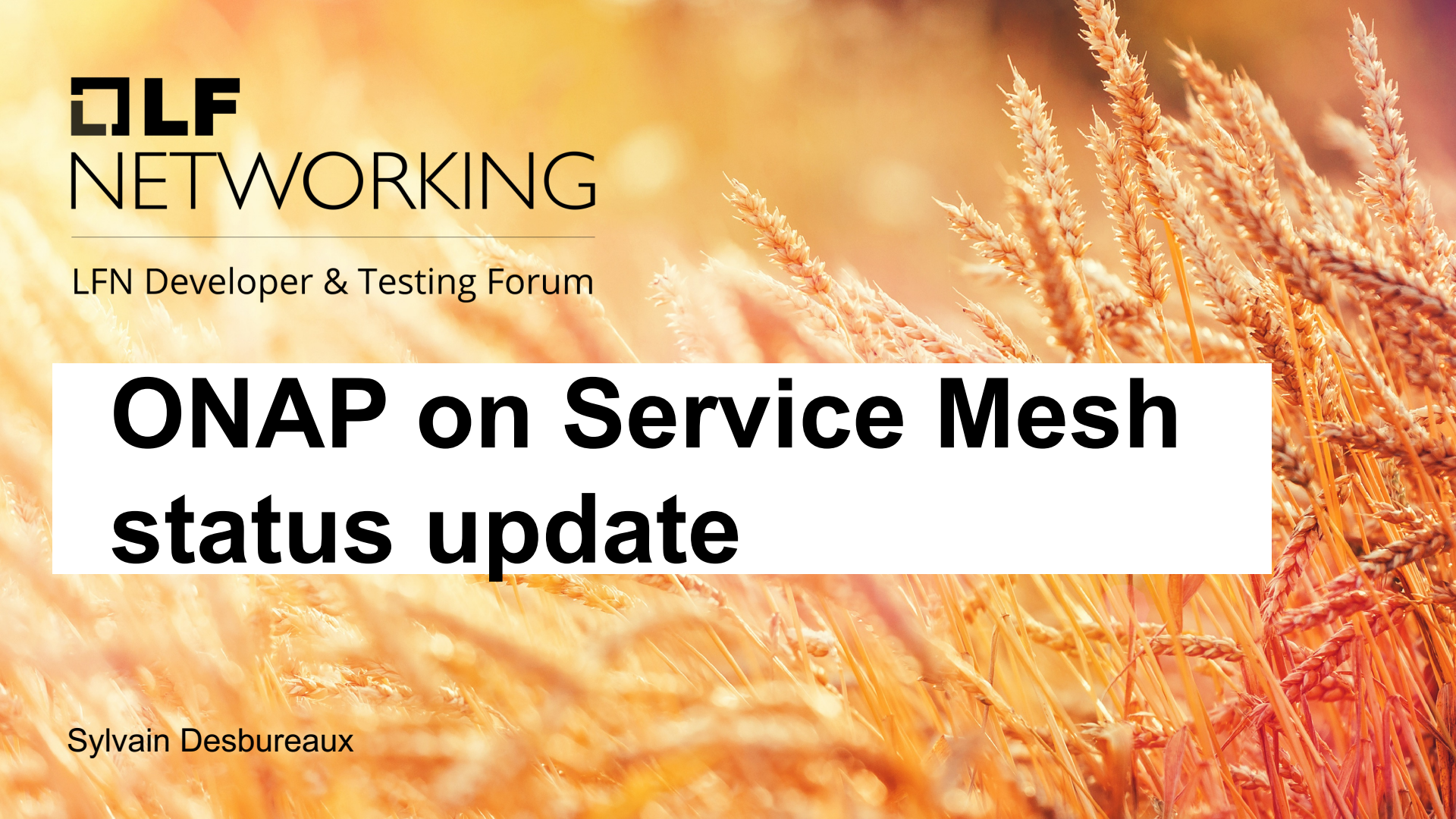
