



Threat Modeling Meets Model Training

Web App Security Skills for AI

Breanne Boland, Product Security Engineer, Gusto



whoami

- I live in Brooklyn (previously Oakland and Seattle)
- I do product security things
- I write novels and make stained glass pieces
- I ~~here~~ tend cats
- I give advice about secure implementations of AI (among other tasks at work)

The current(ish) state of AI

AI

The secret sauce!
On everything!
So just... sauce!

What it does well

Identifying patterns
Accessibility, sometimes

Problems

AI on AI on AI helps
No single measure
fixes everything

So new, so special?

Special problems, but
familiar ones too

AI output

You cannot innately
trust it! Not ever!

Humans made AI

So AI has human
problems: bias,
inaccuracies, etc.

Vocabulary lesson

- AI (artificial intelligence) vs. LLM (large language models)
- Model and model training
- Fine tuning
- RAG (retrieval-augmented generation)
- Prompts (of engineering and injection fame)
- Hugging Face



Web application security concepts that apply to AI too

This is the longest part
of the talk! For reasons!

XSS! And other “please don’t put code there” issues

- It’s always ~~DNS~~ XSS
- Do user queries get added to the DOM?
- Does input get stored? (It generally should in some form.)
- A lot of the answer: good old sanitization, escaping, and encoding
 - Works on text added to the DOM and stored text too!
 - Add it to input and output! Spare no text!
- Are you *really sure* the LLM won’t include code in responses? *Ever?*
 - Keep asking!

Authn, authz, and all things access control

- Spoiler: access control is very very hard with AI (like it's easy normally 😭)
- Who's allowed to access your AI feature? Who can run up your bill?
- What's your AI feature allowed to access? How is it controlled?
 - It's... complicated. Best option: layer methods, like ABAC, prompt elements, user context
- Who is your AI feature acting as when it accesses resources?

State-changing operations

- One way to keep AI on the desired path is by only allowing it access to a narrow slice of API endpoints
- Another is confining it to the current user's context
- Human verification for state-changing operations
- Just directing the user to the page needed to do what they want

Data (of course)

Only provide it what it needs. It can't leak what you don't give it.

- One option: use prompt engineering to try to keep certain data from being submitted or stored.
- If sensitive data is part of the training data, without several layers of guardrails, it's always possible that it will leak what you provide.

Without guardrails, testing, and other measures, **there is no way to ensure that what's put into an LLM will not come out**

✨ There is no
guaranteed way to
ensure that what's put
into a model won't come
out ✨

Data (of course)

Only provide it what it needs. It can't leak what you don't give it.

- One option: use prompt engineering to try to keep certain data from being submitted or stored.
- If sensitive data is part of the training data, without several layers of guardrails, it's always possible that it will leak what you provide.

Without guardrails, testing, and other measures, **there is no way to ensure that what's put into an LLM will not come out**


Users will *always* put data you don't expect into places you don't want or expect it to be, so be ready!

Where does your LLM live?

- If you can, host the model yourself (not always possible, though)
- If you must go third party (common, alas), make sure the vendor is reliable
- Third-party uptime problems happen
 - If your LLM isn't reachable, it's more difficult to roll back to a previous version (because they may not be comparable) or to enact another workaround. Do some disaster planning.
- Wherever it lives: if your company wants to use AI, they need to fund the resources to review, secure, and maintain it.

Scary yet alluring free software

- Traditional software libraries can seem opaque, but LLMs go further
- Helpful: Hugging Face model cards, ML-BOMs
- You have to ask questions, persist, and find out everything you can
- And still... keep an eye on the news. Things happen.
- It's a good idea to cultivate some light red-teaming skills to give things a poke if you don't have dedicated resources



What are concerns are particular to AI?

