

# Threat Modeling

Integrating Threat Modeling into the SDLC (Software Development Lifecycle)

# Who am I

- Former OWASP Newcastle Chapter leader and ISC2 North-East England Chapter Leader
- Currently Senior Manager at Hargreaves Lansdown responsible for security testing.
- Previous 2 years as Executive Consultant at NCC Group embedded in clients where introducing threat modeling was a major topic.

# What this talk is and isn't

It **IS** observations and lessons learned from creating and trying to embed threat modeling process as a standard practice with development teams

It **ISN'T** how to do threat modeling. This has been done many times before. But I will give enough of an overview for those new to threat modeling to follow the talk.

# Agenda

The value of integrating threat modeling in the SDLC



Pros and cons to some of the approaches and tools



Challenges and push back posed by development teams



Ways to respond to challenges for the best chance of success

# Threat Modeling

The threat modelling manifesto defines it as:

“analysing **representations** of a system to highlight concerns about security and privacy characteristics.”

In its simplest form, threat modeling is asking 4 questions about a system:

- What are we building?
- What can go wrong?
- What are we going to do about it?
- Did we do a good job?

# Threat Modeling

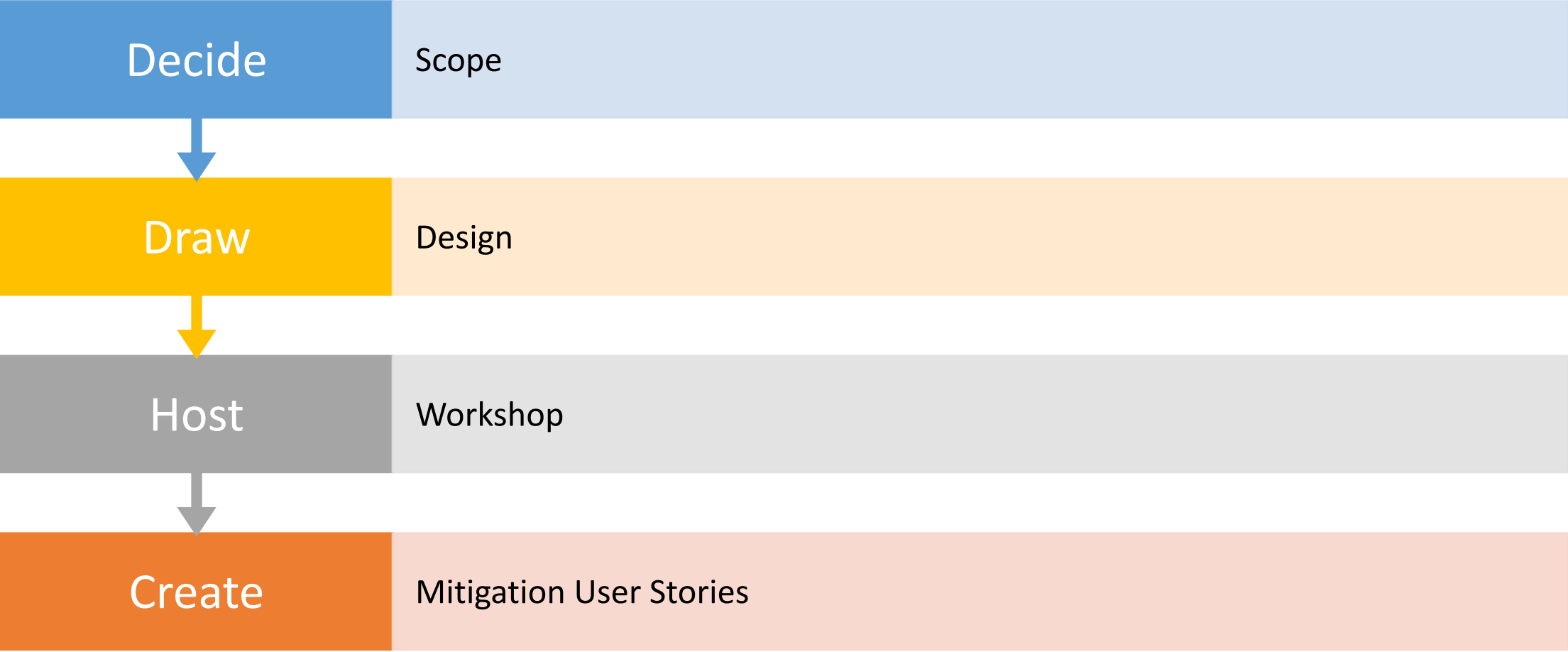
Microsoft SDL (Security Development Lifecycle) says:

- Threat modeling should be part of your **routine development lifecycle**, enabling you to progressively refine your threat model and further reduce risk.

