dstny Engage

DEMO#3

Security : Mariam Osama

Business Value

The intended customer doesn't need to have the background to integrate the cluster and security is a very essential side of any business. Based on that the customer adjures that the security tool is part of the cluster and that this tool helps detect the existing vulnerabilities and proposes the means to solve it.

dstny Engage

SPRINT 2

Recap

Scanning Tool

The Kubernetes Cluster Scanner Tool is a utility designed to perform security scans on Kubernetes deployments using various security tools. This tool automates the scanning process across different namespaces, identifying vulnerabilities and security risks in your Kubernetes environment.

- Trivy :

Comprehensive Vulnerability Database: Trivy maintains a vast database of vulnerabilities, making it highly effective in identifying security issues.

Container Image Scanning: Trivy scans container images for vulnerabilities, a crucial step in securing Kubernetes deployments.

Regular Updates: Trivy is actively maintained and updated to keep pace with emerging vulnerabilities.

Open Source: Trivy is an open-source tool, making it accessible and customizable for our needs

- Kube-hunter:

Active Scanning: Kube-hunter actively probes Kubernetes clusters to discover potential security weaknesses.

Ethical Hacking: It simulates real-world attacks, helping us identify vulnerabilities before malicious actors do.

Community-Driven: Kube-hunter is community-driven, benefiting from a wide range of contributors and expertise.

Insights into Cluster Weaknesses: It provides insights into various attack vectors and cluster configurations.

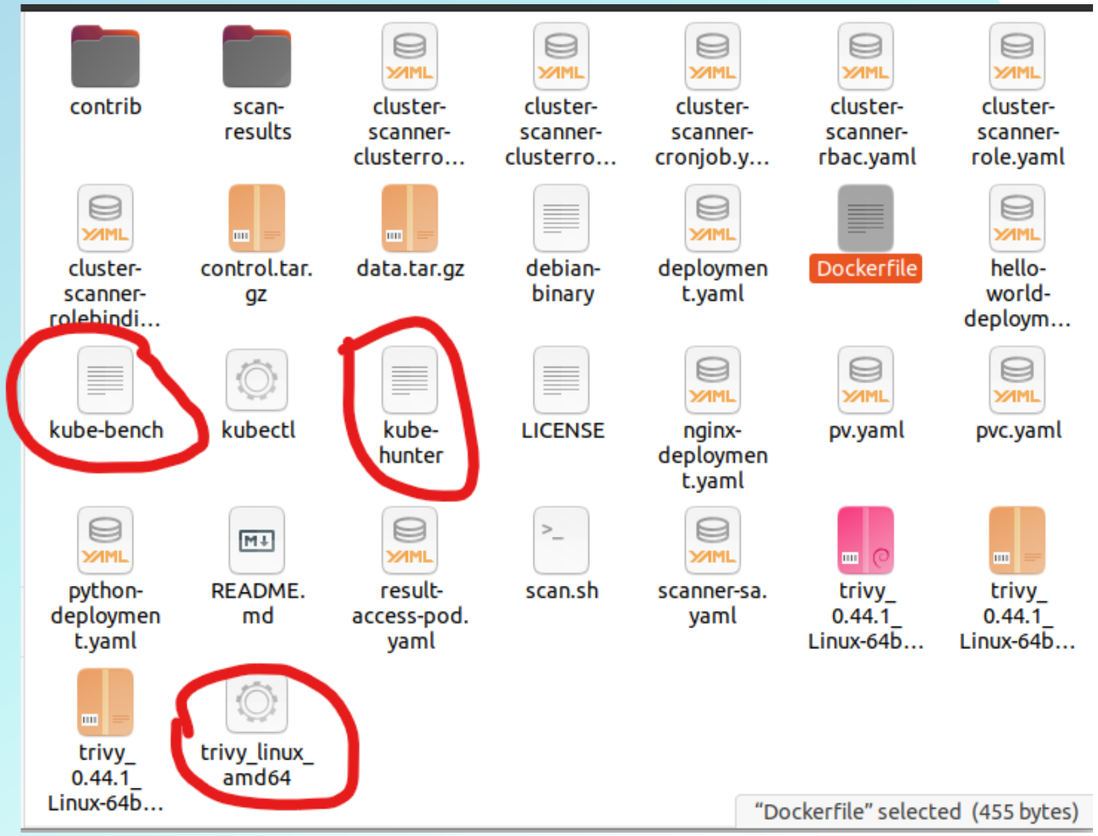
- Kube-bench:

CIS Benchmark Compliance: Kube-bench assesses Kubernetes clusters against CIS (Center for Internet Security) benchmarks, ensuring compliance with industry best practices.

