Effective DevSecOps - How to Integrate Security into Pipelines

Shain Singh

Cloud/5G Security Architect – APCJ Lead

ss@f5.com

A Journey not a Destination

Who am I?

Shain Singh
Cloud/5G Security Architect @F5
email: ss@f5.com

Social

https://linkedin.com/in/shsingh

shsingh@ieee.org

https://twitter.com/shainsingh

https://github.com/shsingh

https://shain.io

Professional Memberships

THE LINUX FOUNDATION

IEEE Advancing Technology for Humanity

AISA

OWASP®

CSA cloud security alliance®

usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

acm

ISACA®

Why do I think about integrating security into pipelines?

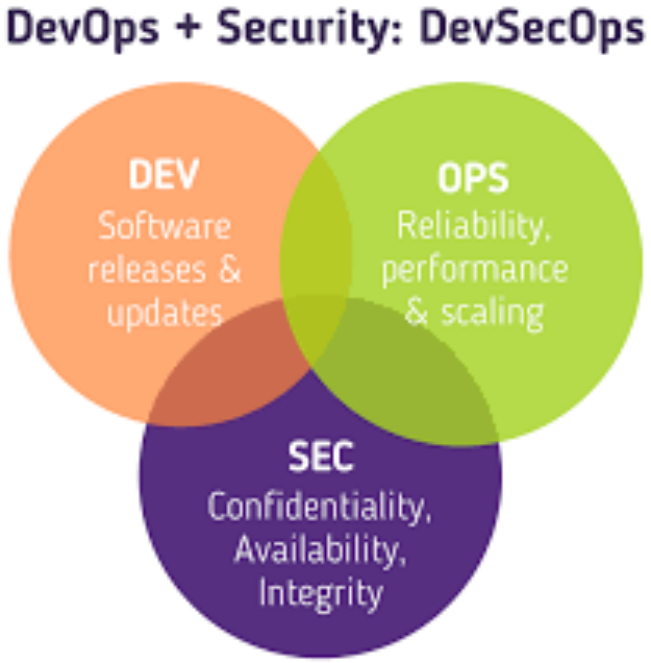
Make Security Great Again™

- Blue Teaming should be as fun as Red Teaming
- Create cultural shift in organizations by embracing ***DevOps principles***
  - Security should move from a “NO by default” to a “YES with caveats”
  - Meeting developers halfway encourages them to do the same
- Leverage toolsets and methodologies that are becoming common-place for application and infrastructure deployment

Continuous Learning™

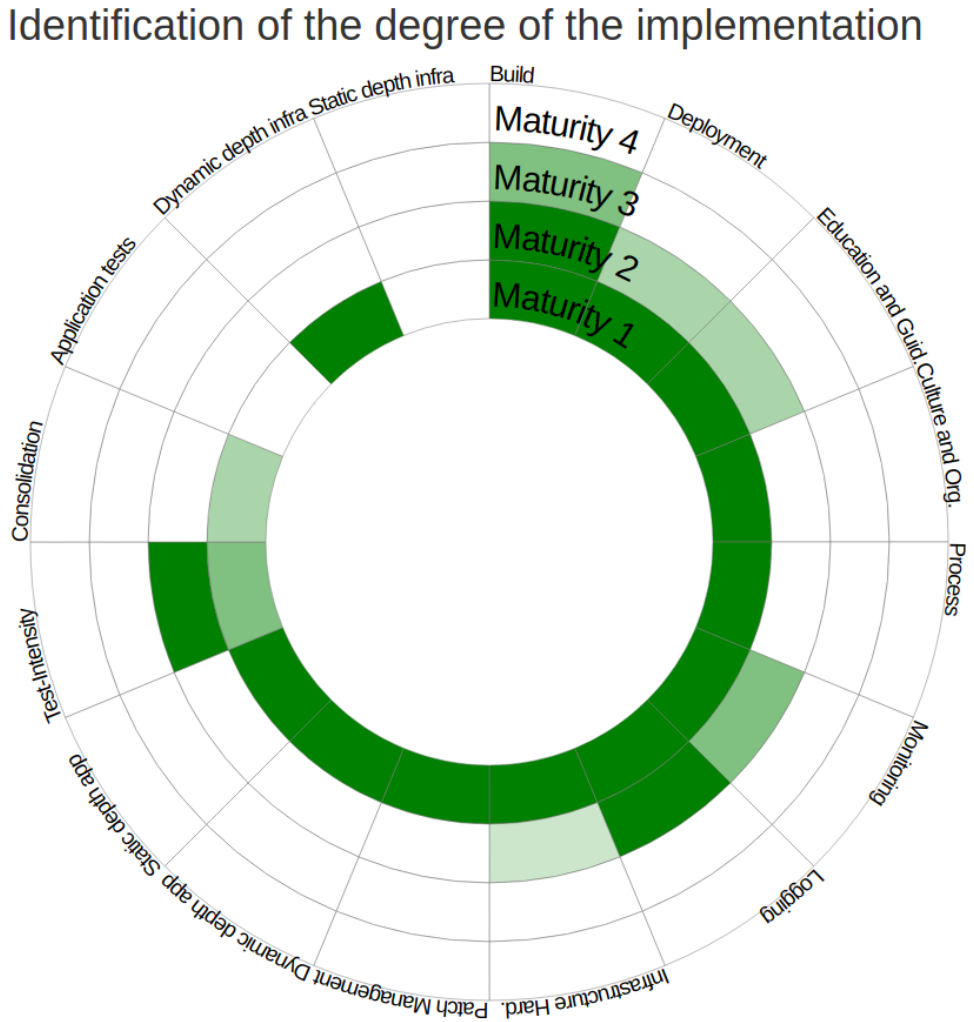
- I am a curious security practitioner - constantly learning how these new technologies can help with raising the bar by speaking to customers and also other practitioners