Microsoft SDL defined 5 steps:

1. Defining security requirements.
2. Creating an application diagram.
3. Identifying threats.
4. Mitigating threats.
5. Validating that threats have been mitigated.

Source: <https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling>

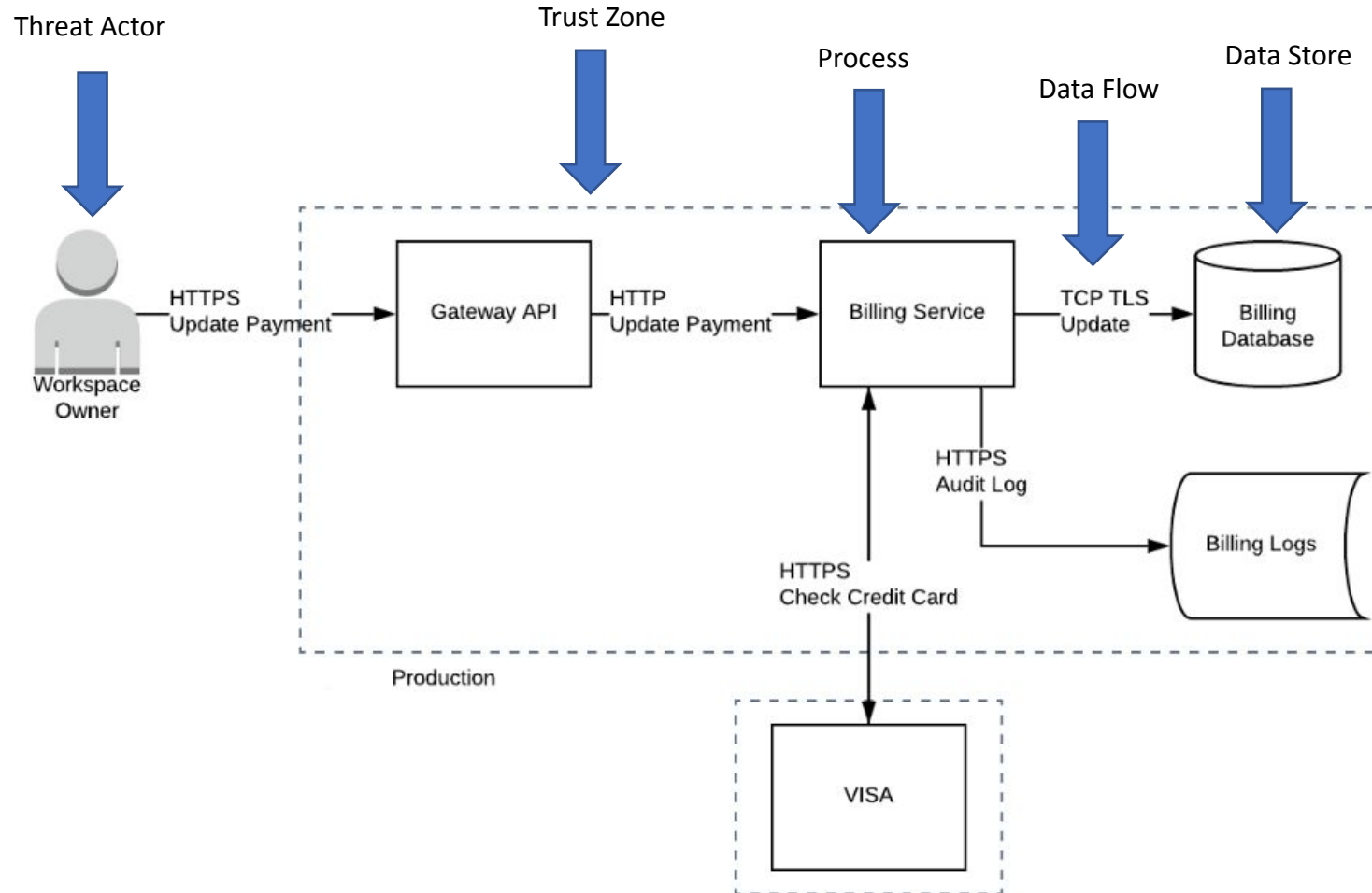


