DEEPSEC

# How To Combat Risks Directly From Within Your IDE

## Agile Threat Modeling

Christian Schneider @cschneider4711 www.Christian-Schneider.net

# `whoami`

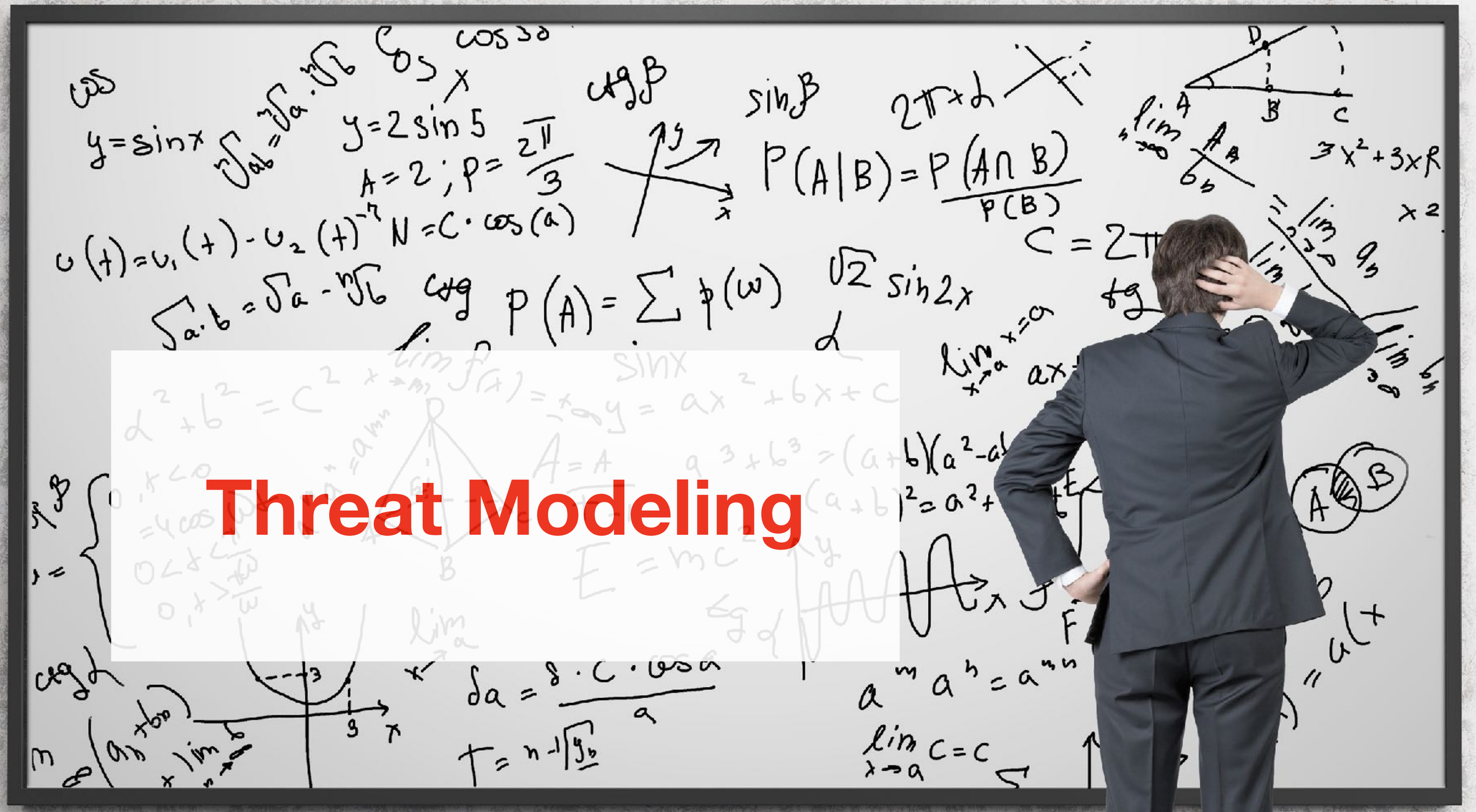
**Christian Schneider**

Security Architect, Pentester, Trainer

Agile Threat Modeling
Security Architecture
DevSecOps
Pentesting

www.Christian-Schneider.net
mail@Christian-Schneider.net
@cschneider4711 on Twitter

# Threat Modeling



Marathon 01b - with thick client - Threat Modeling Tool 2016

File Edit View Settings Diagram Reports Help

Marathon Modell

Corporate Network

Web-Server DMZ

App-Server DMZ

DB-Server DMZ

Office Network

Human User

Browser

HTTPS

Internet Boundary

Web Server: Reverse-Proxy (Apache)

AJP

Web Application: Marathon (Tomcat)

JDBC

File Acess

File System Share (Photos)

SQL Database (Postgres)

HTTPS

Direct SQL Access

Browser

Thick Client: SQL-Client

Admin

Stencils

Generic Process

OS Process

Thread

Kernel Thread

Native Application

Managed Application

Thick Client

Browser Client

Browser and ActiveX Plugins

Web Server

Windows Store Process

Element Properties

**Web Application**

| | |
|---|---|
| Name | Web Application: M |
| Out Of Scope | ☐ |
| Reason For Out Of Scope | |

**Configurable Attributes**

| | |
|---|---|
| Code Type | Unmanaged |

**As Generic Process**

| | |
|---|---|
| Running As | Standard User With |
| Isolation Level | Sandbox |
| Accepts Input From | Local or Network S |
| Implements or Uses an Authentication Mechanism | Yes |
| Implements or Uses an Authorization Mechanism | Yes |
| Implements or Uses a Communication Protocol | No |

Messages No issues found

| Description | Severity | Diagram | Ignore |
|---|---|---|---|

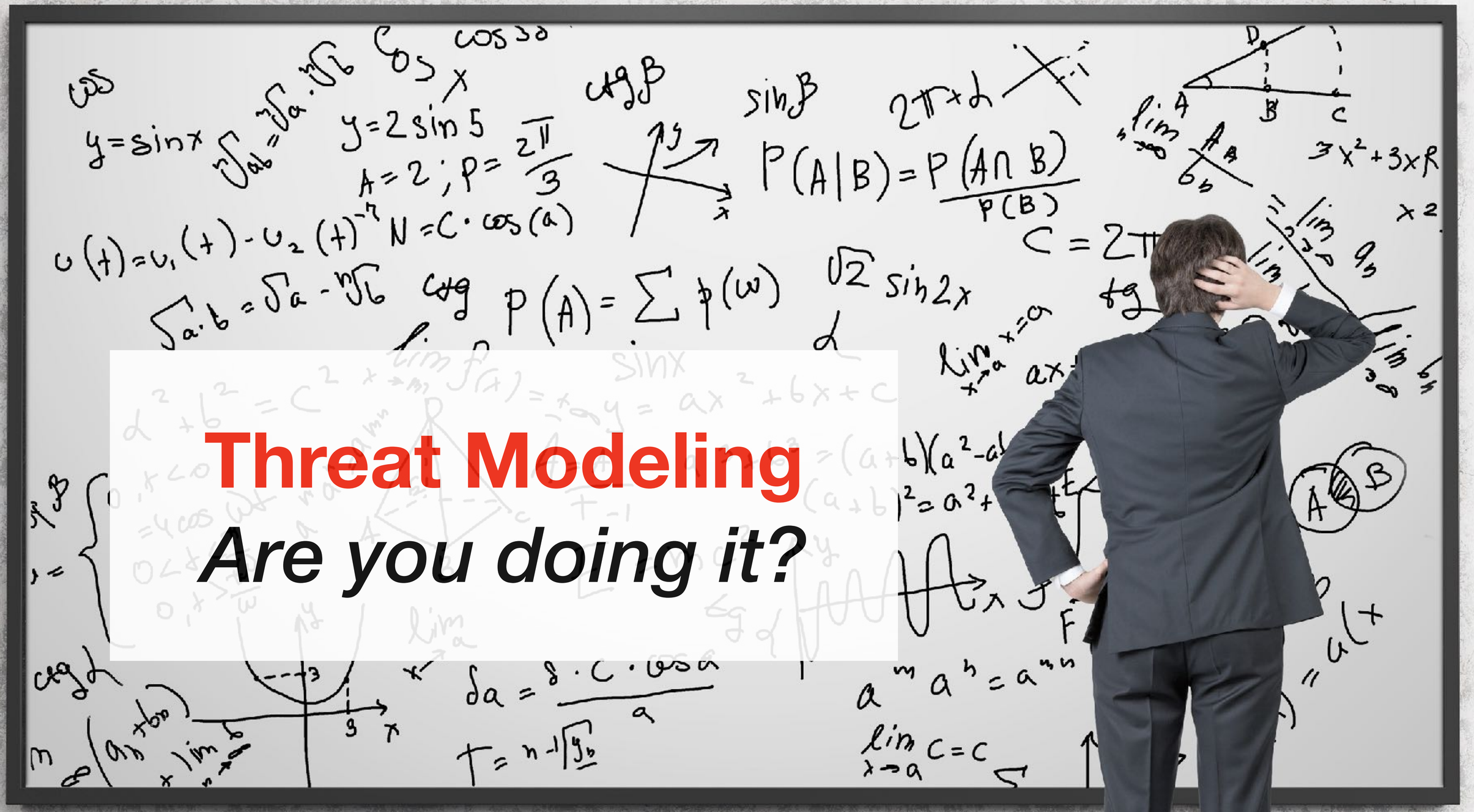
Messages - No issues found | Notes - no entries