What exactly is DevSecOps?



Why have we not made much progress?

TOO MUCH TOO SOON CAN INCREASE FRICTION BETWEEN TEAMS AND OVERHEAD WITH SECURITY SCAN RESULTS



DevSecOps Maturity Model (DSOMM) Level 1
Basic understanding of security practices

Recommendations:
- Never fail a build pipeline – security scans will have false positives
- Investigate static and dynamic tools for the DevOps pipeline
- Build expertise with tools and analyse results
- Collaborate with development teams to resolve issues

DevSecOps Maturity Model (DSOMM) Level 2
Adoption of basic security practices

Recommendations:
- Investigate tweaking tools from their default settings for tuning
- Storing results from tools in a consolidated environment
- Starting a security champion program

DevSecOps Maturity Model (DSOMM) Level 3
High adoption of security practices

DevSecOps Maturity Model (DSOMM) Level 4
Advanced deployment of security practices at scale

https://owasp.org/www-project-devsecops-maturity-model/

Industry standards define a good set of baselines to start

Cloud Controls Matrix
Security Guidance For Critical Areas of Focus in Cloud Computing

Benefits, Risks and Recommendations For Information Security

Cybersecurity Framework

Secure Cloud Computing Architecture

CIS Benchmarks

Application Security consistency

Declarative language for describing application security

CONSISTENT POLICY ACROSS DIFFERENT DATA-PLANES AND DEPLOYMENT SCENARIOS

Same Declarative Policy

F5 Advanced WAF

Traditional applications

OWASP Top 10
SSL/TLS Inspection
Scripting
Threat Campaigns
Proactive Bot Defense
App-Layer DoS Protection

NGINX App Protect

Modern applications

OWASP Top 10
SSL/TLS Inspection
Scripting
Threat Campaigns
Proactive Bot Defense
App-Layer DoS Protection

```
{
    "policy": {
        "name": "bot_defense_policy",
        "template": {
            "name": "POLICY_TEMPLATE_NGINX_BASE"
        },
        "applicationLanguage": "utf-8",
        "enforcementMode": "blocking",
        "blocking-settings": {
            "violations": [
                {
                    "name": "VIOL_THREAT_CAMPAIGN",
                    "alarm": false,
                    "block": false
                }
            ]
        },
        "signature-sets": [
            {
                "name": "High Accuracy Signatures",
                "block": true,
                "alarm": true
            }
        ],
        "bot-defense": {
            "settings": {
                "isEnabled": true
            },
            "mitigations": {
                "classes": [
                    {
                        "name": "trusted-bot",
                        "action": "alarm"
                    },
                    {
                        "name": "untrusted-bot",
                        "action": "block"
                    },
                    {
                        "name": "malicious-bot",
                        "action": "block"
                    }
                ]
            }
        },
        "open-api-files": [
            {
                "filename": "https://domain.com/swagger-definition.json"
            }
        ]
    }
}
```

