LUNTRY

# KUBERNETES AUDIT LOG
# НА СТРАЖЕ БЕЗОПАСНОСТИ КЛАСТЕРОВ

**Дмитрий Евдокимов**

Founder&CTO Luntry

# Обо мне

> Я не верю в то, что систему можно сделать надежной и безопасной, не понимая того, как она устроена.

## Основатель и технический директор Luntry

Более 15 лет опыта в ИБ

## Специализация — безопасность контейнеров и Kubernetes

Автор ТГ-канала k8s(in)security

## Эксперт в сфере безопасности контейнерных сред

- Организатор конференции «БеКон» по БЕзопасности КОНтейнеров
- Бывший редактор рубрик в журнале «ХАКЕР», автор серии статей
- Автор курса «Cloud Native безопасность в Kubernetes»
- Член программного комитета CFP DevOpsConf и KUBER CONF

## Спикер

| | | | | |
|---|---|---|---|---|
| VK Kubernetes | Confidence | ZeroNights | БеКон | HITB |
| DevOpsConf | HackInParis | KuberConf | BlackHat | PHDays |
| Kazhackstan | HighLoad++ | OFFZONE | DevOops | SAS |

LUNTRY

# О компании Luntry

Luntry — это Комплексная Защита
на всем жизненном цикле
контейнерных приложений
и средств оркестрации на базе Kubernetes

Продукт в реестре Минцифры
https://reestr.digital.gov.ru/reestr/1057835/

Получение сертификата ФСТЭК планируется в четвертом квартале 2025 года.

LUNTRY.RU

# Функциональность Luntry

Контроль Kubernetes-ресурсов

Контроль состояния Kubernetes-кластеров

Контроль соответствия кластера стандартам


Управление уязвимостями образов и best practice

Сетевая безопасность

Анализ прав доступа

Защита Runtime

# Функциональность Luntry

Контроль Kubernetes-ресурсов

Контроль состояния Kubernetes-кластеров

Контроль соответствия кластера стандартам


Управление уязвимостями образов и best practice

Сетевая безопасность

Анализ прав доступа

Защита Runtime

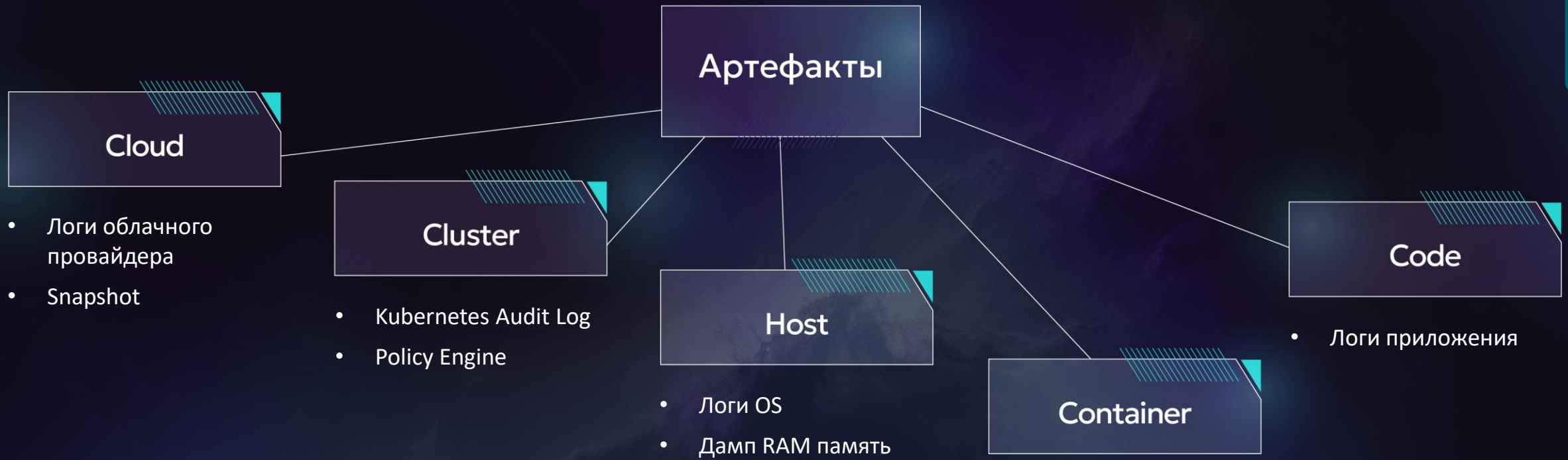
# План вебинара

LUNTRY

| | |
|---|---|
| 01 | Знакомство с Kubernetes Audit Log |
| 02 | Погружение в политику аудита |
| 03 | Способы обнаружения злоумышленника по логам |
| 04 | Как Luntry помогает решить задачи, связанные с Kubernetes Audit Log |
| 05 | Выводы |


# ЗНАКОМСТВО
# С KUBERNETES AUDIT LOG

# Источники данных в K8s окружениях




- Логи облачного провайдера
- Snapshot

- Kubernetes Audit Log
- Policy Engine

- Логи OS
- Дамп RAM память
- Дамп ФС (диск)

- Дамп RAM память
- Дамп ФС
- Runtime события

- Логи приложения

LUNTRY

# Подсистема аудита

## Auditing

Kubernetes *auditing* provides a security-relevant, chronological set of records documenting the sequence of actions in a cluster. The cluster audits the activities generated by users, by applications that use the Kubernetes API, and by the control plane itself.

Auditing allows cluster administrators to answer the following questions:

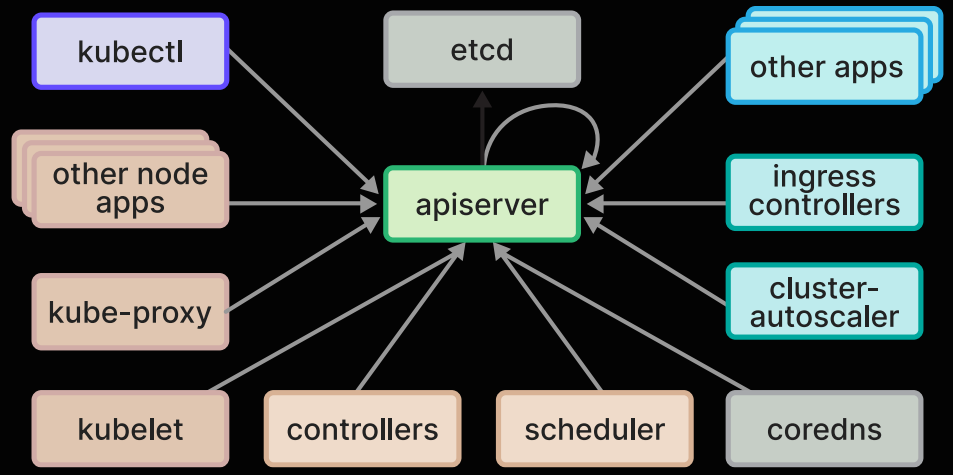
- what happened?
- when did it happen?
- who initiated it?
- on what did it happen?
- where was it observed?
- from where was it initiated?
- to where was it going?

Audit records begin their lifecycle inside the kube-apiserver component. Each request on each stage of its execution generates an audit event, which is then pre-processed according to a certain policy and written to a backend. The policy determines what's recorded and the backends persist the records. The current backend implementations include logs files and webhooks.

LUNTRY

# База

- По умолчанию выключен
- Нужно включать на всех Master Nodes в Control Plane
- Очень много параметров настройки
- Изменение параметров требует перезапуск API-сервера
- Дает дополнительную нагрузку на систему
- Требует определения Audit Policy
- Смотрит за всеми* операциями с YAML файлами




> The audit logging feature increases the memory consumption of the API server because some context required for auditing is stored for each request. Memory consumption depends on the audit logging configuration.

