
black hat®
ARSENAL
APRIL 3-4, 2025
MARINA BAY SANDS / SINGAPORE
#BHAS @BlackHatEvents


black hat
ARSENAL
KubeAPI-Inspector: discover the secrets hidden in apis
Qiqi Xu
#BHAS @BlackHatEvents

About us

- Security researcher at the Dawn Security Lab of JD.com
- Dawn Security Lab established in March 2021, Dawn Security Lab focuses on cutting-edge offensive and defensive security research and product development.
- https://dawnslab.jd.com/

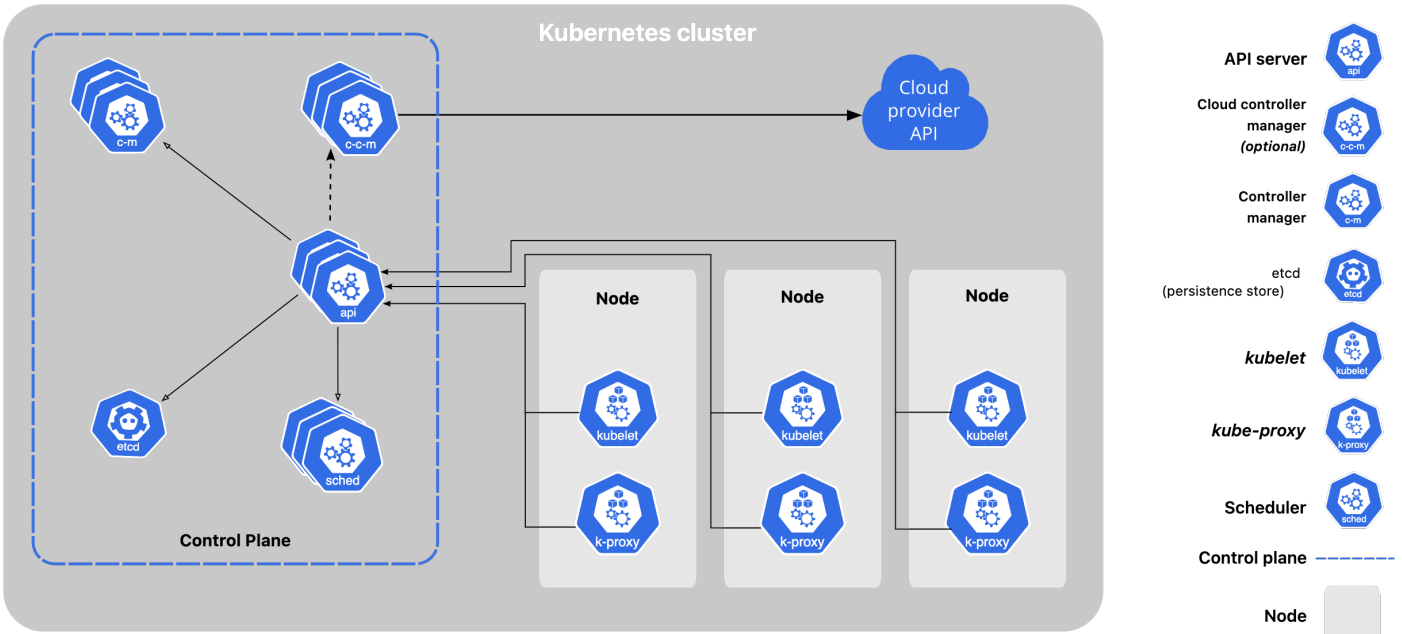
Agenda

- Introduction
- How the tool works
- Workshop in action & Demo
- Next & Future

Introduction

In Kubernetes, API is "everything"

