Security in DevOps

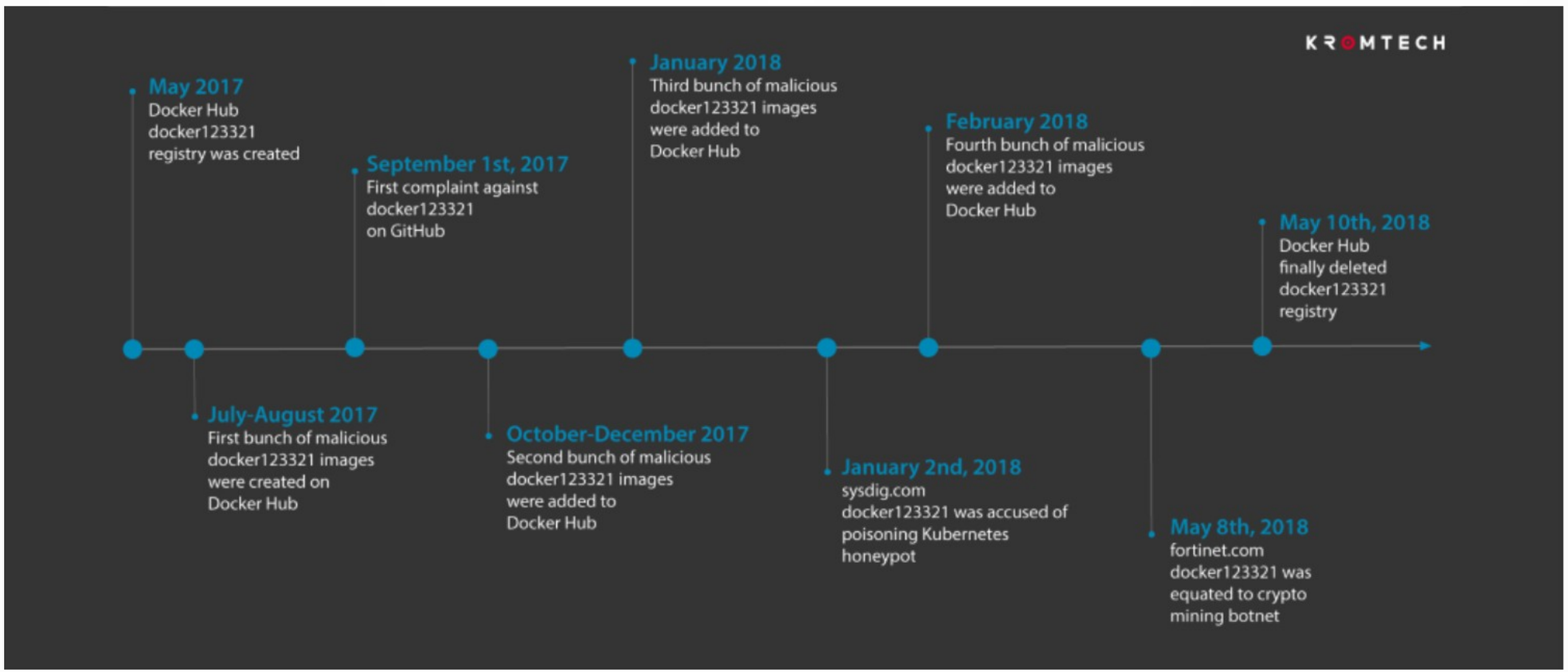
Who am I

- Floris Meester
- Security consultant/Trainer/Coder
- CISSP, CEH, CEI
- floris@tekkamaki.nl

Challenges

- Traditional security works against DevOps agility (end of the chain)
- Security vendors only sell endpoint and perimeter security
- Use of third party or OSS libraries
- Automated pull of dependencies
- Containers from public registries

Challenges


KROMTECH
May 2017
Docker Hub
docker123321
registry was created
July-August 2017
First bunch of malicious
docker123321 images
were created on
Docker Hub
September 1st, 2017
First complaint against
docker123321
on GitHub
October-December 2017
Second bunch of malicious
docker123321 images
were added to
Docker Hub
January 2018
Third bunch of malicious
docker123321 images
were added to
Docker Hub
January 2nd, 2018
sysdig.com
docker123321 was accused of
poisoning Kubernetes
honeypot
February 2018
Fourth bunch of malicious
docker123321 images
were added to
Docker Hub
May 8th, 2018
fortinet.com
docker123321 was
equated to crypto
mining botnet
May 10th, 2018
Docker Hub
finally deleted
docker123321
registry

Challenges

Well, That Escalated Quickly!
How Abusing Docker API Led to Remote Code Execution, Same Origin Bypass and Persistence in The Hypervisor via Shadow Containers

Challenges

- Rapid development and deployment of applications/microservices
- New technologies have lesser-known vulnerabilities/weaknesses
- Containers are **not** a security solution
- Cloud environments do **not** outsource risk

Challenges

runc - Malicious container escape - CVE-2019-5736

Public Date: February 11 2019 at 12:00 AM
Updated Tuesday at 11:57 AM - English

Twelve malicious Python libraries found and removed from PyPI

One package contained a clipboard hijacker that replaced victims' Bitcoin addresses in an attempt to hijack funds from users.

possibly

compromised modues

14%

of npm modules

50M/month

downloads count of impacted modules

Challenges - OWASP top 10

1. SQL Injection
2. Broken Authentication
3. Data exposure
4. XEE
5. Broken Access control
6. Misconfiguration
7. XSS
8. Insecure deserialization
9. Known vulnerabilities
10. Insufficient logging/monitoring

3 Pillars of InfoSec (CIA)

- Confidentiality
- Integrity
- Availability/Authentication
- Non repudiation

DevSecOps/SecDevOps/DevopsSec

- Developers and operators with security functions
- Introduce security early in the SDLC
- Introduce security in every part of the SDLC
- Automate security controls/processes where possible
- Firewalling, SDN endpoint security, vulnerability testing
- Integrate in development and CI/CD pipeline so agility is not lost

DevSecOps/SecDevOps/DevopsSec

- Everybody in the team is responsible for security
- Business should be aligned, security is not only about risk !
- Create security policies for DevOps
- Introduce effective AppSec tools with ease of use
- AppSec tools should provide reports with clear actions

DevSecOps possible solutions

- Use microservices (with containers)
- Treat containers apps as if they were on the host (namespaces)
- Setup infrastructure as code (Iac)
- Setup network as code (for instance SDN solutions)
- Use proven and trusted frameworks
- Security professionals should enable development to use tools