LUNTRY

# Обработка Audit Log

Документация по системе аудита

## Log backend

The log backend writes audit events to a file in JSONlines format. You can configure the log audit backend using the following `kube-apiserver` flags:

- `--audit-log-path` specifies the log file path that log backend uses to write audit events. Not specifying this flag disables log backend. `-` means standard out
- `--audit-log-maxage` defined the maximum number of days to retain old audit log files
- `--audit-log-maxbackup` defines the maximum number of audit log files to retain
- `--audit-log-maxsize` defines the maximum size in megabytes of the audit log file before it gets rotated

## Webhook backend

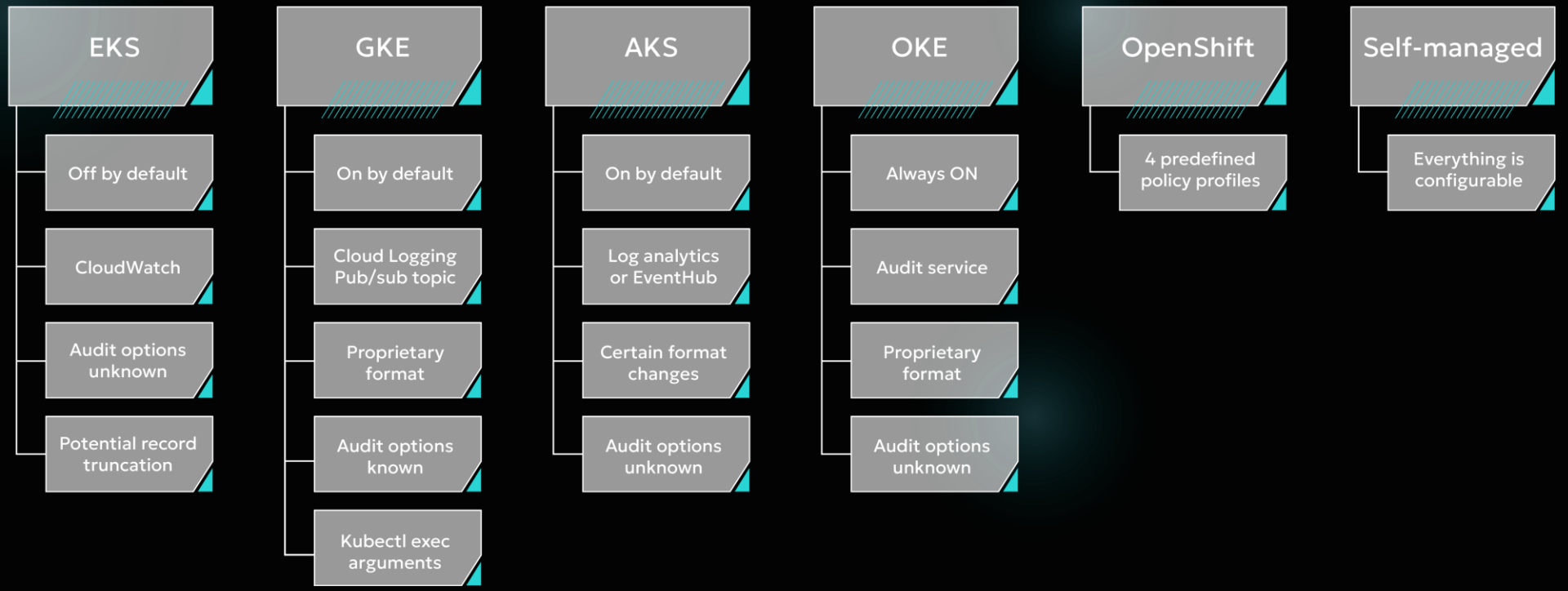
The webhook audit backend sends audit events to a remote web API, which is assumed to be a form of the Kubernetes API, including means of authentication. You can configure a webhook audit backend using the following kube-apiserver flags:

- `--audit-webhook-config-file` specifies the path to a file with a webhook configuration. The webhook configuration is effectively a specialized kubeconfig.
- `--audit-webhook-initial-backoff` specifies the amount of time to wait after the first failed request before retrying. Subsequent requests are retried with exponential backoff.

The webhook config file uses the kubeconfig format to specify the remote address of the service and credentials used to connect to it.

# Ситуация в Managed Kubernetes

- Нельзя контролировать конфигурацию механизма Audit Log
- Нельзя влиять на детализацию Audit Policy
- Ситуация может варьироваться от провайдера к провайдеру



# Лучшие практики

CIS Kubernetes Benchmark

NSA/CISA Kubernetes Hardening Guide

Kubernetes Security Technical Implementation Guide (STIG)

PCI Security Standards Council: Guidance for Containers and Container Orchestration Tools

NIST Special Publication 800-190 “Application Container Security Guide“

Приказ ФСТЭК России №118. Требования по безопасности информации к средствам контейнеризации

*3.2.1 Ensure that a minimal audit policy is created (Manual)*

**Profile Applicability:**

- Level 1 - Master Node

**Description:**

Kubernetes can audit the details of requests made to the API server. The `--audit-policy-file` flag must be set for this logging to be enabled.

| **7. Container Orchestration Tool Auditing** | |
|---|---|
| **7.1** Existing inventory management and logging solutions may not suffice due to the ephemeral nature of containers and container orchestration tools integration. | **a.** Access to the orchestration system API(s) should be audited and monitored for indications of unauthorized access. Audit logs should be securely stored on a centralized system. |

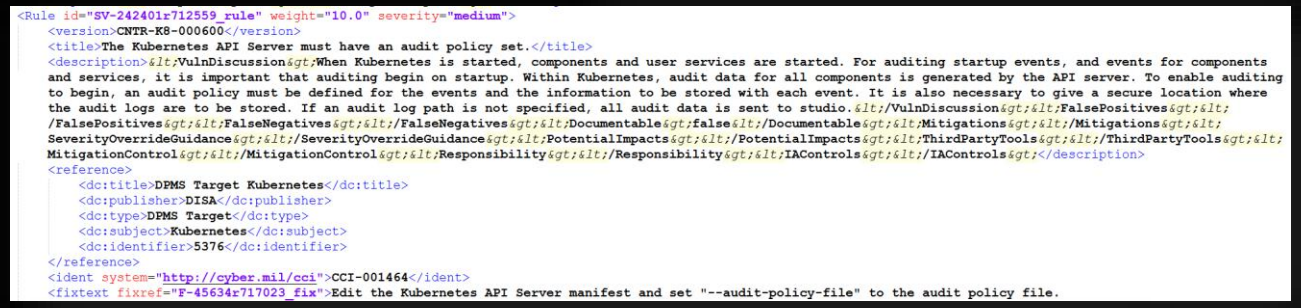
National Security Agency | Cybersecurity and Infrastructure Security Agency | Kubernetes Hardening Guidance

Another precaution to take when utilizing an external log server is to configure the log forwarder within Kubernetes with append-only access to the external storage. This protects the externally stored logs from being deleted or overwritten from within the cluster.

**Kubernetes native audit logging configuration**

The `kube-apiserver` resides on the Kubernetes control plane and acts as the front end, handling internal and external requests for a cluster. Each request, whether generated by a user, an application, or the control plane, produces an audit event at each stage in its execution. When an audit event registers, the `kube-apiserver` checks for an audit policy file and applicable rule. If such a rule exists, the server logs the event at the level defined by the first matched rule. Kubernetes' built-in audit logging capabilities perform no logging by default.