# What does that mean?

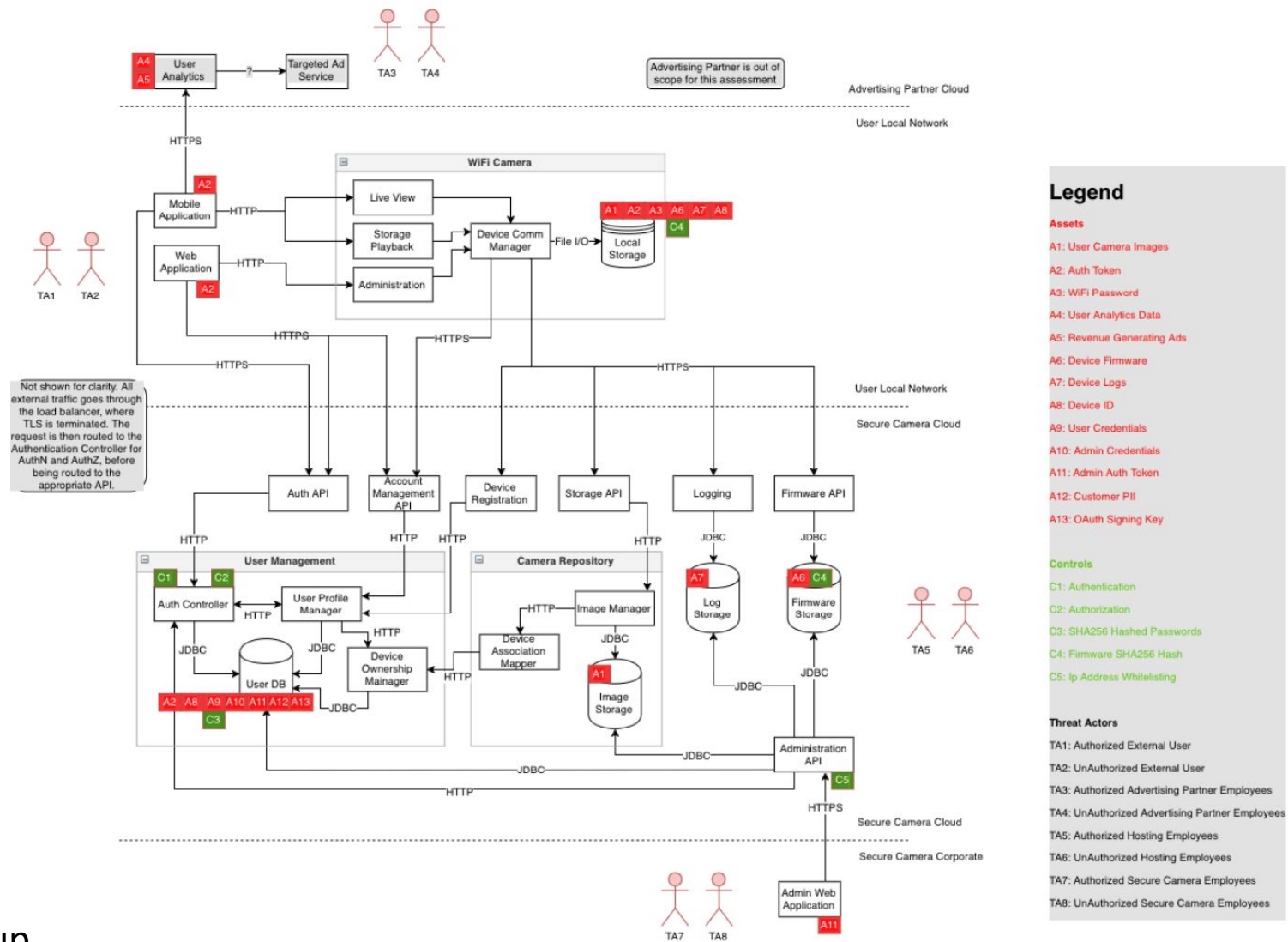
At the end of Threat Modelling, we only ask for three things:

1. We thought about security
2. We *\*might\** have drawn a diagram to analyse (that we can reuse)
3. We came up with a list of threats and mitigations (that we feed to our issue trackers as user stories)

