To DevSecOps or not to DevSecOps: is that a question ?

Using an Archetype-based model of Security in DevOps

Mario Platt
Strategy Director

OPEN SECURITY SUMMIT

Today's Agenda

- The 2 schools of thought
- Who is it for ?
- DevOps Security Archetype model
- Meet the Archetypes
- Grappling with Constraints and Bottlenecks
- Helping Archetypes - Governance and Maturity
- Helping Archetypes - Team Topologies
- Do we need DevSecOps or not, then ?

There are 2 schools of thought with regards to DevSecOps and within or according to the Infosec industry. These are all people I highly respect and look up to as well, but I believe there are underlying themes to both sides of the argument.

Those advocating FOR DevSecOps have arguments relating to collaboration between different teams and as a ‘point in time’ need whilst Engineering teams build security into their practices.

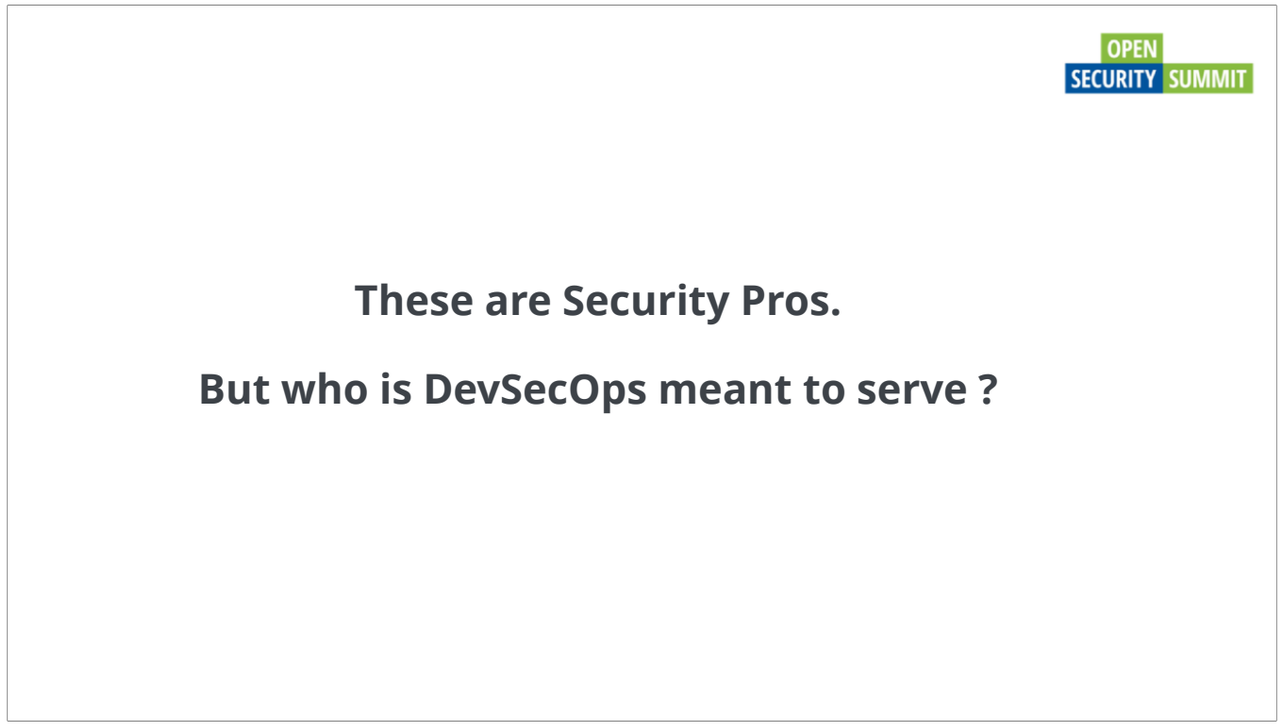
Those less impressed by the USE of DevSecOps seem to focus their arguments on the possibility for silo-ing of activities, excessive marketing buzzwords and fear in relation to current role identities and focus on the fact that security needs to be in the hands of Engineering because that’s where it’s needed, and not in some Governance function detached from development where ownership is not effective (the doers must own and have agency over scope and outcomes)

What DevSecOps isn't

There is also a 3rd (less common) view that DevSecOps is Compliance too, and I couldn't agree less with that.

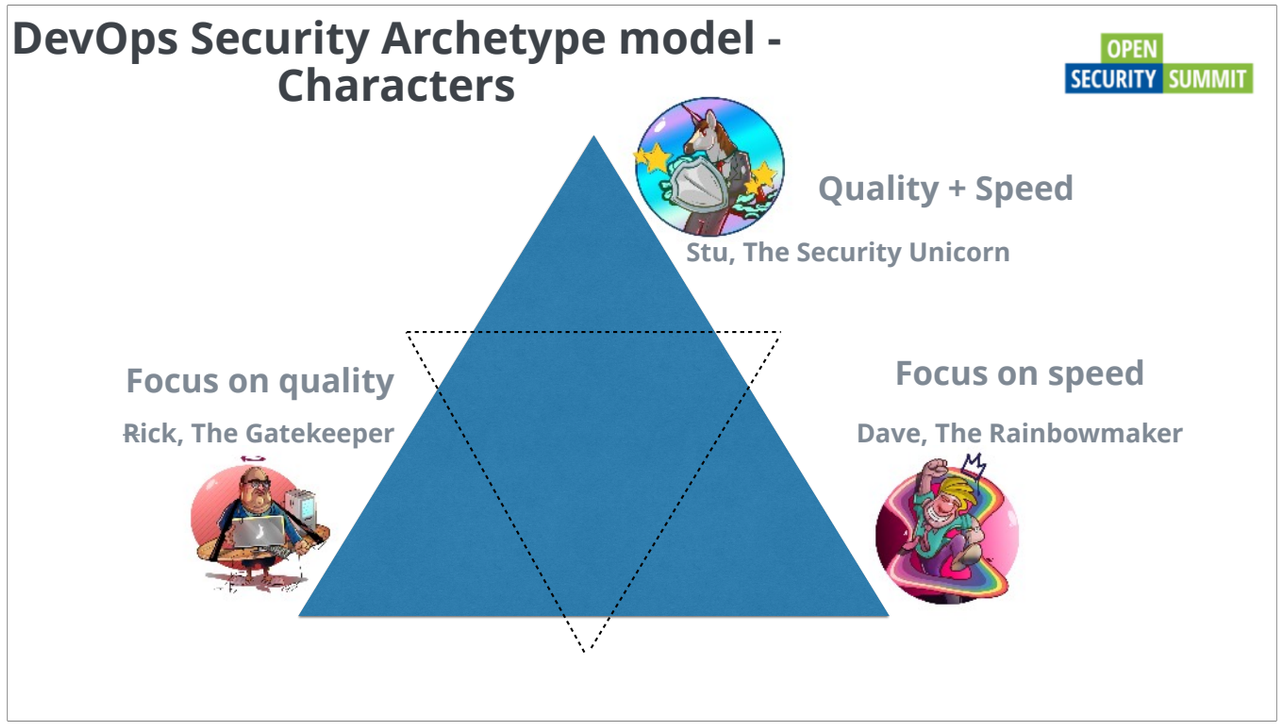
No one called Security Engineering 'Compliance' 10 years ago, so we shouldn't be starting now either. It's good that or if we're doing it as it expectedly make Compliance easier but they're not the same thing, and with poor communication can be completely detached from each other.

It can only become or help Compliance once we have effective traceability between policy requirements and technical controls, and when communication between the different stakeholders allow for one part of the process to inform the other during both design and operations.



But this is the opinion of Security professionals. We may be the ones with bigger interest in DevSecOps (if there’s such a thing) but we’re not it’s only customers, I don’t think. Who are we to or who can we serve with such a term ?

I believe it can be beneficial to both Control & Governance functions, which Infosec is but one of them (others would be Finance, Service Management, Programme Office etc) and also to the Engineering organisation, but it will serve different sometimes even opposing objectives so the model we'll talk about next is meant to provide some basis for meaningful conversations between these stakeholders.



Meet the ‘DevOps Security Archetype Model’. This was built based on exploration with peers and experience in consulting to organisations of all sizes, from startups to government and healthcare institutions and appreciating the differences between them when developing a tailored strategy for their needs as a business but also from understanding what the Security Unicorns do different and how they approach the same challenges.

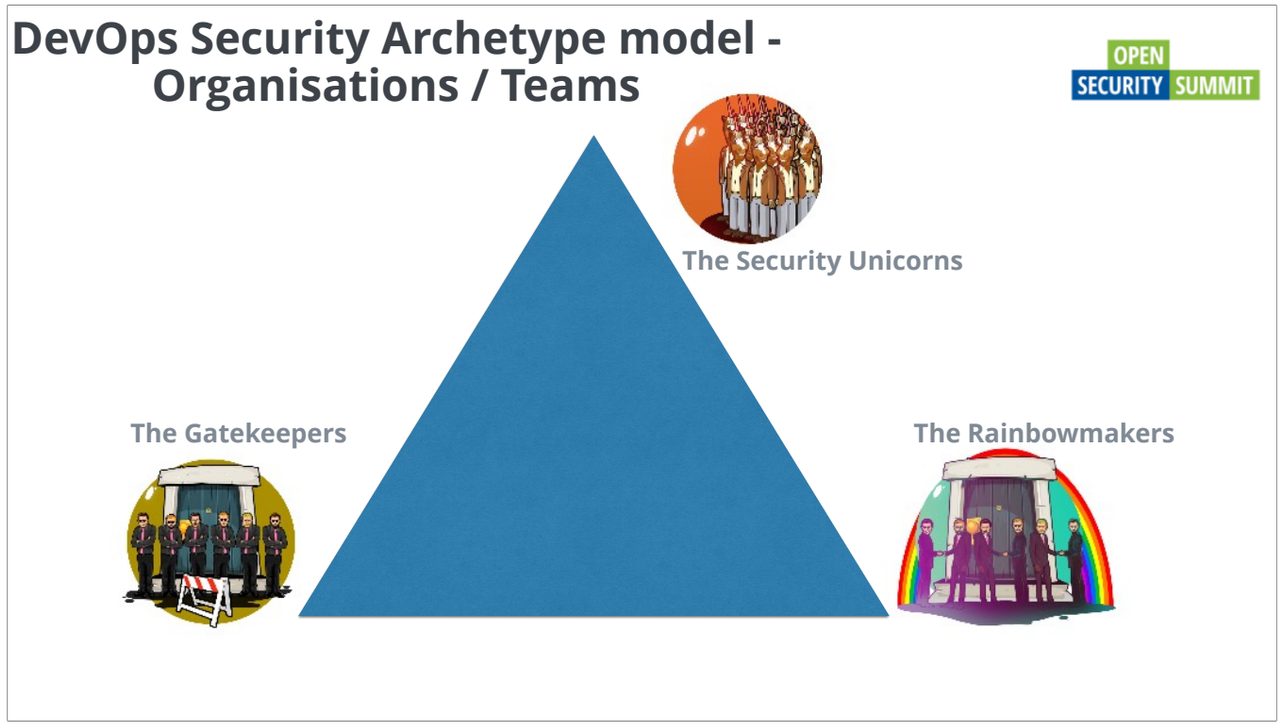
We have 3 basic archetypes in this model.

