ONAP Security
- AAF Security Strategy
- Proposed Security Deliverables for Casablanca

ONS Breakout Session

March 26, 2018, 4pm, PDT

Jonathan Gathman, AT&T, Architect of AAF

Amy Zwarico, AT&T

Stephen Terrill, Ericsson

ONAP
OPEN NETWORK AUTOMATION PLATFORM

AAF Security Strategy

ONS Breakout Session

March 26, 2018, 4pm, PDT

Jonathan Gathman, AT&T, Architect of AAF

Amy Zwarico, AT&T

What will you get out of this talk?

- What is AAF?
- How do I use AAF to secure my App?
- When/how can I start?

ONAP
OPEN NETWORK AUTOMATION PLATFORM

What is AAF?

What is AAF?

- AAF stands for “Application Auth Framework”
  - Originally “Auth” was “Authorization”, but now supports implementations for
    - Authentication
    - Authorization
- AAF consists of
  - CADI Framework - a library used by services to:
    - Authenticate with one ***or more*** Authentication Protocols (more on that in a bit)
    - Authorize in a FINE-GRAINED manner using AAF Components
  - AAF Components – RESTful Services:
    - Service (primary) – All the Authorization information (more on that in a bit)
    - Locate – how to find ***ANY OR ALL*** AAF instances across any geographic distribution
    - OAuth 2.0 – new component providing Tokens and Introspection (no time to discuss here)
    - GUI – Tool to view and manage Authorization Information, and create Credentials
    - Certman – Certificate Manger, create and renew X509 with Fine-Grained Identity
    - FS – File Server to provide access to distributable elements (like well known certs)
    - Hello - Test your client access (certs, OAuth 2.0, etc)
  - Cassandra as global replicating Data Store

Brief (very Brief) History of AAF

AAF was started as the Client CADI Framework

Goal: Allow Developers to CORRECTLY use required Security:

1. Without Special Coding
2. More than one Protocol supported ***SIMULTANEOUSLY***
   1. Example: Basic Auth, SSO Cookies, 2 - way TLS
   2. This also allows NEW Protocol Plugins as needed
      1. New Security packages (i.e. OAuth 2.0)
      2. Organization Specific Protocols (i.e. custom built SSO)

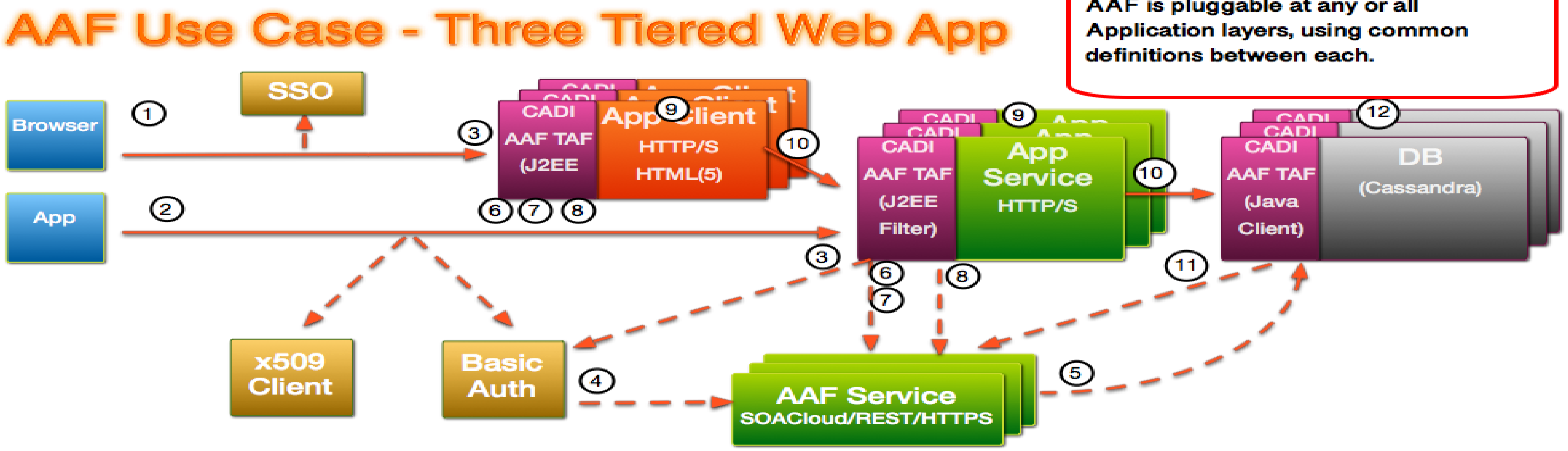
Authorization for Fine-Grained (AAF) modeled from an AT&T Deployment Tool

