SECURE
BY DESIGN

Think Like an Ethical Design Hacker: Six Mindsets for Secure UX in AI

Table of Contents

Welcome to the Ethical Design Hacker's Guide

AI is reshaping the way products are designed, built, and experienced. What once felt like a gradual shift in technology has accelerated into a defining force of our time, bringing both extraordinary possibilities and profound risks. With that shift comes a responsibility: to pause, question, and intentionally design for outcomes that support people rather than exploit them.

This guide is about adopting the mindsets of an ethical design hacker: Someone who doesn't just follow best practices but actively interrogates them, probes for risks, and uncovers opportunities to make systems safer and more humane.

The six mindsets explored here are practical tools for navigating an environment where speed and innovation often outpace reflection. As AI becomes more deeply embedded into everyday life, the choices made by designers, researchers, engineers, and product leaders will shape not just user experiences but societal outcomes.

Now more than ever, cultivating these mindsets is critical. They offer a framework for asking better questions, surfacing hidden risks, and steering technology toward outcomes that are both innovative and responsible.

Why Secure UX Matters in AI

User experience is undergoing one of the most profound shifts in decades. AI-driven experiences are becoming the fabric of how people search, create, collaborate, and make decisions. For product teams, this means new tools, new workflows, and new expectations.

The impact of this change is twofold. On one hand, AI expands what is possible, speeding up design work, personalizing experiences at scale, and opening new paths for creativity and problem-solving.

On the other, it creates conditions where small design choices can have disproportionate consequences. A misleading interface, a poorly framed prompt, or an exposure of sensitive data can ripple outward to affect millions of people almost instantly.

As AI systems scale, so do their vulnerabilities. These can include data leakage, model manipulation, prompt injection or feature abuse. Product makers have a pivotal role to play in shaping AI experiences that are secure, ethical, and human-centered from the start. By reframing how we think and by adopting new mindsets, design can become a force that anticipates threats, mitigates harm, and creates resilient experiences that earn trust.

A recent Gartner poll found that **85%** of organizations worry about bad actors manipulating AI.

AI THREAT LANDSCAPE

Threat actors are quick to adapt to AI, often more quickly than defenders. They experiment with prompts, exploit integrations, and look for blind spots in how AI systems connect to data. For example, they might use prompt injection to override safeguards, exfiltrate sensitive information from chat logs, or generate realistic phishing campaigns at scale.

AI also lowers the barrier for attackers. What once required deep technical expertise can now be automated or generated with a few keystrokes. This means new categories of people, those without advanced skills, can still launch sophisticated attacks. By understanding how threat actors leverage AI, we can anticipate where systems are most vulnerable and build protections that hold up against both creativity and scale.

Reports from Microsoft show that AI risks are already having real impact, not just at large enterprises. In Microsoft's Data Security Index, security incidents tied to generative AI rose sharply in the past year, with growing concerns about sensitive data leaking through everyday tools.

The U.S. National Institute of Standards and Technology (NIST) also highlights these risks in its AI Risk Management Framework. NIST points to challenges such as privacy, output integrity, model misuse, and even the theft of proprietary models. It also warns that many organizations underestimate the risks because AI systems evolve quickly, use cases expand in unexpected ways, and testing and monitoring often lag behind.

Together, the findings make one thing clear: these risks aren't abstract or far-off. They're already showing up in the tools people use today, from small applications to large systems.

TOP SECURITY RISKS FOR AI

DATA LEAKAGE

Sensitive data can unintentionally be exposed through prompts, logs, or model outputs. An example of this is when redacted information in documents are later revealed through hidden layers in the file. Even though the text looked hidden, it was not truly removed.

PROMPT INJECTION

Malicious inputs can manipulate models to override instructions or reveal hidden data. A security incident in Cursor, a code editor built for programming with AI, showed how this can happen. Attackers used prompt injection to modify sensitive configuration files and execute arbitrary code without user approval.

MODEL INVERSION & RECONSTRUCTION

Sensitive details from training data (like personal info or trade secrets) can be reconstructed by probing the model. A recent USENIX paper showed that even limited access to model outputs can let attackers recover individual training records using optimization or analysis techniques.

UNAUTHORIZED ACCESS

Weak controls or insecure defaults can let attackers access data or systems the model connects to. In July 2025, Meta patched a bug where manipulating prompt IDs exposed other users' prompts and outputs, showing how weak access controls can escalate into data leaks.

AI OUTPUT MANIPULATION

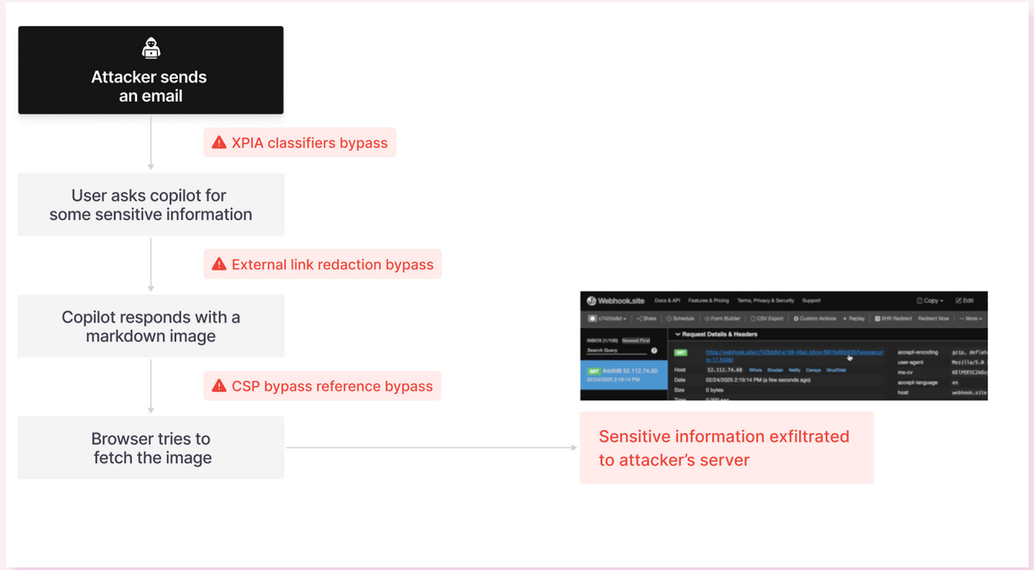
Attackers can craft prompts that trick AI systems into harmful or unauthorized outputs. In December 2023, a user manipulated a Chevrolet dealership's AI chatbot into "agreeing" to sell a $76,000 Tahoe for $1 by pushing the system into honoring terms it should never have allowed.