Hardening Guidance: It offers guidance on hardening the cluster's security posture.

Automation: Kube-bench automates the benchmarking process, saving time and ensuring consistency.

Widely Adopted: Kube-bench is widely adopted by the Kubernetes community and recognized as a standard for cluster security.




```
FROM alpine:3.15.3

WORKDIR /app

RUN apk add --no-cache wget
RUN mkdir /app/scan-results

COPY trivy_linux_amd64 /usr/local/bin/trivy
COPY kube-bench /app/
COPY kube-hunter /app/

COPY scan.sh .

RUN chmod +x scan.sh
RUN chmod +x /usr/local/bin/trivy

CMD ["./scan.sh"]
```

```
# Function to print a message with timestamp
print_with_timestamp() {
    echo "$(date -u +"%Y-%m-%dT%H:%M:%SZ") $1"
}

# Install kubectl if not present
if ! command -v kubectl &> /dev/null; then
    print_with_timestamp "kubectl not found, installing..."
    wget -O /usr/local/bin/kubectl https://storage.googleapis.com/kubernetes-release/release/v1.22.2/bin/linux/amd64/kubectl
    chmod +x /usr/local/bin/kubectl
fi

print_with_timestamp "Cluster scan started."

# Iterate through all namespaces
for ns in $(kubectl get namespaces -o=jsonpath='{.items[*].metadata.name}'); do
    print_with_timestamp "Scanning namespace: $ns"

    # Iterate through deployments in the namespace
    for deployment in $(kubectl get deployments -n $ns -o=jsonpath='{.items[*].metadata.name}'); do
        print_with_timestamp "Scanning deployment: $deployment"

        # Run vulnerability scan using Trivy
        trivy_image=$(kubectl get deployment $deployment -n $ns -o=jsonpath='{.spec.template.spec.containers[0].image}')
        trivy_results="/app/scan-results/${ns}_${deployment}_$(date -u +"%Y%m%d%H%M%S")_trivy-results.json"
        trivy image --cache-dir /tmp/trivy --format json -o $trivy_results $trivy_image

        # Run kube-bench
        kube_bench_results="/app/scan-results/${ns}_${deployment}_$(date -u +"%Y%m%d%H%M%S")_kube-bench-results.json"
        ./kube-bench > $kube_bench_results

        # Run kube-hunter
        kube_hunter_results="/app/scan-results/${ns}_${deployment}_$(date -u +"%Y%m%d%H%M%S")_kube-hunter-results.json"
        ./kube-hunter --remote > $kube_hunter_results
```

```
/app/scan-results # ls
default_cluster-scanner_20230824160124_trivy-results.json
default_cluster-scanner_20230824160418_kube-bench-results.json
default_cluster-scanner_20230824160418_kube-hunter-results.json
default_python-deployment_20230824160418_trivy-results.json
default_python-deployment_20230824160749_kube-bench-results.json
default_python-deployment_20230824160749_kube-hunter-results.json
hello_hello-world-deployment_20230824160750_trivy-results.json
hello_hello-world-deployment_20230824160846_kube-bench-results.json
hello_hello-world-deployment_20230824160846_kube-hunter-results.json
kube-system_coredns_20230824160846_trivy-results.json
kube-system_coredns_20230824160916_kube-bench-results.json
kube-system_coredns_20230824160916_kube-hunter-results.json
kubernetes-dashboard_dashboard-metrics-scraper_20230824160916_trivy-results.json
kubernetes-dashboard_dashboard-metrics-scraper_20230824161020_kube-bench-results.json
kubernetes-dashboard_dashboard-metrics-scraper_20230824161020_kube-hunter-results.json
```


```
        {
          "VulnerabilityID": "CVE-2022-40897",
          "PkgName": "setuptools",
          "PkgPath": "usr/local/lib/python3.9/site-packages/setuptools-58.1.0.dist-info/METADATA",
          "InstalledVersion": "58.1.0",
          "FixedVersion": "65.5.1",
          "Status": "fixed",
          "Layer": {
            "Digest": "sha256:d271c014c3a0adbd37dae2ce6245711e2876ace65aa5c9b7859a6c515063c45f",
            "DiffID": "sha256:ddab61a14989358634f27d99fed2910a51c05aba9f83ec630386edcf80c8bb53"
          },
          "SeveritySource": "nvd",
          "PrimaryURL": "https://avd.aquasec.com/nvd/cve-2022-40897",
          "DataSource": {
            "ID": "osv",
            "Name": "Python Packaging Advisory Database",
            "URL": "https://github.com/pypa/advisory-db"
          },
          "Title": "Regular Expression Denial of Service (ReDoS) in package_index.py",
          "Description": "Python Packaging Authority (PyPA) setuptools before 65.5.1 allows remote attackers to cause a denial of service via HTML in a crafted package
or custom PackageIndex page. There is a Regular Expression Denial of Service (ReDoS) in package_index.py.",
          "Severity": "MEDIUM",
          "CweIDs": [
            "CWE-1333"
          ],
          "CVSS": {
            "ghsa": {
              "V3Vector": "CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H",
              "V3Score": 7.5
            },
            "nvd": {
```