Security compliance

- ISO 27001
  - Access control
  - Cryptography
  - Operation security
  - Communication security
  - Business continuity management
  - Incident management
  - Compliance with internal policies and external laws
  - Cloud service control

Security compliance

- FIPS standard for cryptographic modules (HSM)
- OWASP cryptographic storage and key management cheat sheet aimed at developers
- Cloud security alliance
  CCM (Cloud Controls Matrix)
  - Includes FedRAMP, ISO 27001, NIST and PCI
  - CAIQ self assesment questions

Security compliance

- Center for internet security (CIS)
  - National checklist program (NCP)
    - OS
    - Database
    - Virtualization
    - Applications

Security compliance

National Checklist Program Repository

The National Checklist Program (NCP), defined by the NIST SP 800-70, is the U.S. government repository of publicly available security checklists (or benchmarks) that provide detailed low level guidance on setting the security configuration of operating systems and applications.

NCP provides metadata and links to checklists of various formats including checklists that conform to the Security Content Automation Protocol (SCAP). SCAP enables validated security products to automatically perform configuration checking using NCP checklists. For more information relating to the NCP please visit the information page or the glossary of terms. Please note that the current search fields have been adjusted to reflect NIST SP 800-70 Revision 4.

Search for Checklists using the fields below. The keyword search will search across the name, and summary.

Checklist Type:	Compliance	**Content Type:**	Any.....
Authority:	Any.....	**Tool Compatibility:**	Any.....
Target:	Any.....	**Keyword:**	apache

Search Reset

There are **559** matching records. Displaying matches **1** through **20**.

1 2 3 4 5 6 7 8 9 10 > >>

vijfhart IT-OPLEIDINGEN

TEKKAMAKI SECURITY

Security compliance

- GDPR
  - Lawfulness, transparency, fairness
  - Purpose limitation
  - Accuracy of data
  - Data minimization
  - Integrity/confidentiality
  - Storage limitation

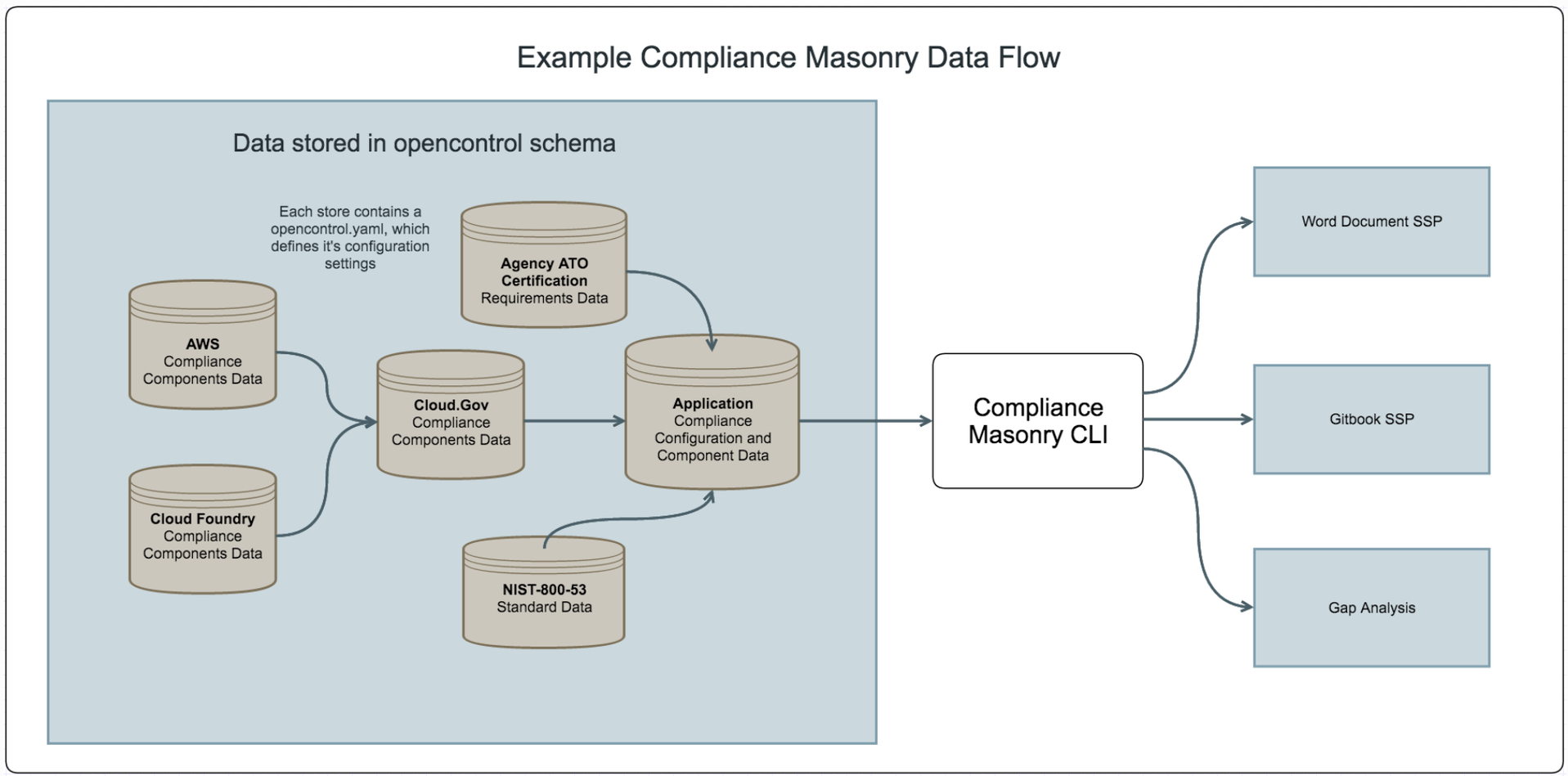
Security compliance

- Cloud security alliance (CSA) top cloud security issues
  - Data breaches
  - Weak IAM
  - Insecure API’s
  - Application vulnerabilities
  - Account hijack
  - Malicious insider
  - APT
  - Data loss

Tooling

- OpenSCAP tools (baseline/continues scan/compliance)
- CIS-CAT Lite (compliance testing)
- OSSEC (hids/file monitoring)
- Dev-sec hardening (benchmark/hardening automation)
- Hashicorp Vault (secrets management, data in rest)
- OWASP Dependency Check (library dependency check)
- Retire.js (javascript libraries)
- Inspec (compliance)
- Gauntlt (hooks in security tools, easy plain text configuration)
- OpenVAS (CVE monitoring/vulnerability scanning)
- Opencontrol/Compliance Masonry