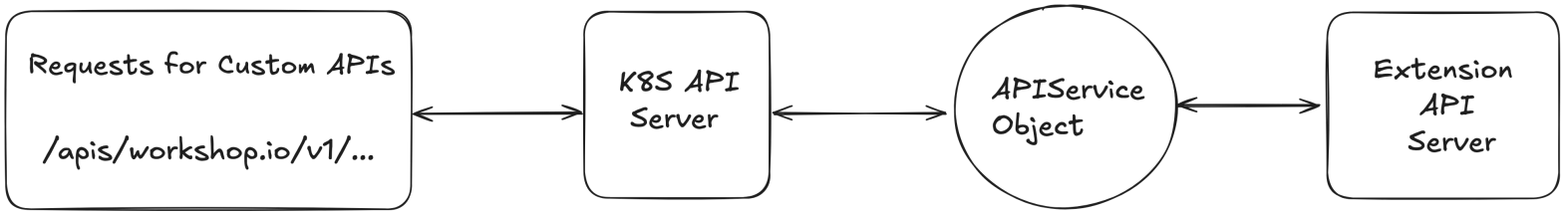
- External components and users all communicate with API serer
- Everyting operates(CRUD) via API
- Multiple scenarios: hybrid cloud, multi-tenancy, AI/ML



Ref: https://kubernetes.io/docs/concepts/overview/

The Extension API Server

- Extension API Server are a core extension capability of Kubernetes.
- Through Extension API Server, you can add functionality to Kubernetes.
- Independent API service



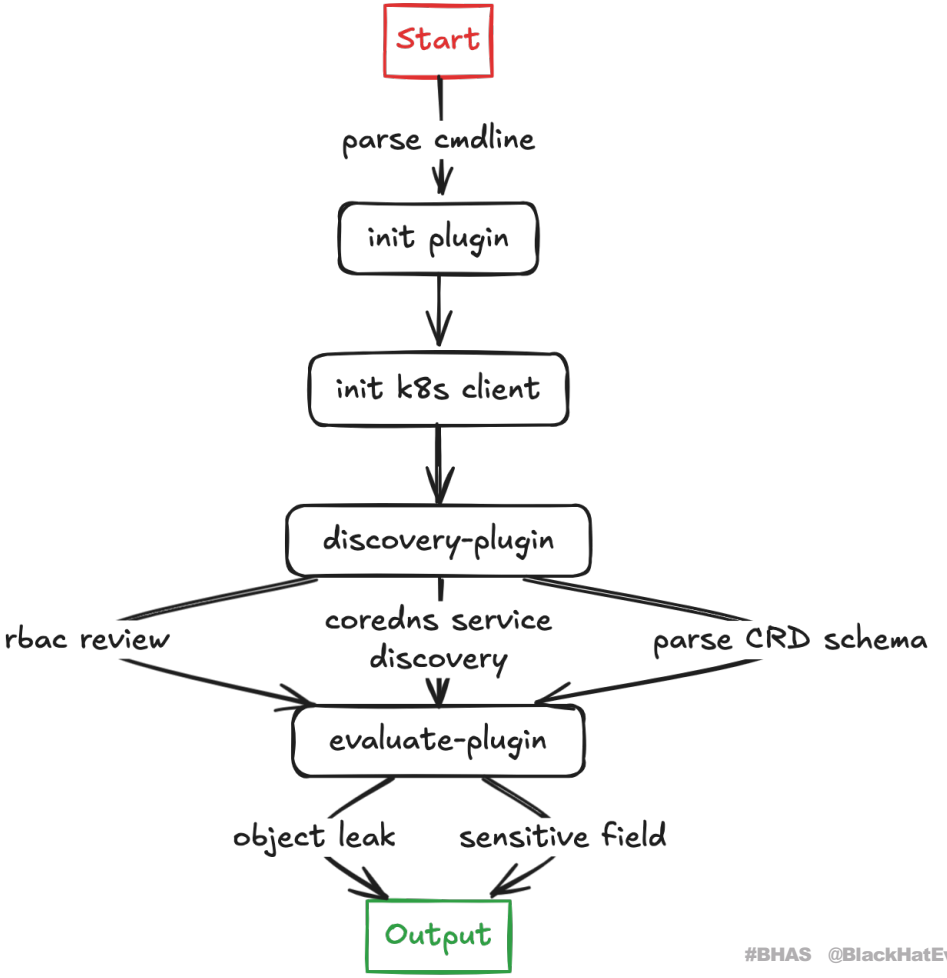
Bridging the Gap in K8S API Security

- Excellent open-source K8S security projects exist: CDK (penetration testing), Kubescape (misconfiguration scanning), Falco (runtime security), Cilium (network security)
- While these tools excel at container runtime and cluster baseline security, they offer limited Kubernetes API security analysis.
- We introduce kubeapi-inspector to fill this gap, focusing on identifying API-level vulnerabilities.