**Kubernetes audit logging capabilities are disabled by default**




- **Protect: Protective Technology**
  - PR.PT-1: Audit/log records are determined, documented, implemented, and reviewed in accordance with policy

регистрации событий безопасности в средстве контейнеризации;

# Kubernetes Audit Log против Dynamic Admission Webhook

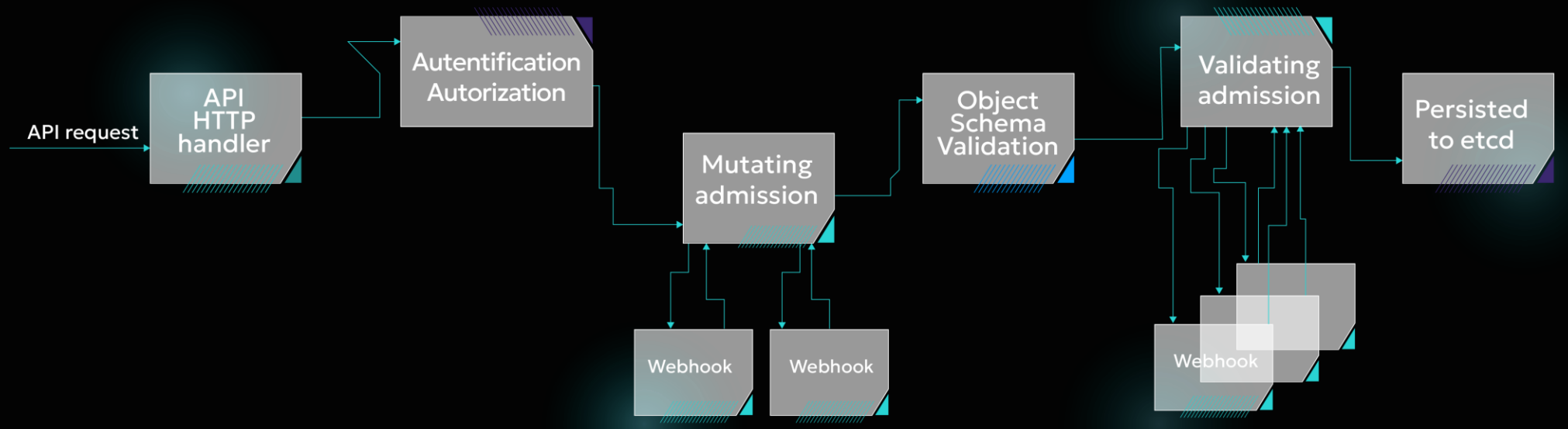
На базе Dynamic Admission Webhook создаются Policy Engine

**+ Плюсы:**

- Можно получать информацию о взаимодействии с Kubernetes API без настройки Control Plane
- Фиксированный формат данных
- Отвязка от облачного провайдера

**— Минусы:**

- Не фиксирует операции чтения (get/list/watch)
- Работает только с аутентифицированными и авторизированными событиями
- Не смотрит за ресурсами Events,ValidatingWebhookConfiguration, MutatingWebhookConfiguration
- Не знает ничего про impersonation




LUNTRY

LUNTRY

# ПОГРУЖЕНИЕ
# В ПОЛИТИКУ АУДИТА

# Политика

- Audit Policy - audit.k8s.io/v1 "unpublished" API
- Флаг --audit-policy-file=
- PolicyRule сопоставляет запросы к Kubernetes API на основе метаданных с уровнем аудита.
  Запросы должны соответствовать правилам в каждом поле.
- Примеры политик:
  - Google Cloud
  - Alibaba Cloud
  - Yandex Cloud

| Field | Description |
|---|---|
| `level` **[Required]** `Level` | The Level that requests matching this rule are recorded at. |
| `users` `[]string` | The users (by authenticated user name) this rule applies to. An empty list implies every user. |
| `userGroups` `[]string` | The user groups this rule applies to. A user is considered matching if it is a member of any of the UserGroups. An empty list implies every user group. |
| `verbs` `[]string` | The verbs that match this rule. An empty list implies every verb. |
| `resources` `[]GroupResources` | Resources that this rule matches. An empty list implies all kinds in all API groups. |
| `namespaces` `[]string` | Namespaces that this rule matches. The empty string "" matches non-namespaced resources. An empty list implies every namespace. |
| `nonResourceURLs` `[]string` | NonResourceURLs is a set of URL paths that should be audited. `*` s are allowed, but only as the full, final step in the path. Examples:<br>• `/metrics` - Log requests for apiserver metrics<br>• `/healthz*` - Log all health checks |
| `omitStages` `[]Stage` | OmitStages is a list of stages for which no events are created. Note that this can also be specified policy wide in which case the union of both are omitted. An empty list means no restrictions will apply. |
| `omitManagedFields` `bool` | OmitManagedFields indicates whether to omit the managed fields of the request and response bodies from being written to the API audit log.<br>• a value of 'true' will drop the managed fields from the API audit log<br>• a value of 'false' indicates that the managed fileds should be included in the API audit log Note that the value, if specified, in this rule will override the global default If a value is not specified then the global default specified in |

# Принцип написания политики

Политику можно разбить на 3 блока:

1. Детализируем важное
2. Исключаем лишнее
3. Остальное в общем режиме

Логика: От частного к общему

```yaml
apiVersion: audit.k8s.io/v1
kind: Policy
omitStages:
  - "RequestReceived"
rules:
  - level: RequestResponse
    resources:
    - group: ""
      resources: ["pods"]
  ...
  - level: None
    users: ["system:kube-proxy"]
    verbs: ["watch"]
    resources:
      - group: ""
        resources: ["endpoints", "services", "services/status"]
  ...
  - level: Metadate
```

LUNTRY

# Сообщение (Event)

- Event — сообщение со всей информацией по аудиту
- Audit annotations — специализированные аннотации

| Field | Description |
|---|---|
| apiVersion<br>string | audit.k8s.io/v1 |
| kind<br>string | Event |
| level **[Required]**<br>Level | AuditLevel at which event was generated |
| auditID **[Required]**<br>k8s.io/apimachinery/pkg/types.UID | Unique audit ID, generated for each request. |
| stage **[Required]**<br>Stage | Stage of the request handling when this event instance was generated. |
| requestURI **[Required]**<br>string | RequestURI is the request URI as sent by the client to a server. |
| verb **[Required]**<br>string | Verb is the kubernetes verb associated with the request. For non-resource requests, this is the lower-cased HTTP method. |
| user **[Required]**<br>authentication/v1.UserInfo | Authenticated user information. |
| impersonatedUser<br>authentication/v1.UserInfo | Impersonated user information. |
| sourceIPs<br>[]string | Source IPs, from where the request originated and intermediate proxies. The source IPs are listed from (in order):<br>1. X-Forwarded-For request header IPs<br>2. X-Real-Ip header, if not present in the X-Forwarded-For list<br>3. The remote address for the connection, if it doesn't match the last IP in the list up to here (X-Forwarded-For or X-Real-Ip). Note: All but the last IP can be arbitrarily set by the client. |
| userAgent<br>string | UserAgent records the user agent string reported by the client. Note that the UserAgent is provided by the client, and must not be trusted. |
| objectRef<br>ObjectReference | Object reference this request is targeted at. Does not apply for List-type requests, or non-resource requests. |
| responseStatus<br>meta/v1.Status | The response status, populated even when the ResponseObject is not a Status type. For successful responses, this will only include the Code and StatusSuccess. For non-status type error responses, this will be auto-populated with the error Message. |
| requestObject<br>k8s.io/apimachinery/pkg/runtime.Unknown | API object from the request, in JSON format. The RequestObject is recorded as-is in the request (possibly re-encoded as JSON), prior to version conversion, defaulting, admission or merging. It is an external versioned object type, and may not be a valid object on its own. Omitted for non-resource requests. Only logged at Request Level and higher. |
| responseObject<br>k8s.io/apimachinery/pkg/runtime.Unknown | API object returned in the response, in JSON. The ResponseObject is recorded after conversion to the external type, and serialized as JSON. Omitted for non-resource requests. Only logged at Response Level. |
| requestReceivedTimestamp<br>meta/v1.MicroTime | Time the request reached the apiserver. |
| stageTimestamp<br>meta/v1.MicroTime | Time the request reached current audit stage. |
| annotations<br>map[string]string | Annotations is an unstructured key value map stored with an audit event that may be set by plugins invoked in the request serving chain, including authentication, authorization and admission plugins. Note that these annotations are for the audit event, and do not correspond to the metadata.annotations of the submitted object. Keys should uniquely identify the informing component to avoid name collisions (e.g. podsecuritypolicy.admission.k8s.io/policy). Values should be short. Annotations are included in the Metadata level. |

