Practical Approach For "Lightway" Threat Modeling Automation

Vitaly Davidoff

CISSP, CSSLP



DEEPSEC
IN-DEPTH SECURITY

Agenda

DEEPSEC

- What is Threat Modeling
- Existing Methodologies
- Problems in common solution
- Lightway Threat Modeling “As a Code”
- Risks Based Security Tests Orchestration
- CI/CD Overview
- Tools
- Things to warry about



$WhoAmI

DEEPSEC

- AppSec Domain Lead at Citi Innovation Lab
- 15 years of experience as a developer
- 5+ years experience in application security
- Martial Arts instructor



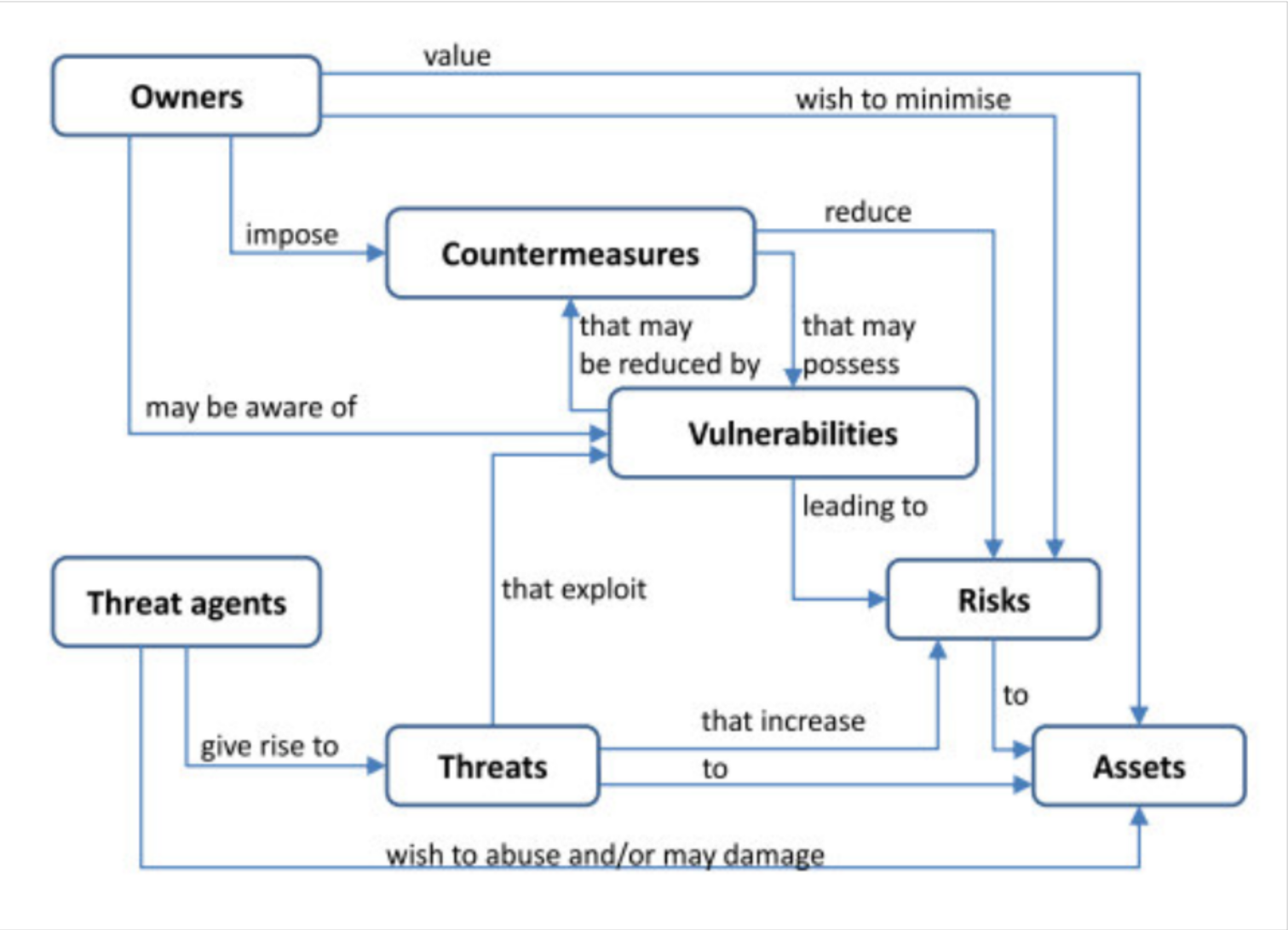
Basic Security Terminology

DEEPSEC

- Asset
- Threat
- Vulnerability
- Attack (or Exploit)
- Abuse Case
- Countermeasure (Security Control)
- Risk

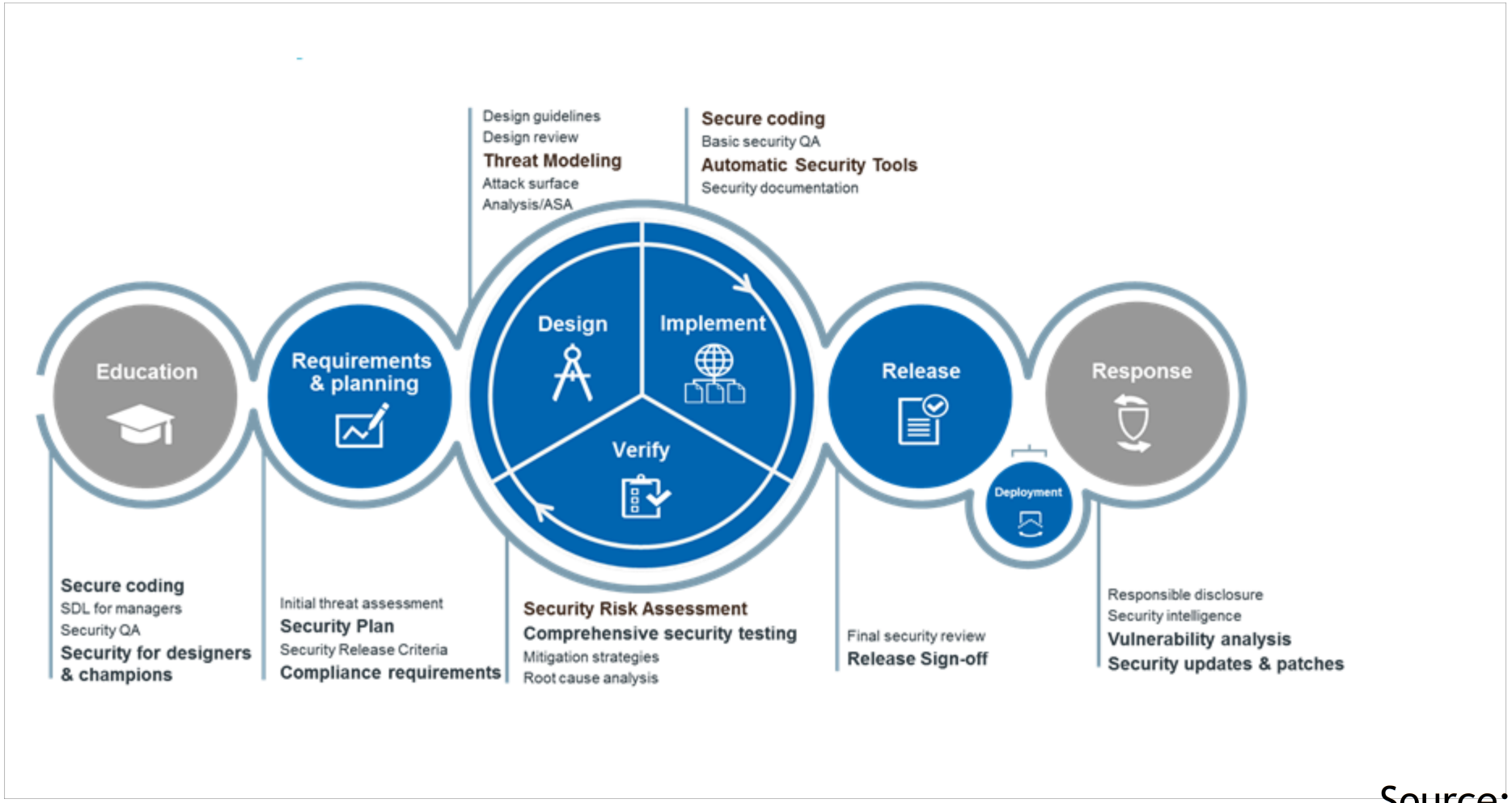


Basic Security Terminology



Source: ISO 15408:2005

Secure SDLC Process



Source: MicroFocus

What Is Threat Modeling

DEEPSEC

- A powerful way to identify potentials threats, visualize risks and understand the security of the application
- A starting point to create robust security minded test plans
- The most reliable way to:
  - Understand the security implications of system architecture
  - Find business-process and system level security risks
  - Ensure you get the most impact for your security investment