Dave, The Rainbowmaker or magic maker is a reference to a cartoon that existed for over a decade, where we have in a boxing ring all the shiny security controls and on the other side we have Dave who clicks all the links and finds creative ways of avoiding security. Dave is measured and rewarded on delivering features and stable platforms, so has an overall focus on Speed but also stability and ensuring he can support the business in making money and be successful whilst removing any impediments to this speed, though not necessarily in a reckless way.

Rick is a reference to how most other people outside of Compliance tend to perceive control functions particularly Infosec which still has a big reputation for being the department of ‘No’ (everyone else usually refers to him by changing the first letter of his name to something else, I’m sure you can guess what), and for the creation of policies and procedures he expects are adhered to at all times in order to keep the organisation compliant to a standard or at least to be in a position to evidence due-diligence has been performed and help manage cyber risk, though these are often uncontextualised and there’s no real understanding of the realities of business operations and the impact these rules have. Rick is measured and rewarded for identifying and managing non-compliances, passing audits successfully and generally for helping avoid bad outcomes so there’s an overall focus on quality (a biased and narrow view of it, at least) of the things being produced.

Finally, we have Stu The Security Unicorn which is a reference to those individuals who have good embedded security practices in their engineering processes and have ownership and agency over security in the products and services they’re accountable for. Security is generally perceived as another element of their quality assurance, as

performance, cost and other metrics also are and security is never “something someone else does”. They’ve figured out how to ensure that quality and speed are not mutually exclusive, but indeed mutually supportive.



This model is applicable not just to individuals, but also to the way the organisations operate in the same fashion.

We often find ourselves in a company which operates in a very Gatekeeping fashion, but where multiple stakeholders are mostly Rainbowmakers. And particularly in startups, and when they fail to account for cultural-fit of security people, have whole organisations and processes made of Rainbowmakers and a few Gatekeeping stakeholders likely to have some permanent frustrations with lack of their preferred form of Governance.

The Security Unicorn organisations are those who figured out how to do security at scale, such as the Netflix's, the Facebook's, Adena Health, Github and many others that a lot of us praise

This model is meant to help us think about their needs and expectations, and hopefully create targeted plans to help them improve their outcomes and decrease frustrations.

One thing to note about the model, is that many will default to seeing all Gatekeepers as Governance-focused individuals and all Engineers or Designers as Rainbowmakers, which isn't necessarily the case. In practice, you'll find Governance people who have characteristics and largely align with the Rainbowmaker archetype and vice-versa. Also to note, that the characteristics and constrains of each archetype are defined as the extremes or edges of the diagram, and usually reality will be more nuanced.

Meet Dave and his team of Rainbowmakers

Dave and the teams aren't negligent or stupid.

They lack **situational awareness** to do better.

Process, social practice and cognitive load considerations are failing THEM

A big part of how Rainbowmakers are perceived by Gatekeepers is claims that they're either negligent, stupid or just plain don't care about security. It fails to realise that no Developer creates insecure code deliberately, we should assume people mean well and produce the best quality artefacts with the current knowledge they possess, visibility they have and overall constraints (time is just one of them and so is culture) imposed by the demands of their practice in their organisational context.

The main challenge relates to lack of situational awareness with regards to the security of what they produce, and this is mainly a process and social practice issue and failing to account for cognitive load of their already difficult practice, and it's those that are failing them.

Meet Dave and his team of Rainbowmakers

Essential characteristics:

- No integrated security telemetry
- Avoids engagement with Compliance
- No secure baselines or modelling of threats
- Limited automated testing overall
- No product level security reporting
- Security is someone else's job, or output of pentests/audits

Some of the essential characteristics will include lack of integrated security telemetry (they don't have any tooling that can help them with identifying vulnerabilities they should address).
They will generally avoid engagement with Compliance or Infosec teams, as they tend to misunderstand the context or even the scope of what their job is. They'd even go so far as to suggest Compliance doesn't understand what it takes to get their job done and experience is one of mandating things, as opposed to any real collaboration. They only see Compliance when preparing for audits or on annual awareness courses
There are no agreed and secure baselines in place or modelling of threats of the systems being built, so there isn't a shared mental model by the team on what can wrong with the system and what measures they can put in place to prevent bad outcomes
This is not a security thing per-se, but there's usually limited automated testing overall so the practices which could be leveraged to integrate security into operational processes are usually missing too
There's no product level security reporting, so security is seen as an "organisational thing" which leads it to being abstract, reactive and uncontextualised as opposed to being a "product thing" where teams understand and manage their backlogs according to relative prioritisation amongst all the other functional, bug and improvement work
Security is also generally perceived as someone else's job, and not something that the teams need to manage. There's the expectation that, at best, someone needs to give them requirements and they need to work towards delivering them. Or worse yet, security being generally absent through lifecycle and seen as "the things you need to fix as part of the output of the pentest or audit"

Meet ~~R~~ick and his team of Gatekeepers

Dick and the teams aren't thick or business averse

They **lack understanding of control reliance** in a DevOps world and have **inertia due to past success** of their current model

Team Topologies and poor traceability from technical checks to Compliance objectives are failing THEM

One of the big complaints I hear from Rainbowmakers, is that the Gatekeepers are either thick, business-averse or just plain difficult coming up with all esoteric scenarios of what can go wrong or how the world is going to end in the next 3 months.

What more likely is happening is that Gatekeepers lack understanding of how control reliance, particularly the form and appropriateness of how to build security controls in a DevOps environment, actually works and how that affects the software development lifecycle and what automation can mean for their practice, but also about inertia to change due to having success with how we did security in on-prem non-continuous integration environments. Most of the principles are still applicable, but they have different forms which doesn't fit their current model of good security.

Here an issue of team topology and understanding of their evolution and also poor traceability between what Engineers do and Compliance objectives are and that's what is failing them.

Meet Rick and his team of Gatekeepers

Essential characteristics:

- Gated processes supported by committees and Review Boards
- Limited understanding and trust of modern development practices & engineers
- Policies and standards developed outside of Engineering context
- Lack of technical knowledge, or worse, outdated threat models

Some of the essential characteristics of Gatekeepers is they usually have or expect Gated processes and meeting with Review Boards to express their happiness or have the opportunity to effectively block something. Most organisations claiming Agility and DevOps still try to embed all these committees and gates in some fashion.

They'll have very limited understanding of modern development practices (there's a lot of rubbish guidance out there) and they neither trust their engineering teams (with claims of them being Cowboy's happening often) nor have any trust in automation and the benefits of short feedback loops ie not matter what you do or what evidence you can provide, they would still prefer to have a pen-tester manually check your app for every release if they could get away with it.

Usually they'll develop policies which are verbatim copies of security standards like ISO 27001 AND ISO 27002 (which is the real problem because it goes into too much outdated detail), without any understanding or consideration for the context they live in and these are usually written in ways that Rainbowmakers can't make sense of or understand what it means to them in their context, which removes or contributes to not having any sense of agency and ownership

As we had a recent paradigm shift and co-evolution practices in software development, they'll usually lack the technical knowledge to fully understand patterns in use and think of Cloud systems and orchestration with a mental model for on-prem.
They haven't understood how Distributed systems lead to availability, how immutability relates to integrity or how designing for ephemerality can support confidentiality objectives

Meet Stu and his team of Security Unicorns

Stu and the teams are..... OK :)

They have a great Engineering culture and are a learning organisation.

Compliance colleagues **don't work in silos**, but **collaborate to establish process and then trust** that the teams are focused in developing quality software

Stu and the team of Security Unicorns are OK. They have a great engineering culture and are a learning organisation. By this I mean it in the form Andrew Clay Shaffer refers to it as "organisations become graduate studies in the skills they require to be successful". So it's not about Rainbowmakers or Gatekeepers being good "generic experts" in their own functional silos, but on learning and having the requisite know-how and materials to make your company successful in its current sociotechnical context

There's true collaboration to establish good practices and trust that the teams are rooting for the same objectives and business benefits.

Meet Stu and his team of Security Unicorns

Essential characteristics:

- Security is embedded into DevOps practices
- Security is an element of product/service quality
- Teams keep up to date threat models of what they build
- Their process has the right security at the right time.
- Compliance IS Code
- Less Command and Control from Compliance, more collaboration

For the unicorns, Security is embedded into their practices and not seen as something separate from product or service quality. Operating models such as SRE help make this clearer to teams, but not a necessity.

Teams understand the threat models of their applications and failure modes, and do work to mitigate occurrence of incidents and mitigations for potential impact.

Their process considers security end-to-end (what should be feedback at IDE level, how much security and which tests should run after every commit as opposed to in a QA environment alongside full integration tests where there's more time for further validations, what conditions needs to be met before we expect a build to fail on a security issue and can the developers manage false positives from their own code repositories)

Compliance IS code, in that Gatekeepers don't tell or send a spreadsheet that Rainbowmakers need to fill in but the security requirements are expressed as code and tests that software needs to pass, as with other elements of quality.

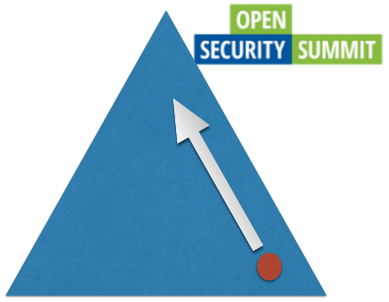
There's less of a command and control, Ivory Tower and hiding behind the keyboard type approach from Compliance teams and more true collaboration and even guided adaptability where constant iterations build up the overall security posture over time. THey're not constantly fighting an "audit-gap", they're iterating to make their products more secure

OPEN SECURITY SUMMIT

But in their current practices, all are affected by Bottlenecks

Bottlenecks to going up the triangle

- **Lack of Security Telemetry**
- **Product Management security prioritisation**
- **Security Expertise by DevOps and Software Engineers**
- **Poor Process assurance and practices**
- **Low sense of agency and ownership for security**



I'd argue the number one challenge for Rainbowmakers is the lack of Security telemetry in the form of a baseline.
In many organisations this covers some of what we would consider basics such as dependency scanning, static code analysis, secrets scanning and basic configuration hardening
Another bottleneck is usually in the form of Product visibility of security work. You talk with Devs and Engineers and they all know dozens of security issues they'd like to address, but they're not on a backlog nor is there any periodic review or clear alignment to business goals or attributes, nor can you use metadata to filter only for security work.