Flexibility of policy depending on deployment scenarios

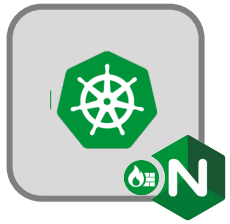
PER-APPLICATION AND PER-SCENARIO BASED SECURITY POLICY

API Gateway

Security Controls

~~WAF signatures disabled~~
Threat Campaigns enabled
Bot Mitigation enabled
Open API schema enforcement enabled
L7 DoS Mitigation enabled

https://github.com/apcj-f5/nginx-waf-templates/tree/master/examples/api%20gateway%20protection



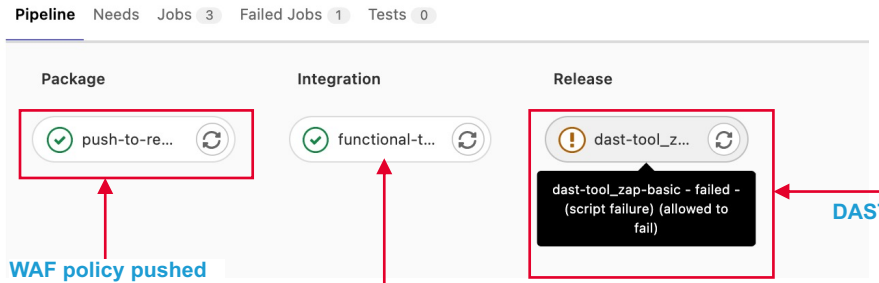
Kubernetes Ingress Controller

Security Controls

Security Controls:
~~WAF signatures disabled~~
Threat Campaigns enabled
Bot Mitigation enabled
L7 DoS Mitigation enabled

https://github.com/apcj-f5/nginx-waf-templates/tree/master/examples/kubernetes%20ingress%20controller%20protection

Example – ensuring effectiveness of WAF policy



awaf_policy-hapi-fhir.json 766 Bytes — Edit | Web IDE | Lock | Replace | Delete

```
{
   "policy": {
       "name": "policy-hapi-fhir",
       "description": "HAPI Policy for FHIR",
       "template": {
           "name": "POLICY_TEMPLATE_API_SECURITY"
       },
       "enforcementMode": "blocking",
       "server-technologies": [
           {
               "serverTechnologyName": "MySQL"
           },
           {
               "serverTechnologyName": "Unix/Linux"
           }
       ],
       "signature-settings": {
           "signatureStaging": false
       },
       "policy-builder": {
           "learnOnlyFromNonBotTraffic": false
       },
       "open-api-files": [
           {
           "link": "https://gitlab.com/shainsingh/hapi-fhir/-/raw/master/hapi-fhir_swagger.json"
           }
       ]
   }
}
```