dstny Engage

SPRINT 3

What is done

1.Presitent Volume

Initially

I used static provisioning

Open

pv.yaml
~/Desktop/cluster-scanner

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: scan-results-pv
spec:
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  hostPath:
    path: /app/scan-results
```

Open

pvc.yaml
~/Desktop/cluster-scanner

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: scan-results-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
```

Currently

Using dynamic provisioning

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: scan-results-pvc
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: longhorn
  resources:
    requests:
      storage: 1Gi
```

```
1 FROM alpine:3.15.3
2
3 WORKDIR /app
4
5 RUN apk add --no-cache wget
6 RUN mkdir /app/scan-results
7
```

```
1 #!/bin/sh
2
3
4 # Install kubectl if not present
5 if ! command -v kubectl &> /dev/null; then
6     echo "kubectl not found, installing..."
7     wget -O /usr/local/bin/kubectl https://storage.googleapis.com/kubernetes-release/release/v1.22.2/bin/linux/amd64/kubectl
8     chmod +x /usr/local/bin/kubectl
9 fi
10
11 # Specify the directory for storing scan results within the pod
12 RESULTS_DIR="/app/scan-results"
13
14 echo "-----------------------------------------------------------------------Scanning Tool Starting-----------------------------------------------------------------------
15 # Iterate through all namespaces
16 for ns in $(kubectl get namespaces -o=jsonpath='{.items[*].metadata.name}'); do
17     echo "Scanning namespace: $ns"
18
19     # Iterate through deployments in the namespace
20     for deployment in $(kubectl get deployments -n $ns -o=jsonpath='{.items[*].metadata.name}'); do
21         echo "Scanning deployment: $deployment"
22
23         # Run vulnerability scan using Trivy
24         trivy_image=$(kubectl get deployment $deployment -n $ns -o=jsonpath='{.spec.template.spec.containers[0].image}')
25         trivy_results="${ns}_${deployment}_$(date -u +"%Y%m%d%H%M%S")_trivy-results.json"
26         trivy image --cache-dir /tmp/trivy --format json -o "$RESULTS_DIR/$trivy_results" "$trivy_image"
27
28         echo "Scanning completed for deployment: $deployment"
29     done
30 done
31         # Run kube-bench
32         kube_bench_results="$(date -u +"%Y%m%d%H%M%S")_kube-bench-results.json"
33         ./kube-bench > "$RESULTS_DIR/$kube_bench_results"
34
35         # Run kube-hunter
36         kube_hunter_results="$(date -u +"%Y%m%d%H%M%S")_kube-hunter-results.json"
37         ./kube-hunter --remote > "$RESULTS_DIR/$kube_hunter_results"
38
39
40
41 echo "-----------------------------------------------------------------------------Scanning Tool Done-----------------------------------------------------------------------
42 echo "-----------------------------------------------------------------Look For the results in result acess pod-----------------------------------------------------------
```

2.Deploying In Cluster

