Agile Threat Modeling with OpenSource Tools
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

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?
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...
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:

```markdown
???
```
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)
... 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

Runs totally offline (of course)
Running Threagile

Either as

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

Available as a Docker container:

docker run --rm -it threagile/threagile
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 and contains

- Data Assets
- Technical Assets
- Communication Links
- Trust Boundaries
— and optionally more things
Example Model: Data Assets

data_assets:

Customer Contracts: &customer-contracts # this example shows:

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, important

availability: operational # values: archive, operational
Example Model:
Technical Assets

<table>
<thead>
<tr>
<th>Apache Webserver</th>
</tr>
</thead>
<tbody>
<tr>
<td>id: apache-webserver</td>
</tr>
<tr>
<td>description:</td>
</tr>
<tr>
<td>type: process # values: external-entity, process</td>
</tr>
<tr>
<td>usage: business # values: business, devops</td>
</tr>
<tr>
<td>used_as_client_by_human: false</td>
</tr>
<tr>
<td>out_of_scope: false</td>
</tr>
<tr>
<td>justification_out_of_scope:</td>
</tr>
<tr>
<td>size: application # values: system, service</td>
</tr>
<tr>
<td>technology: web-server # values: see help</td>
</tr>
<tr>
<td>tags:</td>
</tr>
<tr>
<td>- linux</td>
</tr>
<tr>
<td>- apache</td>
</tr>
<tr>
<td>- aws:ec2</td>
</tr>
<tr>
<td>internet: false</td>
</tr>
<tr>
<td>machine: container # values: physical, virtual</td>
</tr>
<tr>
<td>encryption: none # values: none, transparent</td>
</tr>
<tr>
<td>owner: Company ABC</td>
</tr>
<tr>
<td>confidentiality: internal # values: public, private</td>
</tr>
<tr>
<td>integrity: critical # values: archive, open</td>
</tr>
<tr>
<td>availability: critical # values: archive, open</td>
</tr>
<tr>
<td>justification_cia_rating:</td>
</tr>
<tr>
<td>multi_tenant: false</td>
</tr>
<tr>
<td>redundant: false</td>
</tr>
<tr>
<td>custom_developed_parts: true</td>
</tr>
</tbody>
</table>
Example Model: Referencing Data Assets (Processed & Stored)

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

dataFormatsaccepted: # sequence of formats like: json, xml, serialization, file, csv
- json
- file
Example Model: Communication Links

```json
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

```
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 ;)

*(including custom developed ones)*
Model Graph Generation (Data Flows)
PDF & Excel Report Generation
Impact Summary (before & after mitigation)

Management Summary

The Glove 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 advice has 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 broaden 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:

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

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):

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 not mitigated, 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 not mitigated, attackers might be able to read sensitive files (configuration data, key/rsion files, deployment files, business data files, etc.) from 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 not mitigated, 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 not mitigated, 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: 8 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

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

- 53 unchecked
- 0 in discussion
- 5 accepted
- 25 in progress
- 0 false positive

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
26. 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):

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 execution.

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.) from 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

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.

**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 Attack Atractiveness (RAA) value of 95 % 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: 0 / 2 Risks - Exploitation likelihood is Likely with Medium impact.
When a file system is accessed Path-Traversal or Local-File-Inclusion (LFI) or Remote-File-Inclusion (RFI) 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 (e.g., a 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 integrity rating of this 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 (which attached 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: 0 / 2 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

This chapter discusses and assigns the risks by function which are most likely able to mitigate these. In total 84 potential risks have been identified during the threat model which 11 should be checked by Business Side, 14 should be checked by Architecture, and 48 should be checked by Development. Risk findings are available and link to the corresponding chapter.

**Business Side**


Medium: Missing Two-Factor Authentication (2FA): 0/9 Risks - Exploitation likelihood is 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: 2/9 Risks - Exploitation likelihood is Likely with Medium impact.

Apply an authentication method to the technical asset. To protect highly sensitive data use of two-factor authentication (or human users).

**Development**

High: SQL/DB Injection: 1/1 Risk - Exploitation likelihood is Very Likely with High impact. Try to use parameterbinding to be safe from injection 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 path-level.

Low: XML External Entity (XXE): 1/1 Risk - Exploitation likelihood is Likely with Medium impact.

When a third-party product is used instead of custom developed software, check if the product applies the proper mitigation and ensure a reasonable path-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: 0/8 Risks - Exploitation likelihood is Likely with Medium impact.

Apply safe cloud practices (like CAS benchmarks, ISSAP recommendations, vendor recommendations, DevSec Hardening Framework, DSSAT 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.

**Assigned by Function - Some Example Application**

Medium: Missing Vault (Secret Storage): 1/1 Risk - Exploitation likelihood is Unlikely with Medium impact.

Consider using a vault (Secret Storage) to secure and access config secrets (e.g., encryption keys, client certificates, etc.).

Medium: Patch instead of Roll Deployment: 3/3 Risks - Exploitation likelihood is Unlikely with Medium impact.

Try to avoid pull-based deployments (like SSH scripts or scenarios offer) over push-based deployments.