How the tool works

Feature

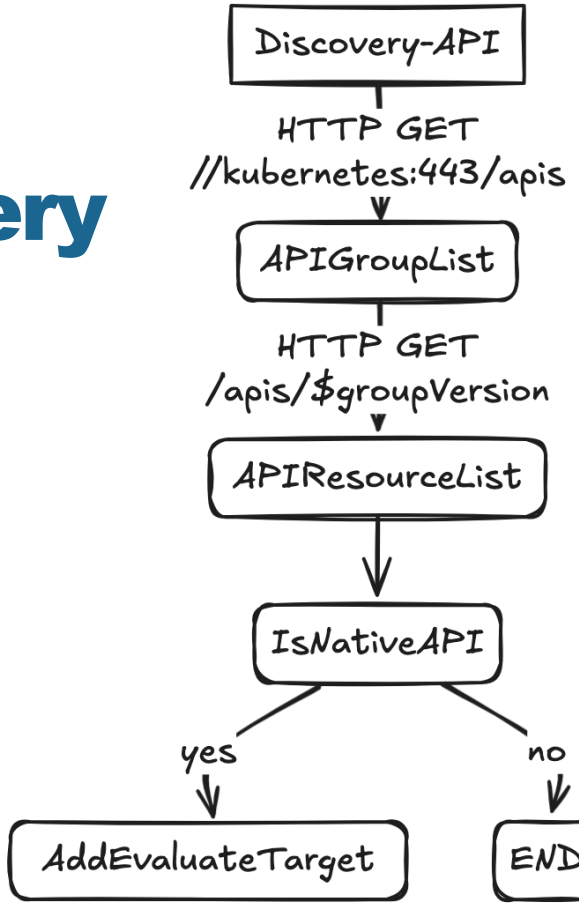
- Single Binary, Built with Golang
- Post-Exploitation tools
- Auto discover api token/address/endpoint
- Find object sensitive field
- api fuzzing



Automated Target Discovery

- Use in-cluster config (K8S auto-mounts default service account).
- Use Client-go Discovery API to find K8S endpoints.

```
→  ~ kubectl --as=bh --as-group=system:authenticated auth can-i --list
Resources                                       Non-Resource URLs   Resource Names   Verbs
selfsubjectaccessreviews.authorization.k8s.io   []                  []              [create]
selfsubjectrulesreviews.authorization.k8s.io    []                  []              [create]
                                                [/api/*]            []              [get]
                                                [/api]              []              [get]
                                                [/apis/*]           []              [get]
                                                [/apis]             []              [get]
                                                [/healthz]          []              [get]
                                                [/healthz]          []              [get]
                                                [/livez]            []              [get]
                                                [/livez]            []              [get]
                                                [/openapi/*]        []              [get]
                                                [/openapi]          []              [get]
                                                [/readyz]           []              [get]
                                                [/readyz]           []              [get]
                                                [/version/]         []              [get]
                                                [/version/]         []              [get]
                                                [/version]          []              [get]
                                                [/version]          []              [get]
```



Service discovery? Target discovery!

