DevSecOps.

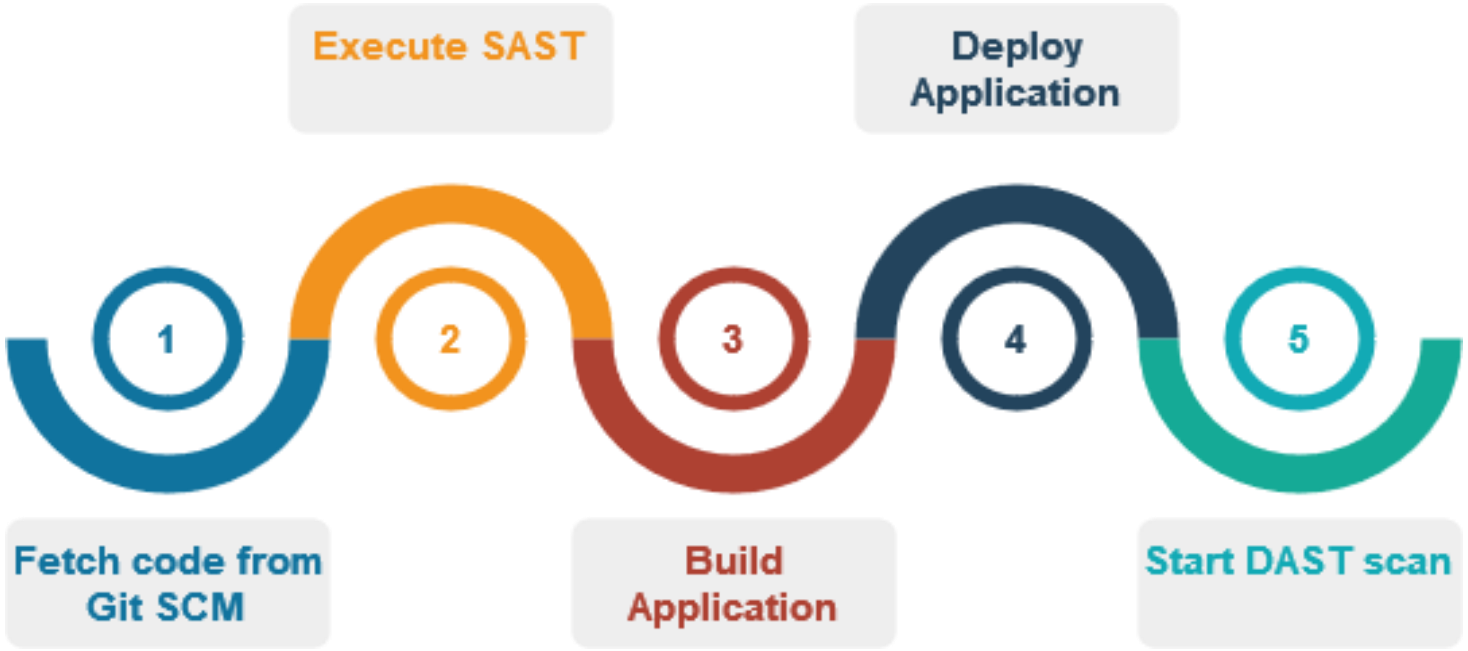
DevSecOps practice

- Identifying and fixing vulnerabilities.
- Balancing speed and security.
- Security is considered a bottleneck.
- Lack of resources and knowledge gap.
- Roles and responsibility alignment.

DevSecOps principles

- Shift-Left Security
- Automation
- Continuous Integration and Continuous Deployment (CI/CD)
- Collaboration and Communication
- Infrastructure as Code
- Security Testing
- Continuous Monitoring
- Culture of Security

DevSecOps pipeline model



Static application security testing (SAST)