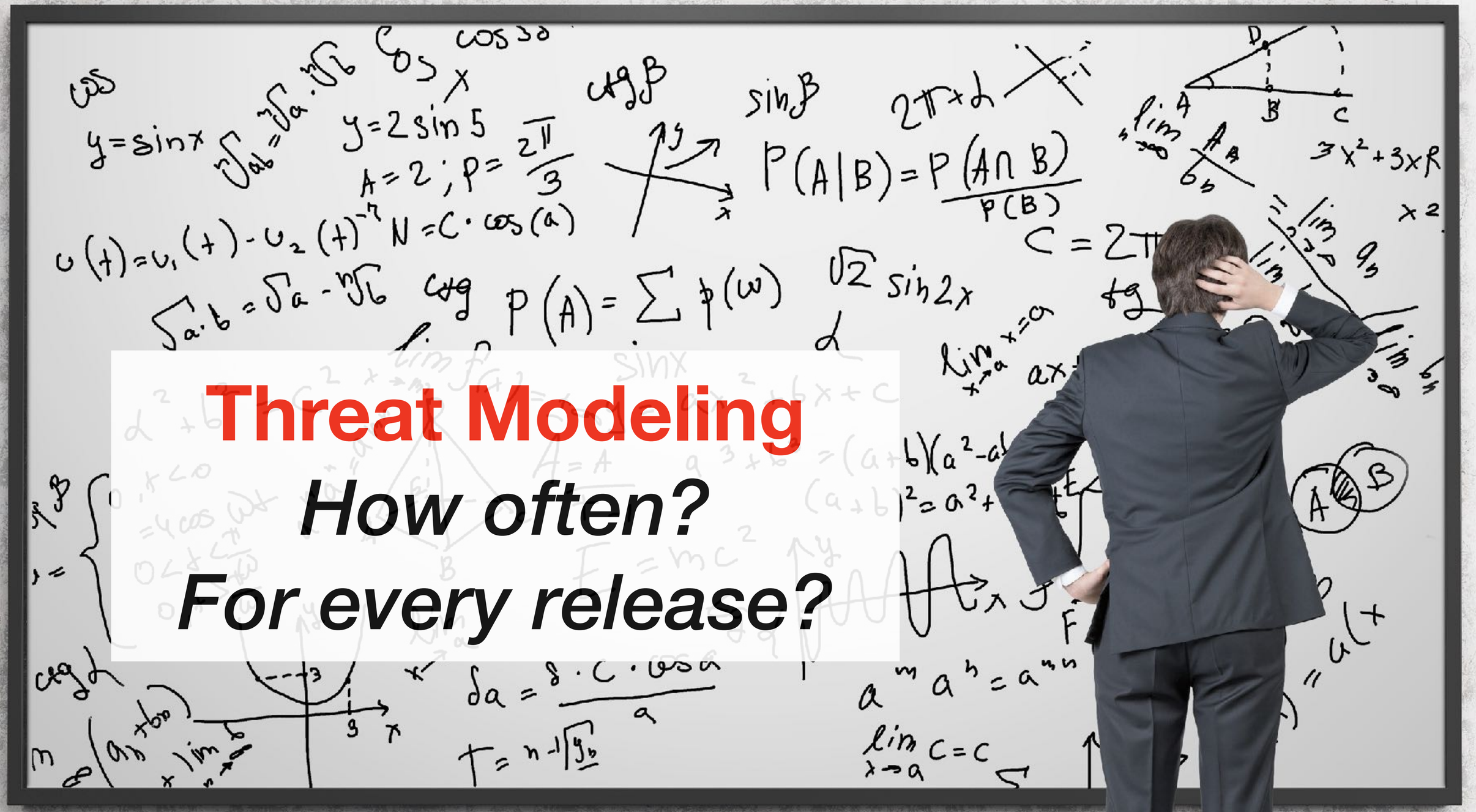
Zur Suche Text hier eingeben

17:43
02.11.2018

# Threat Modeling

*Are you doing it?*

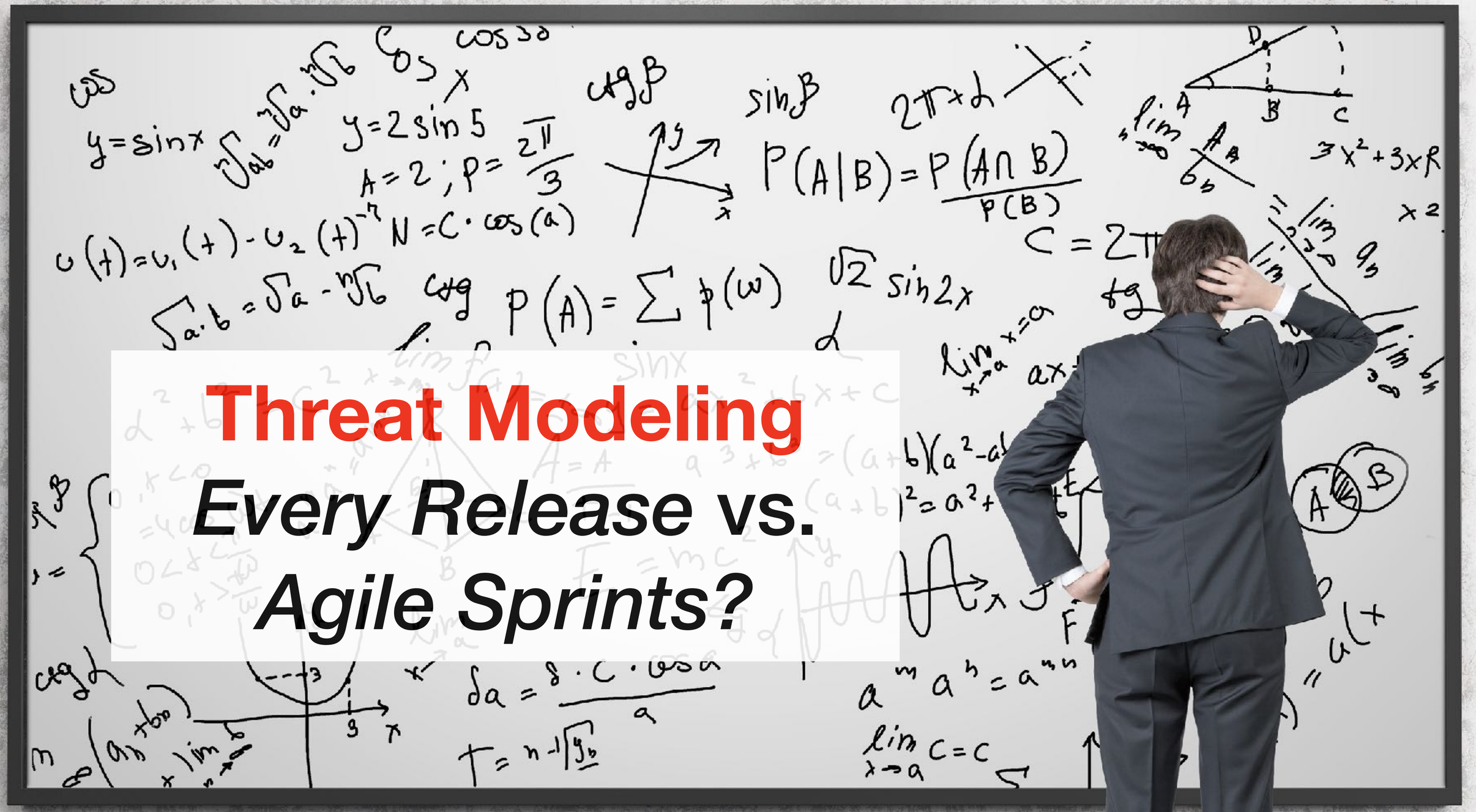




# Threat Modeling

***How often?***

***For every release?***

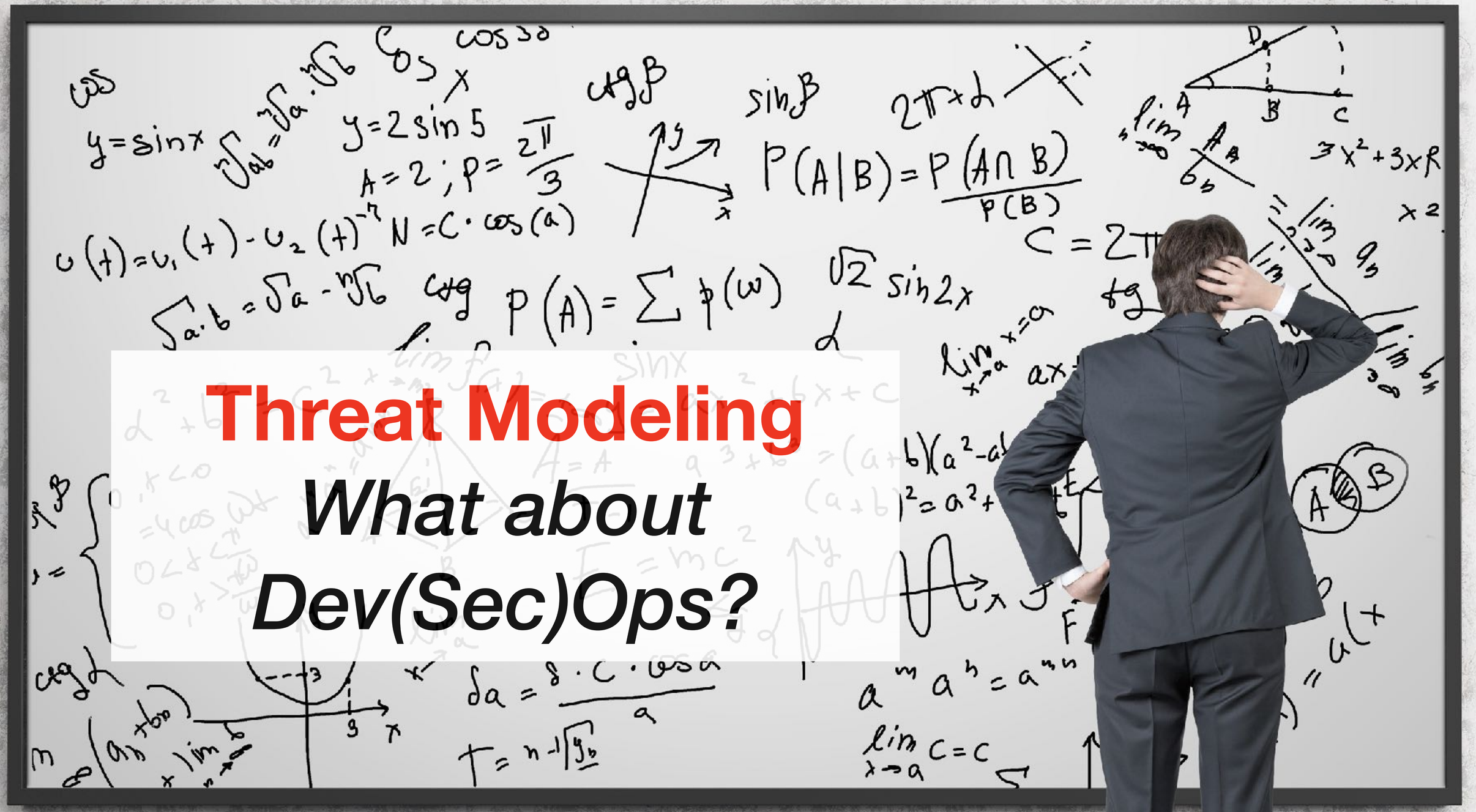


# Threat Modeling

## *Every Release vs. Agile Sprints?*

# Threat Modeling

## *What about Dev(Sec)Ops?*



# DevSecOps

In DevSecOps paradise
everything appears to be code
(or at least some kind of automation magic)

# Threat Models as Code?

Why not let threat models
also be something like code?

# Benefits of Code:

?

# Benefits of Code:

## Editable in any IDE

(even vi or emacs)

# Benefits of Code:

Checked-in into the source tree

# Benefits of Code:

## Diff-able and revert-able

(even branch-able and merge-able when you need to)

# Benefits of Code:

Collaboration-capable

# Benefits of Code:

Testable and verifiable

# Benefits of Code:

Reproducible and repeatable

# Benefits of Code:

Clearly states its most recent update in the revision history

(or the lack thereof)

# Benefits of Code:

**Developers love code**

**(and they know the application best)**

# Benefits of Code:

??? some more ???

# Drawbacks of Code:

?

# Drawbacks of Code:

It’s code…

Someone has to write it…

# Drawbacks of Code:

Some people find code hard to read

(why?)

# Drawbacks of Code:

Starts with the details
not the abstractions

# Drawbacks of Code:

Not easy to spot the "Big Picture"
by looking at the details

# Drawbacks of Code:

??? some more ???

# Threat Modeling

## Dev(Sec)Ops-style

# Idea.

Use some textual simple to read markup language like YAML…

(easier to read than code and understood by all IDEs)

# Idea..

… and in it describe your:

- Data
- Components
- Communication Links
- Trust Boundaries

# Idea...

… and use an open-source tool to analyze it as a graph of connected components with data flowing between them

# Idea....

… which generates nice:

- Model Graphs
- Potential Risks / Threats
- Hardening Recommendations
- Reports / Documentation

(for the compliance folks)

# Agile Threat Modeling

**Idea:** Bridge the gap between ***classic threat modeling*** and ***agile development teams***.

Threat Models as declarative YAML file containing
- Data Assets
- Components
- Communication Links
- Trust Boundaries

Checked-in along with the source-tree.

Benefits of YAML model file: diff-able, collaboration capable, testable, verifiable, …

# Threagile - Agile Threat Modeling Toolkit

***Open-Source*** *on GitHub & DockerHub*

Modeled elements contain technology and protocol type on detailed level.

Threagile analyzes the model YAML file as a graph of connected components with data flowing between them and generates:

- Model Graphs / Diagrams
- Potential Risks / Threats
- Hardening Recommendations
- Reports / Documentation
- … as PDF, Excel, and JSON (for DevSecOps automation in build pipelines)

Custom identified risks (during workshops for example) can be added as well.

# Threagile - Agile Threat Modeling Toolkit

Technology-aware model types

~40 Coded risk rules checking the graph (and growing)

Custom risk rule plugin interface

Calculation of RAA (Relative Attacker Attractiveness) for each component

Calculation of DBP (Data Breach Probability) for each data asset

Model macros to automate certain model modifications

Risk mitigation state maintained in same YAML file

Released as open-source software

# Running Threagile

Either as

- command-line interface (CLI), or
- server with REST API

Available as a Docker container:

```
docker run --rm -it
threagile/threagile
```

```
 _____ _                                  _ _
|_   _| |__   _ __ ___   __ _   __ _(_) | ___
  | | | '_ \| '__/ _ \/ _` |/ _` | | |/ _ \
  | | | | | | | |  __/ (_| | (_| | | |  __/
  |_| |_| |_|_|  \___|\__,_|\__, |_|_|\___|
                               |___/
Threagile - Agile Threat Modeling


Documentation: https://threagile.io
Docker Images: https://hub.docker.com/orgs/threagile
Source Code: https://github.com/threagile
License: Open-Source (MIT License)
Version: 1.0.0 (20200721134459)


Usage: threagile [options]


Options:

  -background string
        background pdf file (default "background.pdf")
  -create-editing-support
        just create some editing support stuff in the output directory
  -create-example-model
        just create an example model named threagile-example-model.yaml in the output directory
  -create-stub-model
        just create a minimal stub model named threagile-stub-model.yaml in the output directory
  -custom-risk-rules-plugins string
        comma-separated list of plugins (.so shared object) file names with custom risk rules to load
  -diagram-dpi int
        DPI used to render: maximum is 240 (default 120)
  -execute-model-macro string
        Execute model macro (by ID)
  -generate-data-asset-diagram
        generate data asset diagram (default true)
```

# First Steps with Threagile

Create either a minimal stub model or a filled example model

The YAML file is the only source of input to Threagile an contains

- Data Assets
- Technical Assets
- Communication Links
- Trust Boundaries

— *and optionally more things*

# Example Model: Data Assets

```yaml
data_assets:



  Customer Contracts: &customer-contracts # this example sho
    id: customer-contracts
    description: Customer Contracts (PDF)
    usage: business # values: business, devops
    tags:
    origin: Customer
    owner: Company XYZ
    quantity: many # values: very-few, few, many, very-many
    confidentiality: confidential # values: public, internal
    integrity: critical # values: archive, operational, impo
    availability: operational # values: archive, operational
```

# Example Model: Technical Assets

```yaml
Apache Webserver:
  id: apache-webserver
  description:
  type: process # values: external-entity, pr
  usage: business # values: business, devops
  used_as_client_by_human: false
  out_of_scope: false
  justification_out_of_scope:
  size: application # values: system, service
  technology: web-server # values: see help
  tags:
    - linux
    - apache
    - aws:ec2
  internet: false
  machine: container # values: physical, virt
  encryption: none # values: none, transparen
  owner: Company ABC
  confidentiality: internal # values: public,
  integrity: critical # values: archive, oper
  availability: critical # values: archive, c
  justification_cia_rating:
  multi_tenant: false
  redundant: false
  custom_developed_parts: true
```

# Example Model:
# Referencing Data Assets (Processed & Stored)

```yaml
data_assets_processed: # sequence of IDs to reference
  - customer-accounts
  - customer-operational-data
  - customer-contracts
  - internal-business-data
data_assets_stored: # sequence of IDs to reference
  - client-application-code
  - server-application-code
data_formats_accepted: # sequence of formats like: json, xml, serialization, file, csv
  - json
  - file
```

# Example Model: Communication Links

```yaml
communication_links:
  ERP System Traffic:
    target: erp-system
    description: Link to the ERP system
    protocol: https # values: see help
    authentication: token # values: none, credentials, session-id, token,
    authorization: technical-user # values: none, technical-user, enduser
    tags:
    vpn: false
    ip_filtered: false
    readonly: false
    usage: business # values: business, devops
    data_assets_sent: # sequence of IDs to reference
      - customer-accounts
      - customer-operational-data
      - internal-business-data
    data_assets_received: # sequence of IDs to reference
      - customer-accounts
      - customer-operational-data
      - customer-contracts
      - internal-business-data
```

# Example Model: Trust Boundaries

```yaml
trust_boundaries:


  Web DMZ:
    id: web-dmz
    description: Web DMZ
    type: network-cloud-security-group # values: see help
    tags:
    technical_assets_inside: # sequence of IDs to reference
      - apache-webserver
      - marketing-cms
    trust_boundaries_nested: # sequence of IDs to reference


  ERP DMZ:
    id: erp-dmz
    description: ERP DMZ
    type: network-cloud-security-group # values: see help
    tags:
      - some-erp
    technical_assets_inside: # sequence of IDs to reference
      - erp-system
      - contract-fileserver
      - sql-database
    trust_boundaries_nested: # sequence of IDs to reference
```

