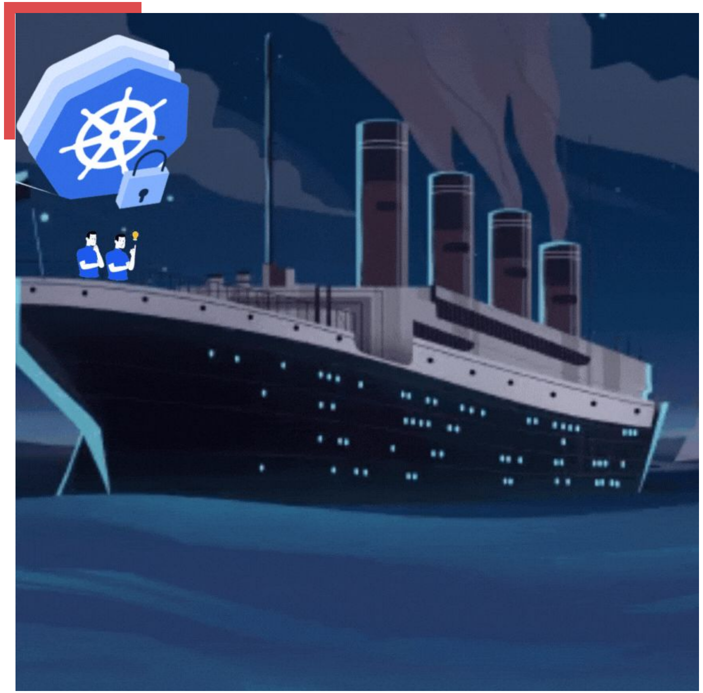
Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Technology
Arts Sciences
TH Köln



GRANULAR ACCESS CONTROL TO KUBERNETES COMPONENT USING OPENID CONNECT

Presenter → Dikshita Kalita
Supervisor → Prof. Dr. Martin Leischner
Mentor → Richard Clauß
Date → 20.01.2023

Agenda

01 Motivation

02 Introduction

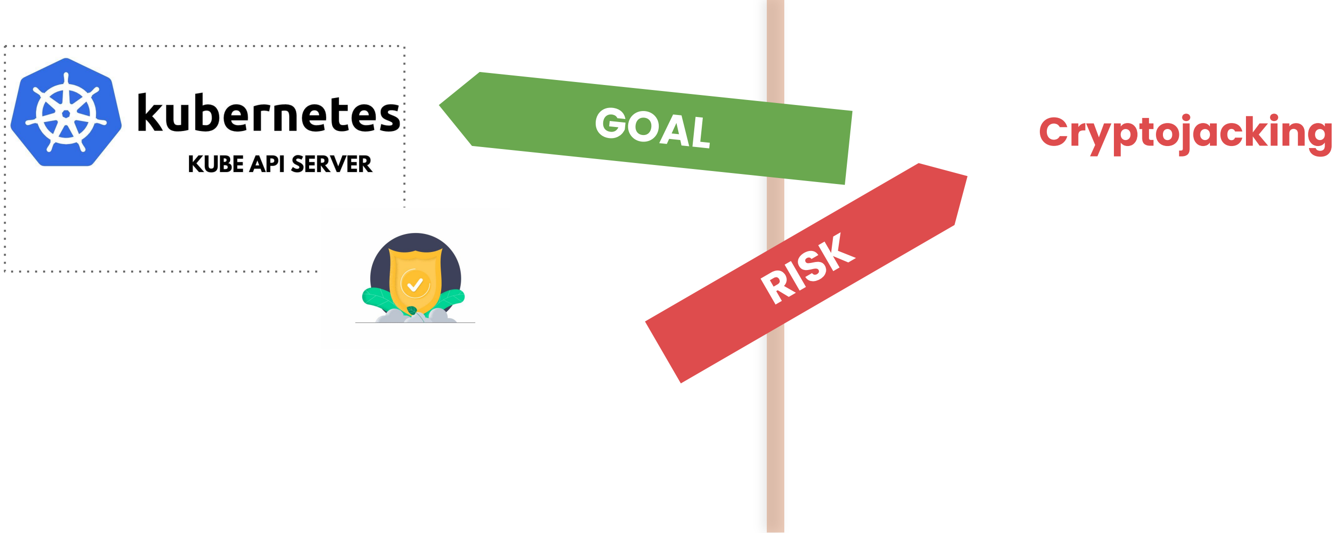
03 Summary

04 Result

05 State of art

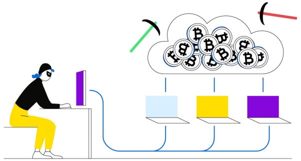
06 Sources

01 Motivation



02 Introduction | What is Cryptojacking ?

Cyberattack where attackers hijack a target’s computer to mine cryptocurrency illegally without the user’s awareness



? Why cryptocurrency?

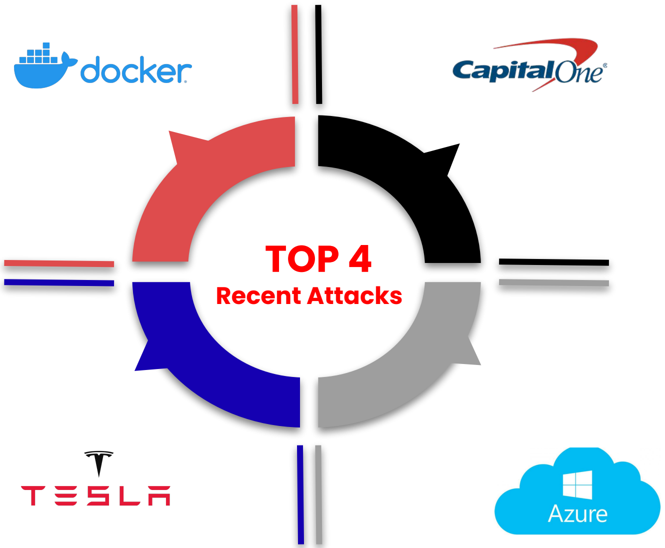
- It is an integral and widely used means of global value transfer.

Which platforms attackers use? ?

