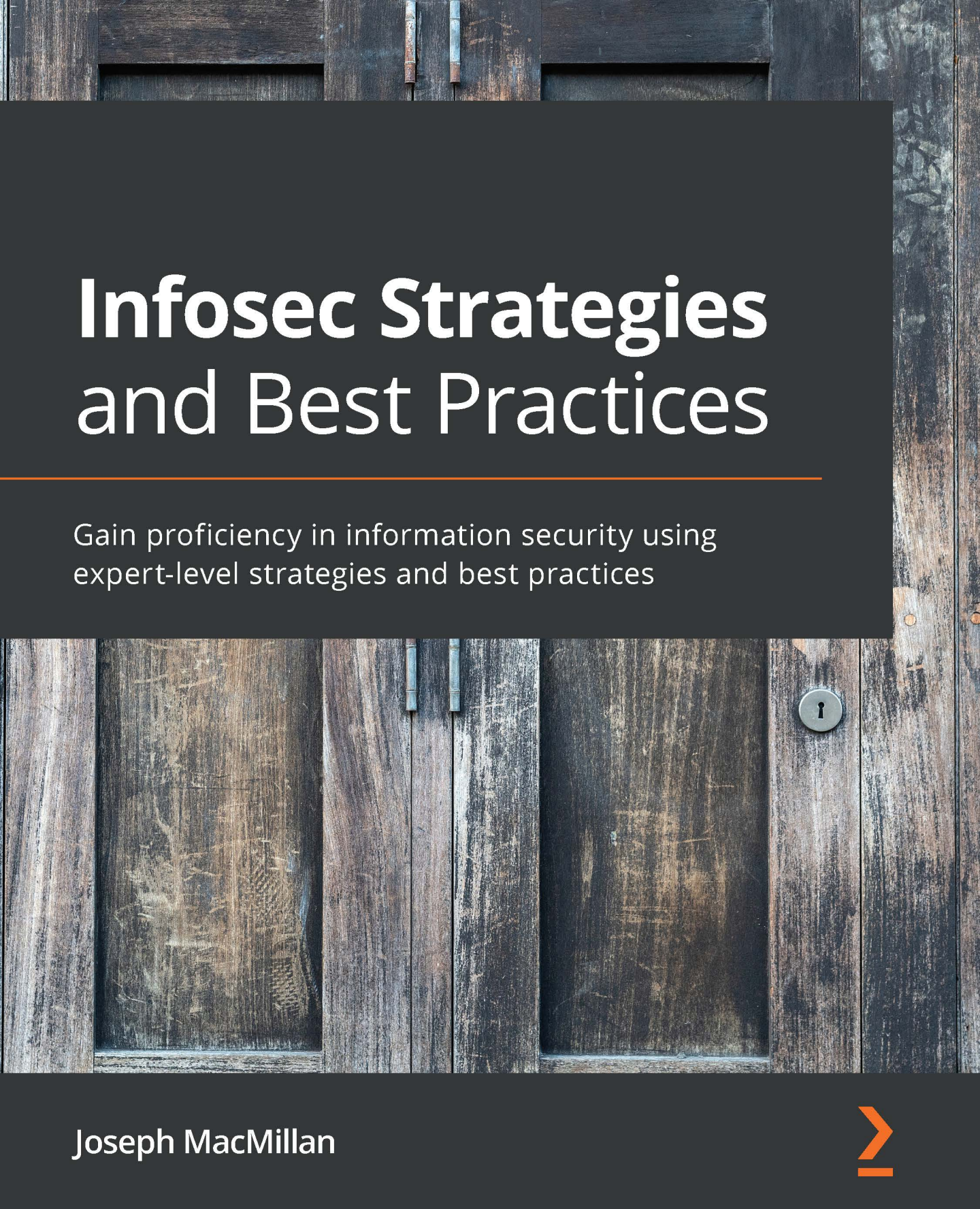


Infosec Strategies and Best Practices

Gain proficiency in information security using expert-level strategies and best practices

Joseph MacMillan

Infosec Strategies and Best Practices

Gain proficiency in information security using expert-level strategies and best practices

Joseph MacMillan

Packt

BIRMINGHAM—MUMBAI

Infosec Strategies and Best Practices

Group Product Manager: Wilson D'souza
Publishing Product Manager: Sankalp Khatri
Senior Editor: Shazeen Iqbal
Content Development Editor: Romy Dias
Technical Editor: Nithik Cheruvakodan
Copy Editor: Safis Editing
Project Coordinator: Shagun Saini
Proofreader: Safis Editing
Indexer: Rekha Nair
Production Designer: Shankar Kalbhor

First published: April 2021

Production reference: 1240521

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

ISBN 978-1-80056-635-4

www.packt.com

To Helen, my best friend and the love of my life. Her support made all of the difference while I was trying to write this thing.

Contributors

About the author

Joseph MacMillan is a technological utopian and cybersecurity junkie, currently living in Amsterdam.

Employed by Microsoft as a Global Black Belt for Cybersecurity, Joseph uses his experience in senior information security roles to transform businesses into secure organizations rooted in risk management principles to drive decision making.

Much of Joseph's work has been focused on enabling businesses to achieve their goals by removing the ambiguity surrounding risk, which enables the CEO and C-Suite to plan and achieve their goals in a secure manner with confidence.

Joseph holds various certifications, including the CISSP, CCSP, CISA, CSSLP, AlienVault Certified Engineer, and ISO 27001 Certified ISMS Lead Auditor.

I would like to take this opportunity to thank my wife and best friend, Helen, for her support and care along the way. Furthermore, I would like to thank all of the members of the Packt team who have supported me in the creation of this book, from beginning to end.

About the reviewer

Kyle Reidell has world-class experience leading, developing, and architecting cybersecurity and engineering solutions for numerous government agencies, as well as Fortune 500 companies, and cutting-edge technology start-ups. His background is truly multi-disciplinary: from developing and defending global operations centers to securing communications for the highest levels of government and designing cloud-native architectures while continuing to serve as a Cyber Officer in the Air National Guard.

Kyle is a Marine Corps veteran who is actively engaged as a mentor for aspiring youth and cybersecurity professionals. He holds multiple degrees and industry certifications, including a master's degree in information security.

I would like to thank my family, especially my wife and son, for the continuous support they have provided throughout my career and endeavors; I could not have done any of this without them!

Table of Contents

Section 2: Closing the Gap: How to Protect the Organization

3

Designing Secure Information Systems

4

Designing and Protecting Network Security

5
Controlling Access and Managing Identity

Section 3: Operationalizing Information Security

6
Designing and Managing Security Testing Processes

7
Owning Security Operations

8
Improving the Security of Software

Preface

In this book, we will cover various topics within the **information security** (**InfoSec**) domain, and help you to translate your organization's strategic requirements into actionable improvements in securing their most valuable assets.

