Pluggable Security

Andrew Baxter, Ian Blackwell and Tian Lee

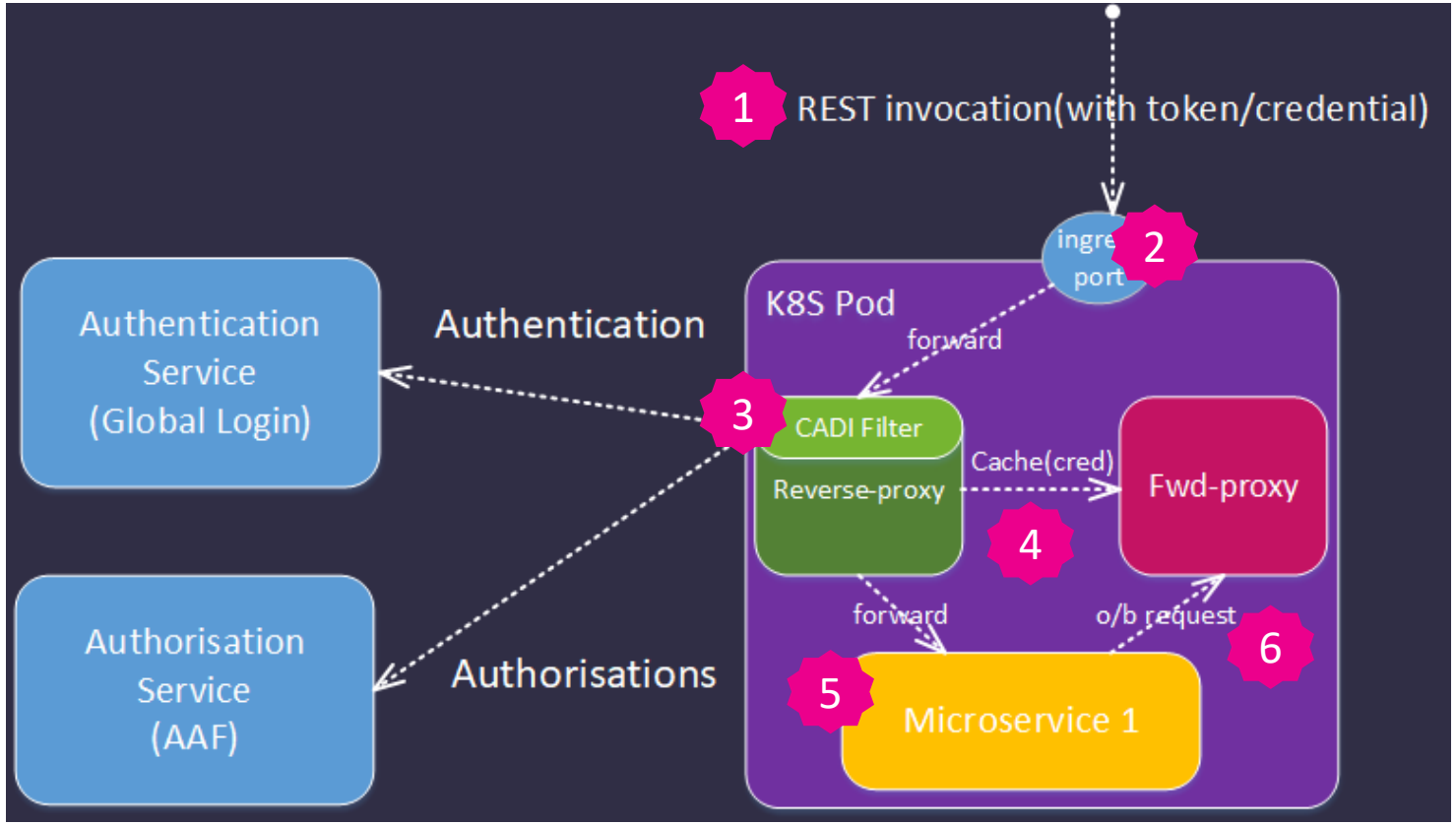
Agenda

- Pluggable Security: Business Drivers
- Pluggable Security: Overview
- Deploying pluggable security in ONAP and ECOMP
- Configuration
- Code Overview
- Failure Scenarios and Debugging
- Source Control
- Build and Deploy Jobs
- Documentation

