UBCIS

Ultimate Benchmark for Container Image Scanning

Preliminary work paper / Short

Shay Berkovich
Jeffrey Kam
Glenn Wurster

Security Research Group @ BlackBerry
Importance of Image Scanning

Traditional VM vs container image – where the source came from?

Unless you are building the base image by yourself, attacker has degree of access to your base images

Vulnerability propagation: reusability and deployability

FROM debian:latest
COPY ...
USER ...
docker pull debian
docker run -it debian bash
Scope

Container Image Scanning will solve:
Find known 3rd-party vulnerabilities in your product

Container Image Scanning will NOT solve:
Detect undiscovered vulnerabilities
Protect against malicious containers
Protect against backdoors
Solve container runtime security problems
1. None of these sets is a superset of all others
2. None of these sets encompasses ALL image vulnerabilities
3. Each set contains at least one FP
Component detection (libraries, packages, software)

Vulnerability matching

Reporting
Benchmark

Fedora:29
Ubuntu:18.10
Debian:10.2
Alpine:3:9:4
CentOS:7.7.1908

Feeds

Manual judging
Stability

- Not very old – should not contain 100s of vulns
- Not under active maintenance
- Benchmark will still change as new vulnerabilities are discovered for existing package versions

Popularity

Relevance

“A Study of Security Vulnerabilities on Docker Hub”,
Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, 2017
Benchmark

- 98 manually-judged Debian vulns
- 72 manually-judged Ubuntu vulns
- 10 manually-judged Alpine vulns
- 662 CentOS vulns
- 211 Fedora vulns
Applicability Classes

TP – simple cases where vulnerability is clearly applicable
- Recently-found
- Triaged as unimportant

FP – vulnerability is clearly not applicable
- Backport Patches
- Non-functional packages
- Kernel vulns
- Architecture mismatch

But what if there is something in the middle?
Applicability Classes

I = Inconclusive

▪ Untriaged

Ubuntu 12.04 ESM (Precise Pangolin): DNE
Ubuntu 14.04 ESM (Trusty Tahr): needs-triage
Ubuntu 16.04 LTS (Xenial Xerus): needed

CVE-2020-13776 (updated on June 24)

▪ Environmental – build arguments, usage pattern etc.

Notes
Fixed by: https://vcs.pcre.org/pcre?view=revision
Only an issue when UTF support disabled

CVE-2019-20838

▪ Invalidated by comments

Notes
https://gitlab.com/
No security impact,

CVE-2018-1000654

▪ Future unclear

Notes
NOT-FOR-US: Conceptual weakness in PGP keyserver design

CVE-2019-13050
# Applicability Classes

**MM** = different feeds disagree on fixed or affected package versions

<table>
<thead>
<tr>
<th>CVE-2018-12886</th>
<th>CVE-2020-10029</th>
</tr>
</thead>
<tbody>
<tr>
<td>From (including)</td>
<td>Up to (including)</td>
</tr>
<tr>
<td>4.1</td>
<td>8.0</td>
</tr>
<tr>
<td>buster</td>
<td>8.3.0-6</td>
</tr>
</tbody>
</table>
bullseye, sid   | 8.4.0-4         | vulnerable|

**D** = disputed by maintainers

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>CVE-2019-9192</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DISPUTED</strong></td>
<td>In the GNU C Library (\1)* in grep, a different issue than CVE</td>
<td></td>
</tr>
</tbody>
</table>
The truth is …

Many times, even when having all the vulnerability reports, scanners don’t know whether vulnerability is applicable or not.

The decision of what to do with I / MM / D is environmental.
Quality metrics:

- Precision = TP / (TP+FP)
- Recall = TP / (TP+FN)
- F-measure = 2 * Precision * Recall / (Recall + Precision)
# UBCIS – Ultimate Benchmark for Container Image Scanners

<table>
<thead>
<tr>
<th>System</th>
<th>Trivy</th>
<th>Anchore</th>
<th>Clair</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
<td>F-measure</td>
</tr>
<tr>
<td>Debian-10.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relaxed</td>
<td>0.78</td>
<td>0.98</td>
<td>0.87</td>
</tr>
<tr>
<td>ParanoI</td>
<td>1.00</td>
<td>0.69</td>
<td>0.82</td>
</tr>
<tr>
<td>Alpine-3.9.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relaxed</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Paranoid</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Ubuntu-18.10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relaxed</td>
<td>NA</td>
<td>0.00</td>
<td>NA</td>
</tr>
<tr>
<td>Paranoid</td>
<td>NA</td>
<td>0.00</td>
<td>NA</td>
</tr>
</tbody>
</table>

*as of 30/06/2020

Relaxed: I / MM / D => FP
Paranoid: I / MM / D => TP
Practical Recommendations
Before Choosing Scanner

Assess risk tolerance

- Can you afford to miss vulnerabilities (relaxed mode), or must treat all vulnerabilities as potentially critical (paranoid mode)?
- Scanners with a better paranoid mode score will generally raise more alerts, requiring more resources.
Before Choosing Scanner

Look at the deployment environment

- What base image are you using?
- Is the image supported by the scanner?
Before Choosing Scanner

Based on risk (#1) and base image (#2), use the benchmark results (Table 2) to select the appropriate scanner.
Before Choosing Scanner

From our experience, no image had zero vulnerabilities. A lack of vulnerabilities points to configuration problems or an unsupported image.
Before Choosing Scanner

Combining multiple scanners in a CI/CD pipeline is a good idea. Decide how to merge the results.
After Choosing Scanner

Keep watching the benchmark for changes.
A word on registry-embedded scanning

Relying on container registry scanning?

Amazon ECR (clair)

Harbor  (clair) starting v1.10  (trivy)

Sysdig  

Azure Container Registry  

Google Container Registry
Open Source

https://github.com/blackberry/UBCIS

Two modes:

- Superscan
- Scorer
Thank you for listening

@sshaybbc
sshayb