Tooling - Masonry



Tooling - Masonry

```yaml
name: Name of the component
key: Key of the component (defaults to the filename if not present)
documentation_complete: Manual check if the documentation is complete (for gap analysis)
schema_version: 3.0.0
references:
  - name: Name of the reference ie. EC2 website
    path: Relative path of local file or URL ie. diagrams/diagram-1.png
    type: Type of reference ie. Image, URL
  - name: Name of the reference ie. EC2 website
    path: Relative path of local file or URL ie. diagrams/diagram-1.png
    type: Type of reference ie. Image, URL
verifications:
  - key: Key of verification
    name: Name of verification
    path: Relative path of local file or URL ie. diagrams/diagram-1.png
    type: Type of reference ie. Image, URL
  - key: Key of verification
    name: Name of verification
    path: Relative path of local file or URL ie. diagrams/diagram-1.png
    type: Type of reference ie. Image, URL
satisfies:
  - standard_key: Standard Key (NIST-800-53)
    control_key: Control Key (CM-2)
    narrative:
      - key: The optional key that represents a particular section of the control. If the key is not speci
        text: The narrative text for the particular section / entire control if there is no key specified
    implementation_statuses:
      - Used for gap analysis, can only be one of the following:
      - partial
      - planned
      - complete
      - none
    control_origins:
      - shared
```

Security coding frameworks

- Python
  - Flask Security
- ASP.NET
  - ASP.NET Core
- NodeJS
  - Passport Framework
- Ruby
  - Devise Security
- Java
  - Spring Security
  - Shiro

Data Governance

- RBAC
- ABAC
- Classification by metadata

Threat Modeling

- STRIDE
  - Spoofing – credentials, certificates
  - Tampering – hashing, digital signatures
  - Repudiation – logging, authentication
  - Information Disclosure – encryption, RBAC, ABAC
  - DOS – load balancing, wasstraat
  - Escalation of privileges - authorization

Threat Libraries - CAPEC

CAPEC™ Common Attack Pattern Enumeration and Classification
A Community Resource for Identifying and Understanding Attacks

Home > CAPEC List > CAPEC-1000: Mechanisms of Attack (Version 3.0)

Home | About | CAPEC List | Community | News | Search

CAPEC-1000: Mechanisms of Attack

View ID: 1000
Structure: Graph

Objective

This view organizes attack patterns hierarchically based on mechanisms that are frequently employed when exploiting a vulnerability. The categories that are members of this view represent the different techniques used to attack a syste
patterns to align with more than one category depending on one's perspective. To counter this, emphasis was placed such that attack patterns as presented within each category use a technique not sometimes, but without exception.

Relationships

The following graph shows the tree-like relationships between attack patterns that exist at different levels of abstraction. At the highest level, categories exist to group patterns that share a common characteristic. Within categories, meta
Below these are standard and detailed level patterns that are focused on a specific methodology or technique used.

Expand All | Collapse All

1000 - Mechanisms of Attack

- Engage in Deceptive Interactions - *(156)*
- Abuse Existing Functionality - *(210)*
- Manipulate Data Structures - *(255)*
- Manipulate System Resources - *(262)*
- Inject Unexpected Items - *(152)*
- Employ Probabilistic Techniques - *(223)*
- Manipulate Timing and State - *(172)*
- Collect and Analyze Information - *(118)*
- Subvert Access Control - *(225)*

Threat Libraries - CWE

Common Weakness Enumeration
A Community-Developed List of Software Weakness Types

Home > CWE List > CWE- Individual Dictionary Definition (3.2) ID Lookup:

Home | About | CWE List | Scoring | Community | News | Search

CWE VIEW: Development Concepts

View ID: 699
Type: Graph
Status: Incomplete

Downloads: Booklet | CSV | XML

Objective

This view organizes weaknesses around concepts that are frequently used or encountered in software development. Accordingly, this view can align closely with the perspectives of developers, educators, and assessment vendors. It provides a variety of categories that are intended to simplify navigation, browsing, and mapping.

Audience

Stakeholder	Description
Software Developers	Software developers use this view to better understand potential mistakes that can be made in specific areas of their code. The use of concepts that developers are familiar with makes it easier to navigate.
Educators	Educators use this view to teach future developers about the types of mistakes that are commonly made within specific parts of a codebase.

Relationships

The following graph shows the tree-like relationships between weaknesses that exist at different levels of abstraction. At the highest level, categories and classes exist to group weaknesses. A category is a CWE entry that contains a set of other entries that share a common characteristic. Classes are weaknesses that are described in a very abstract fashion, typically independent of any specific language or technology and are more general than a base weakness. Within classes, base level weaknesses are used to present a more specific type of weakness that is still mostly independent of a resource or technology, but with sufficient details to provide specific methods for detection and prevention. A variant is a weakness that is described at a very low level of detail, typically limited to a specific language or technology. A chain is a set of weaknesses that must be reachable consecutively in order to produce an exploitable vulnerability. A composite is a set of weaknesses that must all be present simultaneously in order to produce an exploitable vulnerability.

