Automating Vulnerability Mapping from Tools

FIRST - PSIRT Technical Colloquium 2020 - March 4-5
About Us
NVIDIA Product Security Tools

Dee Annachhatre
NVIDIA Product Security
Tools Development
dannachhatre@nvidia.com

Jessica Butler
NVIDIA Product Security
Tools Development
jessicab@nvidia.com
Intro
Moving the Ball Forward

Pitfalls of manual process
All the Data - Oh My!
Cataloging Portfolio
Self-Service Registration tool
Mapping the Data - Oh Yeah!
Notifications
Issue Management - Yes, Please!
Calculating Risk
open source scanning
VULNERABILITY DETECTION
MANUAL PROCESS

1. REQUEST RECEIVED
2. GATHER SOURCE LOCATION
3. RUN SCANNING TOOL
4. MANUALLY GROUP DATA
5. SHIP REPORT TO REQUESTER
All the Data - OH MY!
Defining the PORTFOLIO

PRODUCTS
• Top level
• Shippable or deployable
• Executive ownership
• Versioning and EOL

COMPONENTS
• Logical segregation of product
• 1:n source code projects
• n:n products
• Build level ownership
• 1:n developer teams

DEPENDENCIES
• Internal components
• External open source software
• External third-party software
• Nestable

OPEN SOURCE SOFTWARE
• Versioning detection
• Vulnerability mapping
• Fix recommendations
• Fix verification
Self Service

REGISTRATION

Automating component mapping via build tool synchronization

1. PRODUCT INFO
2. 
3. 
4. 
5. 
6. 
Self Service

REGISTRATION

Automating component mapping via build tool synchronization

1. PRODUCT INFO
2. ADD/IMPORT CONTACTS

Contacts

Import Contacts from... Import

Name: Jessica Butler
Email: jessicab@nvidia.com

Active Directory Username: dannachatra

BU VP
Escalations
PLC Security Prime
PLC Security PIC

Primary contact for this component, product, platform, service...
Self Service

REGISTRATION

Automating component mapping via build tool synchronization

1. PRODUCT INFO
2. ADD/IMPORT CONTACTS
3. SYNC BUILD TOOLS

How is your project built?

We’ll try to synch with manifests or build scripts so we can automatically create software components.

- **Android Manifest**
- **Jenkins Project(s)**
  - [ ] DDX-Cloud-WebService-Deploy-Pod
  - [ ] DDX-Cloud-WebService-Deploy-Pod
  - [ ] NNQ-FFN-Healthcheck-Non-SEC

[Back] [Next]
Self Service

REGISTRATION

Automating component mapping via build tool synchronization

Components and Source Code Repositories

Great! You're almost done, just verify the results and manually add anything that we weren't able to pull.

Please verify the source code detected in the build tools entered before. Please enter source code location manually only if it was not found.

+ Enter source code location manually

Add and validate source code location(s).

source code location * branch name
FOR GIT SSH REQUIRED REQUIRED FOR GIT ONLY

Select valid source code location(s) to associate.

Select or add new component to add source code to.

Component

Submit

pytorch

git@github.com:pytorch/pytorch.git master
Self Service

REGISTRATION

Automating component mapping via build tool synchronization

Static Analysis Projects

Please add the mapping for static analysis projects to source code so we can enhance reports!

Add and validate static analysis project(s):

☐ This product does not use Static Analysis or I am not ready to provide the details.

server URI *
project name *
Type *
Validate

Select valid static analysis project(s) to associate.

Server URI
Project Name

checkman.nvidia.com
PyTorchDemo

Select source code to associate to static analysis project.

Source Code

Submit

PyTorchDemo

git@github.com:pytorch/pytorch.git master

BUILD
COMPONENTS

VERIFY
COMPONENTS

LINK
SERVICES
Self Service

REGISTRATION

Automating component mapping via build tool synchronization

1. PRODUCT INFO
2. ADD/IMPORT CONTACTS
3. SYNC BUILD TOOLS
4. VERIFY COMPONENTS
5. LINK SERVICES
6. REPORT ENABLED
SELF-SERVICE
REGISTRATION DEMO
Mapping the Data

TOOL 1: Open Source Scan
TOOL 2: SAST
TOOL 3: DAST

reporting micro-service
reporting micro-service
reporting micro-service

Portfolio DB

Reporting Tool
## Notifications

### OSS Scanning Service

<table>
<thead>
<tr>
<th>Description</th>
<th>Use Case</th>
<th>User Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Report</td>
<td>Dashboard of OSS vulnerability distribution and other details including recommended fix</td>
<td>Triggered the morning after successful registration</td>
</tr>
<tr>
<td>Weekly Report</td>
<td>Updated dashboard for scan results</td>
<td>Triggered every Monday morning</td>
</tr>
<tr>
<td>New CVE Report</td>
<td>Dashboard with newly discovered CVEs for packages in Portfolio</td>
<td>Triggered when a new CVE is introduced for an OSS package in Portfolio</td>
</tr>
<tr>
<td>Package Discovery alert</td>
<td>Email about a new undisclosed vulnerability associated with an OSS package</td>
<td>Triggered manually by PSIRT team about an undisclosed vulnerability associated with a particular OSS package</td>
</tr>
<tr>
<td>Portfolio Notification</td>
<td>Notifications regarding updates to the product catalog hierarchy</td>
<td>Registered build has been deleted/modified Underlying repositories have been deleted, Registered owners are invalidated</td>
</tr>
</tbody>
</table>
Issue Management