Business Drivers

- Provides ECOMP the CADI/AAF authentication and authorisation security framework that it requires without requiring customisation of the underlying ONAP code.
- Provides ONAP with the pluggability it requires, such that alternative security providers can be integrated without requiring customisation of the underlying ONAP code.
- Provides the required isolation of the security infrastructure, localising the operational effort required for configuration and patching of microservices.
- Provides a language-independent solution (that supports microservices written in Clojure, Python etc as well as Java).
- Provides a foundation for fine grained access control.

Overview



1. An upstream user or application invokes a REST endpoint on ONAP microservice (MS1) supplying one or more tokens.
2. The request arrives at the K8s POD.
3. CADI authenticates the request and collects the authorisations for the requestor.
4. The reverse proxy stores the request tokens in the forward-proxy and ensures the requestor has the correct privileges to access the service endpoint.
5. The admitted request is forwarded to primary service.
6. If primary service makes o/b http requests these are directed to the forward proxy that appends the request tokens to the outgoing request.

ONAP and ECOMP Deployment

- Sidecar tested and OOM Helm charts committed for the following A&AI microservices in ONAP:
  - Enricher
  - TOSCA-Model XML transformer (Babel)
  - Broker
  - Gizmo
  - Champ
  - Search
  - Validation
  - Resources
- Sidecar relies on a Kubernetes environment
- Adoption of ECOMP Controller and availability of sample OOM Helm charts for a number of deployment scenarios1 provides an opportunity for ECOMP adoption of sidecar security and simplification of microservice security code.

See https://wiki.onap.org/display/DW/Pluggable+Security#PluggableSecurity-7.10Identifiedandsupportedpatternsandfeatures

Configuration – Overview

If configurable applied through OOM Helm charts

- application.properties – not configurable
- cadi.properties
- uri-authorization.json
- reverse-proxy.properties
- forward-proxy.properties
- primary-service.properties

Configuration – application.properties

```
CONFIG_HOME=./config
server.port=10692
server.ssl.enabled=true
server.ssl.protocol=TLS
server.ssl.enabled-protocols=TLSv1.2
server.ssl.key-store=${CONFIG_HOME}/auth/tomcat_keystore
server.ssl.client-auth=want
server.ssl.client-cert=${CONFIG_HOME}/auth/client-cert.p12
server.servlet.contextPath=/
uri.authorization.configuration-file=${CONFIG_HOME}/auth/uri-authorization.json
logging.config=${CONFIG_HOME}/logback-spring.xml
spring.profiles.default=secure,cadi
spring.main.allow-bean-definition-overriding=true
```

Configuration – primary-service & forward-proxy.properties

Primary Service

primary-service.protocol = https
primary-service.host = localhost
This needs to be configured to match the port of the primary service running in the pod
primary-service.port = 9000

Forward Proxy

forward-proxy.protocol = https
forward-proxy.host = localhost
forward-proxy.port = 10680
forward-proxy.cacheurl = /credential-cache

Configuration – cadi.properties

```
cadi_loglevel=DEBUG
cadi_keyfile=config/security/keyfile
cadi_truststore=config/auth/tomcat_keystore
cadi_truststore_password=OBF:1y0q1uvc1uum1uvg1pil1pjl1uuq1uvk1uuu1y10

# Configure AAF
aaf_url=https://aaf.osaaf.org:30247
aaf_env=DEV

aaf_id=demo@people.osaaf.org
aaf_password=enc:92w4px0y_rrm265LXLpw58QnNPgDXykyA1YTrflbAKz

# This is a colon separated list of client cert issuers
cadi_x509_issuers=CN=ONAP, OU=ONAP, O=ONAP, L=Ottawa, ST=Ontario, C=CA

cadi_latitude=80.62
cadi_longitude=72.62
```

