OFF ONE 2022

Kubernetes security: Deception phase

Dmitriy Evdokimov

Founder&CTO Luntry

Moscow, August 25, 2022

WhoAmI

- Founder and CTO of Luntry
- 10+ years in Information Security
- Co-organizer of conferences ZeroNights, DEFCON Russia (#7812)
- Ex-author and editor in “ХАКЕР”
- Author of k8s (in)security Telegram channel
- Authiored“Cloud-Native Security в Kubernetes” course
- Does not believe that you can make a system secure and reliable without understanding it.
- Talks at BlackHat, HITB, ZeroNights, HackInParis, Confidence, SAS, PHDays, OFFZONE, DevOpsConf, KuberConf, VK Kubernetes Conference, HighLoad++, and others.

luntry.ru

Agenda

Main topics

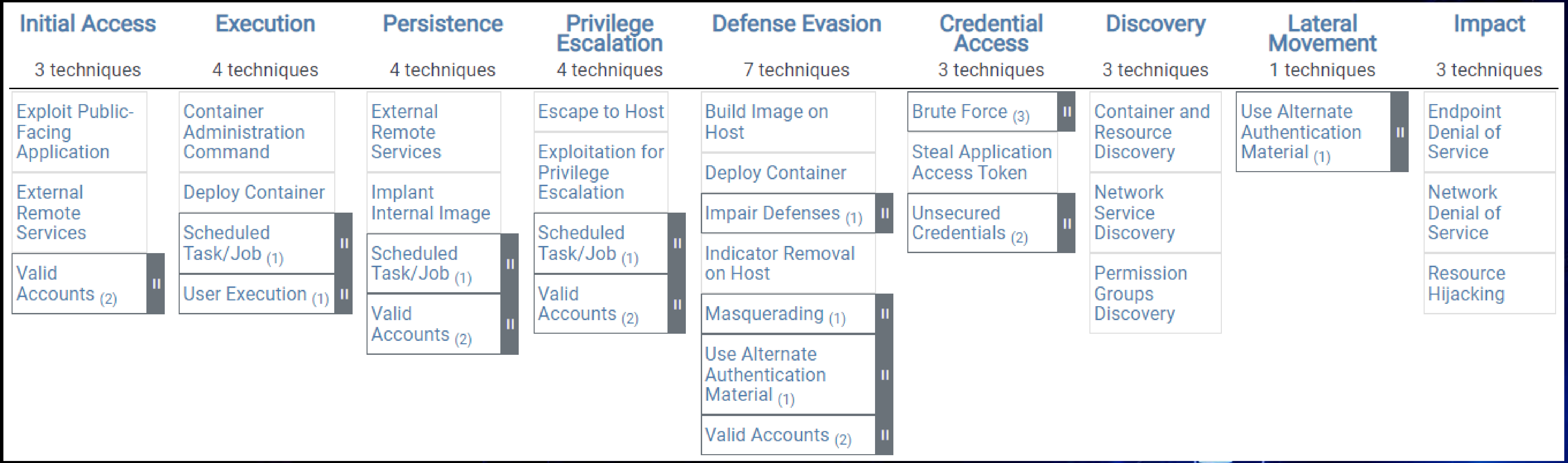
1. Threat management
2. Deception phase
3. Implementation of deception phase in Kubernetes
4. Conclusions

OFF ONE 2022

Threat management

LUNTRY

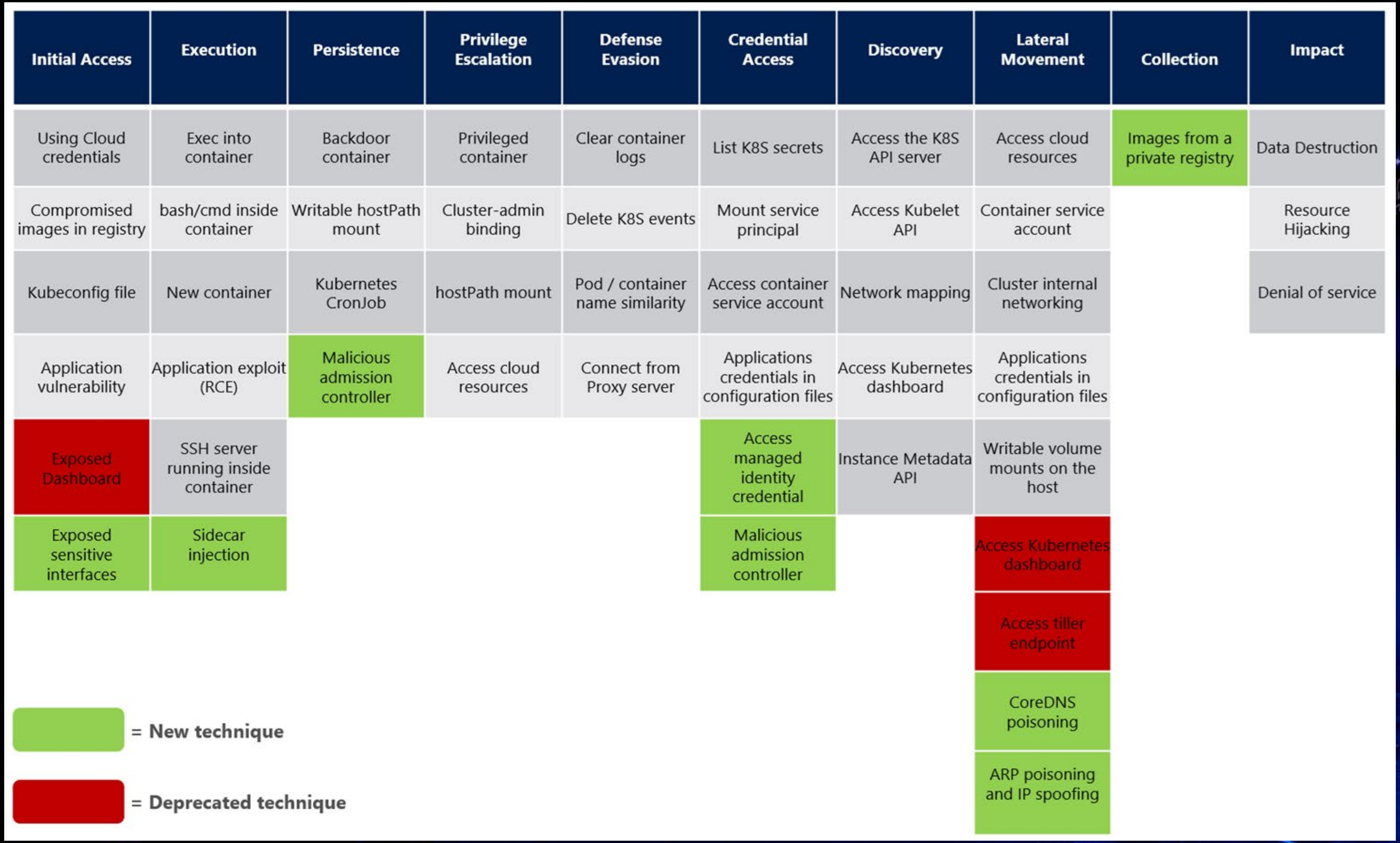
Containers Matrix by MITRE



Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Impact
3 techniques	4 techniques	4 techniques	4 techniques	7 techniques	3 techniques	3 techniques	1 techniques	3 techniques
Exploit Public-Facing Application	Container Administration Command	External Remote Services	Escape to Host	Build Image on Host	Brute Force (3)	Container and Resource Discovery	Use Alternate Authentication Material (1)	Endpoint Denial of Service
External Remote Services	Deploy Container	Implant Internal Image	Exploitation for Privilege Escalation	Deploy Container	Steal Application Access Token	Network Service Discovery		Network Denial of Service
Valid Accounts (2)	Scheduled Task/Job (1)	Scheduled Task/Job (1)	Scheduled Task/Job (1)	Impair Defenses (1)	Unsecured Credentials (2)	Permission Groups Discovery		Resource Hijacking
	User Execution (1)	Valid Accounts (2)	Valid Accounts (2)	Indicator Removal on Host				
				Masquerading (1)				
				Use Alternate Authentication Material (1)				
				Valid Accounts (2)				

Source link.