- Through containerized platforms like Docker and Kubernetes
- Cloud infrastructures provide a greater range of computation capacities, hence attackers attack them

Source: [1]

02 Introduction | Recent attack reports



Sources: [2], [3]

02 Introduction | How cryptojacking happens?

Tactics →

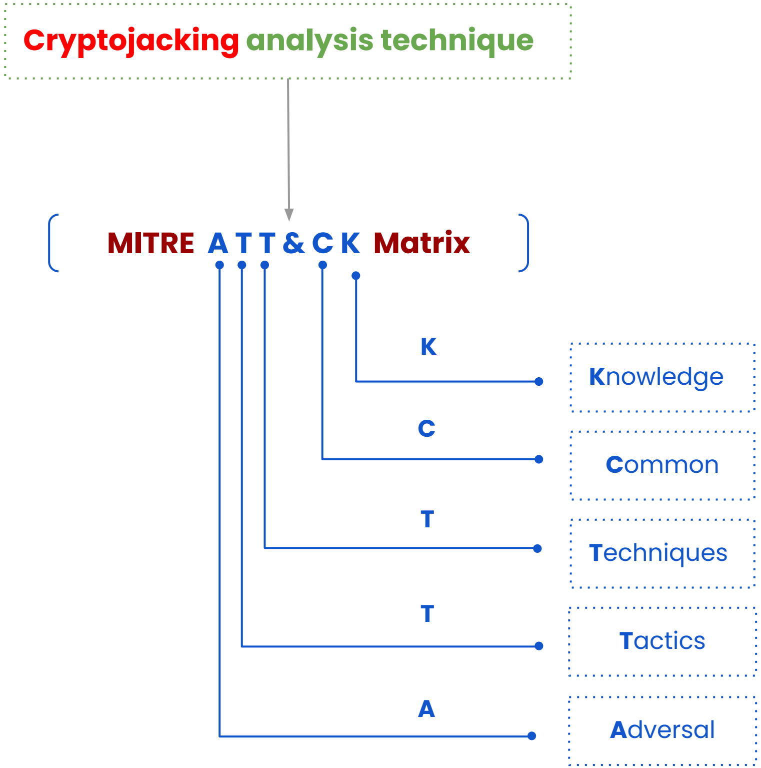
Techniques ↓

Initial access	Execution	Persistence	Privilege escalation	Defense evasion	Credential access	Discovery	Lateral movement	Collection	Impact
Using cloud credentials	Exec into container	Backdoor container	Privileged container	Clear container logs	LIst K8s secrets	Access K8s API server	Access cloud resources	🟨 Images from private registry	Data destruction
Compromised images in registry	Bash/cmd inside container	Writable hostpath mount	Cluster-admin binding	Delete K8s events	Mount service principal	Access Kubelet API	Container service account		Resource hijacking
Kubeconfig file	New container	Kubernetes CronJob	hostPath mount	Pod/container name similarity	Access container service account	Network mapping	Cluster internal networking		Denial of service
Application vulnerability	Application exploit(RCE)	🟨 Malicious admission controller	Access cloud resources	Connect from proxy server	Application credentials in config files	Access kubernetes dashboard	Application credentials in config files		
🟨 Exposed sensitive interfaces	SSH server running inside container				🟨 Access managed identity credentials	Instance Metadata API	Writable volume mounts on host		
	🟨 Sidecar injection				🟨 Malicious admission controller		🟨 CoreDNS poisoning		
							🟨 ARP poisoning and IP spoofing		

🟨 = Newly introduced technique

Source: [4]

02 Introduction | Method to analyse



Cryptojacking analysis technique

MITRE ATT&CK Matrix

K Knowledge

C Common

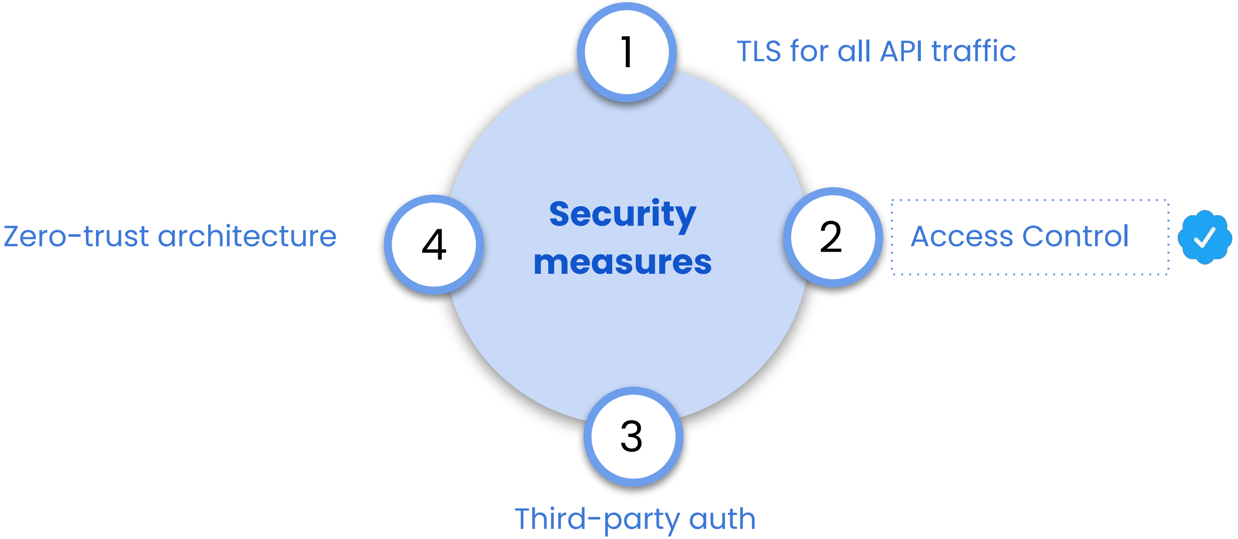
T Techniques

T Tactics

A Adversal

Source: [4]

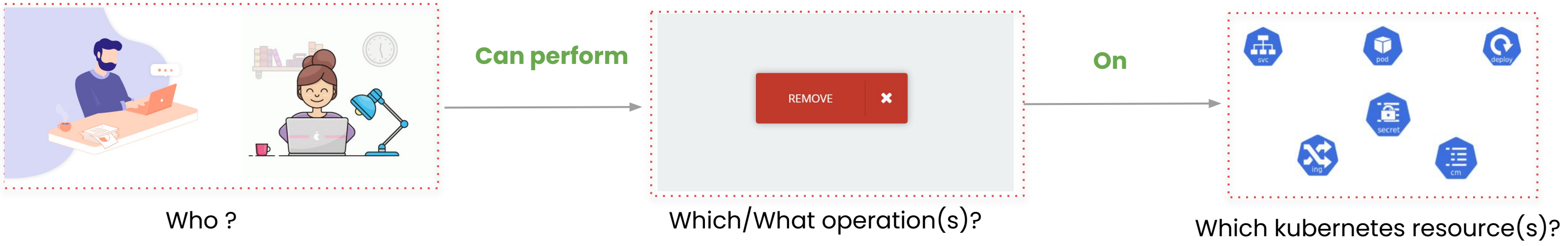
02 Introduction | Important security measures



NOTE: *Integrating Kubernetes with third party auth providers uses the remote platform's identity guarantees (backed up by things like 2FA) and prevents administrators having to reconfigure the kubernetes API server to add or remove users.*

Source: [5], [24], [25]

02 Introduction | Access control in kubernetes



Who ?