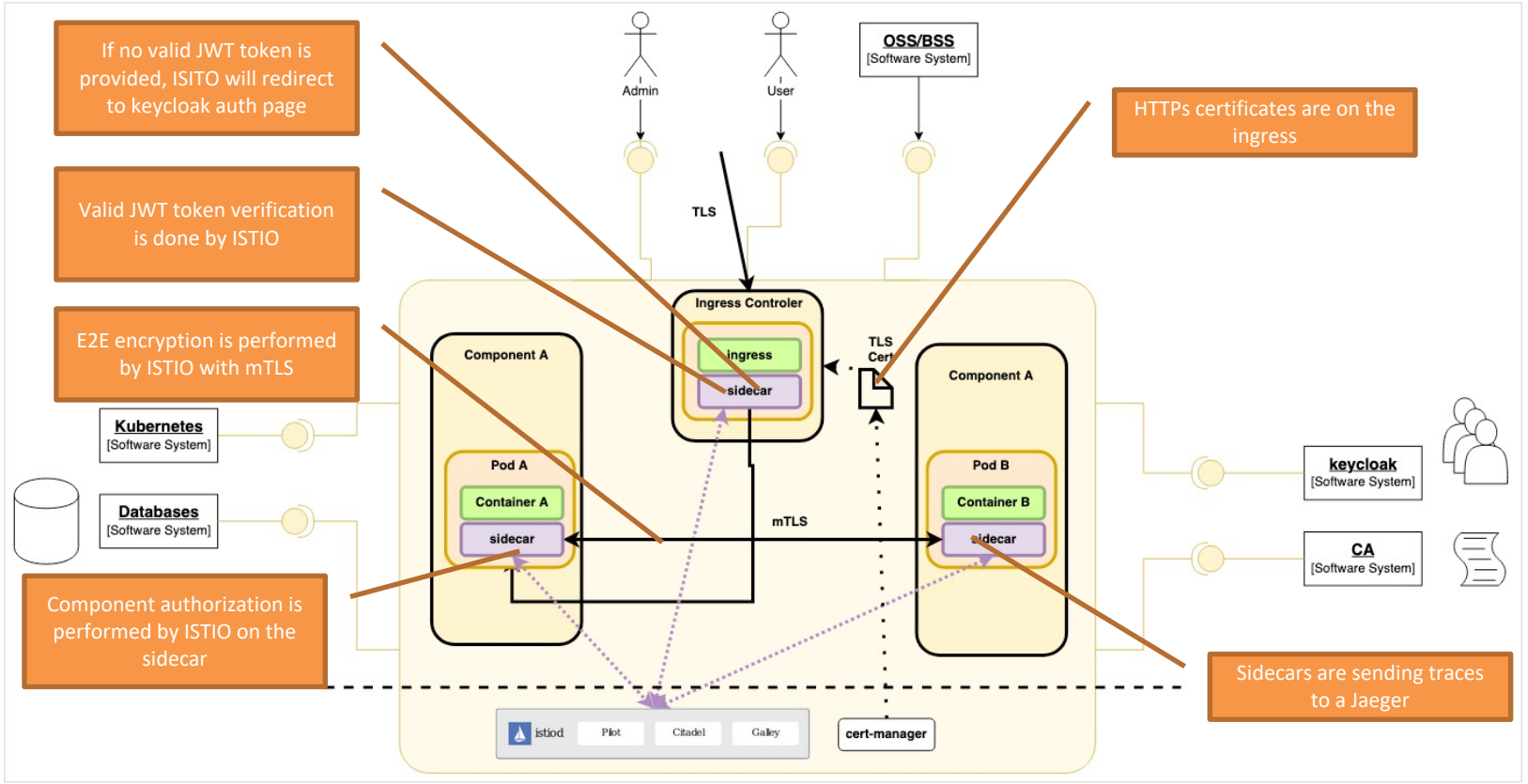