- Kubernetes uses CoreDNS as its default cluster DNS.
- CoreDNS versions before 1.9.0 have a Kubernetes plugin that can leak cluster info.
- This feature can help use to find other targets.

```
root@victim-db69bd6d7-7z8qm:/# dig srv *.*.*.svc.cluster.local

; <<>> DiG 9.18.30-0ubuntu0.24.04.2-Ubuntu <<>> srv *.*.*.svc.cluster.local
;; global options: +cmd
;; Got answer:
;; WARNING: .local is reserved for Multicast DNS
;; You are currently testing what happens when an mDNS query is leaked to DNS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56231
;; flags: qr aa rd; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 5
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 625f3e125a97288d (echoed)
;; QUESTION SECTION:
;*.*.*.svc.cluster.local.       IN      SRV

;; ANSWER SECTION:
*.*.*.svc.cluster.local. 22     IN      SRV     0 16 9153 172-17-0-2.kube-dns.kube-system.svc.cluster.local.
*.*.*.svc.cluster.local. 22     IN      SRV     0 16 53 172-17-0-2.kube-dns.kube-system.svc.cluster.local.
*.*.*.svc.cluster.local. 22     IN      SRV     0 16 8443 192-168-49-2.kubernetes.default.svc.cluster.local.
*.*.*.svc.cluster.local. 22     IN      SRV     0 16 80 172-17-0-4.nginx.default.svc.cluster.local.
*.*.*.svc.cluster.local. 22     IN      SRV     0 16 80 172-17-0-3.nginx.default.svc.cluster.local.
```

Information Disclosure Issue: Add feature to disable wildcard lookups #4984

Closed

micahhausler opened on Nov 16, 2021 · edited by micahhausler Edits

Opening this as a public issue, as there are already public discussions and integration into penetration testing tooling.

CoreDNS's wildcard feature (specifically wildcards in Kubernetes) present an information disclosure.

CoreDNS has several plugins that support wildcard lookups, including the `file`, `etcd`, and `kubernetes` plugins. In environments such as Kubernetes where Service Discovery is desired, full enumeration of Services, Namespaces, and Endpoint IPs is possible even when Kubernetes API permissions or authentication is denied.

Assignees

No one assigned

Labels

enhancement

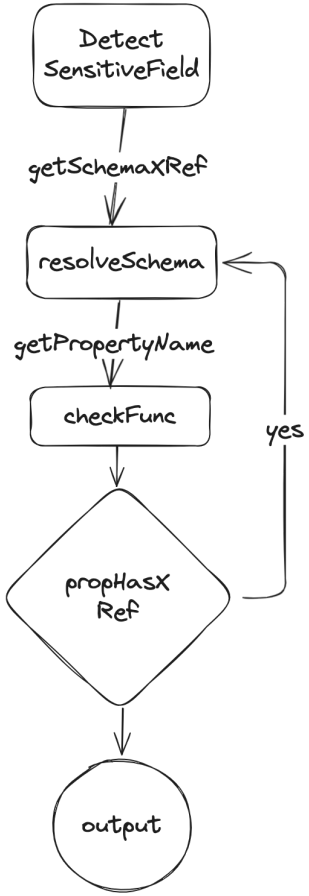
plugin/kubernete

Ref: https://github.com/coredns/coredns/issues/4984

Detecting Sensitive API Fields

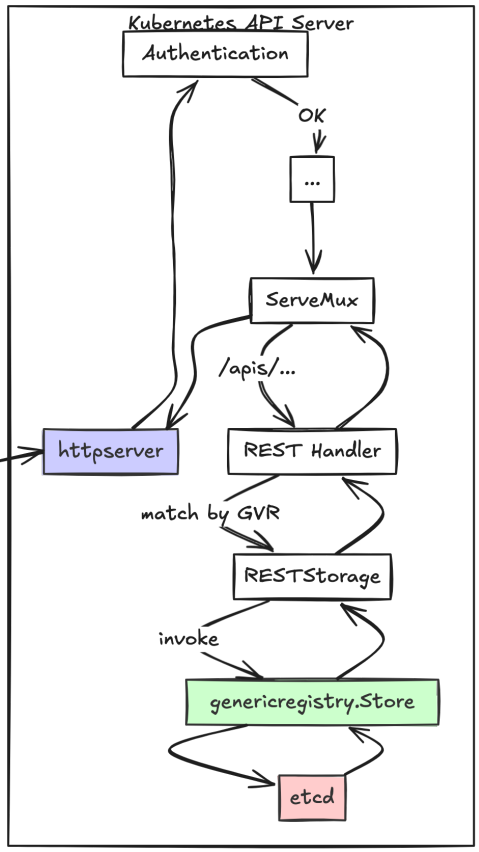
- Built-in sensitive field detection covers all endpoint versions (requires default permissions).
- Match sensitive field patterns by parsing OpenAPI schema

```
4  var sensitivePatterns = []string{
5      `(?i)password`, // match "password", not case-sensitive
6      `(?i)passwd`,
7      `(?i)pwd`,
8      `(?i)secret`,
9      `(?i)token`,
10     `(?i)api[_-]?key`,
11     `(?i)auth[_-]?token`,
12     `(?i)kubeconfig`,
```



Why fuzz the k8s API?

