Shift RIGHT to Fix Bugs Earlier

Security in a DevOps World

John Steven

m1splacedsoul
Defining the hype
# Digital Transformation

## 5 Pillars

<table>
<thead>
<tr>
<th>TABLE STAKES</th>
<th>YOUR OWN VIETNAM</th>
<th>THE BRASS RING</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Outsourcing</strong></td>
<td><strong>Culture</strong></td>
<td><strong>Automation</strong></td>
</tr>
<tr>
<td>Delivered via CSP, OSS</td>
<td>Delivered via DevOps</td>
<td>Delivered via CI/CD</td>
</tr>
<tr>
<td>Highlights self-service</td>
<td>Highlights roles</td>
<td>Highlights tooling</td>
</tr>
<tr>
<td><strong>Effectively OUTSOURCES IT, Dev</strong></td>
<td><strong>Increase delivery responsiveness</strong></td>
<td><strong>emphasize automation</strong></td>
</tr>
<tr>
<td><strong>Self-service asset creation may make governance harder</strong></td>
<td><strong>Cadence stresses scan-based or thorough analyses</strong></td>
<td><strong>LoE can be overwhelming given brittle tech</strong></td>
</tr>
</tbody>
</table>
Defining ‘sw-defined’ security governance
Pillar activities of an ‘engineering-led’ SSI

• VISIBILITY INTO WHAT SOFTWARE/SERVICES ARE RUNNING
• INSIGHT INTO RUNNING SOFTWARE’S COMPOSITION (OSS, CLOUD API USAGE)
• AUTOMATIC EXECUTION OF AT LEAST MINIMUM REQUIRED VULNERABILITY DISCOVERY TESTING AUTOMATICALLY, WITH EACH RELEASE
• AGGREGATION AND SEARCHING OPERATIONAL DATA FOR MEANINGFUL SECURITY INFORMATION ACROSS A VALUE STREAM
• TRACING RUNNING SERVICES TO REPOSITORIES/BUILD/TEAM THAT PRODUCED THEM
• ENABLING ENGINEERING TO REMEDIATE SECURITY DEFECTS USING THE SAME TOOLS AND WORKFLOW AS OTHER TECHNICAL DEBT
• UPDATING NETWORK, HOST, CONTAINER, OR APPLICATION-LAYER CONFIGURATION THROUGH ORCHESTRATION
• INVALIDATING AND ROTATING SENSITIVE ASSETS (CREDENTIALS/KEY MATERIAL) WITHIN A DEPLOYMENT
• AUTOMATIC FAIL-OVER/ROLLBACK TO WORKING ASSETS OR KNOWN-GOOD WORKING CONFIGURATION/BUILD

‘QUICK N DIRTY’ CHECKLIST

Have/do you?:

1. Inventoried software, services, and data
2. Continuously discover asset base and attack surface
3. Integrated with pipeline platforms
4. Co-opted Orchestration for security
Centralized governance: via touchpoints
Bsimm9 view of transitional programs

- Threat model update
- Lightweight SAST tool
- Lightweight SCA tool

- Clean
- Compile
- Unit testing
- Functional tests
- Limited/incremental SAST

- Packaging
- Publishing
- Integration testing
- Risk-based security testing
- SCA

- Load testing
- Performance testing
- Comprehensive SAST
- DAST/IAST
- Penetration testing

- Configuration management tests
- Provisioning runtime environment
- Hardening checks

Diagram:
- Continuous integration
- Continuous delivery/deployment
- Developer’s IDE—source code
- Commit
- Build
- Test
- Pre-deploy
- IDE—source code
- Commit
- Build
- Test
- Pre-deploy
BSIMM1-3 $\rightarrow$ BSIMM9
Bsimm9: An artifact-centric view
BSIMM1-3 → BSIMM10 → Devops study
SW-Defined lifecycle governance
Modern lifecycle

- **SCM (instant)**
  - Nutrition Labels
  - SCA

- **IDE**
  - Dev (sub-second)
    - IDE, Intellisense
    - Grep
    - SCA
    - Sec Reqs.
  - Commit (seconds)
    - Commit Hook
    - Grep
    - SCA

- **Build Pipeline**
  - Build (<= 7min)
    - Stages/steps
    - SAST, SCA

- **Testing Environment**
  - Test (<= 7min)
    - Levels
    - Unit Tests, IAST

- **Orchestration (<= 45min)**
  - Secrets Management
  - Zero-trust
  - Permissions, MAC
  - Audit

- **Containers, Images**

- **Production**

- **Orchestrato**
Pipeline platform tools

• WILL DOMINATE INNOVATION FOR 3-5 YEARS

• “THEIR SIDE OF THE RIVER”
  – Familiar, popular among Devs
  – Used daily/hourly by devs
  – Sw-defined CI/CD workflows

• SECURITY AS ‘BAU’

• COMMODITIZING VULN DISCOVERY
  – SCA by default, free
  – SAST optional, free
  – Secrets management default, free