**Tool Policies**
1. Customize based on product types
2. Slowly increase strength for priority
3. Automate scan setup and configure

**De-duplicate Bugs**
1. Define filters for bug system
2. Reporting MSAs detect issue(s)
3. Portfolio DB determines owner(s)
4. Issue MSA validates new bug (and opens)

**Prioritization**
1. Portfolio DB detects code reuse
2. OSS Reporting MSA maps vulns
3. Report dashboard pinpoints biggest ROI

**Whitelisting**
1. Define format and location for list
2. Use team processes for approval
3. Require approval based on time
4. Synch to Reporting for validation
Reports

Overall Security Risk Profile

OSS - Severity Distribution across Vulnerabilities

OSS - Top 10 packages with high number of vulnerabilities Demo

SATS - Checkmarx Severity Distribution Demo

SAST - Checkmarx CWE Distribution
# Reports

## Security Risk Profile Details

### OSS - Vulnerability Details Demo

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>Severity</th>
<th>Version</th>
<th>Package</th>
<th>Repository</th>
<th>Location</th>
<th>Description</th>
<th>Fix</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2019-10744</td>
<td>Critical</td>
<td>4.17.10</td>
<td>iodash-4.17.10.tgz</td>
<td>ssh://git@github-master.nvidia.com:12051/pstooling/demos/node_example.git</td>
<td>/tmp/tmpbik0nw_d/source/node_modules/babel-types/node_modules/odash/package.json</td>
<td>Versions of iodash lower than 4.17.12 are vulnerable to Prototype Polution. The function <code>defaultsDeep</code> could be tricked into adding or modifying properties of <code>Object.prototype</code> using a constructor payload.</td>
<td>Upgrade to version 4.17.12</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Vulnerability</th>
<th>Severity</th>
<th>Version</th>
<th>Package</th>
<th>Repository</th>
<th>Location</th>
<th>Description</th>
<th>Fix</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2019-10744</td>
<td>Critical</td>
<td>4.17.10</td>
<td>iodash-4.17.10.tgz</td>
<td>ssh://git@github-master.nvidia.com:12051/pstooling/demos/node_example.git</td>
<td>/tmp/tmpbik0nw_d/source/node_modules/babel-types/node_modules/odash/package.json</td>
<td>Versions of iodash lower than 4.17.12 are vulnerable to Prototype Polution. The function <code>defaultsDeep</code> could be tricked into adding or modifying properties of <code>Object.prototype</code> using a constructor payload.</td>
<td>Upgrade to version 4.17.12</td>
<td>2</td>
</tr>
</tbody>
</table>

### SAST - Checkmarx Issue Details Demo

<table>
<thead>
<tr>
<th>CWE</th>
<th>Description</th>
<th>Severity</th>
<th>State</th>
<th>Status</th>
<th>Product</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>348</td>
<td>Missing_CSP_Header</td>
<td>Low</td>
<td>To Verify</td>
<td>Recurrent</td>
<td>LodashProject</td>
<td>1</td>
</tr>
<tr>
<td>348</td>
<td>Missing_HSTS_Header</td>
<td>Medium</td>
<td>To Verify</td>
<td>Recurrent</td>
<td>LodashProject</td>
<td>1</td>
</tr>
<tr>
<td>352</td>
<td>Potentially_Vulnerable_To_Xrf</td>
<td>Low</td>
<td>To Verify</td>
<td>Recurrent</td>
<td>LodashProject</td>
<td>1</td>
</tr>
<tr>
<td>457</td>
<td>Use_of_Uninitialized_Variable</td>
<td>Low</td>
<td>To Verify</td>
<td>New</td>
<td>PyTorchProject</td>
<td>1,397</td>
</tr>
<tr>
<td>123</td>
<td>Unchecked_Array_Index</td>
<td>Medium</td>
<td>To Verify</td>
<td>New</td>
<td>PyTorchProject</td>
<td>455</td>
</tr>
</tbody>
</table>
Calculating Risk

Things to THINK on...

- Number of issues should be normalized
  - more source code
  - more open source in use
- Risk should incorporate multiple factors
  - severity
  - count
  - type
  - disclosure date
  - publicity
  - higher risk products
- Risk should be easily visualized
  - ie. a product with 3 Med issues should look different than product with 10 Med issues, etc
THANK YOU!
Comments, Questions, Follow-UP!

We love chatting with other Security Tools developers to knowledge share. Please contact us if you’re interested in learning more or sharing!!

dannachhatre@nvidia.com
jessicab@nvidia.com