LUNTRY

# Пример Event




```
{
  "kind": "Event",
  "apiVersion": "audit.k8s.io/v1",
  "level": "Metadata",
  "auditID": "5cf6b031-e590-4b3a-bfeb-f3c10a4254c1",
  "stage": "ResponseComplete",
  "requestURI": "/apis/rbac.authorization.k8s.io/v1/clusterroles?fieldManager=kubectl-create&fieldValidation=Strict",
  "verb": "create",
  "user": {
    "username": "kubernetes-admin",
    "groups": [
      "system:masters",
      "system:authenticated"
    ]
  },
  "sourceIPs": [
    "127.0.0.1",
    "192.168.53.9"
  ],
  "userAgent": "kubectl/v1.26.15 (darwin/arm64) kubernetes/1649f59",
  "objectRef": {
    "resource": "clusterroles",
    "name": "test-cluster-admin",
    "apiGroup": "rbac.authorization.k8s.io",
    "apiVersion": "v1"
  },
  "responseStatus": {
    "metadata": {},
    "code": 201
  },
  "requestReceivedTimestamp": "2025-10-29T08:58:12.597887Z",
  "stageTimestamp": "2025-10-29T08:58:12.610398Z",
  "annotations": {
    "authorization.k8s.io/decision": "allow",
    "authorization.k8s.io/reason": ""
  }
}
```

# Очень полезные аннотации

Работа с PSA:

- pod-security.kubernetes.io/exempt
- pod-security.kubernetes.io/enforce-policy
- pod-security.kubernetes.io/audit-violations

Работа с VAP

- validation.policy.admission.k8s.io/validation_failure

Вердикт работы механизма авторизации:

- authorization.k8s.io/decision
- authorization.k8s.io/reason

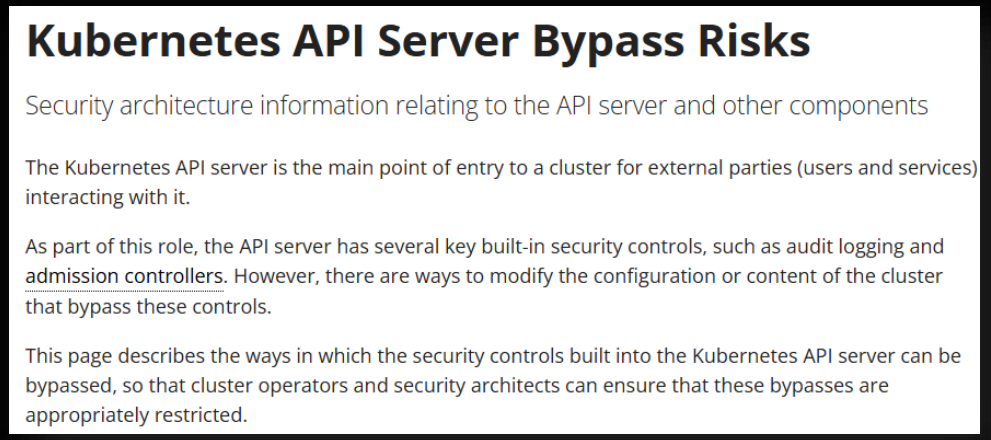
LUNTRY

# Слепые зоны и обходы Kubernetes Audit Log

**Что можно подделать: auditID, sourceIPs, userAgent**

**Слепые зоны:**

- **«Скрытые» StaticPod**
  - Создание Pod в несуществующем namespace с настройкой hostNetwork:true
  - Нужно отключать поддержку StaticPod на kubelet, где это не требуется
- **Интерактивные команды в kubectl exec**
  - Передача данных по стриминговому протоколу SPDY/3.1 или WebSocket (зависит от версии)
  - Атакующий может использовать kubectl-execws
  - Нужен Privileged Access Management (PAM)
- **Взаимодействие с kubelet API через ресурс node/proxy**
  - Работа с контейнерами на Node
  - Нужно следовать принципу наименьших привилегий в RBAC, настройки безопасности для kubelet, нужен Container Runtime Security
- **Прямое взаимодействие с etcd**
  - Утилиты etcdctl, kubetcd
  - Никто не должен находиться на Node с etcd
- **Прямое взаимодействие с Container Runtime Socket**
  - Утилиты типа crictl, nerdctl и т.д.
  - Нужно ограничить доступ пользователей на Nodes



## Kubernetes API Server Bypass Risks

Security architecture information relating to the API server and other components

The Kubernetes API server is the main point of entry to a cluster for external parties (users and services) interacting with it.

As part of this role, the API server has several key built-in security controls, such as audit logging and admission controllers. However, there are ways to modify the configuration or content of the cluster that bypass these controls.

This page describes the ways in which the security controls built into the Kubernetes API server can be bypassed, so that cluster operators and security architects can ensure that these bypasses are appropriately restricted.

Источник

LUNTRY

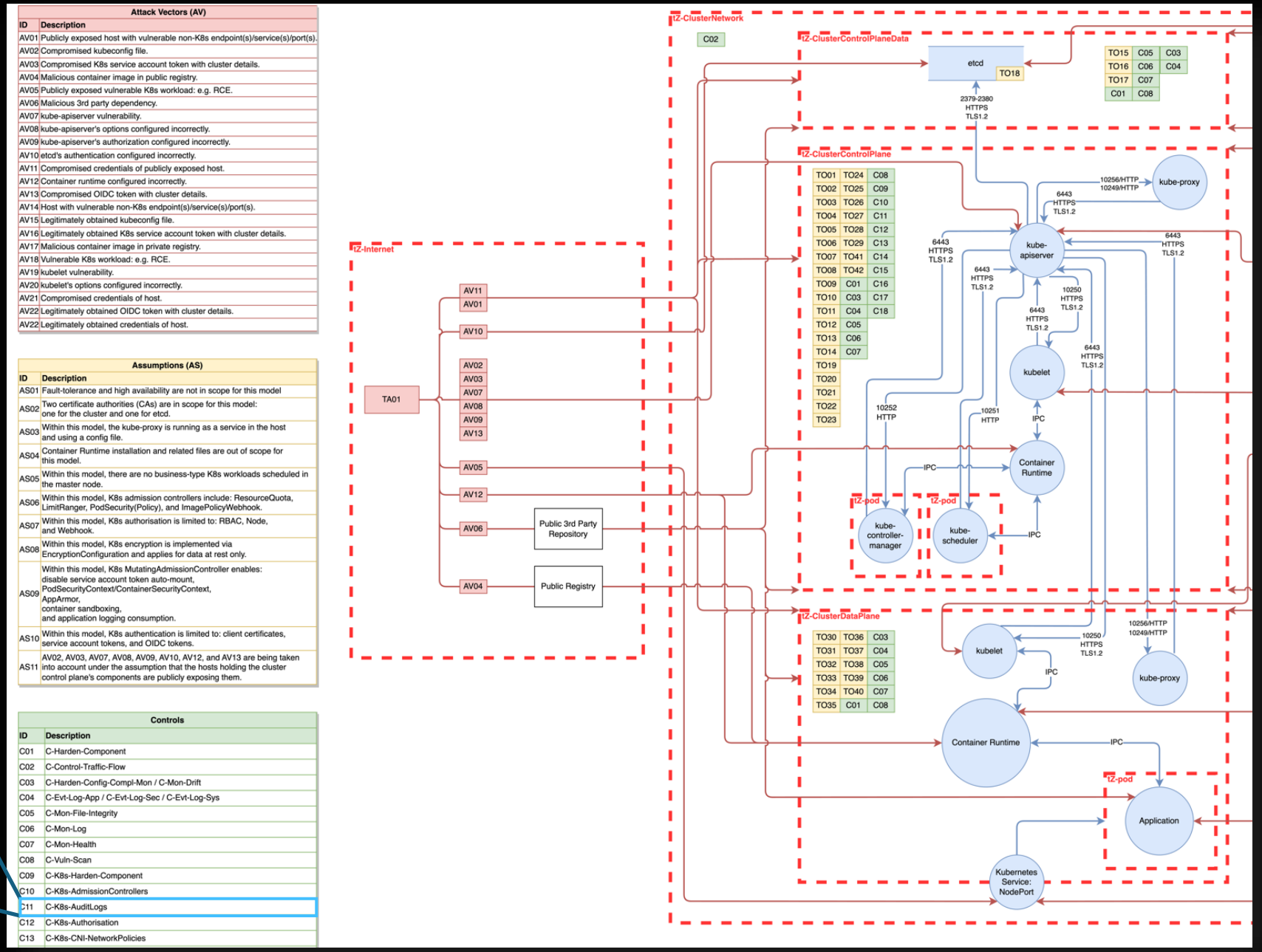
# Покрытие

Источник
[kubernetes-for-soc]