- All API resource objects are stored in etcd, making etcd a critical target. Flaws in API handling could expose this data.
- The Kubernetes API supports operations beyond basic CRUD, including watch, deletecollection, and others. These extended operations increase the attack surface and potential for vulnerabilities.



Understanding Kubernetes API Actions

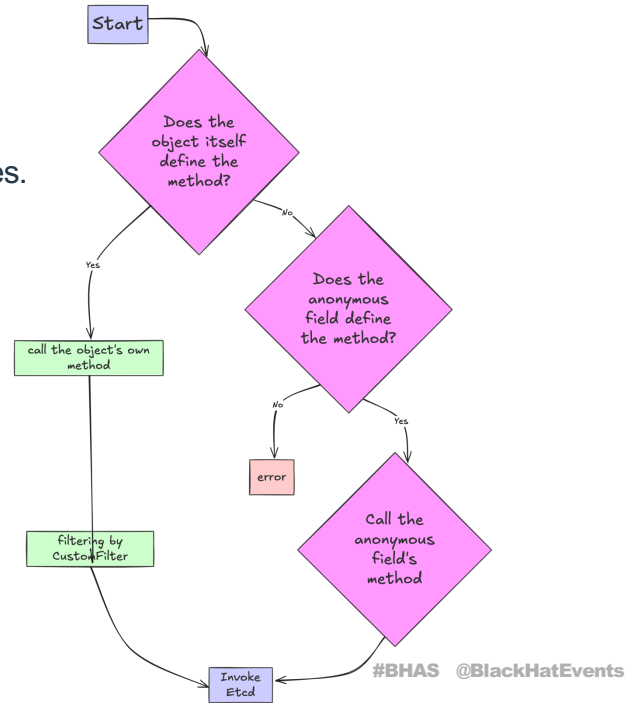
API Verb	HTTP Method	REST Handler	Description
CREATE	POST	Create(ctx, Obj, …)	Create new resource
GET	GET	Get(ctx, name, options)	Get a single resource with specified name
LIST	GET	List(ctx, options)	Get resource list
UPDATE	PUT	Update(ctx, name, …)	Update exsiting resource
PATCH	PATCH	Update(ctx, name, …)	Patch a resource
DELETE	DELETE	Delete(ctx, name, …)	Delete resource
DELETECOLLECTION	DELETE	DeleteCollection(ctx, …, dryRun)	Delete all resources that match options
WATCH	GET ?watch=true	Watch(ctx, options)	Monitor a resource change in real-time

Method Promotion: Filter Bypass Risk

- Developers often require finer-grained access control than RBAC provides.
- Developers implement custom filters at the REST layer to enforce these fine-grained policies.
- Developers may **overlook that embedded (anonymous) structs in Go can promote methods**, potentially bypassing these custom filters.

```
type ClusterRest struct {
    *genericregistry.Store
}

func (s *ClusterRest) ShortNames() []string {
    retur
}

var _ res

//var _ r
//var _ r

func (s *
    retur
}
```

```
func (s *ClusterRest) Delete(ctx context.Context, name string, deleteValidation
rest.ValidateObjectFunc, options *metav1.DeleteOptions) (runtime.Object, bool, error)
func (e *genericregistry.Store) DeleteCollection(ctx context.Context,
deleteValidation rest.ValidateObjectFunc, options *metav1.DeleteOptions, listOptions
*metainternalversion.ListOptions) (runtime.Object, error)
func (e *genericregistry.Store) DeleteReturnsDeletedObject() bool
func (e *genericregistry.Store) Destroy()
func (s *ClusterRest) Get(ctx context.Context, name string, options
*metav1.GetOptions) (runtime.Object, error)
func (e *genericregistry.Store) GetCreateStrategy() rest.RESTCreateStrategy
```



Simple, But Effective

- The **deletecollection** and **watch** verbs are often overlooked in development.
- fuzzed the API by systematically varying HTTP verbs and API endpoint combinations.
- Use **dryRun** with deletecollection to safely explore its behavior without risking data loss or disruption.
- Use **timeoutSeconds** with watch to control connection duration and prevent resource exhaustion during testing.