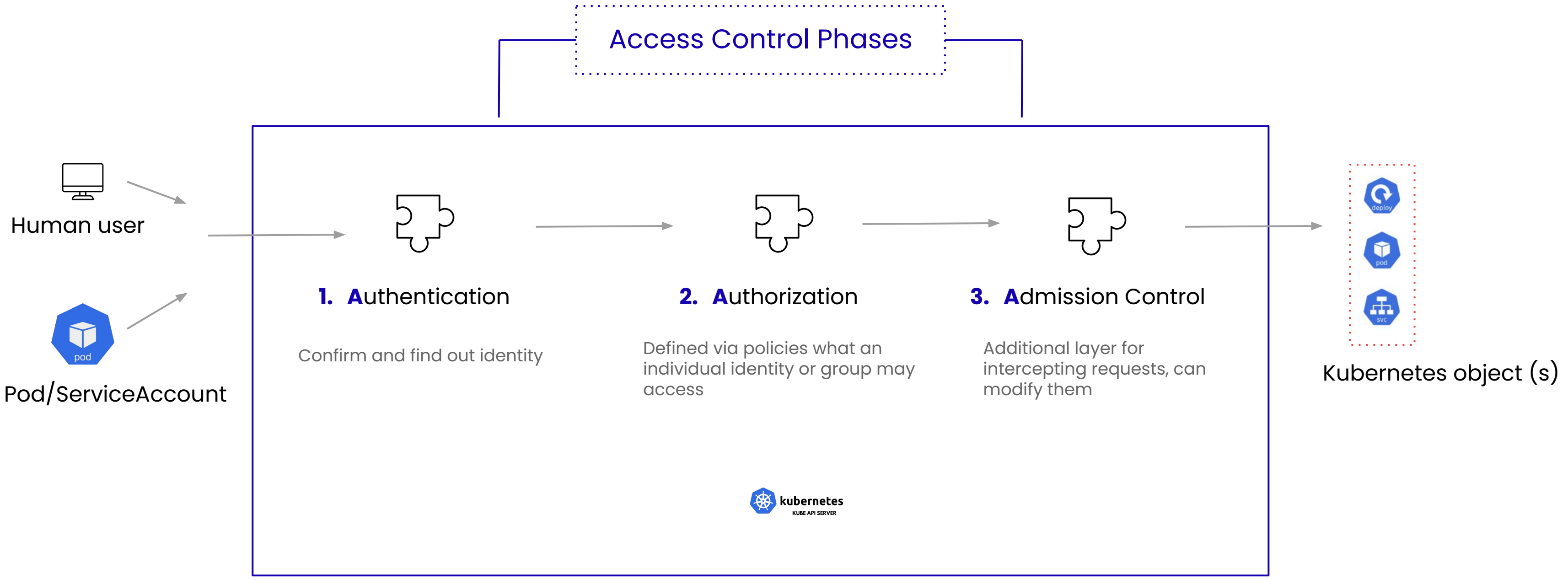
Can perform

Which/What operation(s)?

On

Which kubernetes resource(s)?

02 Introduction | Three phases of access control in kubernetes



Access Control Phases

Human user

Pod/ServiceAccount

1. **A**uthentication

Confirm and find out identity

2. **A**uthorization

Defined via policies what an individual identity or group may access

3. **A**dmission Control

Additional layer for intercepting requests, can modify them

Kubernetes object (s)

Reference: [14], [15]

02 Introduction | Authentication & Authorization strategies

kubernetes

Authentication Strategies

- Static Token 🔴

  Not scalable

- Client certificates | X509 Client Certificates 🔴

  Long-lived and can't be revoked effectively

- Token | JSON Web Tokens (JWTs) [Base64URL encoded JSON objects]

  - OpenID Connect 🟢

    1. Very secure
    2. Tokens are short-lived
    3. No runtime coupling between OIDC provider and kube-API server

Authorization Modes

- Node
- ABAC
- RBAC
- Webhooks

02 Introduction | Implementation of access control

CENTRALIZED AUTHENTICATION

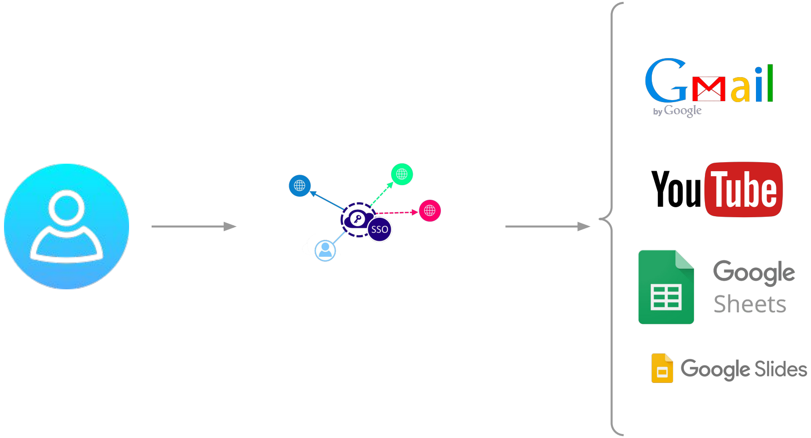
at

Ingress controller

Source: [9]

02 Introduction | What is centralized authentication?

- This service is also called Single sign-on
- Allows a user to access multiple applications *with one set of login credentials*
- Built on a concept called *federated identity*
- Enables sharing of identity across trusted but independent systems