Certificate Manager was created to support Fine-Grained x509 Authentication (2016)

OAuth 2.0 supported added (2017)

Full AAF Suite brought to ONAP , “Beijing” (March 2018)

Example of AAF Elements in Action


AAF Use Case - Three Tiered Web App
AAF is pluggable at any or all Application layers, using common definitions between each.
SSO
Browser
App
CADI AAF TAF (J2EE
App Client HTTP/S HTML(5)
CADI AAF TAF (J2EE Filter)
App Service HTTP/S
CADI AAF TAF (Java Client)
DB (Cassandra)
x509 Client
Basic Auth
AAF Service SOACloud/REST/HTTPS
1
2
3
4
5
6
7
8
9
10
11
12

1. Browser client would go to GUI, using, for instance, SSO Plugin (or Basic Auth)
2. App goes directly to a Service, using x509 or BasicAuth (or other)
3. CADI Filter converts credential to "Principal". If not in cache, AAF is contacted for Permissions protecting GUI with Service ID/ Credential (MechID of App/Pass or X.509 Client Cert (preferred))
4. AAF does provide User/Password features, or can be delegated to other credential service via Plugin
5. If info is not in Service Cache, AAF's DB is contacted using AAF Service ID/ Credential
6. Client App uses Permission Attributes delivered by AAF/AAF Cache for protecting sensitive data/functions (using J2EE method)
7. If not in Cache, Client contacts App Service, using App ID/Credential.
8. CADI Filter converts App ID/Credential to Principal. If not in cache, contacts AAF (with App ID/Cred) for Permissions of Client
9. App protects data based on Client Permissions
10. Component contacts next layer using Service ID/Credential.
11. If ID or Permissions of AppServer not in Cache, contact AAF using AAF Security Plugin for Cassandra, which uses AAF Java Client
12. Cassandra protects Cluster/Keyspace/ColumnFamily w/Permissions

What is AAF? – Review

AAF is a set of Client Libraries (CADI Framework) and RESTful Services that support multiple Authentication Protocols and Fine-Grained Authorization