LF
NETWORKING
LFN Developer & Testing Forum



ONAP on Service Mesh status update

Sylvain Desbureaux

Service Mesh: HLD



Step one: Certificates(1)

- We “just” deploy ONAP on a namespace where Istio is enabled
- OOM changes:
  - Certificates are no more retrieved on the component deployment/statefulset but set on the Ingress
    - Use of cert-manager to manage the certificate lifecycle
- Components change:
  - The component disable AAF integration if any
  - The component must listen on HTTP/gRPC (no HTTPs)
    - Most if not all components are able to do that via configuration change (from simple to hard)
  - The component must talk to other components using HTTP (no HTTPs)
    - Latest SDC distribution clients allows HTTP, not older version so you should upgrade if planning to talk to SDC

Step one: Certificates(2)

- Expected benefits:
  - e2e encryption even for internal communication
  - Same TLS configuration (allowed TLS versions, allowed ciphers, …) for all components
  - Certificate root choice should be simple with several way of doing it thanks to cert-manager
  - Certificate upgrade doesn't require new deployment
- What doesn't change:
  - Previous authorization remains (i.e. NBI needs to use basicAuth to talk to SO as of today for example)

Step one: Certificates

- Istio compatible components:
  - Cassandra
  - Mariadb
  - SDC client is able to run on HTTP
  - AAI
  - CDS (95% done)
  - DMaaP MR
  - SDC (95% done)
  - SO (80% done)

« basic_onboarding » has been successfully launched

- Components using http compatible sdc client and being able to disable https for it
  - AAI
  - CDS (needs a new release but works with “latest” version)
  - SDNC (needs a new release but should work with “latest” version)
  - SO

Step two: Authorization (1)

- On top of step 1, we create “AuthorizationPolicy”[1] that will authorize some components to talk to others
  - For example, Only SO and ESR should be able to initiate request on Multicloud (just an example, it may not be the case)
- OOM changes:
  - The AuthorizationPolicy resources must be created
  - Specific service account per subcomponent must be created (Authorization policy works with them)
- Components change:
  - The component disables MSB integration if any (MSB breaks the ”line of trust”)
  - The internal only component disables its “basicAuth” if it has one (see next slide)

1: https://istio.io/docs/reference/config/security/authorization-policy/

What does mean “The component disables its “basicAuth” if it has one” in term of code

- It will depend obviously on how each projects are behaving
- For SO, it would imply to use https://github.com/onap/so/blob/master/common/src/main/java/org/onap/so/security/SoNoAuthWebSecurityConfigurerAdapter.java as security adapter
  - It’s used in “test” and “aaf” profile so I guess a new profile (“serviceMesh”) should be created
- For SDC, it would mean to create a new filter here (https://github.com/onap/sdc/tree/master/catalog-be/src/main/java/org/openecomp/sdc/be/filters) I guess
- For AAI “traversal”, it would also mean to create an interceptor associated to the a new profile here https://github.com/onap/aai-traversal/tree/master/aai-traversal/src/main/java/org/onap/aai/interceptors/pre I guess

Step two: Authorization (2)

- Expected benefits:
  - Component Authorization for all components
  - Centralized management of Authorization, which can simplify security audit
  - AuthorizationPolicy can be very specific (I allow SO to do only GET or POST to /stuff only) and so we can have a better security than today (“healthcheck” user can do anything he wants on policy)
- What doesn’t change:
  - External access is still based on basic Auth or “open bar” as of today

Step one: Authorization

- All supported components have a service account il all their workloads
- SO has a specific profile in order to disable basic authentication

Step three: simple RBAC (1)

- On top of step 2, we also add JWT configuration on "AuthorizationPolicy"[1] that will authorize some user to access the components
  - If the user has not a valid authentication token, it gets forwarded to keycloak authentication portal to retrieve it.
- OOM changes:
  - The AuthorizationPolicy resources must be created
  - ISTIO must be configured to accept external traffic with a valid JWT token
  - ISTIO must be configured to redirect to auth portal if no auth token or bad one is present
  - Keycloak must be configured with right users / authorization
- Components change:
  - None from step 2 perspective
  - The components with external access disable its "basicAuth" if it has one (same as step 2 for internal only components)

Step three: simple RBAC (2)

- Expected benefits:
  - Centralized User Management (but with a rather simple OK / NOK)
  - External User Management (we can plug directly OpenID connect compatible IAM or LDAP/SAML IAM via Keycloak)