02 Introduction | Workflow of SSO

- Workflow is based upon

Service Provider

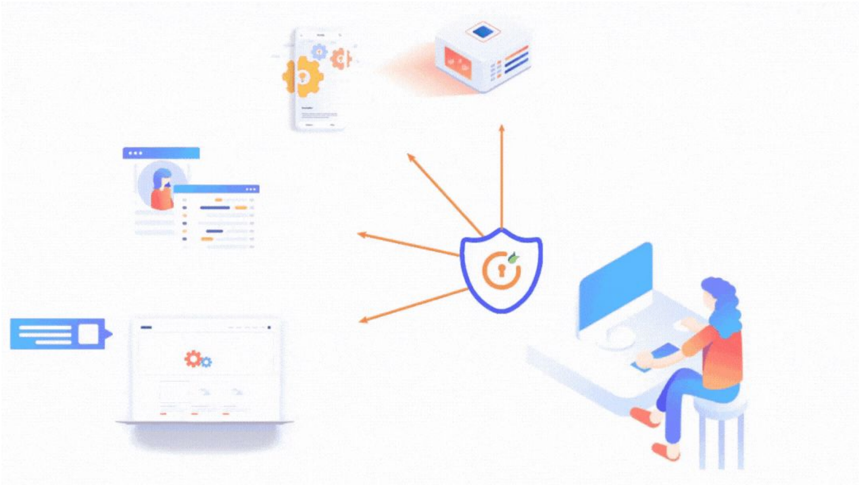
&

Identity Provider

Federation partner providing services to end-users

Federation partner authenticating users and provides authentication token to Service Provider

02 Introduction | Advantages



1 Better administrative control

2 Decreased attack surface

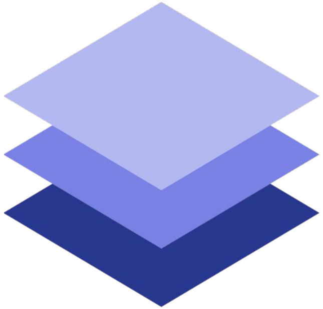
3 Seamless and secure user access

4 Better network security

5 SSO as part of an identity and access management (IAM) solution, utilizes a central directory that controls user access to resources at a more granular level

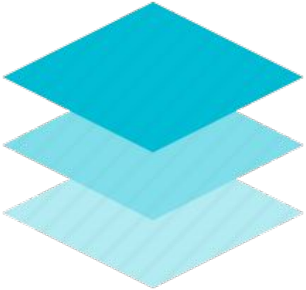
Sources: [20], [21]

02 Introduction | Where to position SSO?



Application Layer

OR



Ingress Layer

REASON

- Developers are free from building, maintaining the authentication logic
- Developers can easily leverage SSO technologies here using native kubernetes API

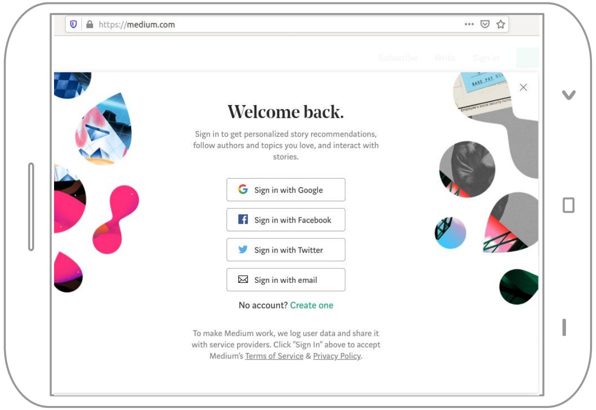
Source: [9]

02 Introduction | Commonly used protocol

Open Authentication / OAuth

OpenID Connect / OIDC

Security Access Markup Language / SAML



https://medium.com

Welcome back.

Sign in to get personalized story recommendations, follow authors and topics you love, and interact with stories.

Sign in with Google

Sign in with Facebook

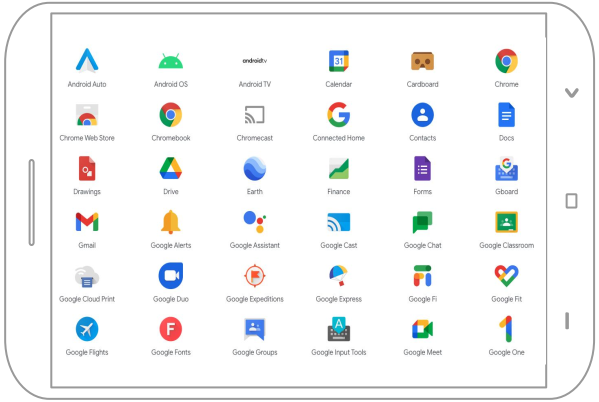
Sign in with Twitter

Sign in with email

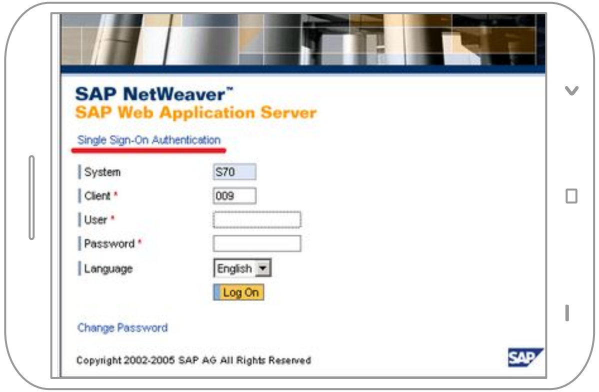
No account? Create one

To make Medium work, we log user data and share it with service providers. Click "Sign In" above to accept Medium's Terms of Service & Privacy Policy.

Credit: Medium



Credit: Google Developer



SAP NetWeaver™
SAP Web Application Server

Single Sign-On Authentication

System	S70
Client *	009
User *	
Password *	
Language	English

Log On

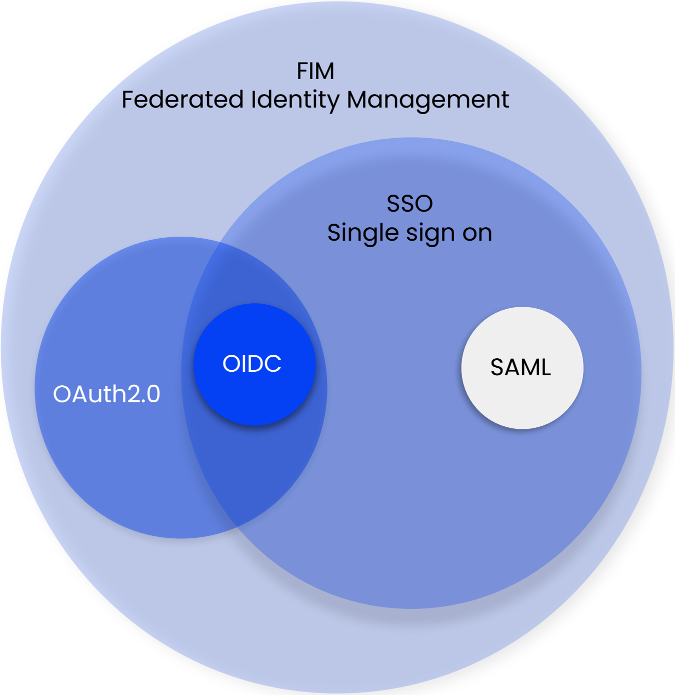
Change Password

Copyright 2002-2005 SAP AG All Rights Reserved

Credit: Wiki sap

Source: [27]