Existing Methodologies

DEEPSEC

- **STRIDE**

  The STRIDE approach to threat modeling was introduced in 1999 at Microsoft, providing a mnemonic for developers to find 'threats to our products'.

- **PASTA**

  The Process for Attack Simulation and Threat Analysis (PASTA) is a seven-step, risk-centric methodology.

- **VAST**

  VAST is an acronym for Visual, Agile, and Simple Threat modeling. The underlying principle of this methodology is the necessity of scaling the threat modeling process across the infrastructure and entire SDLC, and integrating it seamlessly into an Agile software development methodology.

- **Trike**

  The focus of the Trike methodology is using threat models as a risk-management tool. Within this framework, threat models are used to satisfy the security auditing process.



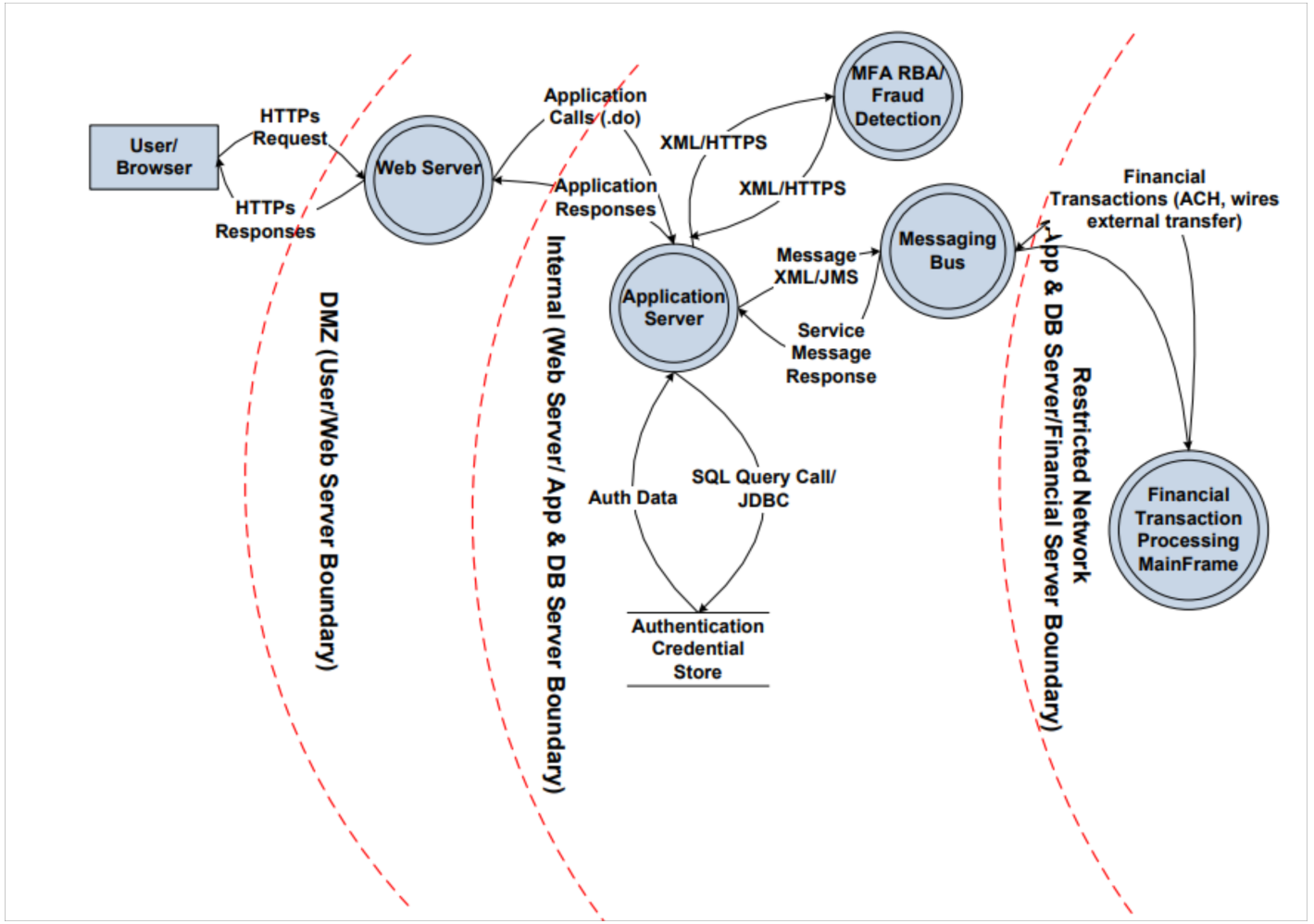
Threat Modeling Process

DEEPSEC

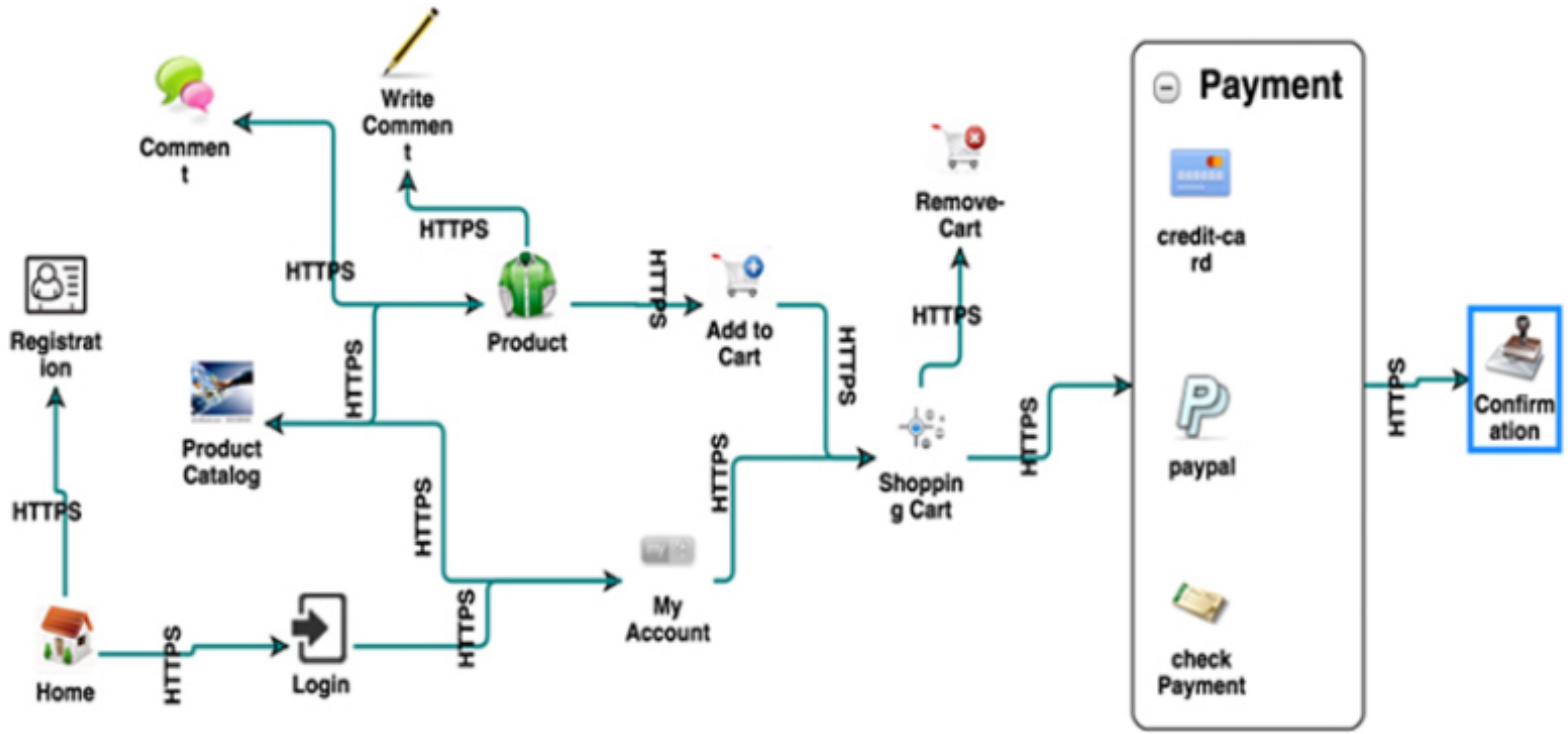
- What are we building?
  - "what are we working on, now, in this sprint/spike/feature?"
- What can go wrong?
- What are we going to do about that?
- Did we do a good enough job?