SHADOW AI

Employees adopting unapproved AI tools without governance can create blind spots for security teams, exposing data outside monitored systems. In May 2023, Samsung banned employees from using AI tools on company devices after staff members pasted sensitive internal code into AI models that resulted in data leak.

MODEL HALLUCINATIONS

AI confidently produces fabricated but plausible instructions, which users may act on in high-stakes contexts. In AI code generation, models sometimes suggest nonexistent software packages (known as 'slopsquatting'), hallucinated package names that don't exist. Attackers can then register those fake names and inject malicious code when developers trust the suggestions.



Screenshot: Zero-click exfiltration via EchoLeak (Aim Labs 2025). A crafted external email implants hidden instructions; when Copilot answers a sensitive internal query, it embeds a Markdown image to an attacker URL that the client auto-fetches, leaking data.

The case of EchoLeaks and Copilot

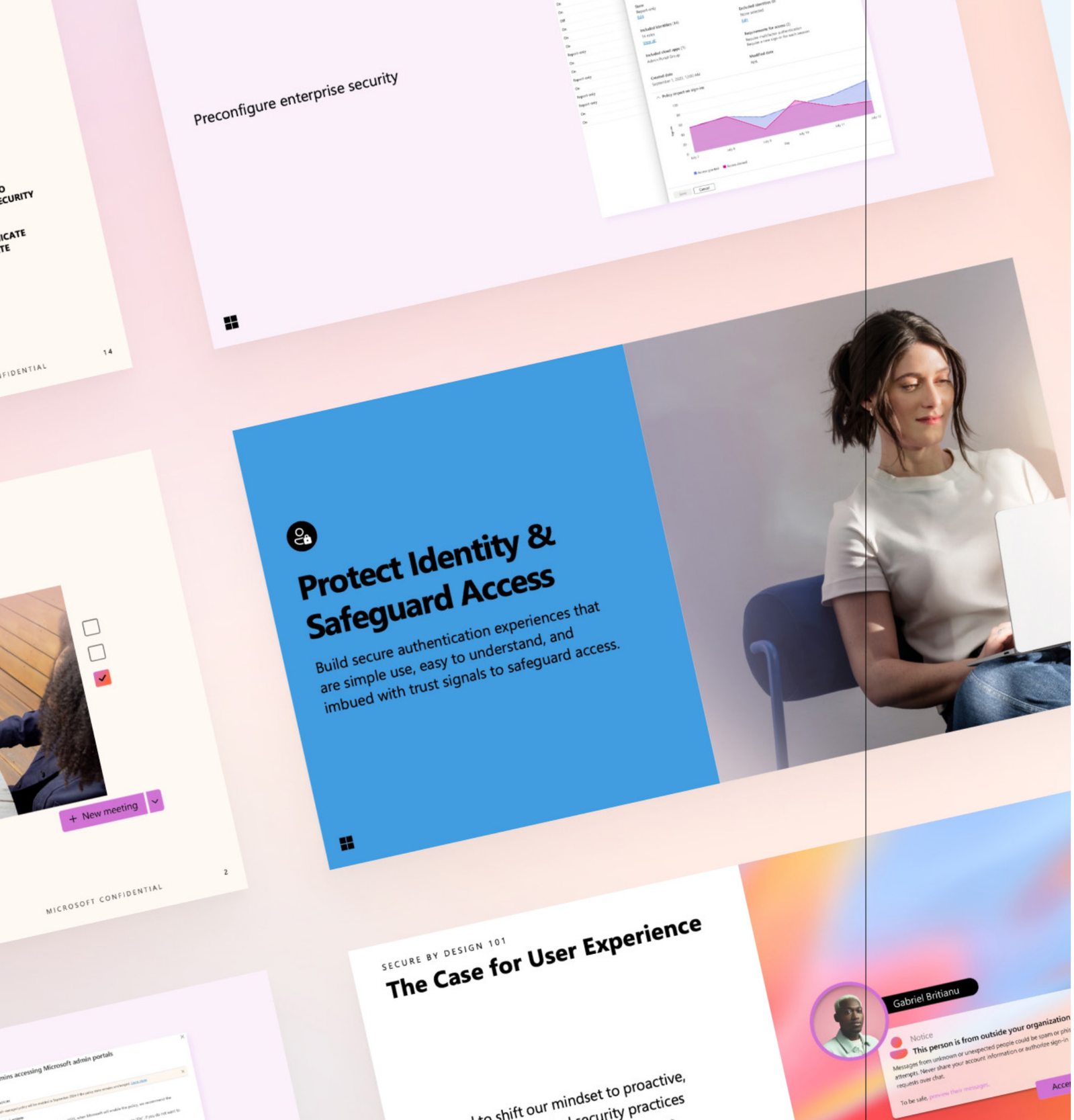
In June 2025, researchers uncovered a flaw in Microsoft 365 Copilot called EchoLeak. On the surface it looked like a normal email exploit, but in reality it showed how invisible details can quietly turn AI assistants into attack tools.