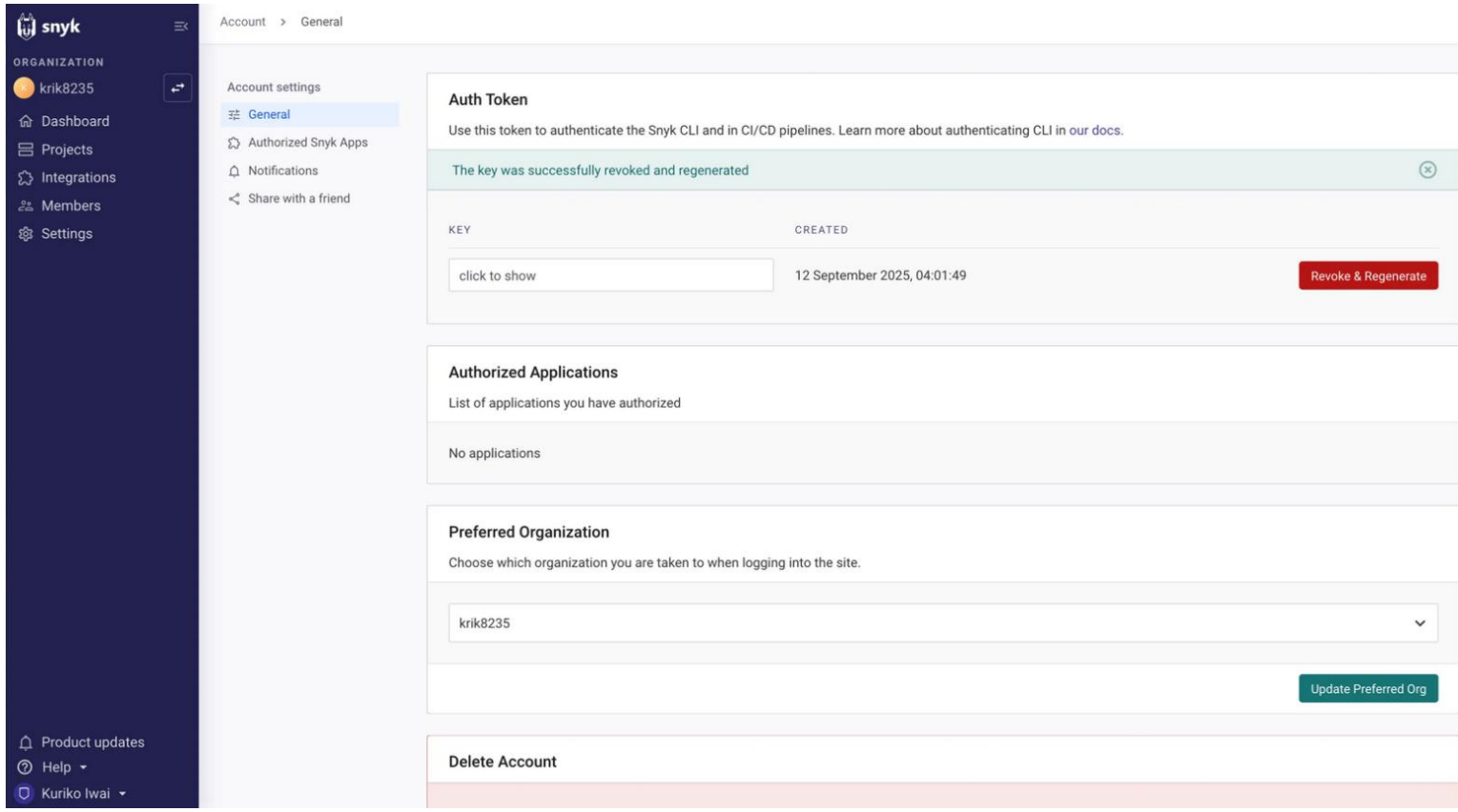
Topic	Static application security testing (SAST)
Definition	Static Application Security Testing (SAST), are 'white-box' tools to inspect source code (Imperva 2024c).
Characteristics	A developer's approach – testers have access to underlying framework, design and implementation to fix vulnerabilities **early** in the development cycle without executing the program. Code is analysed more quickly than humans and real-time feedback with graphical representations show the exact location of vulnerabilities and problem code. Customized reports for dashboards can be produced and the process can be automated.
Example	Some examples of SAST Tools: Aikido Security, SAST by Cycode, Checkmarx, Contrast Security, Fortify, GitLab, HCL AppScan, Snyk, Sonar, Coverity Static Analysis by Synopsys, Veracode.

What is SAST?

- Static Application Security Testing
- Analyze source code without execution
- Look for patterns
  - Vulnerabilities
  - Dangerous data handling
  - No logic flaws
- Types
  - Manual
  - Automated
- SAST is not SCA or IAST

Why SAST?

- DAST, pen testing can catch many defects
- SAST can catch many too
- SAST tools typically look for:
  - User controlled (tainted) data
  - Data flow defects
  - Source code syntax problems
  - Specific vulnerabilities


snyk
ORGANIZATION
krik8235
Dashboard
Projects
Integrations
Members
Settings
Product updates
Help
Kuriko Iwai
Account > General
Account settings
General
Authorized Snyk Apps
Notifications
Share with a friend
Auth Token
Use this token to authenticate the Snyk CLI and in CI/CD pipelines. Learn more about authenticating CLI in our docs.
The key was successfully revoked and regenerated
KEY
CREATED
click to show
12 September 2025, 04:01:49
Revoke & Regenerate
Authorized Applications
List of applications you have authorized
No applications
Preferred Organization
Choose which organization you are taken to when logging into the site.
krik8235
Update Preferred Org
Delete Account

Dynamic application security testing (DAST)

Topic	Dynamic application security testing (DAST)
Definition	Dynamic Application Security Testing (DAST), are ‘black-box’ tools to test products during operation and provide feedback on compliance and general security issues. These tools are used during the testing phase (late in the cycle) (Imperva 2024c).
Characteristics	A hacker's approach: testers have no knowledge of the internals. Testing highlights authentication and server configuration issues and is language independent. The tests evaluate the whole application and system, checks memory consumption and resource use and attempts to break encryption algorithms from the outside. The tests check for cross-site scripting, SQL injection, and cookie manipulation, as well as for vulnerabilities in third-party interfaces.
Example	Some popular DAST tools include: The Open Worldwide Application Security Project (OWASP), Zed Attack Proxy (ZAP), Arachni, Web Application Attack and Audit Framework (w3af).