Threat matrix for Kubernetes



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 K8S secrets	Access the K8S API server	Access cloud resources	Images from a private registry	Data Destruction
Compromised images in registry	bash/cmd inside container	Writable hostPath mount	Cluster-admin binding	Delete K8S events	Mount service principal	Access Kubelet API	Container service account		Resource Hijacking
Kubeconfig file	New container	Kubernetes CronJob	hostPath mount	Pod / container name similarity	Access 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	Applications credentials in configuration files	Access Kubernetes dashboard	Applications credentials in configuration files		
Exposed Dashboard	SSH server running inside container				Access managed identity credential	Instance Metadata API	Writable volume mounts on the host		
Exposed sensitive interfaces	Sidecar injection				Malicious admission controller		Access Kubernetes dashboard		
							Access tiller endpoint		
							CoreDNS poisoning		
							ARP poisoning and IP spoofing		

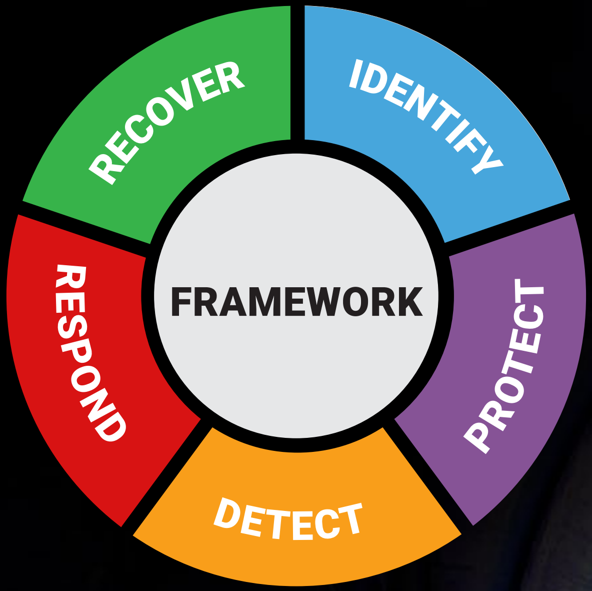
🟩 = New technique

🟥 = Deprecated technique

Source link.

NIST CyberSecurity Framework & Deception

NIST CyberSecurity Framework



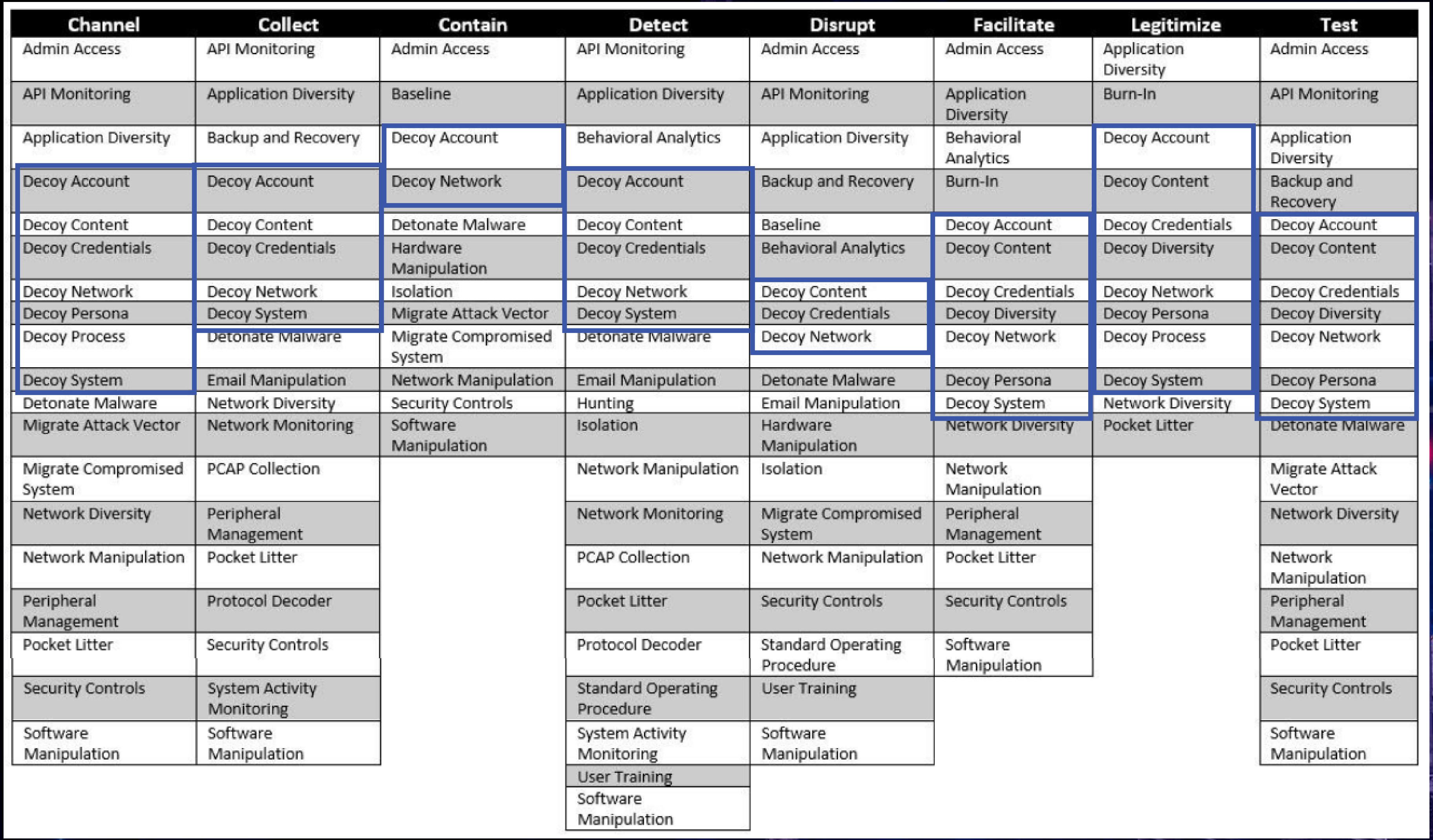
Where is Deception ?!

Shield Matrix by MITRE

Decoys

A publicly accessible knowledge base of **active defense** tactics and techniques based on real-world observations.

Source link.



Channel	Collect	Contain	Detect	Disrupt	Facilitate	Legitimize	Test
Admin Access	API Monitoring	Admin Access	API Monitoring	Admin Access	Admin Access	Application Diversity	Admin Access
API Monitoring	Application Diversity	Baseline	Application Diversity	API Monitoring	Application Diversity	Burn-In	API Monitoring
Application Diversity	Backup and Recovery	Decoy Account	Behavioral Analytics	Application Diversity	Behavioral Analytics	Decoy Account	Application Diversity
Decoy Account	Decoy Account	Decoy Network	Decoy Account	Backup and Recovery	Burn-In	Decoy Content	Backup and Recovery
Decoy Content	Decoy Content	Detonate Malware	Decoy Content	Baseline	Decoy Account	Decoy Credentials	Decoy Account
Decoy Credentials	Decoy Credentials	Hardware Manipulation	Decoy Credentials	Behavioral Analytics	Decoy Content	Decoy Diversity	Decoy Content
Decoy Network	Decoy Network	Isolation	Decoy Network	Decoy Content	Decoy Credentials	Decoy Network	Decoy Credentials
Decoy Persona	Decoy System	Migrate Attack Vector	Decoy System	Decoy Credentials	Decoy Diversity	Decoy Persona	Decoy Diversity
Decoy Process	Detonate Malware	Migrate Compromised System	Detonate Malware	Decoy Network	Decoy Network	Decoy Process	Decoy Network
Decoy System	Email Manipulation	Network Manipulation	Email Manipulation	Detonate Malware	Decoy Persona	Decoy System	Decoy Persona
Detonate Malware	Network Diversity	Security Controls	Hunting	Email Manipulation	Decoy System	Network Diversity	Decoy System
Migrate Attack Vector	Network Monitoring	Software Manipulation	Isolation	Hardware Manipulation	Network Diversity	Pocket Litter	Detonate Malware
Migrate Compromised System	PCAP Collection		Network Manipulation	Isolation	Network Manipulation		Migrate Attack Vector
Network Diversity	Peripheral Management		Network Monitoring	Migrate Compromised System	Peripheral Management		Network Diversity
Network Manipulation	Pocket Litter		PCAP Collection	Network Manipulation	Pocket Litter		Network Manipulation
Peripheral Management	Protocol Decoder		Pocket Litter	Security Controls	Security Controls		Peripheral Management
Pocket Litter	Security Controls		Protocol Decoder	Standard Operating Procedure	Software Manipulation		Pocket Litter
Security Controls	System Activity Monitoring		Standard Operating Procedure	User Training			Security Controls
Software Manipulation	Software Manipulation		System Activity Monitoring	Software Manipulation			Software Manipulation
			User Training				
			Software Manipulation				