- What doesn't change:
  - User access authorization is only performed on the first component (NBI, VID, UUI, …)
    - If User is granted, NBI will use its own AuthorizationPolicy with SO / AAI and so every allowed NBI users will be authorized for everything (Cannot read only AAI for user X and service create for user Y).
  - Some User grant can be provided according to the path but not to the body

Step three: simple RBAC

- PoC with Istio / keycloak / oauth2-proxy and “simple” services has been performed

- Services used
  - Kiali (auth only)
  - Grafana (auth + group mapping)
  - Httpbin (auth only)

Step four: full RBAC (1)

- On top of Step 3, component retrieve JWT token and:
  - Use it to check if user has right to perform the wanted action (if needed)
  - Add it to “south” request headers so next component can check if the client has the right to perform (NBI pass to SO which will ultimately pass it to Multicloud, so Multicloud can verify that this specific user has the right on this cloud)
- OOM changes:
  - None from step 3 perspective
- Components change:
  - The component retrieves JWT token header (see next slide)
  - The component verify that action is allowed according to this token (optional)
  - The component add Authorization header to the ”south” HTTP requests he’s doing (see next next slide)

What does mean “The component retrieves JWT token header ” in term of code

- It will depend obviously on how each projects are behaving but it’s roughly the same than retrieving basic Auth header

- For spring boot code, it’s something like:

```
public String someMethod(@RequestHeader("Authorization") String token) {}[1]
public String someMethod(@RequestHeader("X-Auth-Request-Access-Token") String token) {}[1]
```

  - Obviously, an interceptor can be made and handle this part on every requests

- For python flask code, it’s something like:

```
from flask import request
request.headers.get('your-header-name’)[2]
request.headers.get(' X-Auth-Request-Access-Token’)[2]
```

1: https://www.baeldung.com/spring-rest-http-headers
2: https://flask.palletsprojects.com/en/1.1.x/quickstart/#accessing-request-data

What does mean “The component add Authorization header to the ”south” HTTP requests he’s doing” in term of code

- Again, it’ll depend on implementation
- On spring boot, an interceptor / filter can do that for you (some examples here: https://stackoverflow.com/questions/46729203/propagate-http-header-jwt-token-over-services-using-spring-rest-template)
- If you’re using a ”client” (sdc client, so client, …), then the client must be able to add some headers on the request (if the client use RestTemplate from Spring Boot, not sure if possible, you interceptor/filter will do the job for you)
- On Python request, it’s just adding the header

Step four: full RBAC (2)

- Expected benefits:
  - Real RBAC (user propagation allows to verify it has the right to end component and not only the first).
  - We can create several profiles (admin, designer, serviceOwner, …) and components can rely on them know if the user has the rights to perform the intended actions
  - We can start “low” (YES/NO approach per profile) and go deeper (user X has no right on resource Y), allowing us to be multi tenant

Step four: full RBAC

- SO is handling Authorization headers retrieval and pass to the next components
  - It'll have to be changed to retrieve the token and generate the authorization header

Cherry on the cake: tracing (1)

- Tracing is enabled by default when ISTIO is started.
- But ISTIO has no clue that south request is part of a bigger request
- We must then “help him” to do that

- OOM changes:
  - None from step 4 perspective

- Components change:
  - The component retrieves “B3” headers (https://github.com/openzipkin/b3-propagation) (see next slide)
  - The component add these headers to the ”south” HTTP requests he’s doing

What does mean “The component retrieves “B3 headers” in term of code

- It’s the same that for JWT token, except for
  - Spring boot:
    - Spring Boot Sleuth (https://spring.io/projects/spring-cloud-sleuth) can handle this for you (and add them to south request if RestTemplate is used)
    - It will also add the request/span into the logs, for better understand of the logs
  - Go
    - https://github.com/openzipkin/zipkin-go/propagation/b3 may help you
- So the « cost » of such feature is very low

Cherry on the cake: tracing (2)

- Expected benefits
  - E2e tracing will help to find bottlenecks
  - Added to logging, we can also have e2e logs (find why NBI creation had an issue on SDN-C preload may be tricky today for example)