OPEN NETWORK AUTOMATION PLATFORM

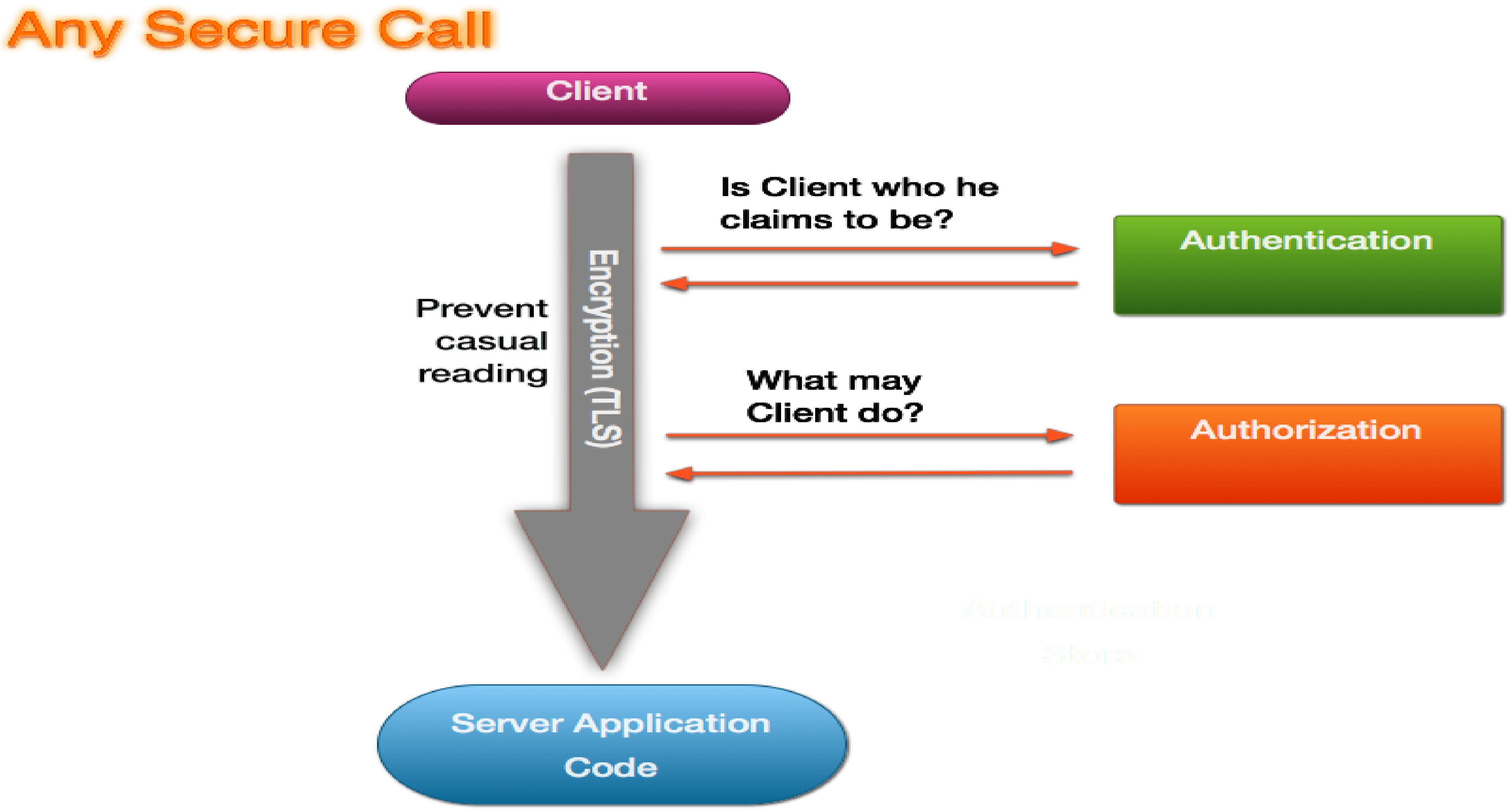
Why should I use AAF to secure my App?

The Big ”Why”s

- Security layer’s done. You can focus on YOUR app
- Create common Authorization method across MicroService model
  - The smaller the service element, the less it makes sense to create your own Authorization scheme.
  - AAF provides Organizational meaning to individual components
- Create common Suites of Tools out of MicroServices
  - AAF separates Role from Permission
  - A Role, then becomes a Suite of Permissions from potentially large suite of other tools
    - Example: If AAF was applied to ONAP, becoming a committer would give you
      - Tie in LinuxFoundation Identity Service
      - Committer rights in Garret for your App
      - Cassandra rights for your App (sign in with your LinuxFoundation ID, or use Certificate)
      - Jenkins rights for your App
      - Appropriate Deployment rights to common DEVL
      - Any MicroServices that are developed specifically for LinuxFoundation

How do I use AAF to secure my App?

