First-Ever Attack Leveraging Kubernetes RBAC to Backdoor Clusters

By Michael Katchinskiy

Published: 2023-04-21 · Archived: 2026-04-06 01:09:37 UTC

We have recently discovered the first-ever evidence that attackers are exploiting Kubernetes (K8s) Role-Based Access Control (RBAC) in the wild to create backdoors. The attackers also deployed DaemonSets to take over and hijack resources of the K8s clusters they attack. Our research suggests that this campaign is actively targeting at least 60 clusters in the wild.

This blog post is a part of a comprehensive study we conducted on misconfigured K8s clusters in the wild. Our research findings are significant as they shed light on the risks of misconfigurations and how even large organizations can overlook the importance of securing their clusters, leaving them vulnerable to potential disasters with just one mistake.

The Attack

We have recorded and analyzed an attack on one of our K8s honeypots, which utilized the RBAC system to gain persistence. The attacker deployed containers using DaemonSets to run Monero cryptominers.

The initial access was gained via a misconfigured API server that allowed unauthenticated requests from anonymous users with privileges. The attacker sent a few HTTP requests to list secrets and then made two API requests to gain information about the cluster by listing the entities in the *'kube-system'* namespace. Next, the attacker checked if the attack had already been deployed on this specific cluster by looking for a deployment named *'kube-controller'*.

The attacker also attempted to delete some existing deployments in various namespaces, including *'kube-secure-fhgxtsjh'*, *'kube-secure-fhgxt'*, *'api-proxy'*, and *'worker-deployment'*. We assume that the attacker was disabling legacy campaigns or competitors' campaigns to increase available CPU and reduce the chances of being discovered if the server was exhausted.

The most interesting part of this attack was when the attacker used RBAC to gain persistence. The attacker created a new ClusterRole with near admin-level privileges. Since it was a cluster role, it was not bound to a specific namespace. Next, the attacker created a *'ServiceAccount'*, *'kube-controller'* in the *'kube-system'* namespace. Lastly, the attacker created a *'ClusterRoleBinding'*, binding the ClusterRole with the ServiceAccount to create a strong and inconspicuous persistence.

```
{
    "apiVersion": "rbac.authorization.k8s.io/v1",
    "kind": "ClusterRole",
    "metadata":
    {
        "annotations":
        {},
        "name": "system:controller:kube-
controller"
    },
    "rules":
    [
        {
            "apiGroups":
            [
                "*"
            ],
            "resources":
            [
                "*"
            ],
            "verbs":
            [
                "*"
            ]
        },
        {
            "nonResourceURLs":
            [
                "*"
            ],
            "verbs":
            [
                "*"
            ]
        }
    ]
```

```
}
```

Figure 1: ClusterRole created by the attacker with admin-like privileges

At this point, even if the *anonymous user access* is disabled, the attacker created persistence that allows further exploitation of the cluster. Additionally, while binding the *'cluster-admin'* role to a new or suspicious user may set off alarms, the attacker created a clever way to blend in nicely with the API audit logs. Eventually, by setting this legitimate-looking ClusterRoleBinding *'system:controller:kube-controller'*, the attacker could persist under the radar without setting off any alarms.

As part of our environment, we exposed AWS access keys in various locations on the cluster. Later that day, we received a beacon indicating that the access keys were used by the attacker to try and gain further access to the target's cloud service provider account and leverage the attack to steal more resources, data, and get out of the specific scope of the k8s cluster.