Data Flow Diagram (DFD)

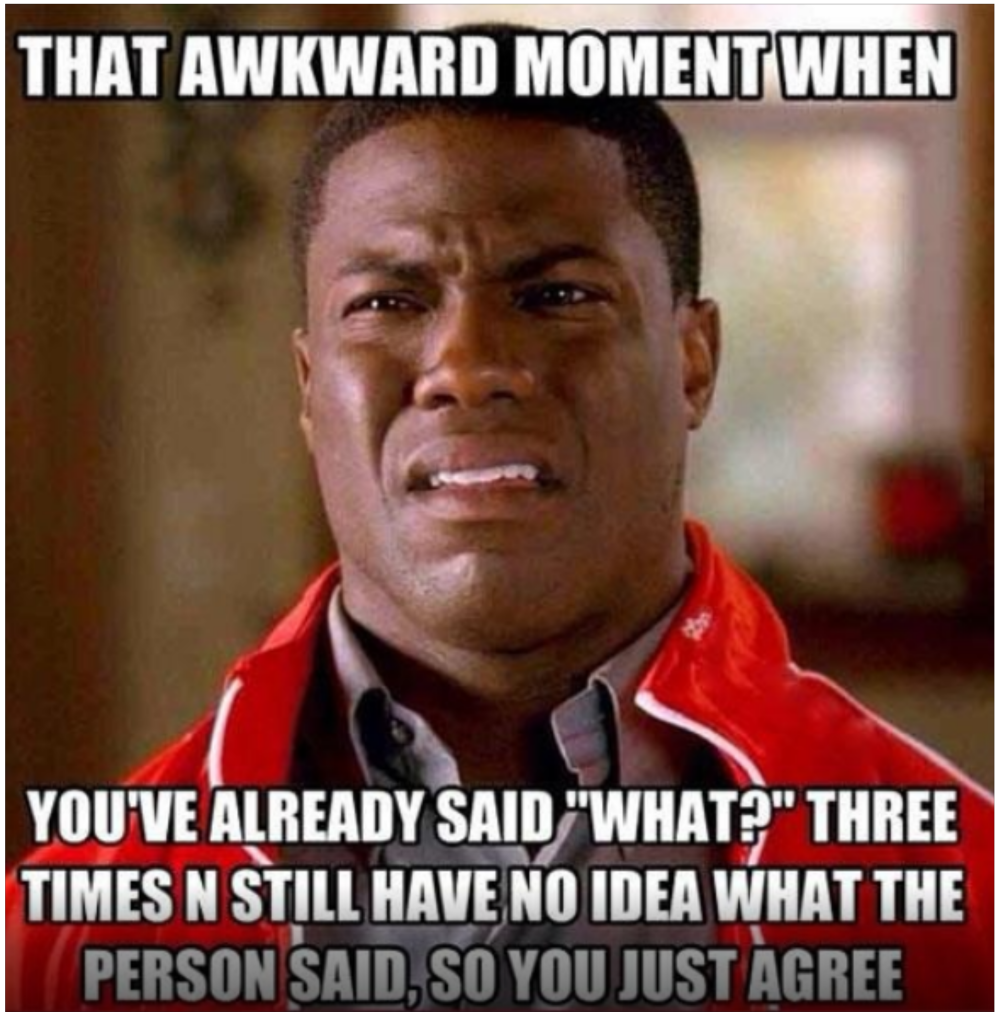


Process Flow Diagram (PFD)



“Think Like A Hacker!”