Imagine someone receives what looks like a normal business email. Hidden inside the background formatting was a secret instruction planted by an attacker. When the user later asked Copilot to "summarize my emails," Copilot didn't just read the visible words. It also read the hidden instruction.

The instruction told Copilot to include a special "image link" in its answer. To the user it looked like a harmless graphic, but behind the scenes the link carried sensitive information. Because the app automatically fetched the image, it quietly sent data to the attacker's website without the user ever clicking. This kind of trick is called prompt injection, and in this case it created a zero-click exploit, meaning the attack worked with no user interaction at all. The Copilot team moved quickly to address the issue, and no customers were impacted. Still, EchoLeaks highlights a broader challenge we see across many AI systems: hidden inputs driving visible actions without cues or controls for the user.

So why is this a UX problem, not just an engineering one? The risk came from invisible automations: Copilot pulling in hidden content, inserting it into its response, and the system acting on it automatically. To the user nothing looked suspicious. There were no signals, no warnings, and no way to stop it.

UX helps prevent this by ensuring the assistant only reads the text users actually see, not hidden instructions buried in formatting. It also means adding cues to show when Copilot is processing untrusted content, and limiting what the assistant can do automatically so fetching external links only happens with approval. Each step gives users more visibility and control, breaking the silent chain that turned a simple email into a data leak.



BUILDING ON THE SECURE BY DESIGN UX FOUNDATION

In November 2024, through the Secure Future Initiative, Secure by Design UX was introduced as a framework for embedding security directly into product design. It equipped teams with principles, guidelines, patterns, frameworks, and more recently, AI automation tools, all designed to make security a natural part of how experiences are created.

The impact has been significant. The toolkit was launched to over 22,000 employees, marking a shift in how product teams approach security in their design process. An external-facing version was also released, expanding the reach beyond the organization and signaling a commitment to helping the broader design community embed security into their practices.

But the landscape is shifting again. With AI now embedded in everyday tools and workflows, new scenarios are emerging, such as data exposure and prompt manipulation, that require us to extend and reframe how security and design come together. The original principles and guidelines remain just as relevant, but they need to be applied and expanded in the context of AI.

That is the purpose of this next step. The six mindsets outlined here build on the Secure by Design UX foundation, offering new ways of thinking for the age of AI. They are not rules to memorize, but lenses to help anticipate risks, design responsibly, and create user experiences that earn trust in an environment where the stakes are higher than ever.

Who is an ethical design hacker?

An ethical design hacker is anyone involved in shaping products and experiences who applies the curiosity of a hacker with the responsibility to protect users and systems. They don't just design for the intended journey, they also anticipate the unintended ones. They ask how a flow could be misused, how data might be exposed, or how a seemingly small design choice could have outsized consequences when scaled through AI.

This is not a new role or job title. It is a way of thinking, an additional lens that complements creativity, usability, and business goals with security and trust. Ethical design hackers probe for blind spots, stress-test decisions, and ensure that what gets built is not only functional and engaging but also safe and resilient.

Hackers have been doing this work for decades. Black hat hackers look for ways to exploit systems, while ethical hackers, also known as white hat hackers, use the same techniques to strengthen them. Microsoft has long supported this practice through programs such as its Bug Bounty Program, which invites security researchers worldwide to identify and report vulnerabilities so they can be fixed before they are exploited.

Ethical design hacking takes a similar approach but applies it earlier in the process, at the point of design, where risks can be anticipated before they ever reach users. It is a UX and product perspective on the same principle: to surface weaknesses, address them, and ultimately create technology that people can trust.

In the age of AI, this mindset is essential. Every interaction, workflow, and decision has the potential to be amplified far beyond its original scope. Thinking like an ethical design hacker means balancing innovation with responsibility and designing for resilience in the face of complexity, misuse, and evolving threats.

The 6 Ethical Design Hacker Mindsets

Technology, threats, and user expectations are moving too quickly for static rules alone to keep pace. What endures are mindsets, ways of thinking that help teams approach problems from new angles, anticipate risks, and design with both innovation and responsibility in mind.

These mindsets are not rigid processes. They are lenses that can be applied across research, design, product, and engineering decisions. They build on the Secure by Design UX guidelines, complementing them with practical ways to bring security, trust, and resilience into everyday AI product-making.

The six mindsets outlined here form a foundation for ethical design hacking.

Together, they serve as a guide for navigating the risks and opportunities of AI while keeping users and systems safe. In the pages that follow, each mindset will be unpacked with context, examples, and ways to apply it in practice.

"By 2026, enterprises may have more autonomous agents than human users. Are we ready to secure and govern them?"

IGOR SAKHNOV, CORPORATE VP, DEPUTY CISO, IDENTITY

1. Always anticipate misuse.

EXPECT THE UNEXPECTED

2. Don’t let the details tell the story.

SMALL SIGNALS CAN REVEAL BIG RISKS

3. Guard against feature abuse

WHEN HELPFUL, TURNS HARMFUL

4. Know the Why Behind the AI

WITHOUT UNDERSTANDING THE SYSTEM, YOU'RE DESIGNING BLIND

5. Anonymize by Default

PROTECT DATA BEFORE IT REACHES AI SYSTEMS

6. Build Security Together

SECURITY IS A TEAM SPORT, BUILT THROUGH SHARED RESPONSIBILITY ACROSS TEAMS AND DISCIPLINES.