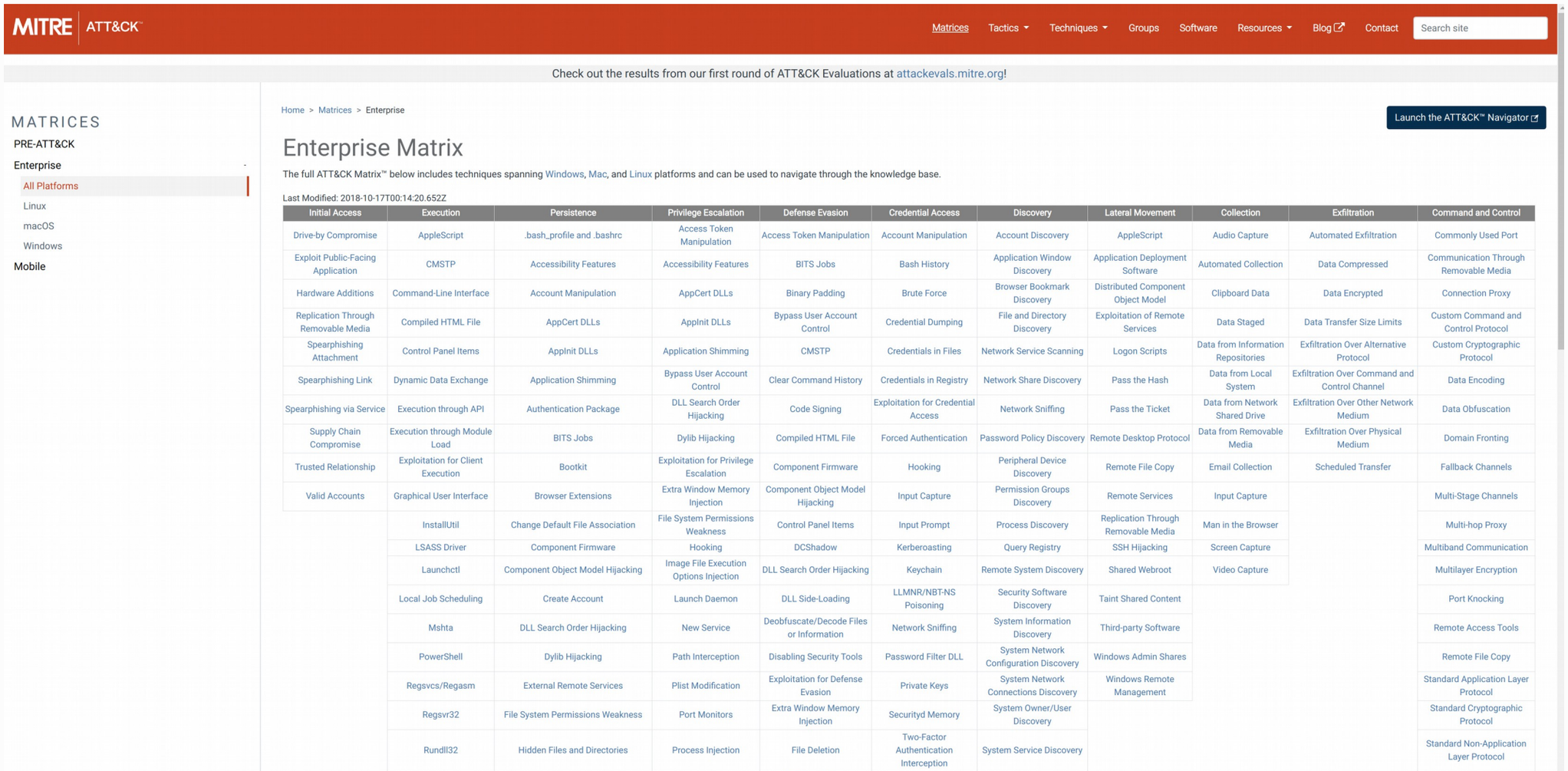
Show Details:

Expand All | Collapse All

699 - Development Concepts

- Configuration - *(16)*
  - J2EE Environment Issues - *(4)*
  - .NET Environment Issues - *(519)*
- Data Processing Errors - *(19)*
- Pathname Traversal and Equivalence Errors - *(21)*
- Numeric Errors - *(189)*
- 7PK - Security Features - *(254)*
- 7PK - Time and State - *(361)*
- Error Conditions, Return Values, Status Codes - *(389)*
- Resource Management Errors - *(399)*
- Channel and Path Errors - *(417)*
- Handler Errors - *(429)*
- Behavioral Problems - *(438)*
- Business Logic Errors - *(840)*
- Web Problems - *(442)*
- User Interface Security Issues - *(355)*
- Initialization and Cleanup Errors - *(452)*
- Pointer Issues - *(465)*
- Mobile Code Issues - *(490)*
- Often Misused: Arguments and Parameters - *(559)*
- Expression Issues - *(569)*
- Violation of Secure Design Principles - *(657)*
- Bad Coding Practices - *(1006)*

Threat Libraries - ATT&CK



MITRE | ATT&CK

Matrices Tactics Techniques Groups Software Resources Blog Contact Search site

Check out the results from our first round of ATT&CK Evaluations at attackevals.mitre.org!

MATRICES

PRE-ATT&CK

Enterprise

All Platforms

Linux

macOS

Windows

Mobile

Home > Matrices > Enterprise

Launch the ATT&CK™ Navigator

Enterprise Matrix

The full ATT&CK Matrix™ below includes techniques spanning Windows, Mac, and Linux platforms and can be used to navigate through the knowledge base.

Last Modified: 2018-10-17T00:14:20.652Z

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Exfiltration	Command and Control
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Automated Exfiltration	Commonly Used Port
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	BITS Jobs	Bash History	Application Window Discovery	Application Deployment Software	Automated Collection	Data Compressed	Communication Through Removable Media
Hardware Additions	Command-Line Interface	Account Manipulation	AppCert DLLs	Binary Padding	Brute Force	Browser Bookmark Discovery	Distributed Component Object Model	Clipboard Data	Data Encrypted	Connection Proxy
Replication Through Removable Media	Compiled HTML File	AppCert DLLs	AppInit DLLs	Bypass User Account Control	Credential Dumping	File and Directory Discovery	Exploitation of Remote Services	Data Staged	Data Transfer Size Limits	Custom Command and Control Protocol
Spearphishing Attachment	Control Panel Items	AppInit DLLs	Application Shimming	CMSTP	Credentials in Files	Network Service Scanning	Logon Scripts	Data from Information Repositories	Exfiltration Over Alternative Protocol	Custom Cryptographic Protocol
Spearphishing Link	Dynamic Data Exchange	Application Shimming	Bypass User Account Control	Clear Command History	Credentials in Registry	Network Share Discovery	Pass the Hash	Data from Local System	Exfiltration Over Command and Control Channel	Data Encoding
Spearphishing via Service	Execution through API	Authentication Package	DLL Search Order Hijacking	Code Signing	Exploitation for Credential Access	Network Sniffing	Pass the Ticket	Data from Network Shared Drive	Exfiltration Over Other Network Medium	Data Obfuscation
Supply Chain Compromise	Execution through Module Load	BITS Jobs	Dylib Hijacking	Compiled HTML File	Forced Authentication	Password Policy Discovery	Remote Desktop Protocol	Data from Removable Media	Exfiltration Over Physical Medium	Domain Fronting
Trusted Relationship	Exploitation for Client Execution	Bootkit	Exploitation for Privilege Escalation	Component Firmware	Hooking	Peripheral Device Discovery	Remote File Copy	Email Collection	Scheduled Transfer	Fallback Channels
Valid Accounts	Graphical User Interface	Browser Extensions	Extra Window Memory Injection	Component Object Model Hijacking	Input Capture	Permission Groups Discovery	Remote Services	Input Capture		Multi-Stage Channels
	InstallUtil	Change Default File Association	File System Permissions Weakness	Control Panel Items	Input Prompt	Process Discovery	Replication Through Removable Media	Man in the Browser		Multi-hop Proxy
	LSASS Driver	Component Firmware	Hooking	DCShadow	Kerberoasting	Query Registry	SSH Hijacking	Screen Capture		Multiband Communication
	Launchctl	Component Object Model Hijacking	Image File Execution Options Injection	DLL Search Order Hijacking	Keychain	Remote System Discovery	Shared Webroot	Video Capture		Multilayer Encryption
	Local Job Scheduling	Create Account	Launch Daemon	DLL Side-Loading	LLMNR/NBT-NS Poisoning	Security Software Discovery	Taint Shared Content			Port Knocking
	Mshta	DLL Search Order Hijacking	New Service	Deobfuscate/Decode Files or Information	Network Sniffing	System Information Discovery	Third-party Software			Remote Access Tools
	PowerShell	Dylib Hijacking	Path Interception	Disabling Security Tools	Password Filter DLL	System Network Configuration Discovery	Windows Admin Shares			Remote File Copy
	Regsvcs/Regasm	External Remote Services	Plist Modification	Exploitation for Defense Evasion	Private Keys	System Network Connections Discovery	Windows Remote Management			Standard Application Layer Protocol
	Regsvr32	File System Permissions Weakness	Port Monitors	Extra Window Memory Injection	Securityd Memory	System Owner/User Discovery				Standard Cryptographic Protocol
	Rundll32	Hidden Files and Directories	Process Injection	File Deletion	Two-Factor Authentication Interception	System Service Discovery				Standard Non-Application Layer Protocol
				File Permissions						