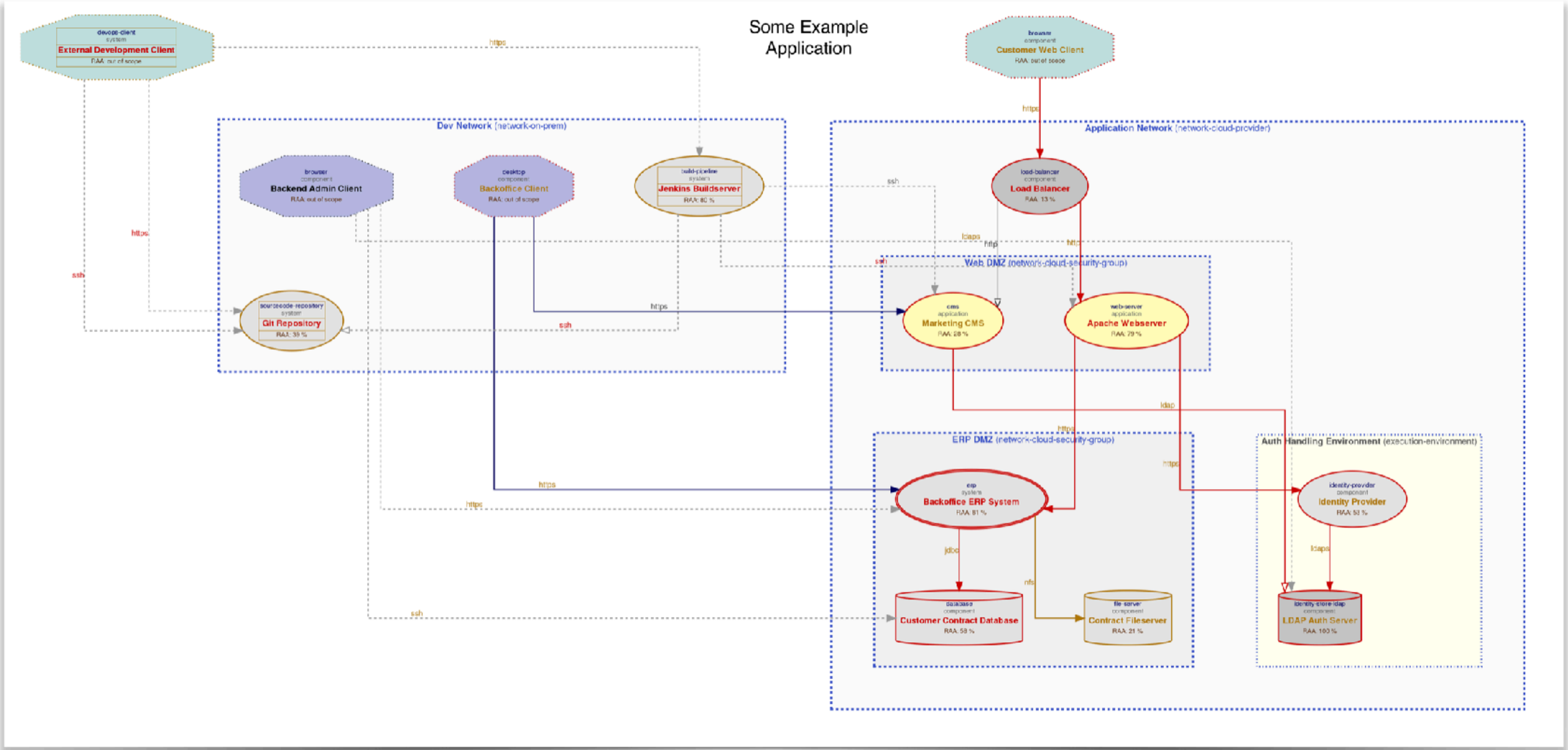
# Execute a Threagile Run

Processes the YAML model file

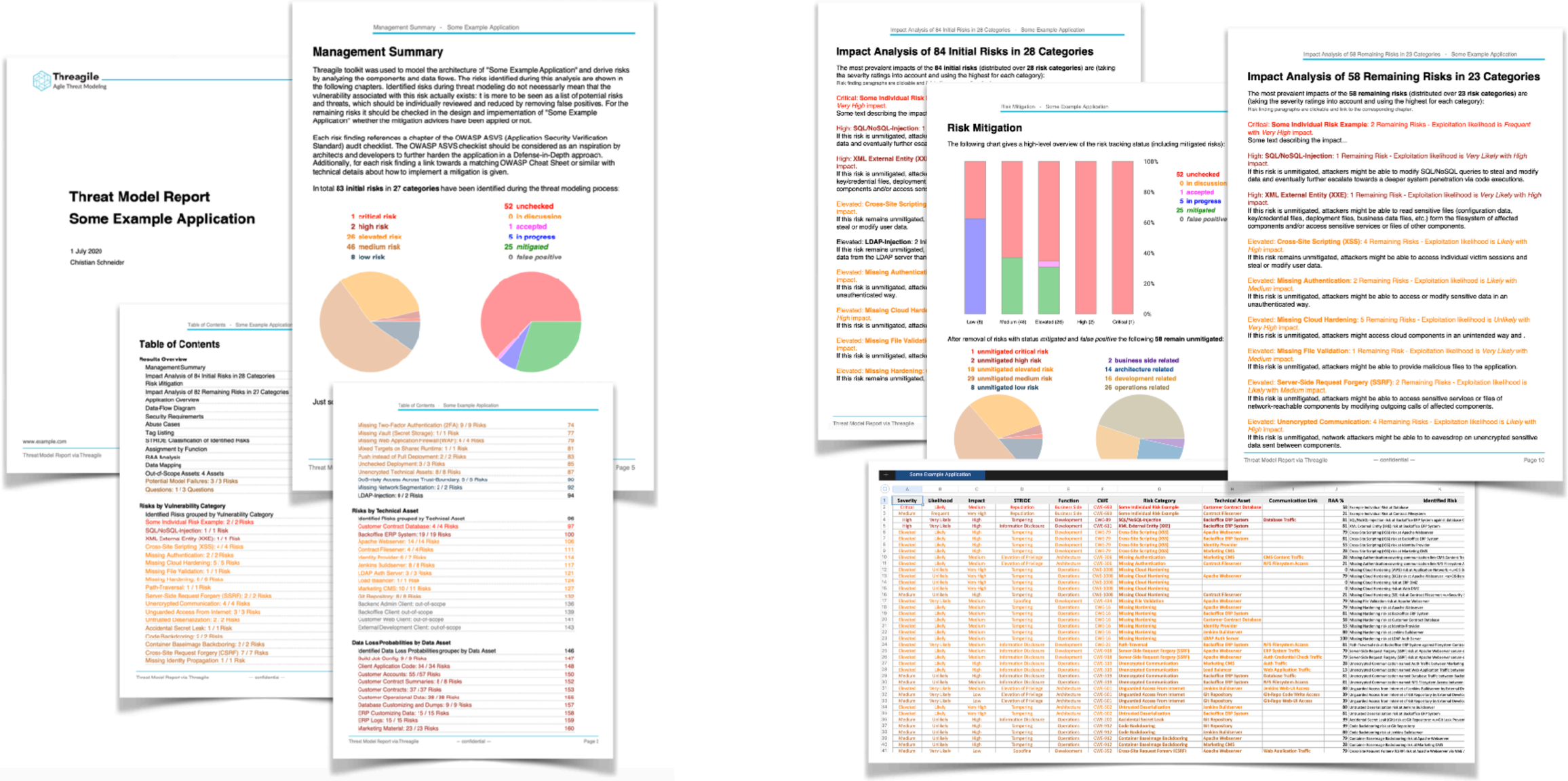
Executes Risk-Rules *(including custom developed ones)*

Creates some nice risk output ;)

# Model Graph Generation (Data Flows)



# PDF & Excel Report Generation



# Impact Summary (before & after mitigation)



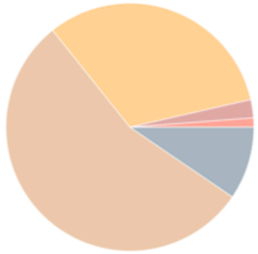
## Management Summary

Threagile toolkit was used to model the architecture of "Some Example Application" and derive risks by analyzing the components and data flows. The risks identified during this analysis are shown in the following chapters. Identified risks during threat modeling do not necessarily mean that the vulnerability associated with this risk actually exists: it is more to be seen as a list of potential risks and threats, which should be individually reviewed and reduced by removing false positives. For the remaining risks it should be checked in the design and implementation of "Some Example Application" whether the mitigation advices have been applied or not.

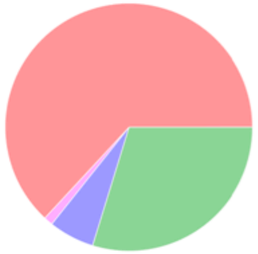
Each risk finding references a chapter of the OWASP ASVS (Application Security Verification Standard) audit checklist. The OWASP ASVS checklist should be considered as an inspiration by architects and developers to further harden the application in a Defense-in-Depth approach. Additionally, for each risk finding a link towards a matching OWASP Cheat Sheet or similar with technical details about how to implement a mitigation is given.

In total **84 initial risks** in **28 categories** have been identified during the threat modeling process:

**1 critical risk**
**2 high risk**
**27 elevated risk**
**46 medium risk**
**8 low risk**



**53 unchecked**
**0 in discussion**
**1 accepted**
**5 in progress**
**25 *mitigated***
**0 *false positive***



Just some **more** custom summary possible here...





## Impact Analysis of 84 Initial Risks in 28 Categories

The most prevalent impacts of the **84 initial risks** (distributed over **28 risk categories**) are (taking the severity ratings into account and using the highest for each category):
Risk finding paragraphs are clickable and link to the corresponding chapter.

Critical: **Some Individual Risk Example**: 2 Initial Risks - Exploitation likelihood is *Frequent* with *Very High* impact.
Some text describing the impact...

High: **SQL/NoSQL-Injection**: 1 Initial Risk - Exploitation likelihood is *Very Likely* with *High* impact.
If this risk is unmitigated, attackers might be able to modify SQL/NoSQL queries to steal and modify data and eventually further escalate towards a deeper system penetration via code executions.

High: **XML External Entity (XXE)**: 1 Initial Risk - Exploitation likelihood is *Very Likely* with *High* impact.
If this risk is unmitigated, attackers might be able to read sensitive files (configuration data, key/credential files, deployment files, business data files, etc.) form the filesystem of affected components and/or access sensitive services or files of other components.

Elevated: **Cross-Site Scripting (XSS)**: 4 Initial Risks - Exploitation likelihood is *Likely* with *High* impact.
If this risk remains unmitigated, attackers might be able to access individual victim sessions and steal or modify user data.

Elevated: **LDAP-Injection**: 2 Initial Risks - Exploitation likelihood is *Likely* with *High* impact.
If this risk remains unmitigated, attackers might be able to modify LDAP queries and access more data from the LDAP server than allowed.

Elevated: **Missing Authentication**: 2 Initial Risks - Exploitation likelihood is *Likely* with *Medium* impact.
If this risk is unmitigated, attackers might be able to access or modify sensitive data in an unauthenticated way.

Elevated: **Missing Cloud Hardening**: 5 Initial Risks - Exploitation likelihood is *Unlikely* with *Very High* impact.
If this risk is unmitigated, attackers might access cloud components in an unintended way and .

Elevated: **Missing File Validation**: 1 Initial Risk - Exploitation likelihood is *Very Likely* with *Medium* impact.
If this risk is unmitigated, attackers might be able to provide malicious files to the application.

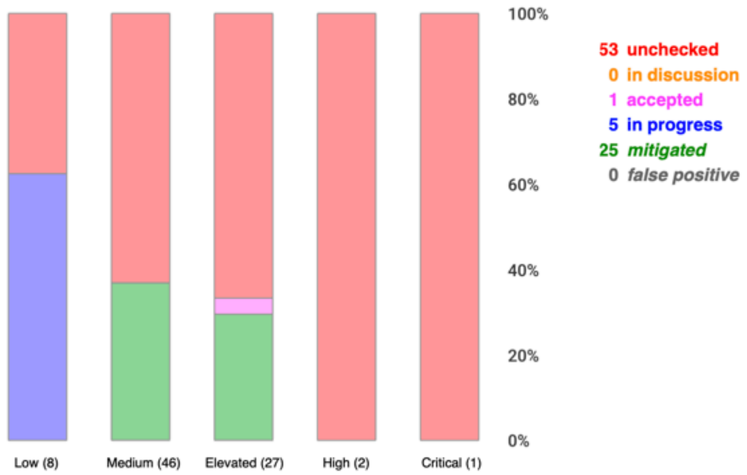
Elevated: **Missing Hardening**: 6 Initial Risks - Exploitation likelihood is *Likely* with *Medium* impact.
If this risk remains unmitigated, attackers might be able to easier attack high-value targets.

# Risk Mitigation



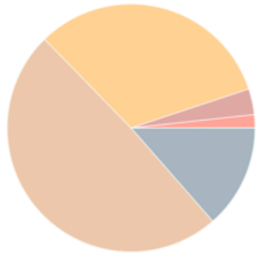
## Risk Mitigation

The following chart gives a high-level overview of the risk tracking status (including mitigated risks):

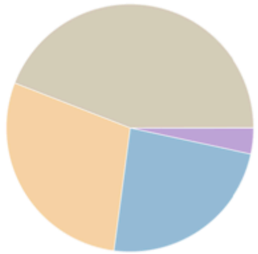



After removal of risks with status *mitigated* and *false positive* the following **59 remain unmitigated**:

1 unmitigated critical risk
2 unmitigated high risk
19 unmitigated elevated risk
29 unmitigated medium risk
8 unmitigated low risk



2 business side related
14 architecture related
17 development related
26 operations related







## Impact Analysis of 59 Remaining Risks in 24 Categories

The most prevalent impacts of the **59 remaining risks** (distributed over **24 risk categories**) are (taking the severity ratings into account and using the highest for each category):
Risk finding paragraphs are clickable and link to the corresponding chapter.

Critical: **Some Individual Risk Example**: 2 Remaining Risks - Exploitation likelihood is *Frequent* with *Very High* impact.
Some text describing the impact...

High: **SQL/NoSQL-Injection**: 1 Remaining Risk - Exploitation likelihood is *Very Likely* with *High* impact.
If this risk is unmitigated, attackers might be able to modify SQL/NoSQL queries to steal and modify data and eventually further escalate towards a deeper system penetration via code executions.

High: **XML External Entity (XXE)**: 1 Remaining Risk - Exploitation likelihood is *Very Likely* with *High* impact.
If this risk is unmitigated, attackers might be able to read sensitive files (configuration data, key/credential files, deployment files, business data files, etc.) form the filesystem of affected components and/or access sensitive services or files of other components.

Elevated: **Cross-Site Scripting (XSS)**: 4 Remaining Risks - Exploitation likelihood is *Likely* with *High* impact.
If this risk remains unmitigated, attackers might be able to access individual victim sessions and steal or modify user data.

Elevated: **Missing Authentication**: 2 Remaining Risks - Exploitation likelihood is *Likely* with *Medium* impact.
If this risk is unmitigated, attackers might be able to access or modify sensitive data in an unauthenticated way.

Elevated: **Missing Cloud Hardening**: 5 Remaining Risks - Exploitation likelihood is *Unlikely* with *Very High* impact.
If this risk is unmitigated, attackers might access cloud components in an unintended way and .

Elevated: **Missing File Validation**: 1 Remaining Risk - Exploitation likelihood is *Very Likely* with *Medium* impact.
If this risk is unmitigated, attackers might be able to provide malicious files to the application.

Elevated: **Path-Traversal**: 1 Remaining Risk - Exploitation likelihood is *Very Likely* with *Medium* impact.
If this risk is unmitigated, attackers might be able to read sensitive files (configuration data, key/credential files, deployment files, business data files, etc.) from the filesystem of affected components.

# STRIDE Classification of Risks



## STRIDE Classification of Identified Risks

This chapter clusters and classifies the risks by STRIDE categories: In total **84 potential risks** have been identified during the threat modeling process of which **8 in the Spoofing** category, **33 in the Tampering** category, **2 in the Repudiation** category, **18 in the Information Disclosure** category, **5 in the Denial of Service** category, and **18 in the Elevation of Privilege** category.

Risk finding paragraphs are clickable and link to the corresponding chapter.

