



# Effective DevSecOps - How to Integrate Security into Pipelines

**Shain Singh**

Cloud/5G Security Architect – APCJ Lead

[ss@f5.com](mailto:ss@f5.com)

# A Journey not a Destination

# Who am I?



Shain Singh

Cloud/5G Security Architect @F5

email: [ss@f5.com](mailto:ss@f5.com)

## Social

 <https://linkedin.com/in/shsingh>

 [shsingh@ieee.org](mailto:shsingh@ieee.org)

 <https://twitter.com/shainsingh>

 <https://github.com/shsingh>

 <https://shain.io>

## Professional Memberships



# 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?



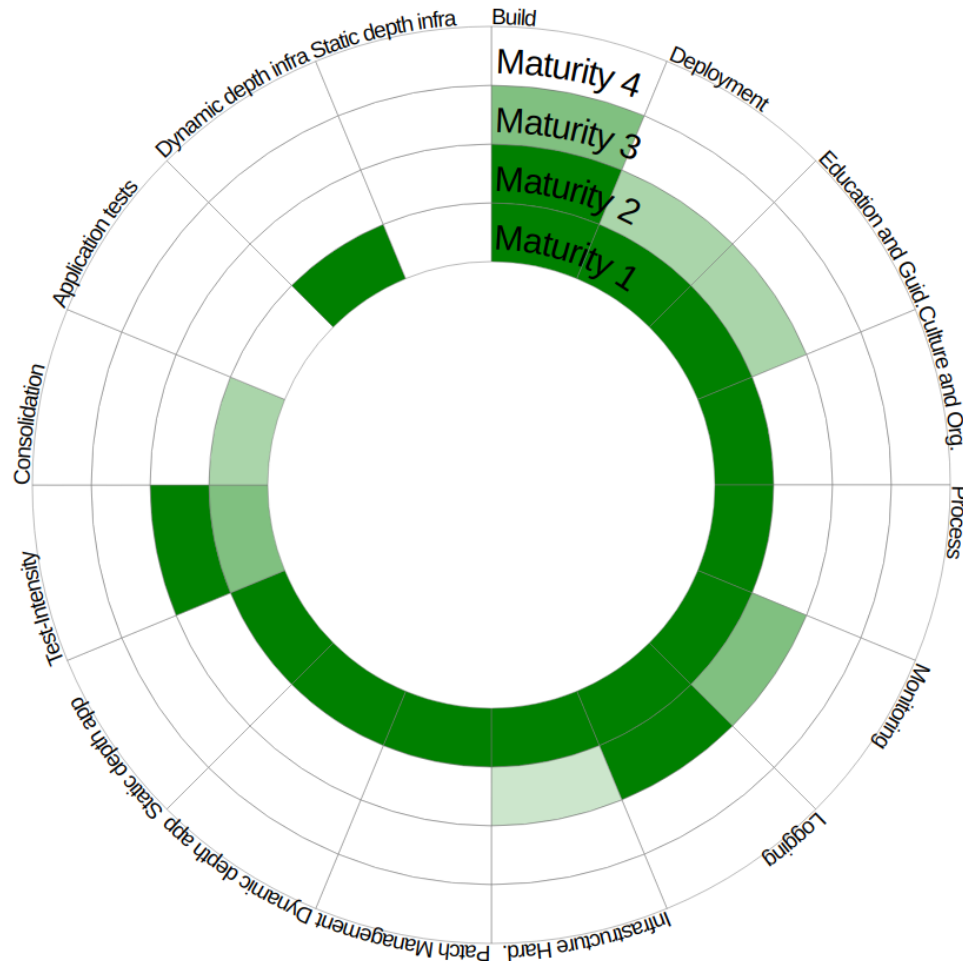
## DevOps + Security: DevSecOps



# Why have we not made much progress?

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

Identification of the degree of the implementation



## 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*

# 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



**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*

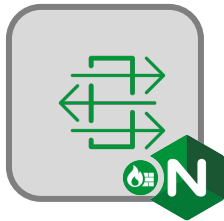
- OWASP Top 10
- SSL/TLS Inspection
- Scripting
- Threat Campaigns
- Proactive Bot Defense
- App-Layer DoS Protection

Same Declarative Policy

```
{
  "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>

```
{
  "policy": {
    "name": "api_gateway_policy",
    "template": {
      "name": "POLICY_TEMPLATE_NGINX_BASE"
    },
    "applicationLanguage": "utf-8",
    "enforcementMode": "blocking",
    "blocking-settings": {
      "violations": [
        {
          "name": "VIOL_THREAT_CAMPAIGN",
          "alarm": true,
          "block": true
        }
      ]
    },
    "signature-sets": [
      {
        "name": "All-Signatures",
        "block": false,
        "alarm": false
      }
    ],
    "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"
      }
    ]
  }
}
```