Cherry on the cake: tracing

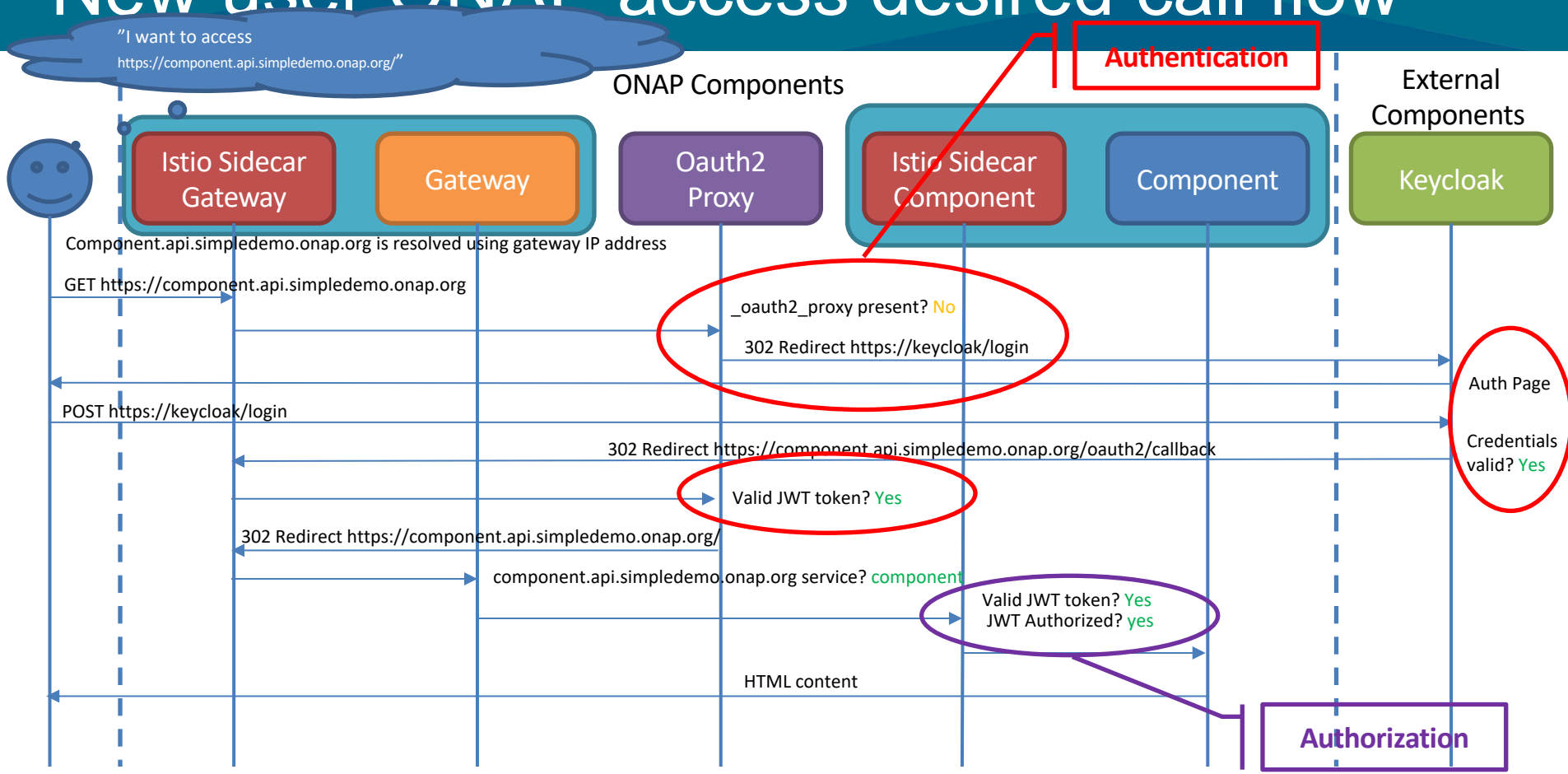
- SO is propagating the header in its core components

