KubeCon | CloudNativeCon

North America 2023

CLOUD NATIVE SECURITY

OPEN AND SECURE:

Lessons Learned from TAG Security Assessments

Justin Cappos, New York University
Andrés Vega, Messier 42 (M42)

Introducing the “Open and Secure” Book

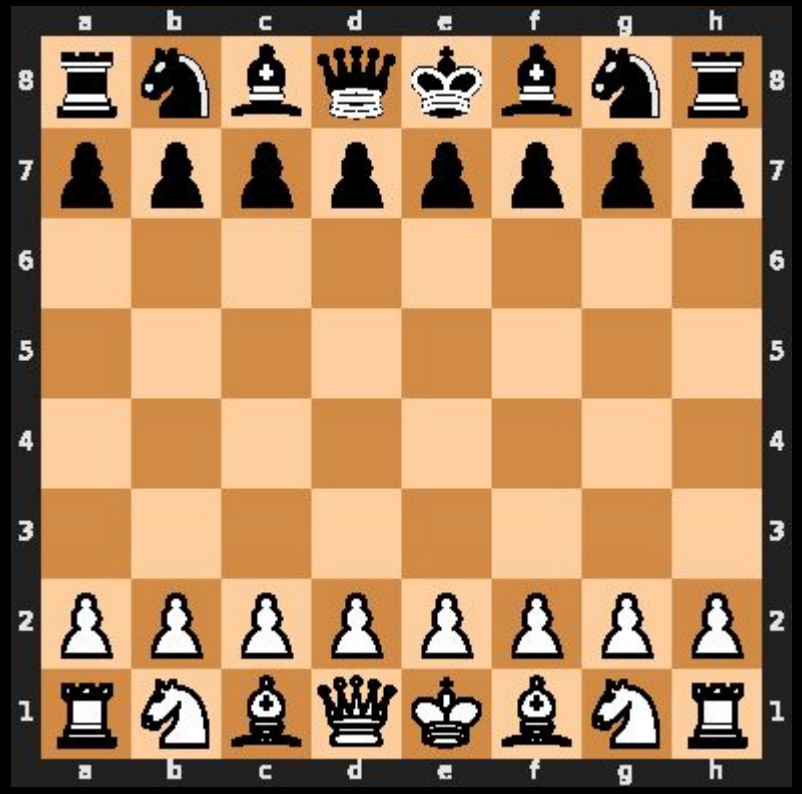
A comprehensive guide to applying threat modeling and enhancing the security of cloud native open source projects.

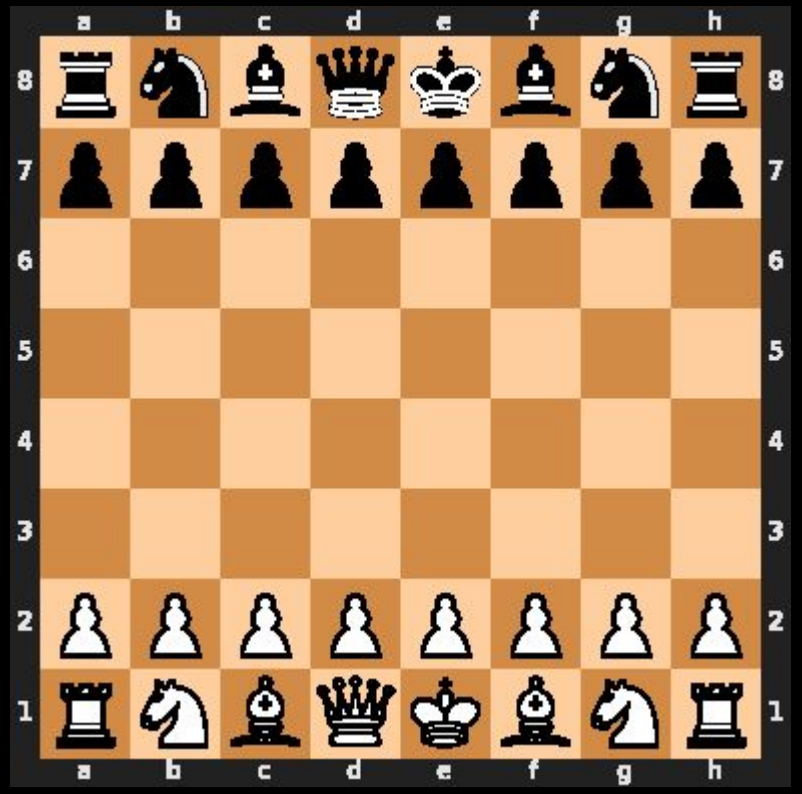
Specifically designed for project maintainers, this book offers insight into assessing and mitigating security risks starting by producing your own assessment.

