SHOSTACK
+ ASSOCIATES

Inherent Threats

Clarifying a property of threats:
Are they inherent to the system?

Shostack + Associates White Paper #4
March 2024
by Adam Shostack

Context

As we consider the things that can go wrong with our system and what we'll do about them, an important aspect of threats starts to emerge. Some are easily fixed, but others are not. This can be frustrating and confusing.

A crucial aspect of a threat is how inherent it is to the application. Some threats are purely accidental or the result of an oversight. For example, data can be exposed to the Internet because of a cloud misconfiguration. Others are inherent to a system. For example, a banking app can inappropriately reveal your bank balance, or it can transfer money to the wrong place. There are ways to reduce the odds that each will happen, but they're inherent to the ways the bank serves its customers.

In fact, as banking has moved online over the years, it has increased both functionality for and threats to customers. For simplicity, table 1 ignores that controls and mitigations were added along the way.

State	Change: What's added	New threats
Not online	Create a "brochure website"	Website defacement, spoofing. (Impact to customers likely small.)
Brochure website	Add read-only backend access via the bank website	Information disclosure (balances, payments)
Account balance, transactions on the web	Send money	Theft of funds

Table 1: Increasing Threats

This white paper shines a light on why some threats are easy to address and others are not, and the strategies that can help deal with those complex mitigations.

Inherent, intrinsic, or essential?

There are three closely related words — inherent, intrinsic, and essential — that people often use interchangeably. Understanding the nuanced differences between them may be helpful. The thesaurus says that inherent is used when there are negative consequences, intrinsic lacks that connotation of negativity, and something is essential if it's so important that it cannot be removed without a fundamental alteration.

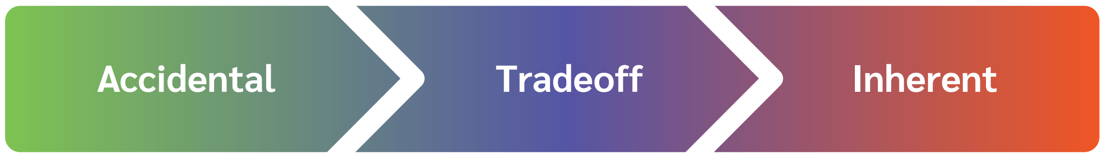
Is the threat inherent?

Many systems are vulnerable to threats because no one thought to look for them. Examples include world-readable cloud storage or APIs that lack authentication. A smaller number of threats are present because the people who looked for them lacked the time, skill, or understanding to identify them. For example, if a penetration tester had spent less time discovering the parts of a system and had more time to analyze it, they might have found a threat. A few threats remain present because, over time, an accident becomes a tradeoff. For example, when an important client starts using an authentication-free API, the cost of fixing it includes communicating with clients and coordinating a switch-off date. These tradeoffs are discussed further in the section "Tradeoffs."

At the other end of the spectrum are those threats that are inherent to a system. For example, a banking app could send money to the wrong place. A messaging app might carry harassing or offensive messages. Moreover, a system may be vulnerable to threats by design. Large language models like ChatGPT will hallucinate and show biases. Credit systems accept that some people won't be able to pay their debts, and different lenders manage that differently. Accepting that allows them to operate at a larger scale and with less security friction and make more money overall. Or go bust. It can be a tricky balance, and it's one that businesses make intentionally.

Assessing threats across a spectrum

We can assess where every threat exists on this spectrum:



Accidents

When a threat hasn't been addressed because of an oversight, the response can be either to eliminate the feature that brings it in or to put a fully effective mitigation in place. For example, if we're worried about tampering with files, we can move files from /tmp to a private directory, adjust the permissions on cloud storage, or add good authentication and integrity to an API. If that last approach involves a keyed MAC using a strong hash over all the fields, then there are threats to the hash algorithm and threats to the keys, but the spoofing or tampering threats to the API calls are effectively mitigated.

Tradeoffs

There are times when a tradeoff is inherent to a defense. These tradeoffs often manifest in authentication, usability, machine learning, and compatibility.

Authentication is one classic locus for tradeoffs. One example is using authenticator apps for stronger authentication, which requires people to make more effort and to have a smartphone (still a problem for poor people in general, and in less wealthy countries in particular). Authentication via text message has been deprecated by NIST since 2017's Special Publication 800-63b, may not work for people who are roaming or otherwise inaccessible, and may be resisted by those who don't want the business to have their cell phone number.

Usability tradeoffs often manifest in authentication but extend far beyond it. Warnings are a popular way to help the people using the software to do so safely, but sometimes they seem designed to transfer the risk via "CYA" dialogs, such as "some documents can harm your computer, click here to get your job done." It's important to realize that good design can result in systems that are both more usable and more secure rather than cobbling something together, and that's a subject that's covered in books like *Usable Security* (Garfinkel and Lipford, 2014).