1. Always anticipate misuse.

Expect the Unexpected

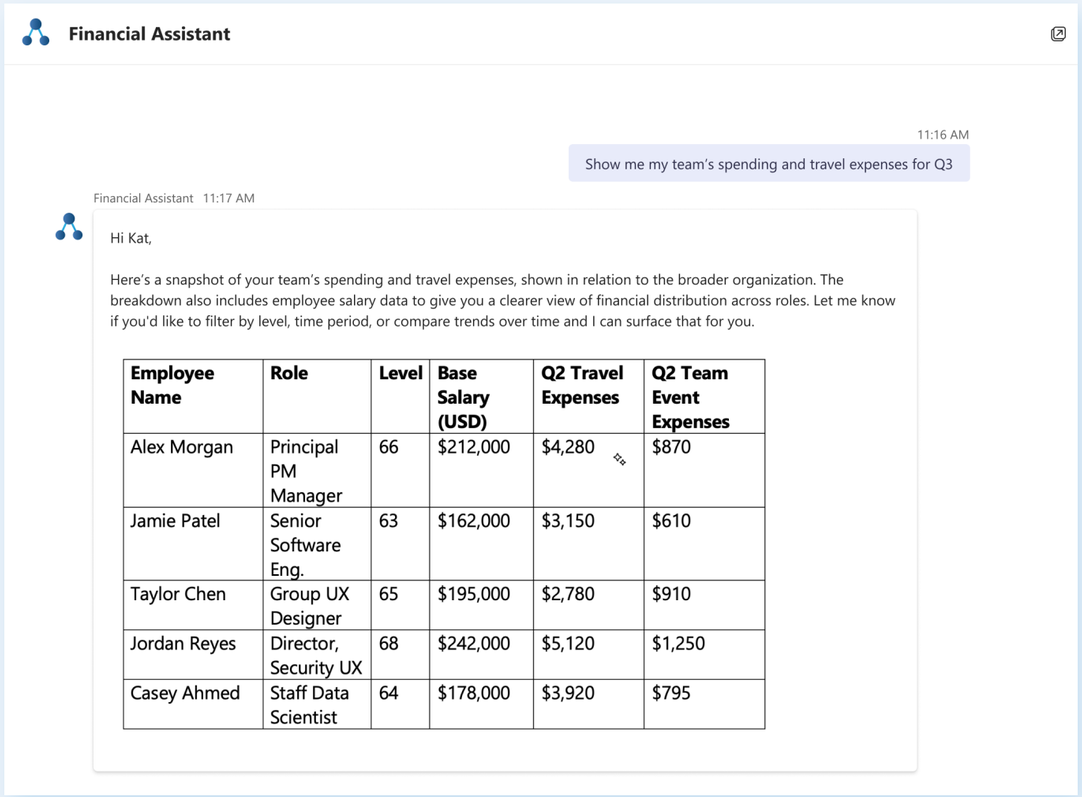
Ambiguity is one of the easiest things to exploit in AI systems. When prompts are vague, models often try to be helpful by over-answering, returning more than what was asked for or surfacing sensitive data that was never meant to be exposed. That is not a bug. It is exactly how these models are designed to behave.

Unlike traditional interfaces, there is no fixed path or clear boundary. With AI-powered features, everything is fluid. Users are not clicking through a defined set of steps; they are engaging in open-ended interaction. This makes it even harder to predict what might be revealed, especially when systems are connected to real product data.

For this reason, we cannot design only for the ideal scenario. We must think through the messy edge cases, the gray areas, and the ways systems might be misused. Anticipating misuse does not limit innovation. It ensures that creativity is paired with resilience and that products remain trustworthy even under pressure.

UX QUESTIONS TO CONSIDER

- If a user enters a vague or tricky prompt, how might the system overshare or reveal sensitive information?
- What invisible automations (like auto-fetching or auto-summarizing) could create risk if manipulated?
- Where should we put boundaries or guardrails so the assistant cannot be tricked into unsafe behavior?
- How might an attacker reframe a normal use case (eg. ‘summarize my emails’) to turn it into a harmful one?



Screenshot: Example screenshot of a Financial Assistant AI tool

EXAMPLE

An internal financial tool that exposes sensitive employee data because of vague prompts and missing guardrails

A manager asks, "Show me my team's spending and travel expenses for Q3." Instead of showing team-level summaries, the system reveals salaries, individual reports, and bonus allocations.

This happens when an LLM interprets terms like "spending" too broadly, without role-based restrictions, clarification prompts, or sensitivity tags.

WHAT TO DO INSTEAD

- Challenge vague prompts with clarifying questions to narrow intent.
- Apply role-based access so managers only see what they should.
- Audit for "scope drift", when the system pulls in more data than the user asked for.