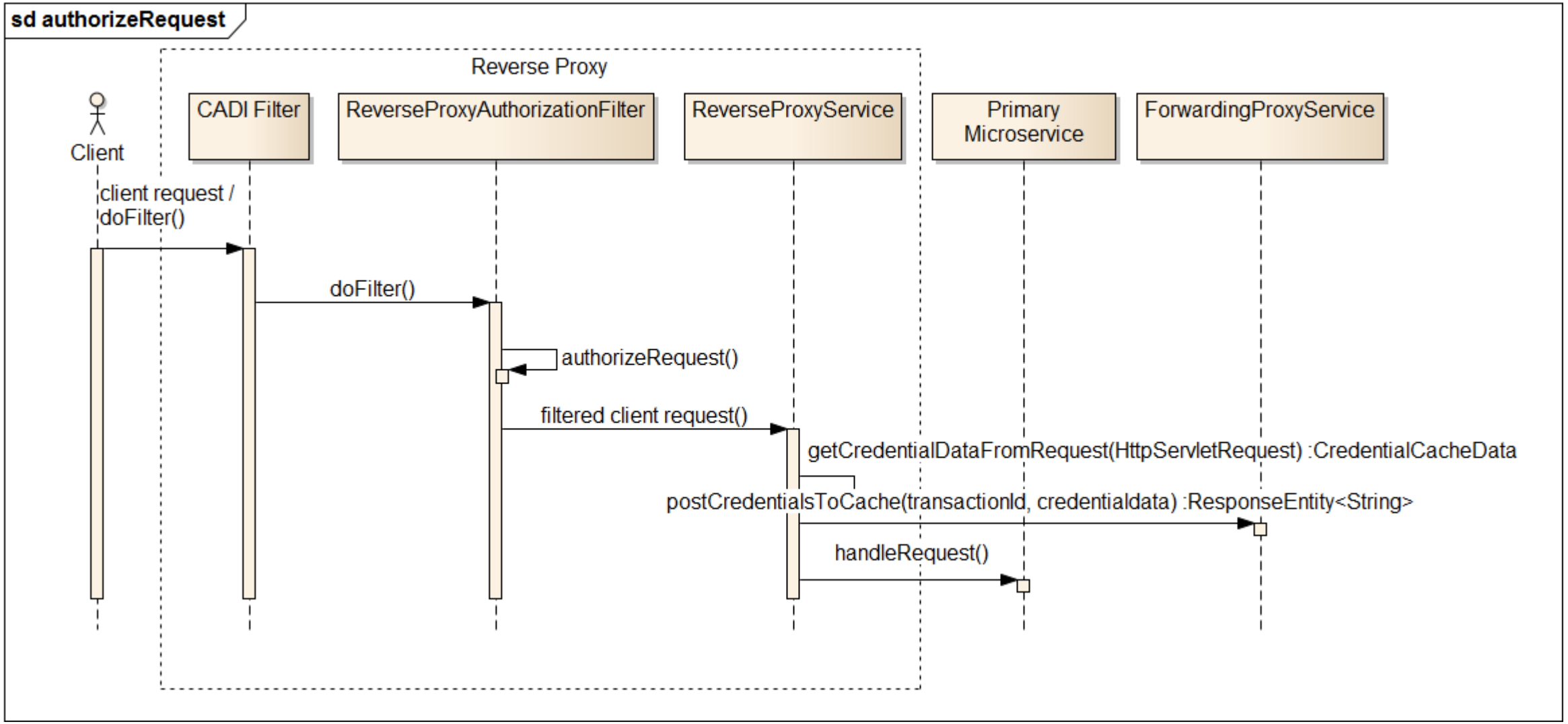
Configuration – uri-authorization.json

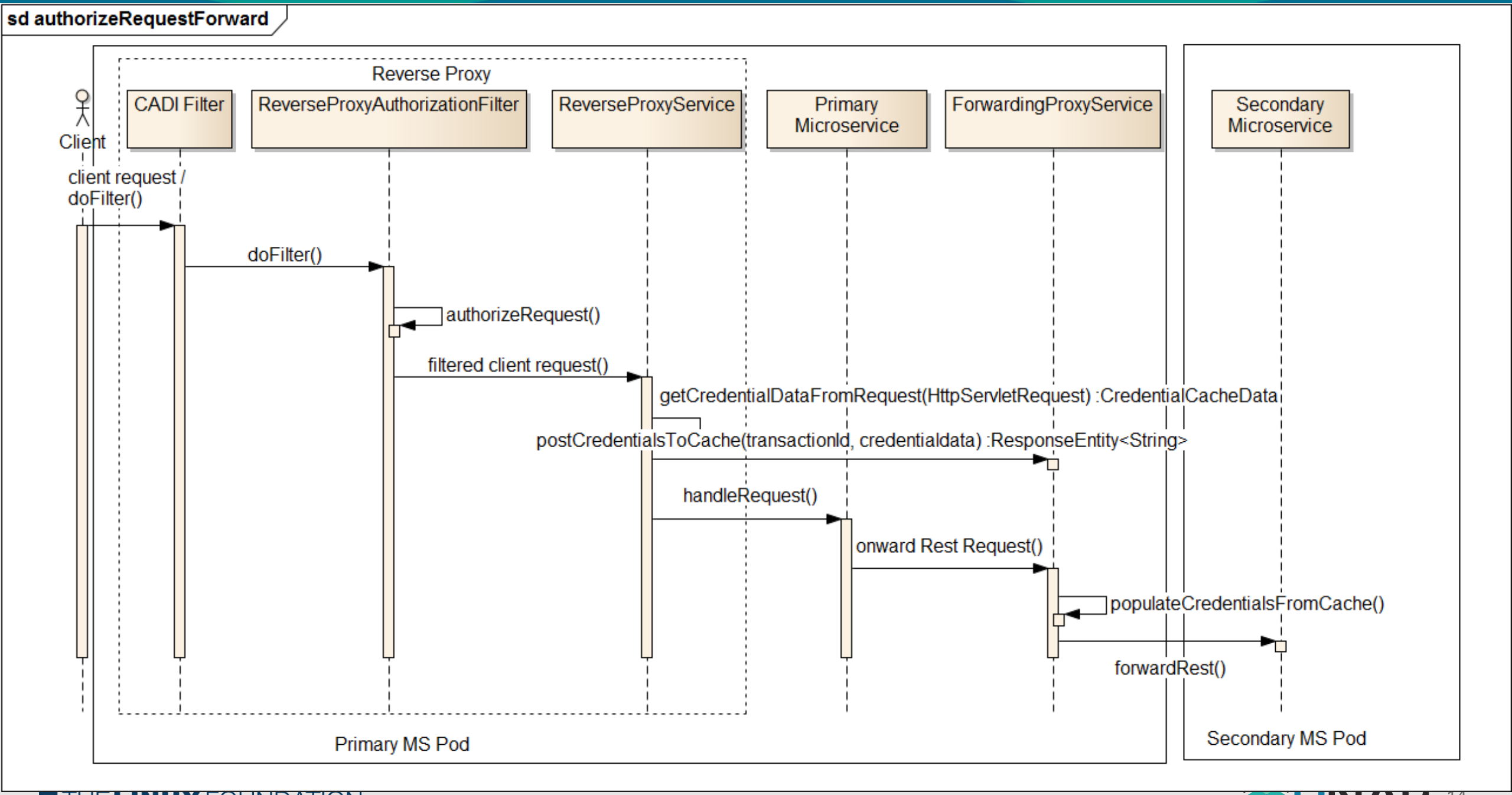

```
{
    "uri": "\/aai\/v13\/cloud-infrastructure\/cloud-regions$",
    "permissions": [
      "test\\.auth\\.access\\|rest\\|write",
      "test\\.auth\\.access\\|rpc\\|write"
     ]
  },
  {
    "uri": "\/aai\/v13\/cloud-infrastructure\/cloud-regions\/cloud-region\/[^\/]+[\/][^\/]+$*",
    "permissions": [
      "test.auth.access\\|clouds\\|read",
      "test.auth.access\\|tenants\\|read"
    ]
  }
```

