Security Architecture

Google Goals

- Protect user / customer data
- Provide availability
- Accountability
    - Recovery
    - Trust

Threats / Attacks

- Theft of servers / other physical attacks
- DoS
- Network attacks (both over Internet and internal)
- Insider attacks
- Bugs in their own code or third party libraries
- Malicious services / servers

Google DS Model

- DS has multiple servers, each of which run multiple VMs
- Each VM runs 1 or multiple services on it, these communicate through encrypted HTTPS / TLS
- Servers have to register with GFE to communicate on Google infrastructure
    - All internal services have to go through this as a reverse proxy
    - All RPC requests have to go through here
- Isolation:
    - VMs, gVisor, SFI (software fault isolation via wasm / NaCl)

RPC over ATLS

- Receives requests from some service or engineer
- Authorization is done by some central policy (so people can just look to see who is allowed to talk to them / who they are allowed to talk to)
- All requests are audited / logged
    - Google provides access transparency: you can see every RPC made

End-User-Tickets

- When you make an HTTPS request, your cookie is attached
- It goes through GFE and then arrives at Gmail
- This is sent to the end-user authentication service, which converts the cookie to a ticket
- This ticket is attached to all future RPCs to show that this RPC is made on behalf of the end user (you)
- End-user authentication service:
    - Gives tickets that are short-lived so they cannot be reused for long

Network Security

- Some cluster controller has to give certificates to machines running different services if they want to make an RPC to each other
    - That way they have the proper certificates to actually be able to encrypt / authenticate with one another
    - I.e. a certificate that verifies this is a valid instance of Gmail or Contacts running

- How does this cluster controller know who to give the certificate to? How does it know this machine is running the proper Gmail
    - Trusted hardware = Titan chip which monitors the boot process to make sure everything is the correct code
    - When a machine boots up, it generates a new public key and secret key
    - After this happens, it now wants to be added to the cluster, so it contacts the cluster
        - It uses the Titan chip's secret key to sign a bunch of data about hashes about the code that booted up, including the public key it wants to use
        - It sends this signed bit with the public key of the server to the cluster controller
        - Cluster controller has a list of Titan chip keys that can be used to decrypt this and ensure an actual Titan chip was used to encrypt this and that everything matches up
        - After this, we know that this is good, we store the public key, and we use that to now send all information to that machine
- But now we know the hardware is trusted, what about software?
    - Everything goes through the cluster controller
    - The cluster controller downloads the code, and analyzes policies to make sure the code is correct
    - It then sends this to the machine, which creates a new VM to run the code
    - Code must be approved by an engineer other than the author

DoS Prevention

- Authenticate as soon as possible to get rid of
- Overprovision