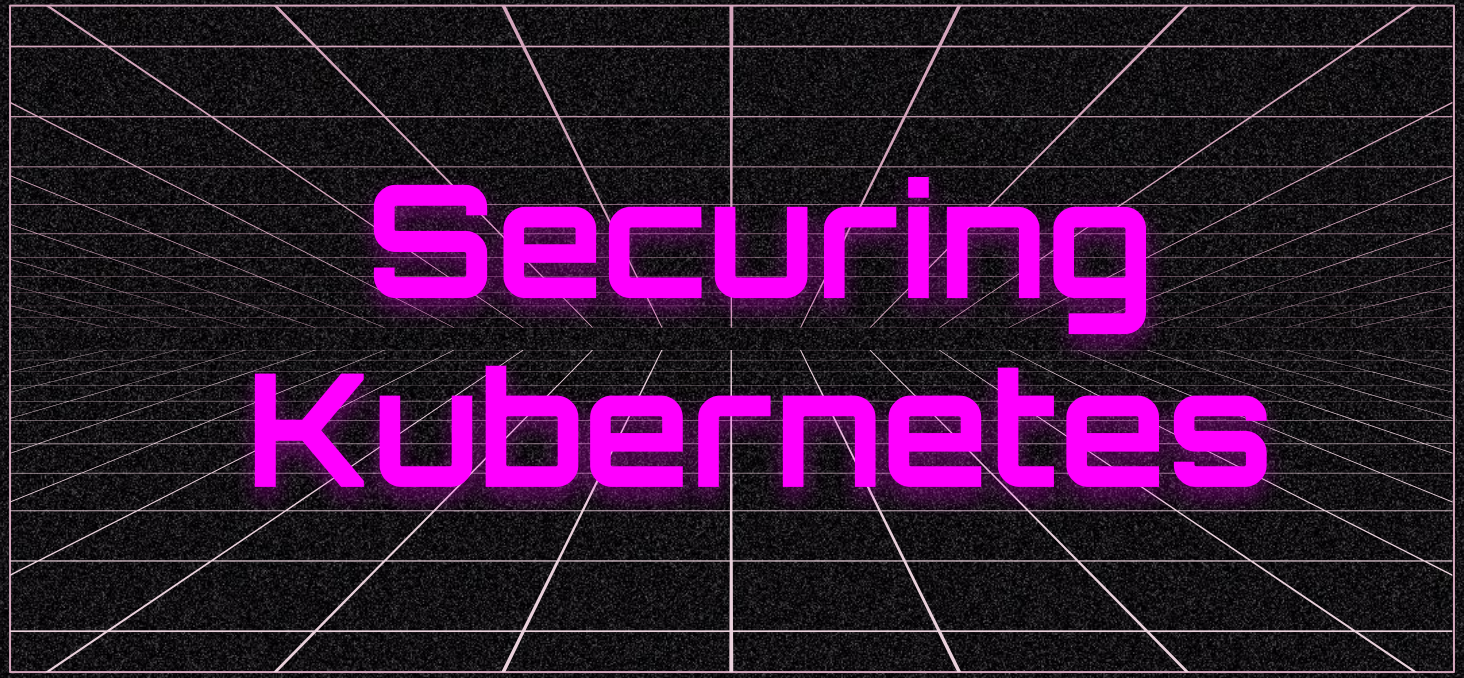


Protecting Your Cloud: A Simple Guide to Securing Kubernetes

Outline

01 The Problem

Current Kubernetes security challenges and attack vectors

02 Defense Strategy

Core security strategies for protecting a Kubernetes cluster across multiple levels

03 Runtime Defense

Tools and practices for detecting and mitigating threats in real-time.

04 Next Steps

Call to action for continuous security improvement.

01

The Problem

What is Kubernetes?

Before Kubernetes

- Scaling is PAINFUL
- Dev/Test/Prod Environment inconsistent
- Downtime is COMMON

After Kubernetes

- Scaling is AUTOMATED
- Consistent environment
- Downtime is minimized

What is Kubernetes?