MITRE Engage

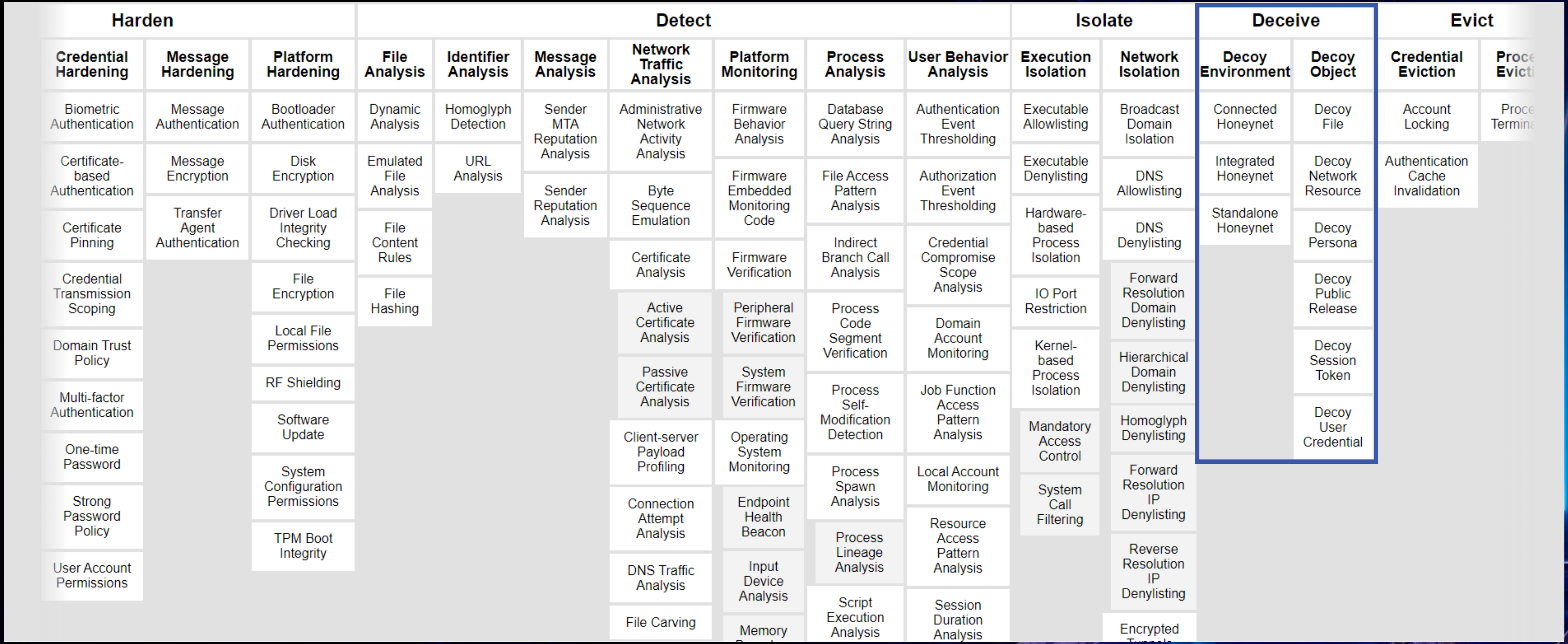
Active Defense

Goals	Prepare	Expose		Affect			Elicit		Understand
Approaches	Plan	Collect	Detect	Prevent	Direct	Disrupt	Reassure	Motivate	Analyze
Activities	Cyber Threat Intelligence	API Monitoring	Introduced Vulnerabilities	Baseline	Attack Vector Migration	Isolation	Application Diversity	Application Diversity	After-Action Review
	Engagement Environment	Network Monitoring	Lures	Hardware Manipulation	Email Manipulation	Lures	Artifact Diversity	Artifact Diversity	Cyber Threat Intelligence
	Gating Criteria	Software Manipulation	Malware Detonation	Isolation	Introduced Vulnerabilities	Network Manipulation	Burn-In	Information Manipulation	Threat Model
	Operational Objective	System Activity Monitoring	Network Analysis	Network Manipulation	Lures	Software Manipulation	Email Manipulation	Introduced Vulnerabilities	
	Persona Creation			Security Controls	Malware Detonation		Information Manipulation	Malware Detonation	
	Storyboarding				Network Manipulation		Network Diversity	Network Diversity	
	Threat Model				Peripheral Management		Peripheral Management	Personas	
					Security Controls		Pocket Litter		
					Software Manipulation				

Source link.

D3FEND Matrix by MITRE

A knowledge graph of cybersecurity countermeasures



Harden			Detect						
Credential Hardening	**Message Hardening**	**Platform Hardening**	**File Analysis**	**Identifier Analysis**	**Message Analysis**	**Network Traffic Analysis**	**Platform Monitoring**	**Process Analysis**	**User Behavior Analysis**
Biometric Authentication	Message Authentication	Bootloader Authentication	Dynamic Analysis	Homoglyph Detection	Sender MTA Reputation Analysis	Administrative Network Activity Analysis	Firmware Behavior Analysis	Database Query String Analysis	Authentication Event Thresholding
Certificate-based Authentication	Message Encryption	Disk Encryption	Emulated File Analysis	URL Analysis	Sender Reputation Analysis	Byte Sequence Emulation	Firmware Embedded Monitoring Code	File Access Pattern Analysis	Authorization Event Thresholding
Certificate Pinning	Transfer Agent Authentication	Driver Load Integrity Checking	File Content Rules			Certificate Analysis	Firmware Verification	Indirect Branch Call Analysis	Credential Compromise Scope Analysis
Credential Transmission Scoping		File Encryption	File Hashing			Active Certificate Analysis	Peripheral Firmware Verification	Process Code Segment Verification	Domain Account Monitoring
Domain Trust Policy		Local File Permissions				Passive Certificate Analysis	System Firmware Verification	Process Self-Modification Detection	Job Function Access Pattern Analysis
Multi-factor Authentication		RF Shielding				Client-server Payload Profiling	Operating System Monitoring	Process Spawn Analysis	Local Account Monitoring
One-time Password		Software Update				Connection Attempt Analysis	Endpoint Health Beacon	Process Lineage Analysis	Resource Access Pattern Analysis
Strong Password Policy		System Configuration Permissions				DNS Traffic Analysis	Input Device Analysis	Script Execution Analysis	Session Duration Analysis
User Account Permissions		TPM Boot Integrity				File Carving	Memory		

Isolate		Deceive		Evict	
Execution Isolation	**Network Isolation**	**Decoy Environment**	**Decoy Object**	**Credential Eviction**	**Proce Evict**
Executable Allowlisting	Broadcast Domain Isolation	Connected Honeynet	Decoy File	Account Locking	Proce Termin
Executable Denylisting	DNS Allowlisting	Integrated Honeynet	Decoy Network Resource	Authentication Cache Invalidation	
Hardware-based Process Isolation	DNS Denylisting	Standalone Honeynet	Decoy Persona		
IO Port Restriction	Forward Resolution Domain Denylisting		Decoy Public Release		
Kernel-based Process Isolation	Hierarchical Domain Denylisting		Decoy Session Token		
Mandatory Access Control	Homoglyph Denylisting		Decoy User Credential		
System Call Filtering	Forward Resolution IP Denylisting				
	Reverse Resolution IP Denylisting				
	Encrypted				

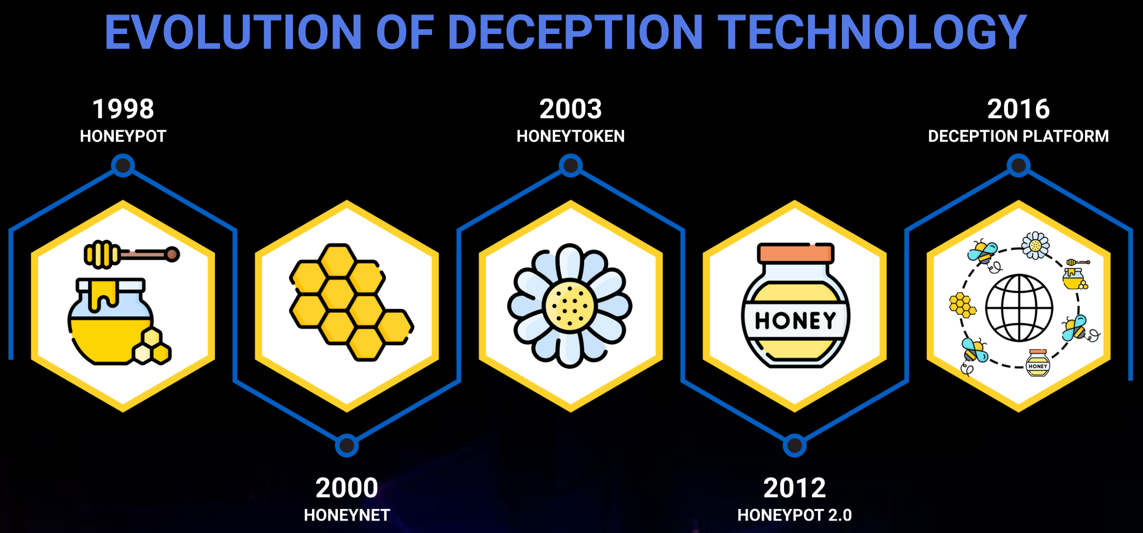
Source link.