DEEPSEC

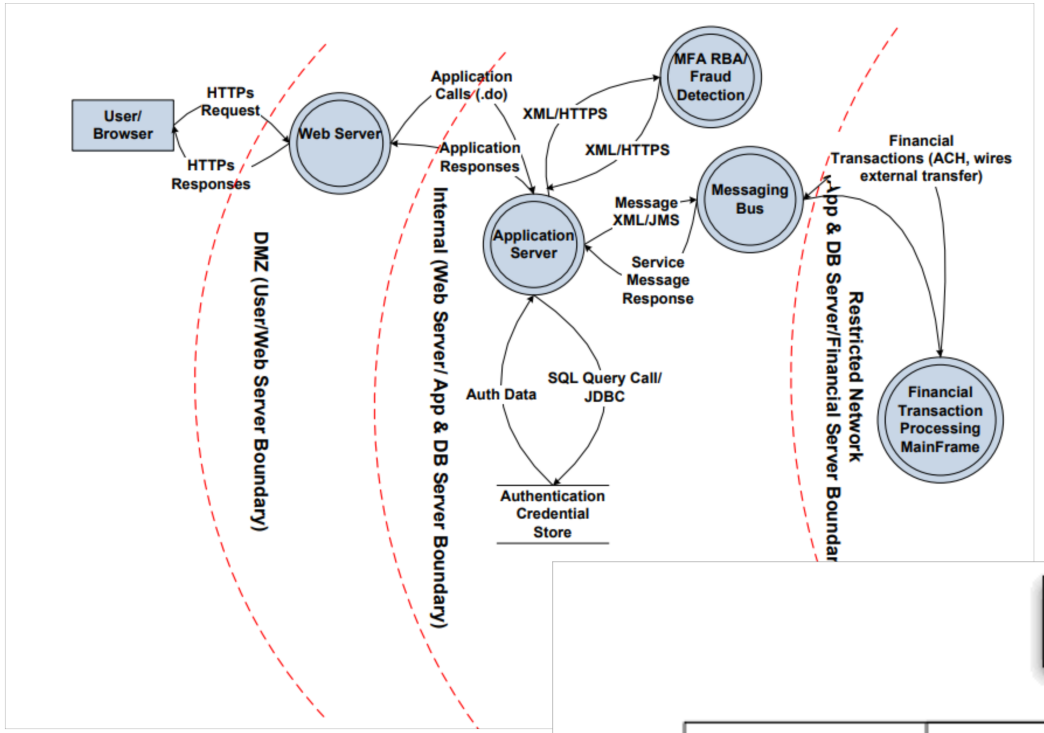


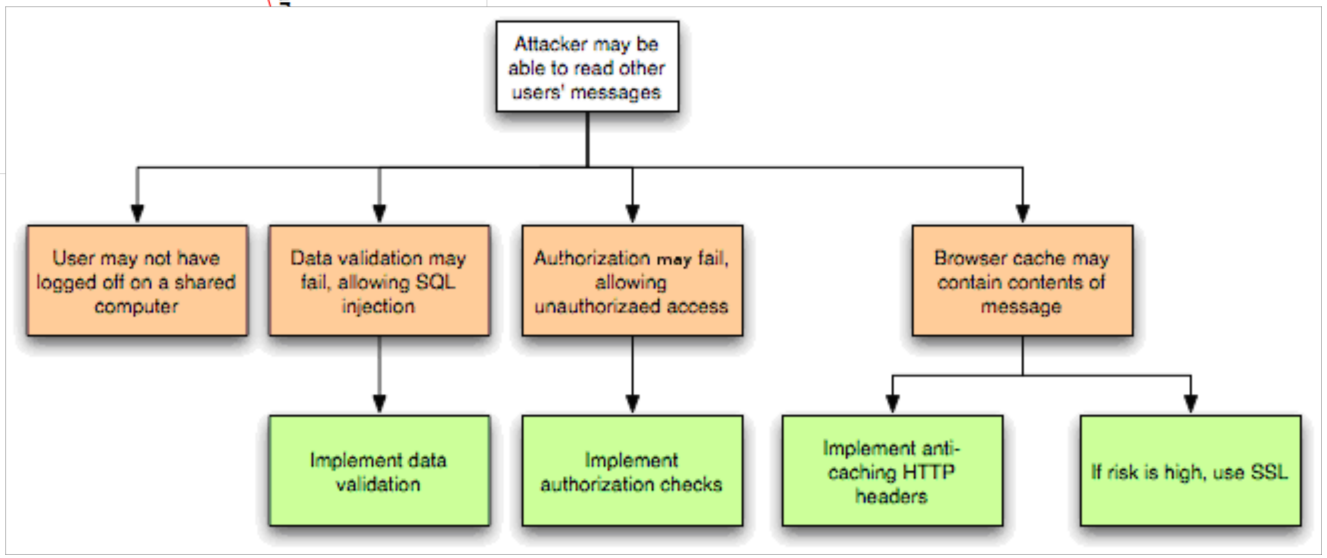
STRIDE – Identify Threats

DEEPSEC

Threat	Property we want
Spoofing	Authentication
Tampering	Integrity
Repudiation	Nonrepudiation
Information Disclosure	Confidentiality
Denial of Service	Availability
Elevation of Privilege	Authorization