### Spoofing

Elevated: **Missing File Validation**: 1 / 1 Risk - Exploitation likelihood is *Very Likely* with *Medium* impact.
When a technical asset accepts files, these input files should be strictly validated about filename and type.

Medium: **Cross-Site Request Forgery (CSRF)**: 7 / 7 Risks - Exploitation likelihood is *Very Likely* with *Low* impact.
When a web application is accessed via web protocols Cross-Site Request Forgery (CSRF) risks might arise.

### Tampering

High: **SQL/NoSQL-Injection**: 1 / 1 Risk - Exploitation likelihood is *Very Likely* with *High* impact.
When a database is accessed via database access protocols SQL/NoSQL-Injection risks might arise. The risk rating depends on the sensitivity technical asset itself and of the data assets processed or stored.

Elevated: **Cross-Site Scripting (XSS)**: 4 / 4 Risks - Exploitation likelihood is *Likely* with *High* impact.
For each web application Cross-Site Scripting (XSS) risks might arise. In terms of the overall risk level take other applications running on the same domain into account as well.

Elevated: **LDAP-Injection**: 0 / 2 Risks - Exploitation likelihood is *Likely* with *High* impact.
When an LDAP server is accessed LDAP-Injection risks might arise. The risk rating depends on the sensitivity of the LDAP server itself and of the data assets processed or stored.

Elevated: **Missing Cloud Hardening**: 5 / 5 Risks - Exploitation likelihood is *Unlikely* with *Very High* impact.
Cloud components should be hardened according to the cloud vendor best practices. This affects their configuration, auditing, and further areas.

Elevated: **Missing Hardening**: 0 / 6 Risks - Exploitation likelihood is *Likely* with *Medium* impact.
Technical assets with a Relative Attacker Attractiveness (RAA) value of 55 % or higher should be explicitly hardened taking best practices and vendor hardening guides into account.





### Information Disclosure

High: **XML External Entity (XXE)**: 1 / 1 Risk - Exploitation likelihood is *Very Likely* with *High* impact.
When a technical asset accepts data in XML format, XML External Entity (XXE) risks might arise.

Elevated: **Path-Traversal**: 1 / 1 Risk - Exploitation likelihood is *Very Likely* with *Medium* impact.
When a filesystem is accessed Path-Traversal or Local-File-Inclusion (LFI) risks might arise. The risk rating depends on the sensitivity of the technical asset itself and of the data assets processed or stored.

Elevated: **Server-Side Request Forgery (SSRF)**: 2 / 2 Risks - Exploitation likelihood is *Likely* with *Medium* impact.
When a server system (i.e. not a client) is accessing other server systems via typical web protocols Server-Side Request Forgery (SSRF) or Local-File-Inclusion (LFI) or Remote-File-Inclusion (RFI) risks might arise.

Elevated: **Unencrypted Communication**: 4 / 4 Risks - Exploitation likelihood is *Likely* with *High* impact.
Due to the confidentiality and/or integrity rating of the data assets transferred over the communication link this connection must be encrypted.

Medium: **Accidental Secret Leak**: 1 / 1 Risk - Exploitation likelihood is *Unlikely* with *High* impact.
Sourcecode repositories (including their histories) as well as artifact registries can accidentally contain secrets like checked-in or packaged-in passwords, API tokens, certificates, crypto keys, etc.

Medium: **Missing Vault (Secret Storage)**: 1 / 1 Risk - Exploitation likelihood is *Unlikely* with *Medium* impact.
In order to avoid the risk of secret leakage via config files (when attacked through vulnerabilities being able to read files like Path-Traversal and others), it is best practice to use a separate hardened process with proper authentication, authorization, and audit logging to access config secrets (like credentials, private keys, client certificates, etc.). This component is usually some kind of Vault.

Medium: **Unencrypted Technical Assets**: 0 / 8 Risks - Exploitation likelihood is *Unlikely* with *High* impact.
Due to the confidentiality rating of the technical asset itself and/or the processed data assets this technical asset must be encrypted. The risk rating depends on the sensitivity technical asset itself and of the data assets stored.

### Denial of Service

Low: **DoS-risky Access Across Trust-Boundary**: 5 / 5 Risks - Exploitation likelihood is *Unlikely* with *Low* impact.
Assets accessed across trust boundaries with critical or mission-critical availability rating are more prone to Denial-of-Service (DoS) risks.

# Assignment by Function



## Assignment by Function

This chapter clusters and assigns the risks by functions which are most likely able to ch
mitigate them: In total **84 potential risks** have been identified during the threat modelin
which **11 should be checked by Business Side**, **14 should be checked by Architect**
**should be checked by Development**, and **40 should be checked by Operations**.
Risk finding paragraphs are clickable and link to the corresponding chapter.

### Business Side

Critical: **Some Individual Risk Example**: 2 / 2 Risks - Exploitation likelihood is *Freq*
*Very High* impact.
Some text describing the mitigation...

Medium: **Missing Two-Factor Authentication (2FA)**: 0 / 9 Risks - Exploitation likelih
*Unlikely* with *Medium* impact.
Apply an authentication method to the technical asset protecting highly sensitive data
two-factor authentication for human users.

### Architecture

Elevated: **Missing Authentication**: 2 / 2 Risks - Exploitation likelihood is *Likely* with
impact.
Apply an authentication method to the technical asset. To protect highly sensitive dat
the use of two-factor authentication for human users.

Elevated: **Unguarded Access From Internet**: 3 / 3 Risks - Exploitation likelihood is
with *Medium* impact.
Encapsulate the asset behind a guarding service, application, or reverse-proxy. For a
maintenance a bastion-host should be used as a jump-server. For file transfer a
store-and-forward-host should be used as an indirect file exchange platform.

Elevated: **Untrusted Deserialization**: 2 / 2 Risks - Exploitation likelihood is *Likely* wi
impact.
Try to avoid the deserialization of untrusted data (even of data within the same trust-
long as it is sent across a remote connection) in order to stay safe from Untrusted De
vulnerabilities. Alternatively a strict whitelisting approach of the classes/types/values
deserialize might help as well. When a third-party product is used instead of custom
software, check if the product applies the proper mitigation and ensure a reasonable

Medium: **Missing Identity Propagation**: 1 / 1 Risk - Exploitation likelihood is *Unlikel*
*Medium* impact.
When processing requests for endusers if possible authorize in the backend against
propagated identity of the enduser. This can be achieved in passing JWTs or similar
checking them in the backend services. For DevOps usages apply at least a technical-user
authorization.





Medium: **Missing Vault (Secret Storage)**: 1 / 1 Risk - Exploitation likelihood is *Unlikely*
*Medium* impact.
Consider using a Vault (Secret Storage) to securely store and access config secrets (lik
credentials, private keys, client certificates, etc.).

Medium: **Push instead of Pull Deployment**: 2 / 2 Risks - Exploitation likelihood is *Unli*
*Medium* impact.
Try to prefer pull-based deployments (like GitOps scenarios offer) over push-based dep

Medium: **Unchecked Deployment**: 3 / 3 Risks - Exploitation likelihood is *Unlikely* with
impact.
Apply DevSecOps best-practices and use scanning tools to identify vulnerabilities in so
byte-code,dependencies, container layers, and optionally also via dynamic scans again
test systems.

### Development

High: **SQL/NoSQL-Injection**: 1 / 1 Risk - Exploitation likelihood is *Very Likely* with *High*
Try to use parameter binding to be safe from injection vulnerabilities. When a third-party
is used instead of custom developed software, check if the product applies the proper m
and ensure a reasonable patch-level.

High: **XML External Entity (XXE)**: 1 / 1 Risk - Exploitation likelihood is *Very Likely* with
impact.
Apply hardening of all XML parser instances in order to stay safe from XML External En
vulnerabilities. When a third-party product is used instead of custom developed softwar
the product applies the proper mitigation and ensure a reasonable patch-level.

Elevated: **Cross-Site Scripting (XSS)**: 4 / 4 Risks - Exploitation likelihood is *Likely* with
impact.
Try to encode all values sent back to the browser and also handle DOM-manipulations
way to avoid DOM-based XSS. When a third-party product is used instead of custom de
software, check if the product applies the proper mitigation and ensure a reasonable pa

Elevated: **LDAP-Injection**: 0 / 2 Risks - Exploitation likelihood is *Likely* with *High* impac
Try to use libraries that properly encode LDAP meta characters in searches and queries
access the LDAP sever in order to stay safe from LDAP-Injection vulnerabilities. When
third-party product is used instead of custom developed software, check if the product a
proper mitigation and ensure a reasonable patch-level.

Elevated: **Missing File Validation**: 1 / 1 Risk - Exploitation likelihood is *Very Likely* with
impact.
Filter by file extension and discard (if feasible) the name provided. Whitelist the accepte
types and determine the mime-type on the server-side (for example via "Apache Tika" o
checks). If the file is retrievable by endusers and/or backoffice employees, consider per
scans for popular malware (if the files can be retrieved much later than they were uploa
apply a fresh malware scan during retrieval to scan with newer signatures of popular ma





Also enforce limits on maximum file size to avoid denial-of-service like scenarios.

Elevated: **Path-Traversal**: 1 / 1 Risk - Exploitation likelihood is *Very Likely* with *Medium* impact.
Before accessing the file cross-check that it resides in the expected folder and is of the expected type and filename/suffix. Try to use a mapping if possible instead of directly accessing by a filename which is (partly or fully) provided by the caller. When a third-party product is used instead of custom developed software, check if the product applies the proper mitigation and ensure a reasonable patch-level.

Elevated: **Server-Side Request Forgery (SSRF)**: 2 / 2 Risks - Exploitation likelihood is *Likely* with *Medium* impact.
Try to avoid constructing the outgoing target URL with caller controllable values. Alternatively use a mapping (whitelist) when accessing outgoing URLs instead of creating them including caller controllable values. When a third-party product is used instead of custom developed software, check if the product applies the proper mitigation and ensure a reasonable patch-level.

Medium: **Cross-Site Request Forgery (CSRF)**: 7 / 7 Risks - Exploitation likelihood is *Very Likely* with *Low* impact.
Try to use anti-CSRF tokens or the double-submit patterns (at least for logged-in requests). When your authentication scheme depends on cookies (like session or token cookies), consider marking them with the same-site flag. When a third-party product is used instead of custom developed software, check if the product applies the proper mitigation and ensure a reasonable patch-level.

### Operations

Elevated: **Missing Cloud Hardening**: 5 / 5 Risks - Exploitation likelihood is *Unlikely* with *Very High* impact.
Apply hardening of all cloud components and services, taking special care to follow the individual risk descriptions (which depend on the cloud provider tags in the model).

Elevated: **Missing Hardening**: 0 / 6 Risks - Exploitation likelihood is *Likely* with *Medium* impact.
Try to apply all hardening best practices (like CIS benchmarks, OWASP recommendations, vendor recommendations, DevSec Hardening Framework, DBSAT for Oracle databases, and others).

Elevated: **Unencrypted Communication**: 4 / 4 Risks - Exploitation likelihood is *Likely* with *High* impact.
Apply transport layer encryption to the communication link.

Medium: **Accidental Secret Leak**: 1 / 1 Risk - Exploitation likelihood is *Unlikely* with *High* impact.
Establish measures preventing accidental check-in or package-in of secrets into sourcecode repositories and artifact registries. This starts by using good .gitignore and .dockerignore files, but does not stop there. See for example tools like *"git-secrets" or "Talisman"* to have check-in preventive measures for secrets. Consider also to regularly scan your repositories for secrets accidentally checked-in using scanning tools like *"gitleaks" or "gitrob"*.

# Relative Attacker Attractiveness (RAA)

## RAA Analysis

For each technical asset the **"Relative Attacker Attractiveness"** (RAA) value was calculated in percent. The higher the RAA, the more interesting it is for an attacker to compromise the asset. The calculation algorithm takes the sensitivity ratings and quantities of stored and processed data into account as well as the communication links of the technical asset. Neighbouring assets to high-value RAA targets might receive an increase in their RAA value when they have a communication link towards that target ("Pivoting-Factor").

The following lists all technical assets sorted by their RAA value from highest (most attacker attractive) to lowest. This list can be used to prioritize on efforts relevant for the most attacker-attractive technical assets:

Technical asset paragraphs are clickable and link to the corresponding chapter.

**LDAP Auth Server**: RAA 100%
LDAP authentication server

**Backoffice ERP System**: RAA 81%
ERP system

**Jenkins Buildserver**: RAA 80%
Jenkins buildserver

**Apache Webserver**: RAA 75%
Apache Webserver

**Customer Contract Database**: RAA 58%
The database behind the ERP system

**Identity Provider**: RAA 53%
Identity provider server

**Git Repository**: RAA 39%
Git repository server

**Marketing CMS**: RAA 28%
CMS for the marketing content

**Contract Fileserver**: RAA 21%
NFS Filesystem for storing the contract PDFs

**Load Balancer**: RAA 13%
Load Balancer (HA-Proxy)

Sensitivity rating of stored & processed data

Attacker paths to the highest-valued targets:
Components with access to these are ranked higher also

Nice example: Build-Pipelines with many deployment connections…

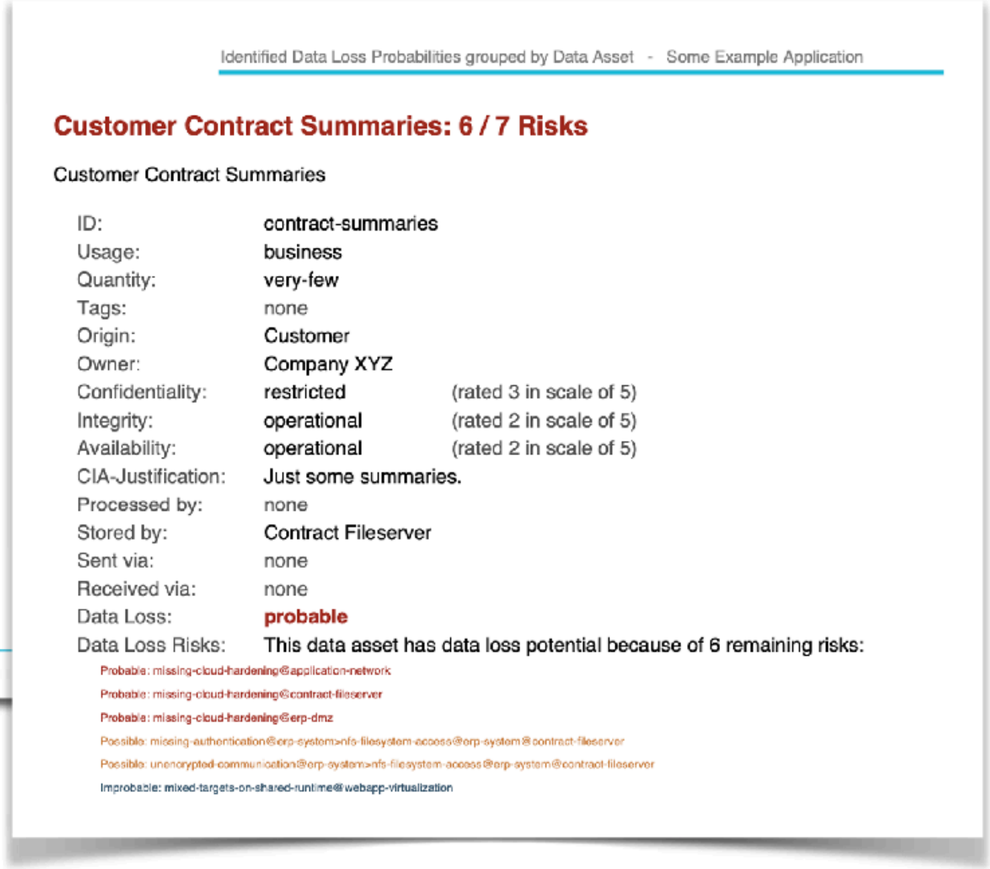
Reflected in the created data flow diagram