Deception phase

Deception phase

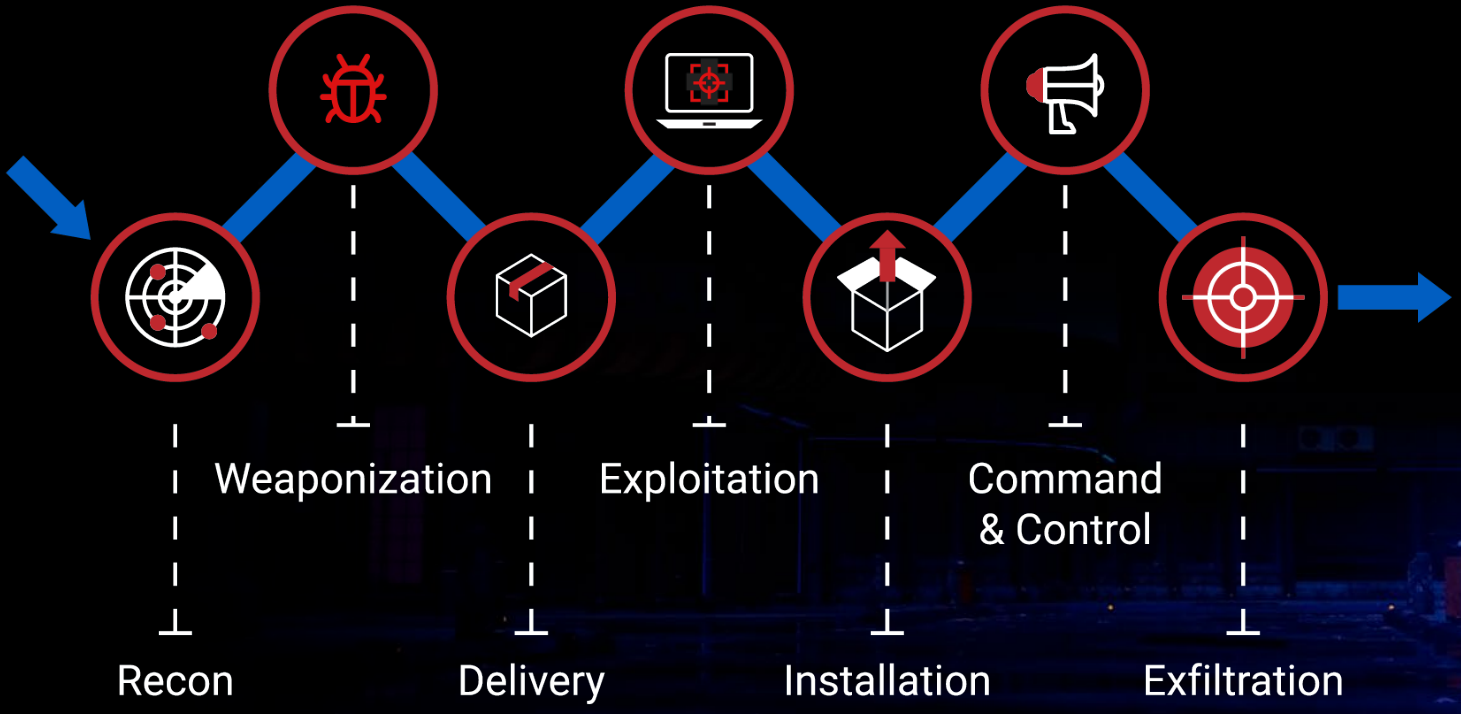
From reactive to active security

- Idea
  - Decoys
  - Traps
- Concept
  - "Detection through Deception"
  - "Security Through Deception“
- Benefits:
  - Easy to get started
  - No/Low false positives
  - Attack agnostic
  - Doesn’t increase the attack surface
  - Low overhead



Cyber kill chain

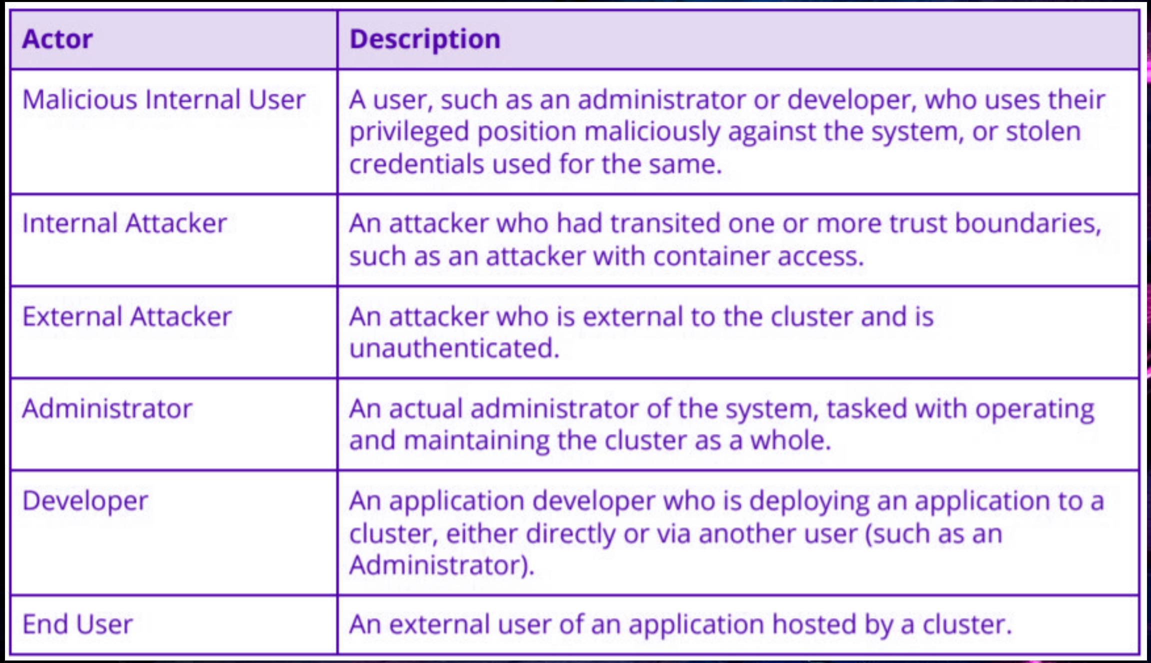
- A defender only has to make one mistake to get compromised.
- An attacker only has to make one mistake to get detected.



Threat Actors

Not all adversaries are the same

- Different adversary models have different entry points and opportunities
- Deception phase has to be organized considering relevant models:
  - But adversaries can switch models
  - Different decoys can help catch different adversaries
  - We need a complex approach



Actor	Description
Malicious Internal User	A user, such as an administrator or developer, who uses their privileged position maliciously against the system, or stolen credentials used for the same.
Internal Attacker	An attacker who had transited one or more trust boundaries, such as an attacker with container access.
External Attacker	An attacker who is external to the cluster and is unauthenticated.
Administrator	An actual administrator of the system, tasked with operating and maintaining the cluster as a whole.
Developer	An application developer who is deploying an application to a cluster, either directly or via another user (such as an Administrator).
End User	An external user of an application hosted by a cluster.

Source link.

OFFZONE 2022

Deception phase in K8s

LUNTRY

Implementation requirements

The cloud-native way

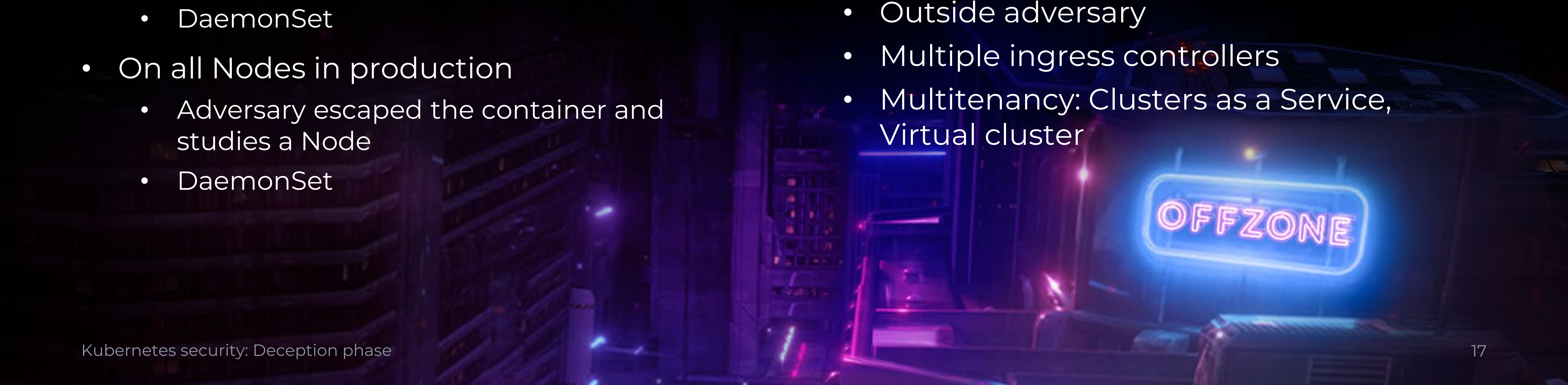
1. Lives with GitOps
2. Does not require extra effort from development teams
3. Minimum labor resources required

Spoiler: It’s easy to do in Kubernetes;)

How to deploy bait and traps?

Decoy Environment: Connected Honeynet, Integrated Honeynet, Standalone Honeynet

- Inside production microservices (Pod)
  - Adversary entered a microservices and investigates files & envs
  - MutatingAdmissionWebhook
- Next to production microservices
  - Adversary studies network environment
  - DaemonSet
- On all Nodes in production
  - Adversary escaped the container and studies a Node
  - DaemonSet
- On a special Node in production
  - Redirect adversary
  - Kubernetes pod to node scheduling: nodeSelector, Node affinity, taints and tolerations
- In a special Cluster
  - Outside adversary
  - Multiple ingress controllers
  - Multitenancy: Clusters as a Service, Virtual cluster



What to use as decoy?