02 Introduction | SSO in a broader picture



Legends :

FIM	Refers to a trust relationship created between two or more domains or identity management systems.
SSO	Feature available within FIM architecture
OAuth2.0	Framework considered to be part of FIM architecture. It focuses on trusted relationship allowing user identity information to be shared across the domains.
OIDC	Authentication layer built on top of OAuth 2.0 to provide Single Sign-on functionality
SAML	Security Access Markup Language

Source: [10]

02 Introduction | What is OpenID Connect?

- It adds the missing identity layer to OAuth 2.0
- It provides authentication in the form of ID Tokens

- API security model that controls access to APIs
- Does not provide any (standardized) way for the client to request or control user authentication.

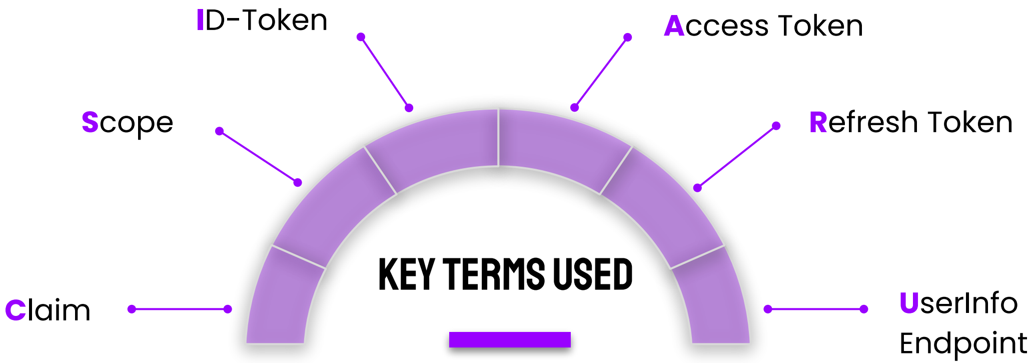
Source: [19]

02 Introduction | Working of OpenID Connect

- OIDC uses the same components and architecture as OAuth, ***but to authenticate***.

ACTORS

- OpenID Provider (OP)
- Relying Party (RP)
- End-user

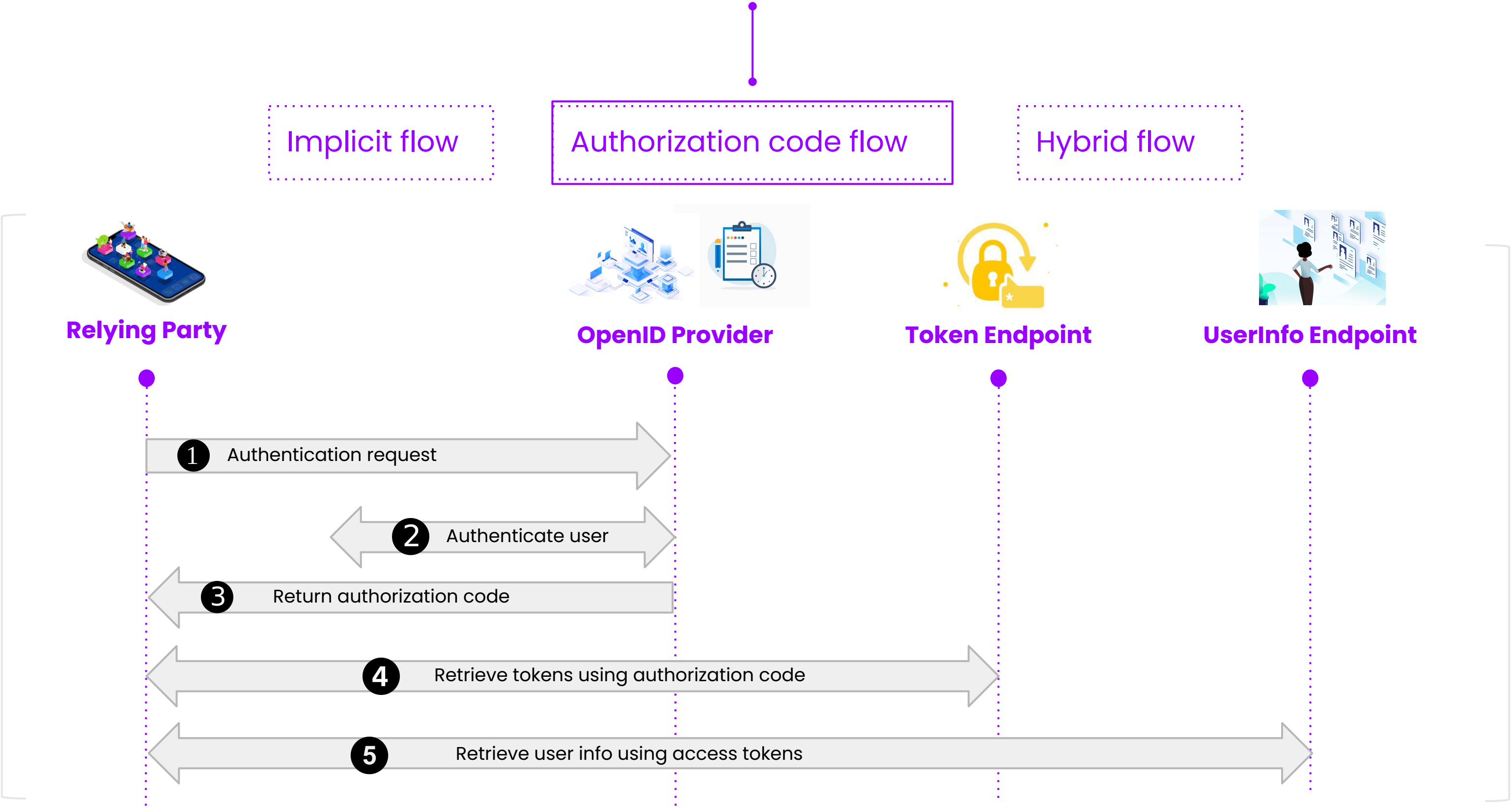


KEY TERMS USED

- ID-Token
- Access Token
- Scope
- Refresh Token
- Claim
- UserInfo Endpoint

Source: [18]