You can expect to learn about a wide range of InfoSec paradigms, including the foundations of risk management, implementing processes and controls, designing information systems securely, and managing the day-to-day activities required to ensure security is maintained at your organization.

Upon completion, you should be well on your way toward converting the theory of your InfoSec certifications into actionable and practical changes you can make to ensure your organization is more secure. Beyond that, delving deeper into any and all of the topics covered in this book will help you to progress in your career as an InfoSec professional.

Who this book is for

This book is for those who are looking to begin (or have recently begun) working in an InfoSec role. Perhaps you've been taking courses and studying for an industry-standard certification such as the CISSP or CISM, but you're looking for a way to convert the concepts (and seemingly endless number of acronyms) from theory into practice and start making a difference in your day-to-day work at your organization.

What this book covers

Chapter 1, *InfoSec and Risk Management*, establishes the core principles of InfoSec and ensures the topics central to the discipline are well-understood.

Chapter 2, *Protecting the Security of Assets*, implements effective processes to ensure you can identify the assets of an organization and avoid common pitfalls that InfoSec professionals encounter.

Chapter 3, *Designing Secure Information Systems*, explores how to assess architectures and systems for vulnerabilities and mitigate those vulnerabilities with controls, including cryptography.

Chapter 4, *Designing and Protecting Network Security*, covers designing secure network systems, selecting the appropriate network components, and ensuring their effectiveness for your organization's requirements.

Chapter 5, *Controlling Access and Managing Identity*, examines both physical and digital access to your organization, and the various aspects of selecting and implementing the appropriate identity and access management controls.

Chapter 6, *Designing and Managing Security Testing Processes*, covers adopting a mindset of continuous improvement by testing existing implementations and utilizing any findings to optimize your InfoSec program.

Chapter 7, *Owning Security Operations*, covers aligning the day-to-day tasks involved with maintaining InfoSec to an organization's strategies.

Chapter 8, *Improving the Security of Software*, covers enforcing secure practices in procuring and developing software.

To get the most out of this book

Bring your inquisitive nature and interest in securing information systems. This book covers an extremely wide set of subjects, offering the opportunity to investigate further on your own. If a topic interests you, make sure you delve deeper into the content available online.

After completing this book, challenge the conclusions made, don't accept anything as a hard-and-fast rule, and cater all the solutions to suit you and your organization.

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: `http://www.packtpub.com/sites/default/files/downloads/9781800566354_ColorImages.pdf`.

Conventions used

The following is how tip and information notes will be shown throughout this book:

> **Tips or important notes**
> Appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at `customercare@packtpub.com`.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit `www.packtpub.com/support/errata`, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the Internet, we would be grateful if you would provide us with the location address or website name. Please contact us at `copyright@packt.com` with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit `authors.packtpub.com`.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit `packt.com`.

Section 1: Information Security Risk Management and Governance

In this section, we'll establish the core principles of information security, and ensure the topics central to the discipline are well understood.

This section contains the following chapters:

- *Chapter 1*, *Infosec and Risk Management*
- *Chapter 2*, *Protecting the Security of Assets*

1
InfoSec and Risk Management

As this is the first page of this book, I'm meant to tell you why you might want to buy this book, instead of any of the others. Well, if the following describes you, then this book is going to help you in your career:

You are looking to begin (or have recently begun) working in an information security role. Perhaps you've been taking courses and studying for an industry-standard certification such as the CISSP or CISM, but you're looking for a way to convert the concepts (and seemingly endless number of acronyms) from theory into practice, and start making a difference in your day-to-day work at your organization.

In this book, we're going to help you turn the theory of your certifications into actionable and practical changes to make your organization more secure, and also help you progress your career as an information security professional.

Has that sold you? Is this book in your shopping cart now? Great – then let's get started.

This first chapter will go over the major topics that heavily influence decisions made by information security professionals: risk management and governance structures. That may not sound like a barnburner, full of thrills and excitement, but if you can manage to master the basics found in this first chapter, I can actually promise you that you will be a highly effective, well-oiled risk management machine in no time. Now if *that* doesn't make you want to read on, what would?

Let's get a bit more formal and list the main topics we're going to cover in this chapter:

- Basic InfoSec terminology
- Understanding why risk management is important
- Performing a basic risk assessment
- Considering legal regulations, investigations, and compliance structures
- Proven methodologies in creating a strategy

And so, let's begin!

Basic InfoSec terminology

Before we get into some heavier topics, first, we need to establish a common set of terms and ensure we're speaking the same language in order to simplify the ideas in this book. Otherwise, this book could become too complex, or even nearly impossible to decode as a result. My intention for this book is to reduce ambiguity and ensure you're right here with me.

For example, on the first page, I said that this book is going to help you with "making a difference in your day-to-day work at your organization." In that sentence, we can see a common InfoSec term: **organization**. Whether your information security knowledge is being utilized by a corporation, government, small business, firm, non-profit, extracurricular group, or some other structure, they all fit under the umbrella term of "organization," and that's the one we're going to use in this book.

Let's look at a few other basic terms. How about **information**? Information includes data but can also include common knowledge or intellectual property. For example, the new company slogan that is written down on a whiteboard after a brainstorming session is now "information," which should be protected, even if it isn't "data" in terms of it sitting on a hard drive in ones and zeros.

That logically brings me to **security**, or *the state of protection*. Information has the ability to be secured in various ways, most commonly noted as a triangle or "triad" of concepts called the **CIA Triad**. You'll remember this from other InfoSec books and training you have done, but **CIA** stands for **confidentiality**, **integrity**, and **availability**. They're extremely important in navigating the day-to-day of information security, and that's why they're being brought up again:

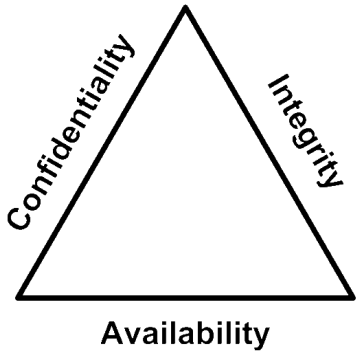


Figure 1.1 – The CIA Triad

Confidentiality is the concept of keeping information secret and ensuring only those that have been authorized to can see it. **Integrity** is the concept of being sure the information hasn't been changed without permission. **Availability** means that those with permission to access the information can access it in a timely fashion. The reason the *CIA Triad* is often represented as a triangle (as in the preceding diagram) is because the three principles work together as an interconnected system that represents the security of an asset, or group of assets.

The often-forgotten "N" in the CIA Triad is **non-repudiation**, which is a way of saying that if an entity has authored (or changed) some information, it is authoritatively indisputable. It is the ability to prove the origin of said data.

Now that we've covered some of the basics, I'd like to discuss why risk management is important to you and your organization.

Understanding why risk management is important

In this book, we've going to keep referencing *risk*. To ensure we're on the same page, it's worthwhile taking the time to define and discuss various *risk management* topics, and why they're important.