```
satrtingg
Scanning namespace: cattle-fleet-clusters-system
Scanning namespace: cattle-fleet-system
Scanning deployment: fleet-controller
2023-09-08T16:21:58.506Z        INFO    Vulnerability scanning is enabled
2023-09-08T16:21:58.506Z        INFO    Secret scanning is enabled
2023-09-08T16:21:58.506Z        INFO    If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2023-09-08T16:21:58.506Z        INFO    Please see also https://aquasecurity.github.io/trivy/v0.44/docs/scanner/secret/#recommendation for faster secret detection
2023-09-08T16:21:58.730Z        INFO    Detected OS: suse linux enterprise server
2023-09-08T16:21:58.732Z        INFO    Detecting SUSE vulnerabilities...
2023-09-08T16:21:58.737Z        INFO    Number of language-specific files: 1
2023-09-08T16:21:58.738Z        INFO    Detecting gobinary vulnerabilities...
Scanning completed for deployment: fleet-controller
Scanning deployment: gitjob
2023-09-08T16:22:00.942Z        INFO    Vulnerability scanning is enabled
2023-09-08T16:22:00.943Z        INFO    Secret scanning is enabled
2023-09-08T16:22:00.943Z        INFO    If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2023-09-08T16:22:00.943Z        INFO    Please see also https://aquasecurity.github.io/trivy/v0.44/docs/scanner/secret/#recommendation for faster secret detection
2023-09-08T16:22:01.205Z        INFO    Detected OS: suse linux enterprise server
2023-09-08T16:22:01.206Z        INFO    Detecting SUSE vulnerabilities...
2023-09-08T16:22:01.210Z        INFO    Number of language-specific files: 1
2023-09-08T16:22:01.210Z        INFO    Detecting gobinary vulnerabilities...
Scanning completed for deployment: gitjob
Scanning namespace: cattle-global-data
Scanning namespace: cattle-global-nt
Scanning namespace: cattle-impersonation-system
Scanning namespace: cattle-system
Scanning deployment: rancher
2023-09-08T16:22:04.006Z        INFO    Vulnerability scanning is enabled
2023-09-08T16:22:04.006Z        INFO    Secret scanning is enabled
2023-09-08T16:22:04.006Z        INFO    If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2023-09-08T16:22:04.006Z        INFO    Please see also https://aquasecurity.github.io/trivy/v0.44/docs/scanner/secret/#recommendation for faster secret detection
2023-09-08T16:22:04.376Z        INFO    Detected OS: suse linux enterprise server
2023-09-08T16:22:04.378Z        INFO    Detecting SUSE vulnerabilities...
2023-09-08T16:22:04.383Z        INFO    Number of language-specific files: 16
2023-09-08T16:22:04.383Z        INFO    Detecting gomod vulnerabilities...
```

Tool running inside the cluster

****in its own namespace