Workshop In Action & Demo

Workshop: simple multi-tenant cluster

- We'll use a custom resource: workshop.io/v1alpha1/cluster.
- Each Cluster resource includes tenant ownership and kubeconfig information.
- The example include two tenants, tenant-1 and tenant-2, each with their own Cluster resource, tenant isolation is enforced

```go
// Cluster is the Schema for the clusters API
type Cluster struct {
    metav1.TypeMeta   `json:",inline"`
    metav1.ObjectMeta `json:"metadata,omitempty" protobuf:"bytes,1,opt,name=metadata"`

    Spec   ClusterSpec   `json:"spec,omitempty" protobuf:"bytes,2,opt,name=spec"`
    Status ClusterStatus `json:"status,omitempty"`
}

// ClusterSpec defines the desired state of Cluster
type ClusterSpec struct {
    Tenant     string `json:"tenant,omitempty" protobuf:"bytes,1,opt,name=tenant"`
    Name       string `json:"name,omitempty" protobuf:"bytes,2,opt,name=name"`
    APIServer  string `json:"apiserver,omitempty" protobuf:"bytes,3,opt,name=apiserver"`
    Kubeconfig []byte `json:"kubeconfig,omitempty" protobuf:"bytes,4,opt,name=kubeconfig"`
}
```

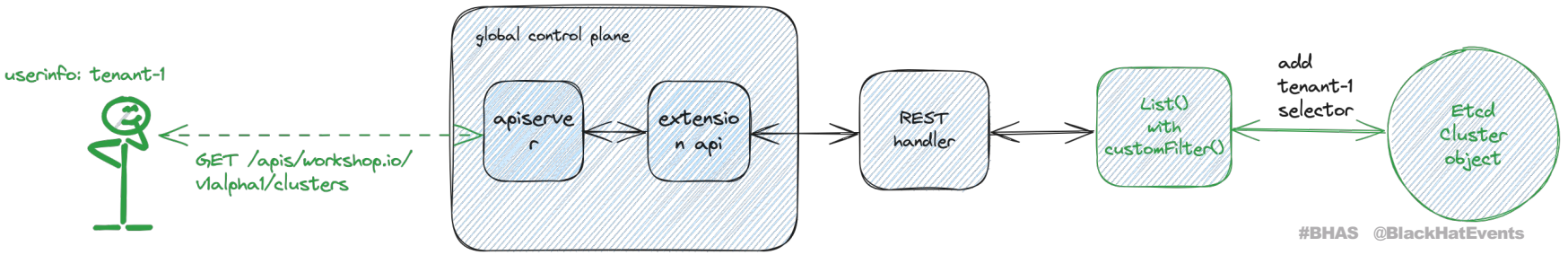
```yaml
apiVersion: apiregistration.k8s.io/v1
kind: APIService
metadata:
  name: v1alpha1.workshop.io
spec:
  version: v1alpha1
  versionPriority: 1000
  group: workshop.io
  groupPriorityMinimum: 10000
  insecureSkipTLSVerify: true
  service:
    name: workshop-apiservice
    namespace: kubeapi-inspector-workshop
---
apiVersion: v1
kind: Service
metadata:
  name: workshop-apiservice
  namespace: kubeapi-inspector-workshop
  labels:
    app: workshop-apiservice
spec:
  ports:
    - name: apiservice
      port: 443
      protocol: TCP
      targetPort: 443
  selector:
    app: workshop-apiserver
```

Workshop: simple multi-tenant cluster

- In the intended setup, a request from tenant-1 for Cluster resources will only return cluster-1, which they own.