- Real world examples
- Step-by-step guide on completing assessments
- Resources and tools
- Supporting a more secure ecosystem

Security is often misunderstood

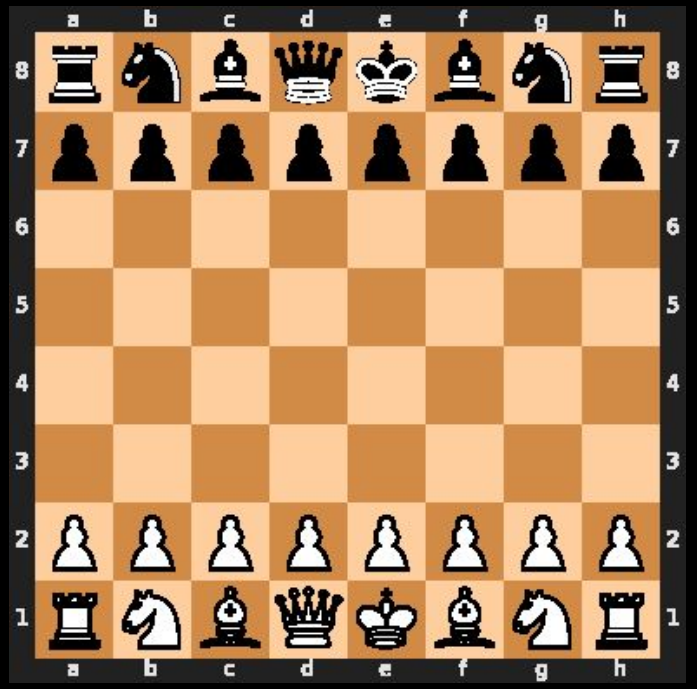
Failures (Unlike Threats) Are Random Events





A Strategy to Mitigate Random Events

And why combatting randomness can be predictable



"Furthest algorithm" is playing black

"Furthest algorithm"

If you can win, do that

else if you can move a pawn, move the furthest pawn you can

else move randomly

26 LOC (Python)

This loses ~1.2% of the time vs random, ties ~1.4%, and wins ~97.4% of the time

(Again, unlike threats)

Facing an Intelligent Adversary in Security

In security, the adversary isn’t predictable program, but a cunning human mind.

- Strategic and Calculated: Adversaries meticulously plan their attacks.
- Observant: They study defenses and probe for weaknesses.
- Adaptive: They can change their tactics in response to our defenses.

Intelligent Adversary: Dylan

Stats at a glance:

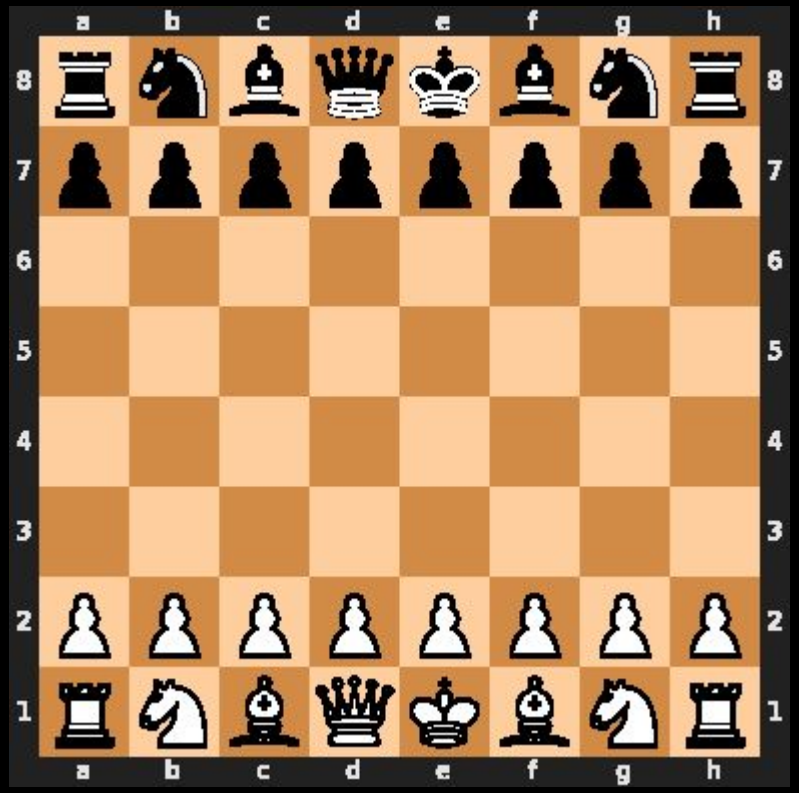
- Age: 5 years 9 months
- Height: 3’11” Weight: 42 lb
- Favorite food: Hamburgers

Chess Journey

- In just a few months, Dylan has grasped the chess fundamentals.
- Knows most chess moves.
- Is learning to anticipate his opponent’s strategies.

