

RESOURCE

Software security roadmaps

What they are and how they can help you



As much as we wish it would, secure software doesn't happen overnight.

There's no single action you can take or technology you can deploy that will solve this complex problem for you.

If that's not what you were hoping to hear, we get it - it's natural to wish for a simple solution. While we may not be able to provide that, we can certainly simplify the steps you need to take.

Like most of the harder challenges we face in software development, increasing the security of software is a journey that takes careful planning, a lot of collaboration, and a healthy dose of iterating as you learn more.

It's the type of complex journey that goes more smoothly when you have a map.

In this paper we'll look at what software security roadmaps are, why they're a useful tool to use as part of your overall software security approach, and how you can build one of your own.

We'll finish up with a look at some of the common challenges you might encounter when building and implementing your plan, so you can minimize and avoid some bumps along the way.

What's a software security roadmap?

Like any other sort of planned activity, software security takes time to achieve and requires implementing a number of changes and behaviors, normally in a particular order.

A roadmap is a plan to get you from where you are now to where you want to be, and a software security roadmap helps us make the transition described above.

The first thing to remember when building a roadmap is that what matters most is the structure and pragmatism of your plan — not how fancy your document is. You won't find a PDF template that solves all your problems here.

What we'll do instead is help you form a roadmap that suits your organization's planning style and communications culture. After all, a plan that ignores these things is often a plan that no one reads, understands, or implements.

Security should always be part of how your company operates, and this means working with these factors rather than thinking of them as separated or different. A software security roadmap needs to include the following:

- An understanding of your current security state or maturity level
- An intended destination in relation to this starting state and the timeline in which you intend to get there
- A series of activities, actions, or changes that will happen during this transition
- Expected results and outcomes for each change

You can establish this information by answering the questions shown below. We'll step through each one in more detail soon, but first we need to understand why you might want to have a roadmap and what benefits it can bring to your team as well as your overall security program.

To start with, we need to understand why you might want to have a roadmap and what benefits it can bring to your team as well as your overall security program.



Why have a software security roadmap?

We're all different when it comes to planning and strategy. Some of us like having a clear plan in advance and ticking off the stages as we go through them, and some prefer to take things more intuitively, figuring details out as we go.

Whatever your natural style, a software security roadmap is an incredibly powerful tool to help you improve the security of your codebases.

Let's go through why.

Software security requires consistent changes and practices applied to all software

Having a roadmap isn't about fixing the bugs in your current active project — it's about changing your approach and practices so every piece of code you write is designed and built to be secure.

A roadmap sets out a direction for your overall approach to software security in a way that can be measured over time. This allows security to be an ongoing, continuous process rather than just an item in your current sprint.

Software security roadmaps help you communicate your security maturity and approach

There are many reasons why you may need to share your software security approach and indications of your maturity beyond your immediate team. It could be a new customer or partner needing to do some due diligence as part of their risk management process. Or maybe your organization needs to comply with industry or regulatory standards (like PCI DSS or HIIPA) which includes a requirement for software security. Whatever the need, a software security roadmap can be a powerful tool to communicate your current maturity as well as show that you're actively working to keep improving things.

Even if you don't have an external stakeholder to share your security approach with, a well designed and documented software security roadmap is a powerful tool to use inside your organization.

Whether it's onboarding new team members, communicating your approach and strategy to leaders, supporting requests for budget, people, or resources, or just explaining to another team why something happens as it does, your software security roadmap can help.

Having a concrete document you can easily share can speed up conversations, reduce ambiguity, and help everyone share a clear understanding of the importance of and approach to software security.

Software security roadmaps include measurement of progress, so you can iterate if things don't work

Though there's no denying software security is important, it's also important to be prepared for the fact that when it comes to the actions you take to weave security through your software development process, not all of them will work out.

Having a roadmap with a clear measure of what improvement would look like and how you'll measure it will allow you to understand if a new initiative is helping your ambitions or, in fact, making things worse.

Now we know what a software security roadmap is and why they're useful, let's jump into the real reason you're reading this: to learn how to build your own roadmap and start maturing your software security practices.