First, we're going to define **risk** as *the potential for loss*. The components that make up *risk* are **assets**, **threats**, **vulnerabilities**, **likelihood**, and **impact**, each of which we will define and explore in this section, and continually reference throughout this book.

For the time being, **information security risk** can be summarized like so:

> *"Information security risk is the potential for loss, as measured through the combined impact and likelihood of a threat exploiting a vulnerability in one or more information system assets."*

Imagine that by understanding the *likelihood* of a *threat* exploiting a *vulnerability*, combined with the *impact* of that *threat* exploiting that *vulnerability*, you can measure the level of *risk* associated with that **security event**. Your organization faces many risks on a daily basis, and those risks may or may not already have **risk scores** and annualized costs associated with them, as determined by performing a **risk assessment** to determine the impact of the threat exploiting a vulnerability, and the likelihood that the threat will exploit that vulnerability.

Consider the following simplified formula:

Risk = Impact x Likelihood

Organizations are interested in minimizing their potential for loss through a process called **risk management**, by which they identify, evaluate, and prioritize risks, and implement layers of mitigating controls that reduce the impact and likelihood of a loss.

So, *why is risk management important for information security*? Risk management isn't about removing all the risk an organization faces, or adding bureaucracy for bureaucracy's sake. Risk management provides transparency and accountability into the daily operations of an organization, and aligns the organization's risk with its strategic objectives. Risk management also helps organizations prepare for adverse events that could negatively impact its success.

In the modern context, information systems are central to the ways in which most organizations do business, and so *information security risk management* is becoming a larger part of every organization's overall risk management focus.

In order to cover *risk management* effectively, it's important to go into a few key *risk* topics and definitions first. Let's get started!

Understanding assets

Let's look at that long sentence I wrote out earlier to summarize *risk*:

> *"Information security risk is the potential for loss, as measured through the combined impact and likelihood of a threat exploiting a vulnerability in one or more information system assets."*

Looking at the structure of what comprises a risk, we can see that, first, you cannot have a *risk* without an **asset**. Your organization has many *assets*: digital assets, physical assets, and even reputational assets. For this book, I'm going to use the definition of *assets* as *anything of value to the organization*. Different *assets* have different *risks* associated with them, because different *assets* have different *threats*, different *vulnerabilities*, and different levels of *impact* in the event of loss of, exposure of, or making modifications to that *asset* (or the processing it performs).

Now, there are some company assets that will not have a significant information security risk associated with them, such as a box of T-shirts from a company picnic a couple years ago. Let's focus on documenting your organization's "crown jewel" information assets into an *asset register* to start; we can always add more assets to the register as we go along.

An **asset register** or **asset inventory** is a document that lists the organization's assets. Creating a document like this (and keeping it up to date) is going to help with various aspects of the risk management process, because inside this *asset register*, you will begin to structure various organizational aspects surrounding how assets are handled, and identify risks based on those assets. The information inside this document could include *asset owners*, *asset classification*, *asset value*, and so on.

Before we get too complex, I believe it's worthwhile to show how we might create an extremely simple *asset register* for our organization. It might look something like this to begin with:

ID	Name of Asset	Asset Description	Notes

Essentially, in this document, we're going to give each asset an ID, a short name, a description, and any notes that may be associated with that asset. We could add columns for "asset owners," "asset classification," or "asset value" as required. It depends on your requirements and how you would like to structure the various documents we will cover in this book.

If you're looking for some inspiration for what assets you might put into your *asset register*, some example *information assets* could include *information systems*, such as *servers, software, payment terminals, routers, or laptops*. However, this can also include *information* itself, such as *datasets, images, blueprints, formulas*, or other *intellectual property or (IP)*, for example.

Now that we've covered assets, let's move on to understanding *threats*.

Understanding threats

Looking back at that one long sentence about risk, we can see that you also cannot have a risk without a **threat**. *Threats* and their *threat actors* (the perpetrators of acts that represent threats) could include *malicious hackers* gaining access to company records, a *rogue employee* selling secrets to *competitors*, *severe storms* obliterating your office locations, or the nice lady in accounting opening a PDF file that releases ransomware across the entire organization's estate.

Each *asset* can have multiple *threats* associated with them, and each *threat actor* could have different goals and abilities, which means they can utilize different methods of exploitation. With different methods comes different likelihoods and different mitigation controls, so each viable *threat* should be considered for each asset in an activity called *threat modeling*, where we document the various threats and threat actors that the assets at our organization face.

To start with a default set of *threats*, consider the following: **malicious outsiders**, **malicious insiders**, and **accidental insiders** have *threat actors*, while **environmental threats** do not.

When thinking about threat actors, consider their ability, their interest, and their tactics. Taking *malicious outsiders* as an example, you could also get more detailed and split them up further; for example:

- **Sole actors** could be experienced or inexperienced, but with the public release of sophisticated tools from major hacking groups and governmental organizations, even those actors that used to be called "Script Kiddies" are going to be able to access or change your information, or even make it unavailable.

- **Group actors** are a potentially distributed team with funding, and could include hacktivists, criminal organizations, or even competitors.
- **State-sponsored actors** have time, state-sponsored funding, and state-sponsored tools, and are the most sophisticated threat, but they have specific interests that might not be shared with the other two groups.

Let me stress that there is no clear-cut "right way" to do this, so you need to make sure to create something that suits your organization. If state-sponsored actors aren't currently a relevant threat actor for your organization, you could create a placeholder for them in your documentation, and note that they are not currently a relevant threat. In the future, if a previously non-relevant threat does become relevant, you already have the placeholder ready to be changed with new information.

Another thing to mention here is that (obviously) **weather patterns** such as hurricanes and floods don't have *threat actors*, and therefore don't have interests or tactics. However, they do have the ability to impact information security, through availability outages or outright data loss. You can understand their likelihood and if your assets are affected by them as a result. Floods cause damage in basements, which means the information assets that are stored in your basement have a higher likelihood of being affected by that *threat*.

We're going to look deeper into threats, including the various threat actor categories and their motivations, in *Chapter 3*, *Designing Secure Information Systems*, so let me get back to discussing the basics around risk.

Understanding vulnerabilities

Now that we've covered assets and threats, the next piece of a risk we must understand is their **vulnerability**. You can't have a risk without a *vulnerability* due to the following (as we've mentioned previously):

"Information security risk is the potential for loss, as measured through the combined impact and likelihood of a threat exploiting a vulnerability in one or more information system assets."

So, with *vulnerabilities*, we're talking about the weakness of an asset that can be exploited by a threat. Vulnerabilities can vary by asset category, with hardware assets being susceptible to overheating or wear and tear, while software assets can be vulnerable through flawed code. We will delve deeper into what constitutes a vulnerability and how to mitigate them throughout this book, so I'm going to stop myself at that simplified definition before going down that rabbit hole.