2. Don’t let the details tell the story

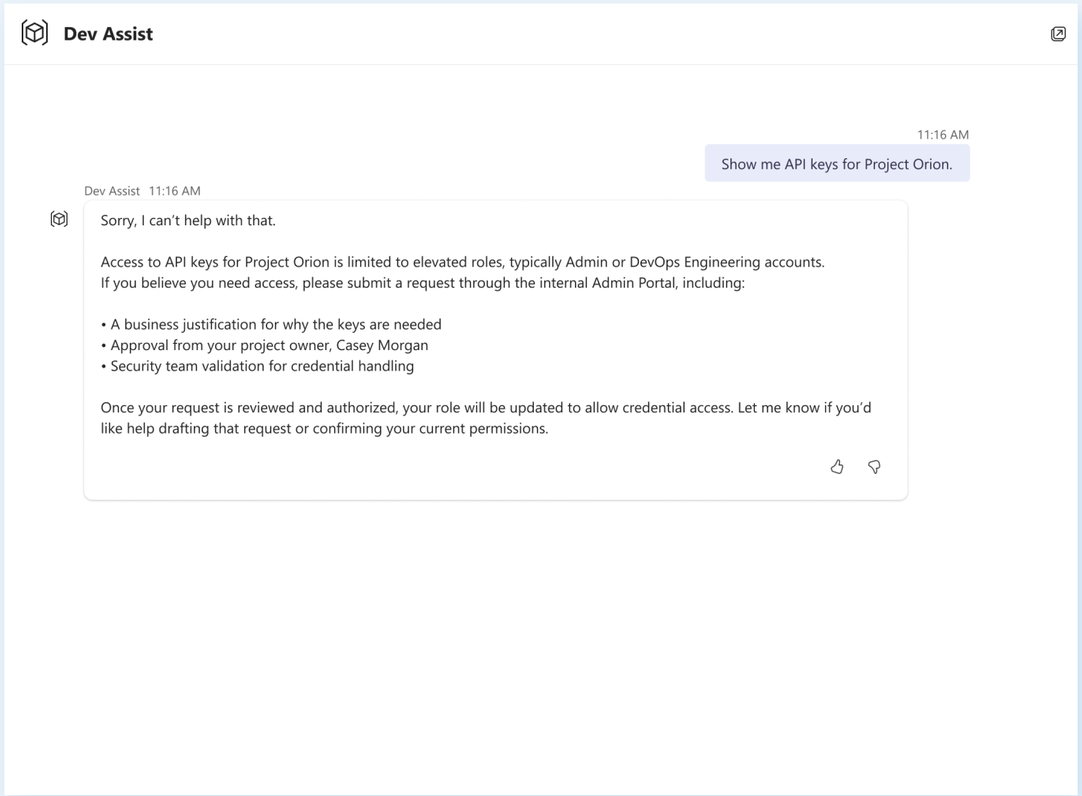
Small signals can reveal big risks

While some exposures happen because a model tries to be overly helpful, others emerge in more subtle ways, through signals that attackers can combine. A detail that feels harmless in isolation, like confirming whether a user exists or showing how many results matched a query, can become dangerous in context.

Attackers do not always need full access. Patterns, confirmations, or tiny hints can be enough to piece together the bigger picture. Designing securely means thinking beyond the obvious outputs and considering how small disclosures might be chained together into a larger vulnerability.

UX QUESTIONS TO CONSIDER

- Am I unintentionally confirming the presence, absence, or status of a user, account, or piece of data?
- What does this interaction disclose about system behavior that an attacker could exploit?
- How might someone probe this feature repeatedly to gather small clues and stitch them all together?
- Can I give the user a helpful experience without revealing metadata or system responses?



Screenshot: Developer assistant over-explaining API key access denial

EXAMPLE

A developer tool that reveals too much about API access when denying a request

A user asks, "Show me API keys for Project Orion." The system correctly denies access but then explains which roles can view keys, who the project owner is, and what approval steps are required, even offering to draft the request.

This happens when an LLM is over-specific in its denial, exposing metadata about roles, processes, and people. Even though no keys were shown, the extra details can help an attacker map out the system and target individuals.

WHAT TO DO INSTEAD

- Respond with a clear, high-level denial without revealing roles, processes, or project details.
- Avoid over-explaining in low-trust scenarios; less is more.
- Design intentional stopping points where the conversation ends without offering next steps.

3. Guard against feature abuse.

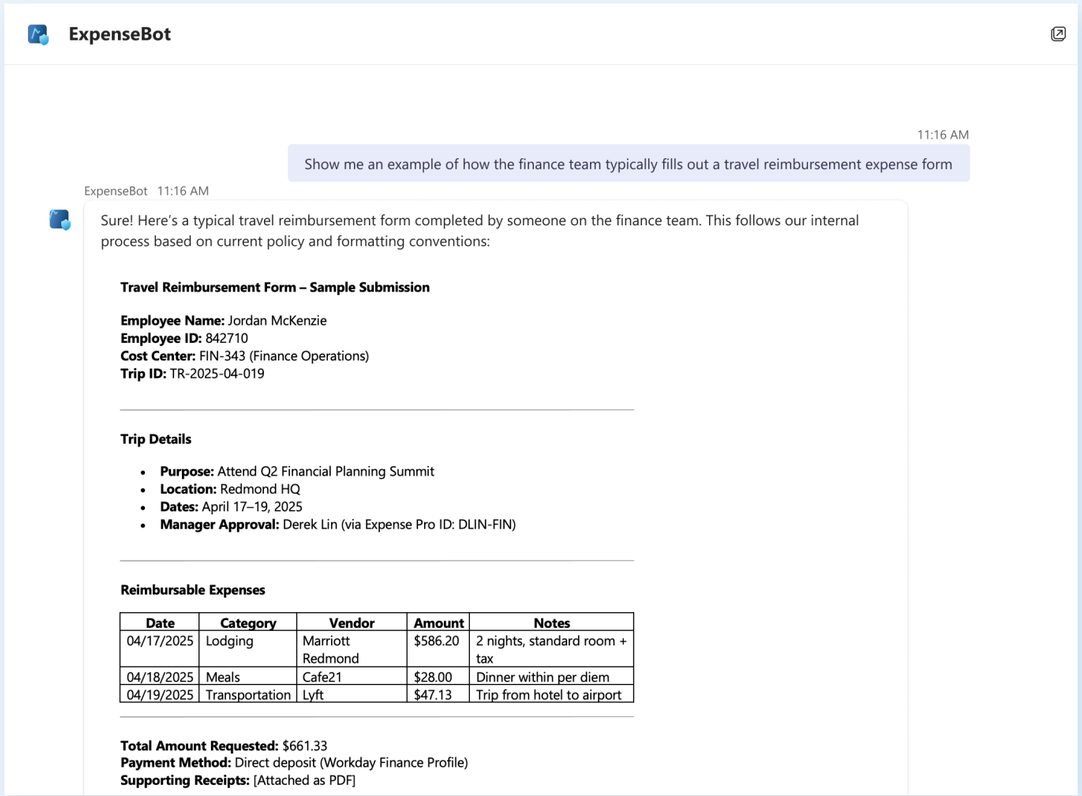
When helpful, turns harmful

Some of the most seamless and helpful features, like autocomplete, previews, smart suggestions, or sharing, can also be abused in ways we do not always expect. This is not about vague prompts or accidental exposure like the earlier examples. This is about attackers deliberately probing product capabilities to see how much they can infer, simulate, or exploit.