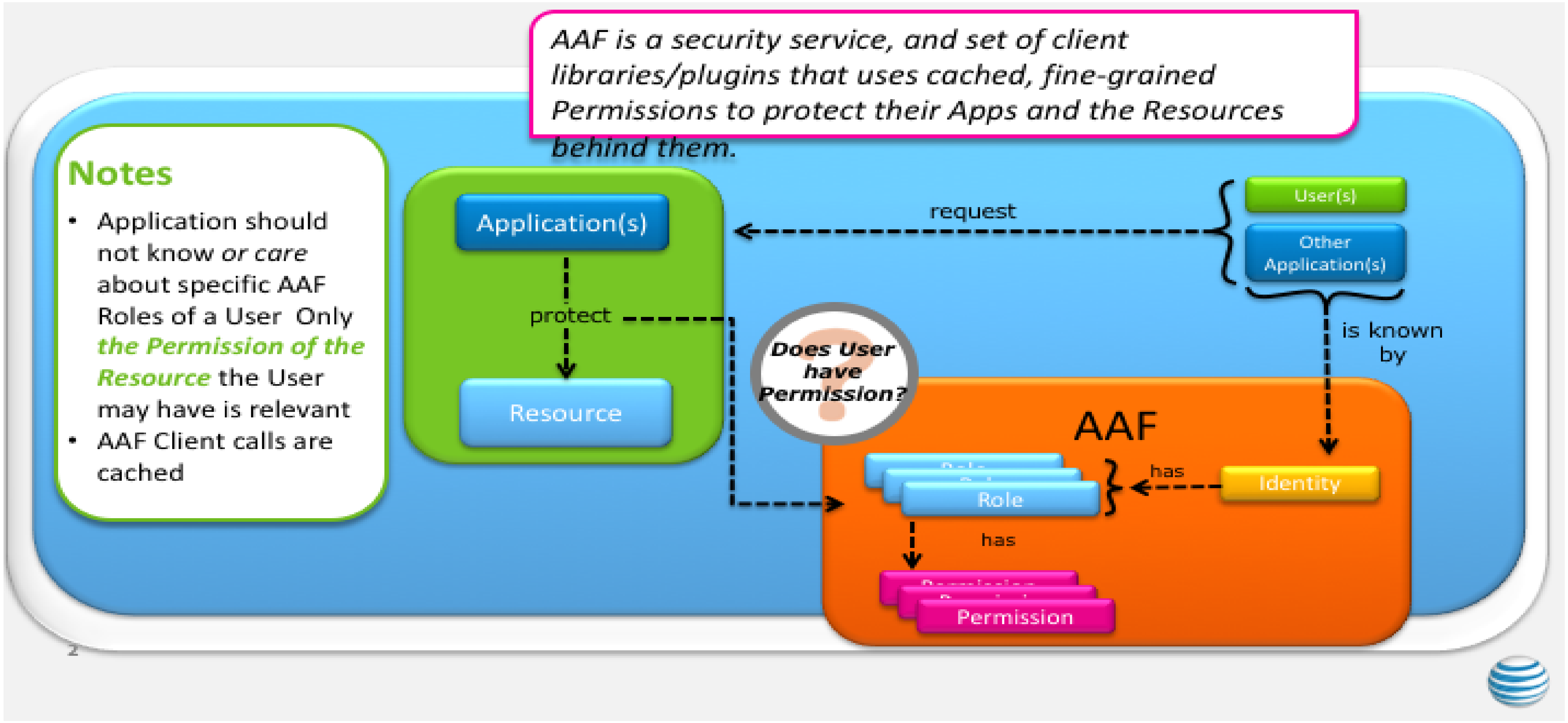
Any Secure Call needs …



Ahead of time - Developers

- Self-Serve AAF Functions for Developers
  - Applications get a “Namespace” in AAF
    - Example “org.onap.dmaap”
  - Create Credentials for their App
    - x509 Client Certificate or
    - User/Password (Basic Auth)
  - Create a Server Certificate (so service can be HTTP/S TLS)
    - Can use the x509 Client Certificate, assuming all clients trust its Certificate Authority
  - Create “Permissions” representing what they want to protect
  - Code to those Permissions

Real time Authorization Process



AAF is a security service, and set of client libraries/plugins that uses cached, fine-grained Permissions to protect their Apps and the Resources behind them.

Notes

- Application should not know *or care* about specific AAF Roles of a User Only ***the Permission of the Resource*** the User may have is relevant
- AAF Client calls are cached

Relationship of AAF Permission to Policy

CADI's goal is to evaluate every transaction for Authentication and Authorization.

Speed is paramount

There is one "Policy" for Authorization:

"Has this Authenticated User been Granted this Permission?"