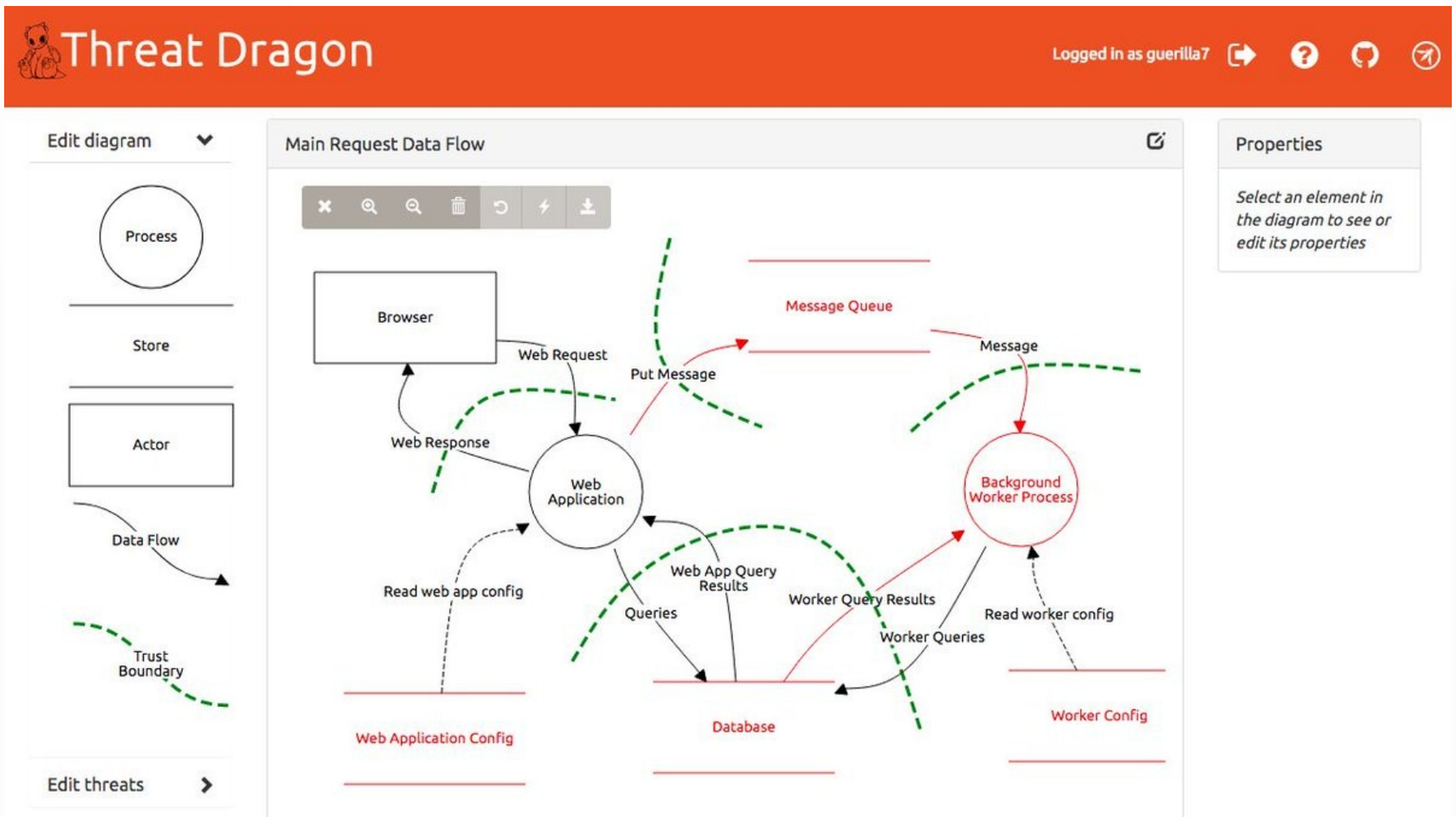
vijfhart IT-OPLEIDINGEN

TEKKAMAKI SECURITY

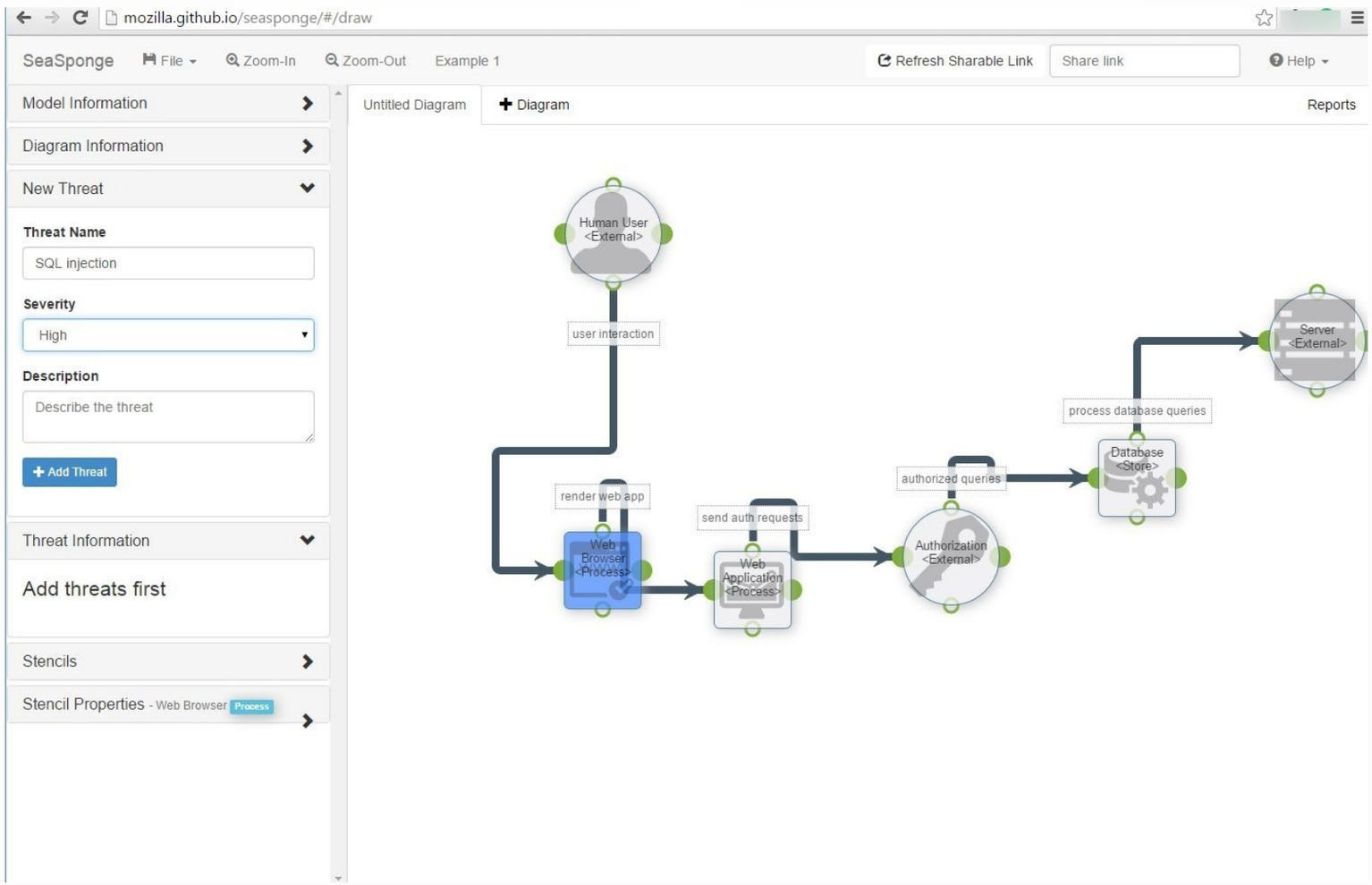
Threat Modeling Tools

- Tools are designed to draw DFD diagrams with trust boundaries and add threat attributes
- Owasp Threat Dragon
- MS threat modeling tool
- Mozilla SeaSponge