Kubernetes is an open source container orchestration engine for automating deployment, scaling, and management of containerized applications

> Terms

Master Node: Machine that run control plane

Control Plane: Services that manage cluster

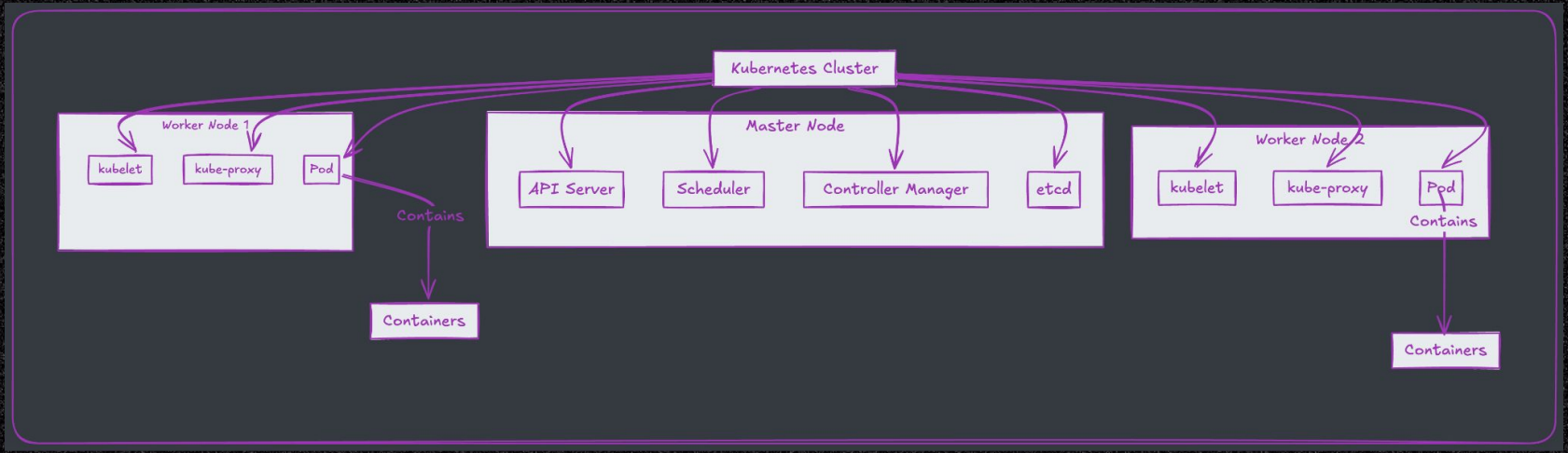
Node: Worker Machine running pods, managed by control plane

Kubelet: Agent that communicates with the control plane to receive command from control plane and report node states

Kube-proxy: Runs on each node, managing service communication by forwarding traffic to the app backends and supporting cluster IPs via environment variables or optional DNS.

Pod: Smallest deployable unit, contains one or more containers

Kubernetes Cluster



Autoscaling

The Cost of Breach

2021 – $1.8M compute costs

Cryptojacking campaign hijacked 2,400+ public clusters through weak kubelet authentication (anonymous auth enabled)

2021 – 500K AWS Credentials Stolen

Malicious Helm chart "redis-optimized" stole AWS credentials.

2024

CVE-2024-9486 – Kubernetes Image Builder versions <= v0.1.37 allows attackers to gain root access to nodes using VM images created with the Proxmox provider due to enabled default credentials during the image build process

Source:

- https://www.microsoft.com/en-us/security/business/security-intelligence-report
- https://www.tenable.com/cve/CVE-2024-9486

The Threat Landscape:

2025's Attack Playbook

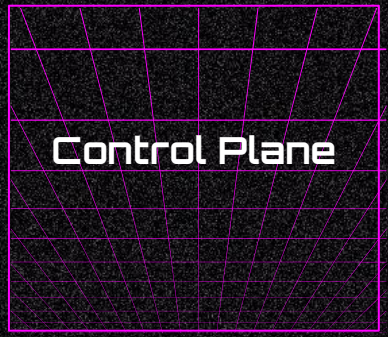
63% from misconfigurations

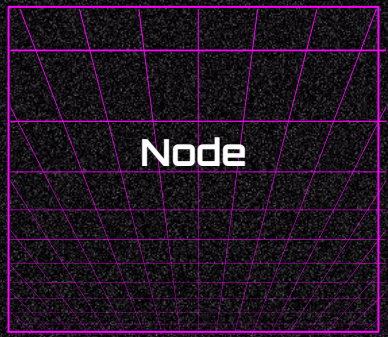
37% (supply chain attack, vulnerability on images, insider attack, etc)