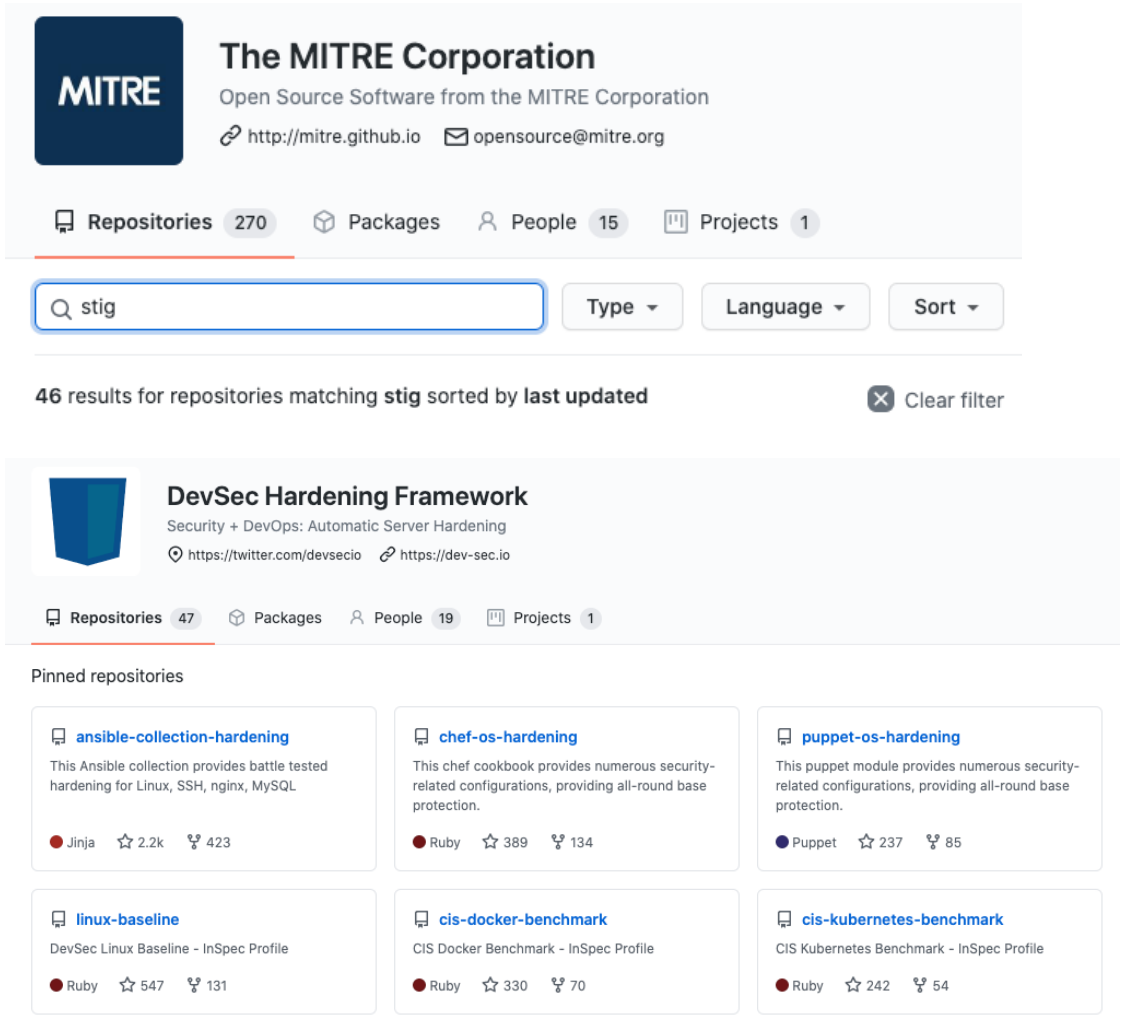
https://gitlab.com/shainsingh/hapi-fhir

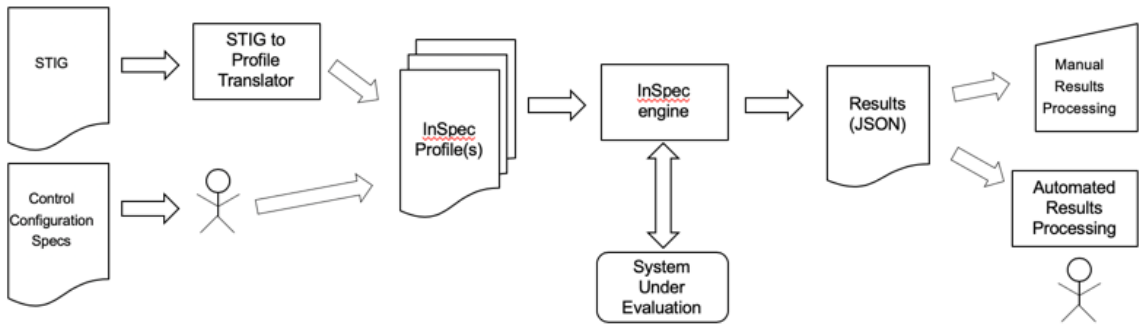
sec-release.gitlab-ci.yml 1 KB — Edit | Web IDE | Lock | Replace | Delete

```
services:
  - docker:dind

sec-dast_baseline:
  stage: sec-release
  image:
    name: owasp/zap2docker-weekly
  before_script:
    - mkdir /zap/wrk
  script:
    - zap-baseline.py -t https://hapi.f5labs.dev/fhir -I -J dast_baseline_scan-results.json
  after_script:
    - mv /zap/wrk/dast_baseline_scan-results.json /builds/shainsingh/hapi-fhir/
  rules:
    - if: $scan_periodic != "nightly"
      when: always
  artifacts:
    paths: [dast_baseline_scan-results.json]
    when: always
    expire_in: one week
  allow_failure: true

sec-dast_full:
  stage: sec-release
  image:
    name: owasp/zap2docker-weekly
  before_script:
    - mkdir /zap/wrk
  script:
    - zap-full-scan.py -t https://hapi.f5labs.dev/fhir -I -J dast_full_scan-results.json
  after_script:
    - mv /zap/wrk/dast_full_scan-results.json /builds/shainsingh/hapi-fhir/
  rules:
    - if: $scan_periodic == "nightly"
      when: always
  artifacts:
    paths: [dast_full_scan-results.json]
    when: always
    expire_in: one week
  allow_failure: true
```