We have now covered some of the key concepts in risk management, and why risk management is important as an information security professional. The next thing I'd like to cover is how we might actually perform a very basic risk assessment in practice, and how we can use risk assessments to understand the risks that our organization faces.

Performing a basic risk assessment

In this section, we're going to briefly cover how we might calculate a risk score in practical terms by utilizing the previously mentioned formula of *Risk = Impact x Likelihood*. By covering this now, we'll be able to dive deeper into this topic in *Chapter 2*, *Protecting the Security of Assets*, and be able to answer questions such as the following:

"Which mitigating controls do we have in place to protect our organization's three most valuable assets?"

However, from the perspective of what we've covered so far, we simply aren't ready to delve deeper into the topic of how we might implement an **Information Security Management System (ISMS)** into our organization. With that being said, we will be, by the end of the next chapter! Until then, it's worth covering how we can assess risk in a very simplified fashion.

Let's imagine that you've been hired as an information security officer for *The Ketchup Company*. Your crown-jewel asset at The Ketchup Company is your secret recipe for ketchup. Consider that The Ketchup Company's top competitor (who sells absolutely terrible ketchup) somehow manages to gain access to your organization's secret recipe and steal it for themselves. What would the impact of that happening be? What is the likelihood of that being possible?

We know that risk is the combined impact and likelihood of a threat exploiting a vulnerability. By understanding our assets, their vulnerabilities, and the threats they may face, we can begin to calculate the risk.

Just for this example, I can tell you now that The Ketchup Company's ketchup recipe is stored on the organization's **network attached storage** (**NAS**), which is accessible from the outside world by visiting the domain "`http://networkstorage.theketchupcompany.com`" through a web browser, with no username or password required for access. Furthermore, I can say that our top competitor being able to access the secret ketchup recipe would cost The Ketchup Company over 5% of its annual revenue per year.

First of all, I'd like to take this opportunity to add two assets to our asset register at The Ketchup Company, like so:

ID	Name of Asset	Asset Description	Notes
1	Ketchup Recipe	The moneymaker for our organization.	The recipe is stored in a PDF file on the company's Network Attached Storage.
2	Network Attached Storage	Storage for all the organization's important documents.	You can access the Network Attached Storage via a browser by visiting `http://networkstorage.theketchupcompany.com`.

Okay; some basic housekeeping is in order. Now, let's use this example scenario to perform a basic risk assessment, and practically cover some of the concepts we've gone into so far.

Defining and calculating impact

In order to calculate risk, we need to define the way your organization sees the various levels of *impact* that it could potentially face as a result of a *security event*. These values will depend on your organization and should be discussed with the appropriate stakeholders, who have a detailed understanding of its finances and operations.

For a simple starter, try this:

The organization losing 5% or more of its annual revenue due a security event is considered "catastrophic," while 3%-5% is "high," 1%-3% is "medium," 0%-1% is "low," and under $300 is "negligible."

Using those definitions, we can create an *impact definition table*, like so:

Impact	Score	Description
Negligible	1	The organization loses up to $300 per occurrence.
Low	2	The organization loses 0-1% of its annual revenue each occurrence.
Medium	3	The organization loses 1-3% of its annual revenue each occurrence.
High	4	The organization loses 3-5% of its annual revenue each occurrence.
Catastrophic	5	The organization loses 5% or more of its annual revenue each occurrence.

This is a considerably basic and immature example of how we might go about defining the potential impact of a security event occurring. Percentage of revenue might not be the best measure, so you can always consider defined sum ranges, or percentage of net income, and so on. It's not necessarily perfect, but this example lets us dip our toe into risk management, to set the scene for expanding various risk-based topics in rest of this book.

Defining and calculating likelihood

Next, in order to understand *likelihood*, we must define the way our organization sees various levels of likelihood. Is one occurrence per year a high level of likelihood, or a low level of likelihood for your organization?

For a simple starter, try this example:

A likelihood score of 5 is defined as "once per year or more," a score of 1 is "once every 10 or more years," and the years in-between get 2, 3, and 4.

Refer to the following table for an example of a likelihood definition table:

Likelihood	Score	Description
Improbable	1	Once every 10+ years
Unlikely	2	Once every 7-9 years
Possible	3	Once every 4-7 years
Likely	4	Once every 2-3 years
Certain	5	Once per year or more

With that, we've successfully defined the way we score *likelihood* for *The Ketchup Company*. Remember, it's up to you to determine the best way to define impact and likelihood at your organization, but again, as we've said, this is just to set the scene for expanding various risk-based topics in the rest of this book.

Calculating risk

Now, we can go back to our example, where we will try to determine the *likelihood* and *impact* scores of a competitor accessing The Ketchup Company's secret recipe for ketchup. Using the details I provided earlier, we know that the recipe is stored on the company's NAS, and that this NAS can be reached from the outside world. Actually, we know that anybody who visits the domain "`http://networkstorage.theketchupcompany.com`" in their browser can access the recipe, without any username or password requirements, and we know that a competitor being able to access this recipe would cost the company over 5% of its annual revenue.

First, let's determine this example's impact. Referring to the table that we created for our *impact* definitions, we can see that a 5% loss of annual revenue would mean the impact score related to that event is "5," or "Catastrophic."

I've included the table for illustration purposes here:

Impact	Score	Description
Negligible	1	The organization loses up to $300 per occurrence.
Low	2	The organization loses 0-1% of its annual revenue each occurrence.
Medium	3	The organization loses 1-3% of its annual revenue each occurrence.
High	4	The organization loses 3-5% of its annual revenue each occurrence.
Catastrophic	5	The organization loses 5% or more of its annual revenue each occurrence.

Now that the *impact* score has been determined, the next step would be to determine the *likelihood* of that event occurring. One way you might try to determine likelihood is through historical records. You might ask the following questions to stakeholders at your organization:

- Has this event happened before?
- How often has it happened?
- When was the last time it happened?

Sometimes, there is no historical data available, if your organization hasn't previously kept a record of breaches or information security events, or they haven't had the adequate level of visibility to know if an event has occurred.

Sometimes, your organization gains a new asset with new vulnerabilities, or maybe new threats arise, and therefore new risks arise as well.

In these cases, it's important to investigate and understand the implications of assets, their threats, their vulnerabilities, and the impact and likelihood of various risk events. We will be doing this as often as necessary to understand the risks the organization faces.

Let's look at our example system. Essentially, accessing the recipe can be done from a web browser. The URL for accessing the recipe could easily be shared outside the organization for anybody to see. Additionally, it might be able to be found via Google search results, or an automated system finding the `networkstorage` subdomain for our `theketchupcompany.com` domain. No username or password is required for anyone to access the recipe.

In this circumstance, let's say that we consider our *threat actor*'s *motivation* (in this circumstance, a competitor with **absolutely terrible** ketchup) and *capability* (it has been previously rumored they've stolen their mustard recipe from The Mustard Company). With that information, we can determine that with the current configuration, and the current lack of any *security controls* to protect the recipe from unauthorized access, our competitor is likely to find the recipe *within the next year*.

