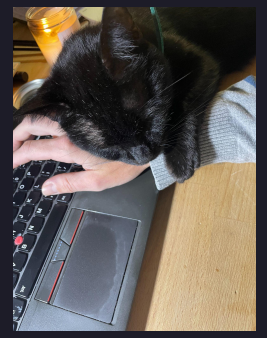
Listening for effective threat modeling

Kelly Kaoudis, Trail of Bits

May 31, 2025

Nice to meet you

- Senior security engineer at Trail of Bits from Boulder, CO
- Trail does both industry security engagements, and also builds DARPA-funded security research tools
- Research: systems security, dynamic program analysis
- Appsec: secure code review, infrastructure review, design review, threat modeling!
- Threat modeling: TRAIL (Threat and Risk Analysis Informed Life Cycle)



What we won’t talk about

There is no client-confidential information in this talk.
We’ve seen each of these scenarios more than once!

What we *will* talk about

- Negative initial results turned into positive outcomes!
- Design-level findings involving process, policy, and the SDLC
- Bringing the client with us through the discovery process
- Things that I have learned are useful for this, with examples
  - Reporting findings however they will be best received
  - Leveraging learner's mindset
  - Documenting threats resulting from oversights, unquestionable assumptions, and self-censoring

1. Listen for the team's jargon and norms.

Identifying common language

As a security champion on a software development team...

Identifying common language

As a security champion on a software development team...

What *didn't* work: inserting identified security risks as business priorities (business priorities only came from management; in this particular culture, I was overstepping)

Identifying common language

As a security champion on a software development team...

What *didn't* work: inserting identified security risks as business priorities

What *didn't* work: talking to management (management set the overall direction of work, not local tactical priorities)

Identifying common language

As a security champion on a software development team...

What *didn't* work: inserting identified security risks as business priorities

What *didn't* work: talking to management

What *didn't* work: not translating from appsec-ese (local team didn't know appsec jargon enough to even understand why they should care)

Identifying common language

As a security champion on a software development team...

What worked:

- Presenting a few risks during sprint planning

Identifying common language

As a security champion on a software development team...

What worked:

- Presenting a few risks during sprint planning
- Phrasing risks and security asks as *performance and correctness improvements*

2. Listen for our own assumptions.

Learner's mindset

As a security engineer...

- Client wanted to know what weak security controls existed, and what controls were missing

Learner's mindset

As a security engineer...

- Example: developers could ssh to prod, *but* some security controls existed (mainly access audit logging)

Learner's mindset

As a security engineer...

- Developers could ssh to prod, but some security controls existed
- I caught myself thinking "something is better than nothing"

Learner's mindset

As a security engineer...

- Developers could ssh to prod, but some security controls existed
- I caught myself thinking "something is better than nothing"

Learner's mindset

As a security engineer...

- Client wanted to know what weak security controls existed, and what controls were missing
- Developers could ssh to prod, but some security controls existed
- I caught myself thinking "something is better than nothing"
- Client's security maturity in other areas meant this was a weak point for the system

Learner’s mindset

Assumption I had made: access logging means this is not great, but ok

What I asked instead: what is actually logged?

Learned that ssh connection origins, timestamps, and system users were logged, *but actions taken on the destination host were not*

Learner’s mindset

Assumption I had made: this access must be limited since it’s for debugging only

What I asked instead: what privileges do devs have on the host?

Learned that *devs all accessed the host as root*

Learner's mindset

Assumption I had made: this access must be limited since it's for debugging only

What I asked instead: what privileges do devs have on the host?

Learned that devs all accessed the host as root

Also learned nothing prevented devs from using `kubectl exec` in prod

Also! learned the entire system would halt if the host the devs had access to went down

Learner's mindset

Assumption I had made: this access must be limited since it's for debugging only

What I asked instead: what privileges do devs have on the host?

Learned that *devs all accessed the host as root*

Also learned nothing prevented devs from using `kubectl exec` in prod

Also! learned the entire system would halt if the host the devs had access to went down

3. Listen for the undocumented things “everyone knows”.

Identifying implicits

- Not written down so can’t learn it directly from the docs
- Clients usually won’t think to tell us directly
- …but key to why the system works

Example

- Another client's system relied on Rego (Open Policy Agent) policies for access control
- Got reasonably pointed to third-party (OPA/Rego) doc
- **What *didn't* work**: asking about how the security control worked within the system and could fail; how the system *itself* worked