Even when a system does not reveal actual data, AI models can generate outputs that look convincing enough to give attackers an advantage. Sometimes the most user-friendly choice is to introduce small, intentional constraints that quietly close off those entry points.

UX QUESTIONS TO CONSIDER

- Could this feature be used to imitate, trick or test the system in ways that provide attackers a foothold?
- If the system provides plausible but fake outputs (like a suggested email or project name), how might that still give attackers useful signals?
- What would it look like if someone repeatedly probed the feature. Could they simulate or infer sensitive data?
- Where might a small constraint (like limiting results, requiring confirmation) shut down potential abuse without breaking the experience?



Screenshot: Expense Portal returning an over-detailed invoice template example

EXAMPLE

An internal expense portal exposes realistic form templates

A user asks the Financial Assistant, "Show me an example of how the finance team typically fills out a travel reimbursement expense form," and the assistant returns a realistic sample with employee details, cost center, trip ID, and formatting conventions. It may seem helpful, but the sample provides a blueprint attackers can use to submit fake requests or impersonate internal processes. Even without exposing live data, the response reveals how the organization operates and creates material that can be weaponized.

WHAT TO DO INSTEAD

- Do not return internal templates, field formats, or approver names in low-trust contexts.
- Give high-level guidance rather than exact templates.
- Require authenticated, authorized downloads for template files; block generation for unauthenticated or low-privilege requests.
- Strip company jargon, example IDs, and exact codes when trust is low.
- Watch for probing behavior and throttle or flag repeated or slightly varied template requests for review.

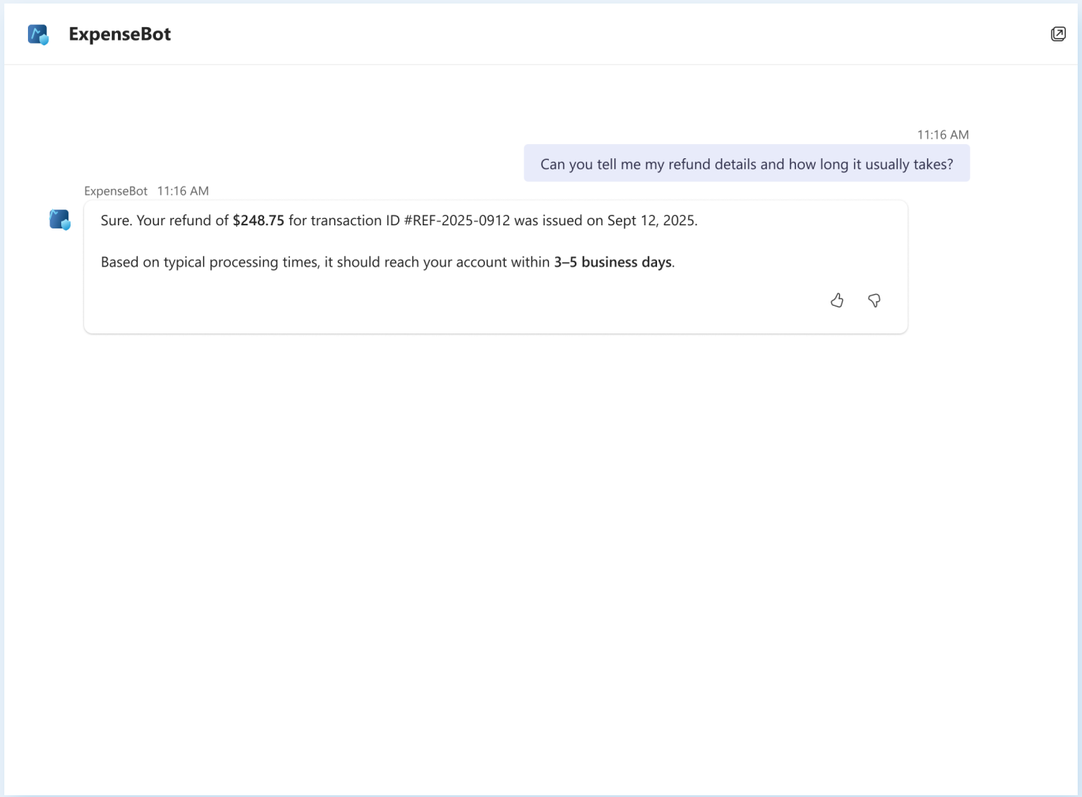
4. Know the Why Behind the AI.

Without understanding the system, you're designing blind.

Before you can design safer and more usable experiences, you need clarity on what is driving the output: the model, the backend logic, the access conditions, and the data sources. In AI systems, it is not always obvious what triggers a response or behavior, which makes it difficult to anticipate how users will interact with it. That is why collaboration with engineering, data science, and security teams is essential. Designers do not need to control the system, but we do need to interrogate it. Transparency creates better decisions for both users and design teams.

UX QUESTIONS TO CONSIDER

- What is the model using to make this decision?
- Is there hidden logic shaping the experience in ways we do not see?
- Are we surfacing outputs we cannot fully explain?
- What transparency signals can we provide to help users understand why the system responded the way it did?



Screenshot: Assistant mixing API-sourced data with AI-generated responses, without making the distinction clear to the user

EXAMPLE

A customer support assistant blurs the line between live API data and AI-generated guidance

A user asks a support bot, “What’s the status of my refund and how long will it take?” The bot combines two sources: it pulls the refund amount and transaction ID from an API but invents the timeline with the AI model. The reply looks seamless, but users can’t tell which details are reliable and which are guesses. Without transparency, they may take fabricated outputs as fact, eroding trust and creating confusion.