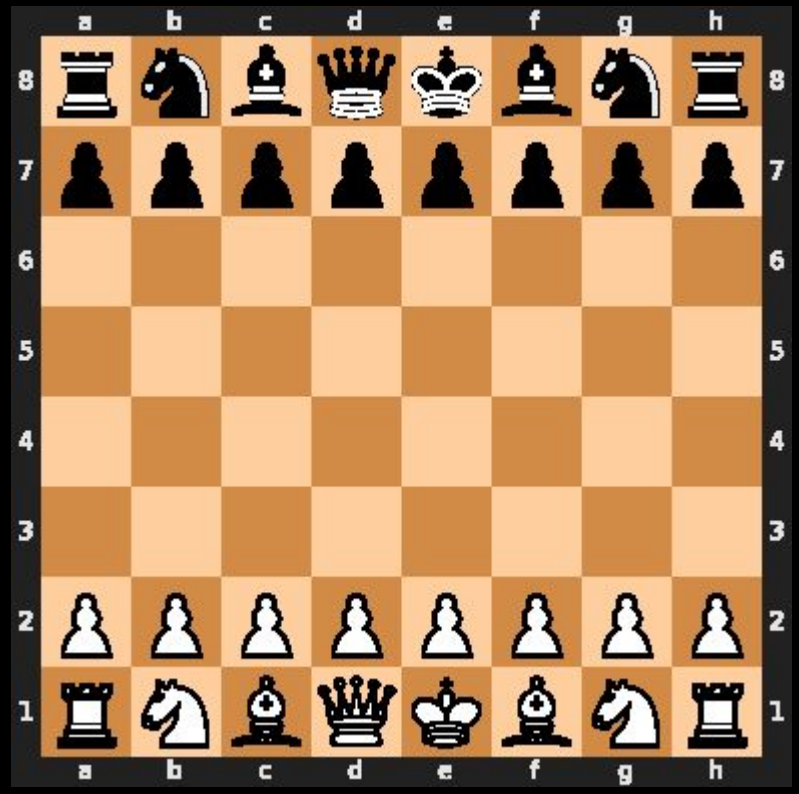
Dylan’s Strategy in Action

Emerging Talent vs The Algorithms



Dylan (white) vs Random

Game against program that mimics chaos



Dylan (white) vs Furthest

Game against program with a set rule

Security Assessments by Threat Modeling

Just as understanding the game is crucial in chess, threat modeling is about comprehending your system and the threat landscape.

- **Scenarios** (The Board): Understand your system as you'd understand the chessboard.
- **Actors** (The Pieces): Each piece on the board represents different roles within a system and potential threat actors.
- **Actions** (Their Moves): Predicting the offense and defense — Define how each role interacts with the others and how threat actors might exploit interactions.
- **Goals** (Win Conditions): Outline what success looks like in terms of defense and safety of your system.
- **Non-Goals** (What you aren't worried about): Acknowledge and define what is outside the scope of immediate concern, just as in chess where some exchanges are deemed acceptable losses for greater strategic advantage.

Scenarios

What are the intended use cases of a system and where should it be used?