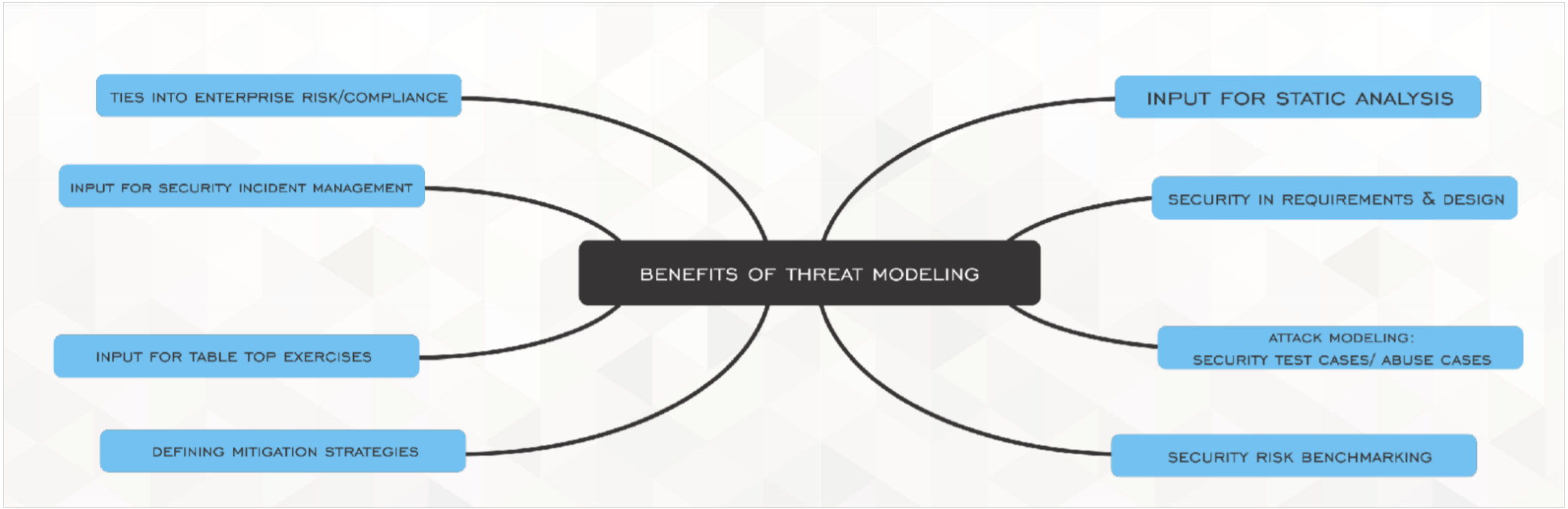




Source: OWASP

Threat Modeling - Benefits

DEEPSEC



Source: We45

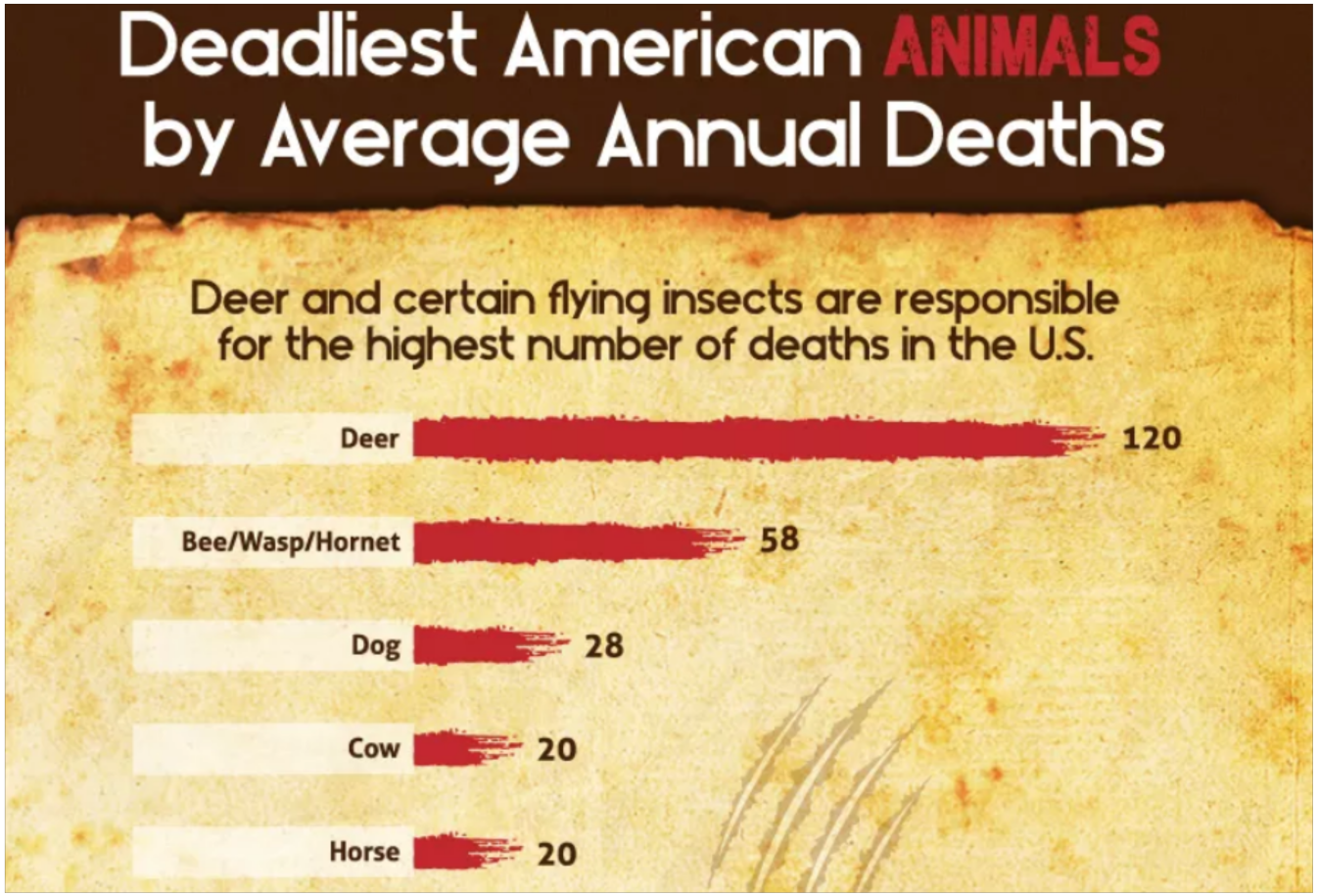
A Note About “Intuitive” Security

DEEPSEC



A Note About “Intuitive” Security

DEEPSEC

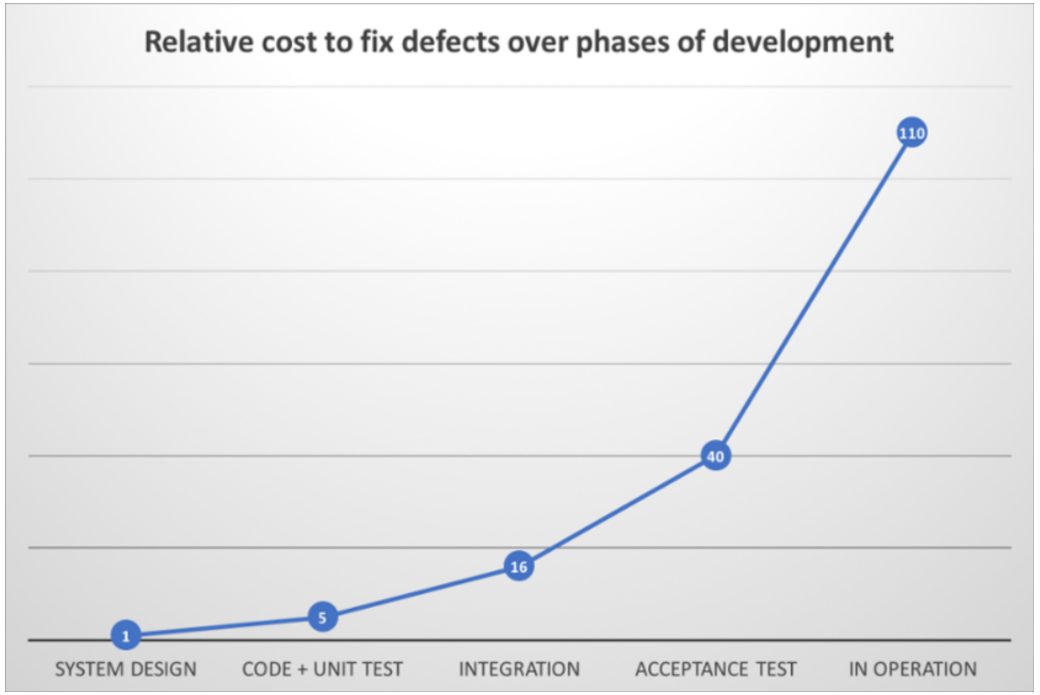


Threat Modeling And Secure SDLC

DEEPSEC

In simple words, at the early stages of the SDLC:

- Every time there is a change in the system's architecture.
- After a security incident has occurred or new vulnerabilities are introduced.
- As soon as the architecture is ready.



Threat Modeling - Responsibilities

DEEPSEC

- Architects
- **Security Specialists**
- Business Analysts
- Developers
- **Security Champions**



Common Problems

DEEPSEC

- Manual process, takes a lot of time
- Not propagated to Developers (in some cases, design defects opened)
- Updated on rare occasions (or not updates at all)
- Proceeded by security team (with very little help from R&D)
- Not integrated with DevOps model (CI/CD)
- Concentrated on Diagrams, not on countermeasures

"Agile and Microservices created a reality where Threat Modeling becomes a bottleneck - heavily resource intensive, requires a full team of expensive security professionals, takes up far too much time, and does not scalable…"

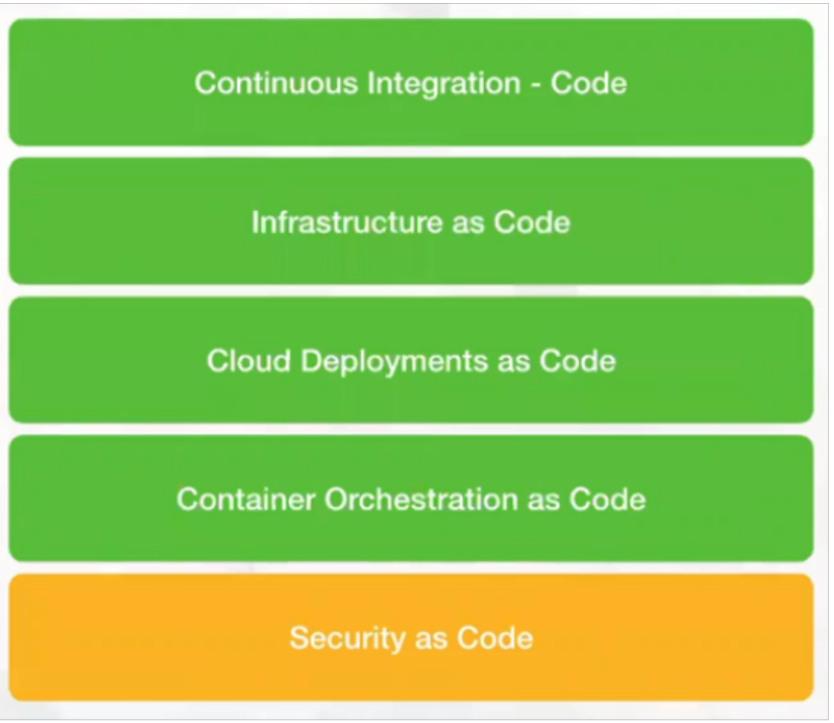


DevOps Automation

DEEPSEC

- Fosters speed
- Minimize human intervention
- Specification based frameworks
- Abstract the complexity away from the developer

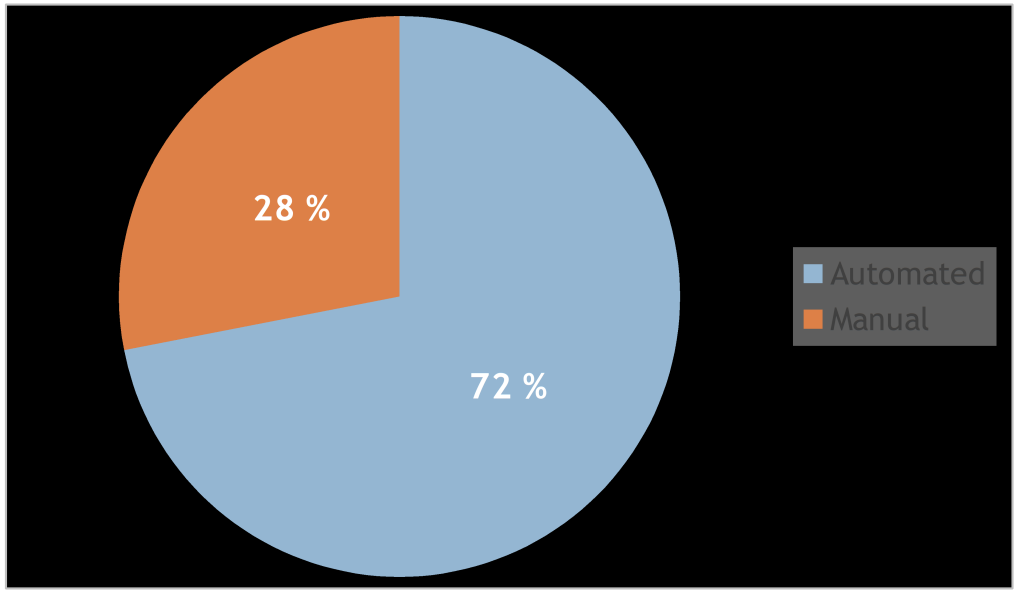
- ✓ **Make everything as a Code!**



“Lightway” Threat Modeling As A Code

DEEPSEC

- CI/CD integration (security tests might be Threat Modeling based)
- Collaboration between different roles
- Iterative Threat Modeling
- Manageable Threat Modeling
- **Just enough Threat Modeling**



