Week 3 Blog

Sree Ram Boyapati	a1775690

What have I done this week?

Learnings:
I have tried to learn more about the tools that I would use to setup server-side infrastructure. To provide more information on how our infrastructure would like, I plan to define the infrastructure components using Terraform[1] and I am trying to setup an encrypted remote backend state for development by all three of us seamlessly.

For the server, I have learned more about google app engine and chosen to go with a standard environment. How we store secret keys is something I am still trying to figure out. We can use environment variables to read the secret keys in application runtime. App Engine provides a standard runtime to use just like Heroku which is a platform-as-a-service provider. Since I have previously used Flask, I have spent time learning how flask would work in the context of the app engine runtime.

I have also studied how logging might work in the context of the app engine and Google Cloud Logging[2] has provided a compliant solution. Based on this I tried to develop a logging configuration for production and testing purposes.

I have detailed the three papers I have studied to know more about

1. How TANs can be securely transported instead of SMS due to privacy/security issues.
2. What information can HIPAA compliant logging should contain
3. How can I prevent relay/replay attacks>

Outcomes:
On the first day, I started working on creating the infrastructure using Terraform in google cloud https://github.cs.adelaide.edu.au/2020-Mobile-and-Wireless-Systems/CovidGuard-F/tree/master/iaac

I have paused it and started working on the verification servers based setup as my teammates started working on the android application in the earliest. As per my earlier design, I am planning to use google app engine.
https://github.cs.adelaide.edu.au/2020-Mobile-and-Wireless-Systems/CovidGuard-F/tree/master/server

The server will use Flask framework and uwsgi for application runtime. I am setting up an entrypoint.sh to run my custom setup which I have used many times in my personal projects. Using

[1] "Terraform." https://www.terraform.io/. Accessed 17 Aug. 2020.
[2] "Writing and viewing logs - Google Cloud." https://cloud.google.com/appengine/docs/standard/python3/writing-application-logs. Accessed 17 Aug. 2020.

ngrok.com I shall expose my development setup to support remote testing in case there are any restrictions due to COVID-19 for real-time debugging. I am presently defining Exposure Notification Server and Verification Server in a single repo and most probably use different route paths in runtime.

Flask framework is also used to run Cloud Functions which are serverless endpoints. I have created the boilerplate and shall work on it after the server is done.

Most Importantly, In the week, I have tried to solve two major problems which I want to add on to original GAEN architecture as novelty -

1. Asymmetric TAN generation using 2FA account token.
   When we register the GUID and generate a registration token, 2FA account token is also generated. The user will add the account token to their Authenticator app. When uploading keys or giving consent, short-lived tokens generated by it can be used as validation of consent
2. Preventing Relay/Replay attacks using different methods
   In Krzysztof et al. 2020, Location information is also used to generate the tag of RPI and this information is not given to the contact user and contact user generates the same tag using least significant bits of time and location data. If they are near during the same time for a minute, they will have the same values. Before android 11, Location data can be accessed by the bluetooth permissions (may not be approved), since we are not sharing the location and only accessing it, possibly we can use it to prevent relay attacks. If not location data, can we use any other data to ascertain that both users were at the same time and location?

In the next few weeks, I plan to write code in all three folders (server, iaac and functions) to test the functionality for android.

I plan to develop eventing names for compliance purposes and audit and developed filters to use to efficient handling of audit logs - https://github.cs.adelaide.edu.au/2020-Mobile-and-Wireless-Systems/CovidGuard-F/blob/master/server/covid/logger.py

Based on tags, I plan to process audit logs to check to store them in a different datastore or use some other features if we have time like intrusion detection.

Paper Review # 1

W. A. Khan, Y. Saleem, G. A. Shah and A. Farooq, "Modified Mobile Transaction Authentication Number System for 2-layer security," 2016 International Conference on Intelligent Systems Engineering (ICISE), Islamabad, 2016, pp. 89-93, doi: 10.1109/INTELSE.2016.7475167.

Challenges and Why did I choose this paper?

To obtain consent from the user, that they authorize the upload to the servers and exposure notification server checks the teleTAN generated with the verification server and checks if it is correct.

Some of the important things that need to be taken care -

1. Can this TANs (Transaction Authentication Number) be generated asymmetrically - Verification server can verify that this TAN is confirmation or generated by this particular user?
2. If symmetric, How can the communicating medium be made secure?

Outcome of the Research

1. SMS based TAN approaches are not secure. They are also a huge privacy issue as the identity of the diagnosed user may be compromised.
2. SMS based approaches are more efficient as there is no QR based setup and are easy to setup.
3. SMS delivers mTAN in an unencrypted fashion and it needs network connection to mobile.