Conclusion

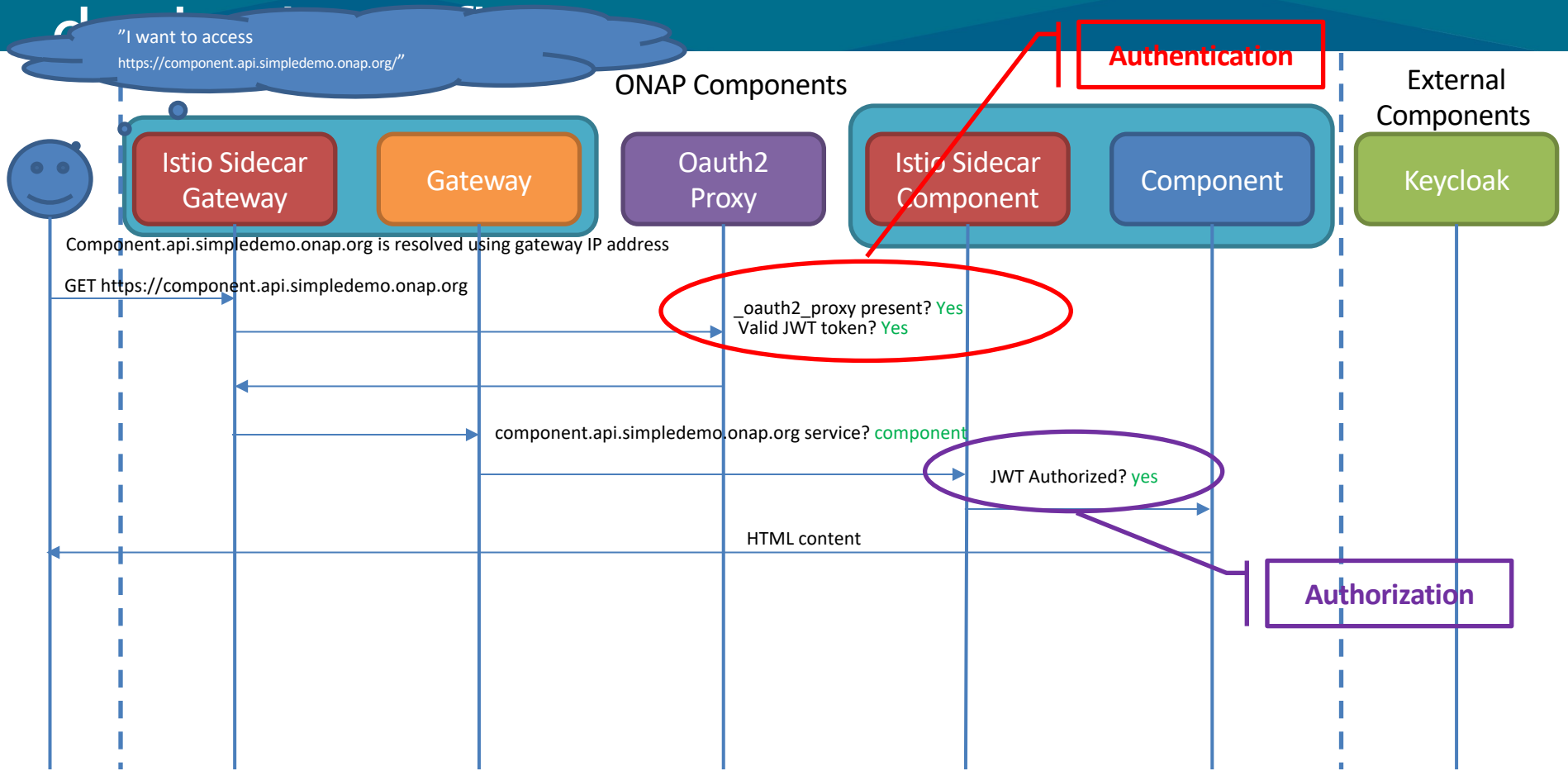
- Works is advancing at slow pace but it’s advancing
- We can hope for service onboarding and creation for Jakarta with service mesh (and start on Authorization)


LF
NETWORKING
LFN Developer & Testing Forum
Thanks

New user ONAP access desired call flow



Already connected user ONAP access



Already connected user ONAP access desired call flow with subcomponent