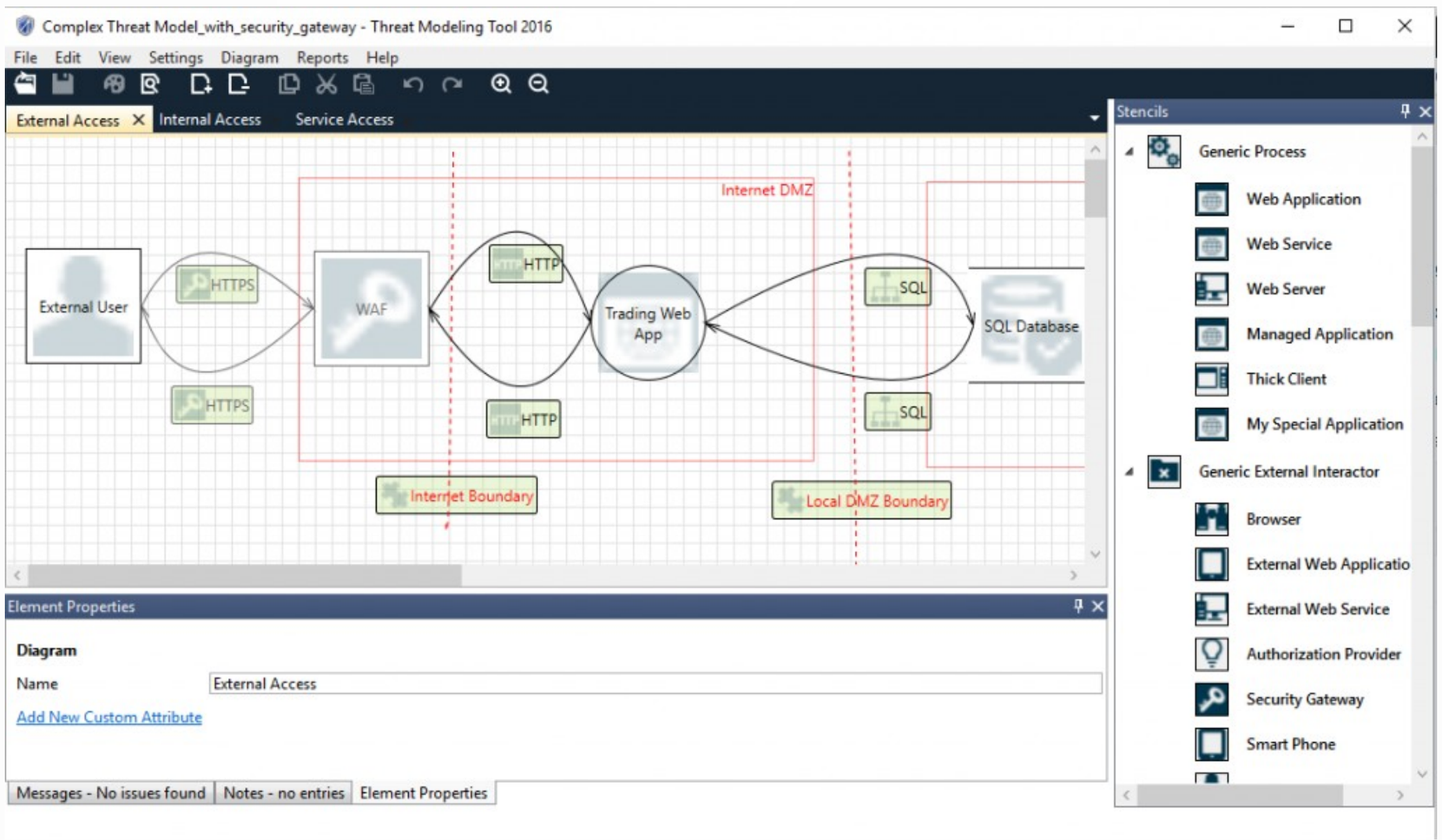
Threat Modeling Tools



Threat Modeling Tools



Threat Modeling Tools



Secure coding standards

- CWE (Insecure examples)
- OWASP Security Knowledge Framework (contains Application security verification standard - ASVS)
- CERT secure coding standards
- OWASP Code Review Project
- Find Security Bugs (plugin for various IDE’s)

Code Scanning Tools

- Retire.js
- Pylint
- SpotBugs IDE plugin
- JSHint
- DREK (Regex scanner)
- Infer (static analyzer for Java/C/C++)
- SonarQube (25+ languages, CI/CD integration)

Secure testing guides

- PCI penetration testing guide
- NIST 800-115 Security testing and Assesment
- OWASP Testing guide

Secure testing tools

- Vulnerability scan – Nessus, OpenVAS
- Port scan - nmap
- Web App scan – Burp, OWASP Zap, Nikto
- Fuzzing - API-fuzzer, Peach
- Github – GittyLeaks, TruffleHog
- SSL/TLS – SSLScan
- SQL injection – SQLMap, SqlninjaVijfhart IT-OPLEIDINGEN

TEKKAMAKI SECURITY

Docker/OCI tools

- Actuary (best practices)
- Clair (CVE scan)
- Anchor Engine (CVE scan)
- Dagda (CVE, NVD analysis)
- Falco (anomaly detection)
- Docker Bench (best practices)

Docker/OCI tools – Docker Bench

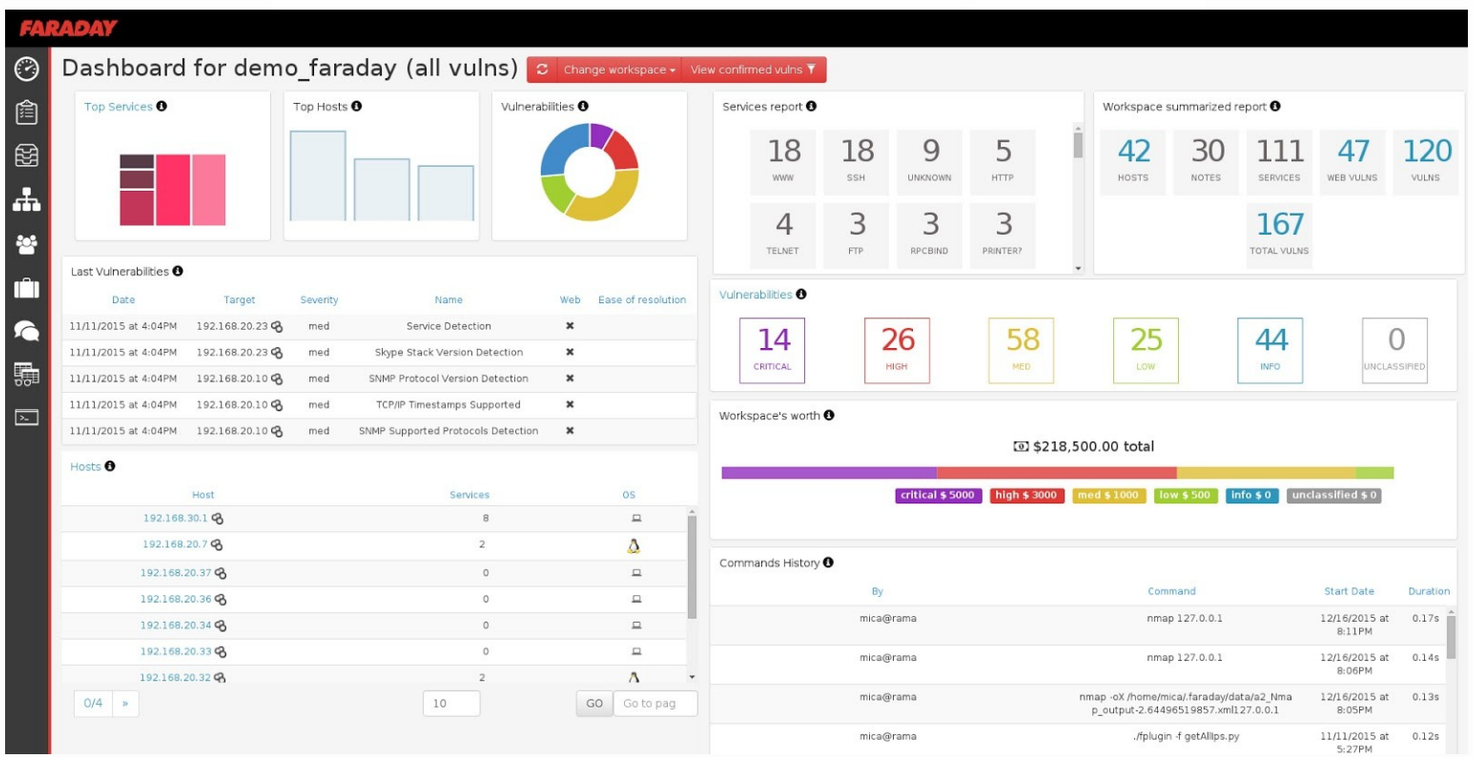
```
# ------------------------------------------------------------------------------
# Docker Bench for Security v1.3.3
#
# Docker, Inc. (c) 2015-
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Inspired by the CIS Docker Community Edition Benchmark v1.1.0.