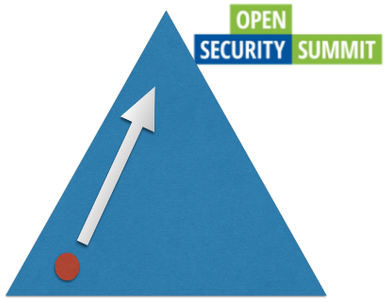
Security expertise is also often missing, either because it's not even identified as a need or in places where it is, an overly optimistic and unrealistic expectation that all Engineers or Devs need to be security experts. That fails to account for issues of cognitive load, incentives and business priorities and generates pushback on taking responsibility. Bland vanilla approaches to security awareness make this worse, not better.

Poor process assurance and practices, in that the teams don't regularly perform the types of practices that increase the likelihood of creating secure software, like threat modelling for instance and/or creation of tests to validate continuous working of security controls

Low sense of agency and ownership for security which is coupled with low expertise and lack of telemetry, but a problem in and of itself which is created both by organisational design and the separation of duties relating to security.

Bottlenecks to going up the triangle

- **Assignment of security responsibilities**
- **Spending money on the wrong things**
- **Gated processes and out-of-band approvals**
- **Control reliance mismatches**
- **Policy ISN'T Code or Stories**
- **Command and control culture**
- **Inertia due to past success**



For the Gatekeepers, one of the main challenges is around team topologies and the assignment of security responsibilities. Most of these are unbalanced with the needs of agency and autonomy of modern development practices and unfortunately we see the non-working extremes more often than not. Either Gatekeepers take too much onto themselves, they don't stick to setting control objective only and own all policies procedures and governance and thus become bottlenecks and the guidance becomes uncontextualised and often wrong for development or on the other extreme they just delegate the whole lot and assume the security know-how and practices exist where they don't and provide no support to building that know-how.

I still see more security budget being spent outside of the realm of the development process than for initiatives that would help with Developer Experience of Security. If it's not helping your frontlines, anything you do will be more ineffective and cost more to fix later.

Gated processes and out of band approvals are also typical expectations, and the usual expectation of forms and spreadsheets to be filled out by Dev/Engineering teams doesn't support process integration or collaboration

Control reliance mismatches in that particular types or forms of controls are expected but may not be present, and also relates to issues of trust ie they expect a commercial product when native capabilities achieve similar benefit

Security policies aren't written as code, or at least expressed as User Stories and in that way enabling better communication between parties

There's also a Command and Control, Ivory-tower type approach of being behind keyboards sending emails and expecting that teams drop what they're doing 'because security is important'. They see themselves more as requirements setters than collaboration partners

Inertia exists due to success of their past model, but having been good at securing on-prem environments doesn't necessarily make you qualified to secure an architecture of microservices and orchestration. Yes, some of the base principles apply but forms vary widely and many haven't kept their skills current or rely on their teams to educate them. There's a lot of product-driven bolt-on security of the past which is a significant impediment to integration of security practices by DevOps and Engineering teams.

There are some further challenges we'll discuss next

Communication between these stakeholders
Traceability and process which support them
The learning journey and mental models they currently have and how they need evolve
Evolution of practices in their organisations
And finally the team topology

The Communication Challenge

Their management systems

The Stories they tell and understand

The language used to communicate concerns

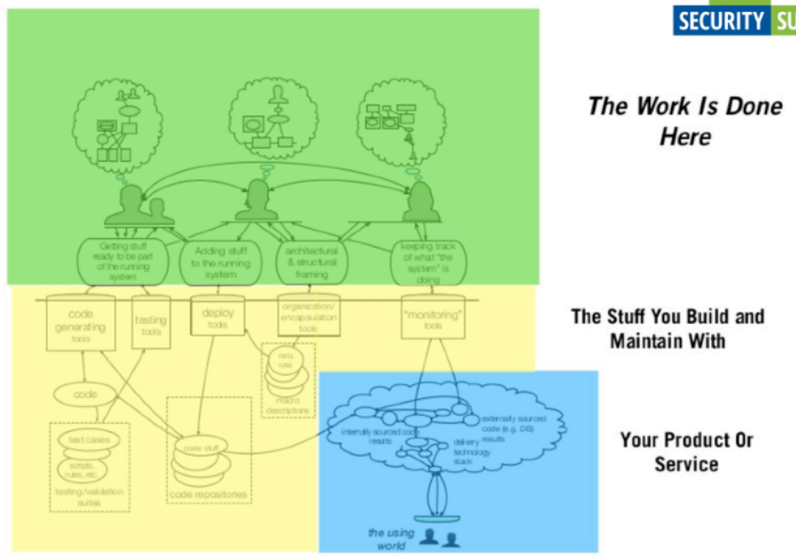
In order to enable better communication we need to consider or understand the mental models of the different stakeholders, which often align to the structure of their management systems and how we can use those more effectively to communicate

The timespan and types of stories different archetypes use and tell each other

And the language they’re using to communicate their concerns

They see the system differently.

They say the same names, but they mean different things



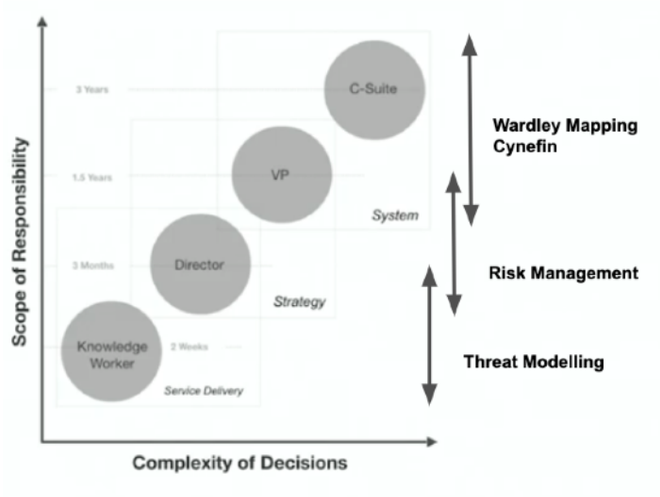
Credit: @allspaw https://www.slideshare.net/jallspaw/resilience-engineering-a-field-of-study-a-community-and-some-perspective-shifting

This slide is in reference to Johh Allspaw's material in Resilience Engineering, in that different stakeholders may talk about what their systems are through the lens of their mental models, the things they build their systems with and the actual products and services provided. They'll use the same names, but their understanding of the system will be very different. This isn't a bad thing though, this heterogeneity and diversity makes organisations more resilient and should be encouraged, but it can introduce challenges of communication

Complexity of decisions vs Scope of responsibility

The timespan of their narratives are different.

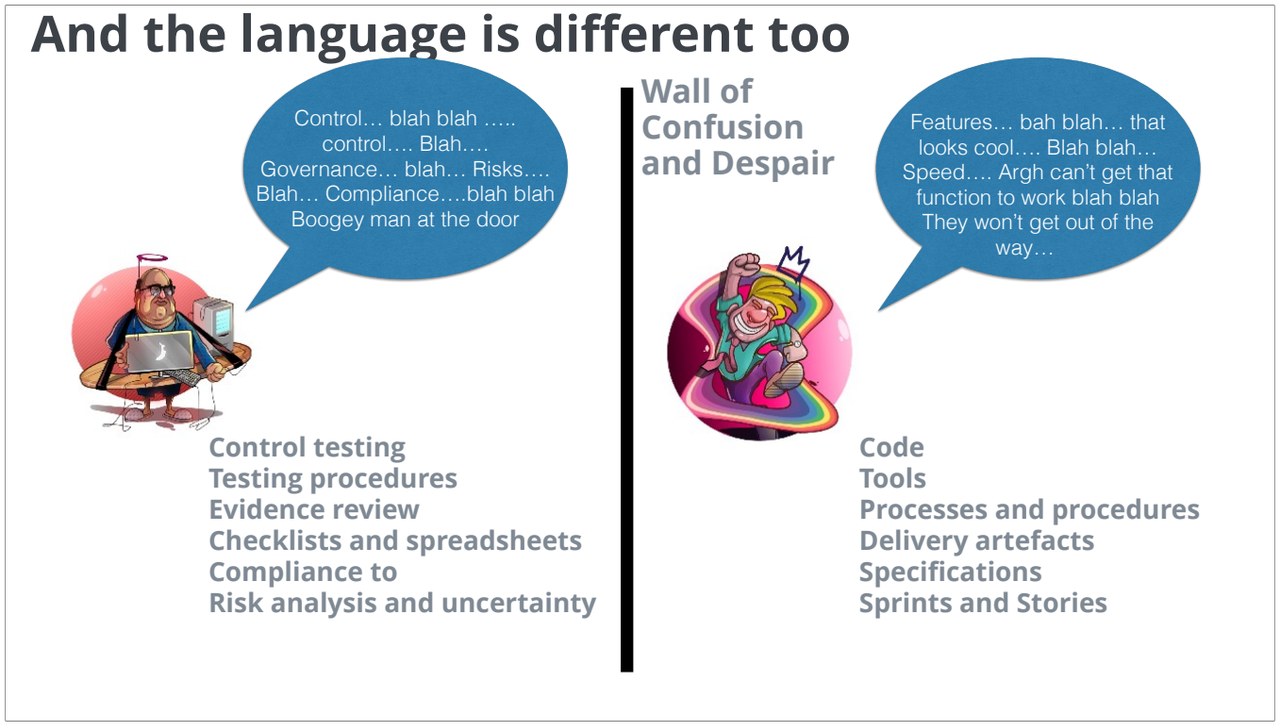
Sprints vs managing risks



@cyetain Jabe Bloom - 'Whole Work: Sociotechnicity & DevOps' - https://www.youtube.com/watch?v=WtfncGAeXWU