A reliable car does not break down

A reliable car is not a submarine

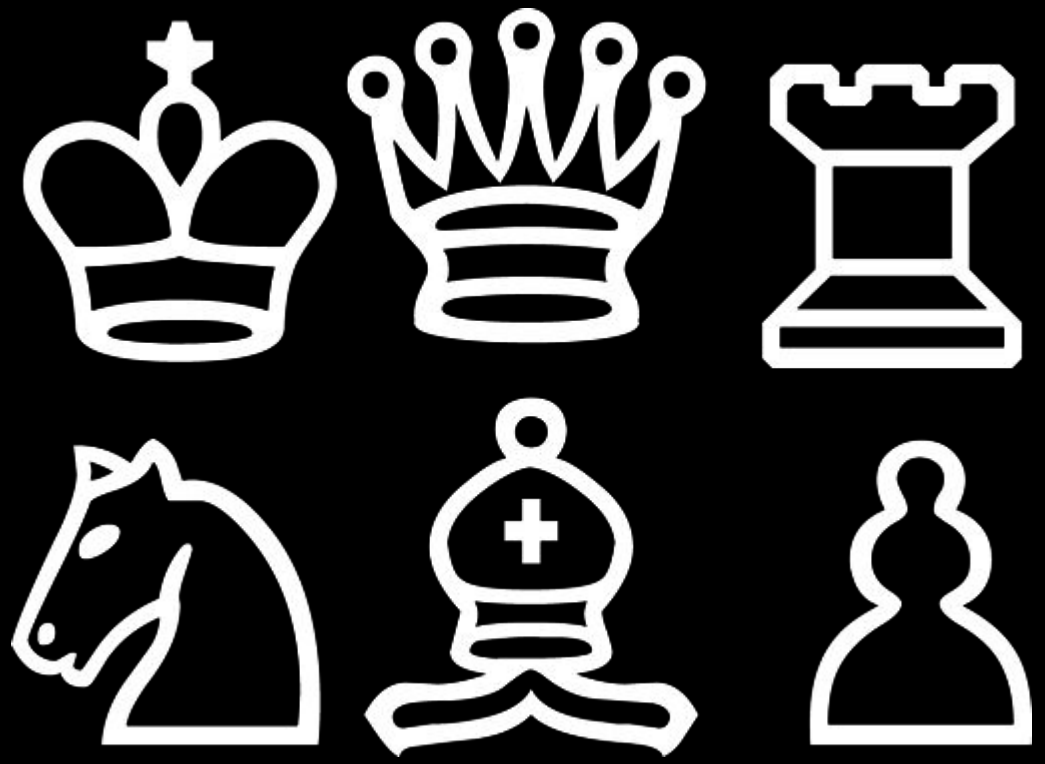
"Can you think of an 'out-of-scenario' use for a system you work with? What would be the 'submarine' equivalent for it?"

How do these 'unintended scenarios' inform your threat modeling?

Actors

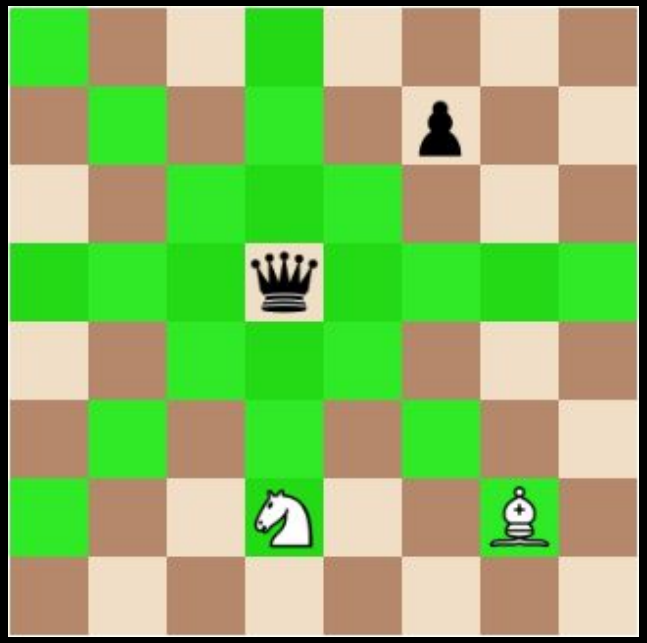
Actors are entities with the capability to interact with or impact the system — each with unique roles, capabilities, and objectives.

- **Knowing the potential of each 'piece' helps in planning defenses.**
- **Misjudging the importance of a 'pawn' actor can lead to oversight in security defenses.**
- **Actors are not limited to malicious attackers but can also include internal users, third-party vendors, or even automated processes within the system.**



Assessing Actions

- **Actions in threat modeling reflect the possible strategies and tactics actors might employ to achieve their goals.**
- **Possible actions can include "accessing data," "exploiting a vulnerability," or "lateral movement"**
- **The queen’s flexibility and range are a good exemplification a well-resourced and motivated actor could take.**



Note: In real life, unlike the predictable paths in chess, actors actions can be less predictable.

Identifying actions can help in prioritizing which 'moves' to defend against by understanding the attack vectors.

Distinguishing Goals

- A clear goal in chess is to checkmate the opponent's king—this is a direct parallel to defining primary security objectives.
- Just as in chess, where the ultimate aim is a strategic win, threat modeling focuses on protecting the most critical assets and functions from potential threats.

Telling Apart Non-Goals

Non-goals are those outcomes that, while perhaps interesting or aesthetically pleasing, do not contribute to the win condition, much like making a 'pretty pattern' or setting up an “art installation” with chess pieces is not the aim of the game.