```
{
  "policy": {
    "name": "ingress_controller_policy",
    "template": {
      "name": "POLICY_TEMPLATE_NGINX_BASE"
    },
    "applicationLanguage": "utf-8",
    "enforcementMode": "blocking",
    "blocking-settings": {
      "violations": [
        {
          "name": "VIOL_THREAT_CAMPAIGN",
          "alarm": true,
          "block": true
        }
      ]
    },
    "signature-sets": [
      {
        "name": "All-Signatures",
        "block": false,
        "alarm": false
      }
    ],
    "bot-defense": {
      "settings": {
        "isEnabled": true
      },
      "mitigations": {
        "classes": [
          {
            "name": "trusted-bot",
            "action": "alarm"
          },
          {
            "name": "untrusted-bot",
            "action": "block"
          },
          {
            "name": "malicious-bot",
            "action": "block"
          }
        ]
      }
    }
  }
}
```



**Kubernetes Ingress Controller**  
Security Controls

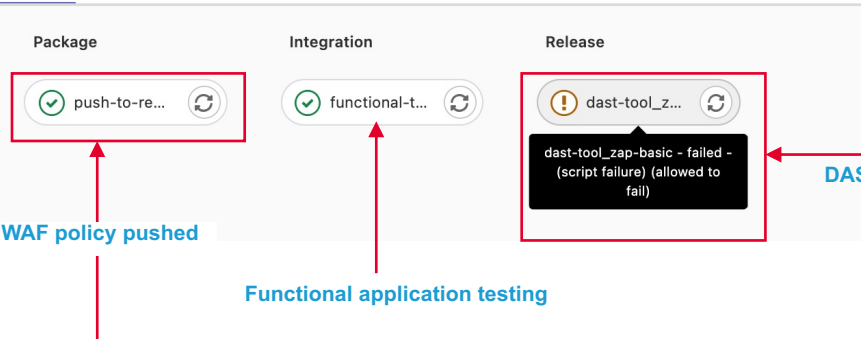
- WAF signatures disabled
- Threat Campaigns enabled
- Bot Mitigation enabled
- L7 DoS Mitigation enabled

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

```
{
  "mitigation_mode": "standard",
  "signatures": "on",
  "bad_actors": "on",
  "use_automation_tools_detection": "on",
  "tls_fingerprint": "on"
}
```

# Example – ensuring effectiveness of WAF policy

Pipeline Needs Jobs 3 Failed Jobs 1 Tests 0



```
sec-release.gitlab-ci.yml 1 KB
1 services:
2   - docker:dind
3
4 sec-dast_baseline:
5   stage: sec-release
6   image:
7     name: owasp/zap2docker-weekly
8   before_script:
9     - mkdir /zap/wrk
10  script:
11    - zap-baseline.py -t https://hapi.f5labs.dev/fhir -I -J dast_baseline_scan-results.json
12  after_script:
13    - mv /zap/wrk/dast_baseline_scan-results.json /builds/shainsingh/hapi-fhir/
14  rules:
15    - if: $scan_periodic != "nightly"
16      when: always
17  artifacts:
18    paths: [dast_baseline_scan-results.json]
19    when: always
20    expire_in: one week
21    allow_failure: true
22
23 sec-dast_full:
24   stage: sec-release
25   image:
26     name: owasp/zap2docker-weekly
27   before_script:
28     - mkdir /zap/wrk
29   script:
30     - zap-full-scan.py -t https://hapi.f5labs.dev/fhir -I -J dast_full_scan-results.json
31  after_script:
32    - mv /zap/wrk/dast_full_scan-results.json /builds/shainsingh/hapi-fhir/
33  rules:
34    - if: $scan_periodic == "nightly"
35      when: always
36  artifacts:
37    paths: [dast_full_scan-results.json]
38    when: always
39    expire_in: one week
40    allow_failure: true
```