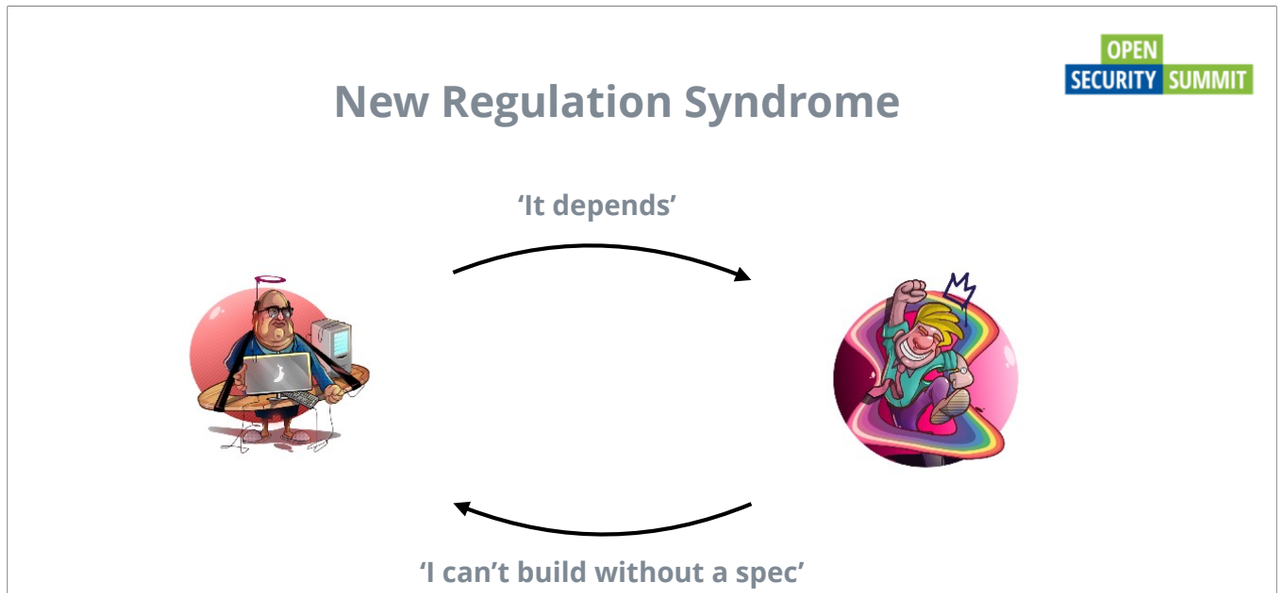
First The timespan of the narratives are different. Gatekeepers talk a lot about risk, so uncertainty and possibility, yearly or multi-year cycles of transformation and change. While Rainbowmakers like to understand what they can do here and now, over the coming sprint or few months at best. Connecting these 2 types of narratives is not trivial, but part of the work that needs to happen to enable better communication and cooperation

And this is where adoption and connection of practices of Risk Management and Threat modelling can help make this connection

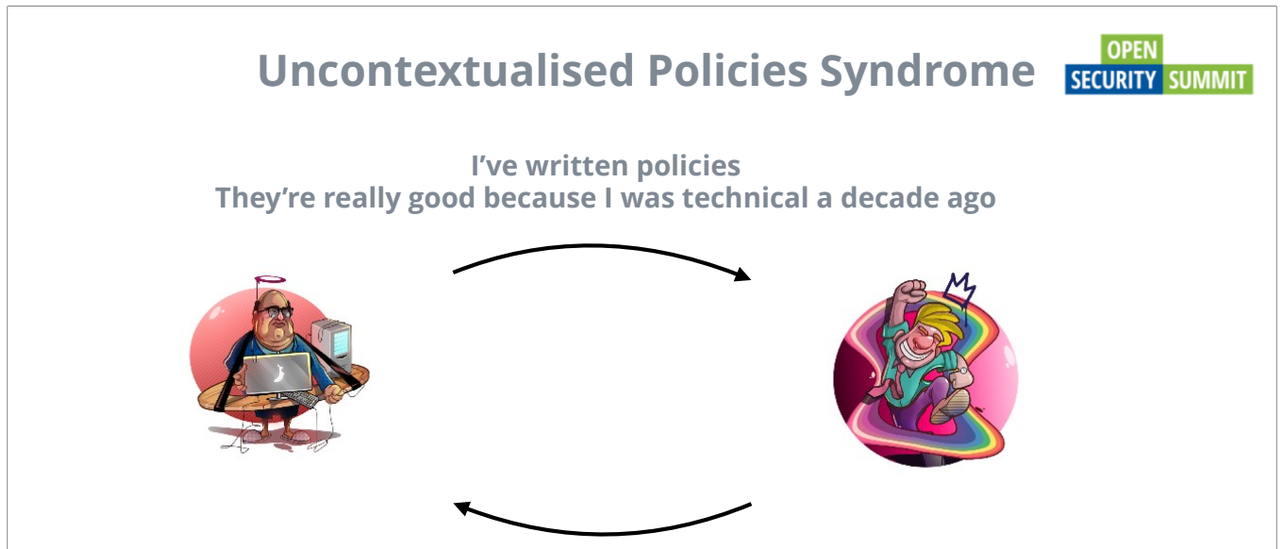


THe language these stakeholders use is also largely different and not conducive to direct collaboration often. Something often requiring Boundary spanners (individuals or roles to help break that communication gap, as they have good understanding of both domains). Missing those, and even with those, we should aim to make this language more traceable if we can't make it shared, which should be the goal

Two syndromes usually happen as a result of this communications mismatch



New regulation syndrome (seen it happen with GDPR and MDR for instance) in that Gatekeepers like to say, as any consultant, “it depends” but Rainbowmakers actually need a specification. So that’s not helpful



And also what I like to call ‘uncontextualised policies syndrome’, where Gatekeepers write ‘pretty words on paper’ that have no chance of being made operational without significant operational impact, or even completely miss ‘what it takes to get the job done’ from the point of view of Rainbowmakers.

These policies typically live in Word documents or spreadsheets, that Rainbowmakers hear about on the annual awareness training but couldn’t actually find them if their lives depended on it or even understand what they mean to their daily practices and how they develop and release code

DoD Enterprise DevSecOps Reference Design

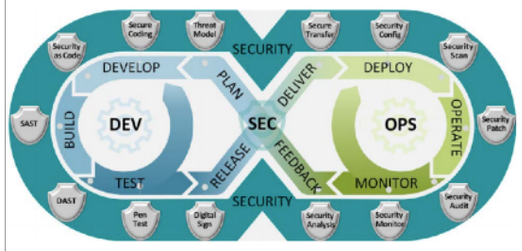
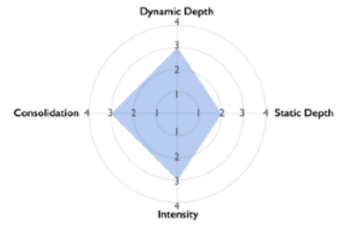


Figure 3: DevSecOps Software Lifecycle

Hysn DSOMM



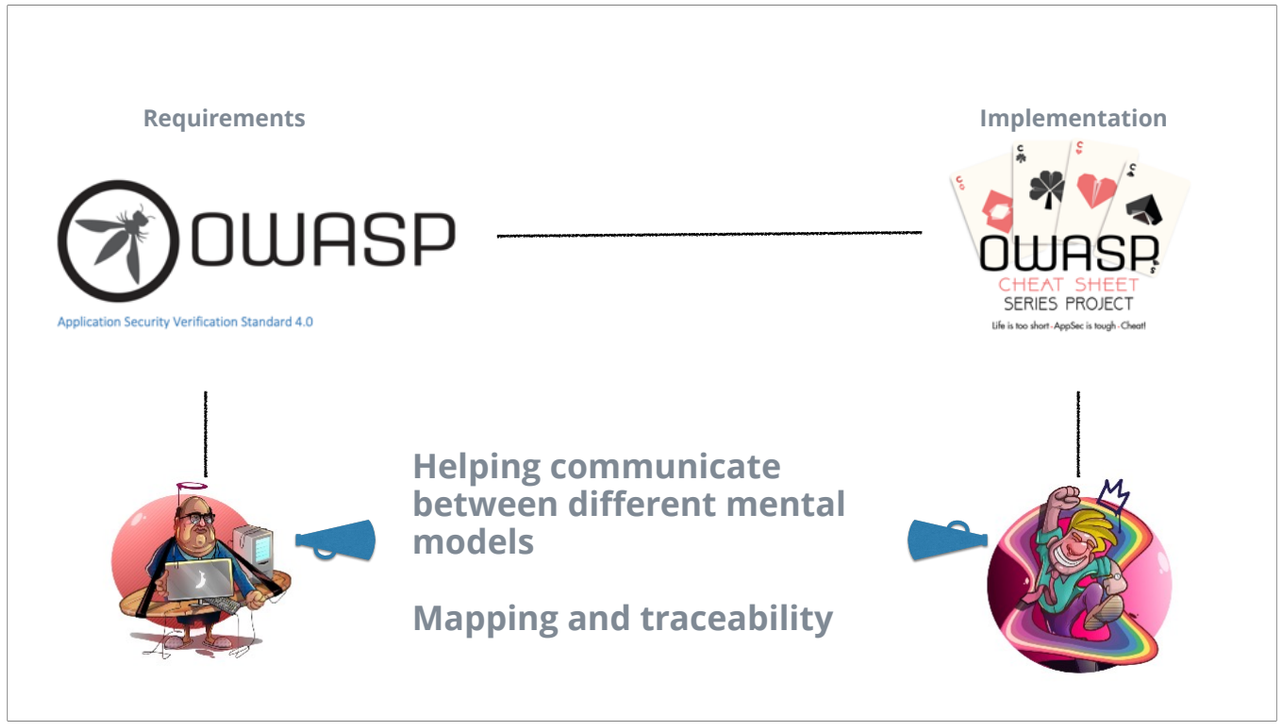
Helping communicate between different mental models

Mapping and traceability

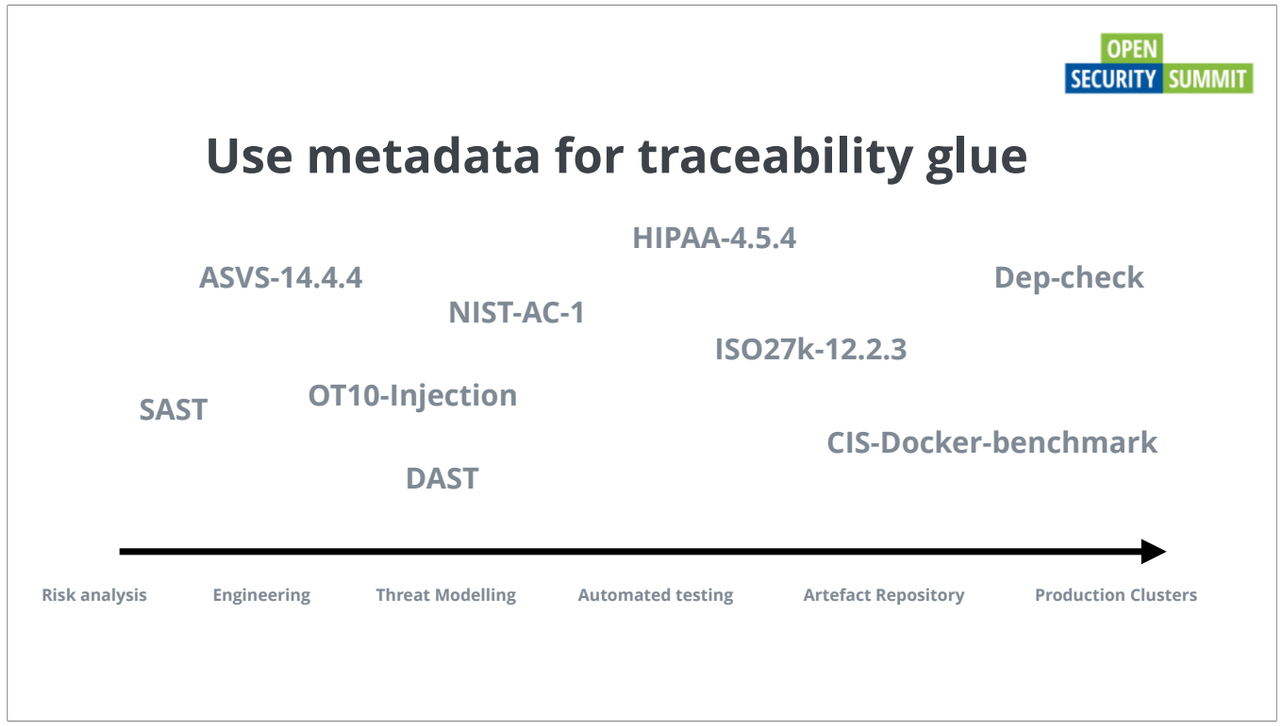
For this, we should think about Governance models happening at different timescales, exposing security concerns in specialist appropriate language that can work for both parties. It’s about using appropriate methods.