| Attack Vectors (AV) | |
|---|---|
| ID | Description |
| AV01 | Publicly exposed host with vulnerable non-K8s endpoint(s)/service(s)/port(s). |
| AV02 | Compromised kubeconfig file. |
| AV03 | Compromised K8s service account token with cluster details. |
| AV04 | Malicious container image in public registry. |
| AV05 | Publicly exposed vulnerable K8s workload: e.g. RCE. |
| AV06 | Malicious 3rd party dependency. |
| AV07 | kube-apiserver vulnerability. |
| AV08 | kube-apiserver's options configured incorrectly. |
| AV09 | kube-apiserver's authorization configured incorrectly. |
| AV10 | etcd's authentication configured incorrectly. |
| AV11 | Compromised credentials of publicly exposed host. |
| AV12 | Container runtime configured incorrectly. |
| AV13 | Compromised OIDC token with cluster details. |
| AV14 | Host with vulnerable non-K8s endpoint(s)/service(s)/port(s). |
| AV15 | Legitimately obtained kubeconfig file. |
| AV16 | Legitimately obtained K8s service account token with cluster details. |
| AV17 | Malicious container image in private registry. |
| AV18 | Vulnerable K8s workload: e.g. RCE. |
| AV19 | kubelet vulnerability. |
| AV20 | kubelet's options configured incorrectly. |
| AV21 | Compromised credentials of host. |
| AV22 | Legitimately obtained OIDC token with cluster details. |
| AV22 | Legitimately obtained credentials of host. |

| Assumptions (AS) | |
|---|---|
| ID | Description |
| AS01 | Fault-tolerance and high availability are not in scope for this model |
| AS02 | Two certificate authorities (CAs) are in scope for this model: one for the cluster and one for etcd. |
| AS03 | Within this model, the kube-proxy is running as a service in the host and using a config file. |
| AS04 | Container Runtime installation and related files are out of scope for this model. |
| AS05 | Within this model, there are no business-type K8s workloads scheduled in the master node. |
| AS06 | Within this model, K8s admission controllers include: ResourceQuota, LimitRanger, PodSecurity(Policy), and ImagePolicyWebhook. |
| AS07 | Within this model, K8s authorisation is limited to: RBAC, Node, and Webhook. |
| AS08 | Within this model, K8s encryption is implemented via EncryptionConfiguration and applies for data at rest only. |
| AS09 | Within this model, K8s MutatingAdmissionController enables: disable service account token auto-mount, PodSecurityContext/ContainerSecurityContext, AppArmor, container sandboxing, and application logging consumption. |
| AS10 | Within this model, K8s authentication is limited to: client certificates, service account tokens, and OIDC tokens. |
| AS11 | AV02, AV03, AV07, AV08, AV09, AV10, AV12, and AV13 are being taken into account under the assumption that the hosts holding the cluster control plane's components are publicly exposing them. |

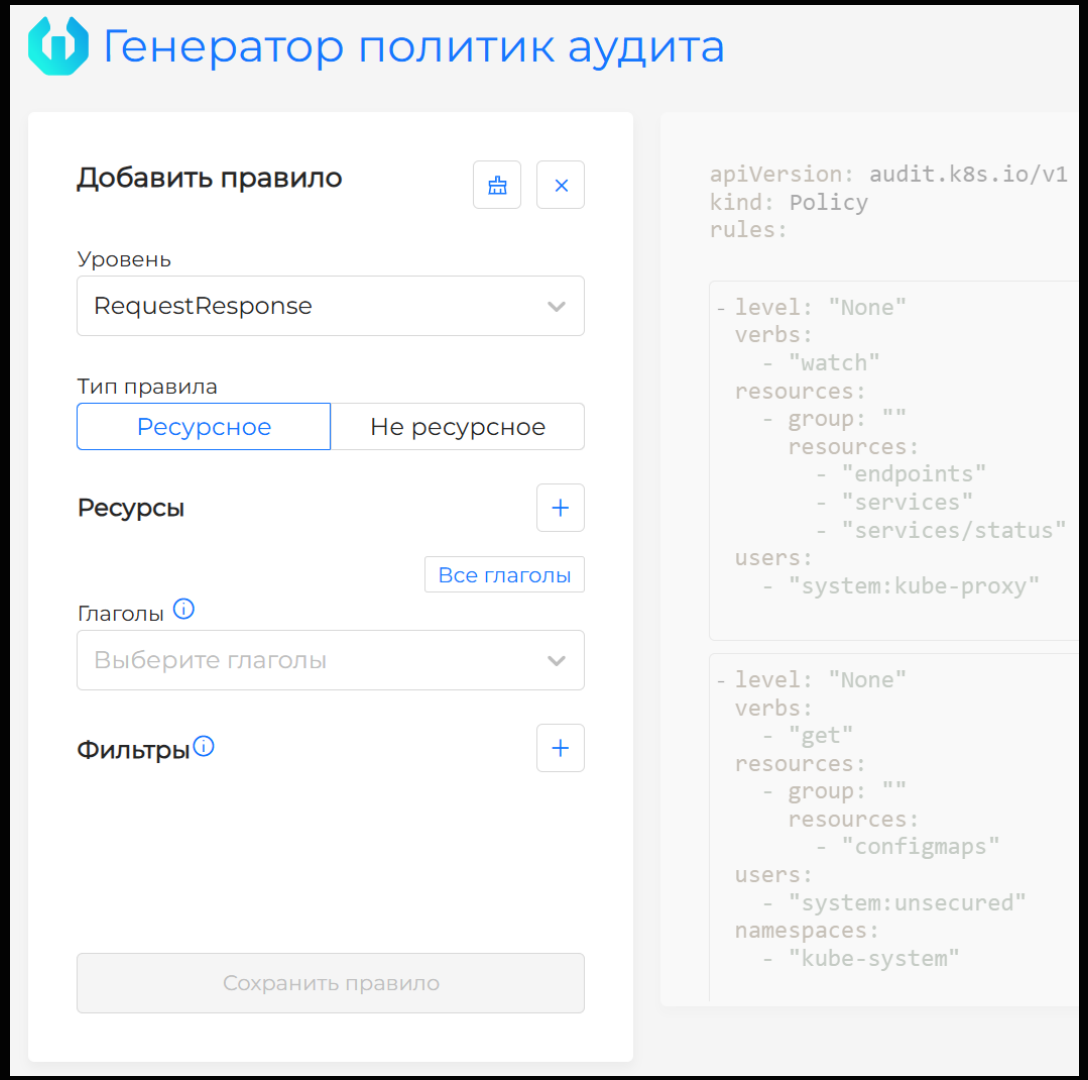
| Controls | |
|---|---|
| ID | Description |
| C01 | C-Harden-Component |
| C02 | C-Control-Traffic-Flow |
| C03 | C-Harden-Config-Compl-Mon / C-Mon-Drift |
| C04 | C-Evt-Log-App / C-Evt-Log-Sec / C-Evt-Log-Sys |
| C05 | C-Mon-File-Integrity |
| C06 | C-Mon-Log |
| C07 | C-Mon-Health |
| C08 | C-Vuln-Scan |
| C09 | C-K8s-Harden-Component |
| C10 | C-K8s-AdmissionControllers |
| C11 | C-K8s-AuditLogs |
| C12 | C-K8s-Authorisation |
| C13 | C-K8s-CNI-NetworkPolicies |




LUNTRY

# Open Source проекты

- Генератор политик аудита
  - https://policyeditor.shturval.tech
- Генерация RBAC из лога
  - audit2rbac
- Анализ лога
  - Falco Kubernetes Audit Events Plugin
  - Falco Rules