Something that has no interactions

- Kubernetes cluster
- Nodes
- Pod/Workload
  - Vulnerable apps
  - Known ports like 80, 44134 (Tiller)
  - Consider NetworkPolicy
- Secret
  - Fake sensitive information
  - ServiceAccount token
- Non-used CRDs
  - Their list is available (/api) throughDefault ServiceAccount
- Ingress, Services, Endpoints
  - Paths
  - DNS records
  - UI: Apache NiFi, Kubeflow, Argo Workflows, Weave Scope, and the Kubernetes dashboard.
- ...

Decoy Environment

Prepared Clusters/Nodes/Workloads/Pods/Containers

Decoy Environment:

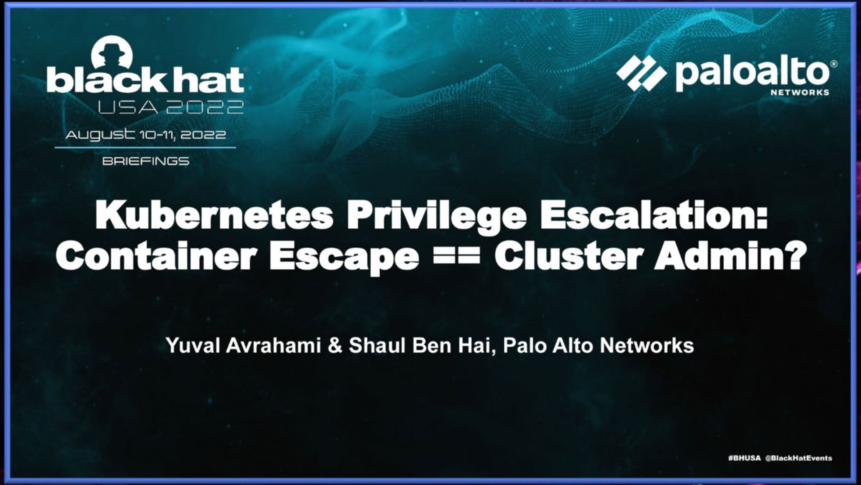
- Connected Honeynet,
- Integrated Honeynet,
- Standalone Honeynet

Initial access	Execution	Persistence	Privilege escalation	Defense evasion	Credential access	Discovery	Lateral movement	Collection	Impact
Application vulnerability	Exec into container						Access cloud resources	Images from a private registry	Data destruction
Exposed sensitive interfaces	Application exploit (RCE)								Resource Hijacking
									DoS

DaemonSet

Guarantee for everywhere

- Can help place decoys on every Node and subnetwork
- Great for detecting:
  - Adversaries inside Pods
    - Scan local IP ranges for open TCP and UDP ports
  - Adversaries on Nodes
    - After container escape
    - Steal secrets from node filesystem


black hat
USA 2022
August 10-11, 2022
BRIEFINGS

paloalto NETWORKS

Kubernetes Privilege Escalation: Container Escape == Cluster Admin?

Yuval Avrahami & Shaul Ben Hai, Palo Alto Networks

#BHUSA @BlackHatEvents

Source link.

Decoy File and Envs

Placing decoys

- Secrets resources and configs are added to a Pod/container as:
  - File
  - Envs
- Through DaemonSet, you can place decoy on Nodes
  - Certificates, keys, …

Initial access	Execution	Persistence	Privilege escalation	Defense evasion	Credential access	Discovery	Lateral movement	Collection	Impact
Using cloud creds					Mount service principal		Apps creds in conf files		
Kubeconfig file					Apps creds in conf files				

MutatingAdmissionWebhook

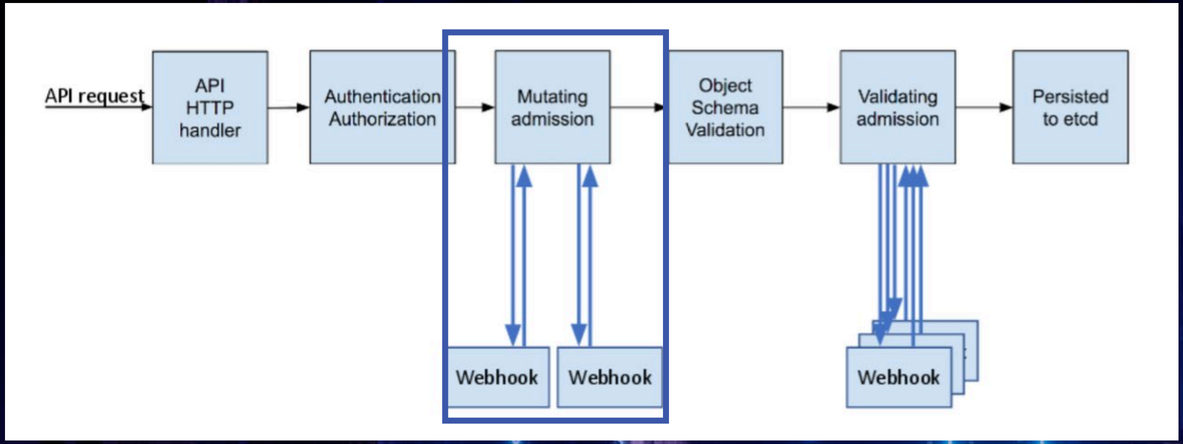
Invisible/transparent modification

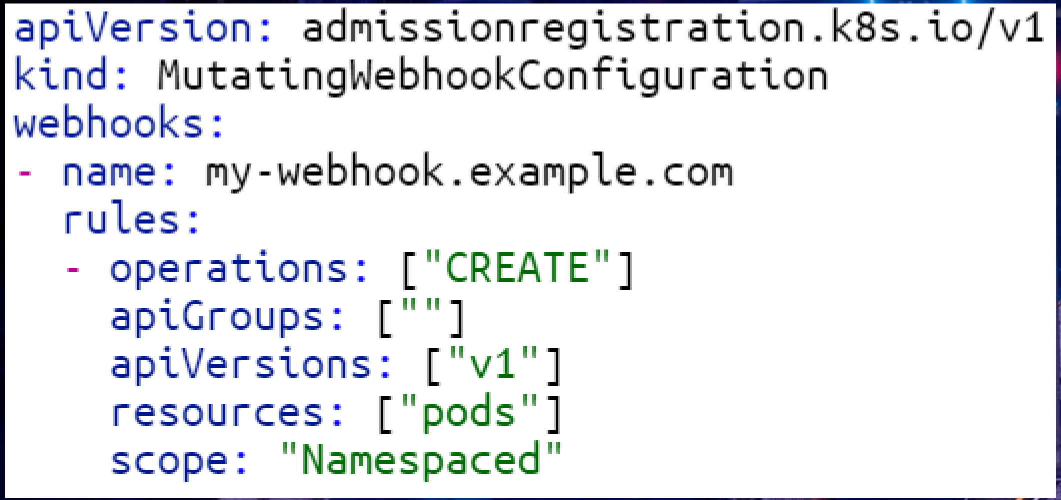
Using MutatingAdmissionWebhook, without bothering the development team, you can:

- Add special IPs and DNSs into containers' env variables and monitor calls
- Add files using init container and monitor calls to them
  - like Secrets Store CSI Driver, Vault Agent Sidecar Injector

You can use Policy Engines and create mutate policy:

- Kyverno
- OPA gatekeeper





```
apiVersion: admissionregistration.k8s.io/v1
kind: MutatingWebhookConfiguration
webhooks:
- name: my-webhook.example.com
  rules:
  - operations: ["CREATE"]
    apiGroups: [""]
    apiVersions: ["v1"]
    resources: ["pods"]
    scope: "Namespaced"
```