If we refer to the *likelihood* definition table we created earlier and consider the information surrounding likelihood we provided previously, we can determine that the likelihood score associated with this specific example is a "5."

See the following table for a reminder:

Likelihood	Score	Description
Improbable	1	Once every 10+ years
Unlikely	2	Once every 7-9 years
Possible	3	Once every 4-7 years
Likely	4	Once every 2-3 years
Certain	5	Once per year or more

So, we now have both an *impact score* and a *likelihood score* for our example. Next, we must calculate the risk. Before we do that, it is worth showing a visual representation of the risk score, based on varying levels of *impact* and *likelihood*. If we were to illustrate the previously mentioned formula of **Risk = Impact x Likelihood** as a **risk matrix**, we would see something similar to the following:

			Impact				
			Negligible	Low	Medium	High	Catastrophic
			1	2	3	4	5
Likelihood	Improbable	1	1	2	3	4	5
	Unlikely	2	2	4	6	8	10
	Possible	3	3	6	9	12	15
	Likely	4	4	8	12	16	20
	Certain	5	5	10	15	20	25

Figure 1.2 – Risk matrix

Let's dissect this diagram:

- The *x* axis represents *impact*, including the numerical "score" associated with each of the defined levels of impact from our previous table.
- The *y* axis represents *likelihood*, including the numerical "score" associated with each of the defined levels of likelihood from our previous table.
- By multiplying a number from the impact axis with a number from the likelihood axis, you get a risk score. In this case, it ranges from 1 to 25.

This is a common way of illustrating the risk formula we've mentioned several times in this chapter, as it helps visualize the incremental increase in the level of risk based on the various levels of *impact* and *likelihood*.

If we were to perform the calculation for our example, where we determined the impact of the event to be a score of "5" and the likelihood of the event to be a score of "5," how would we do that? Yes – it's as simple as multiplying "5" by "5," in order to arrive at a risk score of "25."

Okay, but now what do we do? Is that level of risk okay? Is it unacceptable? Let's look into the next steps.

Risk appetite, risk treatment, and risk acceptance

It's up to your organization's stakeholders to determine the acceptable level of risk, which is also referred to as the organization's **risk appetite**.

Obviously, we are going to want to appropriately define our impact and likelihood to reflect the risk appetite of the organization. If a risk is determined to have the highest score possible, as with our example, but the stakeholders don't seem to mind, this means that their definitions of impact and likelihood aren't being reflected appropriately in our tables defining each.

One more thing we might notice in the risk matrix diagram is the dark black line that separates the risk scores above "7" from the others. This is the *risk appetite* or **risk acceptance level** being represented visually on the risk matrix. In this circumstance, the organization's stakeholders have decided that any risk with a *risk score* above "7" is considered *unacceptable*, and requires further **risk treatment**.

Risk treatment is the way we select controls to modify the risk level. We can "avoid," "reduce," "transfer," and "accept" risk. Let's take a look:

- **Risk avoidance** is achieved by removing the likelihood of the risk entirely, generally by not performing any of the actions that would lead to the risk.
- **Risk reduction** is performed by reducing either the impact or likelihood levels, generally through mitigating controls.
- **Risk transfer** is done by shifting all or part of the risk to a third party, through outsourcing and insurance, for example.
- **Risk acceptance** occurs once the organization acknowledges that a risk doesn't require further mitigation, or it is unfeasible to mitigate the risk further.

Mitigating controls are the methods by which we try to protect each asset from threats. They can include all kinds of methods and tools, but if we continue using the example of The Ketchup Company's ketchup recipe being accessible over the web, a form of *risk reduction* could be to implement a *mitigating control* that requires a password in order to access the secret recipe.

Once that mitigating control is implemented, your **residual risk** is the risk level after the mitigating control is applied, because the mitigating control has modified the risk by reducing the likelihood of the threat being able to access the recipe.

Once the *residual risk* has been reduced below the *risk acceptance level*, we can consider the *residual risk* level to be acceptable, and therefore we achieve *risk acceptance*.

Let's consider further vulnerabilities, such as the following:

- A flawed password entry page
- A weak password, "`hunter2`"
- A direct link to the recipe that doesn't require us to log in

We will need to calculate risk scores for each of these to understand the risk facing the organization. To document the various risks, we can create a **risk register** for the organization. By doing so, we will be building out the organization's *ISMS*. Doesn't that sound like the most fun you've ever had?

Like we said at the start of this section, we will delve deeper into how to go about structuring both an **asset register** and a **risk register** in *Chapter 2, Protecting the Security of Assets*. For now, it makes sense to move on and discuss another concept that is going to prove to be useful for the next chapter: the various legal regulations and compliance requirements facing an information security professional.

Considering legal regulations, investigations, and compliance structures

In information security, there are some red tape and regulatory structures that can be tricky to navigate, or stressful to be a part of. When we're considering compliance, in terms of regulatory and legal requirements, audits, questionnaires, and responsibilities, it might seem like an entirely different job all in itself. Oftentimes, organizations aren't going to have somebody who is focused entirely on those compliance structures, and as a result, you are going to need to feel comfortable in navigating, potentially more than anybody else in the organization.

I would like to discuss how these structures will be a part of your day-to-day responsibilities, and how we can leverage the predefined requirements to determine an acceptable level of risk for our organizations. Furthermore, before we conclude this section, I would like to highlight the importance of continual improvement as a requirement for compliance, and how this requirement enables optimization to be a key focus for you and your team.

Compliance structures

Something that we need to cover is the functionality of the information security team in terms of understanding the requirements for your organization's information security program. These could be from governmental authorities and legislation or certification bodies. For some standard examples, a huge share of organizations are responsible for being compliant with regulations and acts for **privacy** when it comes to processing the personal data of individuals. There are hundreds currently in place globally, and there will be even more by the time you are reading this book. For a few examples of privacy regulations, we can look at the following:

- **The General Data Protection Regulation (EU)** (**EU GDPR**)
- **The Children's Online Privacy Protection Act** (**COPPA**)
- **The Health Insurance Portability and Accountability Act** (**HIPAA**)
- **The California Consumer Privacy Act** (**CCPA**)

These acts and regulations are seen as legal obligations, and there can be huge financial and reputational repercussions for not complying with their requirements.

Most of the privacy regulations have a focus on transparency when it comes to processing **Personally Identifiable Information** (**PII**) or **Protected Health Information** (**PHI**). Some are focused on young people, while others are focused on people in specific jurisdictions, such as California, Brazil, or the EU. Most of them will require you to give your data subjects the ability to *review and revoke access* to any personal data that is being processed by your organization, as well as requiring your organization to be open about how and where the information is being processed, and by whom.