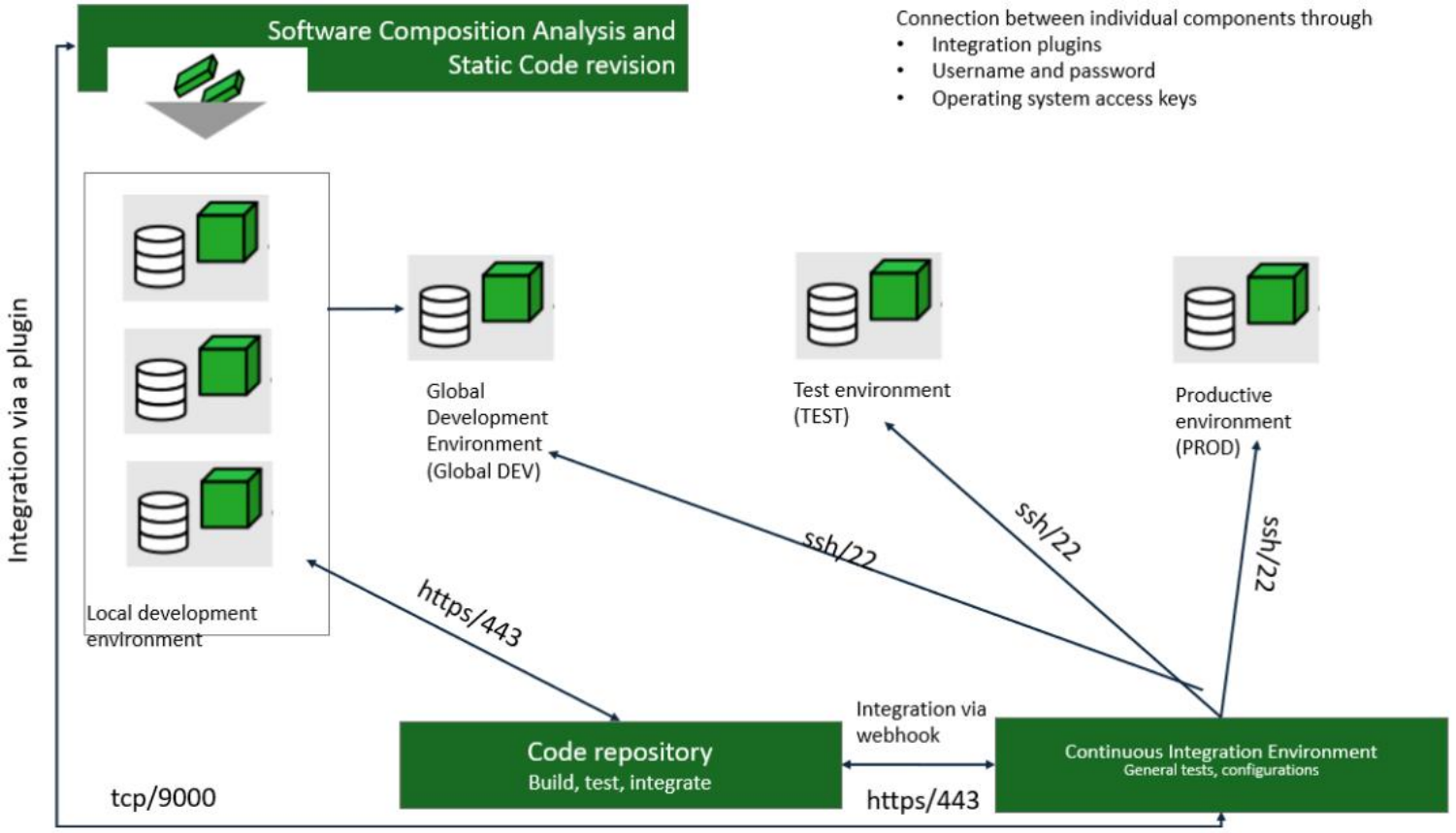
How DAST works

- Simulates attacks
- Analyzes responses
- Tests at runtime

Key benefits of DAST

- Real-world vulnerability detection
- Source code not required
- Low false positives
- Early detection

Segregated environments sample architecture



Trike – 1/2

- a security audit framework - threat modeling as a technique from a risk-management and defensive perspective
- To define a system: enumerating and understanding the system's actors, assets, intended actions, and rules.
- To create an actor-asset-action matrix - columns represent assets and the rows represent actors.
- Each cell of the matrix is divided into four parts, one for each action of CRUD (creating, reading, updating, and deleting). In these cells, the analyst assigns one of three values: allowed action, disallowed action, or action with rules. A rule tree is attached to each cell.

Trike – 2/2

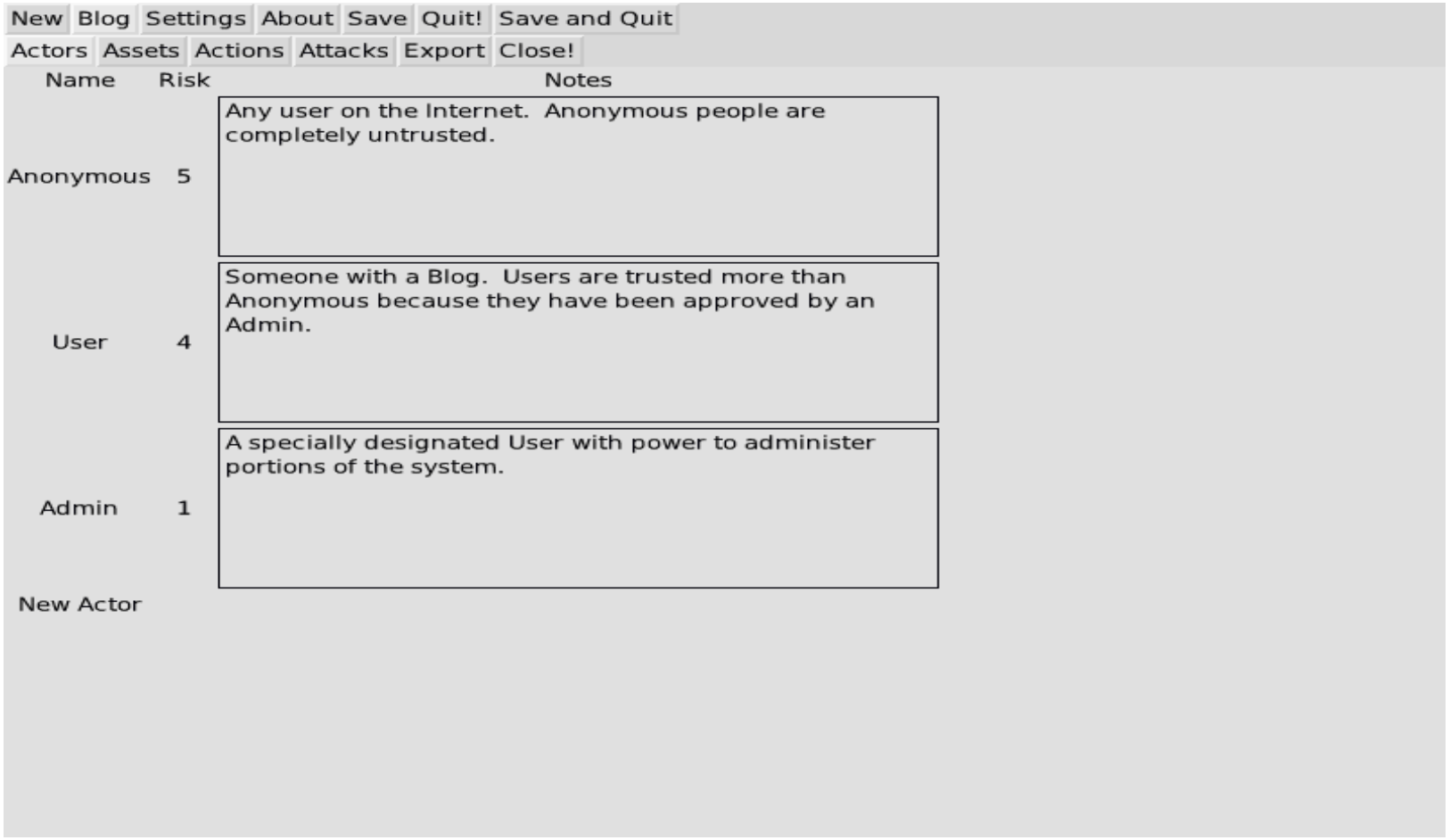
- A data flow diagram (DFD) is built. Each element is mapped to a selection of actors and assets. Iterating through the DFD, the analyst identifies threats, which fall into one of two categories: elevations of privilege or denials of service. Each discovered threat becomes a root node in an attack tree.
- Actors are rated on five-point scales for the risks they are assumed to present (lower number = higher risk) to the asset. Also, actors are evaluated on a three-dimensional scale (always, sometimes, never) for each action they may perform on each asset.