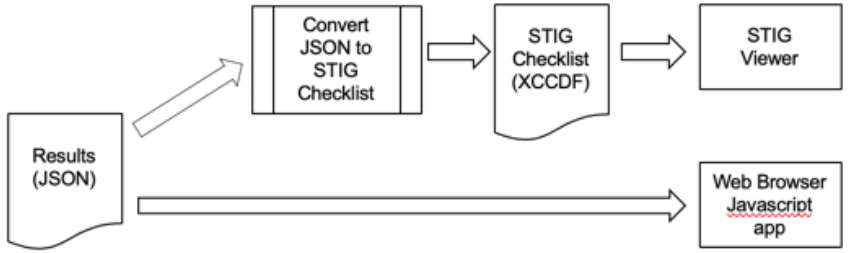
“Marking your Homework”

Compliance as Code



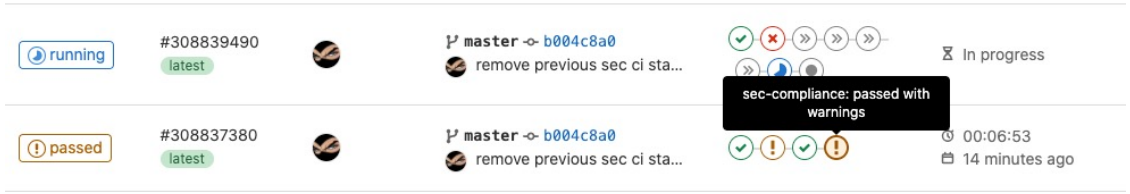


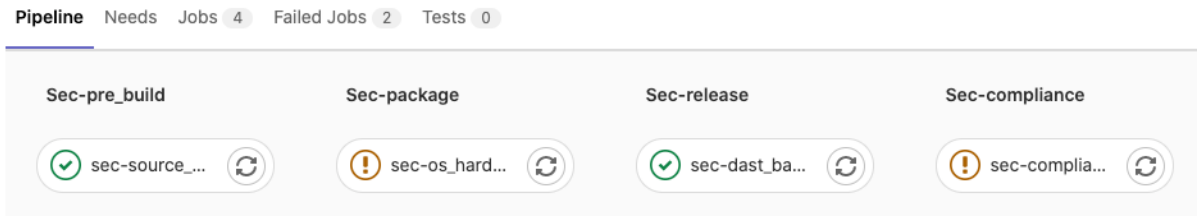
Automating Security Validation Using InSpec



Processing InSpec Results

Example – adding compliance to pipelines





sec-package.gitlab-ci.yml 760 Bytes — Edit | Web IDE

```yaml
services:
  - docker:dind

sec-os_hardening:
  stage: sec-package
  image: ansible/galaxy
  before_script:
    - mkdir -p ~/.ssh
    - echo "$DEPLOYMENT_SERVER_SSH_PRIVKEY" | tr -d '\r' > ~/.ssh/id_rsa
    - chmod 600 ~/.ssh/id_rsa
    - eval "$(ssh-agent -s)"
    - ssh-add ~/.ssh/id_rsa
    - echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config
  script:
    - echo "[prod]" >> inventory.ini
    - echo "$DEPLOYMENT_SERVER" >> inventory.ini
    - export ANSIBLE_STDOUT_CALLBACK=json
    - ansible-galaxy install dev-sec.os-hardening
    - ansible-playbook -i inventory.ini ansible-hardening.yml > sec-os_hardening-results.json
  artifacts:
    paths: [sec-os_hardening-results.json]
    when: always
    expire_in: one week
  allow_failure: true
```