AAF Permissions are like…

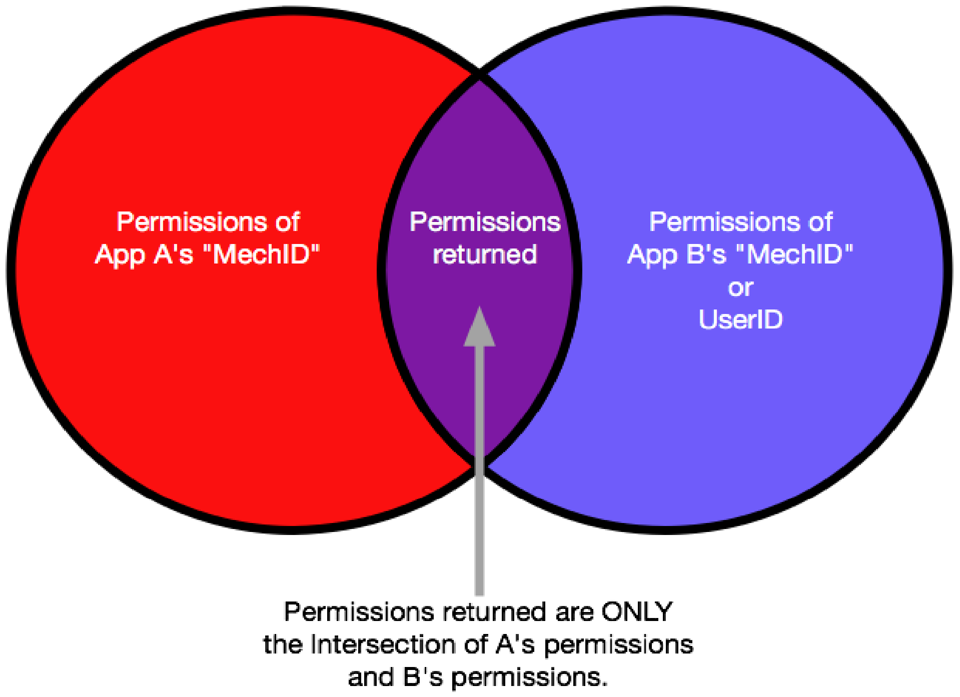
Thus,

An x509 Certificate represents a previous action of Authenticating an End Client by trusting a Certificate Authority.

An AAF Permission represents a previous action of Authorizing a particular Identity to access a Resource.

What Permissions is Returne about a User to the Service?

- Service only gets RELEVANT Permissions, meaning those that it has access to.



- The Service is NOT told what other Permissions a User might have that are beyond its scope.

But how do I code?

Coding is done with CADI Libraries, and there are many was to use the Libraries.

1. J2EE (Servlets) – use the provided "CADIFilter"
   - Filter Authenticates, and if valid, passes on to Servlet Code
   - Surround your code with "if (httpServletRequest.isUserInRole(String)) { … }"
     - The meaning is overloaded. Think "doesThisUserHaveThisPermission("<AAFPermission>")"
2. Java Client Library
   - CADI provides "AAFCon", "AAFAuthn" and "AAFAuthz" classes for direct Java Access
   - This is how we code Plugins
3. Pre-written Plugins – use Java Client above
   - Cassandra
   - Shiro (new for Beijing)
   - Others as needed. This IS Open Source Community!
4. Direct AAF RESTful API
   - This should only happen with CAREFUL attention to Caching your responses in your chosen language.

ONAP
OPEN NETWORK AUTOMATION PLATFORM

When can I start? How can I start?

AAF Availability

- Thanks to the Linux Foundation, AAF is available from gerrit repo
  - Use Maven to compile from ”osaaf”
    - https://gerrit.onap.org/r/#/admin/projects/aaf/authz
  - Stand alone Scripts are provided
  - Docker Builds are provided
    - Use with Docker “Cassandra” for quick “out of the build” kick the tires
- ONAP Devl Environment (https://wiki.onap.org/display/DW/Physical+Labs)
  - AAF “Amsterdam” is currently running: RESTful API version is unchanged.
  - After M4, AAF “Beijing” will be added (on different ports)
  - Build/maintain your own. This is, after all, Open Source!
- CADI Libraries will be available in ”Maven”
- We’ll be working on additional Documentation after M4

ONAP

OPEN NETWORK AUTOMATION PLATFORM

Thank you!

ONAP
OPEN NETWORK AUTOMATION PLATFORM

Proposed Security Deliverables for Casablanca

March 26, 2018

Fundamentals of Open Source Security

	Community Practices	Code Security	Runtime Security
Design	• Badging practices (e.g., CII Badging) • Technical steering committee	• Threat modeling and analysis	• Patching & versioning strategy
Build	• Secure coding practices • Secure, auditable and versioned repositories	• Automated security testing: SAST, SCA, DAST • Code management: coverage, reviews & integrity checks	• Automated security testing: IAST • Manual Penetration testing
Deploy	• Vulnerability management • Integration with external security platforms/tools	• Security defect resolution	• Security orchestration • Runtime application self-protection (RASP)

Casablanca Potential Deliverables

Providing recommendations and project guidance

	Community Practices	Code Security	Runtime Security
Design			
Build			
Deploy			

Casablanca Potential Deliverables

Providing recommendations and project guidance

	Community Practices	Code Security	Runtime Security
Design	• Establish vulnerability remediation guidelines		
Build			
Deploy			