If the Gatekeepers are better used to ticking boxes and having overall governance frameworks that enable clearer auditing, let them use them. Something like DoD Enterprise DevSecOps Reference can work well. Just don’t try and shove it down the Rainbowmakers throats if that’s not how they see their practice, so something like DevSecOps Maturity Model which is iterative in nature may help give the Rainbowmakers what they need and a clear path that they can spec and iterate around and be happy about owning.

Then let’s map it and trace it, to see where each area is and what next steps they can give forward that are mutually beneficial.



Another example is through mapping something like a checklist-formatted standard such as ASVS which can help, and can be mapped to OWASP 20 Proactive Controls or OWASP Cheat Sheets so the Rainbowmakers can have clarity on what the Gatekeepers actually mean and what they need to do about it, with a clear specification and code snippets they can leverage.



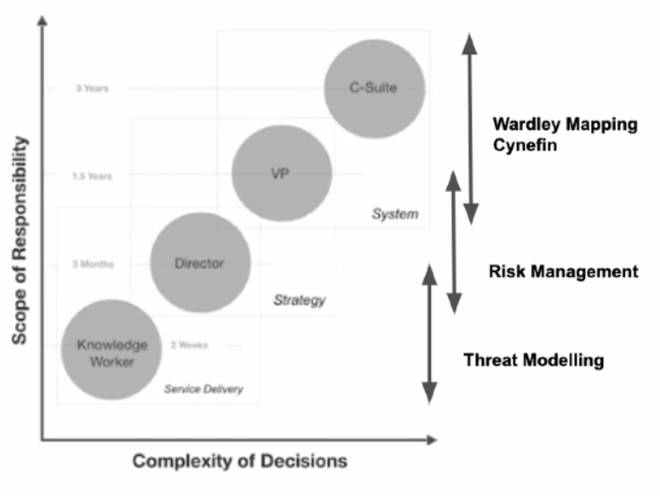
And we need to get better at using metadata to connect these together. Gatekeepers and Rainbowmakers keep looking at different parts of the process for their assurances (what tests are we running, what standards are we meeting, what hardening to we have in place) when in 2020 there's no reason why we can't just query our operational environments for metadata and get an accurate picture of all the security validations which happened through the process so we can all look at the same artefact and have meaningful conversations

But how do we connect them ?

Threat Modelling

Risk Management

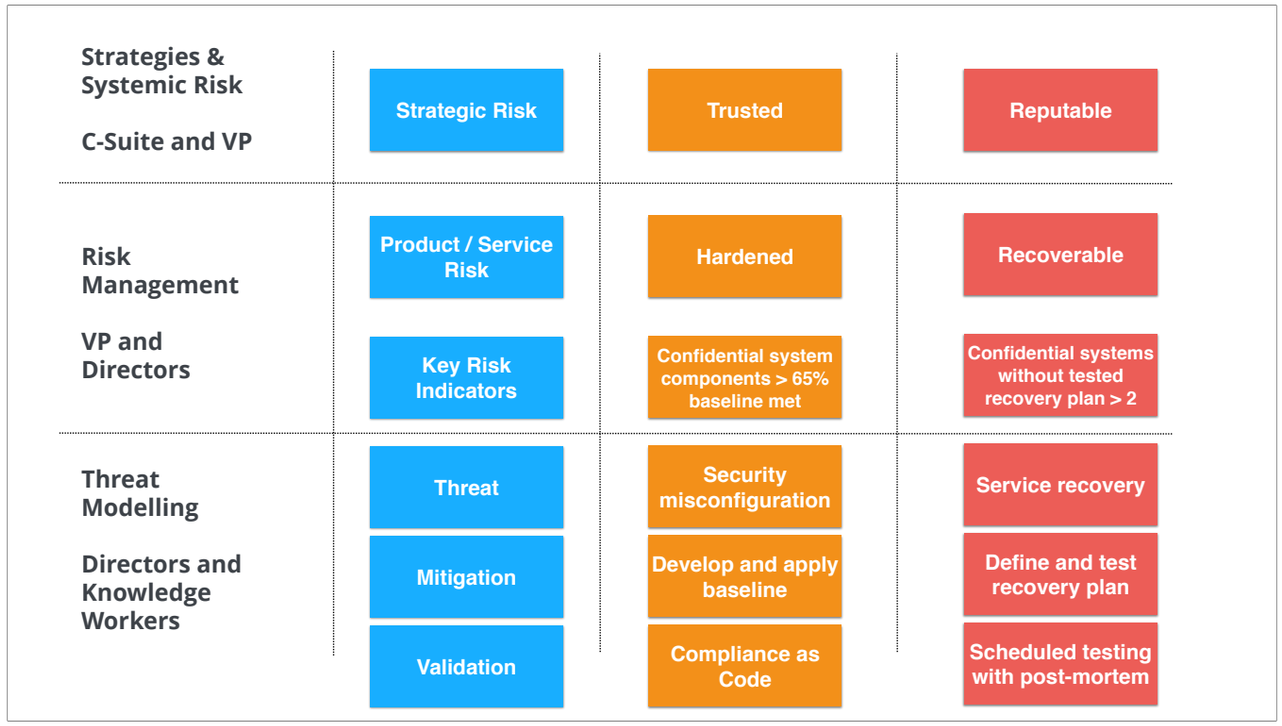
???



Jabe Bloom - 'Whole Work: Sociotechnicity & DevOps' - https://www.youtube.com/watch?v=WtfncGAeXWU

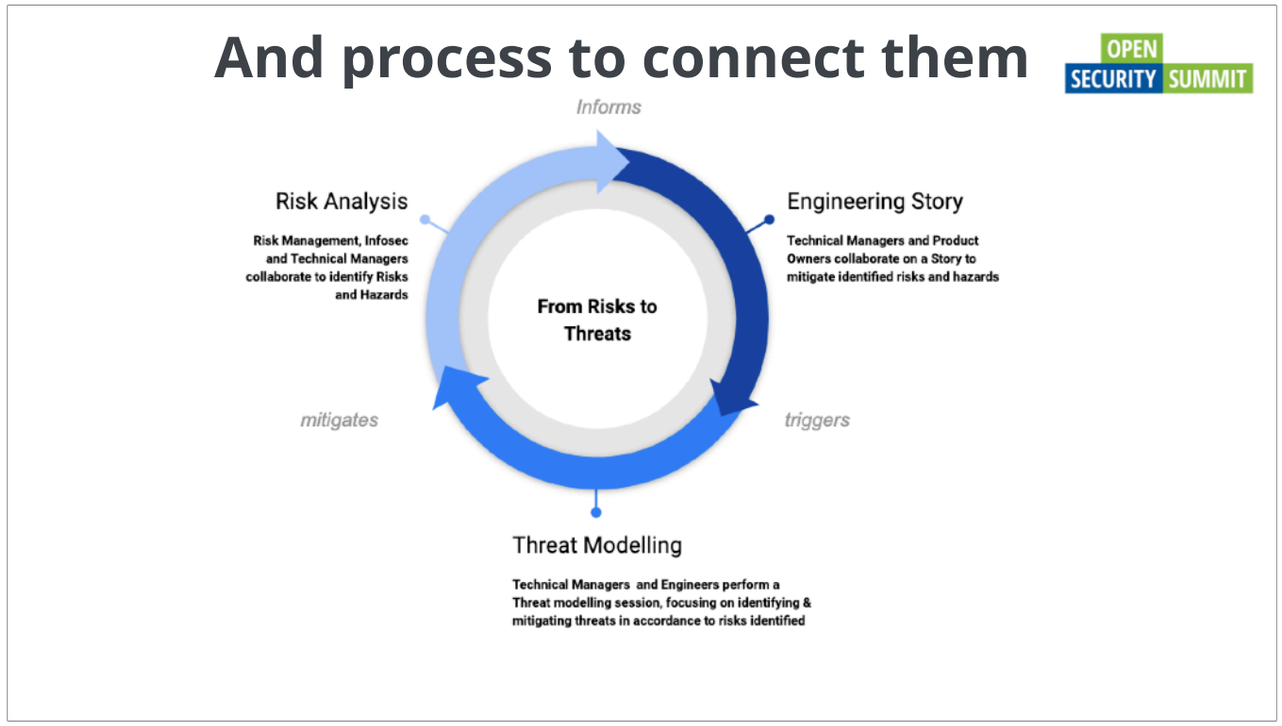
https://medium.com/@marioplatt/social-practices-and-timespan-of-discretion-in-cyber-security-cef4fdc16663

The question then is “how can we effectively connect the practices between risk management and threat modelling”