From Theory to Practice: Evaluating Security

We've outlined the chessboard and the pieces. Now, how do you apply this to the security analysis of your software?

There are two primary techniques to assess and ensure rigor in security:

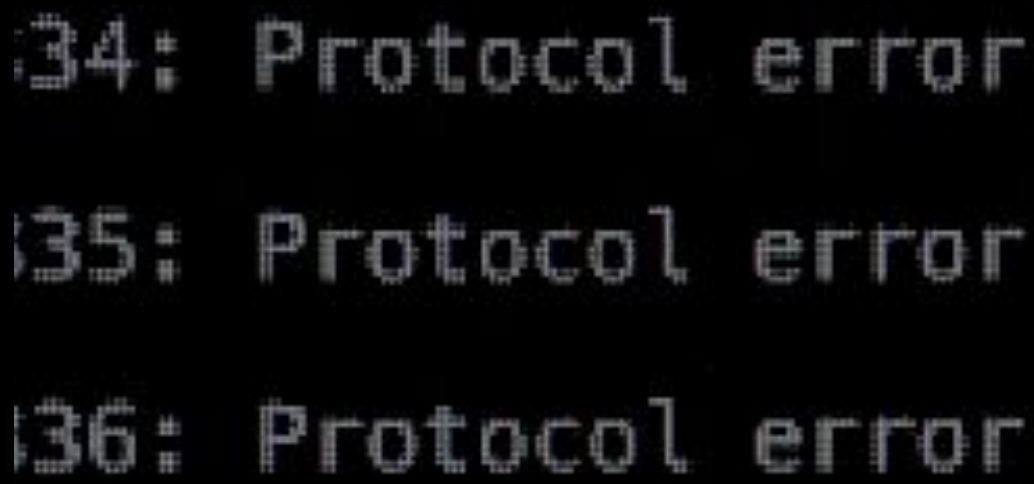
Security Audit

Security Assessment

Understanding Security Audits

Simulate an adversary by attacking the system to uncover vulnerabilities:

- Detect and document: Rigorously find and report flaws.
- Try to break in as a real-world attacker might, to test the resilience of security measures.



Delving into Security Assessment

High level evaluation of your security architecture, offering strategic insights to realign intent with implementation, offering insight into both problems and improvements to prioritize.

In-depth examination of:

- What is the system designed to do?
- What are the design principles?
- How well does implementation align with those principles?
- Are accepted/recommended practices used?

Assessment vs Audit

Security Audit	Security Assessment
Finds and identifies demonstrable/concrete issues	Uncovers design/policy gaps
Directly related to specific tools and codebase	Offers long-term, strategic insights
Results will vary with different tools and between different audits	Provides general, enduring guidance
—--	May require more effort to explain value to management

Follow a Methodology to Enumerate Attacks and Goals

Spoofing identity

- Illegally accessing and then using another user's authentication information

Tampering with data

- Malicious modification
- Unauthorized changes

Repudiation

- Deny performing a malicious action
- Non-repudiation refers to the ability of a system to counter repudiation threats

Information disclosure

- Exposure of information to individuals not supposed to access
- Denial of service

Deny service to valid users

- Threats to system availability and reliability
- Elevation of privilege

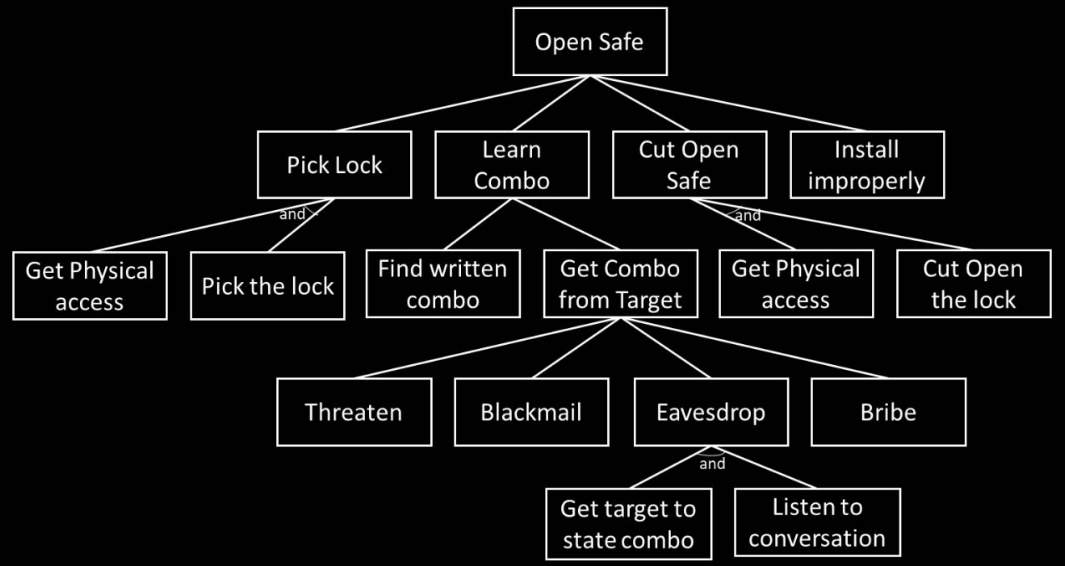
Compromised user gains privileged access