Configuration – Authorization Algorithm

1. Attempt to match URI, tested in order from the config file.
    1. Matching is regex based.
2. On successful match, the permissions returned from the authorization provider, AAF in this case, are compared with the needed permissions in the configuration file
    1. All configuration file permissions are needed.
    2. Additional granted permissions are ignored
    3. Permissions are also matched on a regex basis
    4. AAF wildcarding is supported
3. All needed permissions match then authorized
4. Match fails then next URI is tested
5. No URI & permissions match then unauthorized

Code Overview




sd authorizeRequestForward
Reverse Proxy
Client
CADI Filter
ReverseProxyAuthorizationFilter
ReverseProxyService
Primary Microservice
ForwardingProxyService
Secondary Microservice
client request / doFilter()
doFilter()
authorizeRequest()
filtered client request()
getCredentialDataFromRequest(HttpServletRequest) :CredentialCacheData
postCredentialsToCache(transactionId, credentialdata) :ResponseEntity<String>
handleRequest()
onward Rest Request()
populateCredentialsFromCache()
forwardRest()
Primary MS Pod
Secondary MS Pod

tproxy-config init container & IPTables Rules

IPTable rules are configured for the pod using a Kubernetes init container. Iptables run from the container start.sh script.

All inbound traffic external to the pod is directed to the reverse-proxy port.

```
iptables -t nat -A PREROUTING -p tcp -j REDIRECT --to-port 10690
```

All outbound traffic that's not running under process owner 1001 goes to the forward proxy port. Reverse & forward proxies run under the 1001 owner so their traffic is unaffected.

```
iptables -t nat -A OUTPUT -p tcp -j REDIRECT --to-port 10680 -m owner '!'
--uid-owner 1001
```

Outbound traffic exceptions can be added, e.g. Cassandra DB access

```
iptables -t nat -A OUTPUT -p tcp -j ACCEPT --dport <exclude port number>
```

Logging

In each Kubernetes pod that is configured with pluggable security …

Log files:

Reverse Proxy

```
kubectl exec –it <pod id> -c reverse-proxy -- /bin/bash
/opt/app/rproxy/logs/reverse-proxy/application.log
```

Forward Proxy

```
kubectl exec –it <pod id> -c forward-proxy -- /bin/bash
/opt/app/fproxy/logs/AAF-FPS/application.log
```

Logging: Reverse Proxy Log

2019-02-19 10:30:50,113 DEBUG org.eclipse.jetty.server.Server [qtp107456312-13] REQUEST GET /services/inventory/v13/pserver on HttpChannelOverHttp@227618c1{r=1,c=false,a=DISPATCHED,uri=https://10.69.

1

49.131:30268/services/inventory/v13/pserver,age=1}

2019-02-19 10:30:50,116 INFO org.eclipse.jetty.server.handler.ContextHandler$Context [qtp107456312-13] 2019-02-19T10:30:50.116+0000 DEBUG [cadi] BasicHttpTaf: demo@people.osaaf.org authenticated by cached AAF password, ms=0.019453

2

2019-02-19 10:30:50,117 DEBUG org.eclipse.jetty.servlet.ServletHandler$CachedChain [qtp107456312-13] call filter reverseProxyAuthorizationFilter

3

2019-02-19 10:30:50,117 DEBUG org.onap.aaf.cadi.sidecar.rproxy.ReverseProxyAuthorizationFilter [qtp107456312-13] URI For authorization: /not/allowed/at/all$

2019-02-19 10:30:50,117 DEBUG org.onap.aaf.cadi.sidecar.rproxy.ReverseProxyAuthorizationFilter [qtp107456312-13] Needed permission:test.auth.access.ifYouLikedItYouShouldHavePutAPermissionOnIt

2019-02-19 10:30:50,118 DEBUG org.onap.aaf.cadi.sidecar.rproxy.ReverseProxyAuthorizationFilter [qtp107456312-13] URI For authorization: /one/auth/required$