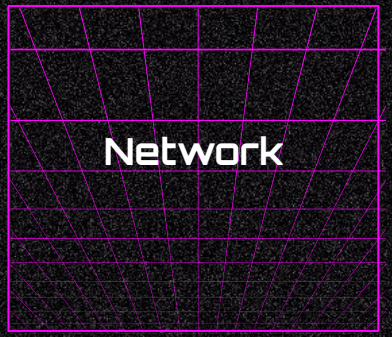
02

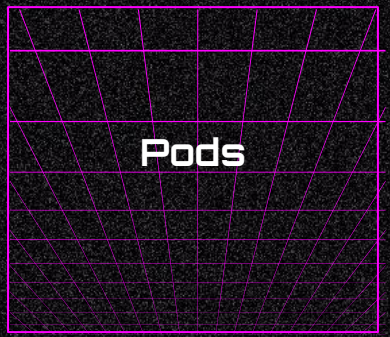
Defense Strategy

Defense Strategy









CONTROL PLANE

The control plane is the brain of the Kubernetes cluster

- Restrict Access: Use bastion hosts or VPNs to limit access to nodes hosting control plane components, ensuring only authorized personnel can interact with them.

- Enforce TLS Encryption: Ensure all communication between control plane components and API servers is encrypted using Transport Layer Security (TLS) to prevent man-in-the-middle attacks

- Role-Based Access Control (RBAC): Implement RBAC policies to enforce least privilege access. Regularly audit permissions and test configurations by impersonating users or user groups via user impersonation

Tools: kube-bench, cert-manager, hashicorp vault

kube-bench

kube-bench is a tool that checks whether Kubernetes is deployed securely by running the checks documented in the CIS Kubernetes Benchmark.

```
$ kubectl apply -f job.yaml
job.batch/kube-bench created

$ kubectl get pods
NAME                      READY   STATUS              RESTARTS   AGE
kube-bench-j76s9   0/1     ContainerCreating   0          3s

# Wait for a few seconds for the job to complete
$ kubectl get pods
NAME                      READY   STATUS      RESTARTS   AGE
kube-bench-j76s9   0/1     Completed   0          11s

# The results are held in the pod's logs
kubectl logs kube-bench-j76s9
[INFO] 1 Master Node Security Configuration
[INFO] 1.1 API Server
...
```

kube-bench

```
[INFO] 4 Worker Node Security Configuration
[INFO] 4.1 Worker Node Configuration Files
[PASS] 4.1.1 Ensure that the kubelet service file permissions are set to 644 or more restrictive (Automated)
[PASS] 4.1.2 Ensure that the kubelet service file ownership is set to root:root (Automated)
[PASS] 4.1.3 If proxy kubeconfig file exists ensure permissions are set to 644 or more restrictive (Manual)
[PASS] 4.1.4 Ensure that the proxy kubeconfig file ownership is set to root:root (Manual)
[PASS] 4.1.5 Ensure that the --kubeconfig kubelet.conf file permissions are set to 644 or more restrictive (Automated)
[PASS] 4.1.6 Ensure that the --kubeconfig kubelet.conf file ownership is set to root:root (Manual)
[PASS] 4.1.7 Ensure that the certificate authorities file permissions are set to 644 or more restrictive (Manual)
[PASS] 4.1.8 Ensure that the client certificate authorities file ownership is set to root:root (Manual)
[PASS] 4.1.9 Ensure that the kubelet --config configuration file has permissions set to 644 or more restrictive (Automated)
[PASS] 4.1.10 Ensure that the kubelet --config configuration file ownership is set to root:root (Automated)
[INFO] 4.2 Kubelet
[PASS] 4.2.1 Ensure that the anonymous-auth argument is set to false (Automated)
[PASS] 4.2.2 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)
[PASS] 4.2.3 Ensure that the --client-ca-file argument is set as appropriate (Automated)
[PASS] 4.2.4 Ensure that the --read-only-port argument is set to 0 (Manual)
[PASS] 4.2.5 Ensure that the --streaming-connection-idle-timeout argument is not set to 0 (Manual)
[FAIL] 4.2.6 Ensure that the --protect-kernel-defaults argument is set to true (Automated)
[PASS] 4.2.7 Ensure that the --make-iptables-util-chains argument is set to true (Automated)
[PASS] 4.2.8 Ensure that the --hostname-override argument is not set (Manual)
[WARN] 4.2.9 Ensure that the --event-qps argument is set to 0 or a level which ensures appropriate event capture (Manual)
[WARN] 4.2.10 Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Manual)
[PASS] 4.2.11 Ensure that the --rotate-certificates argument is not set to false (Manual)
[PASS] 4.2.12 Verify that the RotateKubeletServerCertificate argument is set to true (Manual)
[WARN] 4.2.13 Ensure that the Kubelet only makes use of Strong Cryptographic Ciphers (Manual)
```