Building your software security roadmap

Understanding your current maturity level



Let's start with the basics. Before going any further, we need to understand there's no such thing as "100% secure software". In fact, there's no such thing as "100% secure anything".

Just like you can never guarantee code is completely free from defects, you can never promise to stop every potential attacker.

When we set goals for software security maturity, we're not talking about having X number of bugs in our code or X number of findings on our next penetration test.

Instead, we need to focus on the leading indicators of secure software. These are the measurements we can take and assessments we can do that show we're applying the right sort of security approaches to every part of our software development lifecycle, which in turn means we're minimizing the risk that a security issue remains in our environment.

Home security is a helpful example to think through here. Look at how you secure your house before you go out to dinner. You can't say nobody will break in or cause damage. But you can say you've closed and locked the windows and doors, installed a monitoring system, and bought an insurance policy. Taking these actions means you've taken all the steps you can to:

- Prevent a security incident
- Detect that something has happened, and
- Respond quickly if the worst happens.

Realizing this can be liberating. If it's not possible to be perfect, then assessing where we are isn't about whether we pass or fail - it's about understanding how to get better than we are now.

Security change takes a long time in any area of any business. Setting your sights on moving forward from where you are makes your roadmap and the associated changes it will require more sustainable and achievable.

There are two common schools of thought when it comes to measuring your security maturity: **Product Focused** and **Lifecycle Focused**.

Most organizations will find a middle ground between these two approaches, choosing to use both to assess their security maturity.

	Product focused approach	Lifecycle focused approach
How it works	Measures your overall security maturity by the security present in the as-built application.	Measures the overall security maturity potential of all your software by looking at how security practices are woven through your software development process and how consistently you apply these practices.
What it's good for	Understanding risks present in your code in this moment.	Understanding the actions you're taking and the opportunities that exist for identifying and fixing security issues.
Challenges to keep in mind	In fast-moving teams, your product is often changing quickly and without consistent processes, so the security maturity of your code today may not be indicative of the maturity of the software you build in the future.	It can seem divorced from the process of coding itself.

Common software security measurements

There's a lot of information out there about the importance of choosing what you measure.

Though we won't dig into the nuance of this here, you may find books like Douglas Hubbard's How to Measure Anything or articles like You Are What You Measure (from the Harvard Business Review) useful as you're getting started on the subject of measurement. Why does what you measure matter? Because by measuring it, you're putting a spotlight on it - and this spotlight communicates to everyone on your team that this is an important thing they should care about.

With that in mind, let's look at some common measures we use to assess security maturity in the software development lifecycle.

Measure	What are we really measuring?
% of lifecycle integration	Every software development lifecycle looks different and no two teams will ever agree on the best approach. That's okay though, because just as our lifecycles can be unique, so can our systems of measurement against them. To measure lifecycle integration, we first need to outline all the stages and events in our software development lifecycle and all the potential security considerations at each stage. Once we have this outline, we can estimate how many of the potential measures we have taken.
Participation in security initiatives and activities	Security is a team sport. To fully weave security into your software development practices, you need to involve everyone who's part of those practices — like developers, testers, analysts, and architects. As well as looking at the security actions we take, we should also measure who's engaged and involved, and which roles they represent. If the number and/or diversity of people engaged is low or shrinking, that's a bad sign for our maturity. The more engaged people we have, the more likely it is that we're maturing our software security.
Progress on security debt	Technical debt is a sign of a healthy product-driven lifecycle. It's the technical or implementation items we haven't had time to (or have chosen not to) implement and will address later. Security debt is a subsection of technical debt and it can expose our applications, data, and people to increased risk over time. Security debt has to be paid, so measuring our progress in addressing these issues is a good way to measure momentum.
Frequency of security incidents or issues	 Where our last three measures look at the actions we take as a leading indicator of improved security (lifecycle focused), this is an example of a product-focused metric. It might seem obvious to hope that as our maturity increases, our number of incidents or issues decreases. This metric alone however can be quite complex to dig into. Incidents happen to even the most mature teams, and it's important to remember that sometimes the incident itself matters less than our preparedness to respond and reduce its impact.