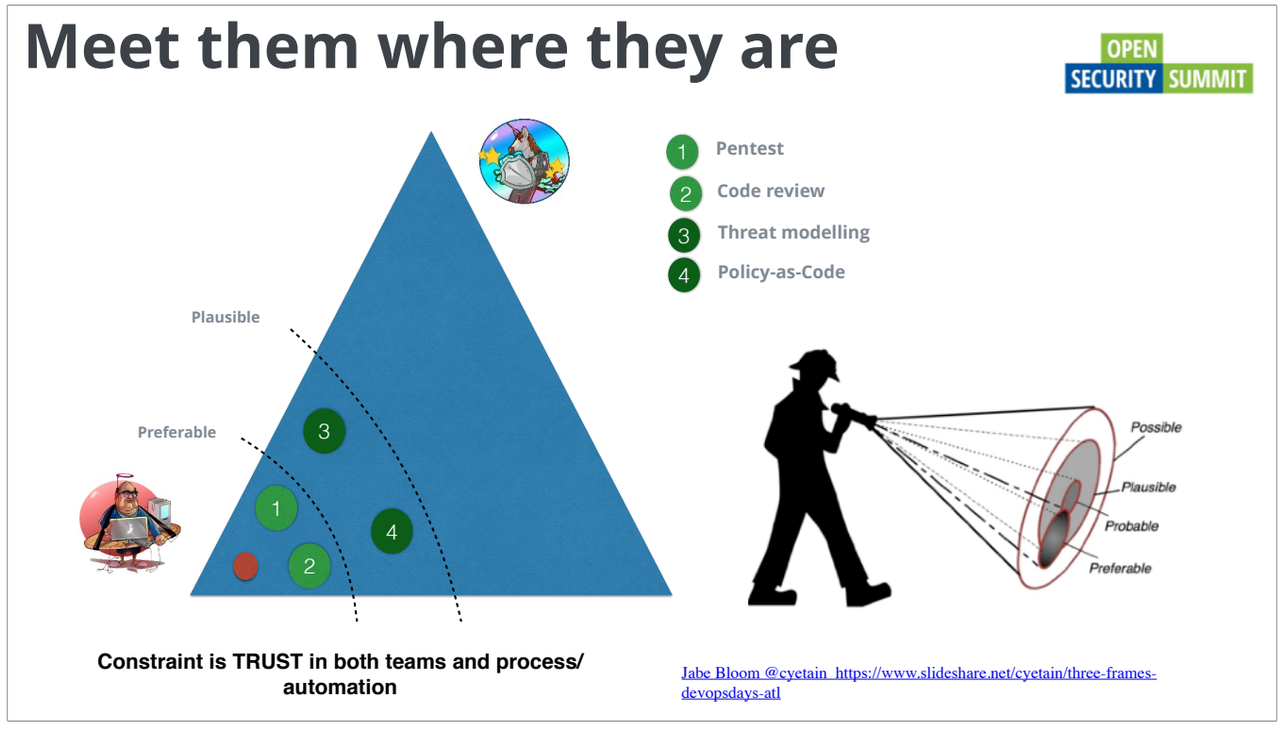
Gatekeepers need to think more about Key Risk Indicators that can be agreed at the Product and organisational level (collaboration with Product Owners is key for this), which can then help inform scope and focus of Threat Modelling sessions and what it means to threats we look out for, mitigations to put in place and what will give us the validations (preferably automated) that our controls are still in place. This is what will allow the security program to scale, and doing it once and having continuous validation that controls are present and working as we expect, and that we have telemetry in place and that can let everyone know when mitigations are missing or security tests which are important have failed

And as with SLOs, these should be highly contextualised to the Product it relates to and not some generic organisational thing



And we also need a process to connect these. More often, we have Risk analysis immediately define mitigation plans which may or may not be considerate of operational constraints or trade-offs. Part of having teams own and have agency over the security of what they build, means that we need to give them the opportunity to contribute and aligning this with development lifecycle means we need to have stories attached to them that are part of product roadmaps, and not just ‘left-field’ asks by Gatekeepers

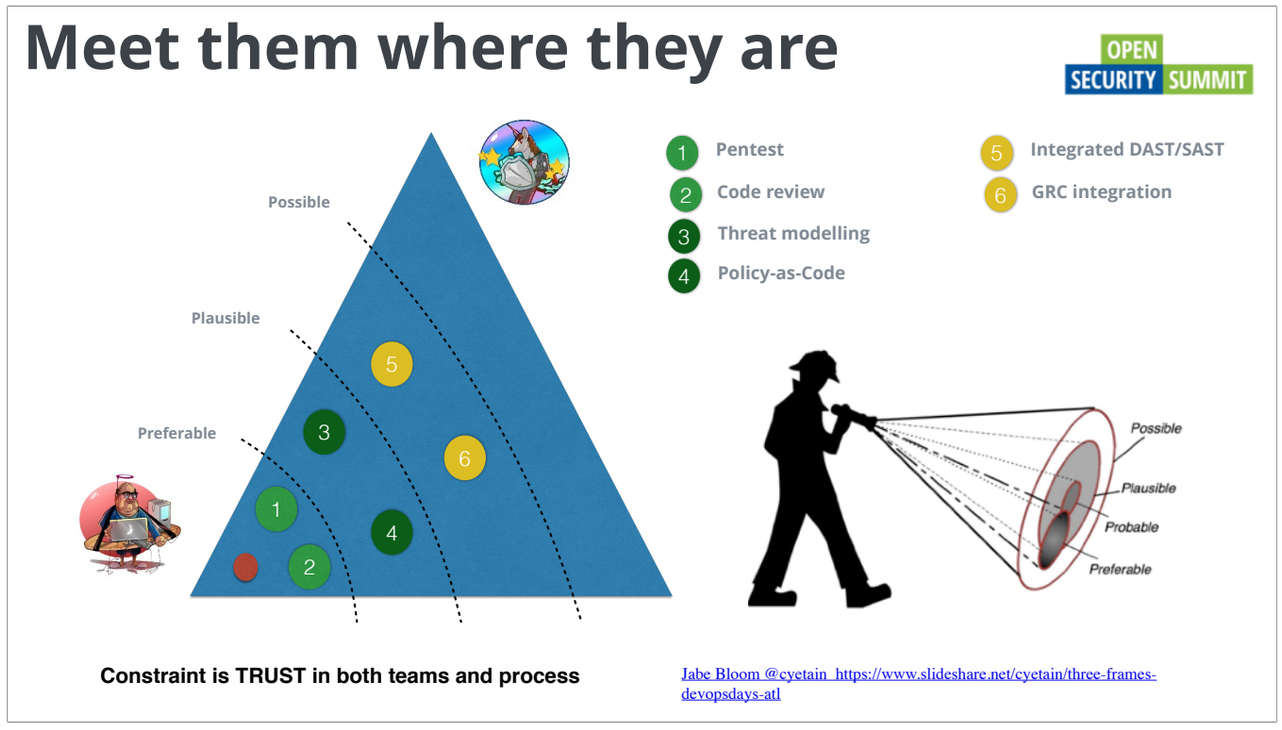
A cycle of performing risk analysis to inform engineering which then triggers targeted threat modelling and which the mitigations put in place can be traced back to the risk treatment plan and addressing the outcome of the risk analysis



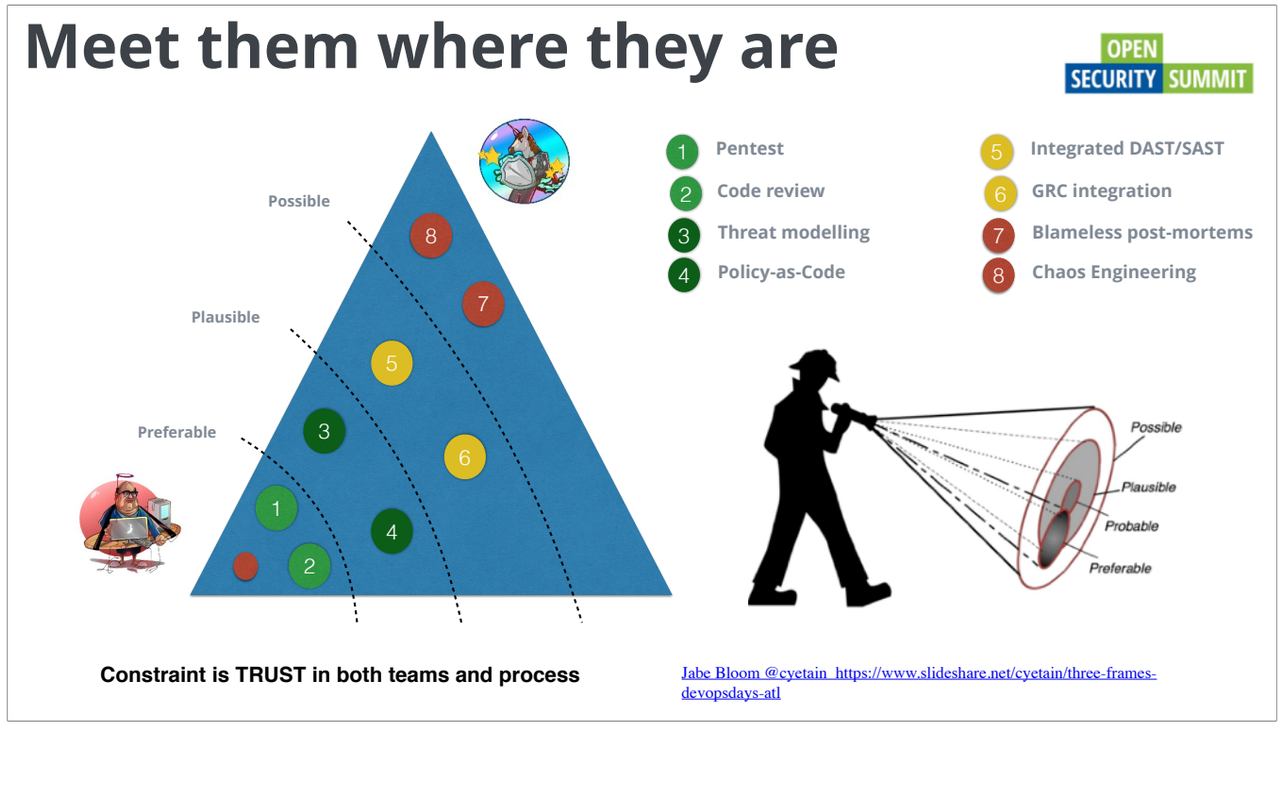
But this won't necessarily come natural as the key constraints is usually trust in both the teams and the process or automation. So we need to understand what some of the constraints are and how we can iteratively address them by meeting our stakeholders where they are.

Left to their own device, a true Gatekeeper will expect or prefer an independent Code Review and Pentest after every change. That's what their model of the world tells them good security is, but that we know would make security the bottleneck of the delivery process.