Decoy Network Resource

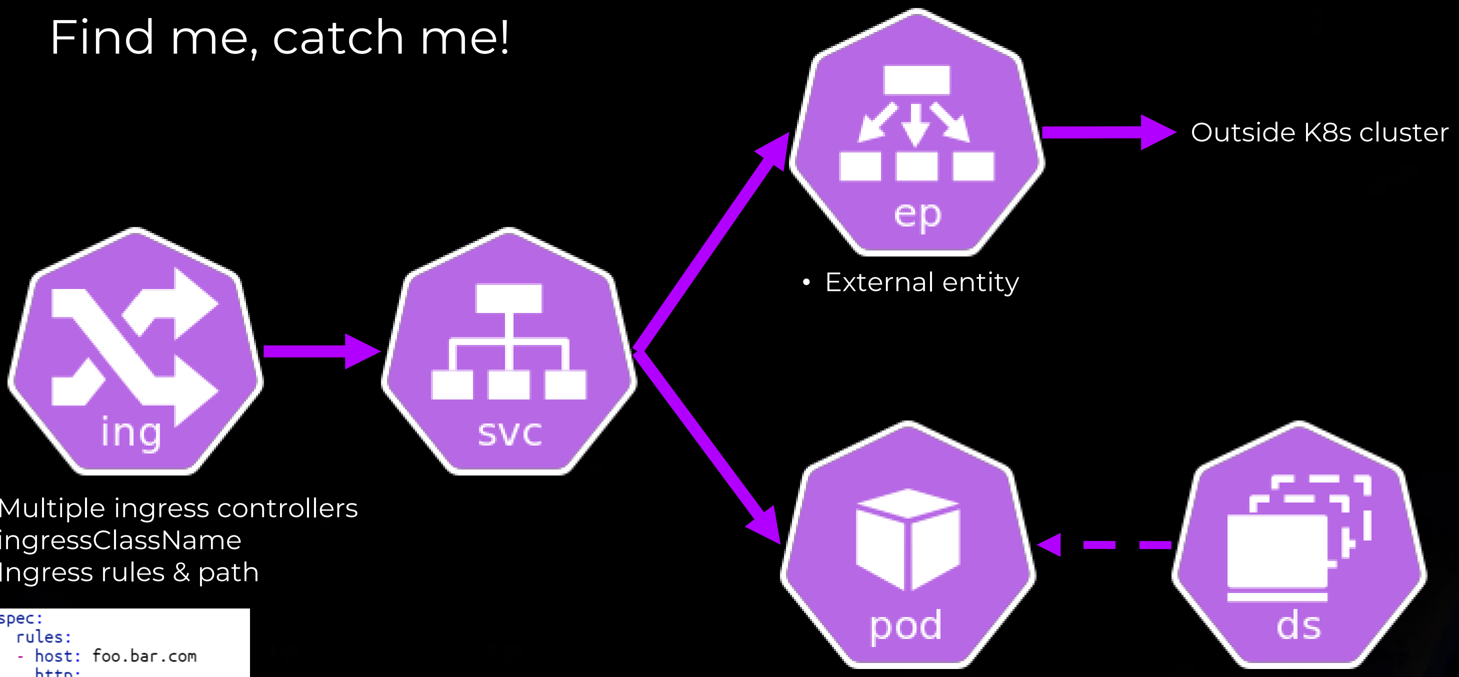
All around is microservices

Usually, it's tightly related to the Decoy Environment.

Initial access	Execution	Persistence	Privilege escalation	Defense evasion	Credential access	Discovery	Lateral movement	Collection	Impact
						Access the K8s API server	Access cloud resources		
						Access Kubelet API	Cluster internal networking		
						Network mapping	Access Kubernetes dashboard		
						Access K8s dashboard	Access Tiller endpoint		
						Instance Metadata API	ARP poisoning and IP spoofing		

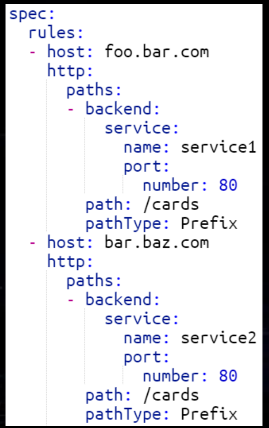
Ingress, Service names & DNS

Find me, catch me!

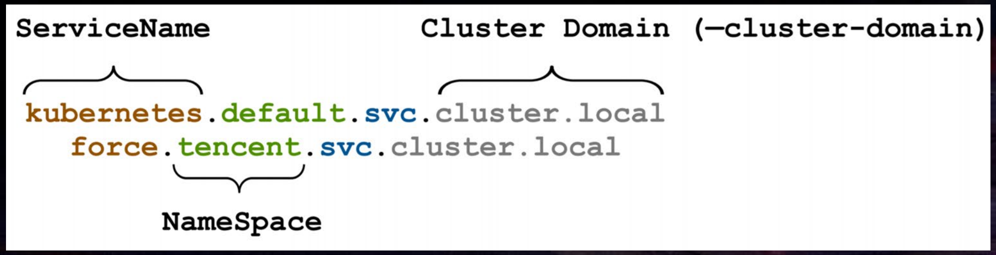


- External entity
- Internal entity

- Multiple ingress controllers
- ingressClassName
- Ingress rules & path

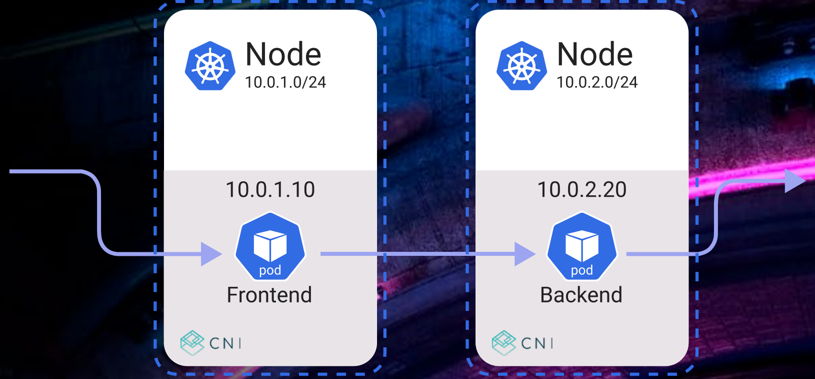


```
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - backend:
          service:
            name: service1
            port:
              number: 80
        path: /cards
        pathType: Prefix
  - host: bar.baz.com
    http:
      paths:
      - backend:
          service:
            name: service2
            port:
              number: 80
        path: /cards
        pathType: Prefix
```



$$\overbrace{\texttt{kubernetes}}^{\text{ServiceName}}\texttt{.default.svc.}\overbrace{\texttt{cluster.local}}^{\text{Cluster Domain (--cluster-domain)}}$$

$$\texttt{force.}\underbrace{\texttt{tencent}}_{\text{NameSpace}}\texttt{.svc.cluster.local}$$



- All Pods have IPs
- All Pods can talk
- PodCIDR[s] per node

- Services for load-balancing
- DNS for service-discovery
- Network Policy for segmentation

Decoy Session token

Kubernetes ServiceAccount Token (SA)

- Everything* goes through Kubernetes API server and RBAC
- Everything is located at /var/run/secrets/kubernetes.io/serviceaccount/token

Initial access	Execution	Persistence	Privilege escalation	Defense evasion	Credential access	Discovery	Lateral movement	Collection	Impact
	Exec into container	Backdoor container	Privileged container	Delete k8s events	List K8s secrets		Container service account		
	New container	Writable hostPath mount	Cluster-admin binding		Access container service account				
	Sidecar injection	Kubernetes CronJob	hostPath mount		Access managed identity credential				
	bash/cmd inside container	Malicious admission controller	Access cloud resources		Malicious admission controller				

Kubernetes Honey/Canary Token

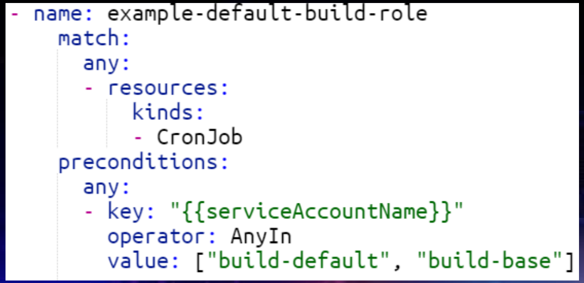
Most searched for

You can monitor:

- Calls to API SelfSubjectAccessReview, SelfSubjectRulesReview
- Denied transactions
- Anomalous calls to /var/run/secrets/kubernetes.io/serviceaccount/token

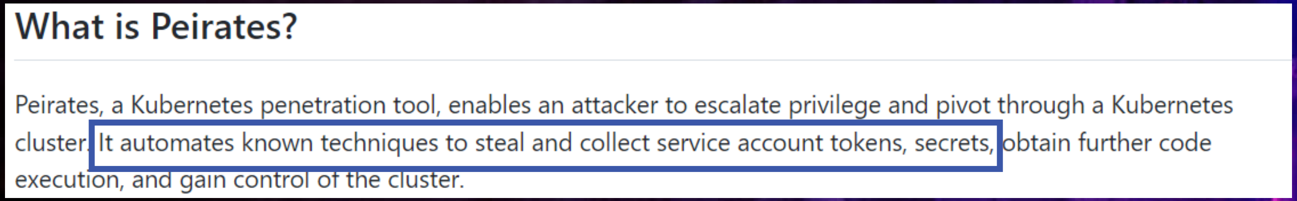
You can check serviceAccountName on Policy Engine as well as:

- Block
- Redirect (trough mutate policy)
- Alert



```
- name: example-default-build-role
  match:
    any:
    - resources:
        kinds:
        - CronJob
  preconditions:
    any:
    - key: "{{serviceAccountName}}"
      operator: AnyIn
      value: ["build-default", "build-base"]
```

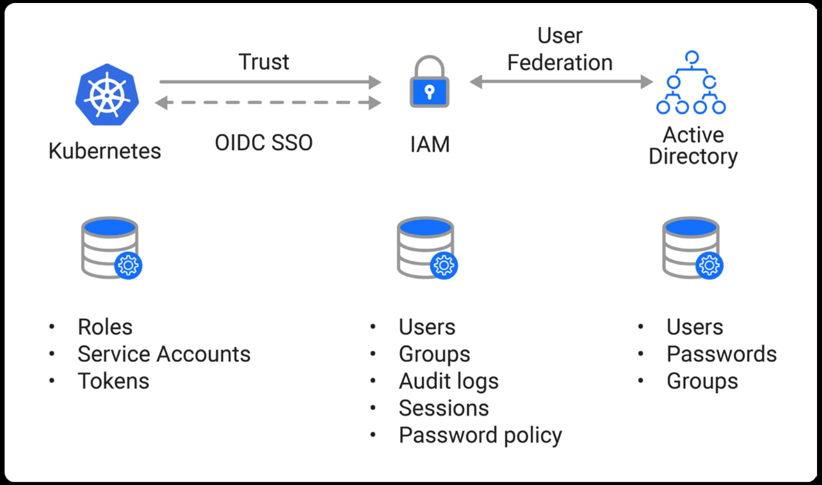
Можно ловить использование Peirates.