When you implement your software security roadmap, it's likely you'll choose to combine two or more of the metrics described above. We recommend this approach over choosing just one.

While each metric provides something powerful to consider and reflects important actions on behalf of your team, on their own they only show part of the picture and may send you in the wrong direction. Using a combination of metrics gives you a more comprehensive view, allowing you to make more informed decisions.

Using a recognised framework

While there are many options to design your own measurements for software security maturity, there are also some great open-source frameworks you can use. Whether you use these alone or as well as a more context-specific system, they give you an existing and globally understood process to apply to your organization. This can come in handy when explaining your approaches to external stakeholders like potential customers or auditors.

OWASP Application Security Verification Standard (ASVS)

https://owasp.org/www-project-application-securityverification-standard

The OWASP ASVS is a product focused application security maturity framework that can be used to validate and confirm that an application has been built to include a range of standard security controls to reduce the risk of confidentiality, integrity, or availability loss.

The ASVS categorizes applications into three levels, referring to the sensitivity and criticality of the process they provide and the data they store. The higher the level, the more important security is to its design and implementation. It then outlines a set of controls that would be expected for each level of application across 14 verification areas including connectivity, authentication, and data storage.

The ASVS contains guidance on how to use it as a metric or baseline assessment tool and can be a great starting point for evaluating the security for a specific application or product.

OWASP Software Assurance Maturity Model (SAMM)

https://owasp.org/www-project-samm

Where the OWASP ASVS is a product focused standard for measuring maturity, OWASP SAMM is a lifecycle focused standard.

OWASP SAMM aims to provide an effective and measurable way to assess the maturity of security initiatives throughout your software development lifecycle. It splits its focus across five business domains that align with the phases of most software development approaches. Each of these domains is associated with a number of associated practices. These practices include the common sorts of security initiatives that you'll typically build as part of your software security roadmap.

OWASP provides resources like self-assessment templates and complete documentation for both these frameworks, making it a straightforward and accessible place to start when calculating your organization's software security maturity baseline.

Setting realistic goals



Whether you've used a recognised framework like those provided by OWASP or you've devised your own measure that helps you understand where you are now, the next step is deciding where you want to get to.

When deciding what this end goal will be, it's worth bearing a few things in mind.

- Software security roadmaps often lay out actions and initiatives that your team will put in place over a period of time. This period of time may be anywhere between one and five years. Be pragmatic with your aims and relate them to the time period your plan covers. Aiming for massive change in maturity over a short period of time is unlikely to end well. Equally, setting a very low goal over a long period of time will send the wrong message to your team about the importance of the work.
- Security needs a team. Your goals need to be supported by having enough people, time, and budget to get the work done. If you're a team of one trying to take it all on alone with only some sticky tape and good intentions, that's going to impact your ability to successfully implement the roadmap.
- Change is the only constant. Your roadmap needs to be flexible enough to handle the level of change in your business. If you work for a large enterprise with relatively slow processes, factor that in. If you work for a high-velocity organization, then you need to assume there may be changes in direction or external influences that pop up quickly and unexpectedly.

Choosing actions and initiatives (or "what do I need to do on the way?")



Just as every team has a slightly different approach to their software development lifecycle, every security roadmap contains a different set of actions and initiatives that can be woven through it.

Think of this section as a guide to the types of actions and initiatives that are common and effective, rather than as a prescriptive or exhaustive list.

To structure our suggestions in this section, on the next page we divide the actions and initiatives by stage of the software development lifecycle and type of control (Preventative, Detective, or Responsive).

This will help you spot your options and make sure you're balancing your chosen initiatives in a way that weaves them throughout every stage of the lifecycle and provides a range of controls.