```
awaf_policy-hapi-fhir.json 766 Bytes
1 {
2   "policy": {
3     "name": "policy-hapi-fhir",
4     "description": "HAPI Policy for FHIR",
5     "template": {
6       "name": "POLICY_TEMPLATE_API_SECURITY"
7     },
8     "enforcementMode": "blocking",
9     "server-technologies": [
10      {
11        "serverTechnologyName": "MySQL"
12      },
13      {
14        "serverTechnologyName": "Unix/Linux"
15      }
16    ],
17     "signature-settings": {
18       "signatureStaging": false
19     },
20     "policy-builder": {
21       "learnOnlyFromNonBotTraffic": false
22     },
23     "open-api-files": [
24       {
25         "link": "https://gitlab.com/shainsingh/hapi-fhir/-/raw/master/hapi-fhir_swagger.json"
26       }
27     ]
28   }
29 }
```

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



# “Marking your Homework”

# Compliance as Code

**MITRE** The MITRE Corporation  
Open Source Software from the MITRE Corporation  
<http://mitre.github.io> [opensource@mitre.org](mailto:opensource@mitre.org)

Repositories 270 Packages People 15 Projects 1

Q stig Type Language Sort

46 results for repositories matching stig sorted by last updated

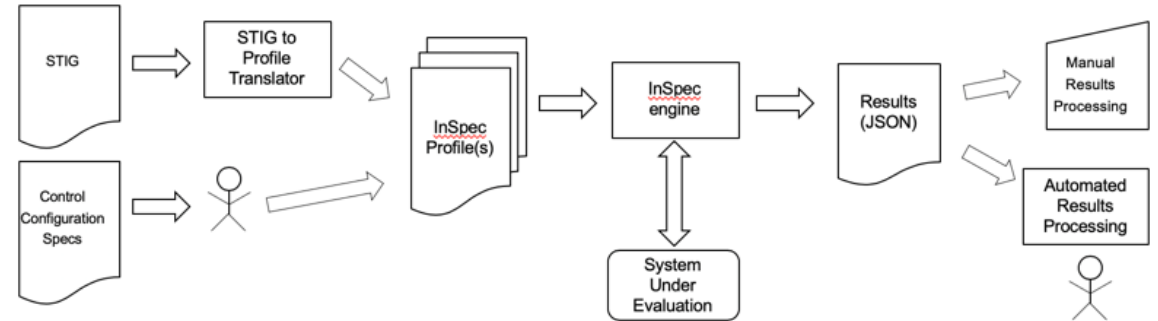
Clear filter

**DevSec Hardening Framework**  
Security + DevOps: Automatic Server Hardening  
<https://twitter.com/devsecio> <https://dev-sec.io>

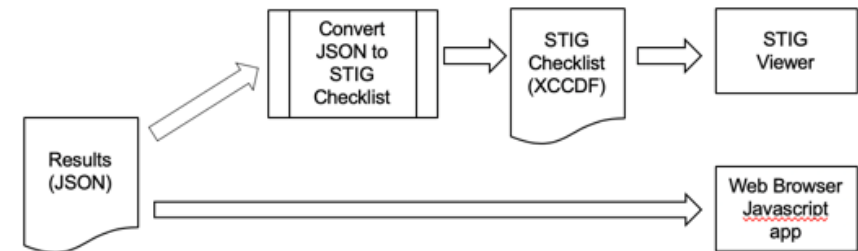
Repositories 47 Packages People 19 Projects 1

## Pinned repositories

<b>ansible-collection-hardening</b> This Ansible collection provides battle tested hardening for Linux, SSH, nginx, MySQL ● Jinja ☆ 2.2k 🍴 423	<b>chef-os-hardening</b> This chef cookbook provides numerous security-related configurations, providing all-round base protection. ● Ruby ☆ 389 🍴 134	<b>puppet-os-hardening</b> This puppet module provides numerous security-related configurations, providing all-round base protection. ● Puppet ☆ 237 🍴 85
<b>linux-baseline</b> DevSec Linux Baseline - InSpec Profile ● Ruby ☆ 547 🍴 131	<b>cis-docker-benchmark</b> CIS Docker Benchmark - InSpec Profile ● Ruby ☆ 330 🍴 70	<b>cis-kubernetes-benchmark</b> CIS Kubernetes Benchmark - InSpec Profile ● Ruby ☆ 242 🍴 54



## Automating Security Validation Using InSpec



## Processing InSpec Results

# Example – adding compliance to pipelines

The screenshot shows two pipeline jobs. The first job, ID #308839490, is in a 'running' state. The second job, ID #308837380, is in a 'passed' state. A tooltip for the second job indicates 'sec-compliance: passed with warnings'. The jobs are performed on a 'master' branch with commit hash 'b004c8a0'.

The screenshot shows a pipeline overview with four stages: 'Sec-pre\_build', 'Sec-package', 'Sec-release', and 'Sec-compliance'. Each stage has a job icon: a green checkmark for 'Sec-pre\_build' and 'Sec-release', and a yellow warning icon for 'Sec-package' and 'Sec-compliance'. The pipeline summary shows 4 jobs, 2 failed jobs, and 0 tests.