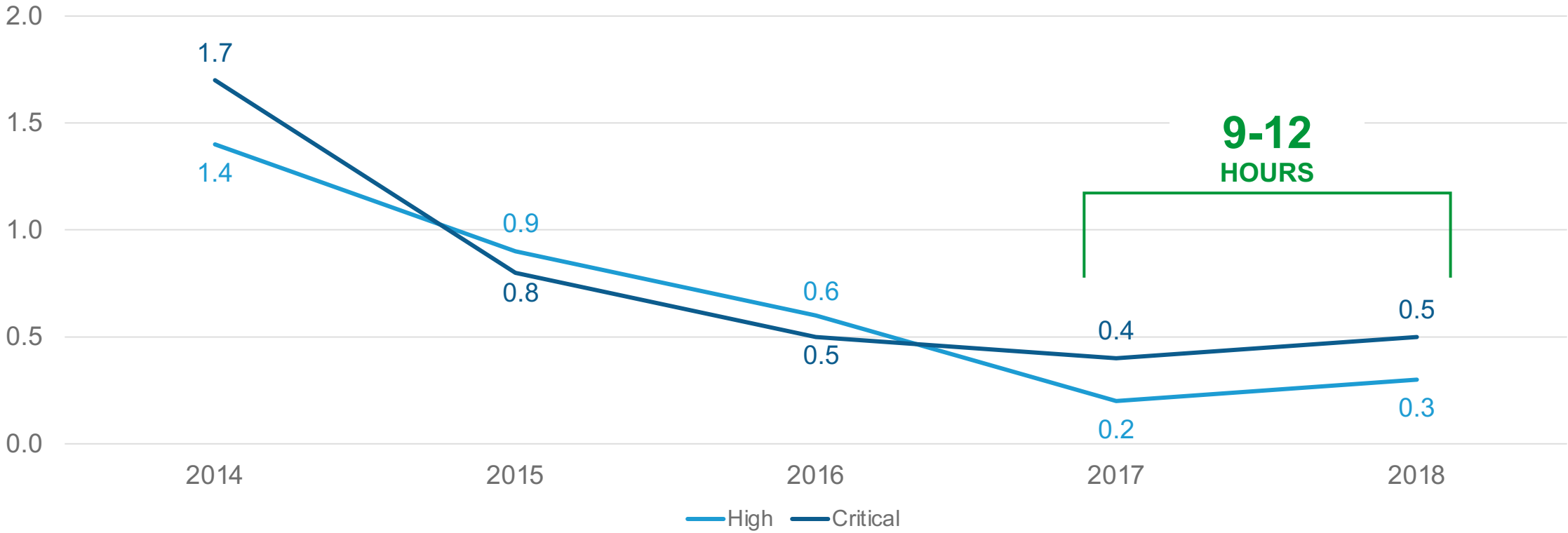
sec-compliance.gitlab-ci.yml 694 Bytes — Edit | Web IDE | Lock | Replace | Delete

```yaml
services:
  - docker:dind

sec-compliance:
  stage: sec-compliance
  image:
    name: chef/inspec
  only:
    - "master"
  environment: production
  before_script:
    - mkdir -p ~/.ssh
    - echo "$DEPLOYMENT_SERVER_SSH_PRIVKEY" | tr -d '\r' > ~/.ssh/id_rsa
    - chmod 600 ~/.ssh/id_rsa
    - eval "$(ssh-agent -s)"
    - ssh-add ~/.ssh/id_rsa
    - echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config
  script:
    - inspec exec https://github.com/dev-sec/linux-baseline -t ssh://root@$DEPLOYMENT_SERVER -i /id_rsa --chef-license accept --reporter json:/opt/sec-
  artifacts:
    paths: [sec-compliance-results.json]
    when: always
  allow_failure: true
```

More apps mean more security

Protecting against Abuse of Functionality

Average days between “HIGH” AND “CRITICAL” CVEs released



Protecting against Abuse of Intent

The Automated Threat Handbook

Web Applications

The Automated Threat Handbook provides actionable information and resources to help defend against automated threats to web applications.

OAT-020 Account Aggregation
OAT-019 Account Creation
OAT-003 Ad Fraud
OAT-009 CAPTCHA Defeat
OAT-010 Card Cracking
OAT-001 Carding
OAT-012 Cashing Out
OAT-007 Credential Cracking
OAT-008 Credential Stuffing
OAT-021 Denial of Inventory
OAT-015 Denial of Service
OAT-006 Expediting
OAT-004 Fingerprinting
OAT-018 Footprinting
OAT-005 Scalping
OAT-011 Scraping
OAT-016 Skewing
OAT-013 Sniping
OAT-017 Spamming
OAT-002 Token Cracking
OAT-014 Vulnerability Scanning

Summarising it all

Key Takeaways

- Remember the “People” and “Process” portions of DevOps
- Start small, then increment - DSOMM Level 1
- Apply declarative WAF policies for use in pipelines
- Intent is to have security across all apps, everywhere
- A *manual WAF policy in transparent mode* may be less effective than a *declarative policy in blocking mode*