# ------------------------------------------------------------------------------

Initializing Fri Jul 14 09:18:42 UTC 2017


[INFO] 1 - Host Configuration
[WARN] 1.1  - Ensure a separate partition for containers has been created
[NOTE] 1.2  - Ensure the container host has been Hardened
[PASS] 1.3  - Ensure Docker is up to date
[INFO]      * Using 17.06.0 which is current
[INFO]      * Check with your operating system vendor for support and security maintenance for Docker
[INFO] 1.4  - Ensure only trusted users are allowed to control Docker daemon
[INFO]      * docker:x:992:vagrant
[WARN] 1.5  - Ensure auditing is configured for the Docker daemon
[WARN] 1.6  - Ensure auditing is configured for Docker files and directories - /var/lib/docker
[WARN] 1.7  - Ensure auditing is configured for Docker files and directories - /etc/docker
[WARN] 1.8  - Ensure auditing is configured for Docker files and directories - docker.service
[INFO] 1.9  - Ensure auditing is configured for Docker files and directories - docker.socket
[INFO]      * File not found
[INFO] 1.10 - Ensure auditing is configured for Docker files and directories - /etc/default/docker
[INFO]      * File not found
[INFO] 1.11 - Ensure auditing is configured for Docker files and directories - /etc/docker/daemon.json
[INFO]      * File not found
[WARN] 1.12 - Ensure auditing is configured for Docker files and directories - /usr/bin/docker-containerd
[WARN] 1.13 - Ensure auditing is configured for Docker files and directories - /usr/bin/docker-runc
```