Requirements

- Actors
  - People (roles) who interact with system.
- Assets
  - Specific pieces of data attacker wants.
- Actions
  - What Actors do to Assets.
  - Ex: Create, Read, Update, Delete.

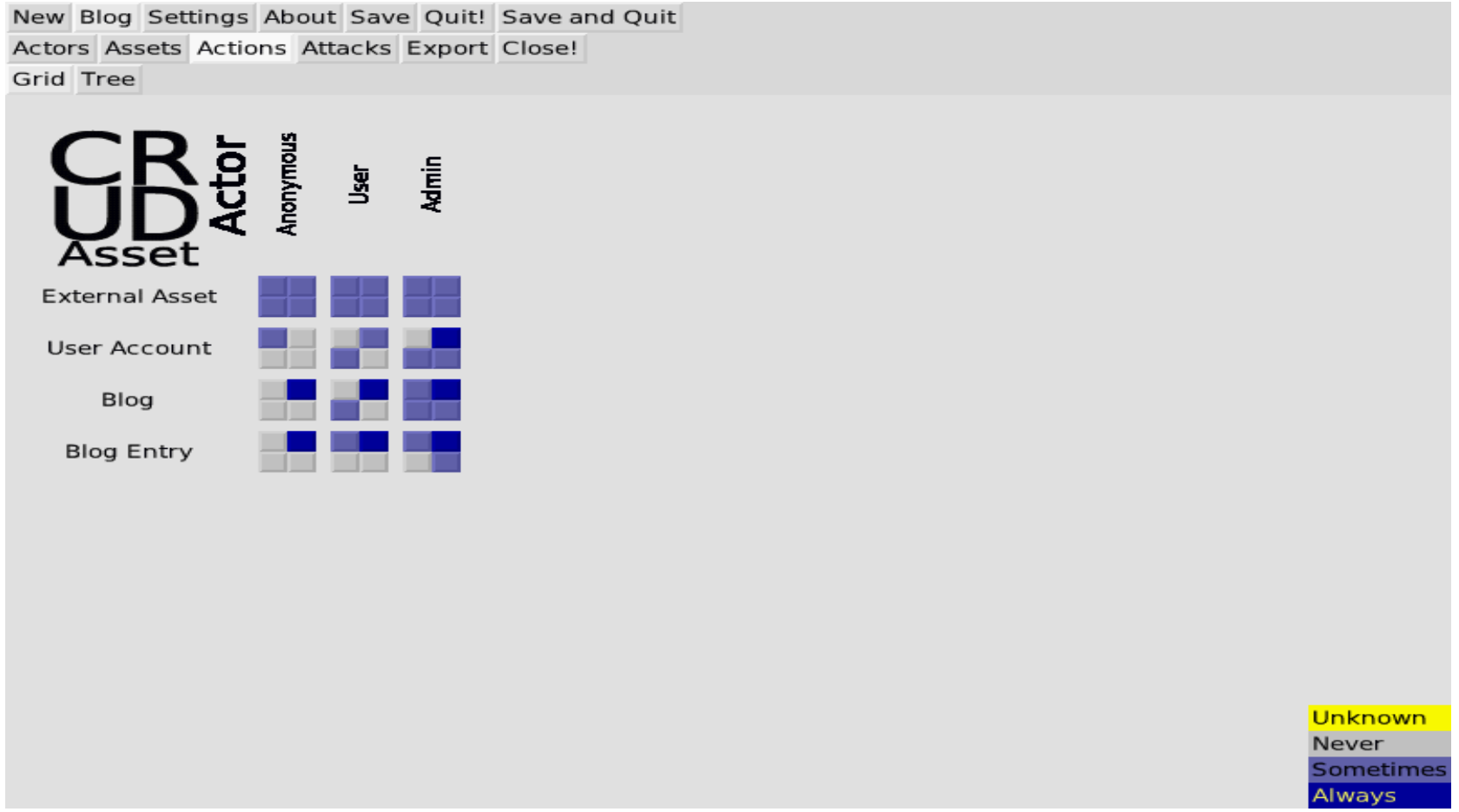
Resource: https://kosh.nku.edu/~waldenj/classes/2006/spring/csc593/presentations/ThreatModeling.ppt

Trike: Actors



Resource: https://kosh.nku.edu/~waldenj/classes/2006/spring/csc593/presentations/ThreatModeling.ppt

Trike: Actor-Asset-Action Matrix



Resource: https://kosh.nku.edu/~waldenj/classes/2006/spring/csc593/presentations/ThreatModeling.ppt

Rules

Rules apply to each Action.