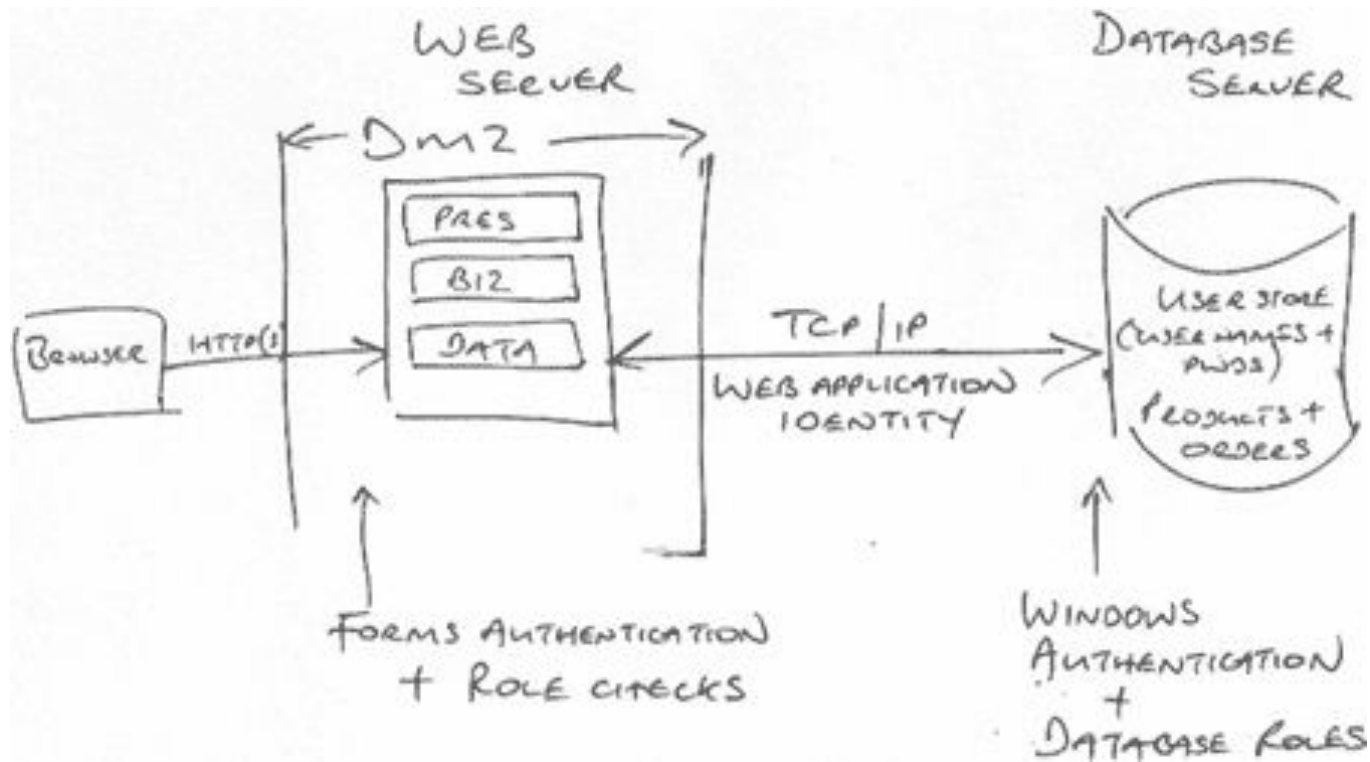




Source: segment.com

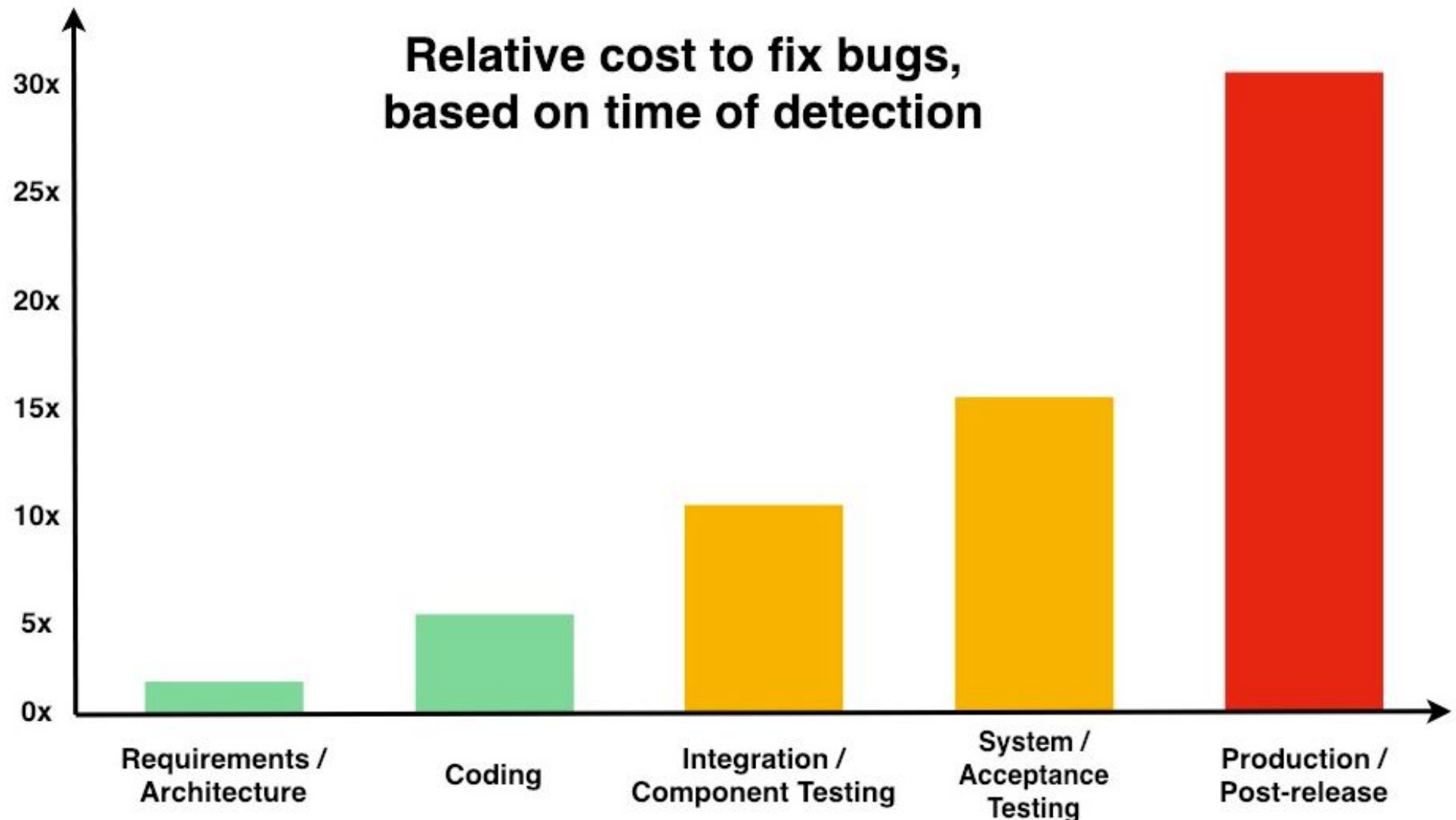


Source: NCC Group

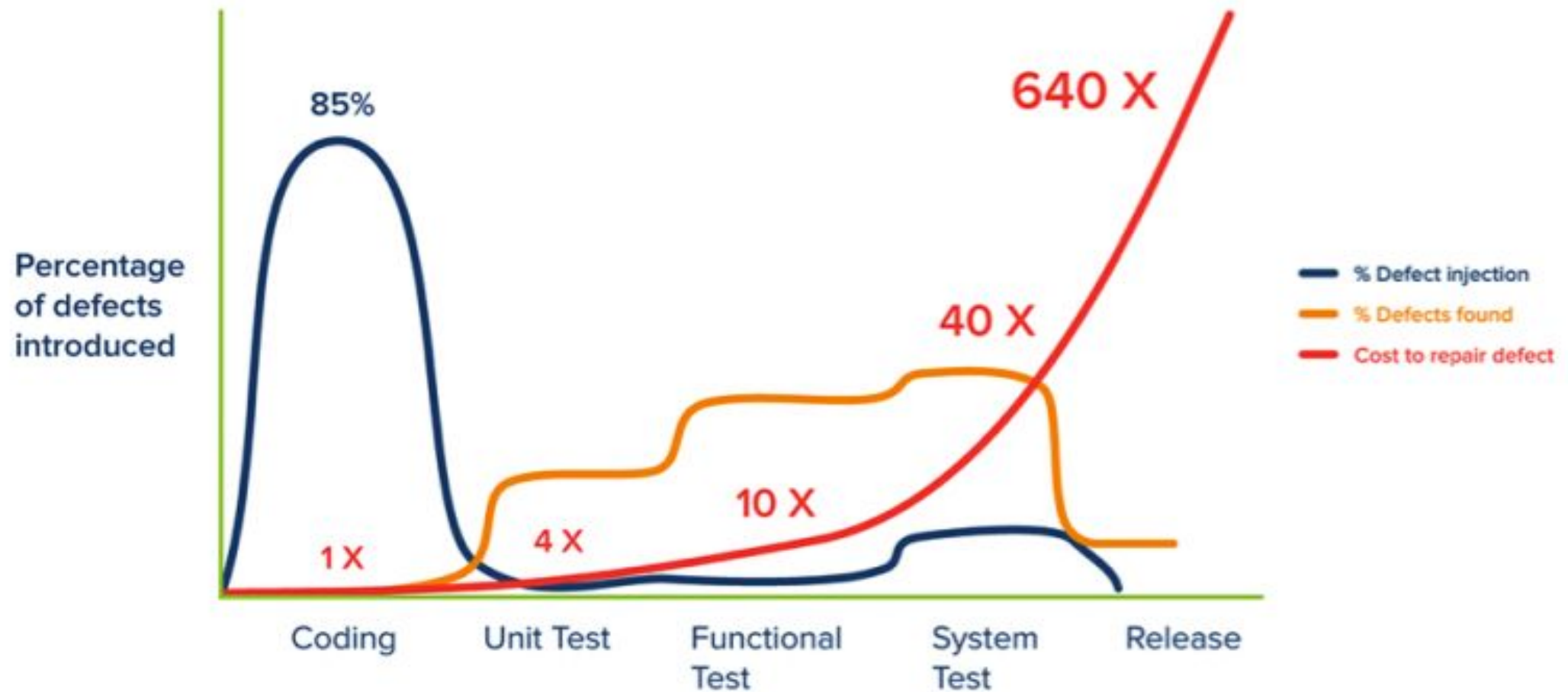


Source: NCC Group

# The value of integrating threat modeling in the SDLC



[Source: NIST Relative Cost to Repair Defects](https://www.nist.gov/system/files/documents/director/planning/report02-3.pdf)



Jones, Capers. *Applied Software Measurement: Global Analysis of Productivity and Quality*.

[Source: The Shift-Left Approach to Software](https://www.stickyminds.com/article/shift-left-approach-software-testing)

<https://www.stickyminds.com/article/shift-left-approach-software-testing>


Confidential

# BSIMM (Building Security In Maturity Model)

## BSIMM

[What is BSIMM](#) [Download the BSIMM](#) [BSIMM Framework](#) [Resources](#) [Q](#)


### Software Security Framework Domains



#### Governance

Practices that help organize, manage, and measure a software security initiative


[Learn more](#)



#### Intelligence

Practices that result in collections of corporate knowledge used in carrying out software security activities throughout the organization


[Learn more](#)



#### SSDL Touchpoints

Practices associated with analysis and assurance of particular software development artifacts and processes

[Learn more](#)



