Boost your Security and Resiliency in Kubernetes

DevConf CZ 2019, Brno

Evis Barbullushi
EMEA Senior Specialist Solution Architect
AppDev OpenShift Team - RedHat
Agenda

- A Bit Of Context
- What This Is Not about
- Chaos Engineering And Security
- Safe Assumptions To Make
- Vulnerability Sources on K8s
- Models For Runtime Vulnerability Exploitation
- Best Practices For Security Chaos Engineering
- Some Example Security Exploit Tests
- How And Where To Run The Tests
A Bit Of Context
What This Is Not About

Not a lecture on..

- OS Security Tuning
- K8S Environment hardening
- Image security
- Design-time security checks
- Static code analysis
- Registry scanning solutions
- Workload isolation techniques
- Network Defense
- ...

- It’s more about ...
Chaos Engineering And Security

Chaos Engineering is the discipline of experimenting on a distributed system in order to build confidence in the system’s capability to withstand turbulent conditions in production. (principlesofchaos.org)

1. Define a desired state
2. Assume desired state is permanent
3. Try to break desired state
4. Check the difference in state and fix deltas

Repeatability:

Security Chaos Engineering
Breaking desired state = Exploiting and reporting security vulnerabilities

Run in production-like environment
Automated process
Damage control
Vulnerability Sources in K8S Environments

- Cluster vulnerabilities due to misconfiguration of core components
  **kube-apiserver, kube-scheduler, kube-controller-manager, etcd, kubelet, kube-proxy, ...**

- Cluster config, pod spec files and data dir access and ownership rights
  `/etc/kubernetes/manifests/*.yaml, ../cni/*, /etc/kubernetes/*.conf, /var/lib/etcd, /var/lib/kubelet/...`

- Cluster general security primitives
  (cluster)admin role misuse, misconfigured rolebindings, suboptimal namespaces, bad network policies, incorrect configuration of admission controllers, SELinux...

- Workload vulnerabilities
  - missing seccomp profiles
  - incorrect use of security contexts
  - incorrect use of pod security policies
  - privileged containers (uncompensated)
  - vulnerable application code
  - unnecessary syscall capabilities
  
  - privileged application, user or sa roles
  - sensitive information in env variables
  - container over-provisioning
  - access to more data than necessary in the application
  - unlimited processes in a pod
  - ...

- Wrong tools for the job
  - Third-party products or solutions not made fit for containers
  - Reliance on traditional security and monitoring solutions
  - Suboptimal environment/namespace segregation and role definitions in the team
Safe Assumptions To Make with Regard To Vanilla Deployments

Common Mistakes and Failures

- Zero-trust network security... NOT
- A lack of metadata-driven deployments
- Bad Secret Management
- Legacy integration and IaaS cloud deployments
- IAM and RBAC (extending it/from the Cloud)
- Platform Hardening (fear of black-box syndrome)
- Configuration Externalization not applied
- (Lack of) Logging, Monitoring
- Nobody is auditing you
Models For Runtime Vulnerability Exploitation

- Intra-Pod
- Pod-to-Pod
- Pod-to-Node
- Intra-Node
- Node-to-Node
- Node-to-Pod
- External-to-Cluster
- Cluster-to-External
Some Example Security Exploit Tests

- Red Metrics from CAdvisor, Kubelet, Heapster,…
- Port scanning (for ports accidentally left open)
- Access to API-Server from outside the cluster
- Attempt etcd remote read/write access
- Force access to certificate authority files
- Inject sidecar container in pod/deployment definition
- NodePort attack
- Try breaking by privilege escalation
- Access nodes of the cluster bypassing bastion
- Registry attack
- Access the metadata service on AWS or GCP to get the IAM tokens or keys from a pod
- Retrieve azure subscription file on the host from a pod
- Try to mount /etc/shadow on the host from the pod
- Get an SA token and reuse it to spin up other containers with elevated rights
- ….
Good Practices for Security Chaos Engineering

- Complete standard documented hardening recommendations before launching exploits
- Build security chaos engineering in the pipeline
- Conduct passive and active (enforcing) testing
- Layer 3, 4 and 7 attacking
- Cloud IaaS layer testing
- Attack vulnerabilities of products and technologies (db, app servers, middleware -> refer to exploit-db.com, cvedetails.com, )
- Build a live policy violation and live alerting system (Prometheus, Grafana)
- Conduct scan at deployment and scheduled jobs
- Don’t try to fix things you break during the test
- ... however control damage and report
How And Where To Run The Tests

HOW
- Use open source tools libraries
  CIS Kubernetes Benchmark: kube-bench, kubernetes-security-benchmark
  Vulnerability scanning for K8s: kube-hunter, kubeaudit, heptio/sonobuoy
- Extend above mentioned libraries (there is plenty tests to add)
- Use kali linux (or alike) built-in features
- Write your own

WHERE
- CI/CD inside the cluster
- CI/CD outside the cluster
- Kubernetes CronJobs dedicated per namespace
- Kubernetes CronJobs across namespaces
Q&A