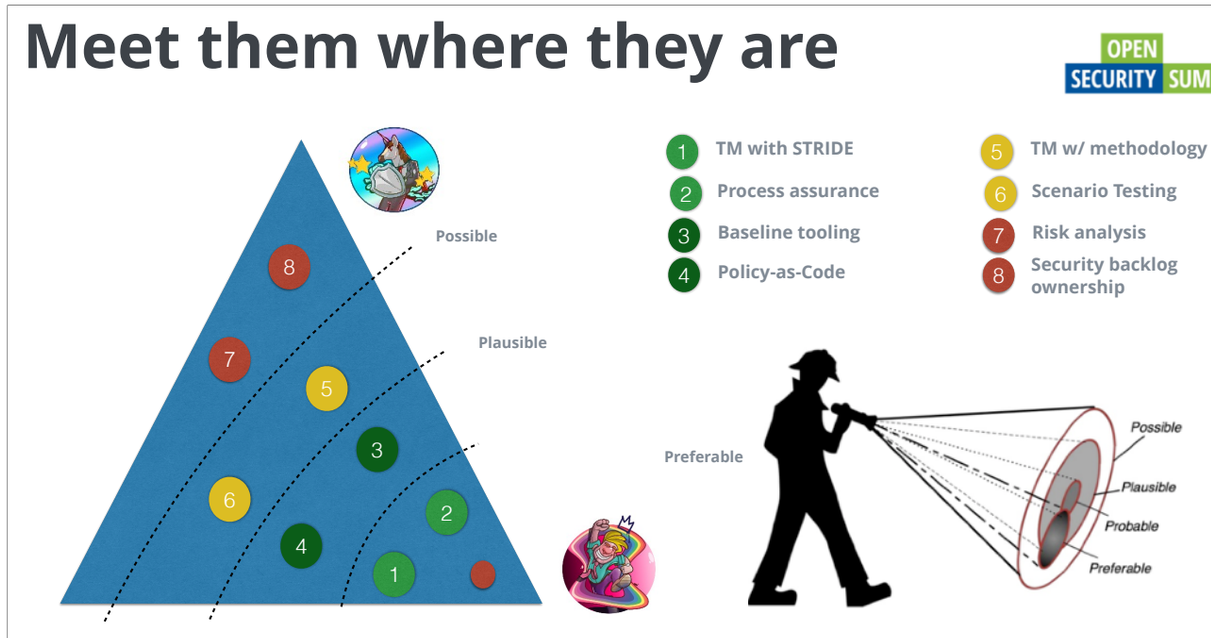
But they can probably comprehend and agree on benefits of Policy as Code and Threat modelling, if we show them how it can be connected and helpful. Policy as Code starting with simple stuff such as Checking cookies or the simpler parts of ASVS.



Once we start building that trust, we can then think of expanding those and start integrating some of those outputs directly into the Governance process so that Gatekeepers can start benefiting from the shorter feedback loops in a way they comprehend.



And then we can maybe think of more “advanced” things which are currently outside his/her view of what’s possible. I’ve specifically added blameless post-mortems here, as a true gatekeeper will likely have written the disciplinary process and his command and control view of the world may say those words, but not really mean it. Or suggesting to a Gatekeeper we’ll purposefully break things in prod is not going to bode well. We’ll have to build up to that, and eventually be at a place where all these practices are part of what Gatekeeper sees as Preferable because they understand the benefits they provide to the whole system and how it enables the creation of more resilient systems and organisations



A similar thing should happen with Rainbowmakers. They may currently lack security expertise, so starting by introducing STRIDE and Threat Modelling and getting them to review and identify threats on their current processes and practices is a great way to get started as that's where their agency will currently sit and that's the main constraint we need to deal with going up the triangle.

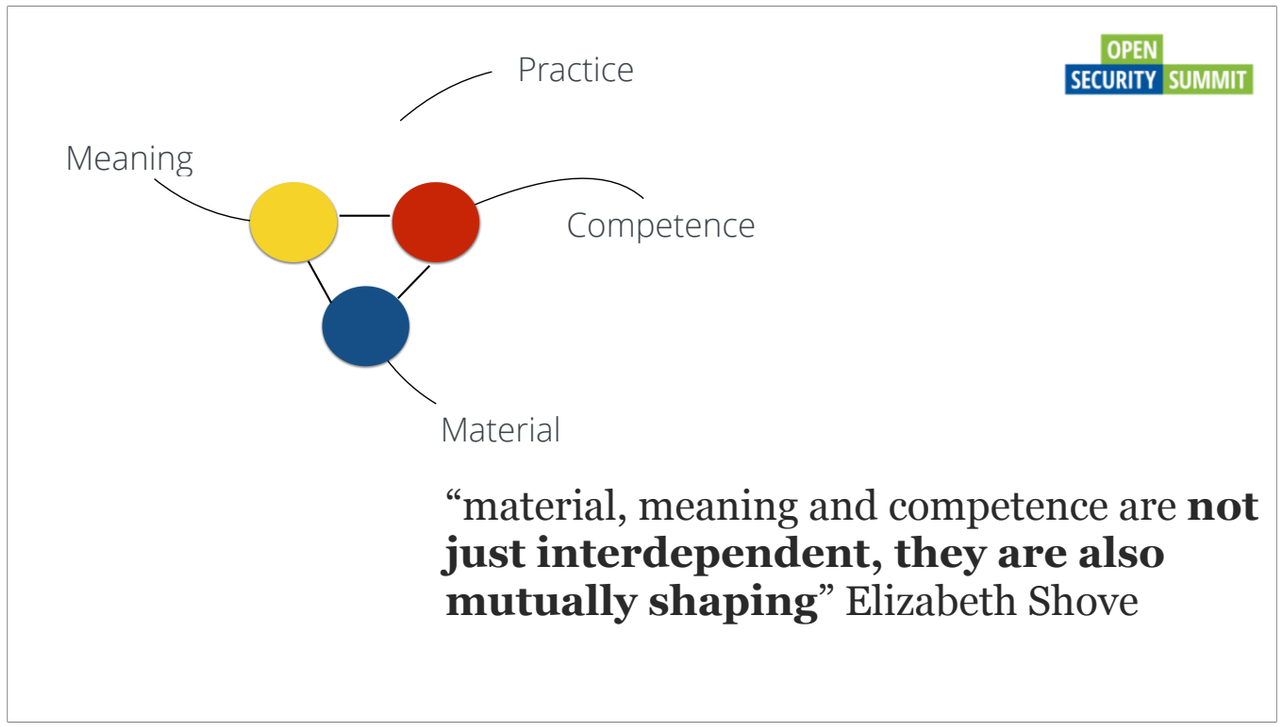
After that we can think about developing and letting them consume some baseline tooling for security visibility, policy as code, and then introduce further expert knowledge like Threat modelling methodologies and creation of security scenarios for testing and eventually have them be integral parts to the risk analysis and owning their own security backlog.

Remember it's the interaction between these and gatekeepers and how much they trust each other that will ultimately dictate success of integrating security in DevOps, so it's always a shared journey

OPEN SECURITY SUMMIT

But looking at the artefacts is not enough

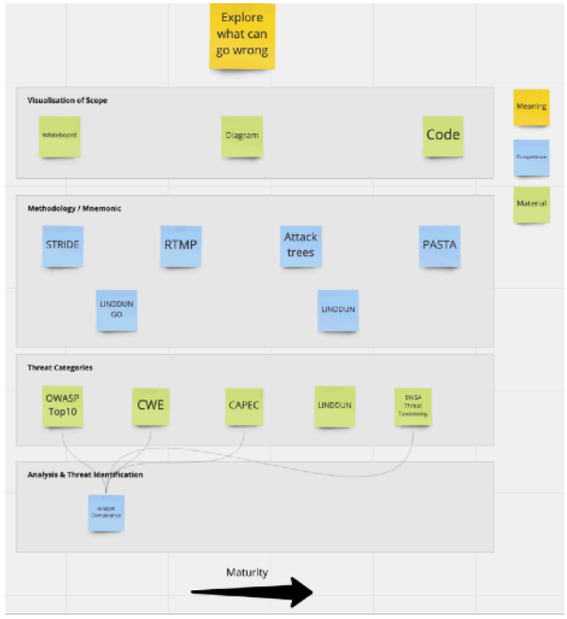
We need to understand their interactions



That’s why I believe and been researching application of Social Practice theory, which breaks down the elements of practice into 3:

Meanings which include symbolic meaning, ideas and aspirations
Competences which encompasses skills, know-how and techniques
And materials which are all the things, technologies, artefacts etc

And understanding they’re not just interdependent but actually mutually shaping so we need to be much more strategic and careful about how we approach each of these and how we relate them



https://medium.com/@marioplatt/social-practice-theory-and-cyber-security-a59364c8e732

An example using Threat modelling is that you should probably start by using whiteboards but eventually code as materials
And start by introducing STRIDE and potentially work up to a full fledged methodology
And in terms of categories start by using OWASP Top 10, but eventually move to something like CWE or CAPEC and also appreciate that whenever you'll introduce any new element there will be other elements which will mutually shape its form and efficacy

Evolve the Meaning of Practices

**“In a DevOps world, a Pentest is not for finding security issues.
It’s to improve process”
Mohammed A. Imran**

My favourite example of evolved meaning of security practices is this quote from my business partner, Imran

<read quote>
This quote fully embodies the aspect of evolution of meanings in that the competences and materials used to perform pen-tests haven’t necessarily changed but it’s meaning needs to change if we are to integrate security and adapt them to our new development practices

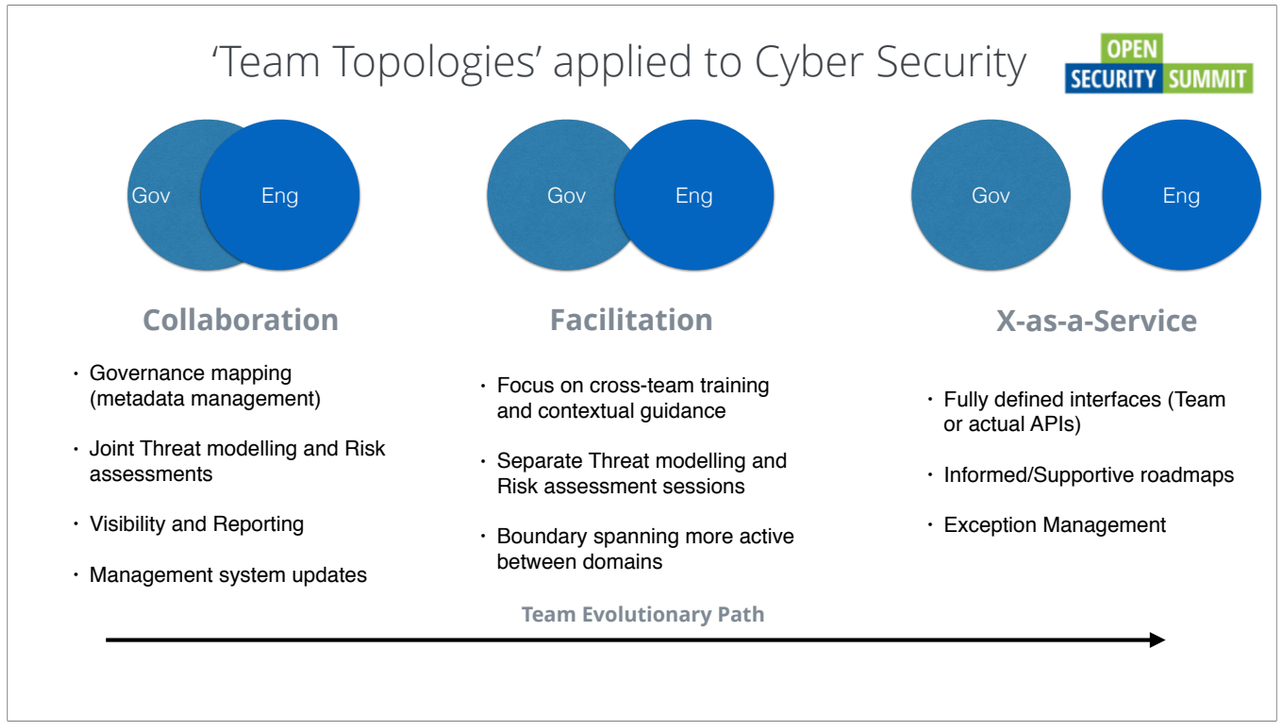
Team interactions also evolve

We need to change our “Team Topologies” based on effectiveness and maturity of our practices

Most fail as they assume shared mental models where they don’t exist yet

And lastly, team interactions also evolve. Gatekeepers and Rainbowmakers still live pretty much in significant tension.