In addition to those governmental compliance requirements, there are standards that are offered by bodies such as the **International Organization for Standardization** (**ISO**), which aim to create a baseline of minimum viable levels of information security. These could include standards such as *ISO 27001*, which we've briefly mentioned already, which focuses on how to manage information security, or *ISO 27018*, which focuses on providing a code of practice for cloud privacy, and so on. Generally, organizations that comply with the requirements set forth in the standard are able to be audited by a third-party certification body, and certifications or accreditations can be earned for compliance with the requirements of that standard.

These certifications are highly valued by customers (both potential and existing) and partner companies, especially in the **Business-to-Business** (**B2B**) world, as a testament by a third party that the minimum baseline of security at the organization has been achieved. Many organizations that are selling their SaaS to other businesses, for example, will show that they hold a **ISO 27001** certification from a certification body such as the **British Standards Institute** (**BSI**), Deloitte, or a similar consulting firm. The various standards have different purposes and suit different organizations.

In order to stay on top of all the requirements, it's highly valuable to use a structured solution to catalogue any and all the requirements at your organization, and quickly report on the current compliance of your environment. Changes to these regulations can occur on a daily basis, and depending on your organization, this could mean that keeping up to date requires a team of people to be distributed globally, working full time and translating legal requirements to a singular compliance requirements matrix, which is then internally audited by the organization. At the time of writing, there are over 200 updates per day from 750 regulatory bodies.

There are many software solutions that are currently offered by various suppliers. Essentially, what they do is simplify your compliance management process by providing automated and continuous assessment scans through your environment(s) to monitor your data protection controls, as well as giving you the functionality to assign tasks and record progress against each requirement. The tools may give you recommended actions and instructions for how to implement controls that improve your compliance posture, and map your existing controls to any new requirements in the evolving compliance landscape. Large organizations are continually being exposed to compliance risk, which could spell millions of dollars/euros/pounds per year being lost, so paying for a tool such as this makes sense for many businesses.

Understanding legal and regulatory requirements

A lot of the requirements in regulations or standards are not exactly the type of thing a software solution is even able to automatically track. How could a software solution know if you, for example, turn off access to all accounts and services during the exit interview of an employee that is leaving the company? Keeping on top of how your organization actually manages the various compliance requirements often requires working with every team in your organization. Yes, this does include legal and HR teams, and will include doing internal audits to ensure your organization does what it says and says what it does.

All these controls are relevant to your information security program and should be documented and reviewed with updates on a regular basis. When I say on a regular basis, you can take that however you want, but keep in mind what the compliance requirements are for your organization. Some standards require proof of annual review and updates to policies and controls, for example.

So, imagine that the employees at The Ketchup Company are on a bit of a hot streak, and they have managed to move what turned out to be a huge amount of PII into the cloud. Some of the data subjects are EU citizens, while others are based in California, and as a result the lead-up to this migration has included determining the following:

- The type of processing and storage that was being done
- The reasons for this processing and storage
- The basis for which they were legally allowed to process and store the data
- Updates to privacy policies and documentation
- Notifying the data subjects regarding the change

Most of the data subjects were probably surprised that their favorite ketchup company held data on them, but in this data-driven society, it would be irresponsible for The Ketchup Company to not understand their customers.

A few of the data subjects decided they didn't want their data being stored by The Ketchup Company and filed a **subject access request** (to which The Ketchup Company complied with in the permitted timeframe), as well as a **data deletion request** for their data (to which, again, they complied with in the permitted timeframe). It sounds like The Ketchup Company has their privacy compliance requirements running smoothly and efficiently!

Compliance isn't all about privacy or certifications to standards, however. As we mentioned previously, there are many different types of requirements from acts and regulations globally, such as the **Sarbanes-Oxley Act** (**SOX**), which requires companies to retain their financial records for up to 7 years, or the **Federal Information Security Modernization Act** (**FISMA**), which mandates all federal agencies to develop protection methods for their information systems. Understanding how the various frameworks, standards, regulations, and laws apply to your organization is an important task for the information security team at your organization, in order to avoid fines and penalties, as well as to provide a good baseline for protection against information security risk.

Responding to and undertaking investigations

Even if you're part of a simple organization with very few employees, there is still a chance that events will either occur on your estate (or in a way that is related to your estate) that require further investigation by yourself, somebody in your organization, or by another organization. This could include anything from finding out who deleted a file from a shared drive to a legal investigation on insider trading, or handing over server logs to law enforcement for them to perform forensics. It is your responsibility as not only an employee, but also as a law-abiding citizen, to comply with these requests to the extent in which you are legally obligated. The last thing you want is to be found obstructing an investigation; it's just simply not worth it.

Many solutions exist so that you can comply with these types of requests, including actions such as placing legal holds on email inboxes to "freeze" activity, or storing snapshots, or ensuring that items such as emails or files aren't actually "deleted" when an employee deletes them. Organizations have more control than ever before, with the help of software and information systems, to ensure that their employees are abiding by the both the law and the protocols devised by the senior managers in the company's defined policies.

When it comes to internal investigations on unauthorized access, or some sort of integrity issue, sometimes, it might be important to ensure that those aiding your investigation aren't able to find out who and what is being investigated. It sounds extremely tricky, but again, there are tools that provide **pseudonymization**, which gives us a way to track the activity of users by assigning aliases. This prevents investigator bias or collusion with the subject.

Another interesting legal aspect of the information security compliance domain is that of **eDiscovery**, or **electronic discovery**. eDiscovery is a way to identify and deliver information that can be used as legal evidence. By leveraging eDiscovery tools, you can find content in mailboxes, shared groups, Teams/Slack/Skype chats, shared drives, and device hard drives. You can finely filter your search results in order to identify and hold and share the results with the parties required, but not extra data that isn't relevant to the investigation. It reduces costs, time, and complexity to use eDiscovery tools during these events. If you are aiding in this process, you will be working with people who make it their business to be great at this work, and providing your knowledge of your estate and its structure is likely going to be your responsibility.

Further compliance optimization

Remember, a huge part of most information security standards is the principle of **continuous improvement**. We have a long way to go in order to make operating in a digital environment secure, and optimization needs to occur constantly! We need to put ourselves into an engineer's state of mind and see the world through those critical eyes. Throughout the rest of this book, I'm going to be reminding you of the requirements for *continuous improvement*, as well as give you examples to apply to your estate.

For a quick example, let's go back to our example organization, The Ketchup Company. Imagine you (as the information security manager) run through an incident response playbook for restoring from a backup in the event that an on-premises server for running a few applications has a hard drive failure. During the simulation, you notice that the spare hard drives that you have stored for this event will be at 90% capacity once the data is restored from backup. You also note that a better state of redundancy than RAID-0 would have prevented this from happening in the first place.

First, you take a look at your *risk register*, and you notice that you overlooked this previously. You add the details of the *risk* to the register, and perform a *risk assessment* to calculate the risk score determined from the *impact* and *likelihood* scores, and find that the level of risk is above the formally defined *risk acceptance level* for The Ketchup Company.