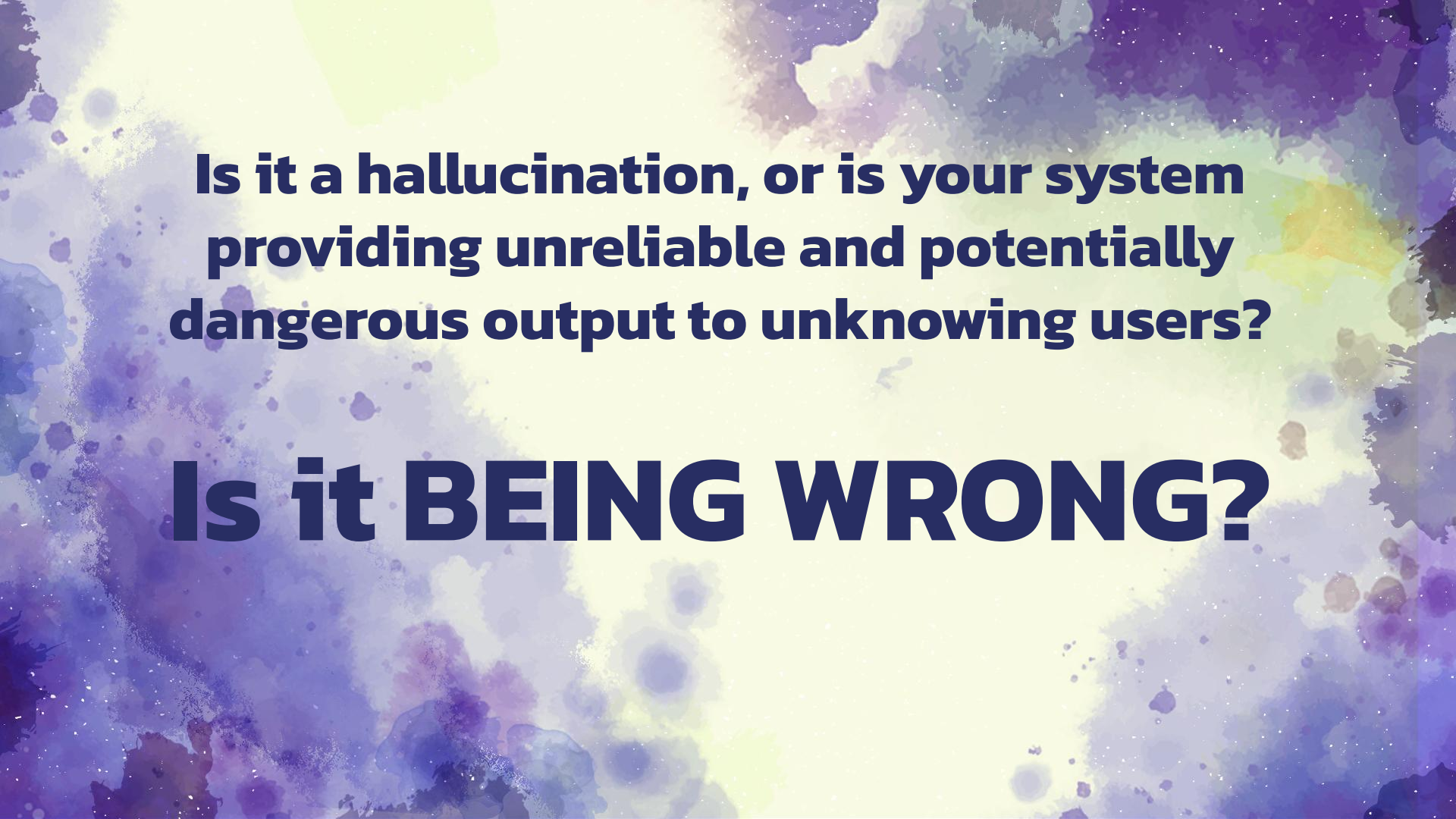
(Notice I didn't say *unique*)

Prompt injection

- Good old “ignore all previous instructions and...” (SQLi of AI)
- This is a really big field of study so if you’re interested in it, dig in
- Some types: direct, indirect, pretext, prompt leak
 - Sometimes, the more types you layer, the more success the attack
- One way to prevent: include a domain of expertise in your prompt, tell the LLM to ignore any questions outside of that
- Layered defenses are your best approach

Hallucinations, or: **BEING WRONG**

- With terms like this, ask who invented it and what their angle was
- People often react in response to the way you act when delivering the news
- “Hallucination” has a bit of whimsy to it. I don’t like using it when warning about inaccurate, unhelpful, or fully dangerous output from LLMs.
- Say it with me: **BEING WRONG**



**Is it a hallucination, or is your system
providing unreliable and potentially
dangerous output to unknowing users?**

Is it BEING WRONG?