For most, we really need a reset to “square one” and start things at the beginning, as their current organisation structure probably assumes people can provide services to each other but its unlikely they can do so effectively as it misses the elephant in the room that there are no shared meanings, effective traceability or efficient communication to build on.



Using concepts from “Team Topologies” book and the 3 interaction modes is a great place to start.

We first need to truly collaborate, and that may even imply it needs to be ONE TEAM for a while. There we can agree on governance mapping and management of metadata, and expose each of the practices like Threat Modelling and Risk assessments to the whole team so we have better appreciate of each others roles, mental models, constraints and trade-offs. This is likely to require changes to each other’s management systems to enable effective communication and informing.

Then, we can move to a Facilitation approach where we can start separating some of those practices and establish clearer boundaries between the scope of each team and overall focus on training

Finally, we may get to a point where all of those are in place and we can have fully defined interfaces (with actual APIs or what”s called Team APIs with regards to the services teams provide each other) where we have informed and supportive roadmaps and mainly manage exceptions allowing everyone to be effective and communicate effectively.

But this is likely to take time, for many probably measured in years so having realistic expectations is important.

So, should YOU DevSecOps or not ?

So, coming back to the tile of this talk, Should YOU DevSecOps or not ?

TRUST + AGENCY = (SECURE) DEVOPS

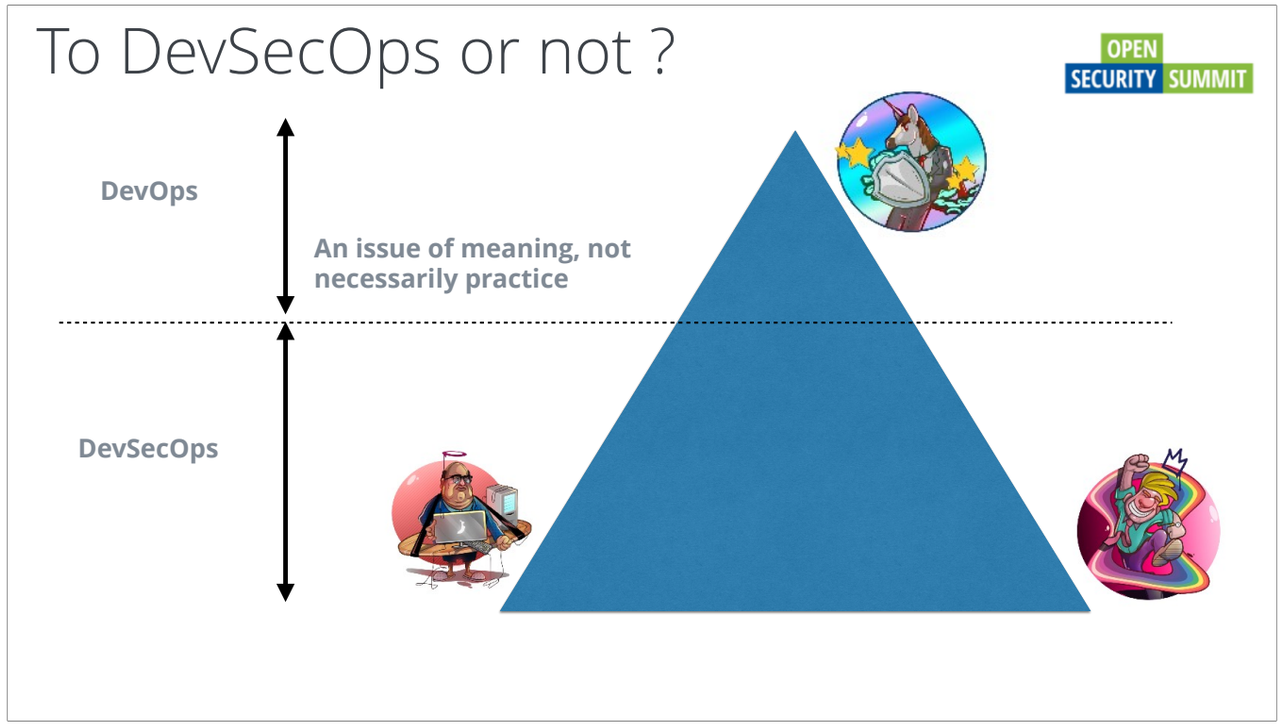


Naive to assume practices exist to make secure systems

But at some point, calling it something other than DevOps may be detrimental to culture and ownership

If we agree that constraints on both archetypes are centered around TRUST by Gatekeepers and AGENCY by the Rainbowmakers, I will argue that it's naive to assume that the organisational practices exist that will create secure systems and promote collaboration around security matters. So giving it a name whilst we build those organisational practices allows us to discuss them and address them.

But at some point it may become detrimental to culture and ownership

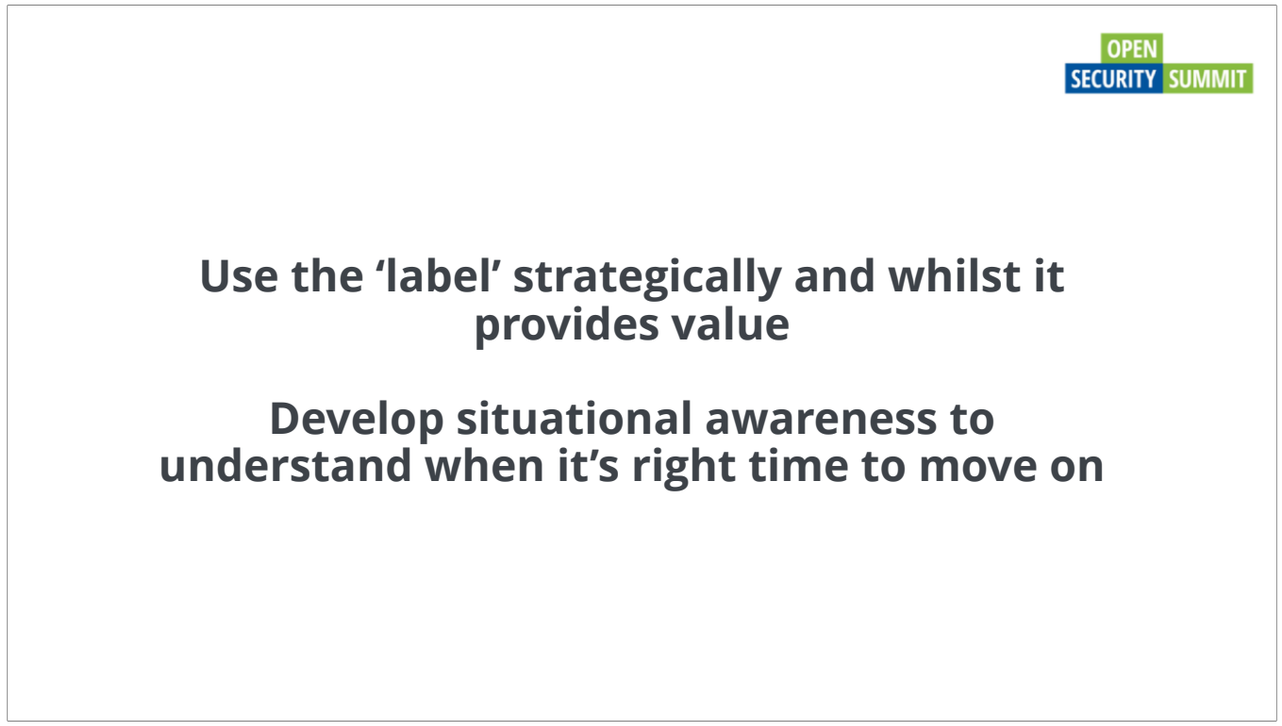


So, my opinion, is that it may be a useful term to introduce so different archetypes can have meaningful conversations about direction of travel and each of them will need to do, learn and change in order integrate security into their DevOps practices, up to the point where it may become detrimental

Some questions to ask to determine if you should use the word DevSecOps in our organisation:

- do you have tooling in place which informs Rainbowmakers of security state ?
- Do your gatekeepers understand what any controls that you have in place mean for their Compliance regimes ?
- Do Gatekeepers and Rainbowmakers set mutually supportive roadmaps, or are they selected in isolation ?
- Do Rainbowmakers have a sense of agency and ownership of the security of their products ?
- Do Product Owners understand and have a backlog of security work that is validated and agreed by the Rainbowmaker teams ?

If any of the answers to these questions are no, then I think that YES you should be adopting DevSecOps as something to discuss and adopt



So, one should use the label strategically and whilst or if it’s providing value, but also develop the situational awareness to understand when it may be the right time to move on from. I’d argue that until we have traceability, policy as code, teams owning backlogs and with agency over the shape and form of security of their products, it helps and allows one to discuss and dip into the body of knowledge which is still being developed detailing the practices.

Because at some point you do want to be in the position where the name DevOps in your organisation is giving you all the security that you need in a sustainable way, and allows both Rainbowmakers and Gatekeepers to dip into the Body of Knowledge and practices that have been created with that name.

But know that whether you use it or not, does NOT change the job to be done which is securing the systems our businesses are producing and practices we perform.

If your strategy mentions the word ‘DevSecOps’ or Security in Devops and you’re not

- helping your Governance teams benefit from short feedback loops and training them to understand DevOps
- Not increasing the agency and ownership of security across your Product or Project teams in THEIR language, not YOURS
- Not enabling the best possible Developer and Engineering Experience of Security you can afford
- Not actively trying to breakdown silo-ed barriers and connect governance systems

You’re probably doing it wrong!

And as a final note

If your strategy mentions the word ‘DevSecOps’ or Security in Devops and you’re not
helping your Governance teams benefit from short feedback loops and training them to understand DevOps
Not increasing the agency and ownership of security across your Product or Project teams in THEIR language, not YOURS
Not enabling the best possible Developer and Engineering Experience you can afford
Not actively trying to breakdown silo-ed barriers and connect governance systems

You’re probably doing it wrong!

Q&A

Mario Platt

mario@practical-devsecops.com

Twitter: @madplatt
LinkedIn: marioplatt
Medium: @marioplatt

OPEN SECURITY SUMMIT