The attacker then created a DaemonSet to deploy containers on all nodes with a single API request. The DaemonSet creation request object contained the container image *'kuberntesio/kube-controller:1.0.1'*, hosted on the public registry Docker Hub. The impact on the cluster was resource hijacking.

```
{
    "name": "kube-controller",
    "namespace": "kube-system",
    "labels":
    {
        "app": "kube-controller"
    },
    "annotations":
    {
        "kubectl.kubernetes.io/last-applied-configuration":
        {
            "apiVersion": "apps/v1",
            "kind": "DaemonSet",
            "spec":
            {
                "containers":
                [
                    {
                        "image": "kuberntesio/kube-
controller:1.0.1",       "imagePullPolicy": "IfNotPresent",
                        "name": "kube-controller"
                    }
                ]
            }
        }
    }
}
```

Figure 2: Deploying a DaemonSet with the container image kuberntesio/kube-controller

When inspecting the container image ‘kuberntesio/kube-controller:1.0.1’ on Docker Hub, we found that the container was pulled 14,399 times since it was uploaded five months ago, indicating that this campaign is widespread. We found another 60 exposed K8s clusters that had evidence of active attacks by this attacker – this proves how large-scale the campaign is.

The container ‘kuberntesio/kube-controller’ has 3 tags, we analyzed all of them. Inside each of the container images we found the binary kube-controller (MD5=2833c82055bf2d29c65cd9cf6684449a), which was detected in VirusTotal as a cryptominer. We also found on each of the container images the configuration file. The wallet address indicated that the attacker had already mined 5 XMR, and at this rate of mining, they could gain another 5 per year ($200 USD) from a single worker.

The container image named ‘kuberntesio/kube-controller’ is a case of typosquatting that impersonates the legitimate ‘kubernetesio’ account. It has amassed millions of pulls, despite having only a few dozen container images. The image also mimics the popular ‘kube-controller-manager’ container image, which is a critical component of the control plane, running within a Pod on every master node, responsible for detecting and responding to node failures. Essentially, it is a widely used K8s component that should exist on the cluster and could deceive practitioners into thinking it is a legitimate deployment rather than a cryptominer. Since it is designed to run continuously, no one would question its presence.

Mapping the Attacks to the Microsoft threat matrix for Kubernetes

As a best practice, our team usually map the components of the attacks to the corresponding techniques of the MITRE ATT&CK framework. In this case, however, MITRE has yet to publish a framework for attacks against Kubernetes cluster. We utilized the threat matrix created by Microsoft, and added some new suggestions:

Initial Access	Execution	Persistence	Defense Escalation	Credential Access	Discovery	Impact
Exposed Sensitive Interfaces	Deploy Daemonset	Create ClusterRole Binding	Masquerading	List k8s Secrets	Accese the Kubernetes API Server	Resource Hijacking
		Create ClusterRole	Stop Competeing Campaigns	List k8s Configmaps		
			Images From a Public Registy	List Pods		

Summary and Mitigation

We are naming this attack RBAC Buster – a new K8s attack aimed to exploit K8s API servers to create a ClusterRoleBinding and gain full access to the cluster with persistence after the misconfiguration is fixed.

To help mitigate such attacks, you can use Aqua Trivy to secure your K8s clusters. Trivy can help to find vulnerabilities, exposed secrets and misconfigurations.

Additionally, Aqua's CNAPP is designed to detect such misconfigurations and protect your K8s clusters.

Michael is a former Security Researcher at Team Nautilus, Aqua's research team. His work focuses on researching and analyzing new attack vectors and threats in cloud native environments. When he isn't at work, he enjoys a good kite-surfing session or making Neapolitan pizza.

Assaf Morag

Assaf is the Director of Threat Intelligence at Aqua Nautilus. He is responsible of acquiring threat intelligence related to software development life cycle in cloud native environments, supports the team's data needs, and helps Aqua and the ecosystem remain at the forefront of emerging threats and protective methodologies. His research has been featured in leading information security publications and journals worldwide, and he has presented at leading cybersecurity conferences. Notably, Assaf has also contributed to the development of the new MITRE ATT&CK Container Framework.
Assaf is leading an O'Reilly course, focusing on cyber threat intelligence in cloud-native environments. The course covers both theoretical concepts and practical applications, providing valuable insights into the unique challenges and strategies associated with securing cloud-native infrastructures.