```
sec-package.gitlab-ci.yml 760 Bytes Edit Web IDE
1 services:
2   - docker:dind
3
4 sec-os_hardening:
5   stage: sec-package
6   image: ansible/galaxy
7   before_script:
8     - mkdir -p ~/.ssh
9     - echo "$DEPLOYMENT_SERVER_SSH_PRIVKEY" | tr -d '\r' > ~/.ssh/id_rsa
10    - chmod 600 ~/.ssh/id_rsa
11    - eval "$(ssh-agent -s)"
12    - ssh-add ~/.ssh/id_rsa
13    - echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config
14   script:
15     - echo "[prod]" >> inventory.ini
16     - echo "$DEPLOYMENT_SERVER" >> inventory.ini
17     - export ANSIBLE_STDOUT_CALLBACK=json
18     - ansible-galaxy install dev-sec.os-hardening
19     - ansible-playbook -i inventory.ini ansible-hardening.yml > sec-os_hardening-results.json
20   artifacts:
21     paths: [sec-os_hardening-results.json]
22     when: always
23     expire_in: one week
24     allow_failure: true
```

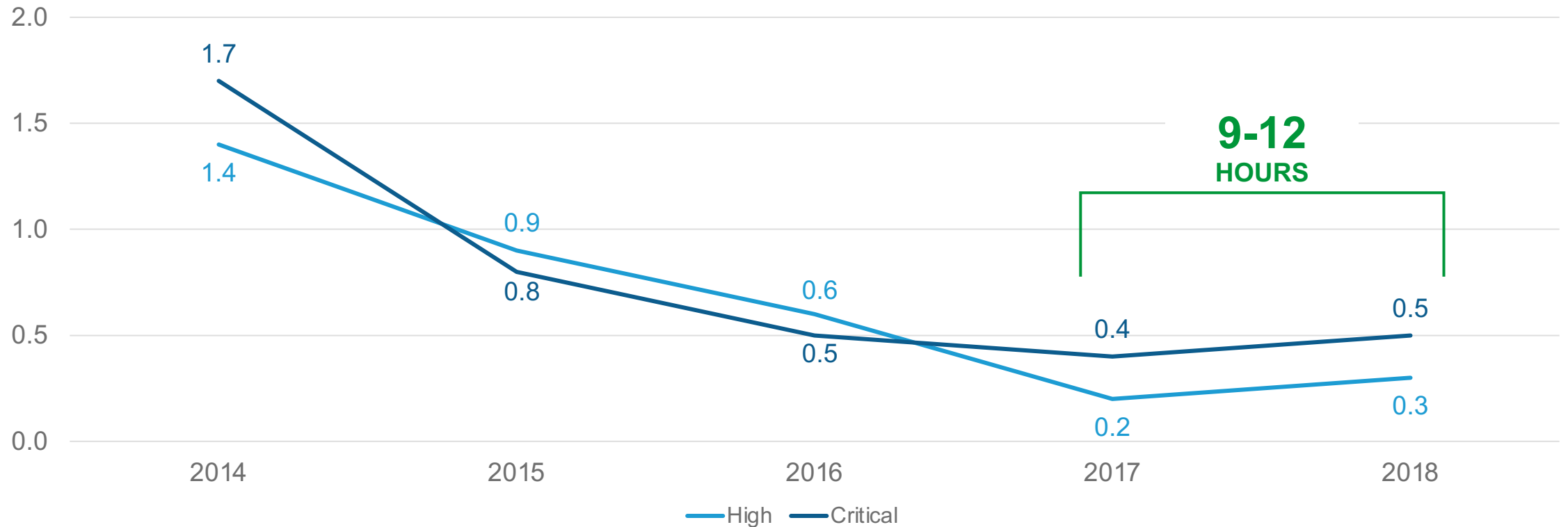
```
sec-compliance.gitlab-ci.yml 694 Bytes Edit Web IDE Lock Replace Delete
1 services:
2   - docker:dind
3
4 sec-compliance:
5   stage: sec-compliance
6   image:
7     name: chef/inspec
8   only:
9     - "master"
10  environment: production
11  before_script:
12    - mkdir -p ~/.ssh
13    - echo "$DEPLOYMENT_SERVER_SSH_PRIVKEY" | tr -d '\r' > ~/.ssh/id_rsa
14    - chmod 600 ~/.ssh/id_rsa
15    - eval "$(ssh-agent -s)"
16    - ssh-add ~/.ssh/id_rsa
17    - echo -e "Host *\n\tStrictHostKeyChecking no\n\n" > ~/.ssh/config
18  script:
19    - inspec exec https://github.com/dev-sec/linux-baseline -t ssh://root@$DEPLOYMENT_SERVER -i /id_rsa --chef-license accept --reporter json:/opt/sec-
20  artifacts:
21    paths: [sec-compliance-results.json]
22    when: always
23    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*