02 Introduction | Overview of authentication workflow in OIDC



Implicit flow

Authorization code flow

Hybrid flow

Relying Party

OpenID Provider

Token Endpoint

UserInfo Endpoint

1 Authentication request

2 Authenticate user

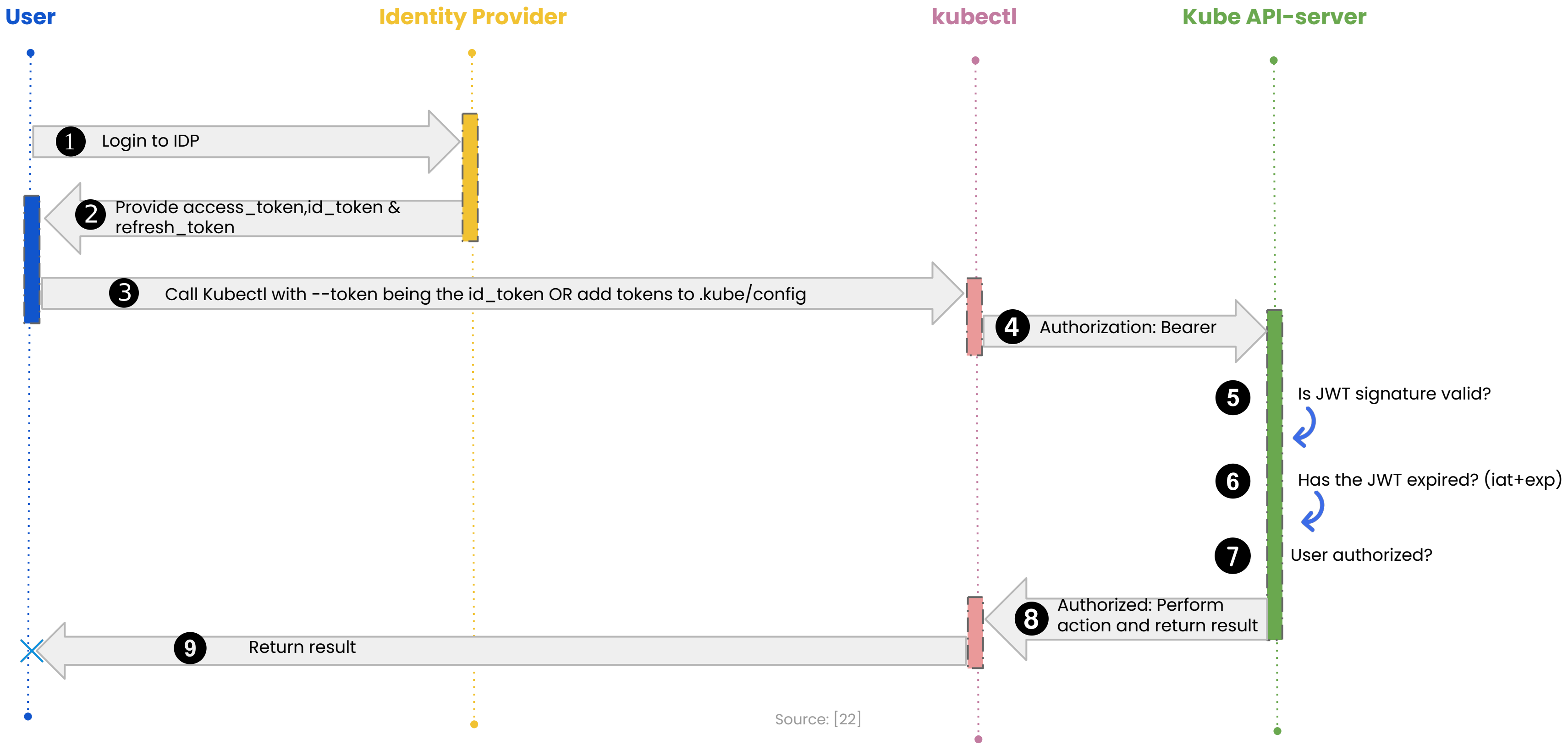
3 Return authorization code

4 Retrieve tokens using authorization code

5 Retrieve user info using access tokens

Source: [23]

02 Introduction | Authentication workflow of OIDC in kubernetes


User
Identity Provider
kubectl
Kube API-server
1 Login to IDP
2 Provide access_token,id_token & refresh_token
3 Call Kubectl with --token being the id_token OR add tokens to .kube/config
4 Authorization: Bearer
5 Is JWT signature valid?
6 Has the JWT expired? (iat+exp)
7 User authorized?
8 Authorized: Perform action and return result
9 Return result

Source: [22]

03 Summary

- With the increased used of containerized platforms, implementing security measures at multiple layers is extremely crucial

- **Possibility of query:**

Can't it serve my security purpose?

NO

Limitations:

- No security for unmanaged resources
- Not able to protect the system against application logic issues

Eg :

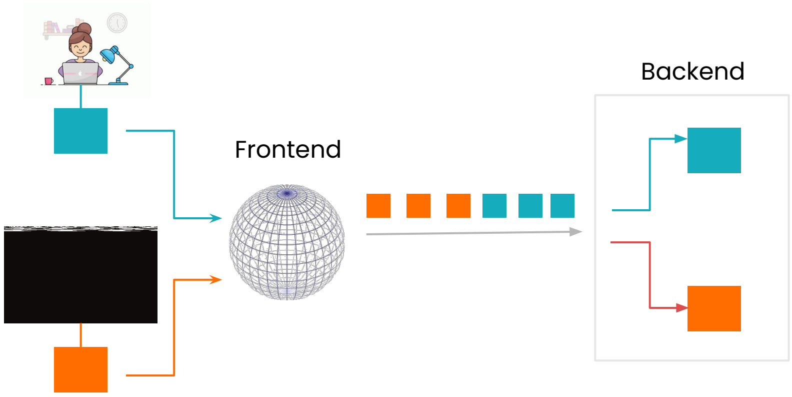
- Vulnerabilities in session maintenance.
- Improper configuration