Casablanca Potential Deliverables

Providing recommendations and project guidance

	Community Practices	Code Security	Runtime Security
Design	• Establish vulnerability remediation guidelines • Document X.509 certificate practices		
Build			
Deploy			

Casablanca Potential Deliverables

Providing recommendations and project guidance

	Community Practices	Code Security	Runtime Security
Design	• Establish vulnerability remediation guidelines • Document X.509 certificate practices		
Build			
Deploy	• Develop key threat analytics		

Casablanca Potential Deliverables

Providing recommendations and project guidance

	Community Practices	Code Security	Runtime Security
Design	• Establish vulnerability remediation guidelines • Document X.509 certificate practices		
Build		• Enhance static code scanning integration/automation	
Deploy	• Develop key threat analytics		

Casablanca Potential Deliverables

Providing recommendations and project guidance

	Community Practices	Code Security	Runtime Security
Design	• Establish vulnerability remediation guidelines • Document X.509 certificate practices		
Build		• Enhance static code scanning integration/automation • Recommended protocols and protocols to avoid	
Deploy	• Develop key threat analytics		

Casablanca Potential Deliverables

Providing recommendations and project guidance

	Community Practices	Code Security	Runtime Security
Design	• Establish vulnerability remediation guidelines • Document X.509 certificate practices		• Design pluggable authorization model
Build		• Enhance static code scanning integration/automation • Recommended protocols and protocols to avoid	
Deploy	• Develop key threat analytics		

Casablanca Potential Deliverables

Providing recommendations and project guidance

	Community Practices	Code Security	Runtime Security
Design	• Establish vulnerability remediation guidelines • Document X.509 certificate practices		• Design pluggable authorization model • Enhanced credentials management, OAuth support
Build		• Enhance static code scanning integration/automation • Recommended protocols and protocols to avoid	
Deploy	• Develop key threat analytics		

Casablanca Potential Deliverables

Providing recommendations and project guidance

	Community Practices	Code Security	Runtime Security
Design	• Establish vulnerability remediation guidelines • Document X.509 certificate practices		• Design pluggable authorization model • Enhanced credentials management, OAuth support
Build		• Enhance static code scanning integration/automation • Recommended protocols and protocols to avoid	• VNF package security:
Deploy	• Develop key threat analytics		

Casablanca Potential Deliverables

Providing recommendations and project guidance

	Community Practices	Code Security	Runtime Security
Design	• Establish vulnerability remediation guidelines • Document X.509 certificate practices		• Design pluggable authorization model • Enhanced credentials management, OAuth support
Build		• Enhance static code scanning integration/automation • Recommended protocols and protocols to avoid	• VNF package security: • Integrity verification at instantiation
Deploy	• Develop key threat analytics		

Casablanca Potential Deliverables

Providing recommendations and project guidance

	Community Practices	Code Security	Runtime Security
Design	• Establish vulnerability remediation guidelines • Document X.509 certificate practices		• Design pluggable authorization model • Enhanced credentials management, OAuth support
Build		• Enhance static code scanning integration/automation • Recommended protocols and protocols to avoid	• VNF package security: • Integrity verification at instantiation • Service provider artifact signing
Deploy	• Develop key threat analytics		

Casablanca Potential Deliverables

Providing recommendations and project guidance

	Community Practices	Code Security	Runtime Security
Design	• Establish vulnerability remediation guidelines • Document X.509 certificate practices		• Design pluggable authorization model • Enhanced credentials management, OAuth support
Build		• Enhance static code scanning integration/automation • Recommended protocols and protocols to avoid	• VNF package security: • Integrity verification at instantiation • Service provider artifact signing
Deploy	• Develop key threat analytics		• PNF use orchestrated by ONAP

Casablanca Potential Deliverables

Providing recommendations and project guidance

	Community Practices	Code Security	Runtime Security
Design	• Establish vulnerability remediation guidelines • Document X.509 certificate practices		• Design pluggable authorization model • Enhanced credentials management, OAuth support
Build		• Enhance static code scanning integration/automation • Recommended protocols and protocols to avoid	• VNF package security: • Integrity verification at instantiation • Service provider artifact signing
Deploy	• Develop key threat analytics		• PNF use orchestrated by ONAP

Your input please

ONAP

OPEN NETWORK AUTOMATION PLATFORM