After going through some discussions with team members, as well as giving it some individual thought, you decide that at the moment, with the budget you have available, you can buy some hard drives and set up some redundancy as a mitigation tactic. You make a support ticket to purchase enough hard drives to quadruple the capacity, create another one to add them to the server, and then increase the redundancy to RAID-1. Is it the perfect solution? No; it's an incremental optimization that reduces the risk of downtime on your server and allows your staff to focus on important things for the future, rather than fighting fires in the present.

Are you done? Of course not. From that optimization activity, you now have some updates for your asset register and risk register, as well as some further investigation for other servers and systems. If you find that one of your servers was running RAID-0, is that likely to be isolated to just that single server? Or are you going to lift the curtain and find that The Ketchup Company has absolutely no redundancy in any of its servers and backups, and now there's a high-priority IT operation required to prevent a catastrophic failure? It's going to be somewhere in-between those two extremes, most likely… but the investigation and findings lead to a project for your organization that can be broken down into small, achievable goals that amass together to form a sum greater than its individual parts. An optimized solution is something you can get funding for by presenting the level of risk facing the organization if the vulnerability is level unmitigated.

Once the project is completed, you perform another *risk assessment* to see if that new optimized level of *redundancy* mitigates the risk of data loss to an *acceptable level*, record your results, and present the results of the project and the economic impact of the improvements to the relevant stakeholders. It sounds like you're using ideologies from *risk management* and continually improving your organization's **risk posture** – nice work!

Proven methodologies in creating a strategy

In this section, we're going to cover some proven methodologies that ensure a high level of success for creating an information security strategy. We'll discuss creating information security policies, procedures, and playbooks, as well as establishing and maintaining a security awareness program, managing third-party risk as a responsibility for any information security professional, and reporting and continual improvement as a focus for highlighting the progress that's been made, which is something an information security professional must consider in an era of growing cyberattacks and increased reliance on digitalization. With these topics in mind, you are setting yourself up for success, with the opportunity to improve and cater the strategy to any long-term goals or changes to the landscape. Let's begin by talking about creating an *ISMS*.

Creating InfoSec policies, procedures, and playbooks

Information security policies, standards, baselines, guidelines, procedures, and playbooks are the backbone of any information security strategy. We have already established that point in this chapter. With that said, how do you make those? How can an individual person manage to create all these documents, full of terms, guidance, and policies for an organization such as The Ketchup Company?

Creating an **ISMS** is an important undertaking for you and your organization to have the appropriate level of visibility into its *information security risk*. I am going to cover the process of creating an ISMS further in *Chapter 2*, *Protecting the Security of Assets*.

Until then, keep in mind that the key to finishing a marathon, a hill climb, or any other metaphor for the daily slog of your job is the same: you begin by taking a single step. Taking each hour at a time, and plugging away at creating a structure suitable to your organization, is going to ensure better visibility and fewer unknowns. When you encounter information security-related topics in your day-to-day conversations and experiences at work, write them down and keep notes. Notice what people say, do, and what happens most often, and use that information to help in your *ISMS* project.

Establishing and maintaining a security awareness, education, and training program

The information security policies need to be understood by all the members of the organization. The requirements need to be established immediately to any new member, and regular reminders need to be provided afterward. Furthermore, keep in mind that you will need to train key staff members on the entire information security program, and other staff members on key points that cater to their jobs. You want to make sure the organization isn't relying on you (and only you) in the event of any outage. You do want to be able to take a holiday, don't you? Beyond that, you want the business to be able to react even after you're long gone, and you want staff members to be aware of their responsibilities when it comes to various information security topics and scenarios.

As a result of that, you are going to want to create training materials to ensure the members of your organization are educated and trained on their requirements from an InfoSec perspective. With new members, depending on the nature of your organization and its size, you might set aside a few hours to introduce yourself and talk about the information security policies of the organization, or you might need to formalize the process a bit more into documents or training videos in order to scale. Generally, new members of an organization will sign an agreement to uphold the policies and practices that have been covered, and that agreement is kept on record and updated annually.

Many people in this industry say that employees are the biggest vulnerability to your organization, and imply that it's the fault of the employee if they are tricked by a phishing scam, for example. To that, I say, *"How can we blame the victim?"* Let's look at a few questions we might want to ask:

- Was there adequate training in place?
- Were there phishing simulations conducted to identify the at-risk employees?
- In their day-to-day work, was there a secure method of performing their tasks available to them?

It is likely that the information security team's responsibility is to ensure the members of the organization are educated on the threats they face. There are so many effective ways to turn a person who would have originally been a liability into a champion for your security strategy, just by showing them how interesting the field is, and by providing incentivization. This is the value of creating engaging and interesting training materials, and this idea should never be neglected in your overall information security strategy.

Managing third-party risk

To be perfectly honest, the cloud is the way things are headed. Newsflash, right? Even the slowest, most risk-averse businesses are heading to the cloud. Banks are moving to the cloud! Governments are moving to the cloud! You name it. From **Amazon Web Services** (**AWS**), **Google Cloud Platform** (**GCP**), or *Microsoft Azure* for hosting, to *Office 365* and *G Suite* for documents, to millions of different SaaS products for project management, source code repositories, accounting software, HR management, data loss prevention, SIEM, and the list goes on and on, the solutions that are being offered to manage information in the modern era are increasingly shifting toward the Shared Responsibility model, and part of the risk your organization faces becomes a third-party responsibility to mitigate. How does that third-party cloud accounting software secure their estate? Do they make that information available? Just because you're not hosting the software on your own servers doesn't mean you are relinquished from all responsibilities when it comes to security, compliance, due diligence, and risk reduction. Your organization's data is being processed by assets under the third-party's control, but it's still important to consider as part of your responsibility.

Utilizing various **Software-as-a-Service** (**SaaS**), **Platform-as-a-Service** (**PaaS**), and **Infrastructure-as-a-Service** (**IaaS**) solutions presents new requirements: crossing international borders with PII is now potentially breaking the law. A third party storing your organization's data in plaintext could be catastrophic in the event of an insider threat viewing that information. Running your platform on a third-party server that, unbeknownst to you, is unpatched and vulnerable to exploits that are readily available on the web? There's another risk to your organization's success. How can you possibly mitigate this? Isn't it the responsibility of the **cloud service provider** (**CSP**) to update their systems or encrypt your data? Not unless they have agreed to such, most likely. How can you ensure an organization says what it does and does what it says?