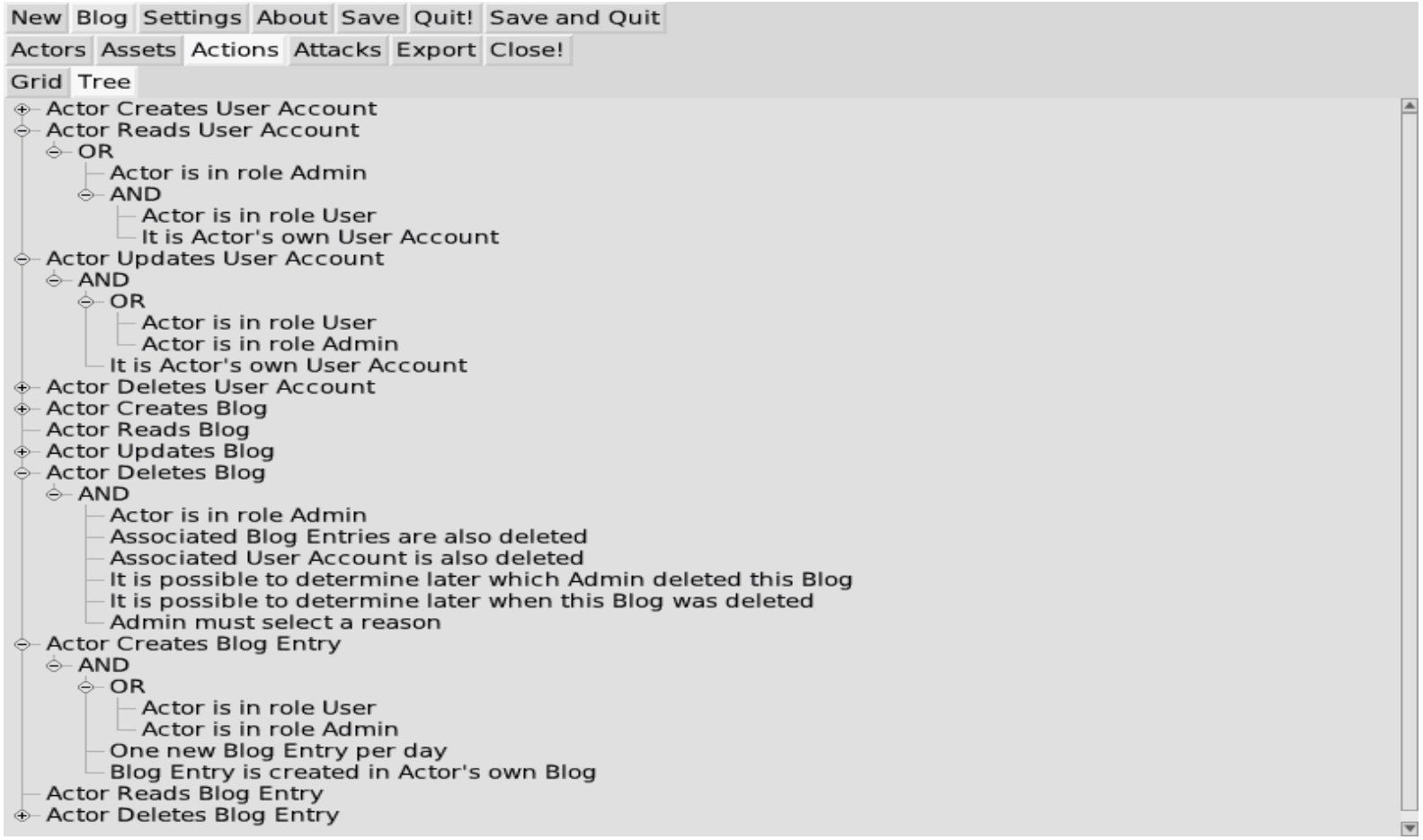
- Limit circumstances in which Actions can occur.
- Boolean tree of conditionals.

Actors are represented as rule:

- User is in Role

Resource: https://kosh.nku.edu/~waldenj/classes/2006/spring/csc593/presentations/ThreatModeling.ppt

Trike: Part of Rules Tree



Resource: https://kosh.nku.edu/~waldenj/classes/2006/spring/csc593/presentations/ThreatModeling.ppt

Threat Generation

Use Actor-Asset-Action matrix.

Two types of threats via Rules:

- Denial of Service: Actor prevented from performing allowed Action.
- Elevation of Privilege: Actor performs an action which is prohibited by matrix.

Resource: https://kosh.nku.edu/~waldenj/classes/2006/spring/csc593/presentations/ThreatModeling.ppt

Example of actor-asset-action matrix

	Asset 1 (i.e. Image Upload)		Asset 2 (i.e. Image Comment)		Asset 3 … Asset N	
Actor 1 / Role A (i.e. Admin)	C | Allowed, Rule: i.e. “Only certain file types” and “Only for own account”	R | Allowed, Rule: i.e. “Only for own account”	C | Allowed, Rule: i.e. “Only for owned photos and friend photos”	R | Allowed, Rule: i.e. "For all friend or public images”	C | (Allowed, Disallowed, Allowed, Rule:)	R | (Allowed, Disallowed, Allowed, Rule:)
	U | Allowed, Rule: i.e. “Only for own account”	D | Allowed, Rule: i.e. “Only for own account”	U | Allowed, Rule: i.e. “For all comments posted by self”	D | Allowed, Rule: i.e. “For all comments posted by self”	U | (Allowed, Disallowed, Allowed, Rule:)	D | (Allowed, Disallowed, Allowed, Rule:)
Actor 2 / Role B (i.e. Anonymous User)	C | Disallowed	R | Disallowed	C | Disallowed	R | Allowed, Rule: i.e. “For photos set to public”	C | (Allowed, Disallowed, Allowed, Rule:)	R | (Allowed, Disallowed, Allowed, Rule:)
	U | Disallowed	D | Disallowed	U | Disallowed	D | Disallowed	U | (Allowed, Disallowed, Allowed, Rule:)	D | (Allowed, Disallowed, Allowed, Rule:)
Actor 3 / Role C …	C | (Allowed, Disallowed, Allowed, Rule:)	R | (Allowed, Disallowed, Allowed, Rule:)	C | (Allowed, Disallowed, Allowed, Rule:)	R | (Allowed, Disallowed, Allowed, Rule:)	C | (Allowed, Disallowed, Allowed, Rule:)	R | (Allowed, Disallowed, Allowed, Rule:)
Actor N / Role Z	U | (Allowed, Disallowed, Allowed, Rule:)	D | (Allowed, Disallowed, Allowed, Rule:)	U | (Allowed, Disallowed, Allowed, Rule:)	D | (Allowed, Disallowed, Allowed, Rule:)	U | (Allowed, Disallowed, Allowed, Rule:)	D | (Allowed, Disallowed, Allowed, Rule:)