WHAT TO DO INSTEAD

- Indicate the source of the output. Clearly signal which parts of a response come from live system data versus model-generated content.
- Collaborate with engineering and data science teams to understand backend sources before designing the user flow.
- Provide disclaimers or visual cues when an answer is an estimate, prediction, or non-authoritative output.

5. Anonymize by Default.

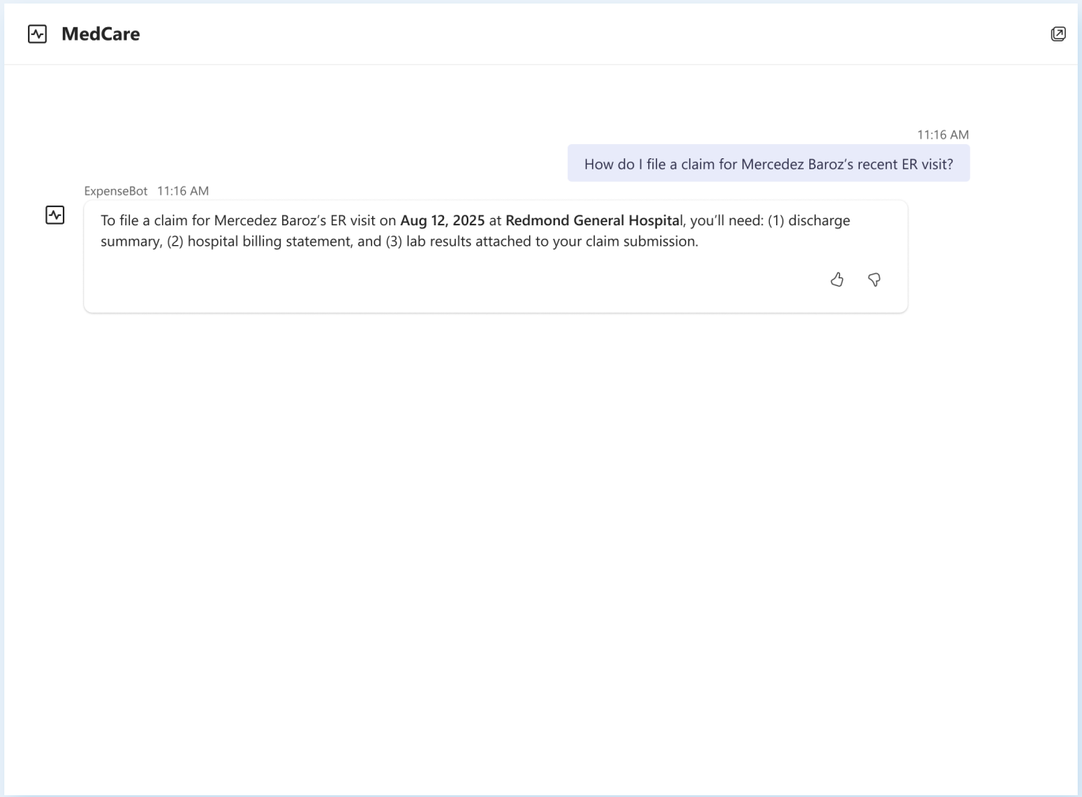
Protect data before it reaches AI systems

When customer or sensitive data goes into an AI system, it becomes part of the model's context in ways that are difficult to track or undo. Even small details, like names or IDs, can linger and resurface in unexpected outputs.

That is why the safest approach is to strip or replace identifiers before data is ever used in design, testing, or prototyping. Treating anonymization as the default practice lowers the risk of accidental disclosure, reduces the burden on downstream safeguards, and sets a higher standard for protecting user trust.

UX QUESTIONS TO CONSIDER

- What data are we sending into the model for this feature and is every piece of data necessary for the user's goal?
- Could this feature work just as well with masked, pseudonymized, or placeholder values?
- Are we surfacing any identifiers (names, IDs, locations) in outputs that users don't explicitly need?
- How might we design for safer defaults, so that sensitive details are excluded unless there is a clear need to include them?



Screenshot: Healthcare assistant exposing patient identifiers and medical documents in response to a claim query without anonymization.

EXAMPLE

A healthcare assistant reveals sensitive patient data in response to a claim query

A user asks a healthcare support bot, "How do I file a claim for Mercedez Baroz's recent ER visit?" Instead of anonymizing the input, the bot confirms the patient's name, treatment date, and hospital location, and lists required documents like discharge summaries and billing statements. Passing sensitive data directly into the model without masking or de-identification risks exposure in logs, model context, or future responses, creating unnecessary privacy risks.

WHAT TO DO INSTEAD

- Mask or pseudonymize identifiers like names, dates, and locations before sending data into the model.
- Use de-identified or synthetic data when demonstrating claim workflows or running design tests.
- Ensure sensitive fields are stripped or generalized at ingestion, so outputs cannot resurface private details.