*Custom calculation algorithms possible as plugins*

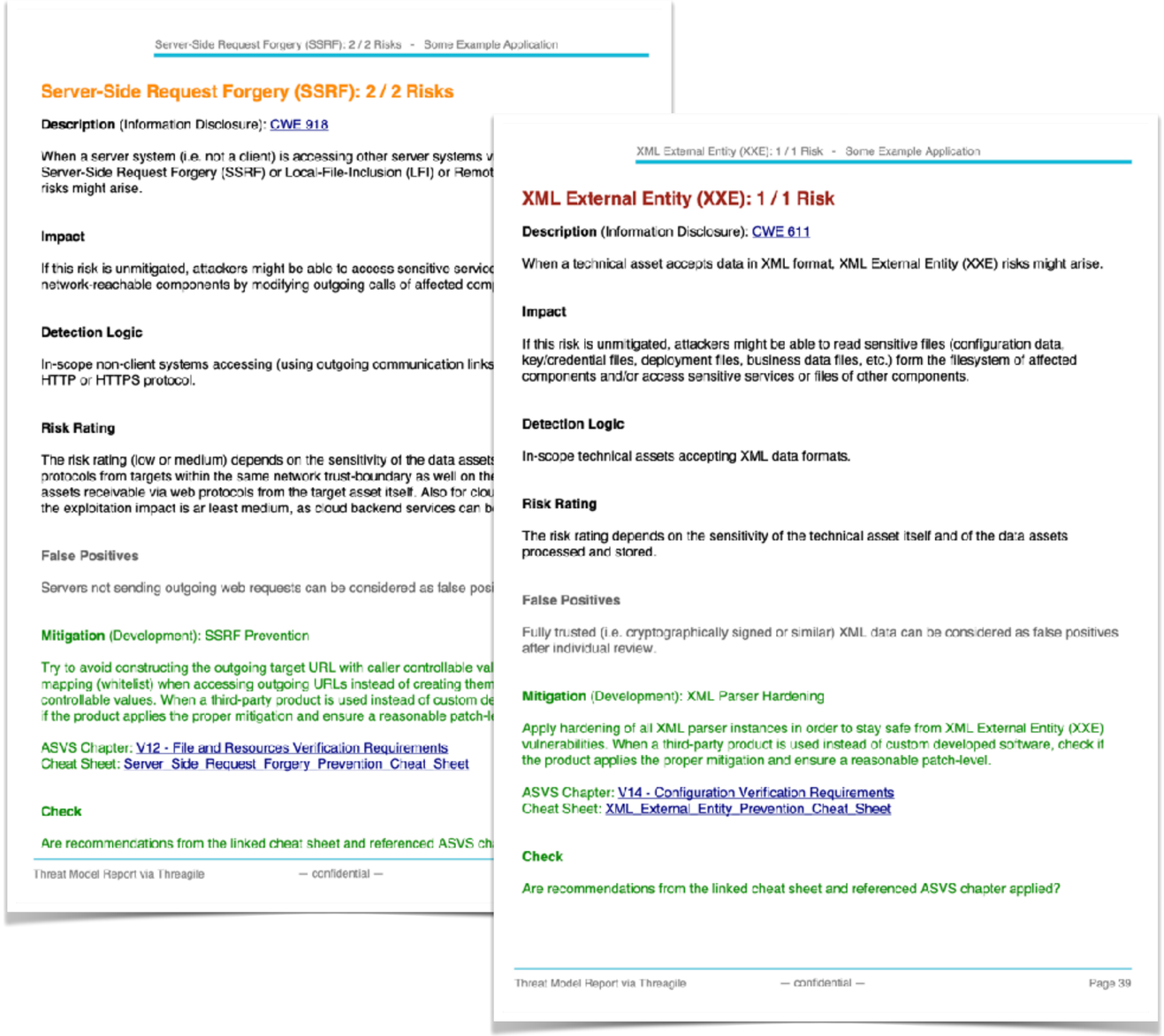
# Data Breach Probabilities (DBP)

“Blast-Impact” of compromised systems

Each Risk-Rule refers to affected targets:
And the data assets stored/processed there

# Risk Mitigation Recommendations



Server-Side Request Forgery (SSRF): 2 / 2 Risks - Some Example Application

## Server-Side Request Forgery (SSRF): 2 / 2 Risks

**Description** (Information Disclosure): CWE 918

When a server system (i.e. not a client) is accessing other server systems v
Server-Side Request Forgery (SSRF) or Local-File-Inclusion (LFI) or Remot
risks might arise.

**Impact**

If this risk is unmitigated, attackers might be able to access sensitive servic
network-reachable components by modifying outgoing calls of affected com

**Detection Logic**

In-scope non-client systems accessing (using outgoing communication links
HTTP or HTTPS protocol.

**Risk Rating**

The risk rating (low or medium) depends on the sensitivity of the data asset
protocols from targets within the same network trust-boundary as well on th
assets receivable via web protocols from the target asset itself. Also for clou
the exploitation impact is at least medium, as cloud backend services can b

**False Positives**

Servers not sending outgoing web requests can be considered as false posi

**Mitigation** (Development): SSRF Prevention

Try to avoid constructing the outgoing target URL with caller controllable val
mapping (whitelist) when accessing outgoing URLs instead of creating them
controllable values. When a third-party product is used instead of custom de
if the product applies the proper mitigation and ensure a reasonable patch-l

ASVS Chapter: V12 - File and Resources Verification Requirements
Cheat Sheet: Server_Side_Request_Forgery_Prevention_Cheat_Sheet

**Check**

Are recommendations from the linked cheat sheet and referenced ASVS ch

Threat Model Report via Threagile — confidential —

XML External Entity (XXE): 1 / 1 Risk - Some Example Application

## XML External Entity (XXE): 1 / 1 Risk

**Description** (Information Disclosure): CWE 611

When a technical asset accepts data in XML format, XML External Entity (XXE) risks might arise.

**Impact**

If this risk is unmitigated, attackers might be able to read sensitive files (configuration data, key/credential files, deployment files, business data files, etc.) form the filesystem of affected components and/or access sensitive services or files of other components.

**Detection Logic**

In-scope technical assets accepting XML data formats.

**Risk Rating**

The risk rating depends on the sensitivity of the technical asset itself and of the data assets processed and stored.

**False Positives**

Fully trusted (i.e. cryptographically signed or similar) XML data can be considered as false positives after individual review.

**Mitigation** (Development): XML Parser Hardening

Apply hardening of all XML parser instances in order to stay safe from XML External Entity (XXE) vulnerabilities. When a third-party product is used instead of custom developed software, check if the product applies the proper mitigation and ensure a reasonable patch-level.

ASVS Chapter: V14 - Configuration Verification Requirements
Cheat Sheet: XML_External_Entity_Prevention_Cheat_Sheet

**Check**

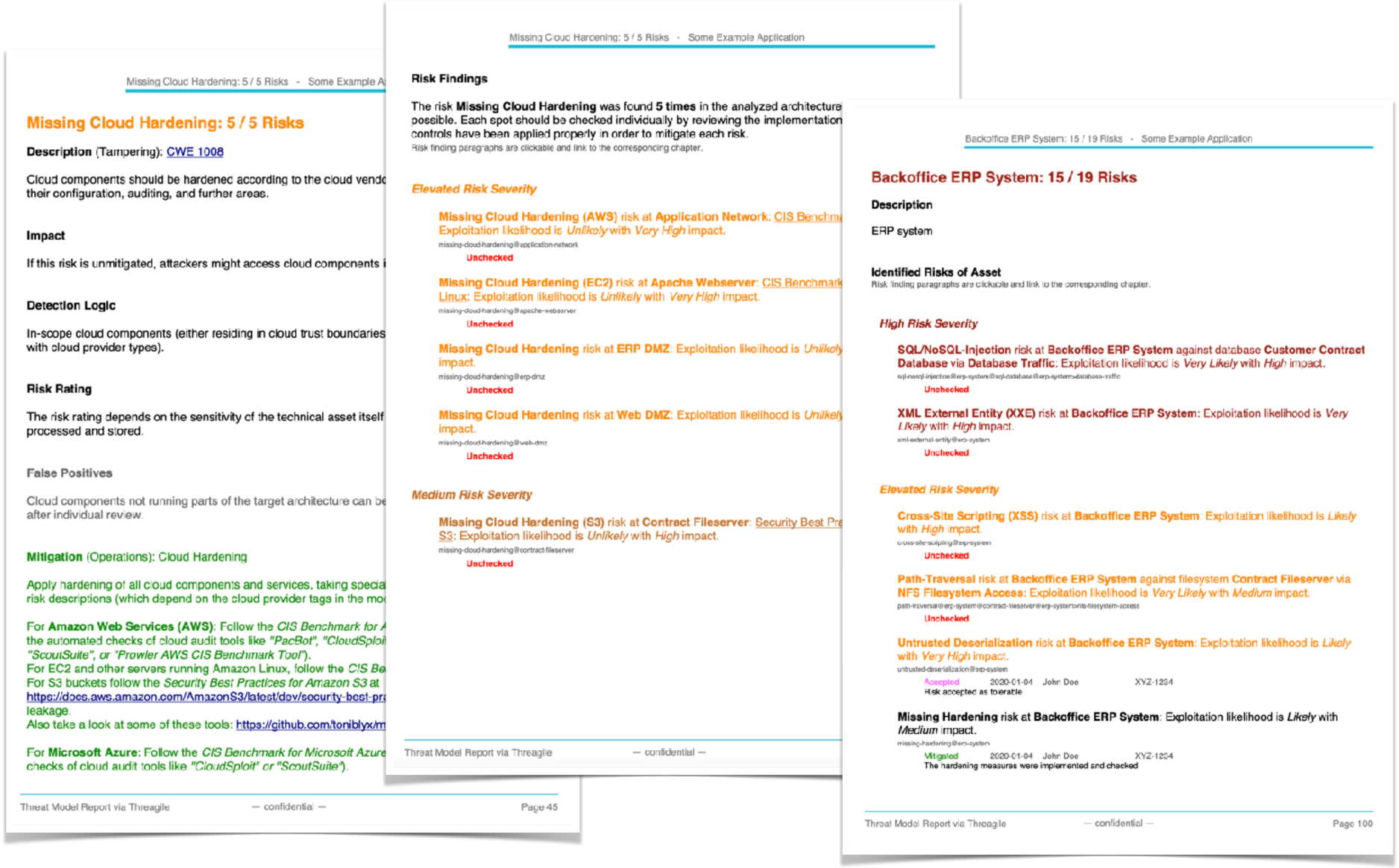
Are recommendations from the linked cheat sheet and referenced ASVS chapter applied?

Threat Model Report via Threagile — confidential — Page 39

Detailed mitigations along with links to

- OWASP ASVS Chapter
- OWASP CSVS Chapter
- OWASP Cheat Sheet
- *etc.*

# **Risk Instances** (by vulnerability & by tech asset)



Missing Cloud Hardening: 5 / 5 Risks - Some Example A

## Missing Cloud Hardening: 5 / 5 Risks

**Description** (Tampering): CWE 1008

Cloud components should be hardened according to the cloud vendo their configuration, auditing, and further areas.

**Impact**

If this risk is unmitigated, attackers might access cloud components i

**Detection Logic**

In-scope cloud components (either residing in cloud trust boundaries with cloud provider types).

**Risk Rating**

The risk rating depends on the sensitivity of the technical asset itself processed and stored.

**False Positives**

Cloud components not running parts of the target architecture can be after individual review.

**Mitigation** (Operations): Cloud Hardening

Apply hardening of all cloud components and services, taking specia risk descriptions (which depend on the cloud provider tags in the mo

For **Amazon Web Services (AWS)**: Follow the *CIS Benchmark for A* the automated checks of cloud audit tools like *"PacBot"*, *"CloudSploi "ScoutSuite"*, or *"Prowler AWS CIS Benchmark Tool"*).
For EC2 and other servers running Amazon Linux, follow the *CIS Be*
For S3 buckets follow the *Security Best Practices for Amazon S3* at https://docs.aws.amazon.com/AmazonS3/latest/dev/security-best-pr leakage.
Also take a look at some of these tools: https://github.com/toniblyx/m

For **Microsoft Azure**: Follow the *CIS Benchmark for Microsoft Azure* checks of cloud audit tools like *"CloudSploit"* or *"ScoutSuite"*).

Threat Model Report via Threagile — confidential — Page 45

Missing Cloud Hardening: 5 / 5 Risks - Some Example Application

**Risk Findings**

The risk **Missing Cloud Hardening** was found **5 times** in the analyzed architecture possible. Each spot should be checked individually by reviewing the implementation controls have been applied properly in order to mitigate each risk.
Risk finding paragraphs are clickable and link to the corresponding chapter.