#### Deployment

Practices that interface with traditional network security and software maintenance organizations

[Learn more](#)

## Intelligence

Creating corporate knowledge used in software security activities throughout the organization

Intelligence includes those practices that result in collections of corporate knowledge used in car organization. Collections include both proactive security guidance and organizational threat mod

### Attack Models

Attack Models capture information used to think like an attacker: **threat modeling**, abuse case development and refinement, data classification, and technology-specific attack patterns.

[Learn more](#)

### Security Features & Design

The Security Features & Design practice is charged with creating usable security patterns for major security controls (meeting the standards defined in the Standards & Requirements practice), building middleware frameworks for those controls, and creating and publishing proactive security guidance.

[Learn more](#)



# NIST SSDF (Secure Software Development Framework)

NIST SP 800-218

Practices	Tasks
<b>Produce Well-Secured Software (PW)</b>	
<b>Design Software to Meet Security Requirements and Mitigate Security Risks (PW.1):</b> Identify and evaluate the security requirements for the software; determine what security risks the software is likely to face during operation and how the software's design and architecture should mitigate those risks; and justify any cases where risk-based analysis indicates that security requirements should be relaxed or waived. Addressing security requirements and risks during software design (secure by design) is key for improving software security and also helps improve development efficiency.	<b>PW.1.1:</b> Use forms of risk modeling – such as <b>threat modeling</b> , attack modeling, or attack surface mapping – to help assess the security risk for the software.



# OWASP SAMM (Software Assurance Maturity Model)



### Threat Modeling

---

- 1 Perform best-effort, risk-based threat modeling using brainstorming and existing diagrams with simple threat checklists.
- 2 Standardize threat modeling training, processes, and tools to scale across the organization.
- 3 Continuously optimization and automation of your threat modeling methodology.

## Annex 5

### List of threats and corresponding mitigations

1. This annex consists of three parts. Part A of this annex describes the baseline for threats, vulnerabilities and attack methods. Part B of this annex describes mitigations to the threats which are intended for vehicle types. Part C describes mitigations to the threats which are intended for areas outside of vehicles, e.g. on IT backends.