2019-02-19 10:30:50,118 DEBUG org.onap.aaf.cadi.sidecar.rproxy.ReverseProxyAuthorizationFilter [qtp107456312-13] Needed permission:test.auth.access.aSimpleSingleAuth

Logging: Reverse Proxy Log

2019-02-19 10:30:50,122 DEBUG org.onap.aaf.cadi.sidecar.rproxy.ReverseProxyAuthorizationFilter [qtp107456312-13] The URI:/services/inventory/v13/pserver doesn't match any in the configuration:/backend$ 4

2019-02-19 10:30:50,122 DEBUG org.onap.aaf.cadi.sidecar.rproxy.ReverseProxyAuthorizationFilter [qtp107456312-13] The URI:/services/inventory/v13/pserver matches:/services/inventory/.* 5

2019-02-19 10:30:50,122 DEBUG org.onap.aaf.cadi.sidecar.rproxy.ReverseProxyAuthorizationFilter [qtp107456312-13] Permission match found - needed permission:org\.onap\.aai\.resources\|*\|.*, granted permission:org.onap.aai.resources|*|get

2019-02-19 10:30:50,122 INFO org.onap.aaf.cadi.sidecar.rproxy.ReverseProxyAuthorizationFilter [qtp107456312-13] Authorized

6

2019-02-19 10:30:50,127 DEBUG org.onap.aaf.cadi.sidecar.rproxy.ReverseProxyService [qtp107456312-13] Validated transaction ID: a0c3cb80-2df5-11e9-b210-d663bd873d93

2019-02-19 10:30:50,127 DEBUG org.springframework.core.log.CompositeLog [qtp107456312-13] HTTP POST https://localhost:10680/credential-cache/a0c3cb80-2df5-11e9-b210-d663bd873d93

Logging: Reverse Proxy Log – Unauthorized

2019-02-19 10:30:52,122 DEBUG org.onap.aaf.cadi.sidecar.rproxy.ReverseProxyAuthorizationFilter [qtp107456312-13] The URI:/services/inventory/v13/pserver doesn't match any in the configuration:/backend$

7

2019-02-19 10:30:52,123 INFO org.onap.aaf.cadi.sidecar.rproxy.ReverseProxyAuthorizationFilter [qtp107456312-13] Unauthorized

2019-02-19 10:30:54,132 DEBUG org.onap.aaf.cadi.sidecar.rproxy.ReverseProxyAuthorizationFilter [qtp107456312-13] The URI:/services/inventory/v13/pserver matches:/services/inventory/.*

2019-02-19 10:30:54,132 DEBUG org.onap.aaf.cadi.sidecar.rproxy.ReverseProxyAuthorizationFilter [qtp107456312-13] Permission match found - needed permission:org\.onap\.aai\.resources\|*\|.*, granted permission:org.onap.aai.resources|*|get

8

2019-02-19 10:30:54,133 INFO org.onap.aaf.cadi.sidecar.rproxy.ReverseProxyAuthorizationFilter [qtp107456312-13] Unauthorized

Logging: Forward Proxy Log

Forward proxy caching transaction id received from reverse proxy & adding to onward request…

```
2019-02-19 10:30:50,146 INFO org.onap.aaf.cadi.sidecar.fproxy.service.ForwardingProxyService
[qtp1890627974-12] Updating credential cache with transaction ID: a0c3cb80-2df5-11e9-b210-
d663bd873d93

2019-02-19 10:30:50,149 INFO org.onap.aaf.cadi.sidecar.fproxy.service.ForwardingProxyService
[qtp1890627974-12] Credential cache successfully updated with transaction ID: a0c3cb80-2df5-11e9-
b210-d663bd873d93


2019-02-19 10:30:50,229 INFO org.onap.aaf.cadi.sidecar.fproxy.service.ForwardingProxyService
[qtp1890627974-12] Request received: /services/champ-service/v1/objects/filter

2019-02-19 10:30:50,230 INFO org.onap.aaf.cadi.sidecar.fproxy.cache.utils.CacheUtils
[qtp1890627974-12] Populating header credentials from cache for transaction ID: a0c3cb80-2df5-11e9-
b210-d663bd873d93

2019-02-19 10:30:50,230 INFO org.onap.aaf.cadi.sidecar.fproxy.cache.utils.CacheUtils
[qtp1890627974-12] Header credentials successfully populated.

2019-02-19 10:30:50,230 INFO org.onap.aaf.cadi.sidecar.fproxy.service.ForwardingProxyService
[qtp1890627974-12] Forwarding request...
```