- Unprivileged one the system
- Effectively penetrated and become part of the trusted system

Unraveling Attack Graphs

Attack graphs are a visual representation mapping out every conceivable threat vector and the associated steps an attacker would employ to reach their end goal.

- **Visualize Threat Pathways:** By illustrating the routes an attacker can take you can help picture complex attack scenarios.
- **Identify Security Weaknesses:** Expose vulnerabilities and the paths that are least resistant to an attack, guiding where defenses can be bolstered.
- **Prioritize Risk Mitigation:** By seeing the complete picture, you can prioritize which vulnerabilities to address first based on the likelihood and impact of each path.
- **Enhance Communication:** Common language and vocabulary for all stakeholders.



Root Node: The end goal of the attacker, such as 'Open Safe', is the starting point from which all potential attack paths diverge.

Branches: Each branch off the root node represents a different method of attack, such as 'Pick Lock' or 'Learn Combo'. These are the high-level strategies an attacker might employ.

Leaves: Individual actions required to execute attack chain, providing a step-by-step breakdown of each attack method.

Interdependencies: Some paths might intersect, showing that certain steps can be common across different methods, suggesting points of control that could mitigate multiple threats.

Thinking About Impact

Exceedingly rare events aren’t usually important even if impact if high

Meteor destroys Earth

Exceedingly low impact events aren’t worth worrying about

Taking more than one free candy

Calculating Expected Impact and Damage

Expected Damage ~= risk * impact

General rule:

Focus on plausible scenarios over extremely rare catastrophes or inconsequential issues, prioritizing broader, far-reaching concerns that can affect the whole ecosystem around your project:.

Example: A critical vulnerability in the project's code leading to widespread data breaches.

Ensuring Comprehensive Coverage

Cloud native systems are highly interconnected with other systems and may have many actors -- an attacker can often move between actors with an attack.

Assessing Interconnected Risks: An attacker's ability to exploit one actor can lead to compromising others.

Critical Isolation: Implementing and maintaining strict isolation between different actors (servers, agents, containers) is crucial in mitigating the spread of an attack.

Visualizing Trust Boundaries: An attack matrix like the one from CNCF TAG Security Assessment helps in visualizing trust boundaries and potential paths an attacker might exploit, and include details on mitigation and the respective score for a certain vulnerability or threat, with notes on the current state and potential risks.