Design

Action/Initiative	Type of control	What will it achieve?	Example
Secure development education	Preventative	All your team members have the skills to understand security and build it through their development practices.	SafeStackPluralsight
Threat assessment and threat modelling	Preventative	A structured way to review your software designs so you can identify common security vulnerabilities and design flaws from the architecture onwards.	 Microsoft STRIDE https://github.com/ xntrik/hcltm

Implementation

Action/Initiative	Type of control	What will it achieve?	Example
Code reviews	Preventative	Shared knowledge across your team as well as a chance to identify design and implementation flaws before code is committed.	 SafeStack Code Review Security Checklist and Implementation Manual
Static analysis security testing (SAST)	Preventative	Reviews of your static code, looking for patterns that may indicate poor security design or implementation.	CheckmarxFortifyGitHubSonarqube
Dynamic analysis security testing (DAST)	Preventative	Reviews of your code as it executes, looking for insecure code pathways or unexpected behaviors.	 Burp Suite OWASP ZAP Invicti (formerly Netsparker)
Credential and password scanners	Preventative	Identification of passwords, secrets, and other sensitive information that shouldn't be stored in your code repositories.	CredScanSonarqubeTruffleHog

Testing

Action/Initiative	Type of control	What will it achieve?	Example
Automate your security testing	Preventative	Automation of tests for security risks in your application, ensuring they can be run frequently throughout development, build, and deployment. Provides consistent and low effort checks for scenarios you've previously defined.	GauntltBDD Security

Build

Action/Initiative	Type of control	What will it achieve?	Example
Check third party dependencies for known vulnerabilities	Preventative and Detective	Notification of components with known vulnerabilities, so you can remediate them before deployment, reducing the likelihood of exposure to automated attacks.	 Snyk OWASP Dependency Checker

Deploy

Action/Initiative	Type of control	What will it achieve?	Example
Configure cloud hosting security checks and monitoring	Detective	These built-in monitoring tools will alert you of deployed components that are poorly or insecurely configured.	Azure SentinelAWS InspectorSecureStack
Vulnerability scanning	Detective	These automated tools can be used to scan your applications on a regular basis to identify trivial or low-hanging application security flaws.	OWASP ZAPQualysNessus

Support

Action/Initiative	Type of control	What will it achieve?	Example
Integrate all logs into a central repository or security information and event management (SIEM) system	Detective	Bringing all your logs together means they're protected from harm and you can correlate activity across multiple systems and components. This enables you to configure alerts for suspicious activity.	DatadogRaygunAzure Security Centre
Create an application security incident response plan (and practice using it)	Responsive	 Gives you a concrete set of ac ons to take when something suspicious happens. The focus of this plan is to: Outline the actions needed to control or limitthe attack Protect systems and data Restore services as soon as safely possible. 	 https:// response.pagerduty.com SafeStack Seminar An AppSec Guide to Incident Response

Additional resources

- https://securestack.com/devsecops-playbook/
- https://www.microsoft.com/en-us/securityengineering/sdl

Measuring progress (or "how will I know this is working?")



There are always two levels at which you should assess the effectiveness of a software security roadmap:

- At the action or initiative level
- As a combined program of work.

Assessing the effectiveness of actions and initiatives

For each action or initiative, you need to define what results you expect to achieve. In some cases this might be straightforward.

For a static code analysis tool, success would look like this.

- The tool is run automatically on every piece of committed code
- Results from the tool are made visible to the team and a plan is made for their remediation
- The number of false positives (issues that aren't really issues) detected by the tool reduces over time.

This set of success criteria is easy to identify and automate, and each one is measurable in a way that produces metrics for analysis and comparison.

When it comes to initiatives and actions based around human processes, measuring success can be a lot more tricky and subjective.

If you wantto ensure all system designs go through a security review before they're implemented, the success criteria would be very different to the static code analysis tool example above. It may look something like this.

- A review is scheduled and held for each design
- The review follows a process that's designed to identify appropriate risks and design flaws
- The review is facilitated by someone with sufficient time, experience, and skill who is not directly involved with the design's creation or implementation (so they can be objective)
- The people who know the system best and the ways it will be used are atthe review
- The results of the review are turned into actions that can be planned for and accommodated into the development process.

As you can see, this listis quite human-centric. Plus, the results of each stage are highly variable and likely to be judged differently based on the person assessing the effectiveness.

In cases like this, it's bestto identify where you can automatically assess a criteria (for example, scheduling a design review and notifying the right people) and then consider your process for those items that need a more human touch.