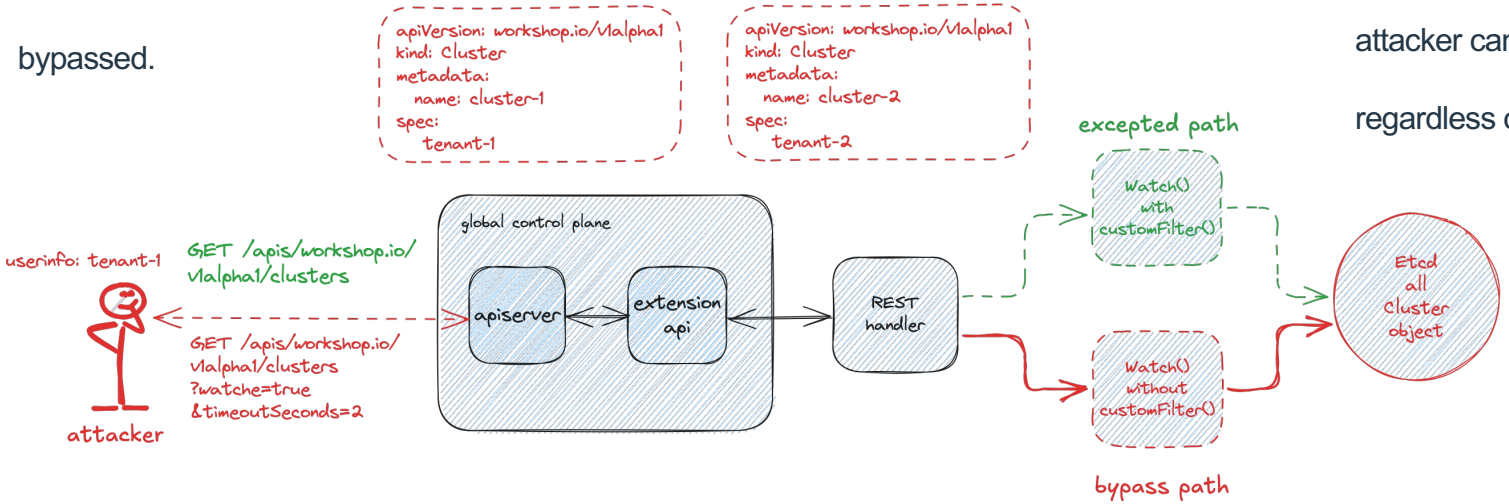
```
apiVersion: workshop.io/v1alpha1
kind: Cluster
metadata:
  name: cluster-1
spec:
   tenant-1
```

```go
func (s *ClusterRest) List(ctx context.Context, options *metainternalversion.ListOptions) (runtime.Object, error) {
    userInfo, ok := genericapirequest.UserFrom(ctx)

    if !ok {
        return &v1alpha1.ClusterList{}, nil
    }

    tenant := getServiceAccountName(userInfo.GetName())
    if options == nil {
        options = &metainternalversion.ListOptions{
            FieldSelector: fields.OneTermEqualSelector("spec.tenant", tenant),
        }
    }

    if options.FieldSelector == nil {
        options.FieldSelector = fields.OneTermEqualSelector("spec.tenant", tenant)
        return options
    }
    options.FieldSelector = fields.AndSelectors(options.FieldSelector, fields.OneTermEqualSelector("spec.tenant", tenant))

    return s.Store.List(ctx, newListOptions)
}
```