✓ **Let ~80% of Threat Modeling be automated!**

Step 1: Pattern* Definition (Example)

DEEPSEC

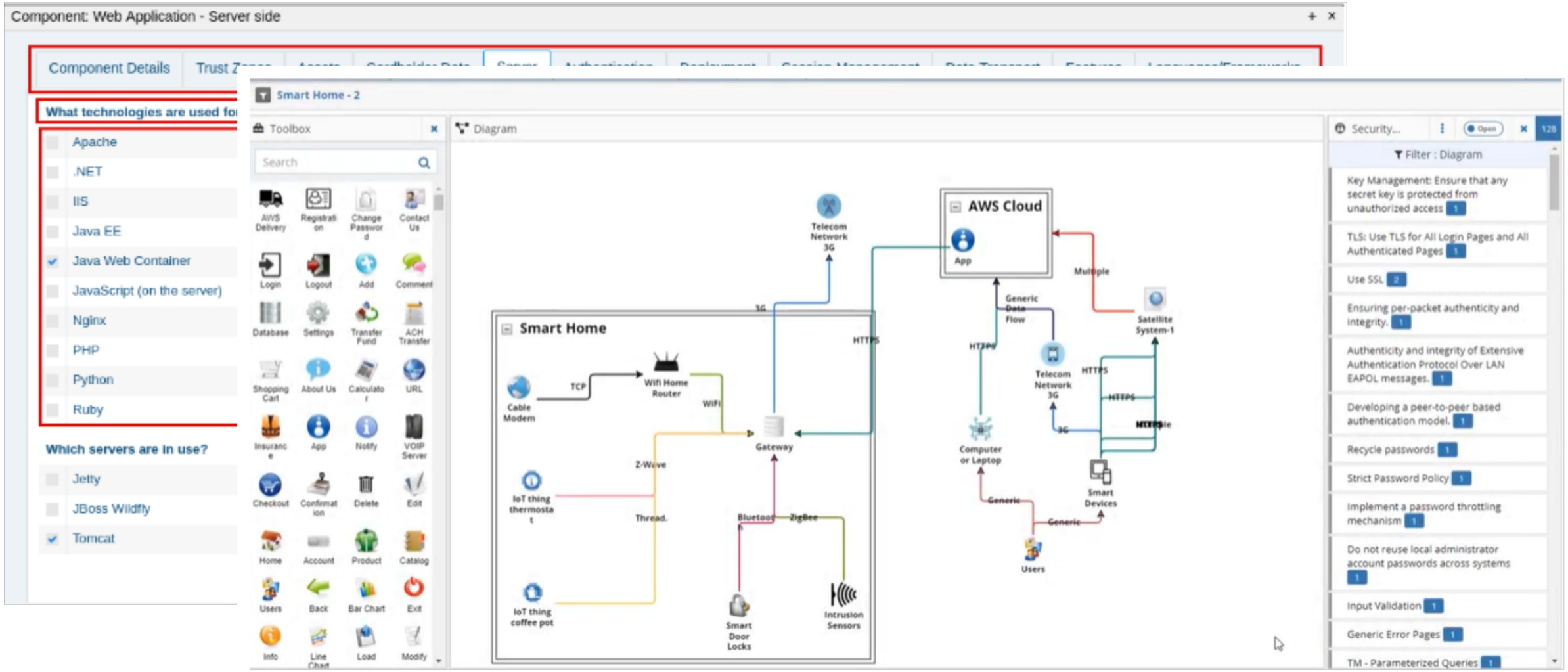
```
{
   "pattern": {
      "class": "HTTP\\WEB\\Service",
      "id": "tmpg134",
      "name":  "User/Pass Authentication against a Service",
      "parent id":  "tmpg112",
      "threats": {
         "TG1": {
            "tests": "SAST", "SCA", "DAST"
            "description": "Dictionary attack against username using common password"
            "contramesures": {
               "groups": [
                  "Implement password quality checks","Rate limit connections from the same IP address", "Require the use of 2FA"
               ]
            }
         },
         "TG2": {
            "tests": "DAST"
            "description": "Legitimate users cannot access service because of DoS"
            "contramesures": {
               "groups": [
                  "Enable up-stream DoS prevention"
               ]
            }
         }
         ...
      }
   }
}
```

* Stephen De Vries - Threat Modeling With Architectural Risk Patterns - AppSecUSA 2016