Rather than tracking a number or value for these subjective assessments, holding regular process retrospectives will give an indication of how successful the process is and identify any opportunities to improve.

Assessing the effectiveness of the overall roadmap

To truly assess whether your roadmap is working, the easiest solution is to return to the beginning. The process you used to define and measure your baseline security maturity is the same process you can use throughoutto check that you're making progress. The OWASP SAMM and ASVS frameworks are perfectfor this.

Using the same framework each me also means you can consistently report your progress over time and provide valid direct comparisons with previous assessments.

> Make your assessment simple. Create reusable templates and ensure they're easy and quick to complete. By making the assessment rapid and low-friction, you ensure it doesn't interfere with work, making it a tool that can be used frequently as a health check.

Avoiding common software security roadmap challenges

Whether you're brand new to this and in the phase of asking Google "how do i make a software security roadmap" or you've done it many times before, there'll always be challenges. A software security roadmap is a pathway to change and change always brings friction, failure, and frustrations.

But there's no reason to let that put you off. In this section, we'll look at some of the common challenges you might face and how to quickly overcome them.

Culture and people challenges

Challenge: My development teams say the new processes are slowing them down

One of the biggest and most frequently-faced challenges is backlash from those involved in the development process. Whether it's a refusal to use the tools or processes provided or the sharing of some less than subtle feedback, something is going wrong and it's important to address it quickly.

Let's look at the most common causes of this type of disengagement.

In terms of communication, processes are typically adopted more fluidly and with less friction when the people whose day-to-day operations will be impacted are involved in the planning and design of these processes.

Don't write your roadmap in isolation. Make it collaborative and do your best to understand the concerns of your team in advance.

Secondly, these issues come from implementing processes and procedures that don't respect the needs of your team.

For example, no software development team is ever bored. They have more than enough work to keep them busy. They like it when their processes are smooth, quick, and frictionless. If we introduce new practices that break these processes, introduce friction, create more work, or slow things down, we're not respecting the challenges and needs of their world.

As you plan your actions and initiatives, make sure you understand the impact they'll have on day-to-day operations and address any issues that may cause friction or pain. This shows you respect your team and their world and often increases their engagement over time.

Challenge: I can't get the right people to engage

Sometimes the problem isn't that nobody wants to engage and participate. Instead, it's that it's not the right people getting involved. On occasion, we may find we have supporters who are very enthusiastic but less experienced, while the more experienced people are too busy and don't get involved.

This can be challenging.

There's nothing wrong with having less experience and wanting to get involved, but we need to remember that these people will need more guidance and support. This will take more resources and time to achieve and may have a slower ramp-up curve before you see results.

To engage your more experienced people, look at what time and opportunities they have to be involved and make the most of them.

Ask for help with specific, measurable, and time-bound activities or with activities that enable those with less experience to do more.

Linking engagement with professional or career development plans can also increase your success rates and incentivise the right people.

Budget and leadership support challenges

Challenge: We have no money – how can I do this?

We're going to answer this question with another question: What if we told you you didn't need stacks of money?

Don't be fooled. The billion dollar application security software industry spends big bucks each year telling us the only way to stay safe is to buy their stuff. In reality, most security tools are about saving you time and applying processes consistently rather than any sort of magic that will keep you safe from everything.

This means that those of us with modest budgets have to spend more time on our processes and on manual effort, and choose where to invest our money more wisely.

Let's look at static analysis tools as an example. At their simplest, they're a fancy way to look for regex in a codebase. As software developers, there's no need for us to feel intimidated by that. If you don't have the budget, use your skills and your time and make something that works for you.

Though this approach often proves unsustainable in the long-term, in the short-term you may be able to use it to prove your plan and show evidence of improvement — which then makes it easier to get leadership support and hopefully, more budget.

Challenge: How do I get my managers and senior staff to support my roadmap?

Whether we read the works of folks like Simon Sinek or not, many of us are now aware that getting people to support you means communicating clearly and sharing why what you're doing matters.

Security isn't going to make your company more profitable or sell more widgets but it can enable the business to make good decisions, reduce risk, and satisfy customer concerns more easily. These are all important things that the leaders in your organization care about.