Data Flow Diagrams

Visual model of system data flow.

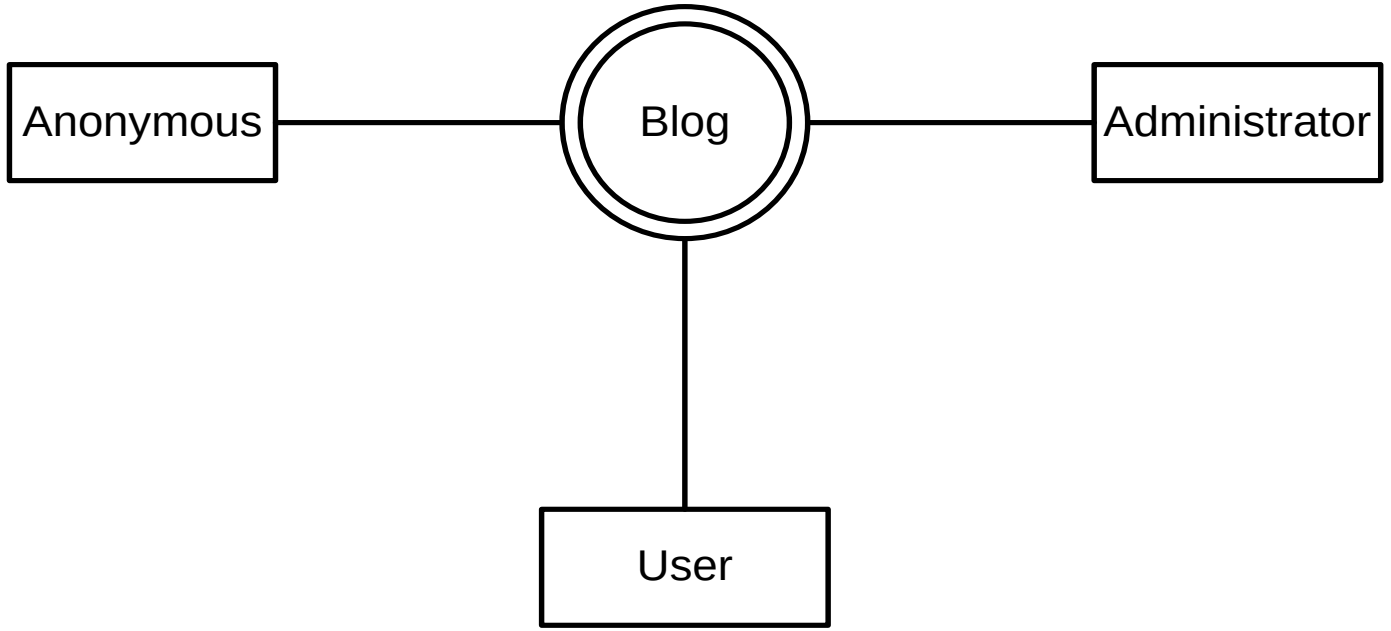
- Rectangles: External actors.
- Circles: Processes.
- Double Lines: Data stores.
- Lines: Data flows.
- Dotted Lines: Trust boundaries.

Hierarchical decomposition

- Until no process crosses trust boundaries.