Medium: Unchecked Deployment: 3/3 Risks - Exploitation likelihood is Unlikely with Medium impact.

Apply DevOps best practices and use scanning tools to identify vulnerabilities as in którym dependencies, container layers, and optionally use static/dynamic scan against attack systems.

Medium: SQL/DB Injection: 1/1 Risk - Exploitation likelihood is Very Likely with High impact. Try to use parameter binding to be safe from injection 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 path-level.

Medium: Cross-Site Request Forgery (CSRF): 7/7 Risks - Exploitation likelihood is Very Likely with Low impact.

Try to obscure CSRF tokens with the double-slash 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 origin 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 path-level.

**Assignment by Function - Some Example Application**

Medium: Missing Vault (Secret Storage): 1/1 Risk - Exploitation likelihood is Unlikely with Medium impact.

Consider using a vault (Secret Storage) to secure and access config secrets (e.g., encryption keys, client certificates, etc.).

Medium: Patch instead of Roll Deployment: 3/3 Risks - Exploitation likelihood is Unlikely with Medium impact.

Try to avoid pull-based deployments (like SSH scripts or scenarios offer) over push-based deployments.

Medium: Unchecked Deployment: 3/3 Risks - Exploitation likelihood is Unlikely with Medium impact.

Apply DevOps best practices and use scanning tools to identify vulnerabilities as in_components dependencies, container layers, and optionally use static/dynamic scan against attack systems.

Development

High: SQL/DB Injection: 1/1 Risk - Exploitation likelihood is Very Likely with High impact. Try to use parameter binding to be safe from injection 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 path-level.

Low: XML External Entity (XXE): 1/1 Risk - Exploitation likelihood is Likely with Medium impact.

When a third-party product is used instead of custom developed software, check if the product applies the proper mitigation and ensure a reasonable path-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: 0/8 Risks - Exploitation likelihood is Likely with Medium impact.

Apply safe cloud practices (like CAS benchmarks, ISSAP recommendations, vendor recommendations, DevSec Hardening Framework, DSSAT 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: SQL/DB Injection: 1/1 Risk - Exploitation likelihood is Unlikely with Medium impact.

Consider using a vault (Secret Storage) to secure and access config secrets (e.g., encryption keys, client certificates, etc.).

Medium: Patch instead of Roll Deployment: 3/3 Risks - Exploitation likelihood is Unlikely with Medium impact.

Try to avoid pull-based deployments (like SSH scripts or scenarios offer) over push-based deployments.

Medium: Unchecked Deployment: 3/3 Risks - Exploitation likelihood is Unlikely with Medium impact.

Apply DevOps best practices and use scanning tools to identify vulnerabilities as in_components dependencies, container layers, and optionally use static/dynamic scan against attack systems.

Medium: SQL/DB Injection: 1/1 Risk - Exploitation likelihood is Very Likely with High impact. Try to use parameter binding to be safe from injection 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 path-level.

Medium: Cross-Site Request Forgery (CSRF): 7/7 Risks - Exploitation likelihood is Very Likely with Low impact.

Try to obscure CSRF tokens with the double-slash 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 origin 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 path-level.

**Assignment by Function - Some Example Application**

Medium: SQL/DB Injection: 1/1 Risk - Exploitation likelihood is Very Likely with High impact. Try to use parameter binding to be safe from injection 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 path-level.

Medium: Cross-Site Request Forgery (CSRF): 7/7 Risks - Exploitation likelihood is Very Likely with Low impact.

Try to obscure CSRF tokens with the double-slash 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 origin 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 path-level.
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:

- 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 Filserver: 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
Detailed mitigations along with links to
- OWASP ASVS Chapter
- OWASP CSVS Chapter
- OWASP Cheat Sheet
- etc.
Risk Instances (by vulnerability & by tech asset)
<table>
<thead>
<tr>
<th></th>
<th>Severity</th>
<th>Likelihood</th>
<th>Impact</th>
<th>STRIDE</th>
<th>Function</th>
<th>CWE</th>
<th>Risk Category</th>
<th>Technical Asset</th>
<th>Communication Link</th>
<th>RAA %</th>
<th>Identified Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Critical</td>
<td>Likely</td>
<td>Medium</td>
<td>Reputation</td>
<td>Business Site</td>
<td>CWE-699</td>
<td>Some individual Risk Example</td>
<td>Contract Filerserver</td>
<td>Database Traffic</td>
<td>95%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>2</td>
<td>High</td>
<td>Very High</td>
<td>Very High</td>
<td>Repudiation</td>
<td>Business Site</td>
<td>CWE-699</td>
<td>Some individual Risk Example</td>
<td>Contract Filerserver</td>
<td>Database Traffic</td>
<td>95%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>3</td>
<td>High</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>95%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>4</td>
<td>High</td>
<td>Likely</td>
<td>High</td>
<td>Information Disclosure</td>
<td>Development</td>
<td>CWE-611</td>
<td>IMM Entity (EVT)</td>
<td>Backoffice ERP System</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>93%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>6</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>7</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>8</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>9</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>10</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>11</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>12</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>13</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>14</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>15</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>16</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>17</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>18</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>19</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>20</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>21</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>22</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>23</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>24</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>25</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>26</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>27</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>28</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>29</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>30</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>31</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>32</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>33</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>34</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>35</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>36</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>37</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>38</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>39</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>40</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
<tr>
<td>41</td>
<td>Elevated</td>
<td>Likely</td>
<td>High</td>
<td>Temporizing</td>
<td>Development</td>
<td>CWE-209</td>
<td>SQL/NoSQL Injection</td>
<td>Backoffice ERP System</td>
<td>Database Traffic</td>
<td>91%</td>
<td>Example Individual Risk at Contract Filerserver</td>
</tr>
</tbody>
</table>
Detail Results as JSON

```
{
  "category": "container-baseline-backdooring",
  "risk_status": "unchecked",
  "severity": "medium",
  "exploitation_likelihood": "unlikely",
  "exploitation_impact": "high",
  "title": "Container Baseline Backdooring Risk at \u003e\u003e 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-baseline-backdooring",
  "risk_status": "unchecked",
  "severity": "medium",
  "exploitation_likelihood": "unlikely",
  "exploitation_impact": "high",
  "title": "Container Baseline Backdooring Risk at \u003e\u003e Marketing CMS",
  "synthetic_id": "container-baseline-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)
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."
        Impact: "If this risk remains unmitigated, the LDAP server in order to stay safe from",
        Check: "Are recommendations from the",
        Function: model.Development,
        STRIDE: model.Tampering,
        DetectionLogic: "In-scope clients accessing",
        RiskAssessment: "The risk rating depends on",
        FalsePositives: "LDAP server queries by search",
        ModelFailurePossibleReason: false,
        CWE: 98,
    }
}

func GenerateRisks() []model.Risk {
    risks := make([])model.Risk, 0
    for _, technicalAsset := range model.ParsedModelRoot.TechnicalAssets {
        incomingFlows := model.IncomingTechnicalCommunicationLinksMappedByTargetId[technicalAsset]
        for _, incomingFlow := range incomingFlows {
                continue
            }
                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)

---

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, and
- stride: repudiation # values: spoofing, tampering, ...
- detection_logic: Some text describing the detection...
- risk_assessment: Some text describing the risk assessment...
- false_positives: Some text describing the most common false positive...
- model_failure_possible_reason: false
- cwe: 693

---

```yaml
risks_identified:
  - Example Individual Risk at Database:
    - 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:

  - Example Individual Risk at Contract Filesystem:
    - 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
      - contract-fileserver
    - 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
Editing Support in IDEs

Live Templates:

Enables Template-based Quick Editing
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, false-positive
    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, false-positive
    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
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 (development client, registry, container image registry, source code repository).

What product is used as the source code repository?

This name affects the technical asset's title and ID plus a

Enter your answer (use 'BACK' to go one step back or 'QUIT' to quit without executing the model macro): Answer (default 'Git'): Answer processed

What type of deployment

Push-based deployments are easier to deploy. Please choose from the following:
1: Push-based Deployment
2: Pull-based Deployment

Enter your answer (use 'BACK' to go one step back or 'QUIT' to quit without executing the model 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: source code
- adding data asset: deployment
- adding technical asset (including communication links): development
tools
- adding technical asset (including communication links): git-sourcecode
- adding technical asset (including communication links): docker-containers
- adding technical asset (including communication links): kubernetes-cluster
- adding technical asset (including communication links): jenkins
- adding technical asset (including communication links): nexus-artifactory
- adding technical asset (including communication links): sonarqube
- 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

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>Last Updated</th>
<th>Commits</th>
</tr>
</thead>
<tbody>
<tr>
<td>.github/workflows</td>
<td>Sample creation</td>
<td>4 hours ago</td>
<td></td>
</tr>
<tr>
<td>thread/gile/output</td>
<td>Update threat model report and data-flow diagram by Threagile</td>
<td>2 hours ago</td>
<td></td>
</tr>
<tr>
<td>LICENSE</td>
<td>Initial commit</td>
<td>4 hours ago</td>
<td></td>
</tr>
<tr>
<td>README.md</td>
<td>README update</td>
<td>3 hours ago</td>
<td></td>
</tr>
<tr>
<td>threagile.yaml</td>
<td>Test commit to execute the action on threat model change</td>
<td>2 hours ago</td>
<td></td>
</tr>
</tbody>
</table>

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 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
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 corporate-wide 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 (+ Layered Graphs):
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 / Helm Import:
“Model-Macro” based wizard to import infrastructure declarations into model.
Upcoming Features (currently in development)

Build Pipeline Plugins (Jenkins, Azure DevOps, etc.):
Close integration into CI/CD pipelines.

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

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

Drawing App Integrations
Import and/or export with draw.io

Your Ideas and Feature Requests:
Feedback welcome: Create feature request tickets on https://github.com/threagile
Released as Open-Source

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

Questions?
www.Christian-Schneider.net
mail@Christian-Schneider.net
@cschliefer4711 on Twitter