```
/app/scan-results # ls
cattle-fleet-system_fleet-controller_20230908162101_trivy-results.json
cattle-fleet-system_fleet-controller_20230908162158_kube-bench-results.json
cattle-fleet-system_fleet-controller_20230908162158_kube-hunter-results.json
cattle-fleet-system_gitjob_20230908162158_trivy-results.json
cattle-fleet-system_gitjob_20230908162201_kube-bench-results.json
cattle-fleet-system_gitjob_20230908162201_kube-hunter-results.json
cattle-system_rancher-webhook_20230908162204_trivy-results.json
cattle-system_rancher-webhook_20230908162206_kube-bench-results.json
cattle-system_rancher-webhook_20230908162206_kube-hunter-results.json
cattle-system_rancher_20230908162201_trivy-results.json
cattle-system_rancher_20230908162204_kube-bench-results.json
cattle-system_rancher_20230908162204_kube-hunter-results.json
cert-manager_cert-manager-cainjector_20230908162209_trivy-results.json
cert-manager_cert-manager-cainjector_20230908162210_kube-bench-results.json
cert-manager_cert-manager-cainjector_20230908162210_kube-hunter-results.json
cert-manager_cert-manager-webhook_20230908162210_trivy-results.json
cert-manager_cert-manager-webhook_20230908162212_kube-bench-results.json
cert-manager_cert-manager-webhook_20230908162212_kube-hunter-results.json
cert-manager_cert-manager_20230908162206_trivy-results.json
cert-manager_cert-manager_20230908162209_kube-bench-results.json
cert-manager_cert-manager_20230908162209_kube-hunter-results.json
kube-system_calico-kube-controllers_20230908162213_trivy-results.json
kube-system_calico-kube-controllers_20230908162215_kube-bench-results.json
kube-system_calico-kube-controllers_20230908162215_kube-hunter-results.json
kube-system_coredns-autoscaler_20230908162218_trivy-results.json
kube-system_coredns-autoscaler_20230908162220_kube-bench-results.json
kube-system_coredns-autoscaler_20230908162220_kube-hunter-results.json
kube-system_coredns_20230908162215_trivy-results.json
kube-system_coredns_20230908162218_kube-bench-results.json
kube-system_coredns_20230908162218_kube-hunter-results.json
kube-system_metrics-server_20230908162221_trivy-results.json
kube-system_metrics-server_20230908162223_kube-bench-results.json
kube-system_metrics-server_20230908162223_kube-hunter-results.json
longhorn-system_csi-attacher_20230908162224_trivy-results.json
longhorn-system_csi-attacher_20230908162226_kube-bench-results.json
longhorn-system_csi-attacher_20230908162226_kube-hunter-results.json
longhorn-system_csi-provisioner_20230908162226_trivy-results.json
longhorn-system_csi-provisioner_20230908162229_kube-bench-results.json
longhorn-system_csi-provisioner_20230908162229_kube-hunter-results.json
longhorn-system_csi-resizer_20230908162229_trivy-results.json
longhorn-system_csi-resizer_20230908162232_kube-bench-results.json
longhorn-system_csi-resizer_20230908162232_kube-hunter-results.json
longhorn-system_csi-snapshotter_20230908162232_trivy-results.json
longhorn-system_csi-snapshotter_20230908162242_kube-bench-results.json
longhorn-system_csi-snapshotter_20230908162242_kube-hunter-results.json
longhorn-system_longhorn-driver-deployer_20230908162243_trivy-results.json
longhorn-system_longhorn-driver-deployer_20230908162308_kube-bench-results.json
longhorn-system_longhorn-driver-deployer_20230908162308_kube-hunter-results.json
longhorn-system_longhorn-ui_20230908162309_trivy-results.json
longhorn-system_longhorn-ui_20230908162311_kube-bench-results.json
longhorn-system_longhorn-ui_20230908162311_kube-hunter-results.json
lost+found
metallb-system_metallb-controller_20230908162311_trivy-results.json
metallb-system_metallb-controller_20230908162323_kube-bench-results.json
metallb-system_metallb-controller_20230908162323_kube-hunter-results.json
monitoring_monitoring-grafana_20230908162324_trivy-results.json
monitoring_monitoring-grafana_20230908162326_kube-bench-results.json
monitoring_monitoring-grafana_20230908162326_kube-hunter-results.json
monitoring_monitoring-kube-prometheus-operator_20230908162326_trivy-results.json
monitoring_monitoring-kube-prometheus-operator_20230908162328_kube-bench-results.json
monitoring_monitoring-kube-prometheus-operator_20230908162328_kube-hunter-results.json
monitoring_monitoring-kube-state-metrics_20230908162329_trivy-results.json
monitoring_monitoring-kube-state-metrics_20230908162330_kube-bench-results.json
monitoring_monitoring-kube-state-metrics_20230908162330_kube-hunter-results.json
/app/scan-results #
```

Files saved in persistent volume

***in result access pod