Opaque training data and the perils of fixing it

- True for all models, even ones you fine tuned
- Using an existing model trained by someone else? It's full of mysteries
- No RAG? Answers might get stale
- Yes RAG? More uncertainty, more risks
- Result: we have to assume all LLM output is malicious, because it might be
- All sorts of stuff might be lurking in there :(sorry

Search for an author in LibGen

For instance, Stephen King, Min Jin Lee, or Fyodor Dostoevsky. Note that different spellings of the same name—“JK Rowling” versus “J. K. Rowling”—may produce different results.

 breanne boland

1 Results

Reinventing Cybersecurity

Jasmine Henry, Alison Gianotto, Coleen Shane, Tracy Bannon, Dr. Meg Layton, Breanne Boland, Angela Marafino, Latha Maripuri, Carlota Sage, Carla Sun

Unreliable output, or: an API would *never**

Tests on tests on tests

- One approach: write a unit test for every problem you've fixed, remove once they become obsolete
- AI can write a bunch for you, but the human has to prune and polish
- Your model and prompt should result in close focus, so your rules and tests should be focused too
- You have to refine your guardrails too. There are existing prompts or rules to start from, but you can't just plop it on. The problem and solution both require nuance.

*Well, not like *that*

Third-party LLMs learning from your users

- Most LLM providers scaled for business use offer zero-retention endpoints and other options to keep their products from training on your data or that of your customers
- You must select these options, particularly if your company handles legally protected data
- Even if you don't: do right by your users and protect them.

Moving on up: new models

- It's more complicated than changing API versions
- New models can work completely differently
- Time for: tests on tests on tests!
- Your fine tuning may also need to adapt
- But if you don't update models periodically and just keep fine tuning, performance suffers. Good luck!

One approach: A/B test with the new model, with a slow rollout, or just keep the old model available in case something goes awry with your cutover



What web
application
security concepts
don't apply?



Broadly... none.

A security issue is a security issue, and we already have tools that take us much of the way toward a thorough assessment for risk in new territory.

Like...

2017

A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring

2021

A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*

* From the Survey

The LLM top ten

Keep in mind the previous risks, and...

- Supply chain risks (we know these)
- Data and model poisoning (sounds a lot like injection, stored XSS)
- Improper output handling (more data spoiling)
- Excessive agency (sounds like authz and access control had an unholy child)
- System prompt leakage (sensitive data, anyone?)
- Vector and embedding weaknesses (leaks and data poisoning but with RAG)
- Misinformation (data, wreckt)
- Unbounded consumption (broken authentication, insecure design)

Takeaways

- Security practitioners have to stay on top of the tech that our engineering cousins want to use.
- LLMs are just technology, and we understand technology. Secure it like everything else.
- If your company wants to use AI, they need to fund the resources to review, secure, and maintain it.
- Thorough, consistent threat modeling gets you 80 percent of the way there.
- Ground yourself with a hobby. Tangible things are an antidote to hype.

Resources

- OWASP Top Ten, Original Recipe
- OWASP LLM Top Ten
- *The Developer's Playbook for Large Language Model Security* by Steve Wilson
- *Mystery AI Hype Theater 3000* Podcast
- Trail of Bits posts
- Jason Haddix's AI red teaming class:
<https://www.arcanum-sec.com>
- Manicode's secure AI education
- Five million breathless hype pieces published every week, some of which are even written by people

Thanks!

Blog version is at breanneboland.com

[@toxoplasmosis@mastodon.social](https://toxoplasmosis@mastodon.social)

Thank you thank you thank you 🧡

CREDITS: This presentation template was created
by **Slidesgo**, including icons by **Flaticon**, and
infographics & images by **Freepik**