LUNTRY

LUNTRY

# СПОСОБЫ ОБНАРУЖЕНИЯ ЗЛОУМЫШЛЕННИКА ПО ЛОГАМ

# Kubernetes Audit Log и матрица угроз

"Threat Matrix for Kubernetes" от Microsoft

| 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 KBS secrets | Access Kubernetes API server | Access cloud resources | Images from a private registry | Data destruction |
| Compromised image in registry | bash/cmd inside container | Writable hostPath mount | Cluster admin binding | Delete KBS events | Mount service principal | Access Kubelet API | Container service account | Collecting data from pod | Resource hijacking |
| Kubeconfig file | New container | Kubernetes CronJob | hostPath mount | Pod / container name similarity | 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 configuration files | Exposed sensitive interfaces | Application credentials in configuration files | | |
| Exposed sensitive interfaces | SSH server running inside container | Container service account | | | Access managed identity credentials | Instance Metadata API | Writable hostPath mount | | |
| | Sidecar injection | Static pods | | | Malicious admission controller | | CoreDNS poisoning | | |

— то, что можно обнаружить в Kubernetes Audit Log

# Пример 1

- Pods без metadata.ownerReferences
- Продакшен нагрузка
  так не запускается
- Большинство примеров атак
  в интернете демонстрируется на
  примере Pod и часто копируется
  script kiddie

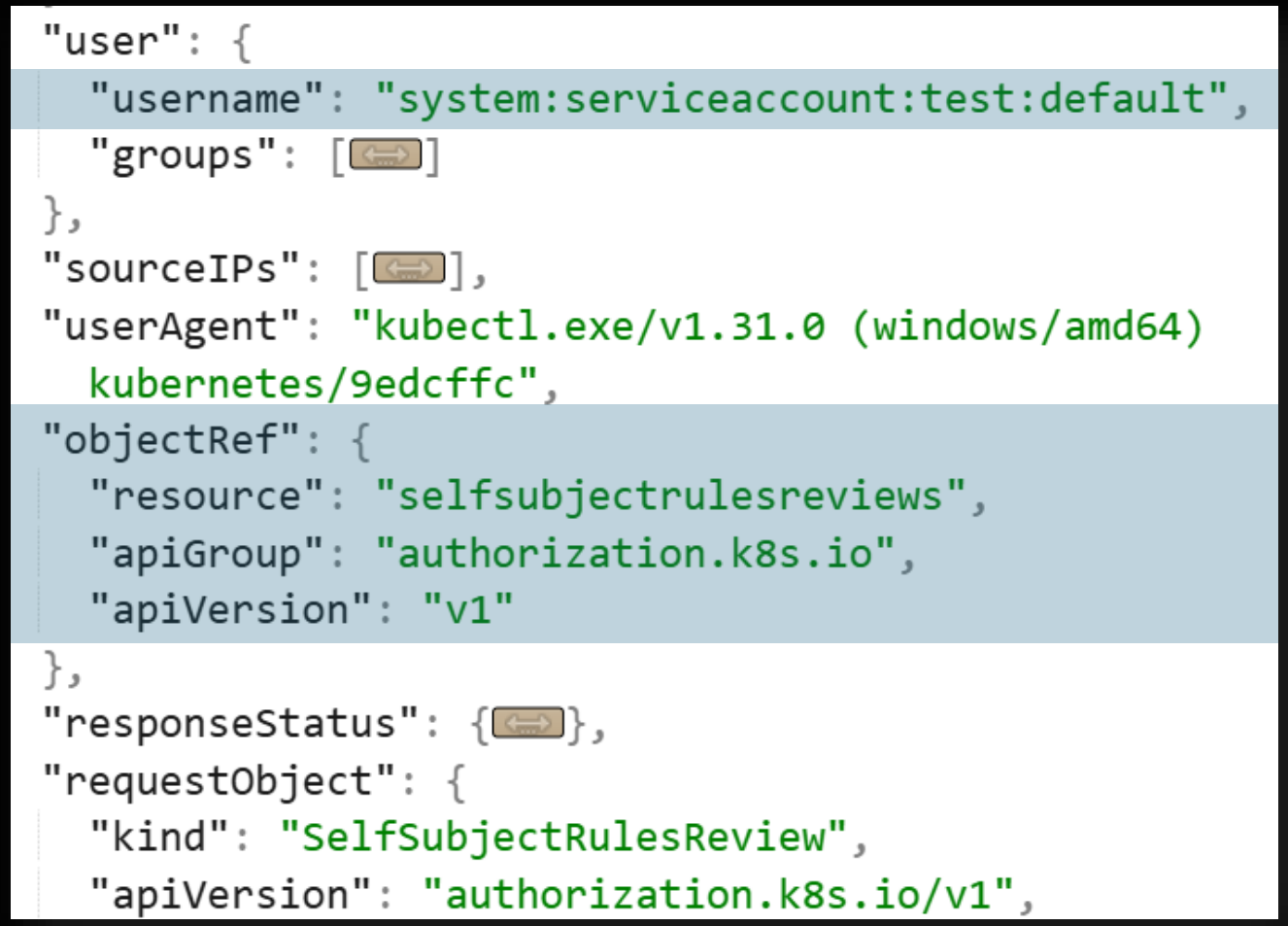


```
"requestObject": {
  "kind": "Pod",
  "apiVersion": "v1",
  "metadata": {
    "name": "priv-exec-pod",
    "namespace": "bank",
    "creationTimestamp": null,
    "labels": {
      "app": "pentest"
    }
  },
  "spec": {
    "containers": [
      {
        "name": "priv-pod",
        "image": "ubuntu",
        "command": [
          "/bin/sh",
          "-c",
          "--"
```

```
"requestObject": {
  "kind": "Pod",
  "apiVersion": "v1",
  "metadata": {
    "generateName": "nginx-deployment-6f87fb7ccf-",
    "creationTimestamp": null,
    "labels": {
      "app": "nginx",
      "pod-template-hash": "6f87fb7ccf"
    },
    "ownerReferences": [
      {
        "apiVersion": "apps/v1",
        "kind": "ReplicaSet",
        "name": "nginx-deployment-6f87fb7ccf",
        "uid": "70ea79a8-3c18-410d-95fc-000edcfed244",
        "controller": true,
        "blockOwnerDeletion": true
      }
    ]
  },
  "spec": {
    "containers": [
      {
        "name": "nginx",
        "image": "nginx:1.27.2",
```

# Пример 2

- Обращение к ресурсу SelfSubjectRulesReview от ServiceAccount микросервиса
- Сами микросервисы не узнают, какие у них права
  - Если только не началось восстание машин;)
- Атакующий внутри Pod может попытаться выяснить, какими правами обладает данный Service Account Token




LUNTRY

# Пример 3

- Отказы в выполнении действия
- Как правило не хватает прав
- Кто-то пытается сделать то, что ему нельзя


```
"requestReceivedTimestamp": "2024-10-30T11:19:19
  .911044Z",
"stageTimestamp": "2024-10-30T11:19:19.912790Z",
"annotations": {
  "authorization.k8s.io/decision": "forbid",
  "authorization.k8s.io/reason": "RBAC ..."
}
```


LUNTRY

# Пример 4

- Создание Bad Pods
- Частный случай — privileged pod
- Обычно требуется только для малого количества системных, инфраструктурных сервисов, которые можно задать как whitelist
- Атакующий может использовать для побега из контейнера

```
"requestObject": {
  "kind": "Pod",
  "apiVersion": "v1",
  "metadata": {⟺},
  "spec": {
    "containers": [
      {
        "name": "priv-pod",
        "image": "ubuntu",
        "command": [
          "/bin/sh",
          "-c",
          "--"
        ],
        "args": [
          "while true; do sleep 30; done;"
        ],
        "resources": {},
        "terminationMessagePath": "/dev/termination-log"
          ,
        "terminationMessagePolicy": "File",
        "imagePullPolicy": "Always",
        "securityContext": {
          "privileged": true
        }
```