QR Code based authenticators is a better approach however, there are few challenges

1. How can we set up a 2FA based approach which offers encryption?
2. Short Lived tokens which are generated on the mobile instead of the server

The research helped us understand how 2FA based on an application is more secure than SMS-Based 2FA or the token which is generated on the fly by the server.

How is the outcome of the research relevant to the project?

To gather consent of the user who decides to share the RPI (Rolling Proximity Identifiers) with the user, It is important that the consent is provided and recorded by the data controller (i.e us).

If the verification server generates the TAN and the ENS server verifies the TAN with the verification server. There are three hops here and the token has to be short-lived. What if we need to upload keys directly without TAN after confirming as positive as the user may stay positive for some time. Instead of 2 API calls, The app can use 2-factor based codes and send a token each time when it sends a payload.

Asymmetric encryption here is more secure as the token for authorization is generated at the mobile rather than the server.

Paper Review # 2

Liu, Brent J. et al. “A HIPAA-Compliant Architecture for Securing Clinical Images.” Journal of Digital Imaging 19 (2005): 172-180.

Challenges

Verification servers may have to integrate with the patient record system of the state or city. Any service interacting with the Patient record system has to be HIPAA compliant. How do we secure our verification server without breaking HIPAA compliance. What are the norms that we should adhere to secure the patient’s health status without exposing it to third parties?

Outcome of the Research

The research provided some data on how the project needs to be structured on the server side. To make a HIPAA compliant architecture, Records of when, where and access status, access type needs to be written to a log for compliance.

To make this a painful effort in large-scale software development, The authors have divided the architecture into several layers -

1. Action Layer
2. Notification Layer
3. Audit Layer
4. Record Layer

Each log of the audit stores the following information

1. Time of the event
2. Patient whose records are accessed.
3. The record of the patient that was accessed.

The logs are all written in syslog format and stored in a ACID-Compliant Database. AuditLogs are then extracted using batch processes to check for intruder detection.

How is this useful for the project

The paper has given us overview on how to go about in setting up intrusion detection and process audit logs.

Using python logging framework, We need to store the following details -

1. Authentication identifier - server version, registration token, TAN
2. Where the data is accessed - EVENT NAME hard-coded based on the function.
3. Patient id - registration token and guid
4. Record Id - data from records server

We set the identifier hipaa-audit in the tags for LOG.Based on the tag, Stream Handlers can add it to a cloud Pub/Sub topic which can be consumed by the intrusion detection system and audit logging databases.

Paper Review # 3

Pietrzak, Krzysztof. “Delayed Authentication: Preventing Replay and Relay Attacks in Private Contact Tracing.” IACR Cryptol. ePrint Arch. 2020 (2020): 418

Challenges

In our presentation, We highlighted how the verification process can be hijacked by replay attacks. The attacker will collect RPIs from a location when two bystander and broadcast them in another location. This will lead to a lot of false positives.

Relay attacks are when in real-time an attacker takes the epiIds and broadcasts them over the internet. Entire companies can shut down due to such attacks.

Replay attacks can be prevented by time-stamping the RPI generated - however, how do we prevent relay attacks as the delay can be very minimal? In ZKP-based approach, both the devices share a proof of contact with each other. In the context of GAEN, how will this proof of attack be generated?

This paper proposes an extendable approach to many protocols splitting the verification of a valid contact in two phases.

Outcome of the Research

In the proposed approach, In the first step, only the message is sent and in the second step, the key is sent. In the case of replay attacks, if the RPIs do not match the keys by which they were generated, the RPI collected is invalid preventing a replay attack or by timestamping the RPI.

The approach over here is very simple - In the relay attack even if the timing of RPI is similar, location will not be the same.

There are two properties -

1. Given message m and random variable p, commitFunction(m, p) != commitFunction(m’, p’)
2. It should be hard to find p’

Instead of generating a tag from Mac(k, challenge) where challenge is given to prove proof of contact. Delayed Mac is generated using timestamp and location coordinates. The other app will generate the same mac using the time and location coordinates of when it received the tag.

Signature and Signature2 are only similar if the delay is minimal and location difference is smaller.
When the secret key using which ephemeral keys are generated is published. tag of the RPI and signature will act as proof of contact for the RPIs.

Since it is hard to find p’ for a given commit message, a message which contains location can not be decrypted. As per evidence, for 10 million users and 10000 cases, p’ has probability of one false positive.

How can we use this in our project?
As part of GAEN protocol, Location permission is already taken and the coarse location of the user can be obtained. Location over here is not being broadcasted rather an hash of it.

However, if the location permission is removed for exposure notification API after android 11 upgrade. Alternative information that two people belong to the same place needs to be obtained. (cell tower signal)? However, the protocol holds good if we can replace GPS coordinates with other suitable information.