- HTTP request smuggling

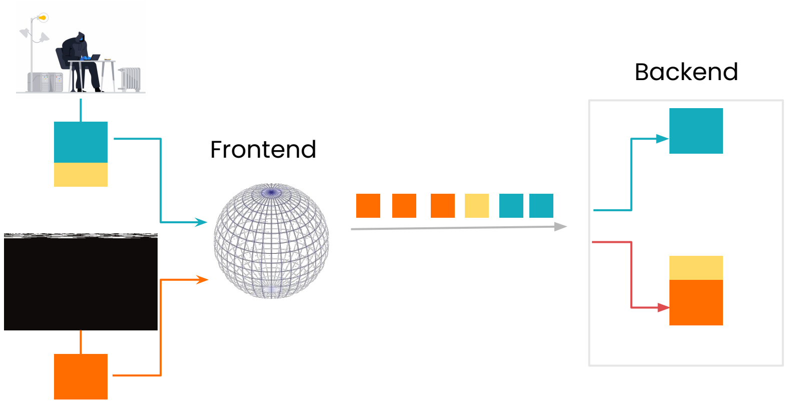
Source: [11], [12], [13]

03 Summary | What is HTTP request smuggling?

Normal HTTP request flow



HTTP request smuggling



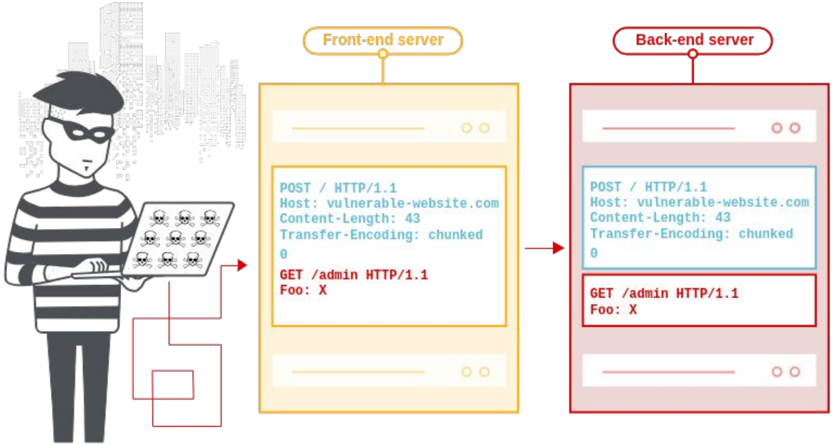
Source: [26]

03 Summary | How do vulnerabilities for this smuggling arise ?

Presence of message body in HTTP request

Content-length header (CL)

Transfer-Encoding header (TE)



```
POST / HTTP/1.1
Host: vulnerable-website.com
Content-Length: 43
Transfer-Encoding: chunked
0
GET /admin HTTP/1.1
Foo: X
```

```
POST / HTTP/1.1
Host: vulnerable-website.com
Content-Length: 43
Transfer-Encoding: chunked
0
```

```
GET /admin HTTP/1.1
Foo: X
```

Source: [26]

03 Summary

Research Question:

“How does using OpenID Connect in addition to reverse-proxy add more security to kube API server?”

- With OIDC in usage :

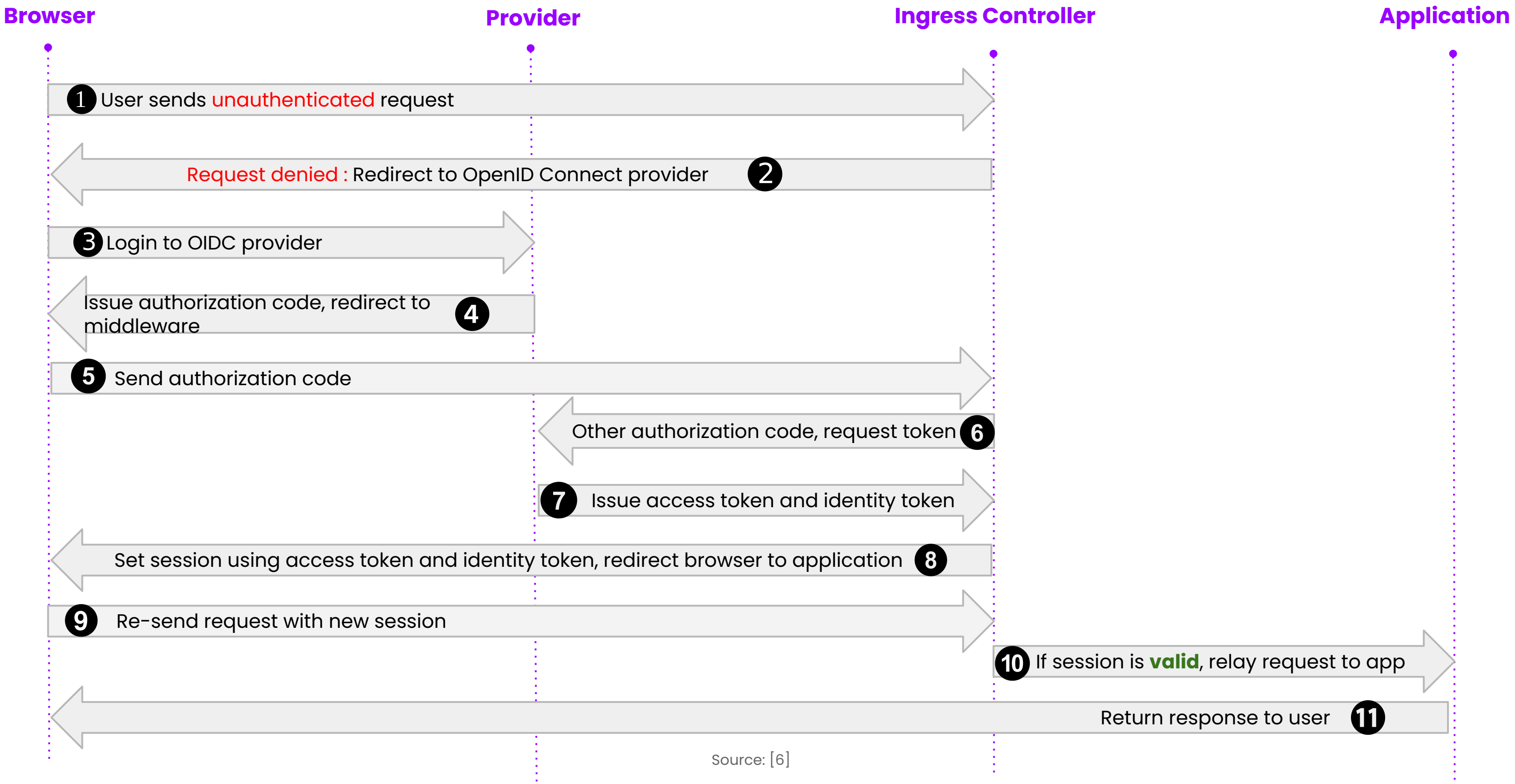
✓ Auto-rotated and easily accessible ID tokens compared to kubernetes secrets