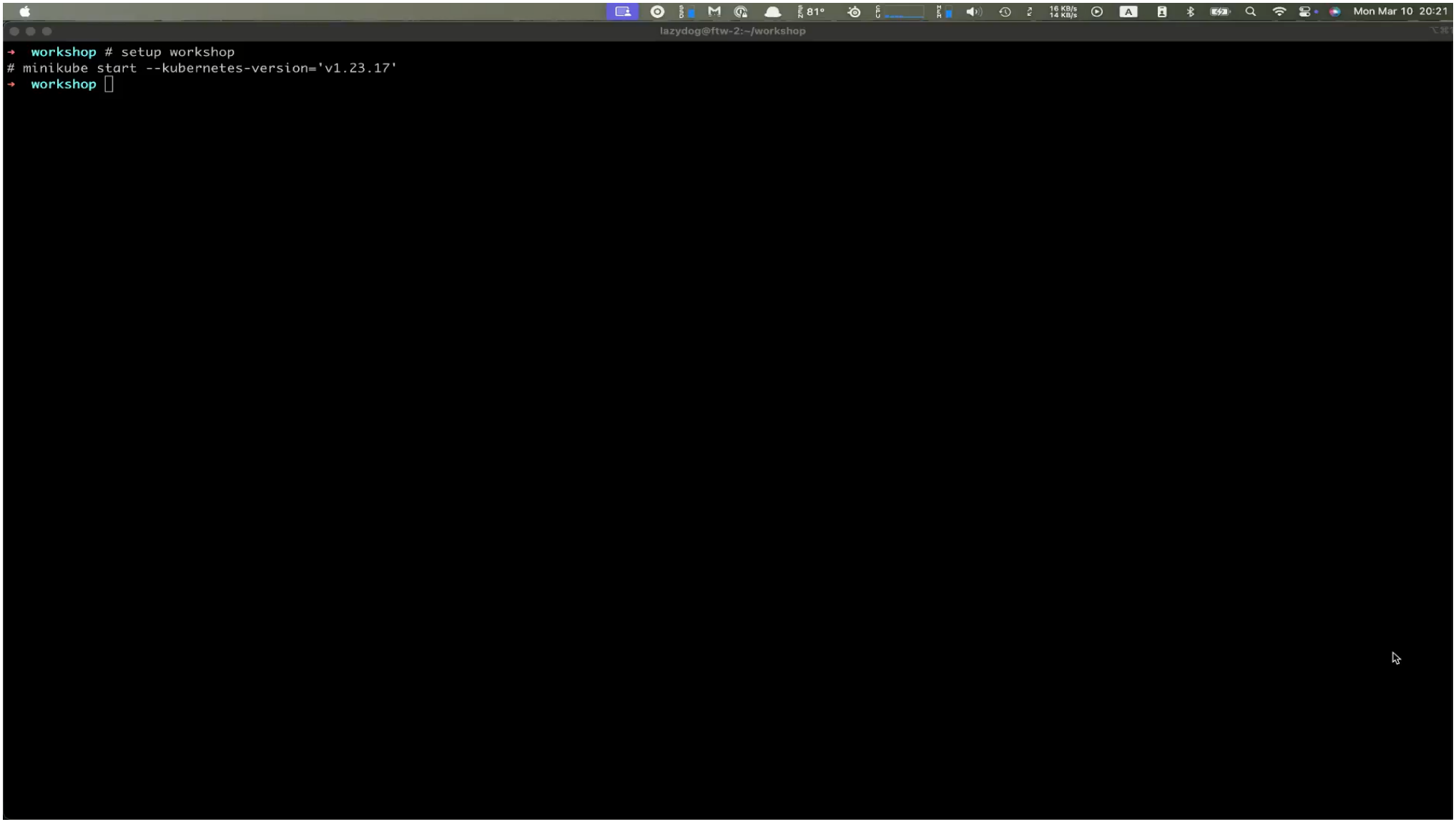
Workshop: Bypass the Resource Filter

- Due to method promotion, the custom filter on ClusterRest List method can be bypassed.



- By crafting a request that targets the embedded Store's List method, an attacker can retrieve all Cluster objects, regardless of tenant.

Demo


Mon Mar 10 20:21
lazydog@ftw-2:~/workshop
➜ workshop # setup workshop
minikube start --kubernetes-version='v1.23.17'
➜ workshop

Next & Future

- Enhance detection of unauthorized access to the Extension API Server
- Expand API fuzzing to include subresource coverage.
- Add more vulnerabilities about kubernetes API in workshop

https://github.com/yeahx/KubeAPI-Inspector

Thanks