***Elevated Risk Severity***

**Missing Cloud Hardening (AWS)** risk at **Application Network**: CIS Benchma Exploitation likelihood is *Unlikely* with *Very High* impact.
missing-cloud-hardening@application-network
**Unchecked**

**Missing Cloud Hardening (EC2)** risk at **Apache Webserver**: CIS Benchmark Linux: Exploitation likelihood is *Unlikely* with *Very High* impact.
missing-cloud-hardening@apache-webserver
**Unchecked**

**Missing Cloud Hardening** risk at **ERP DMZ**: Exploitation likelihood is *Unlikely* impact.
missing-cloud-hardening@erp-dmz
**Unchecked**

**Missing Cloud Hardening** risk at **Web DMZ**: Exploitation likelihood is *Unlikely* impact.
missing-cloud-hardening@web-dmz
**Unchecked**

***Medium Risk Severity***

**Missing Cloud Hardening (S3)** risk at **Contract Fileserver**: Security Best Pre S3: Exploitation likelihood is *Unlikely* with *High* impact.
missing-cloud-hardening@contract-fileserver
**Unchecked**

Threat Model Report via Threagile — confidential —

Backoffice ERP System: 15 / 19 Risks - Some Example Application

## Backoffice ERP System: 15 / 19 Risks

**Description**

ERP system

**Identified Risks of Asset**
Risk finding paragraphs are clickable and link to the corresponding chapter.

***High Risk Severity***

**SQL/NoSQL-Injection** risk at **Backoffice ERP System** against database **Customer Contract Database** via **Database Traffic**: Exploitation likelihood is *Very Likely* with *High* impact.
sql-nosql-injection@erp-system@sql-database@erp-system>database-traffic
**Unchecked**

**XML External Entity (XXE)** risk at **Backoffice ERP System**: Exploitation likelihood is *Very Likely* with *High* impact.
xml-external-entity@erp-system
**Unchecked**

***Elevated Risk Severity***

**Cross-Site Scripting (XSS)** risk at **Backoffice ERP System**: Exploitation likelihood is *Likely* with *High* impact.
cross-site-scripting@erp-system
**Unchecked**

**Path-Traversal** risk at **Backoffice ERP System** against filesystem **Contract Fileserver** via **NFS Filesystem Access**: Exploitation likelihood is *Very Likely* with *Medium* impact.
path-traversal@erp-system@contract-fileserver@erp-system>nfs-filesystem-access
**Unchecked**

**Untrusted Deserialization** risk at **Backoffice ERP System**: Exploitation likelihood is *Likely* with *Very High* impact.
untrusted-deserialization@erp-system
Accepted 2020-01-04 John Doe XYZ-1234
Risk accepted as tolerable

**Missing Hardening** risk at **Backoffice ERP System**: Exploitation likelihood is *Likely* with *Medium* impact.
missing-hardening@erp-system
Mitigated 2020-01-04 John Doe XYZ-1234
The hardening measures were implemented and checked

Threat Model Report via Threagile — confidential — Page 100

*Everything linked and clickable inside the report for easy navigation*

# Excel Report

Some Example Application

| Severity | Likelihood | Impact | STRIDE | Function | CWE | Risk Category | Technical Asset | Communication Link | RAA % | Identified Risk |
|---|---|---|---|---|---|---|---|---|---|---|
| Critical | Likely | Medium | Repudiation | Business Side | CWE-693 | Some Individual Risk Example | Customer Contract Database | | 58 | Example Individual Risk at Database |
| Medium | Frequent | Very High | Repudiation | Business Side | CWE-693 | Some Individual Risk Example | Contract Fileserver | | 21 | Example Individual Risk at Contract Filesystem |
| High | Very Likely | High | Tampering | Development | CWE-89 | SQL/NoSQL-Injection | Backoffice ERP System | Database Traffic | 81 | SQL/NoSQL-Injection risk at Backoffice ERP System against database C |
| High | Very Likely | High | Information Disclosure | Development | CWE-611 | XML External Entity (XXE) | Backoffice ERP System | | 81 | XML External Entity (XXE) risk at Backoffice ERP System |
| Elevated | Likely | High | Tampering | Development | CWE-79 | Cross-Site Scripting (XSS) | Apache Webserver | | 79 | Cross-Site Scripting (XSS) risk at Apache Webserver |
| Elevated | Likely | High | Tampering | Development | CWE-79 | Cross-Site Scripting (XSS) | Backoffice ERP System | | 81 | Cross-Site Scripting (XSS) risk at Backoffice ERP System |
| Elevated | Likely | High | Tampering | Development | CWE-79 | Cross-Site Scripting (XSS) | Identity Provider | | 53 | Cross-Site Scripting (XSS) risk at Identity Provider |
| Elevated | Likely | High | Tampering | Development | CWE-79 | Cross-Site Scripting (XSS) | Marketing CMS | | 28 | Cross-Site Scripting (XSS) risk at Marketing CMS |
| Elevated | Likely | Medium | Elevation of Privilege | Architecture | CWE-306 | Missing Authentication | Marketing CMS | CMS Content Traffic | 28 | Missing Authentication covering communication link CMS Content Traf |
| Elevated | Likely | Medium | Elevation of Privilege | Architecture | CWE-306 | Missing Authentication | Contract Fileserver | NFS Filesystem Access | 21 | Missing Authentication covering communication link NFS Filesystem A |
| Elevated | Unlikely | Very High | Tampering | Operations | CWE-1008 | Missing Cloud Hardening | | | 0 | Missing Cloud Hardening (AWS) risk at Application Network: <u>CIS Be |
| Elevated | Unlikely | Very High | Tampering | Operations | CWE-1008 | Missing Cloud Hardening | Apache Webserver | | 79 | Missing Cloud Hardening (EC2) risk at Apache Webserver: <u>CIS Benc |
| Elevated | Unlikely | Very High | Tampering | Operations | CWE-1008 | Missing Cloud Hardening | | | 0 | Missing Cloud Hardening risk at ERP DMZ |
| Elevated | Unlikely | Very High | Tampering | Operations | CWE-1008 | Missing Cloud Hardening | | | 0 | Missing Cloud Hardening risk at Web DMZ |
| Medium | Unlikely | High | Tampering | Operations | CWE-1008 | Missing Cloud Hardening | Contract Fileserver | | 21 | Missing Cloud Hardening (S3) risk at Contract Fileserver: <u>Security B |
| Elevated | Very Likely | Medium | Spoofing | Development | CWE-434 | Missing File Validation | Apache Webserver | | 79 | Missing File Validation risk at Apache Webserver |
| Elevated | Likely | Medium | Tampering | Operations | CWE-16 | Missing Hardening | Apache Webserver | | 79 | Missing Hardening risk at Apache Webserver |
| Elevated | Likely | Medium | Tampering | Operations | CWE-16 | Missing Hardening | Backoffice ERP System | | 81 | Missing Hardening risk at Backoffice ERP System |
| Elevated | Likely | Medium | Tampering | Operations | CWE-16 | Missing Hardening | Customer Contract Database | | 58 | Missing Hardening risk at Customer Contract Database |
| Elevated | Likely | Medium | Tampering | Operations | CWE-16 | Missing Hardening | Identity Provider | | 53 | Missing Hardening risk at Identity Provider |
| Elevated | Likely | Medium | Tampering | Operations | CWE-16 | Missing Hardening | Jenkins Buildserver | | 80 | Missing Hardening risk at Jenkins Buildserver |
| Elevated | Likely | Medium | Tampering | Operations | CWE-16 | Missing Hardening | LDAP Auth Server | | 100 | Missing Hardening risk at LDAP Auth Server |
| Elevated | Very Likely | Medium | Information Disclosure | Development | CWE-22 | Path-Traversal | Backoffice ERP System | NFS Filesystem Access | 81 | Path-Traversal risk at Backoffice ERP System against filesystem Contract |
| Elevated | Likely | Medium | Information Disclosure | Development | CWE-918 | Server-Side Request Forgery (SSRF) | Apache Webserver | ERP System Traffic | 79 | Server-Side Request Forgery (SSRF) risk at Apache Webserver server-si |
| Elevated | Likely | Medium | Information Disclosure | Development | CWE-918 | Server-Side Request Forgery (SSRF) | Apache Webserver | Auth Credential Check Traffic | 79 | Server-Side Request Forgery (SSRF) risk at Apache Webserver server-si |
| Elevated | Likely | High | Information Disclosure | Operations | CWE-319 | Unencrypted Communication | Marketing CMS | Auth Traffic | 28 | Unencrypted Communication named Auth Traffic between Marketing |
| Elevated | Likely | High | Information Disclosure | Operations | CWE-319 | Unencrypted Communication | Load Balancer | Web Application Traffic | 13 | Unencrypted Communication named Web Application Traffic between |
| Medium | Unlikely | High | Information Disclosure | Operations | CWE-319 | Unencrypted Communication | Backoffice ERP System | Database Traffic | 81 | Unencrypted Communication named Database Traffic between Backoff |
| Medium | Unlikely | Medium | Information Disclosure | Operations | CWE-319 | Unencrypted Communication | Backoffice ERP System | NFS Filesystem Access | 81 | Unencrypted Communication named NFS Filesystem Access between B |
| Elevated | Very Likely | Medium | Elevation of Privilege | Architecture | CWE-501 | Unguarded Access From Internet | Jenkins Buildserver | Jenkins Web-UI Access | 80 | Unguarded Access from Internet of Jenkins Buildserver by External Dev |
| Medium | Very Likely | Low | Elevation of Privilege | Architecture | CWE-501 | Unguarded Access From Internet | Git Repository | Git-Repo Code Write Access | 39 | Unguarded Access from Internet of Git Repository by External Develop |
| Medium | Very Likely | Low | Elevation of Privilege | Architecture | CWE-501 | Unguarded Access From Internet | Git Repository | Git-Repo Web-UI Access | 39 | Unguarded Access from Internet of Git Repository by External Develop |
| Elevated | Likely | Very High | Tampering | Architecture | CWE-502 | Untrusted Deserialization | Jenkins Buildserver | | 80 | Untrusted Deserialization risk at Jenkins Buildserver |
| Elevated | Likely | Very High | Tampering | Architecture | CWE-502 | Untrusted Deserialization | Backoffice ERP System | | 81 | Untrusted Deserialization risk at Backoffice ERP System |
| Medium | Unlikely | High | Information Disclosure | Operations | CWE-200 | Accidental Secret Leak | Git Repository | | 39 | Accidental Secret Leak (Git) risk at Git Repository: <u>Git Leak Preventi |
| Medium | Unlikely | High | Tampering | Operations | CWE-912 | Code Backdooring | Git Repository | | 39 | Code Backdooring risk at Git Repository |
| Medium | Unlikely | High | Tampering | Operations | CWE-912 | Code Backdooring | Jenkins Buildserver | | 80 | Code Backdooring risk at Jenkins Buildserver |
| Medium | Unlikely | High | Tampering | Operations | CWE-912 | Container Baseimage Backdooring | Apache Webserver | | 79 | Container Baseimage Backdooring risk at Apache Webserver |
| Medium | Unlikely | High | Tampering | Operations | CWE-912 | Container Baseimage Backdooring | Marketing CMS | | 28 | Container Baseimage Backdooring risk at Marketing CMS |
| Medium | Very Likely | Low | Spoofing | Development | CWE-352 | Cross-Site Request Forgery (CSRF) | Apache Webserver | Web Application Traffic | 79 | Cross-Site Request Forgery (CSRF) risk at Apache Webserver via Web A |

# Detail Results as JSON


```
{
  "category": "container-baseimage-backdooring",
  "risk_status": "unchecked",
  "severity": "medium",
  "exploitation_likelihood": "unlikely",
  "exploitation_impact": "high",
  "title": "\u003cb\u003eContainer Baseimage Backdooring\u003c/b\u003e risk at \u003cb\u003eApache Webserver\u003c/b\u003e",
  "synthetic_id": "container-baseimage-backdooring@apache-webserver",
  "most_relevant_data_asset": "",
  "most_relevant_technical_asset": "apache-webserver",
  "most_relevant_trust_boundary": "",
  "most_relevant_shared_runtime": "",
  "most_relevant_communication_link": "",
  "data_loss_probability": "probable",
  "data_loss_technical_assets": [
    "apache-webserver"
  ]
},
{
  "category": "container-baseimage-backdooring",
  "risk_status": "unchecked",
  "severity": "medium",
  "exploitation_likelihood": "unlikely",
  "exploitation_impact": "high",
  "title": "\u003cb\u003eContainer Baseimage Backdooring\u003c/b\u003e risk at \u003cb\u003eMarketing CMS\u003c/b\u003e",
  "synthetic_id": "container-baseimage-backdooring@marketing-cms",
  "most_relevant_data_asset": "",
  "most_relevant_technical_asset": "marketing-cms",
  "most_relevant_trust_boundary": "",
  "most_relevant_shared_runtime": "",
  "most_relevant_communication_link": "",
  "data_loss_probability": "probable",
  "data_loss_technical_assets": [
    "marketing-cms"
  ]
},
```

# Risk Rules (~40 and constantly growing)

- risks
  - built-in
    - accidental-secret-leak
    - code-backdooring
    - container-baseimage-backdooring
    - container-platform-escape
    - cross-site-request-forgery
    - cross-site-scripting
    - dos-risky-access-across-trust-boundary
    - incomplete-model
    - ldap-injection
    - missing-authentication
    - missing-authentication-second-factor
    - missing-build-infrastructure
    - missing-cloud-hardening
    - missing-file-validation
    - missing-hardening
    - missing-identity-propagation
    - missing-identity-provider-isolation
    - missing-identity-store
    - missing-network-segmentation
    - missing-vault
    - missing-vault
    - missing-vault-isolation
    - missing-waf
    - mixed-targets-on-shared-runtime
    - path-traversal
    - push-instead-of-pull-deployment
    - search-query-injection
    - server-side-request-forgery
    - service-registry-poisoning
    - sql-nosql-injection
    - unchecked-deployment
    - unencrypted-asset
    - unencrypted-communication
    - unguarded-access-from-internet
    - unguarded-direct-datastore-access
    - unnecessary-communication-link
    - unnecessary-data-asset
    - unnecessary-data-transfer
    - unnecessary-technical-asset
    - untrusted-deserialization
    - wrong-communication-link-content
    - wrong-trust-boundary-content
    - xml-external-entity
  - custom

# Custom Risk Rules (plugin interface)

```go
package ldap_injection

import ...

func Category() model.RiskCategory {
    return model.RiskCategory{
        Id:    "ldap-injection",
        Title: "LDAP-Injection",
        Description: "When an LDAP server is accessed LDAP-Injection risks might arise. " +
            "The risk rating depends on the sensitivi
        Impact:     "If this risk remains unmitigated
        ASVS:       "V5 - Validation, Sanitization ar
        CheatSheet: "https://cheatsheetseries.owasp.c
        Action:     "LDAP-Injection Prevention",
        Mitigation: "Try to use libraries that proper
            "the LDAP sever in order to stay safe fro
            "When a third-party product is used inste
        Check:          "Are recommendations from the
        Function:       model.Development,
        STRIDE:         model.Tampering,
        DetectionLogic: "In-scope clients accessing L
        RiskAssessment: "The risk rating depends on t
        FalsePositives: "LDAP server queries by searc
            "as false positives after individual revi
        ModelFailurePossibleReason: false,
        CWE:                        90,
    }
}
```

```go
func GenerateRisks() []model.Risk {
    risks := make([]model.Risk, 0)
    for _, technicalAsset := range model.ParsedModelRoot.TechnicalAssets {
        incomingFlows := model.IncomingTechnicalCommunicationLinksMappedByTargetId[technical
        for _, incomingFlow := range incomingFlows {
            if model.ParsedModelRoot.TechnicalAssets[incomingFlow.SourceId].OutOfScope {
                continue
            }
            if incomingFlow.Protocol == model.LDAP || incomingFlow.Protocol == model.LDAPS {
                likelihood := model.Likely
                if incomingFlow.Usage == model.DevOps {
                    likelihood = model.Unlikely
                }
                risks = append(risks, createRisk(technicalAsset, incomingFlow, likelihood))
            }
        }
    }
    return risks
}
```