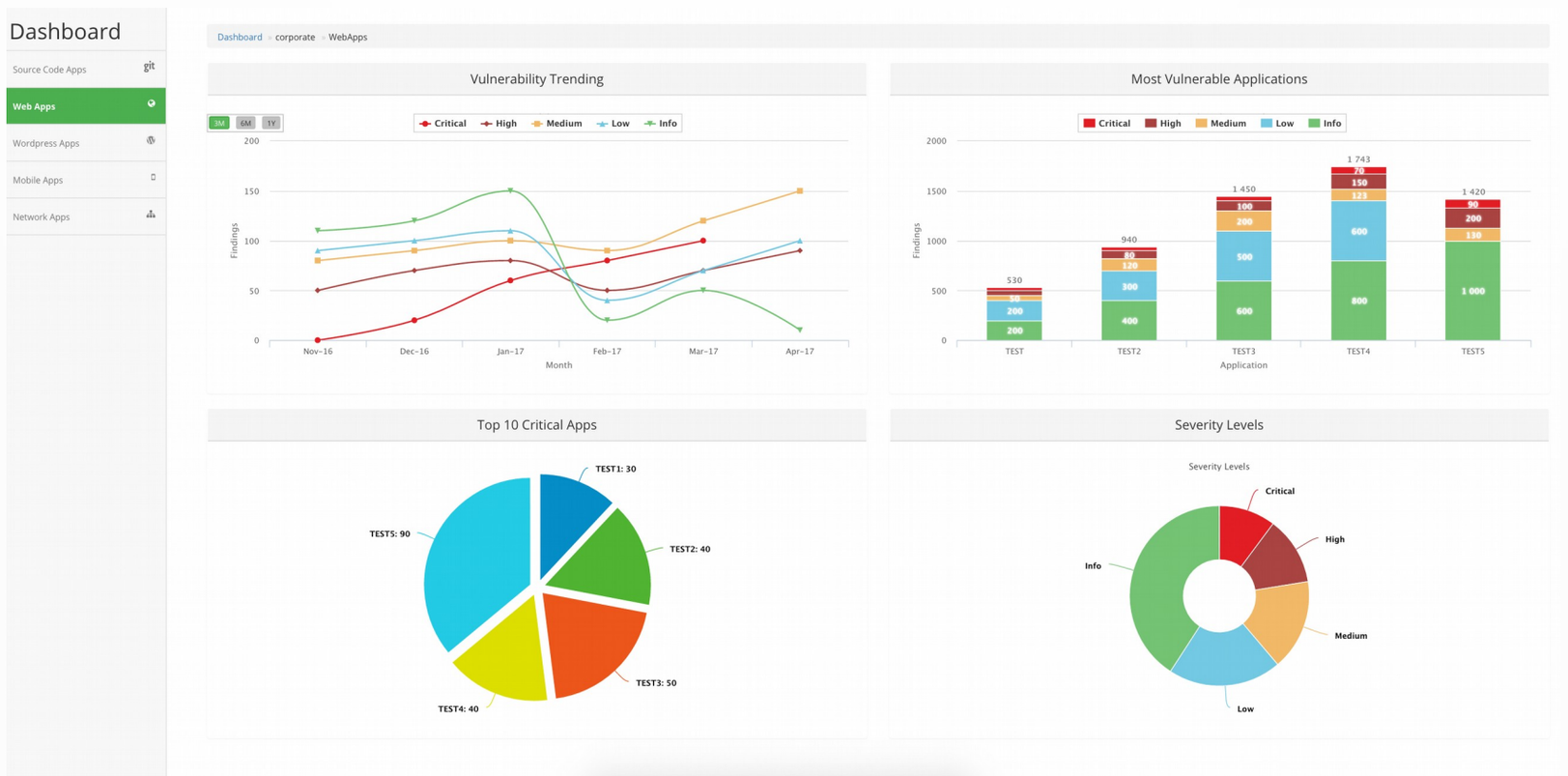
Security integrated tools

- Faraday SEC – integrated multiuser pentest tool



Security integrated tools

- JackHammer – integrated pentest tool



Security integrated tools

- Seccubus
- Offensive web testing framework (OWASP includes NIST tools)
- DefectDojo (OWASP integrate output from various tool in dashboard)

CI integration

- Many tools have “headless” mode, some are available as Jenkins plugin

Zero Trust - Threats

- Perimeter security not sufficient anymore
  - Mobile (inherently insecure !)
  - BYOD
  - Cloud interconnects
  - Containers
  - Virtualization
  - Insiders (permission aggregation, malicious activity etc.)

Zero Trust

- Different model, we assume we get compromised !
  - Use secrets management (like Vault)
  - Authenticate everything and everyone
  - Verify everything and everyone (mfa)
  - Audit everything
  - Least privilege
  - Where possible split responsibilities
  - Need to know
  - Adaptive controls (ABAC)
  - Encrypt everything

What will the course hold

- Infosec theory
- Compliance frameworks theory
- Hands on labs for tracking vulnerabilities
- Hands on labs tooling

Q&A