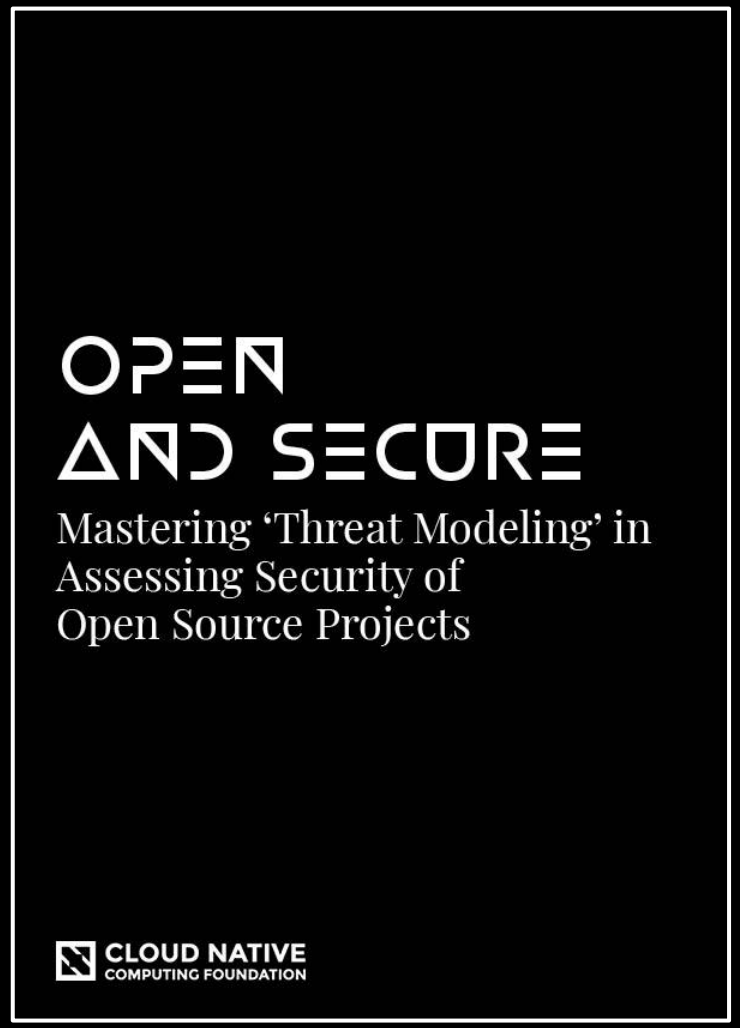
	Server	Agent	Container (same node)	Container (diff node)
Victim Server	N/A: there is one	**CSRVuln (Go):** A buffer overflow in Golang CSR parsing code could lead to remote code execution. Score: 0.448 **X509Vuln (Go):** A buffer overflow in Golang X.509 parsing code could lead to remote code execution, since mTLS client certificates must be parsed for authentication. Score: 0.082 **PROTO (Go):** A buffer overflow in Golang ProtoBuf or TLS implementations could lead to remote code execution. Score: 0.467	**CSRVuln (Go):** A buffer overflow in Golang CSR parsing code could lead to remote code execution. Score: 0.448 **X509Vuln (Go):** A buffer overflow in Golang X.509 parsing code could lead to remote code execution, since mTLS client certificates must be parsed for authentication. Score: 0.082 **PROTO (Go):** A buffer overflow in Golang ProtoBuf or TLS implementations could lead to remote code execution. Score: 0.467	**CSRVuln (Go):** A buffer overflow in Golang CSR parsing code could lead to remote code execution. Score: 0.448 **X509Vuln (Go):** A buffer overflow in Golang X.509 parsing code could lead to remote code execution, since mTLS client certificates must be parsed for authentication. Score: 0.082 **PROTO (Go):** A buffer overflow in Golang ProtoBuf or TLS implementations could lead to remote code execution. Score: 0.467
Victim Agent	**X509Vuln (Go):** A buffer overflow in Golang X.509 im… executio… Score: 0… **PROTO** … ProtoBu… remote c… Score: 0…			
Victim Container (same node)	**X509Vul…** X.509 im… executio… Score: 0…			
Victim Container (diff node)	**X509Vul…** X.509 im… executio… Score: 0…			

	Server	Agent	Container (same node)	Container (diff node)
Victim Server	N/A: There is only one server	Mitigated: The server has validation in place to prevent it from signing CSRs for SPIFFE IDs that are not registered to a particular agent. Furthermore, there is validation to prevent an operator from erroneously registering the server's SPIFFE ID. Agents always validate the server's SPIFFE ID when connecting to it. Score: 0.11	Mitigated: There is validation to prevent an operator from erroneously registering the server's SPIFFE ID. Score: 0.11	Mitigated: There is validation to prevent an operator from erroneously registering the server's SPIFFE ID. Score: 0.11
Victim Agent	**NONE:** The server has the signing keys and can issue new identities at will Score: 57.5	Mitigated: The server has validation in place to prevent it from signing CSRs for SPIFFE IDs that are not registered to a particular agent. Furthermore, there is validation to prevent an operator from erroneously registering a SPIFFE ID representing an agent. Score: 0.115	**ESCAPE:** If a container escape and privilege escalation can be performed, it is possible to read the agent's key from memory. Score: 0.63	Mitigated: There is validation to prevent an operator from erroneously registering the agent's SPIFFE ID. Score: 0.115
Victim Container (same node)	**NONE:** The server has the signing keys and can issue new identities at will Score: 5.5	**NONE:** Agent controls the keys and certificates for all containers authorized to run on it. Score: 5.5	**ESCAPE:** If a container escape and privilege escalation can be performed, it is possible to read neighboring container's keys from memory. Score: 0.231	**ESCAPE:** A container can be authorized to run on multiple nodes. If the container in question is authorized to run on the node with the evil container, then the evil container can obtain a certificate representing the victim container by reading keys from the memory of the local agent. Score: 0.525
Victim Container (diff node)	**NONE:** The server has the signing keys and can issue new identities at will Score: 12.5	**NONE:** A container can be authorized to run on multiple nodes. If the container in question is authorized to run on the node with the evil agent, then the evil agent can obtain a certificate representing the container. Score: 12.5 NOTE: This condition only occurs under certain configurations	**ESCAPE:** A container can be authorized to run on multiple nodes. If the container in question is authorized to run on the node with the evil container, then the evil container can obtain a certificate representing the victim container by reading keys from the memory of the local agent. **Score: 0.525**	**ESCAPE:** A container can be authorized to run on multiple nodes. If the container in question is authorized to run on the node with the evil container, then the evil container can obtain a certificate representing the victim container by reading keys from the memory of the local agent. Score: 0.525