Node

Nodes are the worker machines that run your applications

- Minimize Attack Surface: Disable unnecessary services and ports on nodes to reduce exposure to potential threats

- Regular Updates: Keep node operating systems and container runtimes up-to-date with the latest patches to mitigate vulnerabilities

- Immutable Infrastructure: Use immutable infrastructure practices where possible, ensuring nodes are replaced rather than patched in-place to maintain consistency and reduce drift

- Node Isolation: Use dedicated nodes for sensitive workloads and apply taints/tolerations to ensure proper workload placement

Tools: kube-hunter

Network

Network security ensures secure communication within and outside the cluster:

- Network Policies: Enforce Kubernetes-native network policies to restrict traffic between pods and namespaces, preventing lateral movement in case of compromise

- Encrypt Traffic: Use mutual TLS (mTLS) for service-to-service communication to ensure data confidentiality and integrity

- Ingress/EGress Controls: Configure ingress controllers and egress gateways to monitor and filter incoming/outgoing traffic, blocking unauthorized connections

Tools: Callico

Calico

Manage network policy

It runs as daemonset, ensuring each node run calico pod

$ kubectl create -f https://raw.githubusercontent.com/projectcalico/calico/v3.29.2/manifests/custom-resources.yaml

Use domain names in a global network set

In this method, you create a **GlobalNetworkSet** with the allowed destination domain names in the `allowedEgressDomains` field. Then, you create a **GlobalNetworkPolicy** with a `destination.selector` that matches that GlobalNetworkSet.

In the following example, the allowed egress domains (`api.alice.com` and `*.example.com`) are specified in the GlobalNetworkSet.

```
apiVersion: projectcalico.org/v3
kind: GlobalNetworkSet
metadata:
  name: allowed-domains-1
  labels:
    color: red
spec:
  allowedEgressDomains:
    - api.alice.com
    - '*.example.com'
```

Pods

Pods are the smallest deployable units in Kubernetes

- Least Privilege Principle: Run containers as non-root users and avoid privileged mode unless absolutely necessary

- Image Security: Use trusted base images from verified sources and scan them regularly for vulnerabilities before deployment

- Resource Limits: Set CPU/memory limits to prevent resource exhaustion attacks, such as denial-of-service (DoS)