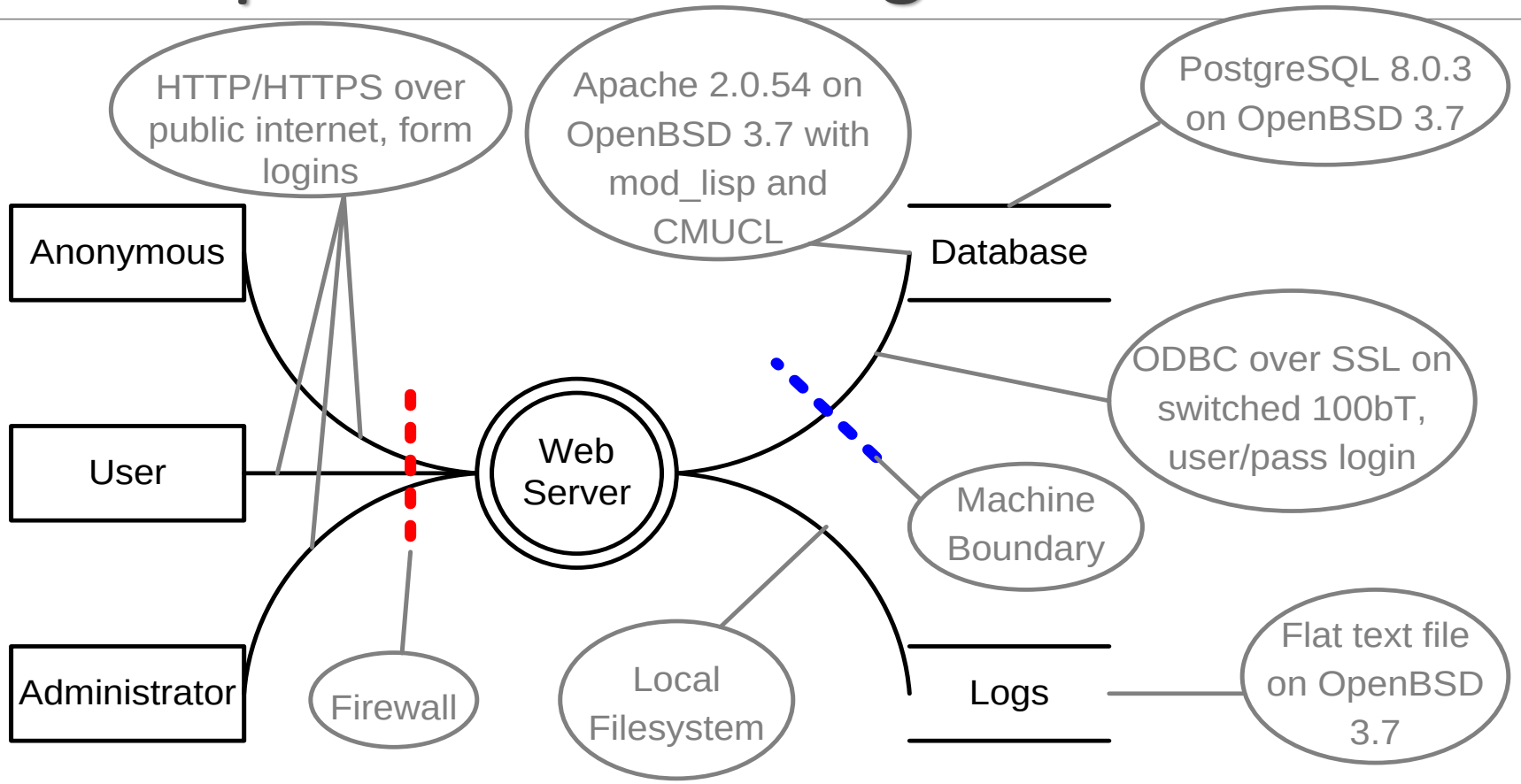
Resource: https://kosh.nku.edu/~waldenj/classes/2006/spring/csc593/presentations/ThreatModeling.ppt

Trike Example: Data Flow Context Diagram



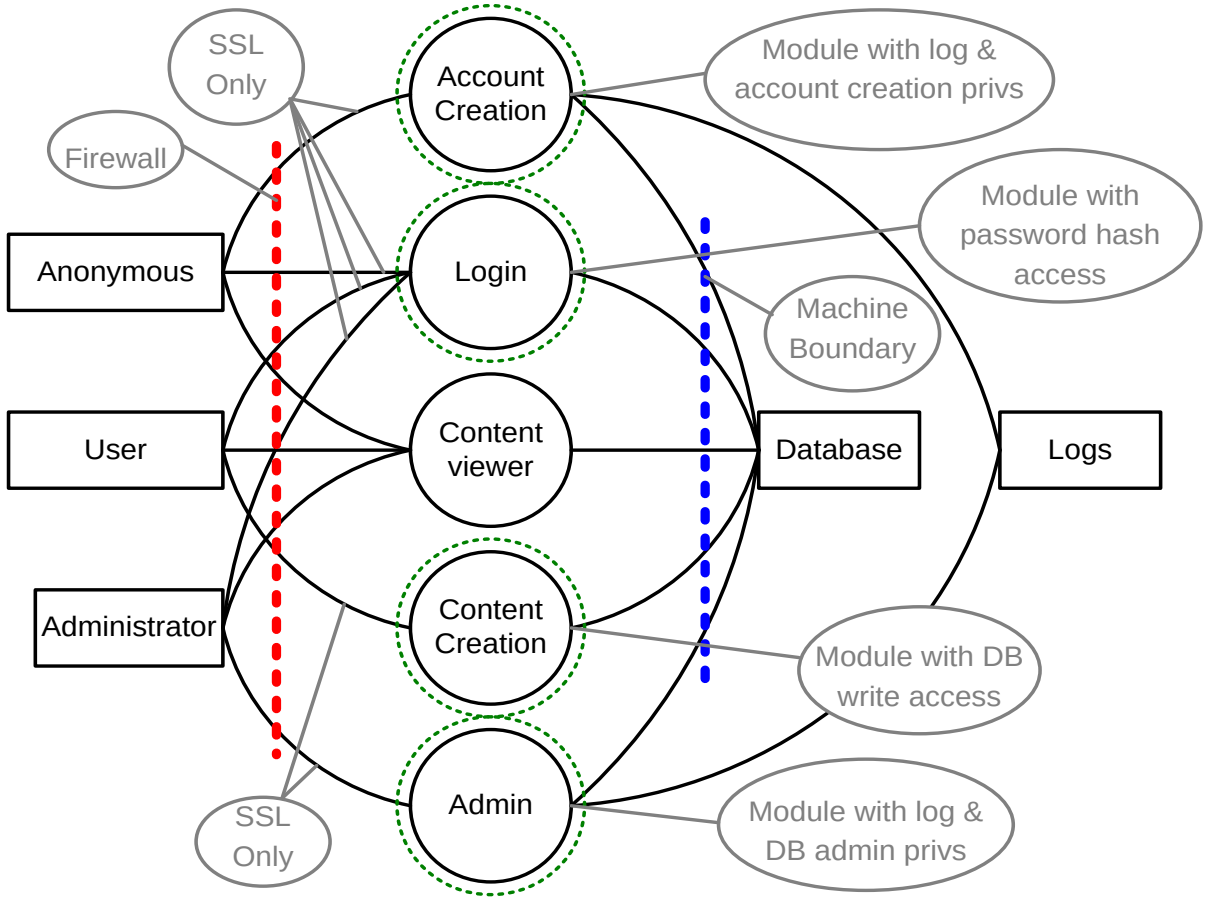
Resource: https://kosh.nku.edu/~waldenj/classes/2006/spring/csc593/presentations/ThreatModeling.ppt

Trike Example: Data Flow Diagram Level 0



Resource: https://kosh.nku.edu/~waldenj/classes/2006/spring/csc593/presentations/ThreatModeling.ppt

Trike Example: Data Flow Diagram Level 1



Resource: https://kosh.nku.edu/~waldenj/classes/2006/spring/csc593/presentations/ThreatModeling.ppt

Attack Trees

Root node is a threat.

