Pipeline Security a Catalyst for DevSecOps
Gets told no a lot

Had lots of outages
What are we talking about today?

- Security, everyone’s responsibility?!
- Shift left into DevSecOps
- Build security into your CI/CD Pipelines
Security?
Well Duh!
Which Ones do you Trust?
Application is Always Trusted
Why so?

@ravilach
Our Domain

cassandra
Shift Left to DevSecOps
Some Terms

R&M stands for Rick and Morty

Abbreviations.com
National Vulnerability Database [NVD]

The NVD is the U.S. government repository of standards based vulnerability management data represented using the Security Content Automation Protocol (SCAP). This data enables automation of vulnerability management, security measurement, and compliance. The NVD includes databases of security checklist references, security-related software flaws, misconfigurations, product names, and impact metrics.
Common Vulnerabilities and Exposures [CVE]

CVE-2018-1002105 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

Current Description

In all Kubernetes versions prior to v1.10.11, v1.11.5, and v1.12.3, incorrect handling of error responses to proxied upgrade requests in the kube-apiserver allowed specially crafted requests to establish a connection through the Kubernetes API server to backend servers, then send arbitrary requests over the same connection directly to the backend, authenticated with the Kubernetes API server’s TLS credentials used to establish the backend connection.

Source: MITRE

View Analysis Description

Severity

CVSS Version 3.x  CVSS Version 2.0

CVSS 3.x Severity and Metrics:

Common Vulnerability Scoring System [CVSS]
Shifting Left
DevSecOps

@Override
public void apply(Project project) {
    PluginContainer plugins = project.getPlugins();
    plugins.apply(DeployedPlugin.class);
    plugins.apply(JavaLibraryPlugin.class);
    plugins.apply(ConventionPlugin.class);
    plugins.apply(InternalDependencyManagementPlugin.class);
    StarterMetadata starterMetadata = project.getTasks().create("starterMetadata", StarterMetadata.class);
    ConfigurationContainer configurations = project.getConfigurations();
    Configuration runtimeClasspath = configurations.getByName(JavaPlugin.RUNTIME_CLASSPATH_CONFIGURATION_NAME);
    starterMetadata.setDependencies(runtimeClasspath);
    File destination = new File(project.getBuildDir(), "starter-metadata.properties");
Static Analysis [SAST]

Or

@ravilach
Dynamic Analysis [DAST]
RASPs
Your Pipelines
Your Pipelines
Least Privilege

"The Bear Necessities"
Role Based Access
Audit Trail

The image shows a JSON object that represents an audit trail, likely from a software system or platform. The code snippet contains object properties such as `id`, `triggeredAt`, `requestMethod`, `resourcePath`, and more, which are part of the audit trail data. The JSON object is structured as follows:

```json
{
  "data": {
    "audits": {
      "nodes": [
      {
        "id": "9mQoXgg5dG-mmDAy4Nh",
        "triggeredAt": 1581532160464,
        "request": {
          "url": "https://app.harness.io/api/app/DESCRIPTION,",
          "resourcePath": "apps/DESCRIPTION",
          "requestMethod": "DELETE",
          "remoteIpAddress": "192.195.83.11"
        }
      }
    ]
  }
}
```
Security Starts with YOU
Some Cool Examples

- Amazon EC2
- Sample Node App
- K8s Delegate

Webgoat Build Report

<table>
<thead>
<tr>
<th>Target</th>
<th>Policy</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security-Critical</td>
<td>com.thoughtworks.owasp:owasp-quickstart:1.4.5</td>
<td></td>
</tr>
<tr>
<td>Security-Critical</td>
<td>jQuery:1.6.4</td>
<td></td>
</tr>
<tr>
<td>Security-Critical</td>
<td>org.springframework:spring-web:5.2.2.RELEASE</td>
<td></td>
</tr>
<tr>
<td>Security-Critical</td>
<td>org.springframework:spring-web:5.2.2.RELEASE</td>
<td></td>
</tr>
<tr>
<td>Security-Critical</td>
<td>org.springframework:spring-web:5.2.2.RELEASE</td>
<td></td>
</tr>
</tbody>
</table>
Thanks!