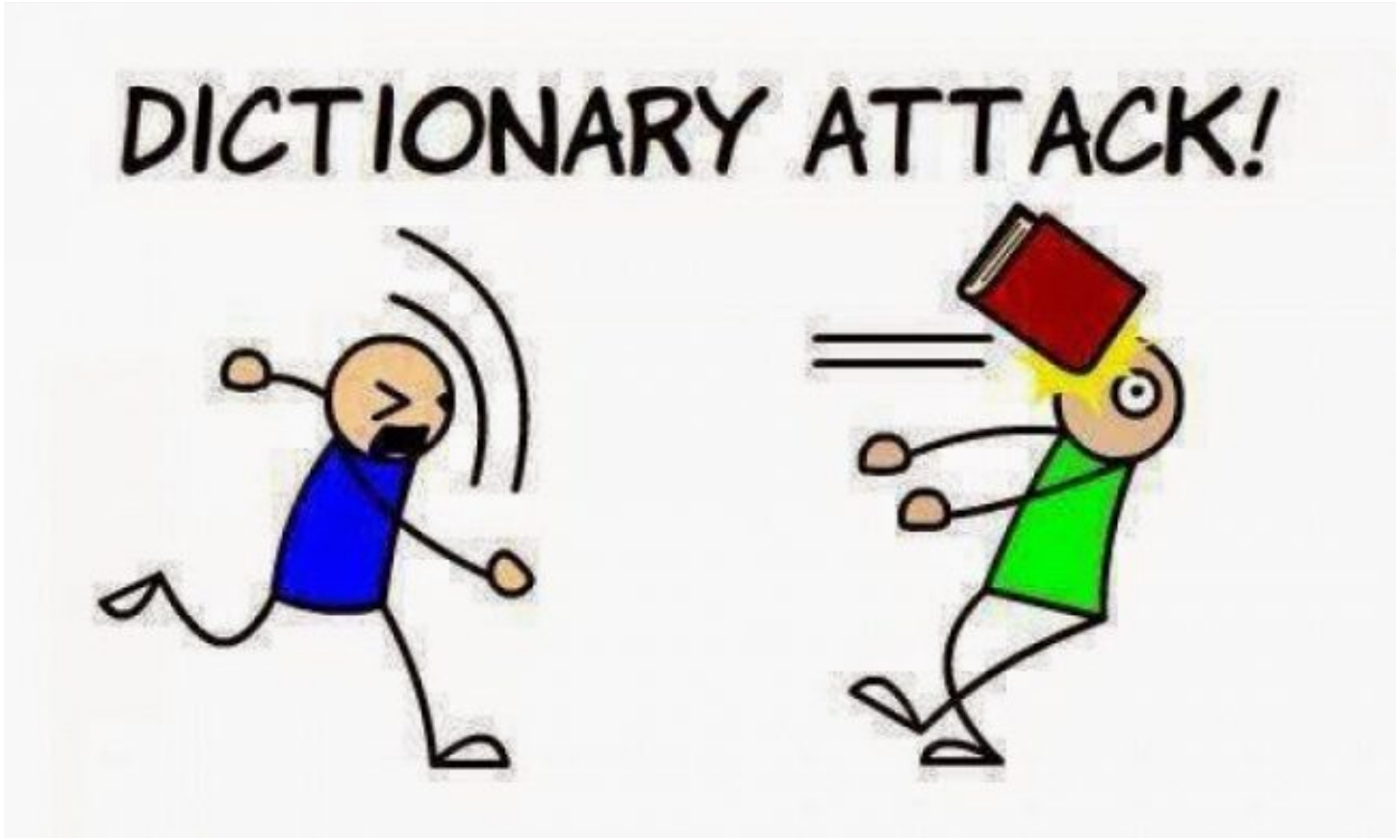


Patterns Selection

DEEPSEC



From The Developer’s Point Of View



Step 2: Threat Modeling YAML File

DEEPSEC

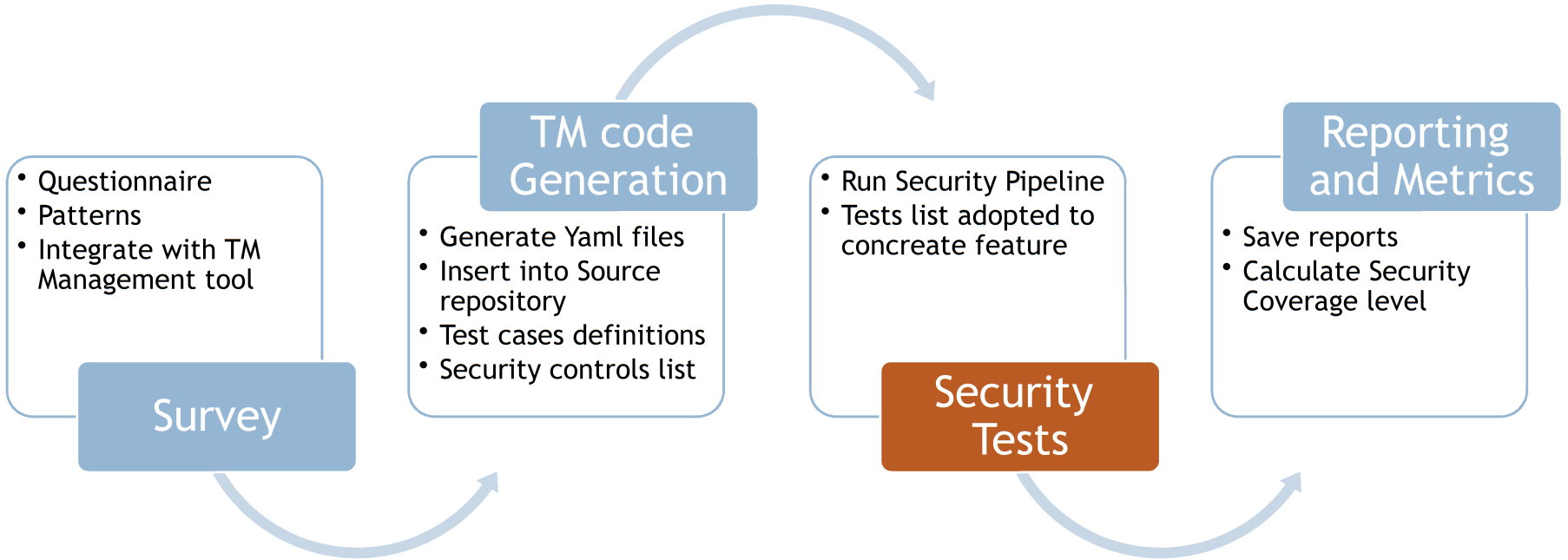
```yaml
login_user: #this is the short_name for the User Story/Functionality
  description: |
    As an employee of the organization,
    I would like to login to the Expense Management application to submit and upload ex
  abuse_cases: # The Key for all abuse cases under the User Story/Functionality
    external_attacker_account_takeover: #unique name for Abuser Story
      description: As an external attacker, I would compromise a single/multiple user a
      threat_scenarios: # Key for all Threat Models under Abuser Story
        sql injection user account access: #Unique Threat Model Name
          description: External Attacker may be able to gain access to user accounts by
          severity: 3 #value from 0-3 0 is lowest on severity, and 3 is High Severity
          cwe: 89,90 #required CWEID if you want to correlate and map to vulnerabilitie
          cases: #linked test cases (optional)
            - sql_injection_auto #the same name as the ones in security_tests
            - sql_injection_manual
            - sql_injection_sqlmap
            - generic_error_messages
        end user weak password:
          description: External attacker may be able to bypass user authentication by c
          severity: 2
          cwe: 521
          cases:
            - default_passwords
            - bruteforce_login
```