E/ECE/TRANS/505/Rev.3/Add.154  
Annex 5

Table A1

List of vulnerability or attack method related to the threats

High level and sub-level descriptions of vulnerability/ threat			Example of vulnerability or attack method	
4.3.1 Threats regarding back-end servers related to vehicles in the field	1	Back-end servers used as a means to attack a vehicle or extract data	1.1	Abuse of privileges by staff ( <b>insider attack</b> )
			1.2	<b>Unauthorized internet access</b> to the server (enabled for example by backdoors, unpatched system software vulnerabilities, SQL attacks or other means)
			1.3	<b>Unauthorized physical access</b> to the server (conducted by for example USB sticks or other media connecting to the server)
	2	Services from back-end server being disrupted, affecting the operation of a vehicle	2.1	<b>Attack on back-end server stops it functioning</b> , for example it prevents it from interacting with vehicles and providing services they rely on



# Pros and cons to some of the approaches and tools

# Approaches

Work towards self-service approach to threat modeling.

Consider these three basic options for your development teams

1. Minimum viable threat model (the 4 questions) as an acceptance criteria
2. 90 minute workshop to go into reasonable depth
3. In-depth (major release, involvement outside dev teams, e.g. architecture review board)

# Tools

Three options I have used:

- Whiteboards
- Diagramming tools
- Dedicated tools

## Useful Toolkit

Support your approach with tools that allow you to increase your productivity, enhance your workflows, enable repeatability and provide measurability.



---

Examples: Miro

---

MS Whiteboard

---

Pros: Tabula rasa blank slate approach forces thinking

---

Simple

---

Flexibility

---

Cons: They don't generate threats and countermeasures

---





---

Examples: Lucid

---

Draw.io

---

Visio

---

Pros: Shape templates used already by  
architecture teams (maybe standardised)


---

Avoid duplicated effort as can take or  
adapt existing architecture diagrams

---

Cons: They don't generate threats and  
countermeasures

---



## Dedicated Tools

### Examples:


- OWASP Threat Dragon
- MS Threat Modeling Tool
- IriusRisk
- Threatmodeler.com
- SD Elements

### Pros:

- Generate threats (also a negative!, more on that later)
- Lots of features! (integration with issue trackers + vuln managers, automation via APIs, RBAC for collaboration, generate threat models from IaC code or AWS connections, audit trail, threats map to standards like PCI, NIST etc., so can be used to measure compliance)

### Cons:

- Security anti-pattern
- Can be costly
- Effort to operationalise all that functionality and roll it out to developers
- APIs poorly implemented
- Drawing functionality and shape templates not good enough to replace Lucid/Draw.io/Visio + diagram import functionality can be poor (limited sources, end up manually mapping threats)
- UIs were not easy to use
- Questionnaires you fill in add threats not remove them (which matters due to “noise” issue)



Challenges posed by development  
teams that push back

# Meet the development teams

You're an appsec manager or consultant. You get the buy in from senior management in security that threat modeling is a gap. They sponsor your approach to create a new threat modeling process and the attempt to embed within development teams.

You speak to senior management in product engineering and get verbal agreement that threat modeling is a good idea, but it doesn't filter down.

You manage to find a dev team that sees themselves as generous in humouring a threat model, they're doing you a favour. They will listen to the security person, tick that box, and never think of threat modeling again!

Other development teams tell you blankly that they need to be able to justify how they spend their time, and they do not have the "spare" time for threat modeling.



# Why the push back from engineering teams?

- Teams have tough targets, are already busy, want to protect their workflows, lead times, process times, value stream
- It's unplanned work for them when you come asking
- Not in their goals as they see it
- They don't understand it
- No policy mentions they need to do it
- No process established
- Nobody trained on it
- No security champions
- Security team not got strong relationship with developer teams, maybe a bit like a silo
- Security lacks credibility having limited or no colleagues with engineering skills - "what can they teach us?" mentality
- Security doesn't get devops, stuck using excel sheets whereas devs are using sprint planning
- They want a threat model tool to do it for them
- Teams already burdened with security process, and maybe it's not working that well (static code analysis, pen test, audits) lack of "trust"

# Hackers

don't give a shit:

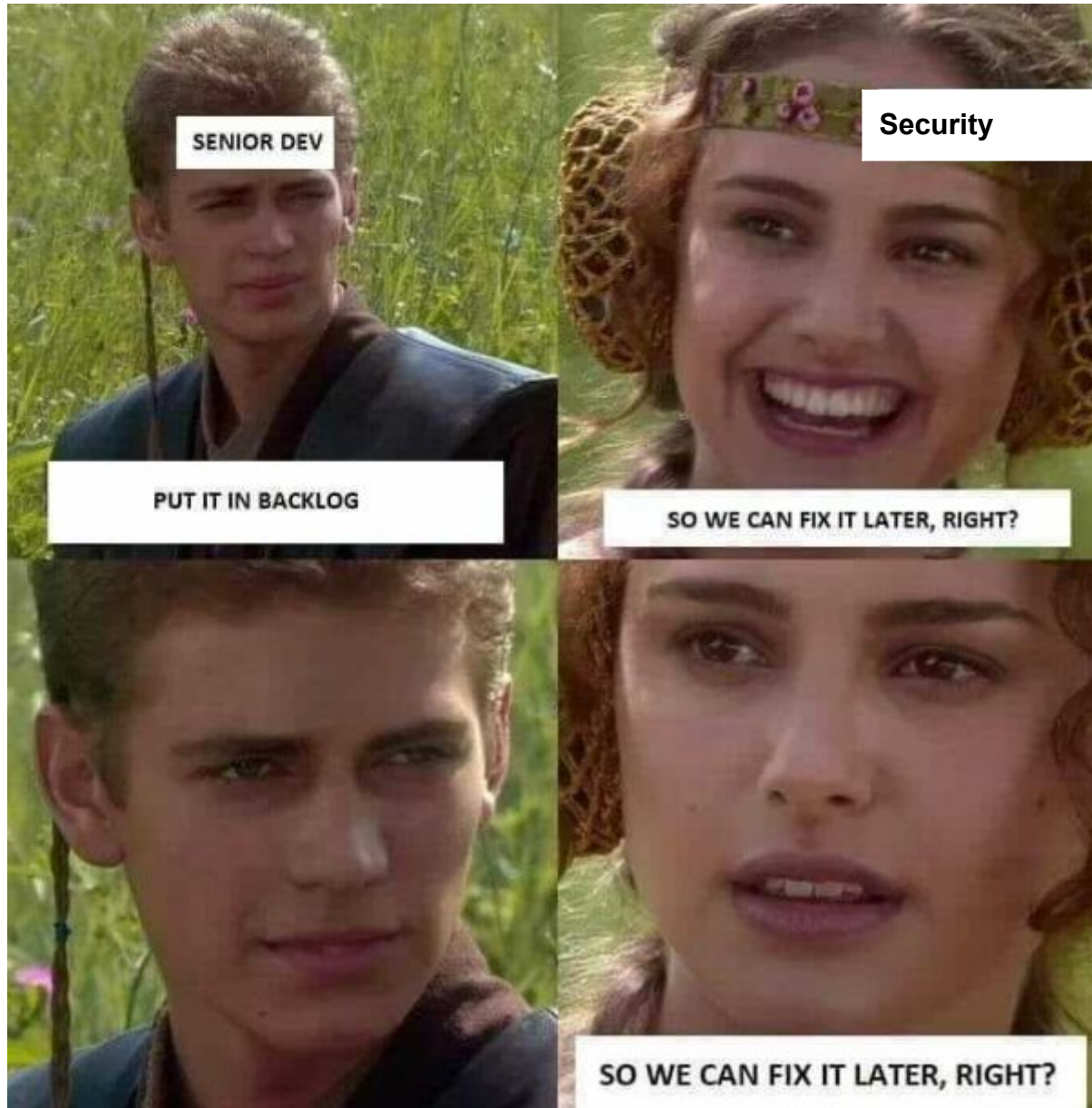


## KIWICON III

28<sup>TH</sup> & 29<sup>TH</sup> NOVEMBER 2009

New Zealand's Hacker con - Wellington

- About your project's scope
- It's managed by a third party
- It's a legacy system
- It's "too critical to patch"
- About your outage windows
- About your budget
- You've always done it that way
- About your Go-Live Date
- It's only a pilot/proof of concept
- About Non-Disclosure Agreements
- It wasn't a requirement in the contract
- It's an internal system
- It's really hard to change
- It's due for replacement
- You're not sure how to fix it
- It's handled in the Cloud
- About your Risk Register entry
- The vendor doesn't support that configuration
- It's an interim solution
- It's [insert standard here] compliant
- It's encrypted on disk
- The cost benefit doesn't stack up
- "Nobody else could figure that out"
- You can't explain the risk to "The Business"
- You've got other priorities
- About your faith in the competence of your internal users
- You don't have a business justification
- You can't show Return on Investment
- You contracted out that risk





How can we respond to these  
challenges for the best chance of  
success



# Tactics to prepare for the best chance of success

- Get top-down buy in
- Make it a requirement
- Have / develop engineering empathy
- Demonstrate the value
- Capture metrics
- Provide the right kind of support
- Ease them into it
- Celebrate achievements
- Avoid security anti-patterns
- Lean on vendors for existing tooling