1

2

3

4

5

6

Failure Scenarios and Debugging

Trace the request through the reverse proxy, primary microservice & for onward requests the forward proxy logs.

Correlate requests using the transaction id header.

Issue	Debug steps
Unauthorized	✓ Check that credentials are supplied in the request ✓ Check the credentials are valid in the IdP ✓ Check the end point configuration for the Idp & AnP ✓ Check the permissions returned from the AnP ✓ Check the permissions & URI configuration in uri-authorization.json
TLS connection errors	✓ Check all container services are up and running ✓ Check service end point configuration ✓ Check that both server & client certificates are valid

Sidecar Deployment Impact

- Pluggable security sidecar containers will add a minimum of ~ 260 MB additional memory requirement per micro service.
- Testing to date shows that deployment of pluggable security will add 95ms on average per request handled by a single microservice
- Testing to date shows that deployment of pluggable security will add 225ms on average per request handled by a microservice with onward calls to another microservice
- Above figures are for a sequence of single requests.
- Concurrent multiple requests under investigation.

Source Control

git clone https://gerrit.onap.org/r/aaf/cadi

In particular

sidecar/fproxy

sidecar/rproxy

sidecar/tproxy-config

Build Jobs

https://jenkins.onap.org/job/aaf-cadi-tproxy-config-master-aai-docker-java-daily/

https://jenkins.onap.org/job/aaf-cadi-rproxy-master-aai-docker-java-daily/

https://jenkins.onap.org/job/aaf-cadi-fproxy-master-aai-docker-java-daily/

Deployment Via OOM

Change in kubernetes/aai/values.yaml

…

Global:

…

installSidecarSecurity: ~~false~~ true

For Resources

profiles:

active: production,dmaap,~~aaf-auth~~ two-way-ssl

Helm Implementation

1. Under <microservice>/resources add
   1. rproxy configuration for the rproxy sidecar: property & configuration files as mentioned earlier + logging config
   2. fproxy configuration for the fproxy sidecar: fproxy properties & logging configuration
2. Under templates, surround the creation of the additional Kubernetes objects with conditionals that test for the installSidecarSecurity value

```
{{ if .Values.global.installSidecarSecurity }}
…
{{end}}
```

3. Needed configmap, deployment, secrets & service. See aai/charts/aai-gizmo as an example.
4. Resources has a revised haproxy config to authorize the health check and restate the certificate based config file.

Helm Implementation & Future Enhancement

- Currently relies on Helm conditional tests to deploy sidecar & init containers.
- The scripts for each microservice that is configured with sidecar security needs a significant amount of “boiler plate” scripting
- In future it may be possible to remove this with Kubernetes features like Admission Controllers, Initializers .Mutating Webhook Admission Controllers seem most appropriate.
- The features of interest are currently either alpha or beta so no implementation of sidecar deployment to date.

Documentation

Developer documentation at …

https://wiki.onap.org/display/DW/Pluggable+Security

Additional documentation in the Reverse & Forward proxy source READMEs

https://gerrit.onap.org/r/gitweb?p=aaf/cadi.git;a=blob;f=sidecar/rproxy/README.md;h=c58e0ba36e456c3ee0add1a28bbc5521ae741f59;hb=refs/heads/master

https://gerrit.onap.org/r/gitweb?p=aaf/cadi.git;a=blob;f=sidecar/fproxy/README.md;h=abd6558f1f1e2d6d0568e324a57909618eb37cb8;hb=refs/heads/master