# MIT 6.858 - Computer Systems Security "Lecture 2 - Security Architecture"

## Lecture 2 - Security Architecture

Security architecture is structuring entire systems in order to defend against large classes of attacks, prevent as-yet unknown attacks, and limit damage from successful attacks. This lecture analyzes a paper written by Google describing their security architecture for Google Cloud Platform. It gives a good overview of the techniques and principles Google uses to implement their security architecture.

You can read the paper here. You should read the paper before going through these notes. The paper describes the architecture Google uses, these notes discuss *why* Google uses that architecture and why it is good.

Google has the following broad security goals - 1. Protect customer data 2. Ensure availability of Google services 3. Track down what went wrong if something does go wrong 4. Limit the damage an attacker can do in case of a successful attack

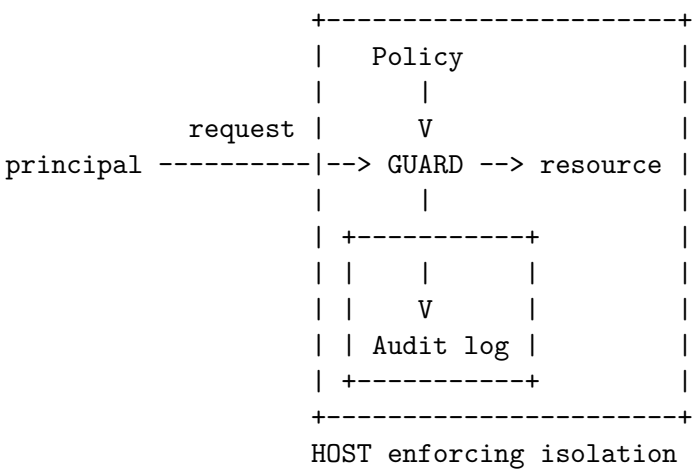
Google has the following hardware/software services in its architecture - 1. Data centers 2. Physical machines in those data centers (servers) 3. Virtual machines in those servers 4. Services running on those virtual machines 5. Applications running on those services 6. Remote Procedure Calls (RPCs) between applications and services 7. Requests coming in through the internet producing RPCs

### Isolation

A fundamental principle used by Google's security architecture is isolation. Each service and/or application is isolated from each other (sometimes virtually, sometimes physically), such that by default no service can access the data and/or resources belonging to any other service. This means that even if one service is compromised or has bugs, it is not able to damage any other service. Without isolation, we cannot hope to build secure systems.

Now, we don't want every service to be 100% isolated all the time. We want services to be able to talk to each other, without which we cannot build a distributed system.

We need to allow inter-service communication in a controlled way. In order to control communication between services, a common pattern is the guard pattern. A guard is placed in the front of every service, who's job is to monitor incoming requests from other services and only let them in if the service is allowed to talk to it.




In the diagram above, a principal can be a user, a device, a service, etc. Resources can be services, like Gmail or Drive, or files and user data.

The guard performs three functions -

1. Authenticate - Identifies who is issuing the request.
2. Authorize - Determines whether that request should be allowed.
3. Audit - Records each request along with its authentication and authorization information, and other useful metadata.