✓ Fine granular authentication and authorization management

✓ Advanced management of HTTP traffic routing in comparison to ingress

✓ Authentication of credentials and authorization leads to decreased HTTP request smuggling

04 Result | Recommended implementation



Browser

Provider

Ingress Controller

Application

1 User sends unauthenticated request

2 Request denied : Redirect to OpenID Connect provider

3 Login to OIDC provider

4 Issue authorization code, redirect to middleware

5 Send authorization code

6 Other authorization code, request token

7 Issue access token and identity token

8 Set session using access token and identity token, redirect browser to application

9 Re-send request with new session

10 If session is **valid**, relay request to app

11 Return response to user

Source: [6]

05 State of art

1. Reverse proxy to authenticate to managed Kubernetes API servers via OIDC by jetstack.io
   (April 2, 2020)

2. Securing Kubernetes services with OAuth2/OIDC by Yussuf Burke, Developer at G-Research
   (January 12, 2021)

3. OpenID Connect Authentication for Kubernetes with Okta and NGINX Ingress Controller by Amir Rawdat of F5
   (September 22, 2021)

4. OpenID Connect: What Is It And How Does It Work? by Traefiklabs
   (No date)

Sources: [6], [7], [8], [9]

06 SOURCES (1/3)

[1] The Chief I/O (Publisher). (no date). Accessed on 23. December 2022
from
https://thechief.io/c/editorial/cryptojacking-attacks-kubernetes-serious-threat-deserves-attention/#:~:text=Cryptojacking%20is%20a%20form%20of,cryptocurrency%20without%20the%20user's%20awareness.
[2] Twain Taylor. (28.07.2020). Accessed on 24. December 2022
from https://techgenix.com/5-kubernetes-security-incidents/
[3] Wei Lien Dang. (02.06.2020). Accessed on 24. December 2022
from https://cloud.redhat.com/blog/cryptojacking-attacks-in-kubernetes-how-to-stop-them
[4] Weaveworks (Publisher). (03.07.2022). Accessed on 25. December 2022
from https://www.weave.works/blog/mitre-attack-matrix-for-kubernetes
[5] Andrew Martin (ControlPlane)(Author). (18.07.2018). Accessed on 26. December 2022
from https://kubernetes.io/blog/2018/07/18/11-ways-not-to-get-hacked/
[6] Traefiklabs (Publisher). (no date). Accessed on 26. December 2022
from https://traefik.io/glossary/openid-connect-everything-you-need-to-know/
[7] jetstack.io (Publisher). (02.04.2020). Accessed on 27. December 2022
from https://github.com/jetstack/kube-oidc-proxy
[8] Yussuf Burke (Author). (12.01.2021). Accessed on 27. December 2022
from https://www.gresearch.co.uk/blog/article/securing-kubernetes-services-with-oauth2-oidc/
[9] Amir Rawdat of F5 (Author). (22.09.2021). Accessed on 27 December 2022
from https://www.nginx.com/blog/implementing-openid-connect-authentication-kubernetes-okta-and-nginx-ingress-controller/
[10] Onelogin (Publisher). (no date). Accessed on 27 December 2022
from https://www.onelogin.com/learn/how-single-sign-on-works#:~:text=With%20SSO%2C%20meaning%20Single%20Sign,also%20called%20a%20login%20portal)

06 SOURCES (2/3)

[11] Zpedla (Publisher). (no date). Accessed on 28 December 2022
from https://www.zscaler.de/resources/security-terms-glossary/what-is-reverse-proxy
[12] Cem Dilmegani (Author). (14.11.2022). Accessed on 28 December 2022
from https://research.aimultiple.com/reverse-proxy/#:~:text=Though%20it%20provides%20security%2C%20there,run%20by%20a%20malicious%20party
[13] Traefiklabs (Publisher). (no date). Accessed on 28. December 2022
from https://traefik.io/blog/improve-application-security-using-a-reverse-proxy/
[14] Kubernetes (Publisher). (09.07.2022). Accessed on 29. December 2022
from https://kubernetes.io/docs/concepts/security/controlling-access/
[15] Alibaba Cloud Native Community (Publisher). (22.06.2020). Accessed on 29. December 2022
from https://www.alibabacloud.com/blog/getting-started-with-kubernetes-%7C-access-control-a-security-measure-in-kubernetes_596331
[16] Alicia Townsend (Author). (22.04.2021). Accessed on 29. December 2022
from https://www.onelogin.com/blog/real-difference-saml-oidc
[17] Virag Mody (Author). (06.08.2020). Accessed on 29. December 2022
from https://goteleport.com/blog/how-oidc-authentication-works/
[18] NHS Digital (Publisher). (17.08.2021). Accessed on 30. December 2022
from
https://digital.nhs.uk/services/identity-and-access-management/nhs-care-identity-service-2/care-identity-authentication/guidance-for-developers/openid-connect-overview
[19] Curity (Publisher). (no date). Accessed on 30. December 2022
from https://curity.io/resources/learn/openid-connect-overview/
[20] Dineshchandgr (Author). (08.07). Accessed on 30. December 2022
from https://medium.com/javarevisited/single-sign-on-sso-saml-oauth2-oidc-simplified-cf54b749ef39

06 SOURCES (3/3)

[21] Onelogin (Publisher). (no date). Accessed on 30. December 2022
from
https://www.onelogin.com/learn/why-sso-important#:~:text=Security%20and%20compliance%20benefits%20of%20SSO&text=SSO%20reduces%20the%20number%20of,%2C%20they%20usually%20don't.
[22] Kubernetes (Publisher). (no date). Accessed on 31. December 2022
from https://kubernetes.io/docs/reference/access-authn-authz/authentication
[23] NHS Digital (Publisher). (17.08.2021). Accessed on 31. December 2022
from
https://digital.nhs.uk/services/identity-and-access-management/nhs-care-identity-service-2/care-identity-authentication/guidance-for-developers/openid-connect-overview
[24] Judith Kahrer (Author). (02.08.2022). Accessed on 31. December 2022
from https://thenewstack.io/best-practices-for-api-security-in-kubernetes/
[25] Gilad David Maayan (Author). (21.04.2022). Accessed on 31. December 2022
from https://nordicapis.com/securing-the-kubernetes-api-server-critical-best-practices/
[26] PortSwigger (Publisher). (no date). Accessed on 06. January 2023
from https://portswigger.net/web-security/request-smuggling
[27] Daniel Lu (Author). (19.02.2021). Accessed on 08. January 2023
from https://www.okta.com/blog/2021/02/single-sign-on-sso/