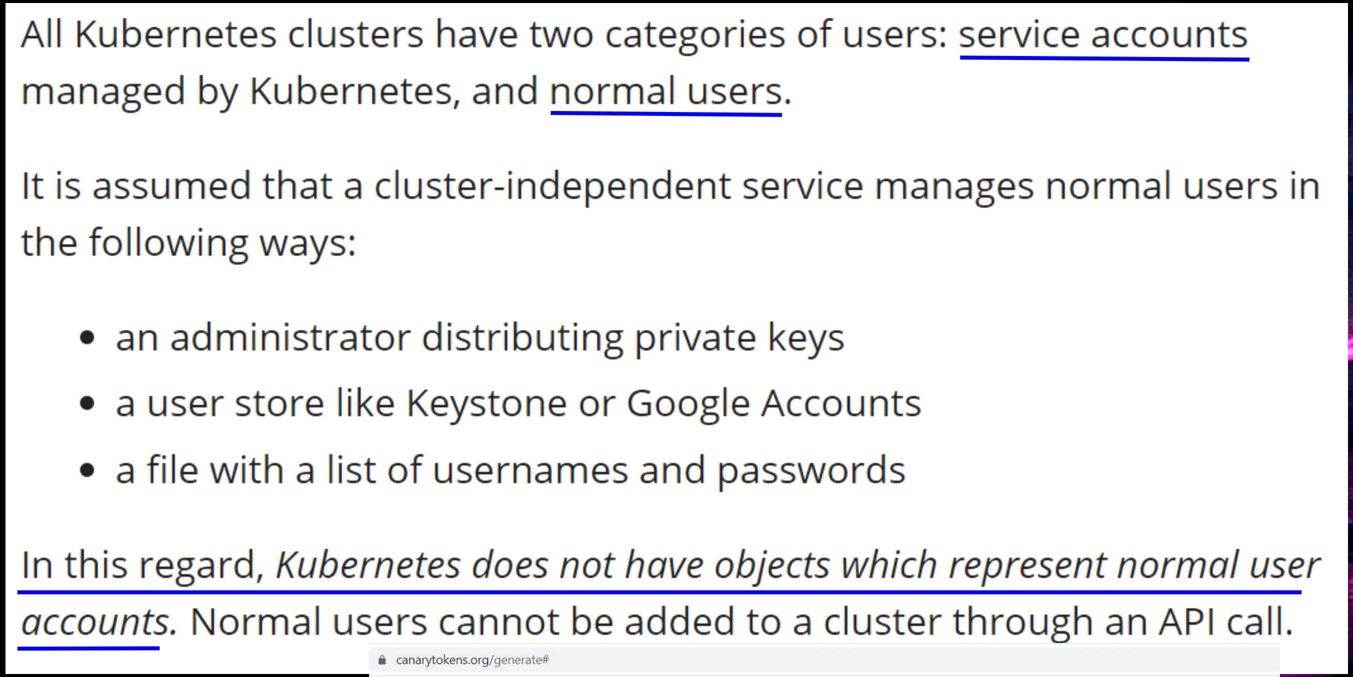


What is Peirates?

Peirates, a Kubernetes penetration tool, enables an attacker to escalate privilege and pivot through a Kubernetes cluster. It automates known techniques to steal and collect service account tokens, secrets, obtain further code execution, and gain control of the cluster.

Decoy User Credential





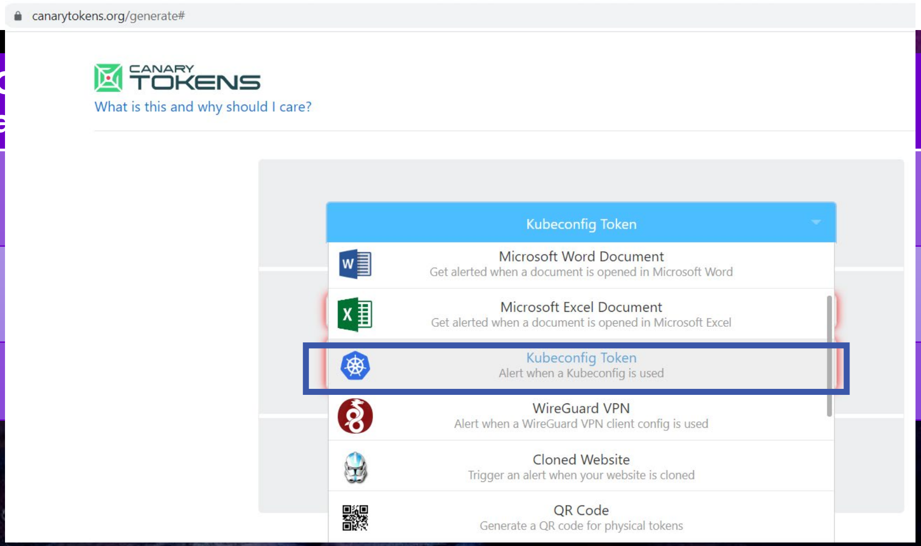
All Kubernetes clusters have two categories of users: service accounts managed by Kubernetes, and normal users.

It is assumed that a cluster-independent service manages normal users in the following ways:

- an administrator distributing private keys
- a user store like Keystone or Google Accounts
- a file with a list of usernames and passwords

In this regard, *Kubernetes does not have objects which represent normal user accounts.* Normal users cannot be added to a cluster through an API call.

Initial access	Execution	Persistence	Privilege escalation	Defense evasion	C a	Impact
Using cloud creds						
Kubeconfig file						



canarytokens.org/generate#

CANARY TOKENS

What is this and why should I care?

Kubeconfig Token

Microsoft Word Document
Get alerted when a document is opened in Microsoft Word

Microsoft Excel Document
Get alerted when a document is opened in Microsoft Excel

Kubeconfig Token
Alert when a Kubeconfig is used

WireGuard VPN
Alert when a WireGuard VPN client config is used

Cloned Website
Trigger an alert when your website is cloned

QR Code
Generate a QR code for physical tokens

OFFZONE 2022

Conclusions

LUNTRY

Conclusions

- Containers are awesome!
  - Speed, Isolation, Portability, …
- Containers orchestrated by Kubernetes are super awesome!
  - Kubernetes makes many processes easy
  - Declarative system
  - API-based approach
- Combine and trick adversaries in new ways ;)
  - You are only limited by your imagination
- Deception phase isn’t a silver bullet, but it’s a cool addon!
  - Defense in depth
  - Identify, Protect, Detect, Respond, Recover

Tank you for your attention!

OFF ONE 2022

Contacts:
- Email: de@luntry.ru
- Twitter: @evdokimovds
- Tg: @Qu3b3c
- Channel: @k8security
- Site: www.luntry.ru

LUNTRY

???

???

https://t.me/k8security/619 для обама доклада

Совсем недавно в рамках SANS Blue Team Summit 2021 был представлен доклад "DeTT&CT(ing) Kubernetes ATT&CK(s) with Audit Logs (https://www.sans.org/presentations/detecting-kubernetes-attacks-with-audit-logs/)" и сейчас доступны как слайды (https://sansorg.egnyte.com/dl/uzWJooP0Rl), так видео (https://www.youtube.com/watch?v=RwKbf8wqzpI&ab_channel=SANSCyberDefense) выступления.

По мне данный материал более наглядно (с примерами и привязками к техникам из MITRE ATT&CK) дополняет работы "Detection Engineering for Kubernetes clusters (https://t.me/k8security/450)" и "Threat Hunting with Kubernetes Audit Logs (https://t.me/k8security/393)", о которых я писал ранее.

В конце, автор еще немного показывает, как со всеми этими логами можно работать в Splunk.

Idea: Virtual ControlPlane

???