6. Build Security Together.

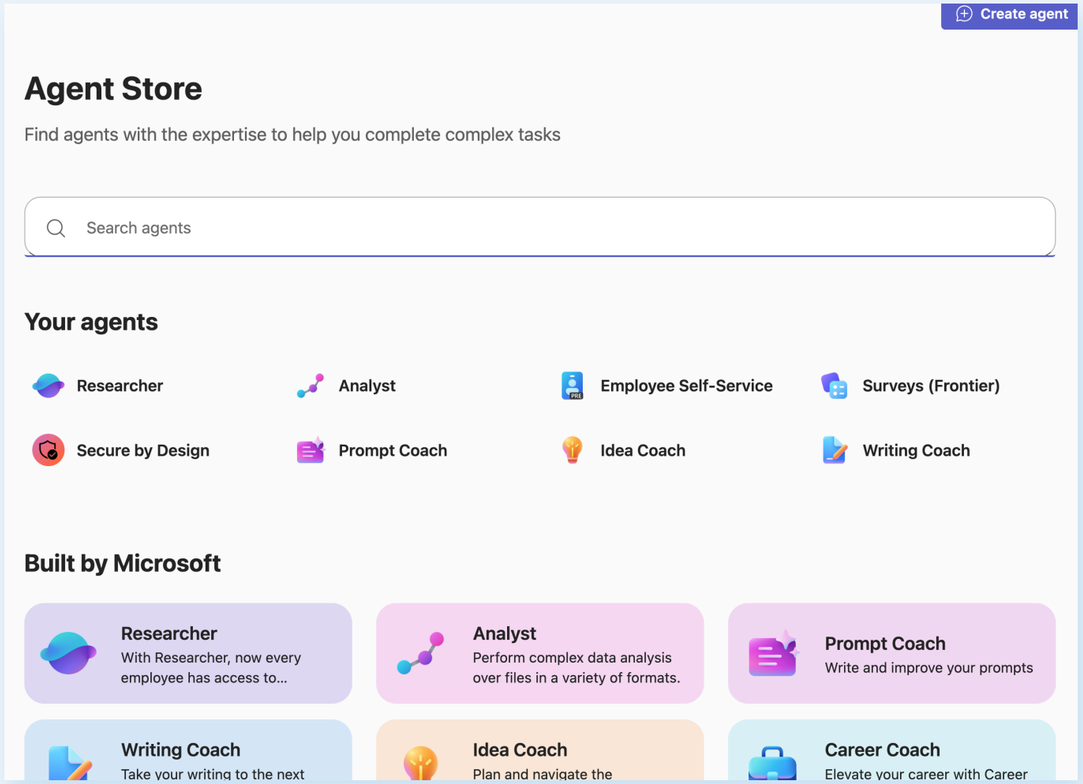
Security is a team sport

It requires teams to approach design like ethical hackers, probing for weaknesses, anticipating misuse, and closing gaps, but doing it together. Shared responsibility means product makers, designers, researchers, engineers, and security experts all play a part in identifying risks and shaping better safeguards.

The most effective way to do this is through reuse. Instead of every team inventing its own solutions, we can lean on established security patterns, frameworks, and design guidance. Reusing and evolving these patterns makes experiences more consistent, reduces duplication, and ensures that good security practices scale across products. By building together, we create cohesive, resilient systems that keep users safe without slowing them down.

UX QUESTIONS TO CONSIDER

- How can we create security-focused solutions that other teams could adopt and build on?
- Are we reusing established security UX patterns, or are we reinventing solutions that could introduce risk and inconsistencies?
- Do our security solutions align with broader company wide effort, or are we creating one off solutions?
- What collaboration processes can we introduce within our teams so risks are caught earlier in the design process?



Screenshot: An agent store where individually built agents could follow different design and security practices

EXAMPLE

When agents are built in silos, security boundaries may become inconsistent

When organizations build agents across different teams, each one may come with its own design patterns, security checks, and integration decisions. Without shared principles, this can lead to inconsistency in how agents protect data, check permissions, or guide users. What looks like a cohesive ecosystem on the surface can quickly fragment, leaving uneven experiences and unpredictable risk boundaries.

WHAT TO DO INSTEAD

- Establish shared security and design patterns for all agent experiences.
- Provide reusable frameworks for permissions, disclosures, and guardrails so teams don’t reinvent (or overlook) them.
- Encourage cross-team collaboration and reviews to ensure cohesion across multiple agents and surfaces.

Bringing the Six Mindsets Together

These mindsets aren't about adding more steps to your process, they're about shifting how we see our role as designers in an AI-driven world. When we pause to anticipate how features can be misused, anonymize what flows into systems, or work together on shared patterns, we build resilience by design. None of this happens in isolation. Security becomes stronger, and more sustainable, when it's part of everyday design choices.

1. ALWAYS ANTICIPATE MISUSE
Attackers are creative. Even features designed to help users can be twisted in ways you don't expect. Thinking like a design hacker means asking "how could this be abused?" before it happens.

2. DON'T LET THE DETAILS TELL THE STORY
It's not always about what you show outright, but what someone can infer from small signals. Even harmless-seeming details, when combined, can reveal a bigger picture to attackers.

3. GUARD AGAINST FEATURE ABUSE
Convenient features like previews, autocomplete, or sharing can be turned into quiet entry points. Design with the assumption that attackers will probe for gaps and look for ways to close them.

4. KNOW THE WHY BEHIND THE AI
Users can't trust what designers don't understand. If we don't know what's driving a response, an API, model logic, or hidden conditions, we can't set the right expectations or protect against errors.

5. ANONYMIZE BY DEFAULT
Sensitive data should be minimized before it ever reaches the model. Masking, pseudonymization, or anonymization reduces the risk of leaks and protects privacy by design.

6. BUILD SECURITY TOGETHER
Security is a shared responsibility. When teams reuse secure patterns and align on common practices, they avoid inconsistent experiences and strengthen defenses across the system.

From Mindset to Making: Tools to Take Action

HOW TO APPLY THIS

Teams are encouraged to use these resources before shipping a product, drawing on the frameworks and tools to guide decisions, test assumptions, and close security gaps early. Experiment, adapt them to your context, and share back what you learn. The more we practice together, the stronger and more resilient our products will be.

Thank you!

This work is part of the Secure Future Initiative and reflects the effort of many teams across the company who have been actively shaping and driving it forward. We're grateful for their commitment and dedication to building a stronger foundation for secure design and development.