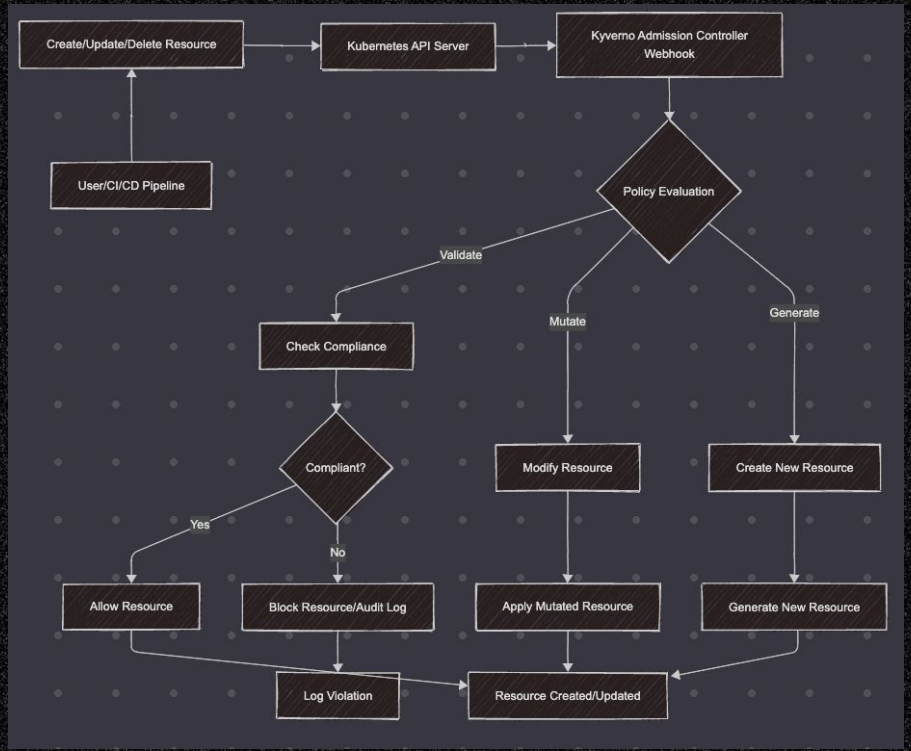
- Security Contexts: Define pod security contexts to enforce security settings like read-only filesystems, seccomp profiles, and AppArmor/SELinux policies

Tools: Trivy, checkov, kyverno

Kyverno

Allows you to manage and enforce policies on your Kubernetes clusters

```bash
helm repo add kyverno https://kyverno.github.io/kyverno/
helm repo update
helm install kyverno kyverno/kyverno --namespace kyverno --create-namespace
```



Example non-compliance pod

Error from server: admission webhook "validate.kyverno.svc" denied the request: All Pods must have the 'app' label.

kube-hunter

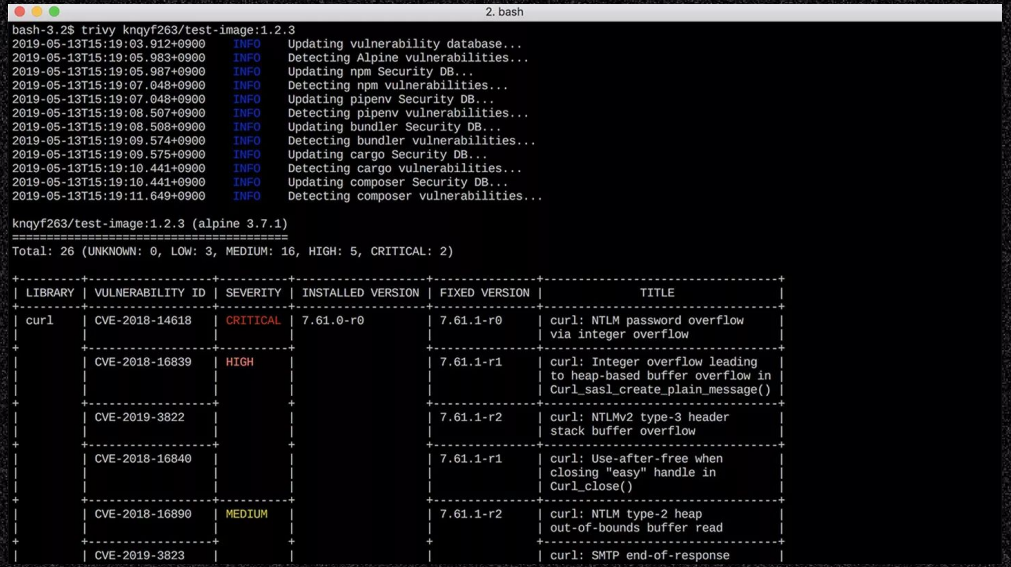
Actively tests your cluster for vulnerabilities, simulating attacks.

```bash
kube-hunter --remote <your-cluster-ip-or-domain>
```

```
Active Hunters:
---------------
* Foothold Via Secure Kubelet Port
  Attempts to demonstrate that a malicious actor can establish foothold into the cluster via a container
 abusing the configuration of the kubelet's secure port: authentication-auth=false.

* Malicious Intent Via Secure Kubelet Port
  Attempts to demonstrate that a malicious actor can leverage existing privileged containers exposed via
 the kubelet's secure port, due to anonymous auth enabled misconfiguration, such that a process can be s
tarted or modified on the host.

* Kubelet Run Hunter
  Executes uname inside of a random container

* Kubelet Container Logs Hunter
  Retrieves logs from a random container

* Kubelet System Logs Hunter
  Retrieves commands from host's system audit

* Azure SPN Hunter
  Gets the azure subscription file on the host by executing inside a container

* API server hunter
  Accessing the api server might grant an attacker full control over the cluster

* Arp Spoof Hunter
  Checks for the possibility of running an ARP spoof attack from within a pod (results are based on the
running node)

* DNS Spoof Hunter
  Checks for the possibility for a malicious pod to compromise DNS requests of the cluster (results are
based on the running node)

* Etcd Remote Access
  Checks for remote write access to etcd, will attempt to add a new key to the etcd DB

* Prove /var/log Mount Hunter
  Tries to read /etc/shadow on the host by running commands inside a pod with host mount to /var/log

* Build Date Hunter
  Hunts when proxy is exposed, extracts the build date of kubernetes

* K8s Version Hunter
  Hunts Proxy when exposed, extracts the version
```