Чтобы злоумышленник общался с Kubernetes API, но не с production

Special k8s entities

Special ;)

- A separate Kubernetes cluster or Node with decoys and extra control
- Pod that does not have any interactions
  - Consider NetworkPolicy
- Special Secret
  - Secrets Store CSI Driver to add critical info types to containers
- Special DNS records
  - Known names and services, like Tiller
  - Interfaces: Apache NiFi, Kubeflow, Argo Workflows, Weave Scope, and the Kubernetes dashboard.
- Fake CRDs that can be seen with Default ServiceAccount

Что запускать/располагать дл мс

control plane node etcd

pod deploy ds

То к чему не должно быть обращений и вза тв

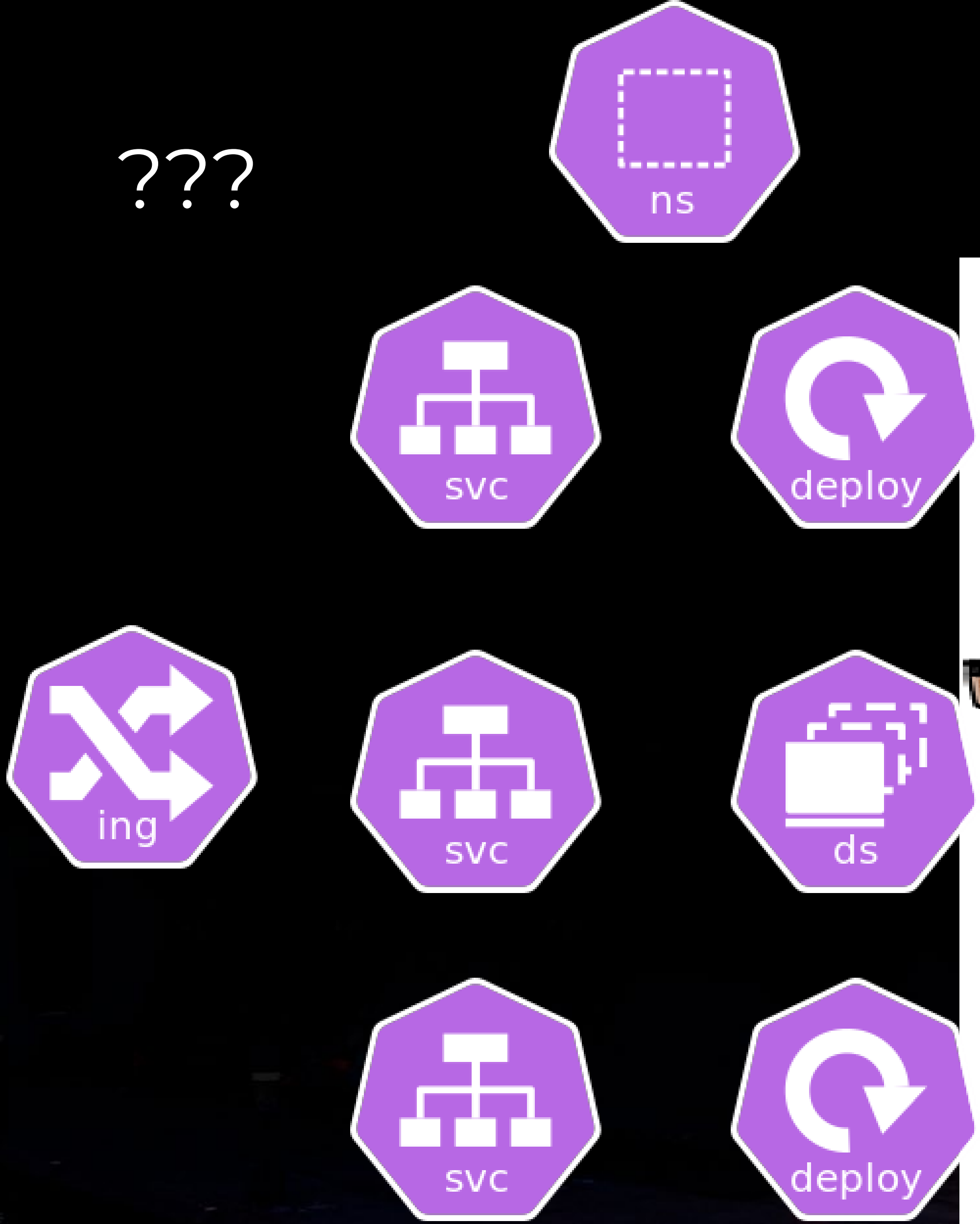
- Kubernetes cluster
- Nodes
- Pod/Workload
  - Учитывать NetworkPolicy
- Secret
  - Secrets Store CSI Driver для ма а критичной инфы в контейнер
- DNS записи
  - Известные имена и сервисы типа Tiller
  - ИНтерфейсы: Apache NiFi, Kubeflow, Argo Workflows, Weave Scope, and the Kubernetes dashboard.

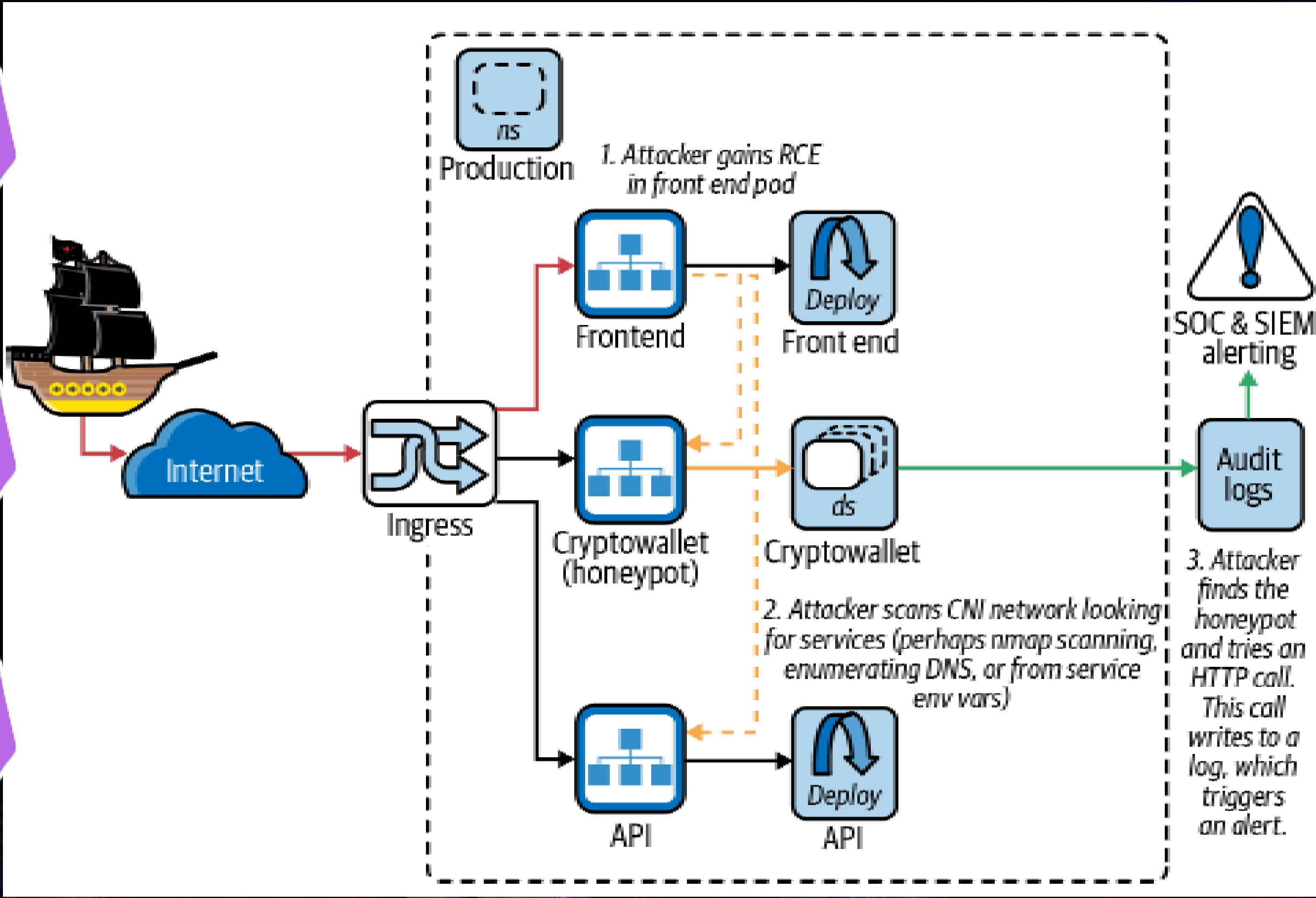
secret

- ециа I Ing ervices, Endpoints
  - если к нему идет обращение то блокировать IP
- Поддельные CRD
  - в системе что видны с помощью Default ServiceAccount
- Специальные Ports от хорошо известных решений
  - Tiller Port TCP/44134

ing svc ep crd

???


ns
svc
deploy
ing
svc
ds
svc
deploy


ns
Production
1. Attacker gains RCE in front end pod
Frontend
Deploy
Front end
SOC & SIEM alerting
Internet
Ingress
Cryptowallet (honeypot)
ds
Cryptowallet
Audit logs
2. Attacker scans CNI network looking for services (perhaps nmap scanning, enumerating DNS, or from service env vars)
3. Attacker finds the honeypot and tries an HTTP call. This call writes to a log, which triggers an alert.
API
Deploy
API

Links

???

- https://github.com/stone-z/canarytokens-k8s
- https://blog.thinkst.com/2021/11/a-kubeconfig-canarytoken.html
- https://app.bountysource.com/teams/canarytokens-docker/issues