LUNTRY

# Пример 5

- Обнаружение, выполнение команд и создание интерактивных сессий внутри контейнеров
- kubectl exec, kubectl debug, kubectl port-forward и т.д.
- Данная необходимость очень опасная и редкая, особенно для production окружения




```
"requestURI": "/api/v1/namespaces/bank/pods/priv-exec
  -pod/exec?command=%2Fbin%2Fbash&container=priv
  -pod&stdin=true&stdout=true&tty=true",
"verb": "create",
"user": {⟺},
"sourceIPs": [⟺],
"userAgent": "kubectl.exe/v1.31.0 (windows/amd64)
  kubernetes/9edcffc",
"objectRef": {
  "resource": "pods",
  "namespace": "bank",
  "name": "priv-exec-pod",
  "apiVersion": "v1",
  "subresource": "exec"
```

# Пример 6

- Обращение от “system:anonymous”, “system:unauthenticated”
- Данные субъекты должны быть отключены и не использоваться

```
"user": {
  "username": "system:anonymous",
  "groups": [
    "system:unauthenticated"
  ]
```

LUNTRY

# Пример 7

- Редактирование правил межсетевого экрана для получения дополнительных сетевых доступов
- Создание, редактирование, удаление NetworkPolicy
  - Native
  - Custom от Calico и Cilium
- Данная операция может производиться ограниченным кругом лиц
- Преимущество Policy-as-Code подхода




```
"verb": "create",
"user": {⟺},
"sourceIPs": [⟺],
"userAgent": "kubectl.exe/v1.31.0 (windows/amd64)
  kubernetes/9edcffc",
"objectRef": {
  "resource": "CiliumClusterwideNetworkPolicy",
  "name": "processing-np",
  "apiGroup": "cilium.io",
  "apiVersion": "v2"
},
```

# Пример 8

- Редактирование политик PolicyEngine или его отключение
- Создание, редактирование, удаление политик
  - OPA Gatekeeper, Kyverno, VAP и т.д.
- Данная операция может производиться ограниченным кругом лиц
- Возможно атакующий пытается отключить средство безопасности
- Преимущество Policy-as-Code подхода




```
"verb": "delete",
"user": {⟺},
"sourceIPs": [⟺],
"userAgent": "kubectl.exe/v1.31.0 (windows/amd64)
  kubernetes/9edcffc",
"objectRef": {
  "resource": "ClusterPolicy",
  "name": "drop-all-capabilities",
  "apiGroup": "kyverno.io",
  "apiVersion": "v1"
},
```

# Пример 9

- Учитываем бизнес логику работы
- Выкатки приложений только в рабочие часы по будням
- Остальное — это аномалии, возможно действия атакующего




```
"objectRef": {
  "resource": "deployment",
  "namespace": "bank",
  "name": "nginx-deployment",
  "apiVersion": "v1"
},
"responseStatus": {⟺},
"requestObject": {⟺},
"requestReceivedTimestamp": "2024-11-2T23:00:32.777737Z"
  ,
"stageTimestamp": "2024-11-2T23:00:32.840496Z",
```

# Пример 10

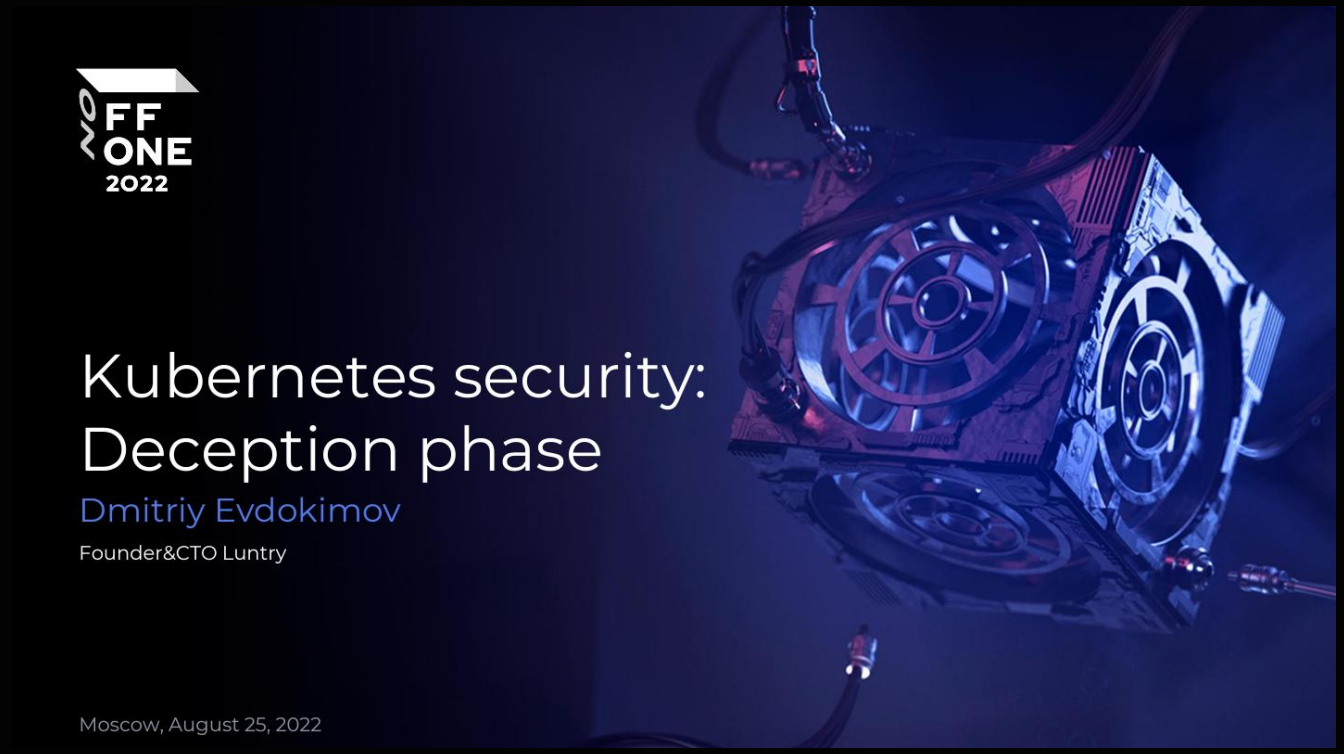
- С учетом роли кластера
- Например, в компании используется GitOps подход
  и все выкатывает только Flux
  или ArgoCD
- Выкатка от другой сущности —
  это аномалия

```
"user": {
  "username": "system:serviceaccount:argocd:argocd
    -application-controller",
```

LUNTRY

# Пример 11

- Используйте приманки и ловушки
  - “Detection through Deception”
  - “Security through Deception”
- Заготовленные ServiceAccount, Secret и т.д.
- Обнаруживаем их использование




Безопасность Kubernetes: Фаза Deception

LUNTRY


# КАК LUNTRY ПОМОГАЕТ РЕШИТЬ ЗАДАЧИ, СВЯЗАННЫЕ С KUBERNETES AUDIT LOG

# Сравнение с Falco

LUNTRY

| | Falco | Luntry |
|---|---|---|
| Реализация | C++, Go | Go |
| Работа с лог-файлами | Webhook backend | Webhook backend |
| Гибкость правил | Низкая<br>(только предопределённый набор полей) | Высокая<br>(произвольные поля, указывая json path) |
| Дефолтный набор правил | + | + |
| Синтаксис правил | Как и в других Falco правилах | Специально созданный на базе json path |
| Управление масштабированием | +/-<br>(реализовано через плагин) | +<br>(отдельный сервис) |

# Пример правила 1

```yaml
- name: "Scheduling workloads on master nodes"
  description: "Detected creation of workload with nodeSelector to place on master/control-plane nodes, which violates control plane
  isolation and can lead to cluster compromise"
  severity: "high"
  tags: []
  rule:
    filters:
      regexp:
        - ".objectRef.resource": "^(pods|deployments|daemonsets|statefulsets|jobs|cronjobs)$"
        - ".verb": "^create$"
      eq:
        - ".requestObject.spec.nodeSelector['node-role.kubernetes.io/master']": ""
          aggregation: or
        - ".requestObject.spec.nodeSelector['node-role.kubernetes.io/control-plane']": ""
          aggregation: or
```