Trivy

Detect vulnerabilities, misconfigurations, and other security issues in container images.



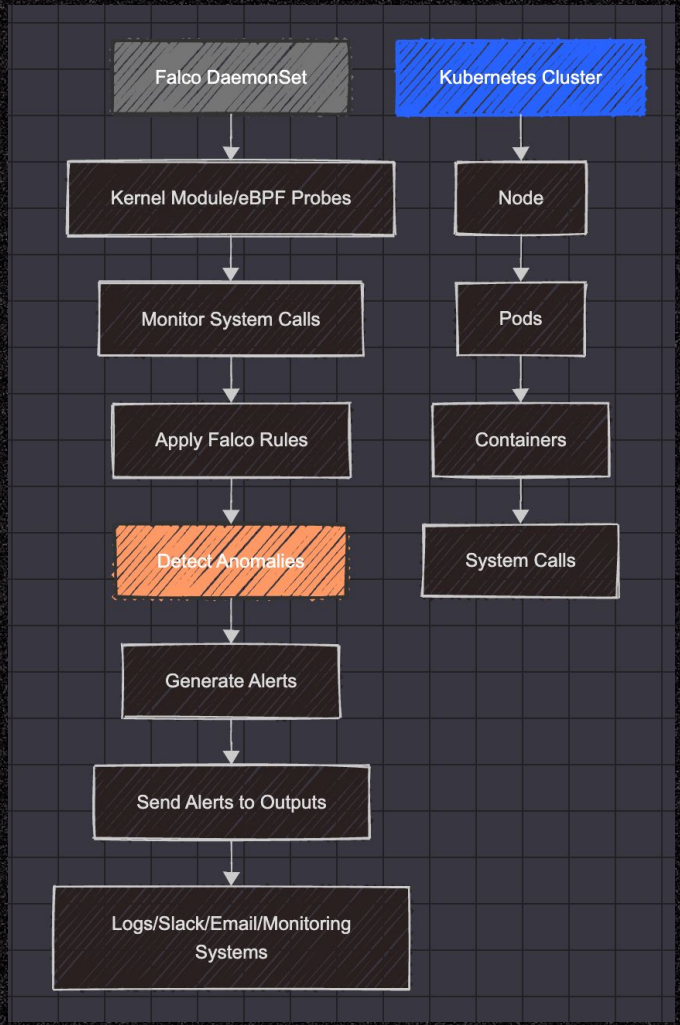
03

Runtime Defense

FALCO

Runtime Security monitoring & detection

$ helm repo add falcosecurity
https://falcosecurity.github.io/charts
$ helm repo update
$ helm install --replace falco --namespace falco
--create-namespace --set tty=true falcosecurity/falco



Falco

04

Next Steps

Next Steps

High Impact Low Effort

- Run kube-bench to audit (policy misconfiguration, restricted access, etc)
- Enforce Network Policies with Calico
- Regular Update Node OS

Next - Next Steps

- Micro segmentation
- Image scanner
- Runtime defense
- etc

Thank you