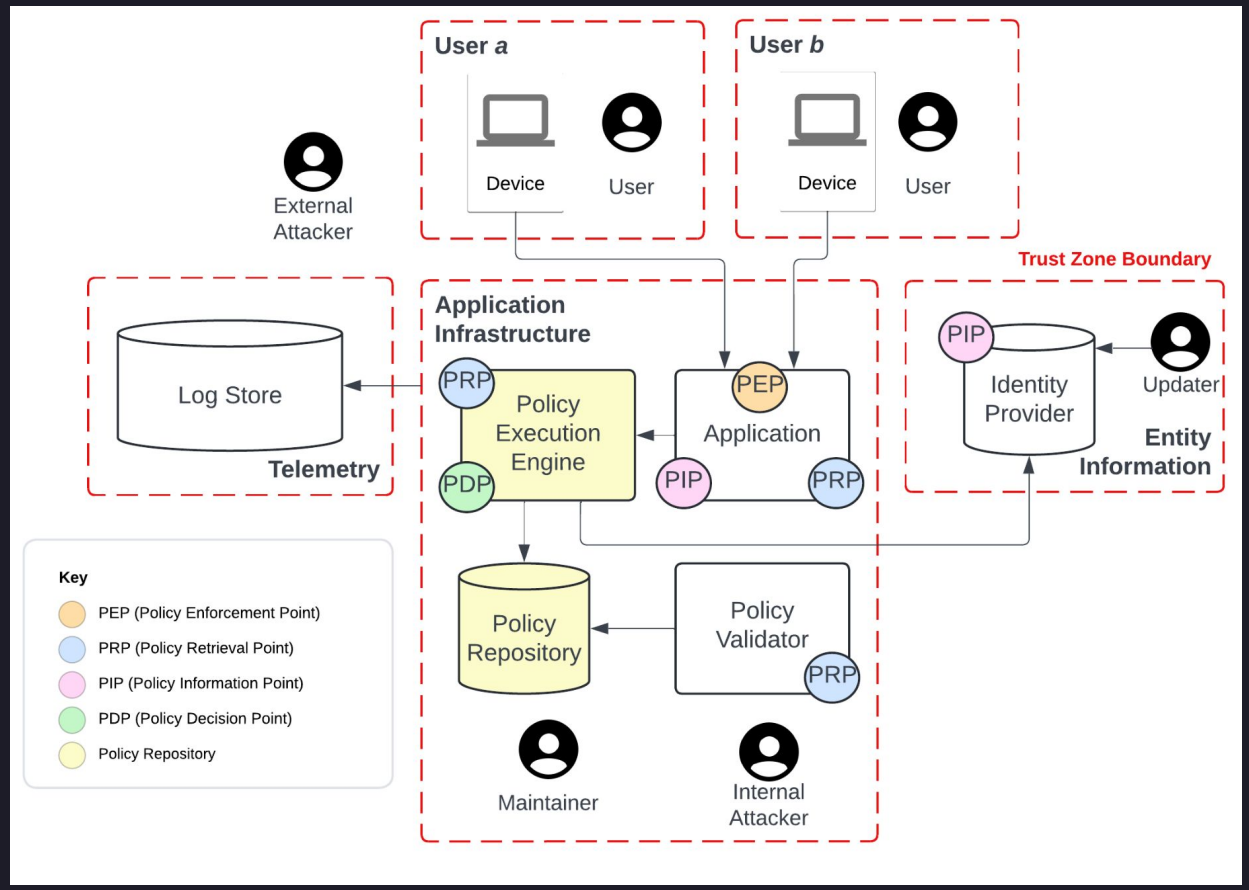
Example

- **What *didn't* work**: asking about how the security control worked within the system and could fail; how the system *itself* worked
- **What *did* work**
  - Eventually asked "what *is* Rego?"
  - Rego policies in this case compiled to WebAssembly and were evaluated in a WASM runtime
  - Not all Rego built-ins are supported in WASM nor in every OPA WASM SDK
  - An error may be thrown if an unsupported built-in is called, but the policy evaluation result set may also be null or empty

Example

- **What *did* work**
  - Asked what Rego was in client's context
  - **Implicit**: Rego policies compiled to WebAssembly and evaluated in a WASM runtime
  - **Implicit:** Not all Rego built-ins are supported in WASM
  - **Oversight**: If a policy evaluated in OPA WASM uses an unsupported built-in, *IAM policy evaluation may fail open*
  - The host app or the OPA SDK in use has to handle the error

A policy engine deployment might look like this



link to report

4. Listen for unquestionable assumptions.

Identifying the unquestionable

- Something that is taken for granted
- If nobody questions it, how do we know if it's true?