• VULNERABILITY DISCLOSURE

• COMMUNITY/CROWD COORDINATION

• AUTO-REMEDIATION
pipeline platform’s as security platform

Development’s Priorities

Security “ease of use” misses
orchestration

stages:
- build
- test
- release
- artifact_scanning
- preprod
- integration
- prod

lint:
- stage: build
- script:
  - docker pull hadolint/hadolint
  - docker run --rm -i hadolint/hadolint < Dockerfile
- allow_failure: true

sast:
- stage: build
- script:
  - id -u
  - id -g
  - docker pull secfilo/bandit
  - docker run --user $((id -u):$((id -g) -v $(pwd)):/src secfilo/bandit bandit -r /src
- allow_failure: true

test:
- stage: test
- script:
  - virtualenv env
  - source env/bin/activate
  - pip install -r requirements.txt
  - python manage.py test taskManager

cat:
- stage: test
- script:
  - pip install safety
  - safety check -r requirements.txt
- allow_failure: true

# you need to setup a server at port 10.0.1.22:8000 for this to work, a simple web server can
# zap-baseline:
- stage: integration
- script:
  - docker run -t owasp/zap2docker-stable zap-baseline.py -t http://10.0.1.22:8000/
- allow_failure: true

Security Alerts
- lodash
- actionview

GitHub
outreach, disclosure, auto-remediation
Sw-assisted monitoring of asset creation
Pillars of a modern inventory

• Discovery, import, and visualization of assets managed by the organization’s cloud and data center virtualization management consoles.

• Scraping and extracting assets and tags from infrastructure-as-code held in code repositories, as well as processing metadata from container and other artifact registries.

• Outside-in web and network scanning for publicly discoverable assets, connectivity to known organizational assets, and related ownership and administrative information.
Visualization CSP (et al) - eg cloud discovery

Solution flow:
1. Connect to CSP API
2. Consume resources, meta-data
3. Port scan assets
4. Conduct rudimentary analysis

Key elements:
• Get beyond virtualization layer
• Understand access rules
• Probe for running services, authN
• Understand (in)secure defaults, presets

QUERYING THE CSP:
• ONLY DISCOVERS CONSTITUENCY
• ACCESS/CONNECTIVITY POLICIES LIMITED BY
  • ‘STATIC’ VIEW (COMPARED TO ORCHESTRATION)
  • CAN NOT TYPICALLY SHOW CONNECTIONS
  • SERVICES RESOLUTION (W/O FAAS)

PROVIDES:
• INFRASTRUCTURE
• DATA STORES
• ‘HEALTH’-LEVEL MEASURE
Solution flow:
1. Consume file-based data (static)
2. Database labels, IDs
3. Subscribe to orchestrator events (dynamic)
4. Track service discovery

Key elements:
• Get beyond clusters, pods
• Understand labels, capabilities
• Identify low-hanging fruit:
  – Secrets
  – Policy violations

PARSING THE ORCHESTRATION:
• ONLY DISCOVERS CONSTITUENCY
  • (OR STATIC RELATIONS IN THE CASE OF NETWORKING)
• IDENTITY (FACTOR 0) IS EXTERNAL
LIMITED BY
• DESCRIBES DYNAMIC ENV, BUT ONLY STATICALLY
• DISCOVERY IS AN ESCAPE CLAUSE
• MISSES ANY HARD-CODED DEPENDENCIES

PROVIDES:
• RELATION BETWEEN CLUSTERS OF COMPONENTS
• SOMETIMES META-DATA AND CAPABILITIES
Automated verification of infrastructure
Market fragmentation is a killer

<table>
<thead>
<tr>
<th>End-point Policy</th>
<th>API Use</th>
<th>AuthN/Z</th>
<th>Data</th>
<th>Monitoring / Logging</th>
<th>Integrity &amp; Provenance</th>
<th>Secrets Mgmt</th>
<th>Availability and DR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fingerprinting</td>
<td>Security control use</td>
<td>Un-authenticated access</td>
<td>Access</td>
<td>Level</td>
<td>Containers</td>
<td>Hard-coded</td>
<td></td>
</tr>
<tr>
<td>Policy Compliance</td>
<td>Dangerous / deprecated</td>
<td>Overly-broad permissions</td>
<td>Sharing</td>
<td>Search</td>
<td>Images</td>
<td>Policy violations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>function use</td>
<td></td>
<td>Encryption</td>
<td>Notification</td>
<td>Filesystems</td>
<td>Credential Affinity</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Availability</td>
<td></td>
<td>seLinux/MAC</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summarizing

• DevOps culture and its benefits are obtained on a Dev or/and ops team basis
  – They can not mandated or instituted top down
• the new activities are hard
  – you can’t buy tools to short-circuit your maturity
  – It will take years to build these capabilities
  – Each ‘activity’ is amenable to incremental progress

• This is the new foundation: sw-defined:

  Lifecycle Management
  Security Governance
  Infrastructure Guard Rails
  Asset / Data Inventory