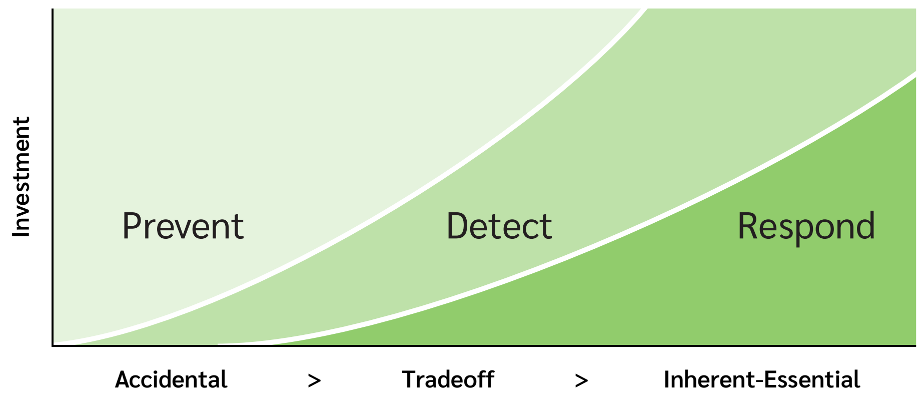
Spam detection systems illustrate another type of tradeoff, often found in **machine learning**. These systems can be tuned to adjust the false positive and false negative rates, leading to either more real mail in the spam box or more spam in the inbox. All statistical and machine learning systems will have these tradeoffs to some extent.

The last common tradeoff is in **compatibility**. New protocols often accumulate features and complexity as experimentation clarifies needs and use cases. (This also applies to file formats, which can be seen as protocols intermediated by filesystems.) As protocols span beyond a small team at a single organization, the effort to upgrade them grows rapidly, the need to support the older system adds complexity, and issues of compatibility often play into the tradeoffs that we make in addressing threats.

For each of these tradeoffs, there will be a mix of preventative, detective, and responsive controls, and the mix will be dependent on both security and general business factors.

Inherent-essential

When a threat is tied to the essence of a system, protective measures cannot be perfect or complete. Therefore, detective and responsive controls become a relatively larger portion of the defensive investments.



“Inherent” influences how you threat model

Understanding these tradeoffs influences how we threat model in two important ways. First, it informs more in-depth threat modeling as we struggle to specify answers to “what are we going to do about it?” Second, it helps us consider inherent threats when we scale threat modeling across hundreds or thousands of systems so we can prioritize what gets attention first.

Informing threat modeling

Assessing a particular threat to determine if it’s inherent or involves tradeoffs can provide clarity about how to address it. I’ve heard from my students that being aware of the different types of threats helps increase their confidence as they start to select mitigations.

Accidental threats are relatively easy to fix and can often be addressed without controversy or risk management work. Those that are inherent will be addressed via detection and response, or possibly by risk acceptance or transfer. If you find threats that will lock in design choices or create compatibility problems to fix, addressing them soon and even delaying a release will pay off.

Threat modeling thousands of systems

People sometimes ask, “How are we going to threat model thousands of systems if each in-depth threat model takes us weeks?” One obvious strategy is to optimize what you do, and the inherent threats to a system inform how important it may be to go deeper to see how well it’s designed and implemented. A brochure system has a lower inherent risk than one with account access. The question may well be: what’s the worst that can possibly happen?

Depending on what sort of issues come up, you might set thresholds for levels of severity:

- Anything that can lead to people being fired is priority 1.
- Things that are likely to result in a million-dollar loss are priority 2.
- Anything whose impact would be “we shut off the system and business can continue for a month” is priority 3.

That might be enough for a rough cut, or you might want more specificity.

You might define a few additional factors that matter to you — is the system customer- or partner-facing, or is it “internal” (whatever that even means)? Is the data in it regulated in some way? Does the system have safety impacts? In each of these cases, the threat is “inherent” to the system, and we don’t need to go deep to discover it. (In fact, the ways executives answer the question “what can go wrong?” will tend to align with inherent threats to the system.)

Conclusion

Understanding if a threat is inherent to a system is tremendously clarifying. It informs how we address that threat. It shows us where residual risk is unavoidable. It dictates our choices of how to balance protection, detection, and response. Last, but certainly not least, it enables us to scale threat modeling across the enormous application inventories that companies develop as they grow.

SHOSTACK + ASSOCIATES

ABOUT SHOSTACK + ASSOCIATES

Adam Shostack founded the company that bears his name in 2016. Shostack + Associates now focuses on delivering great learning experiences, primarily around threat modeling, including classic training and also helping leaders learn to navigate the complex organizational changes that often surround threat modeling.

Get In Touch

If threat modeling isn't delivering what you hope for, then it's our hope that this paper will help. If we can help further, please don't hesitate to reach out for a confidential consultation, at *adam@shostack.org.*

ABOUT ADAM SHOSTACK

Adam is the author of Threat Modeling: Designing for Security and Threats: What Every Engineer Should Learn from Star Wars. He's a leading expert on threat modeling, a consultant, expert witness, and game designer. He has decades of experience delivering security. His experience ranges across the business world from founding startups to nearly a decade at Microsoft.

His accomplishments include:

- Helped create the CVE. Now an Emeritus member of the Advisory Board.
- Fixed Autorun for hundreds of millions of systems
- Led the design and delivery of the Microsoft SDL Threat Modeling Tool (v3)
- Created the *Elevation of Privilege* threat modeling game
- Co-authored *The New School of Information Security*

Beyond consulting and training, Shostack serves as a member of the Blackhat Review Board, an advisor to a variety of companies and academic institutions, and an Affiliate Professor at the Paul G. Allen School of Computer Science and Engineering at the University of Washington.