Example

- Adding roles to a client's RBAC system happened through a webpage that automatically created and filed a ticket
- When the ticket was closed, automation added the role
- Access control? Stock answer: only management could file tickets

Example

- Adding roles to a client's RBAC system happened through a webpage that automatically created and filed a ticket
- When the ticket was closed, automation added the role
- What access control?
- **What *didn't* work**: asking what accounts should have what permissions and when (got the stock answer)

Example

- Adding roles to a client's RBAC system happened through a webpage that automatically created and filed a ticket
- When the ticket was closed, automation added the role
- What access control?
- **What *didn't* work**: asking what accounts should have what permissions and when (got the stock answer)
- **What *didn't* work**: asking if anyone ever reviewed existing group membership (stock answer! again!)

Example

- Adding roles to a client's RBAC system happened through a webpage that automatically created and filed a ticket
- When the ticket was closed, automation added the role
- What access control?
- **What *didn't* work**: asking what accounts should have what permissions and when (got the stock answer)
- **What *didn't* work**: asking if anyone ever reviewed existing group membership (stock answer! again!)
- **What *almost* worked**: asking if tickets could ever close without review (stock answer)

Example

- Adding roles to a client's RBAC system happened through a webpage that automatically created and filed a ticket
- When the ticket was closed, automation added the role
- What access control?
- **Anything else that should have been asked or verified here??**

Identifying the unquestionable

- What access control?
- **What *didn't* work**: asking what accounts should have what permissions and when
- **What *didn't* work**: asking if anyone reviewed group membership
- **What *almost* worked**: asking if tickets could ever close without review
- **What worked**: asking how the management relationship was encoded in the RBAC system (got client to think about why)

5. Listen for self-censoring.

Identifying self-censoring

- Client might think there's no point in saying it as "nobody will listen", or it's taboo

Identifying self-censoring

- Client might think there's no point in saying it as "nobody* will listen", or it's taboo

*I will listen ;)

What happens when the client *stops* self censoring?

- Clients are people, and they might face real consequences
- Sometimes things in this category are important context, but should not be written down or recorded, like new feature details, or discussion of staff turnover or salary
- If it isn't related to security or privacy, it stays off the record!
- If it *is* related to the system's security: fair game, but take extra caution to stay within scope and stick to facts
- If we learn something is insecure and *don't* report it, also might be real consequences *for us*

Identifying self-censoring

- Client might think there's no point in saying it as "nobody will listen", or it's taboo
- ...sometimes, everyone on the client's side already knows whatever *it* is
- Client gets overly helpful about something else, goes on tangents, tries to pass the buck ("oh, someone else might know...")

Example

- Reviewing a system that used an LLM to process user data sourced from a connected third party
- Cleanup of a user's data after they stopped using the system?
- Deletion of third-party user-specific access tokens?
- Deletion of any derived data?
- Responsible parties who owned data cleanup?
- Was the LLM used to process data for all users simultaneously?
- Were user requests recycled as model training data?

Identifying self-censoring

- **What didn't work**: asking about how the client respected GDPR or the California Consumer Privacy Act (CCPA)
- **What didn't work**: asking about data retention timelines
- **What didn't work**: asking what data could, or would be, retained and when
- **What didn't work**: asking about how long third-party access would be retained
- **What didn't work**: asking for a record of derived data
- **What didn't work**: asking who owned the data cleanup features
- **What didn't work**: asking about third-party access revocation

...

Kelly Kaoudis, Trail of Bits, May 31, 2025

Example

- Cleanup of a user's data after they stopped using the system?
- Deletion of third-party user-specific access tokens?
- Deletion of any derived data?
- Responsible parties who owned data cleanup?
- Was the LLM used to process data for all users simultaneously?
- Were user requests recycled as model training data?

(short answer: exactly what you think)

THREAT MODCON 2025 BARCELONA

Takeaways

Listening for effective threat modeling

- **Listen for the team's jargon and norms**; use them as the basis of a common language with which to frame threats and findings
- **Listen for our own assumptions**; invert them into questions to check our understanding and course-correct
- **Listen for the undocumented things "everyone knows"**; interesting threats and findings could result
- **Listen for unquestionable assumptions**; insecure or inadequately private practices, policy, or design might underlie them
- **Listen for self-censoring**; might lead to more findings, but use discretion

thank you for listening! p.s., trailofbits.com/careers - we are hiring :)

Kelly Kaoudis kelly.kaoudis@trailofbits.com, Trail of Bits, May 31, 2025