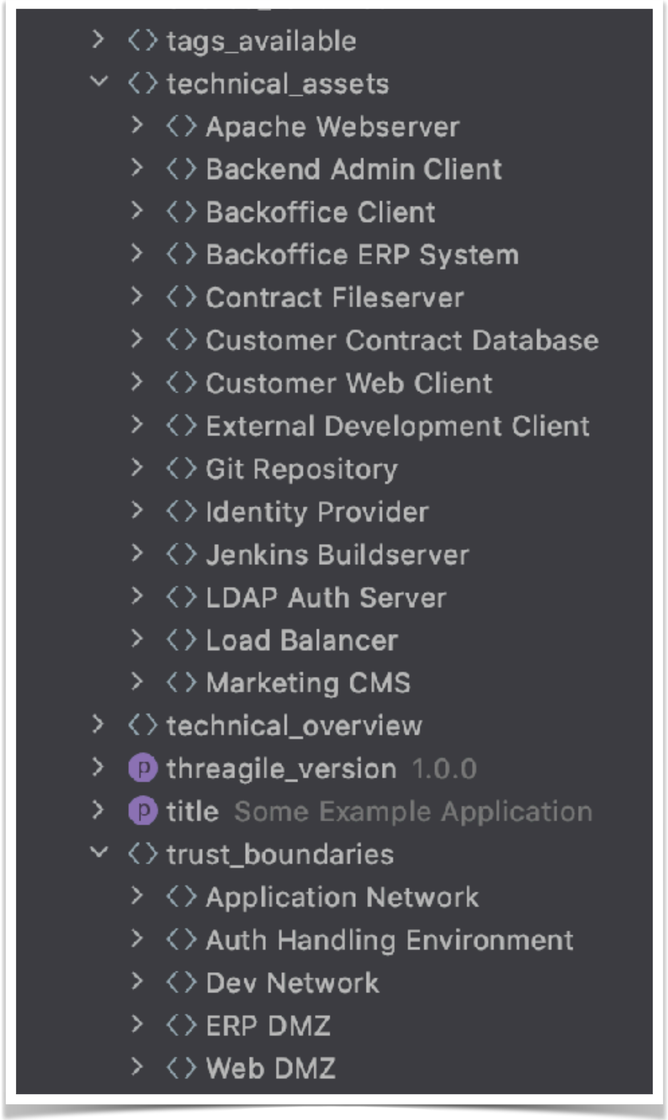
# Manually Identified Risks (put into YAML)

```yaml
Some Individual Risk Example:
  id: something-strange
  description: Some text describing the risk category...
  impact: Some text describing the impact...
  asvs: V0 - Something Strange
  cheat_sheet: https://example.com
  action: Some text describing the action...
  mitigation: Some text describing the mitigation...
  check: Check if XYZ...
  function: business-side # values: business-side, arc
  stride: repudiation # values: spoofing, tampering, r
  detection_logic: Some text describing the detection
  risk_assessment: Some text describing the risk asses
  false_positives: Some text describing the most commo
  model_failure_possible_reason: false
  cwe: 693
```

```yaml
risks_identified:
  <b>Example Individual Risk</b> at <b>Database</b>:
    severity: critical # values: low, medium, elevated, high, critical
    exploitation_likelihood: likely # values: unlikely, likely, very-likely, frequent
    exploitation_impact: medium # values: low, medium, high, very-high
    data_loss_probability: probable # values: improbable, possible, probable
    data_loss_technical_assets: # list of technical asset IDs which might have data loss
      - sql-database
    most_relevant_data_asset:
    most_relevant_technical_asset: sql-database
    most_relevant_communication_link:
    most_relevant_trust_boundary:
    most_relevant_shared_runtime:
  <b>Example Individual Risk</b> at <b>Contract Filesystem</b>:
    severity: medium # values: low, medium, elevated, high, critical
    exploitation_likelihood: frequent # values: unlikely, likely, very-likely, frequent
    exploitation_impact: very-high # values: low, medium, high, very-high
    data_loss_probability: improbable # values: improbable, possible, probable
    data_loss_technical_assets: # list of technical asset IDs which might have data loss
    most_relevant_data_asset:
    most_relevant_technical_asset: contract-fileserver
    most_relevant_communication_link:
    most_relevant_trust_boundary:
    most_relevant_shared_runtime:
```

# Editing Support in IDEs

Nice structured YAML tree in many popular IDEs and YAML editors:



# Editing Support in IDEs

Schema for YAML input available:

Enables syntax validation (error flagging) & auto-completion

```
Apache Webserver:
  id: apache-webserver
  description:
  type: process # values: external-entity, process, da
  usage: business # values: business, devops
  used_as_client_by_human: false
  out_of_scope: false
  justification_out_of_scope:
  size: application # values: system, service, applica
  technology: web-serverrrrr # values: see help
  tags:
    - linux
    - apache
    - aws:ec2
  internet: false
  machine: container # valu
```

Schema validation: Value should be one of: "
"browser", "desktop", "mobile-app", "devops-
"application-server", "database", "file-server"
service-rest", "web-service-soap", "ejb", "sea
registry", "reverse-proxy", "load-balancer", "b
"artifact-registry", "code-inspection-platform
platform", "batch-processing", "event-listene
"identity-store-database", "tool", "cli", "task",
"message-queue", "stream-processing", "ser
"mail-server", "vault", "hsm", "waf", "ids", "ins

```
technology: web # values: see help
tags:
  - linux
  - apache
  - aws:ec2
internet: false
```

web-application
web-server
web-service-rest
web-service-soap

Press ⏎ to insert, →| to replace

```
technology: # values: see help
tags:
  - linux
  - apache
  - aws:ec2
internet: fa
machine: cor
encryption:
owner: Compa
confidential
integrity: c
availability
justificatio
multi_tenant
redundant: f
```

ai
application-server
artifact-registry
batch-processing
block-storage
browser
build-pipeline
cli
client-system
cms
code-inspection-platform
container-platform
data-lake
database
desktop
devops-client

```
  - json
  - file
communicatio
  ERP Syster
    target:
```

iot-device
ips
ldap-server
library
load-balancer
local-file-system
mail-server

ent 1/1 › technical_a
Endpoints

# Editing Support in IDEs

Live Templates:

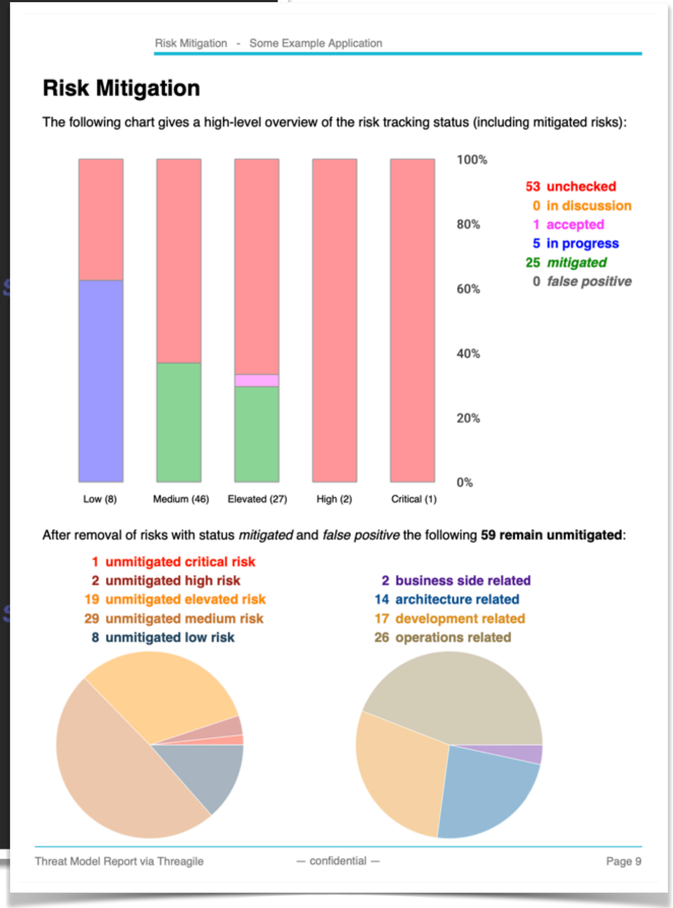
Enables Template-based Quick Editing

```
tech
technical_asset                    Technical Asset
Press ^. to choose the selected (or first) suggestion and insert a dot afterwards  Next Tip
```

```yaml
:
  id:
  description:
  type:
  usage:
  used_as_client_by_human:
  out_of_scope: false
  justification_out_of_scope:
  size:
  technology:
  tags:
  internet:
  machine:
  encryption:
  owner:
  confidentiality:
  integrity:
  availability:
  justification_cia_rating:
  multi_tenant:
  redundant:
  custom_developed_parts:
  data_assets_processed: # sequence of IDs to reference
  data_assets_stored: # sequence of IDs to reference
  data_formats_accepted:
  communication_links:
```

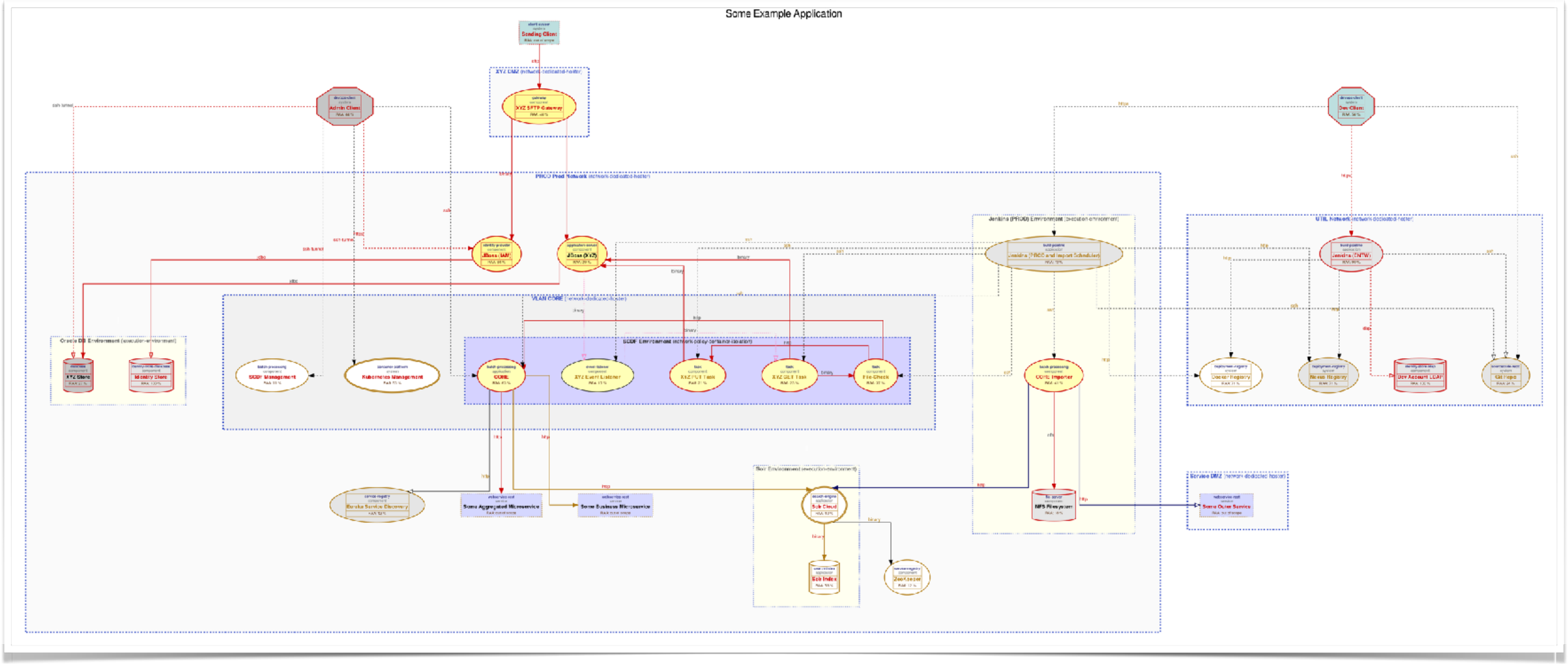
# Risk Tracking (inside YAML file by Risk-ID)

```yaml
risk_tracking:

  untrusted-deserialization@erp-system: # wildcards "*" between the @ characters are possible
    status: accepted # values: unchecked, in-discussion, accepted, in-progress, mitigated, false-positive
    justification: Risk accepted as tolerable
    ticket: XYZ-1234
    date: 2020-01-04
    checked_by: John Doe

  ldap-injection@*@ldap-auth-server@*: # wildcards "*" between the @ characters are possible
    status: mitigated # values: unchecked, in-discussion, accepted, in-progress, mitigated, fals
    justification: The hardening measures were implemented and checked
    ticket: XYZ-5678
    date: 2020-01-05
    checked_by: John Doe

  unencrypted-asset@*: # wildcards "*" between the @ characters are possible
    status: mitigated # values: unchecked, in-discussion, accepted, in-progress, mitigated, fals
    justification: The hardening measures were implemented and checked
    ticket: XYZ-1234
    date: 2020-01-04
    checked_by: John Doe
```




Model-Macro exists for quick seeding of risk instances for tracking in YAML model file

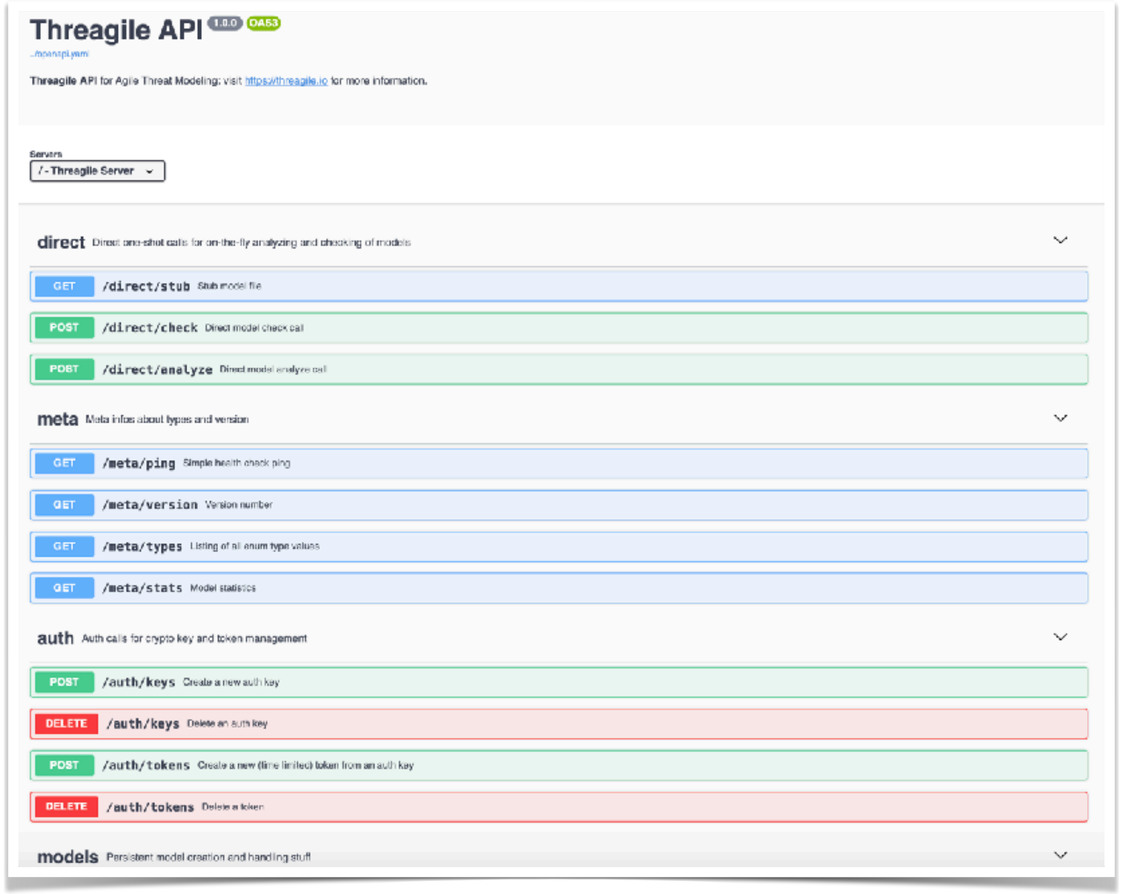
# What About Bigger Models?