Subnodes are attacks to realize threat.

- Attacks may be re-used in other trees.

Hierarchical decomposition

- Until determine risk is acceptable or not.

Resource: https://kosh.nku.edu/~waldenj/classes/2006/spring/csc593/presentations/ThreatModeling.ppt

Trike Attack Tree Example

Resource: https://kosh.nku.edu/~waldenj/classes/2006/spring/csc593/presentations/ThreatModeling.ppt

Attack Graph

Encompasses all attacks vs system.

- Set of interlinked attack trees.

Auto-generation

- High-level attack skeleton.
- Attack libraries
  - Many sub-trees re-appear.
  - Attached to tagged technologies in DFD.
- Need security expertise for full tree.

Resource: https://kosh.nku.edu/~waldenj/classes/2006/spring/csc593/presentations/ThreatModeling.ppt

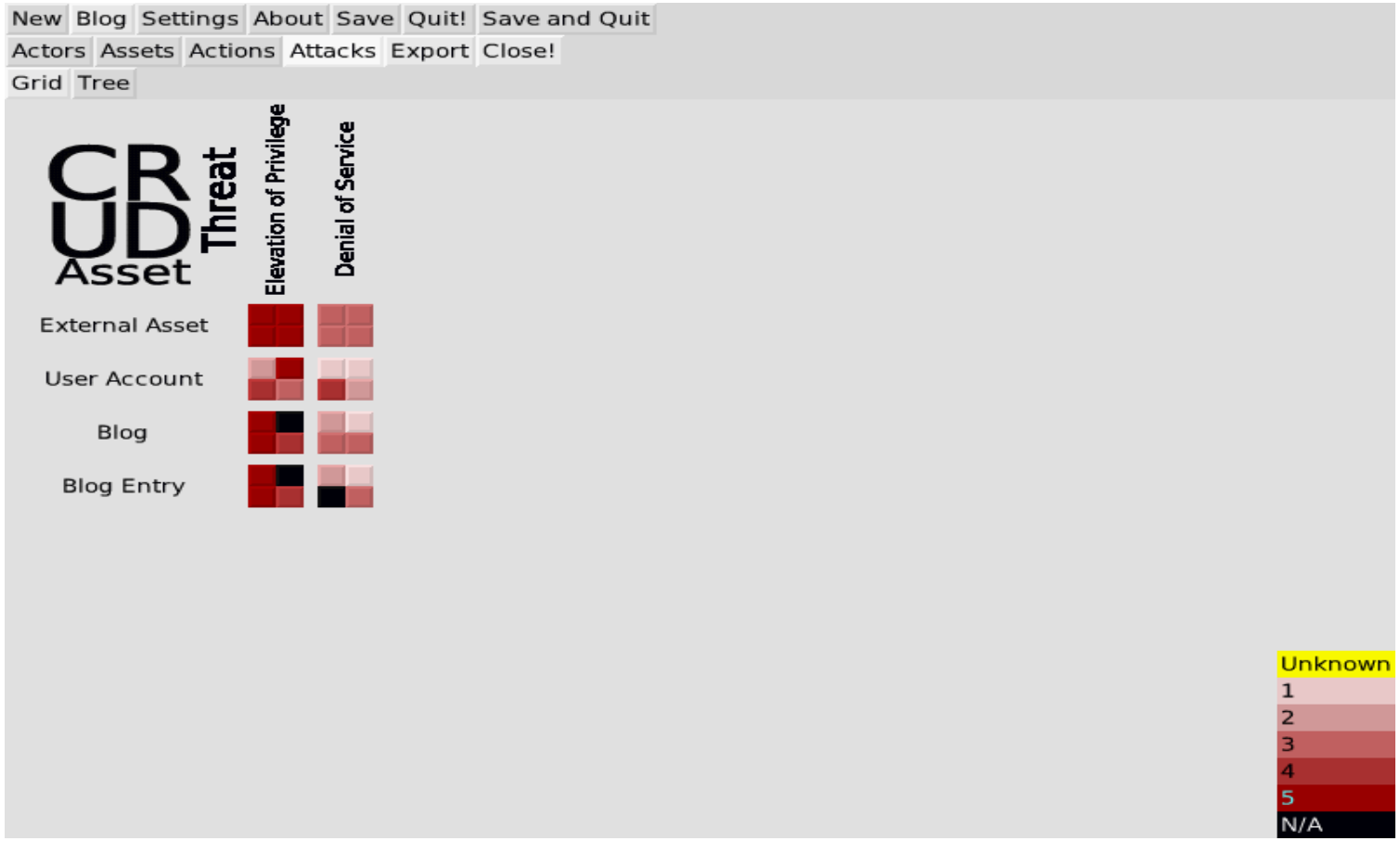
Risk Modeling

1. Business assigns values($) to Assets.
2. Rate Actions on each Asset.
   - 1-5 relative scale, with 5 being worst.
   - Ranked twice: denial, elevation
3. Assign each Actor a risk level 1-5.

Risk = Value of Asset * Action risk.

Resource: https://kosh.nku.edu/~waldenj/classes/2006/spring/csc593/presentations/ThreatModeling.ppt

Trike Threat Risk Grid



Resource: https://kosh.nku.edu/~waldenj/classes/2006/spring/csc593/presentations/ThreatModeling.ppt

Resources

https://beaglesecurity.com/blog/article/dast-in-github-actions.html