In the case of software security, your leaders are probably aware of the rise in security incidents globally. What they may not yet understand is how that relates to their own organization. Finding a way to make this information accessible, non-technical, and linked to overall company strategy is essential.

If you can't get support for a large, multi-year roadmap, break it down into smaller parts and clearly communicate the benefits of what you've achieved at each milestone. The more you do this, the more confident your leaders will be in supporting your plans.

Process and technology challenges

Challenge: There are too many tools to choose from — which one should I buy first?

Stop. Put down that credit card. If you don't know where to start, go back to your roadmap. That should outline the initiatives you're going to implement and in what order.

Your first consideration should always be answering the question "does this help me progress against my plan?"

Secondly, outline what you want to achieve from a tool and how you'll know its a good fit for you. These success criteria should be defined upfront before you do a demo. If possible, communicate with the tool vendor so they can help you measure your success.

Thirdly, don't be afraid to say no and not purchase. Budget is hard to get and easy to spend.

Challenge: This tool seems perfect, but it's not doing what I expected.

The sad truth about this challenge is it's often not the tool's fault. It could be the most sophisticated, cutting-edge technology available, but without someone skilled at the helm, you may not get the value you expect. With the average cyber security salary in the United States reaching \$116,981 in 2022, that's a big investment, even for a larger organization.

If you're not getting what you expect from a tool, speak to the vendor and look at how much time and resources you've committed to getting it embedded, configured, and installed. Chances are that either it wasn't the right tool to begin with or you haven't got the right people with the right support at the controls.

Challenge: My new processes don't work with my legacy projects or lifecycles.

It's very rare for a company that builds software to have the same technologies and stacks in every project they develop and support. Projects that have been inactive the longest often contain technologies we've long since replaced or moved away from.

This technology drift causes challenges when implementing security initiatives in software development lifecycles. Though there's no getting away from that, it doesn't give you license to mark your legacy projects as exempt.

When weaving security through legacy projects and lifecycles, focus on detective and responsive controls which monitor for signs of issues and allow you to respond first. Once these are in place, look at the processes in your roadmap that are less technology-dependent and make sure your legacy teams have the time, skills, and support to implement them.

While this may make your legacy security initiatives more human-focused, it will allow you to keep these foundational systems safe without over committing to re-architecture or re-engineering initiatives.

Motivation and momentum challenge

Challenge: Our progress has stalled – how can we get back on track?

Starting new behaviors and initiatives is easy – keeping them going over time is hard. This is true for improving your health, renovating your home, or securing your applications.

Though we love the dopamine hit of experimenting with new ideas and pioneering change, the grind of getting that change fully embedded and effective can wear anyone down.

Remember that your roadmap is based around the repeated measurement of maturity and effectiveness. Use this to your advantage. If you can see gradual change frequently, you're more likely to keep feeling engaged and motivated than if you have to wait six months to see if your roadmap is working.

Make your initiatives easy to adopt and easy to measure, and give everyone the chance to provide feedback and suggest improvements. The goal is to be a little more secure each day, and that's something everyone can get behind.

Conclusion

Whether you're trying to improve software security in a small and growing company or you're trying to retrofit these ways of working into an already existing or mature software development lifecycle, an effective roadmap is the most important thing you can invest in.

By understanding your current security maturity both from a product and lifecycle perspective you can plan the appropriate steps that will take you and your wider team from where you are to where you want to be. Using your own measures or those provided by organizations like OWASP, you can create a plan that suits your budget, risk profile, and experience.

It's important to remember this journey takes time and often comes with challenges.

While those challenges may be similar to those covered in this white paper or more unique to your context, how you respond to them will determine how effectively you're able to weave security throughout your software development lifecycle.

About SafeStack

SafeStack is a community-centric online training platform that takes a flexible, people-focused approach to ongoing cyber security education at a time when it's never been more needed.

By teaching software development teams to weave in security from idea to maintenance, as well as providing cyber security and privacy awareness training for the wider workforce, SafeStack's training programmes offer a comprehensive way of protecting people, systems, and data in an ever-changing world.