# REST-Server

Also within the Docker container

Playground online available for instant playing as well: https://run.threagile.io



# Model Macros: Interactive Wizards

Interactive wizards reading existing models and modify/enhance them

Useful for repeating, often similar, model tasks like:

- Adding a Build-Pipeline to the model
- Adding a Vault to the model
- Adding Identity Provider and Identity Storage to the model
- etc.

Pluggable interface allows for custom model macros

Threagile
Agile Threat Modeling

# Live Demo

**Enhancing an existing model with a build-pipeline via a model-macro**

**(and inspect changes in Data Flow, RAA, Data Breach Probabilities & Risks)**

# Model Macros: Interactive Wizards

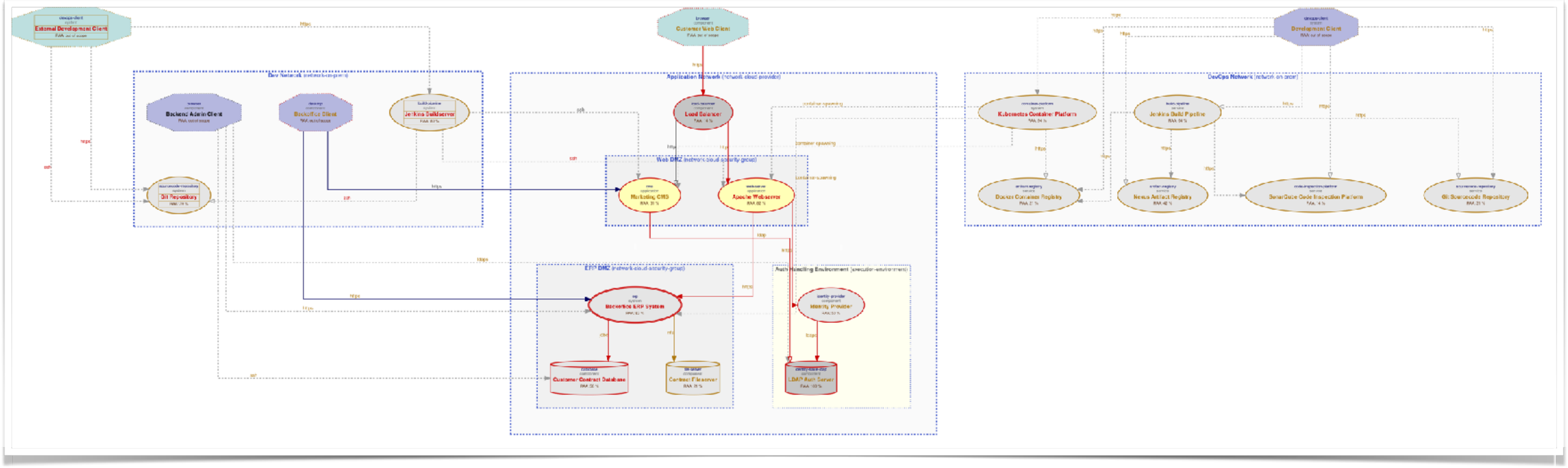
```
==================
Add Build Pipeline
==================
This model macro adds a build pipeline (
registry, container image registry, sour


----------------------------------------
What product is used as the sourcecode r
----------------------------------------
This name affects the technical asset's

Enter your answ
 the model macr
Answer (default
Answer processe


---------------
What product is
---------------
This name affec

Enter your answ
 the model macr
Answer (default
Answer processe


---------------
What product is
---------------
This name affec

Enter your answ
 the model macro)
Answer (default 'Nexus'):
```

```
Enter number to selec
Please select (multip
 select/deselect):
    0: SELECTION PROC
  * 1: apache-webserv
    2: backend-admin-
    3: backoffice-cli
    4: contract-files
    5: customer-clier
  * 6: erp-system
    7: external-dev-c
    8: git-repo
    9: identity-provi
   10: jenkins-builds
   11: ldap-auth-serv
   12: load-balancer
  * 13: marketing-cms
   14: sql-database

Enter number to select/deselect (or 0 wh
```

```
-----------------------------------------------
Of which type shall the new trust boundary be?
-----------------------------------------------
Please choose from the following values (enter value directly or use number):
    1: network-on-prem
    2: network-dedicated-hoster
    3: network-virtual-lan
    4: network-cloud-provider
    5: network-cloud-security-group
    6: network-policy-namespace-isolation

Enter your answer (use 'BACK' to go one step back or 'QUIT' to quit without executing the mod
el macro)
Answer (default 'net
Answer processed


--------------------
What type of deploym
--------------------
Push-based deploymen
Please choose from t
    1: Push-based De
    2: Pull-based De

Enter your answer (u
el macro)
Answer: 2
Answer processed
```

```
#################################################################
Do you want to execute the model macro (updating the model file)?
#################################################################

The following changes will be applied:
 - adding tag: sonarqube
 - adding data asset: sourcecode
 - adding data asset: deployment
 - adding technical asset (including communication links): development-client
 - adding technical asset (including communication links): git-sourcecode-repository
 - adding technical asset (including communication links): docker-container-registry
 - adding technical asset (including communication links): kubernetes-container-platform
 - adding technical asset (including communication links): jenkins-build-pipeline
 - adding technical asset (including communication links): nexus-artifact-registry
 - adding technical asset (including communication links): sonarqube-code-inspection-platform
 - adding trust boundary: devops-network
 - adding shared runtime: kubernetes-container-runtime

Changeset valid

Apply these changes to the model file?
Type Yes or No:
```

# Model Macros: Results



# GitHub Integration (as workflow action)

https://github.com/Threagile/github-integration-example

Threagile / github-integration-example Template

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

main | 1 branch | 0 tags | Go to file | Add file | Code | Use this template

**Threagile** Update threat model report and data-flow diagram by Threagile 45c1674 2 hours ago 9 commits

| | | |
|---|---|---|
| .github/workflows | Sample creation | 4 hours ago |
| threagile/output | Update threat model report and data-flow diagram by Threagile | 2 hours ago |
| LICENSE | Initial commit | 4 hours ago |
| README.md | README update | 3 hours ago |
| threagile.yaml | Test commit to execute the action on threat model change | 2 hours ago |

README.md

## github-integration-example

Example of how to integrate Threagile into GitHub workflows:

This repo acts as some sort of template to see the integration of Threagile into a GitHub workflow in action. Usually here would be a real project with real source and other stuff. Also such a repo contains a **threagile.yaml** file, which contains the threat model input (see the Threagile docs for info about this). *Here we're using the Threagile example YAML file as an example threat model input.*

### GitHub Workflow Integration

# GitHub Integration (as workflow action)

https://github.com/Threagile/github-integration-example

```yaml
on:
  push:
    paths:
      - 'threagile.yaml' # useful to filter this job to execute only when the threat model changes


jobs:

  threagile_job:
    runs-on: ubuntu-latest
    name: Threat Model Analysis
    steps:

      # Checkout the repo
      - name: Checkout Workspace
        uses: actions/checkout@v2

      # Run Threagile
      - name: Run Threagile
        id: threagile
        uses: threagile/run-threagile-action@v1
        with:
          model-file: 'threagile.yaml'

      # Archive resulting files as artifacts
      - name: Archive Results
        uses: actions/upload-artifact@v2
        with:
          name: threagile-report
          path: threagile/output
```

# GitHub Integration (as workflow action)

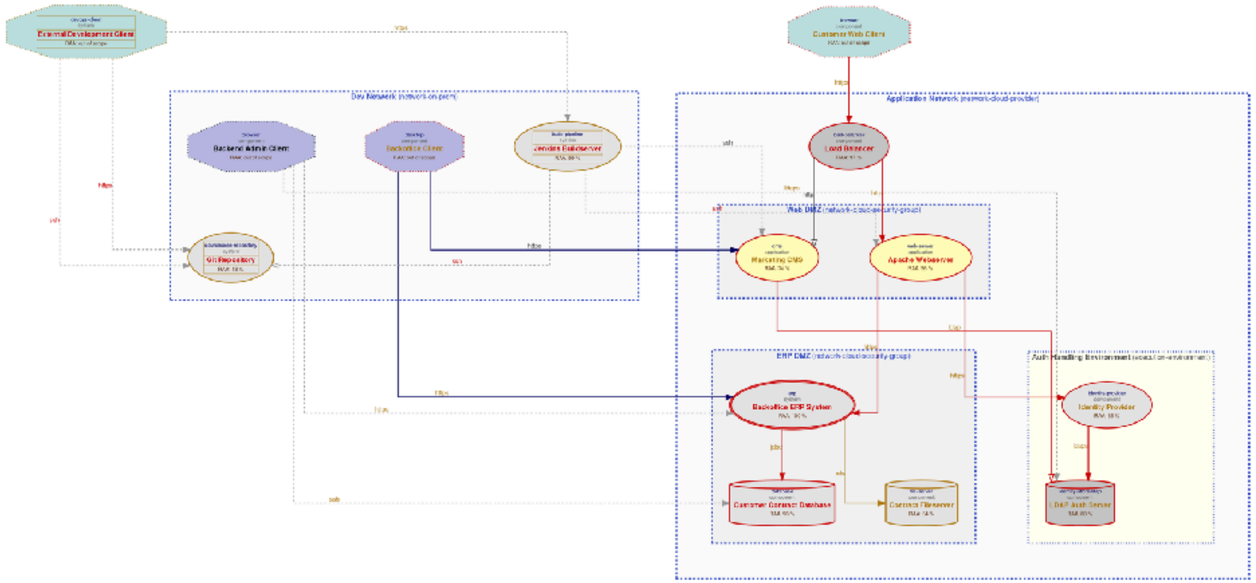
https://github.com/Threagile/github-integration-example

## Threat Model Analysis

The open-source toolkit for agile threat modeling, Threagile, was used to model and analyze potential threats.

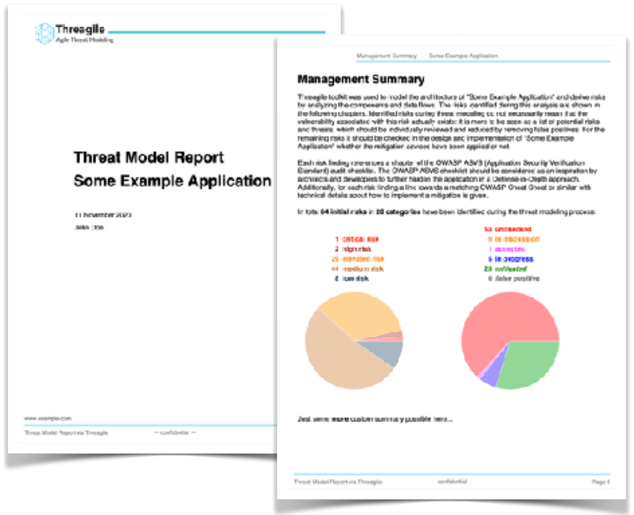
### Data-Flow Diagram (DFD)

The following DFD was generated by Threagile during threat model analysis:



### Threat Model Report

The following report was generated by Threagile during threat model analysis: Threat Model Report



# Possible Effects

**Custom coded risk rules can analyze the model graph**

(helps big corporations with individual policies)

# Possible Effects

**Uniform documentation of system landscape built bottom-up**

(by dev teams in their IDEs along with the codebase)

# Possible Effects

**Instant regeneration of project risk landscape on changes**

(what happens when a data classification changes or some component moves into the cloud)

# Possible Effects

**Instant regeneration of <u>corporate-wide</u> risk landscape on changes**

(just modify a risk rule due to a policy change
and instantly regenerate threat models across all projects)

# Possible Effects

**CI/CD-Pipelines can check the generated JSON for unmitigated risks**

(trend graphs & warning when rollout contains new unchecked high risks)

*Threat Modeling as a part of DevSecOps*

# Possible Effects

**Security is less bottleneck for threat model sign-offs**

(risks rules as code automate threat model vetting)

# Upcoming Features (currently in development)

**More Docs, Samples & Screencasts & Web-based Model Editor:**
Easier on-boarding of new users.

**Model Linking & Model Includes:**
Referencing other models (external systems): reference vs. inclusion as “Sub-Models”.

**Cloud Crawler:**
Crawling Cloud environments (preferably as “Model-Macro”) with wizard to selectively take cloud components into a Threagile model.

**GitLab Integration:**
Further integrations into SCM workflows: preferably via “Actions” and Web-Hooks.

**CloudFormation / Terraform Import:**
“Model-Macro” based wizard to import infrastructure declarations into model.

# Upcoming Features (currently in development)

**Build Pipeline Plugins (Jenkins, …):**
Close integration into CI/CD pipelines.

**LeanIX / EA Integration via API:**
Integration with enterprise architecture tools like “LeanIX”, “Enterprise Architect (EA)” and others.

**Bug Tracker Integration (JIRA, …):**
Bi-directional integration with bug trackers (like JIRA) for risk mitigation state management: preferably via Web-Hooks.

***Your Ideas and Feature Requests:***
Feedback welcome: Create feature request tickets on https://github.com/threagile

# Released as Open-Source

Threagile
Agile Threat Modeling

**Website:**
- https://threagile.io

**Playground:**
- https://run.threagile.io

**Community (Support) Chat:**
- https://gitter.im/threagile/community

**Source:**
- https://github.com/threagile

**Container:**
- https://hub.docker.com/r/threagile

## Q & A

**Questions?**

www.Christian-Schneider.net
mail@Christian-Schneider.net
@cschneider4711 on Twitter