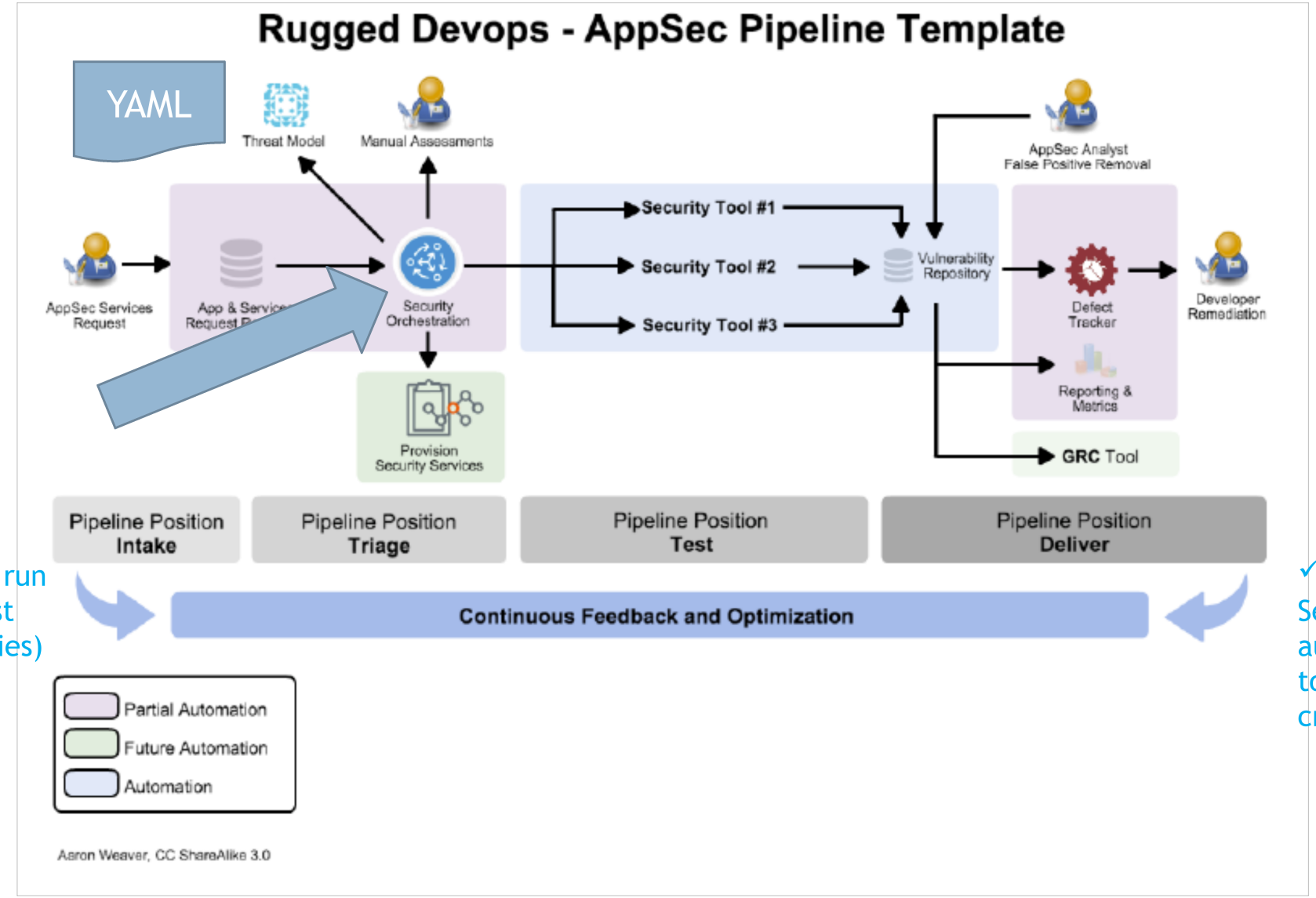
Source: ThreatPlaybook



Step 3: Risk Based Security Test Orchestration

DEEPSEC





✓ Deploy stage will run parallel Security Test pipeline (Asynchronies)

✓ Approval Gate will check Security Test coverage and automatically approve push to production if security criteria achieved

Step 4: CI/CD Integration (Example)

DEEPSEC

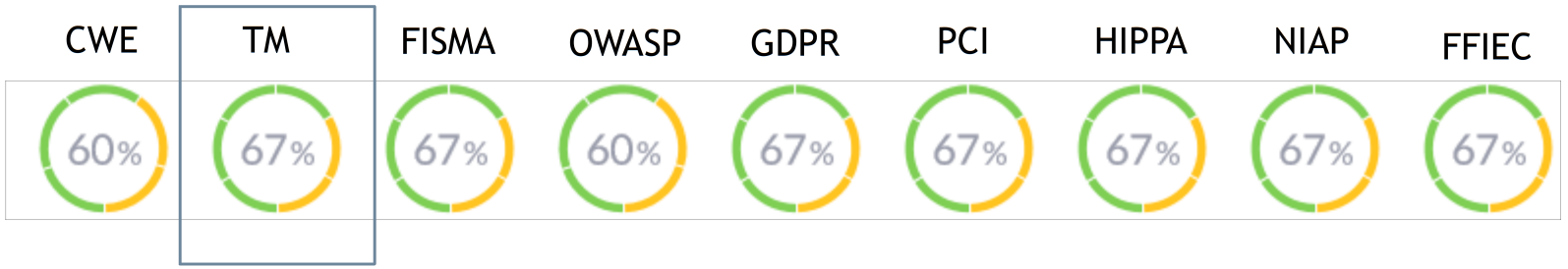
- Process will be triggered by “Security Test” step as part of Build stage
  - Pull Threat Modeling process from Git (If not exists – process will be stopped! As a Gate)
  - Run Security test pipeline (as parallel to functional pipeline)
- During any “final” stage (post-functional tests)
  - Validate if security tests flow finished
  - Check security coverage (As a Gate)
  - If vulnerabilities found – open tickets inside defect management system (Jira)
- During release approve stage
  - **Automatically approve!** If has a good coverage and suitable vulnerabilities score
  - Manual approve only need if security thresholds violated

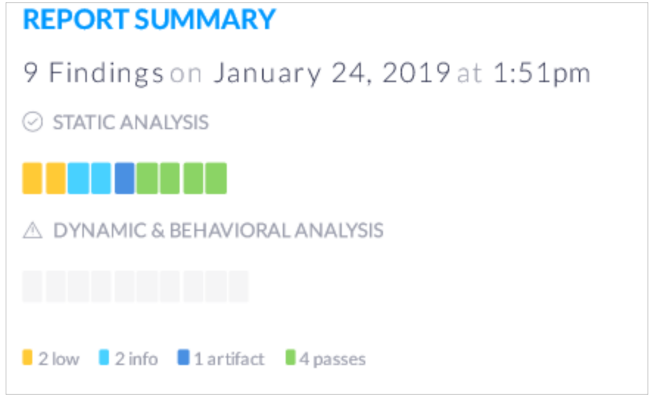


Coverage Review

DEEPSEC

- Coverage calculated by automation tool
- Ticket should be opened automatically with all context needed
- Report contains all tests and findings in relation to Threat Modeling use cases



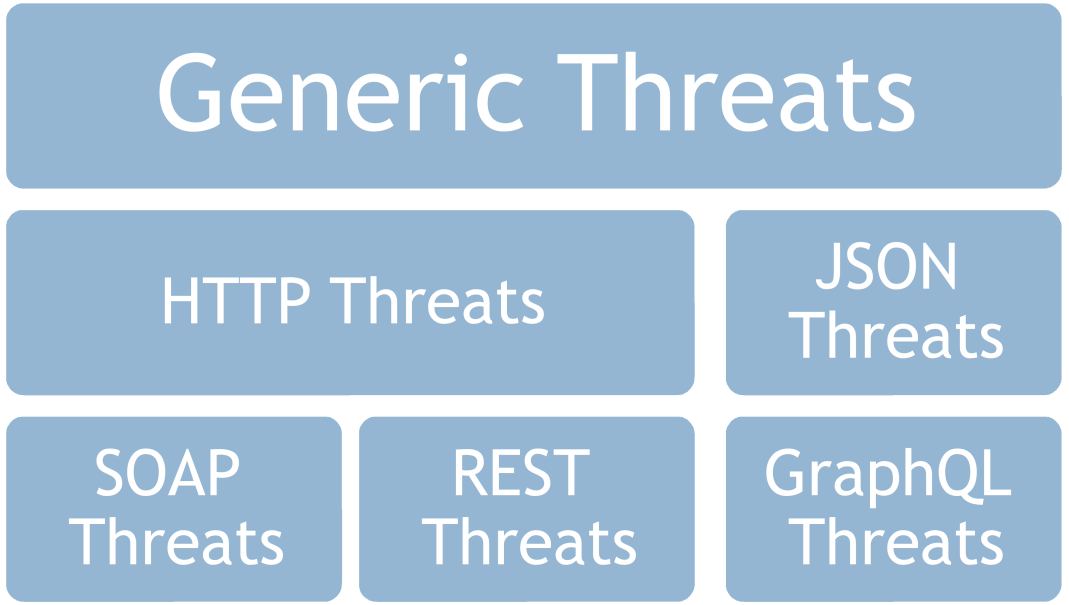


Main Points (Summary)

DEEPSEC

- Based on OWASP ASVS v2 or other standards
- Part of feature onboarding
- Responsibility moved to R&D
- Integrated part of CI/CD flow
- Managed by security specialists

✓ **Template based approach!**





Thinks To Warry About

DEEPSEC

- Scope – Feature/User story
- **No single framework exists**
- **No Threat Modeling orchestration specification**
- Evaluation plan (synchronies vs asynchronies)
- **Integrate manual Threat Modeling process be part of orchestration**



Toolset (Example)

DEEPSEC

- ChatOps Slak – Uses as semi-automated check list questioner
- ThreatPlaybook – Automation for Threat Modeling process and documents
- ThreatModeler – Threats generation based on DFD's
- Jenkins – CI/CD pipeline or Security Pipeline
- Robot – Security Pipeline
- BDD – Security Test (behavior driven)
- Jira – Features and Defects Management system
- IriusRisk – Management framework and Threat Modeling automation
- Orchestron – Management Tool
- Security Compass – Management Tool
- ZeroNorth – tests orchestrator and reports management system



DEEPSEC

Thank You!

vitaly.davidoff@citi.com
LinkedIn: https://www.linkedin.com/in/vitaly-davidoff-07039a1