```
        {
          "VulnerabilityID": "SUSE-SU-2023:2224-1",
          "PkgID": "libcurl4@7.79.1-150400.5.18.1.x86_64",
          "PkgName": "libcurl4",
          "InstalledVersion": "7.79.1-150400.5.18.1",
          "FixedVersion": "8.0.1-150400.5.23.1",
          "Status": "fixed",
          "Layer": {
            "Digest": "sha256:a044828fb7b83c49cc759dbfac3542c94fc2c2a65ca8b222bb8d0bb8a9f2ce63",
            "DiffID": "sha256:52d58f02f883a2c3687215b71f7679b48d75f52564c248ade22d8ebfc0514b91"
          },
          "SeveritySource": "suse-cvrf",
          "DataSource": {
            "ID": "suse-cvrf",
            "Name": "SUSE CVRF",
            "URL": "https://ftp.suse.com/pub/projects/security/cvrf/"
          },
          "Title": "Security update for curl",
          "Description": "This update for curl adds the following feature:\n\nUpdate to version 8.0.1 (jsc#PED-2580)\n\n- CVE-2023-28319: use-after-free in SSH sha256 fingerprint check (bsc#1
23-28320: siglongjmp race condition (bsc#1211231).\n- CVE-2023-28321: IDN wildcard matching (bsc#1211232).\n- CVE-2023-28322: POST-after-PUT confusion (bsc#1211233).\n",
          "Severity": "HIGH",
          "References": [
            "https://bugzilla.suse.com/1211230",
            "https://bugzilla.suse.com/1211231",
            "https://bugzilla.suse.com/1211232",
            "https://bugzilla.suse.com/1211233",
            "https://lists.suse.com/pipermail/sle-security-updates/2023-May/014913.html",
            "https://www.suse.com/security/cve/CVE-2023-28319/",
            "https://www.suse.com/security/cve/CVE-2023-28320/",
            "https://www.suse.com/security/cve/CVE-2023-28321/",
            "https://www.suse.com/security/cve/CVE-2023-28322/",
            "https://www.suse.com/support/security/rating/",
            "https://www.suse.com/support/update/announcement/2023/suse-su-20232224-1/"
          ]
        },
        {
          "VulnerabilityID": "SUSE-SU-2023:2891-1",
          "PkgID": "libcurl4@7.79.1-150400.5.18.1.x86_64",
          "PkgName": "libcurl4",
          "InstalledVersion": "7.79.1-150400.5.18.1",
          "FixedVersion": "8.0.1-150400.5.26.1",
          "Status": "fixed",
          "Layer": {
            "Digest": "sha256:a044828fb7b83c49cc759dbfac3542c94fc2c2a65ca8b222bb8d0bb8a9f2ce63",
            "DiffID": "sha256:52d58f02f883a2c3687215b71f7679b48d75f52564c248ade22d8ebfc0514b91"
          },
          "SeveritySource": "suse-cvrf",
          "DataSource": {
            "ID": "suse-cvrf",
            "Name": "SUSE CVRF",
            "URL": "https://ftp.suse.com/pub/projects/security/cvrf/"
          },
          "Title": "Security update for curl",
```

dstny Engage

SPRINT 4

In the progress

1.Analyzing and Patching

Analyze the results from the scanning tool filtering and prioritizing the vulnerabilities to be patched then recommending the patch management.

```python

import json

# Load Trivy results from JSON file
with open('default_cluster-scanner_20230905175015_trivy-results.json', 'r') as f:
    trivy_data = json.load(f)

# Extract vulnerabilities
vulnerabilities = trivy_data.get("Results", [])

# Filter high-severity vulnerabilities
high_severity_vulnerabilities = []

for result in vulnerabilities:
    if "Vulnerabilities" in result:
        for vulnerability in result["Vulnerabilities"]:
            if vulnerability.get("Severity") == "HIGH":
                high_severity_vulnerabilities.append(vulnerability)

# Print the number of high severity vulnerabilities
print(f"Total High Severity Vulnerabilities: {len(high_severity_vulnerabilities)}")

# Print details of high severity vulnerabilities
for idx, vulnerability in enumerate(high_severity_vulnerabilities, start=1):
    print(f"\nHigh Severity Vulnerability {idx}:")
    print(f"Vulnerability ID: {vulnerability.get('VulnerabilityID')}")
    print(f"Package Name: {vulnerability.get('PkgName')}")
    print(f"Installed Version: {vulnerability.get('InstalledVersion')}")
    print(f"Fixed Version: {vulnerability.get('FixedVersion')}")
    print(f"Description: {vulnerability.get('Description')}")
    print(f"CVSS Score: {vulnerability['CVSS']['bitnami']['V3Score']}")
    print(f"More Info: {vulnerability.get('PrimaryURL')}\n")

```

Making a script that will automate this process

2.ZAP OWASP

OWASP ZAP (Zed Attack Proxy) is a powerful and widely-used open-source security testing tool designed for finding vulnerabilities in web applications during development and testing. ZAP operates as a proxy server, intercepting and inspecting the communication between a user's browser and a web application. Its primary mission is to help developers, security testers, and organizations identify and mitigate security risks early in the software development lifecycle. ZAP provides a comprehensive suite of features, including automated scanning, manual testing, and a variety of tools for discovering common web application security issues, such as cross-site scripting (XSS), SQL injection, and insecure configurations. It also allows users to customize their tests, generate reports, and integrate with CI/CD pipelines. OWASP ZAP is a valuable resource for bolstering web application security and ensuring that vulnerabilities are addressed proactively.

What next:

- exploring the tool and getting familiar with it
- planning on how it will be used in the most efficient way in order to test the security of the clusters API
- eventually applying the planned test cases

THANK YOU

dstny

Engage