Mitigating Risk with Expert Insights

Refer to the insightful book on assessments for a resource to effective security assessments, providing you with useful context to navigate complex security landscapes.

Draw upon the collective wisdom and diverse viewpoints from: Justin Cappos, Ragashree Shekar, Andres Vega, Andrew Martin, Ash Narkar, Dan Lorenc, Jack Kelly, and Marco De Benedictis



Collaborative Security Assessment Process

Project-Led Preliminary Review:

Process starts with a self-assessment, leveraging the frameworks and considerations discussed earlier to analyze their security posture.

Facilitator-Driven Committee Formation:

An Assessment Facilitator curates a team of knowledgeable reviewers, ensuring a broad and informed examination.

Guidance and Refinement:

The committee provides feedback, helping projects fine-tune their security assessment.

Consensus on Security Insights:

The collaborative effort culminates in a joint review, informing both the project, the TOC, and the broader community with collective insights.

Collaborative Security Assessment Process

Project-Led Preliminary Review:

Process starts with a self-assessment, leveraging the frameworks and considerations discussed earlier to analyze their security posture. <- -- Most projects fail here!

Facilitator-Driven Committee Formation:

An Assessment Facilitator curates a team of knowledgeable reviewers, ensuring a broad and informed examination.

Guidance and Refinement:

The committee provides feedback, helping projects fine-tune their security assessment.

Consensus on Security Insights:

The collaborative effort culminates in a joint review, informing both the project, the TOC, and the broader community with collective insights.

TAG Security Pals: Guiding Assessments

A supportive outside team aiding in the initial draft of project self-assessments.

Process Overview: A structured 5-stage method to streamline the assessment.

1: **Preparation:** - Quick Setup (<1 day)
2: **Analysis:** Understand the Project Landscape (~1-2 days)
3: **Drafting:** Craft initial draft of self-assessment (3-5 days)
4: **Collaboration:** Iteration and refinement with the project (~2-3 days)
5: **Finalization:** Completing the self-assessment (1 day)

Pilot program: ~110 NYU students working in groups. Launching now with 27 projects, encompassing most incubating and graduated CNCF projects!

Becoming a TAG Security Reviewer

Start by Engaging:
- Joining TAG Security meetings is encouraged but not mandatory.

Show Your Interest:
- Sign up to participate as an observer for an upcoming assessment.

Active Participation:
- Contribute through insightful queries and critical observations.
- Expect a commitment of approximately 2-3 days spread across two weeks.

Advancement Path:
- Proven reviewers have the opportunity to lead future assessments.

CLOUD NATIVE SECURITY

Maximizing Assessment Impact

Tell us what you think:

- **Your opinion on the assessment structure and practicality are important to us.**

We seek tighter feedback loops:

- **While project maintainers often express their appreciation, we rarely hear from end users. On the occasion that we do, they call out assessments are good basis for internal team resources like hardening guides.**

Share your expertise:

- **Engage with us. Your participation strengthens the resiliency of the ecosystem. Starting to apply the learned concepts in your systems or to contribute to the security community, can be impactful.**

Collaborative Security Milestones


Buildpacks.io

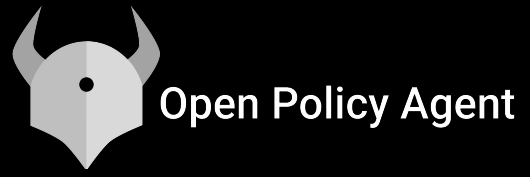

Cloud Custodian


HARBOR


in-toto


KEYCLOAK


Kyverno


Open Policy Agent


PIXIE


spiffe


SPIRE

A Special Thanks to Our Dedicated Maintainers and Contributors for Achieving These Security Milestones Together!

Please scan the QR Code above
to leave feedback on this session