# Пример правила 2

```
- rule:
    filters:
      regexp:
        - ".objectRef.resource": "^(pods|deployments|daemonsets|statefulsets|jobs|cronjobs)$"
        - ".verb": "^create$"
      # Host IPC/Network/PID
      eq:
        - ".requestObject.spec.template.spec.hostIPC": "true"
        - ".requestObject.spec.template.spec.hostNetwork": "true"
        - ".requestObject.spec.template.spec.hostPID": "true"
      # HostPath mount
      exists:
        - path: ".requestObject.spec.template.spec.volumes[].hostPath"
          aggregation: or
      # Privileged container
      eq:
        - ".requestObject.spec.template.spec.containers[].securityContext.privileged": "true"
          aggregation: or
      # Дополнительные Capabilities
      exists:
        - path: ".requestObject.spec.template.spec.containers[].securityContext.capabilities.add[]"
          aggregation: or
```

LUNTRY

# Live Demo Luntry

- Анализ Kubernetes Audit Log
- Встроенная библиотека правил (50+)
- Создание собственных правил
- Собственная Audit Policy с ориентиром на безопасность
- Высокая скорость применения правил
- Простая масштабируемость сервиса
- Снижение нагрузки и лицензии на SIEM ;)

Dashboard Rules Detects Library

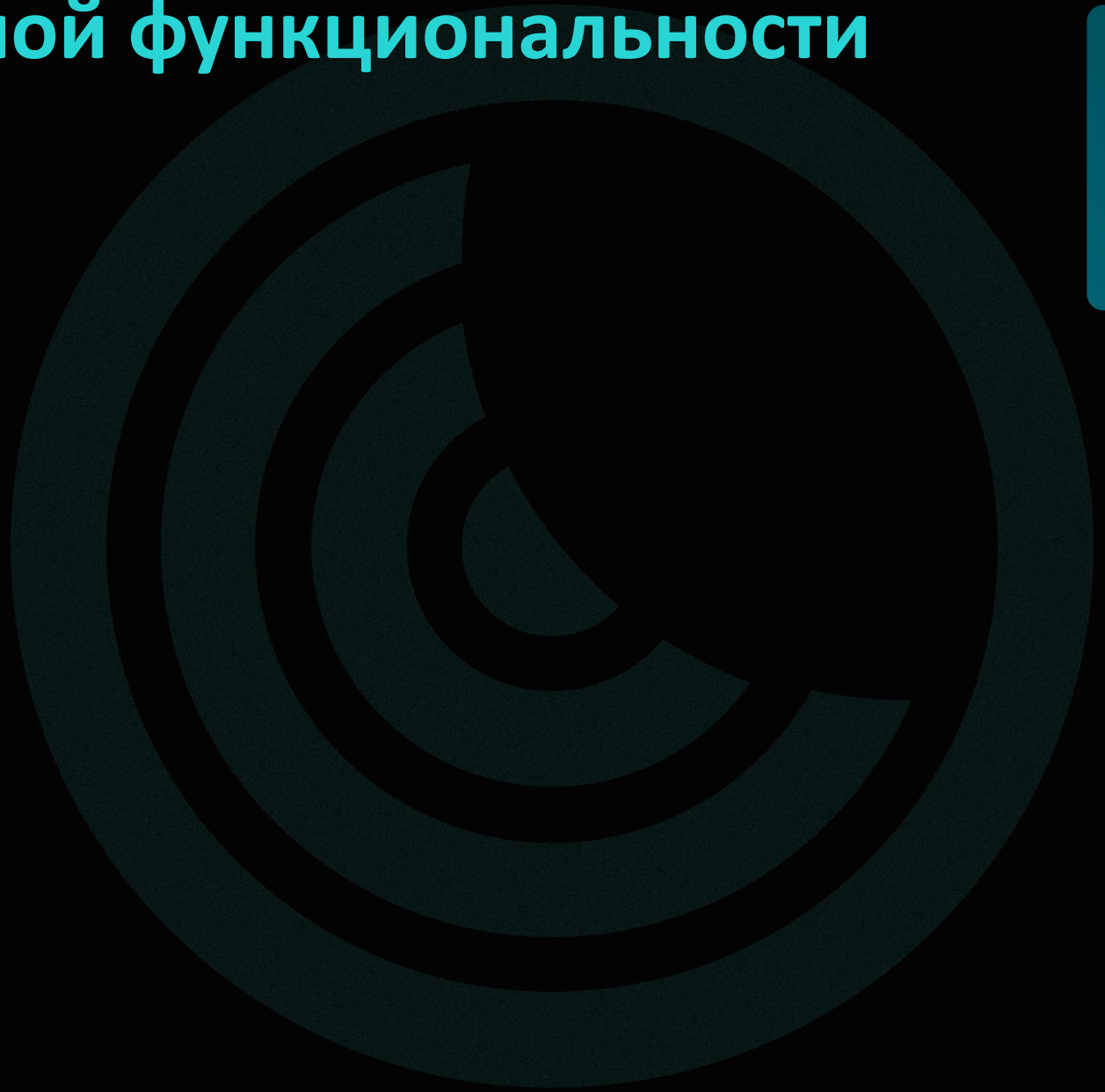
29.10.2025 20:52:00 - 29.10.2025 21:04:20 Severity 4 Rule Name 0 Resource type 0 User name 0 Verb 0 Namespace 0

| Time | Name | Severity | Namespace | Resource type | Resource name | Verb | Source IP | User name | User agent | Decision |
|---|---|---|---|---|---|---|---|---|---|---|
| 29.10.2025 21:04:11 | Unauthorized access to API server | medium | lol | namespaces | lol | create | 10.11.226.79 | system:anonymous | curl/7.68.0 | forbid |
| 29.10.2025 20:56:52 | Creation or modification of CronJob | medium | kube-system | cronjobs | | create | 10.11.226.79 | system:serviceaccount:pentest:mtkpi-sa | k/v1.31.0 (linux/amd64) kubernetes/9edcffc | forbid |
| 29.10.2025 20:53:19 | Creation of Network Policies | medium | default | networkpolicies | test-network-policy | patch | 192.168.53.15 | kubernetes-admin | node-fetch | allow |
| 29.10.2025 20:53:19 | Creation of Network Policies | medium | default | networkpolicies | test-network-policy | patch | 192.168.53.15 | kubernetes-admin | node-fetch | |
| 29.10.2025 20:53:01 | Modification of Kyverno security policies | medium | default | clusterpolicies | advanced-restrict-image-registries | update | 172.20.212.41 | system:serviceaccount:kyverno:kyverno-admission-controller | kyverno/v0.0.0 (linux/amd64) kubernetes/$Format | allow |

# RoadMap Luntry для данной функциональности

- Новый UI
- Обновление библиотеки правил
- Построение деревьев атак

Мы всегда рады вашей обратной связи
и features requests =)



LUNTRY


# ВЫВОДЫ

# Выводы

LUNTRY

01 Анализ  Kubernetes Audit Log обязателен

02 Внимательно подготавливайте Audit Policy

03 Создание правил детектирования очень креативная задача

04 Luntry полностью решает данный аспект работы с Audit Log

# Полезные ссылки

- "Вы еще не читаете Kubernetes Audit Log? Тогда мы идем к вам!", БеКон 2024
- "Kubernetes Audit Log в арсенале SOC", SOC Forum 2024
- "Безопасность контейнеров и Kubernetes для SOC", Вебинар Luntry
- "DeTT&CT(ing) Kubernetes ATT&CK(s) with Audit Logs", SANS 2021
- "Kubernetes Audit Log Gotchas", fwd:cloudsec Europe 2024
- "Экскурсия по матрицам угроз для контейнеров и Kubernetes", VK Kubernetes Conf 2023

LUNTRY

LUNTRY


luntry_official
luntrysolution
luntrysolution
luntry.ru
info@luntry.ru

**ДМИТРИЙ ЕВДОКИМОВ**
Founder & CTO Luntry

de@luntry.ru
Qu3b3c
k8security


# СПАСИБО ЗА ВНИМАНИЕ!