for my HMAC key without knowing the key

- Since we can ensure the confidentiality of the HMAC key, the adversary will not know the key which ensures message integrity

Analysis - Authentication

- The system provides JSON web tokens to any user that supplies valid credentials

- Since we can ensure password confidentiality with respect to outsiders, we can assume that no outsider will be able to provide valid credentials and act as another user to obtain a token

Analysis - Authentication

- Since we CANNOT ensure password confidentiality with respect to the insider, the insider could pretend to be a user and get tokens

- The insider still would not have the RSA keys of the user they are impersonating to encrypt and decrypt messages (this is where the out-of-band communication of RSA keys comes in handy!)