First of all, you'll be happiest when you have legally binding **Service-Level Agreements** (**SLAs**). SLAs are a promise from a third party that they will keep their service to a certain standard of availability, confidentiality, and integrity, among other things. As an example, often, a cloud IaaS SLA might deem that in the event of an outage that causes availability to drop below 99.9% (or 99.99%, or 99.999%) for the month or year, the third-party will reduce the costs to their client. This is sometimes referred to as "three nines," "four nines," and so on. It's a way for your organization to mitigate availability risk by offsetting the monetary loss experienced from that loss of availability. In a way, your SLA is a legally binding control that you are able to report on and ensure is effective. SLAs are negotiable in the sales process and can sometimes go back and forth between the client and supplier multiple times, with redlining and pushing back from either side before an agreement is made. If you work at a CSP, you might find yourself as the information security professional who's responsible for negotiating the SLAs on the other side as well.

In addition to SLAs, you could use what is known as a **Vendor Security Assessment Questionnaire** (**VSAQ**), which is able to be completed by any current vendors or prospective vendors. What these VSAQs do is ask a standard set of questions for due diligence purposes and record-keeping. The response to these questionnaires should be considered during the procurement phase, with risk as the backbone of that analysis. Generally, a VSAQ will go over the security program of the vendor, any controls they have put into place for protecting their estate, the training and awareness that the employees undertake, and the types of audits (along with the audit results) that the vendor regularly undergoes. VSAQs are a way to get an account on record from the person in charge of information security at the company you're paying to process or store your organization's confidential data, business processes, and PII or PHI.

It is your obligation to show that due diligence and due care has been put into each and every decision for outsourcing data processing and storage to a third party. In the event of a breach at the third party's estate, you want to be able to show that you made an informed decision based on the testimony of the third party's questionnaire answers. If they don't treat a specific threat the way you would like, make a note of it, and potentially even discuss the matter with their information security team. You might find out that further improvements are actually on the roadmap.

Don't worry about being a pain when discussing SLAs or VSAQs. Any vendor worth your time has had to answer similar questions and negotiate similar terms with other organizations, and might even have those answers prepared. If you're working at a CSP and are responsible for answering these VSAQs yourself, then you would be very right in cataloguing your responses to those questionnaires, along with creating a VSAQ FAQ. It will save you countless painful hours of repetitive work. In previous roles at SaaS vendors, I've had a privacy and security whitepaper prepared, which answered every question asked in the VSAQs, with drawings and technical details provided, which I would then initially pass to each prospective customer for them to look through and fill their own questionnaires out. It was very helpful in the months leading up to GDPR's effective "start date," where a thousand customers were scrambling to prove due-diligence and due care by crafting their own version of a VSAQ and sending it out to all their suppliers.

Google's VSAQ web app (`https://opensource.google/projects/vsaq`) allows a vendor to either complete their answers via the app or load their answers from a file, and then submit their responses to standard VSAQ questions. Unfortunately, I haven't seen nearly enough vendors or customers using this format. It would save so much time to just standardize and create the "gold standard" VSAQ and let us all get on with making our organizations more secure. Another type of questionnaire is the Cloud Security Alliance **Consensus Assessments Initiative Questionnaire** (**CAIQ**), available at `https://cloudsecurityalliance.org/artifacts/consensus-assessments-initiative-questionnaire-v3-1/`.

Keeping this information in your ISMS is important in the event of a third-party breach. As we mentioned previously, you want to be able to show an appropriate level of **due care**. If you can show that the risk has been measured and deemed acceptable, it could help show that you weren't being negligent. It may seem redundant, but regulators and stakeholders will want to know the details that led to a decision, because they may have to answer to board members, stakeholders, customers, regulators, law enforcement, or governmental entities on the matter.

Continual improvement and reporting

Another important thing to remember is the ideology of continual improvement. New exploits become available, new techniques surface, and systems change constantly. As new vulnerabilities are discovered, and as new threats arise, all of the implemented structures and controls for reducing information security risk at your organization need to get better. It's important to keep up to date with what is happening from a threat and vulnerability point of view, as well as ensuring your asset and risk registers are up to date.

If any of the organization's controls fail or can be circumvented, where does that leave you? If the answer is "wide open," then you don't have a sufficient amount of **defense-in-depth**. It's an important concept to grasp: double-locking. Adequate defense-in-depth might not reduce risk, but instead allows for a control to fail and you to still have a reduced *residual risk*.

In addition to *defense-in-depth*, looking at the world through an engineer's lens is highly beneficial to an information security professional. You can improve processes and systems to make things more lightweight, faster, automated, effective, scalable, accessible, and less expensive with fewer errors. Reducing the costs of one control allows you to build more defense-in-depth and reduces complexity, which is the enemy of security.

In order to chart your progress, you could keep legacy versions of your risk register, potentially by financial quarters, or halves, or years. It will give you a great dataset for the impact you've made at your organization, and will help you justify that pay rise they've been hanging over your head for the past year. Make graphics, put them into PowerPoint presentations, and regularly update your stakeholders at your organization with the progress, wins, and blockers. Depending on your organization, and if you're sitting in the top information security role, you could include the CIO, CTO, CFO, and potentially even CEO in these conversations. Also, the legal and HR departments will have an interest in the topic of insider threats, legal obligations, and other compliance requirements. Don't waste their time, and make sure to keep things non-technical and high-level, catered for C-levels who only care about the bottom line: "How much does this cost, and how much does this save?" You can show your program's return on investment by utilizing risk reporting.

Sometimes, you'll find some really interested stakeholders in these meetings, who are impressed with the findings. Other times, you'll find some people who don't really care, and are just there because they have to be. Make sure to give all your audience members a reason to care. The CFO is going to care about different things than the CTO, but they both have their own reasons for wanting to keep the lights on and the money coming in.

Summary

This chapter has gone over the major topics that heavily influence decisions made by information security professionals: risk management and governance structures. When I introduced this, I said that it probably didn't sound like a barnburner, full of thrills and excitement, but I'm sure now that you're reading this summary, your view on the matter has changed slightly. I would say that with a bit of practice in mastering the basics found in this chapter, my promise to you that you will be a highly effective, well-oiled risk management machine in no time will come true.

In the next chapter, we'll be looking at protecting the security of assets, which is now a much more achievable task. You have an understanding of various core concepts, and we're going to proceed toward leveraging everything we have covered in this first chapter to create a more mature ISMS, as well as develop your skills further by focusing on effective processes to ensure you can identify and protect your organization's assets throughout their life cycle, avoiding some common pitfalls that information security professionals often run into.

Let's do this!

2
Protecting the Security of Assets

Originally, the concept for the start of this chapter was to ask you *"How can you protect an organization if you don't know what assets they have?"*. We both know the answer to that question is *"You can't,"* and we've probably been asked this question a thousand times between the two of us, so I'm not going to ask it. Instead, I'm going to show you how you might structure the appropriate processes at your organization in order to discover and protect its assets.

These various processes combine to create what is known as an **Information Security Management System** (**ISMS**). We covered various key concepts in *Chapter 1*, *InfoSec and Risk Management*, but overall, there was much less structure than most organizations would require. What we want to be able to achieve with the ISMS is to appropriately identify and classify our organization's assets, and ensure the security of those assets is adequately protected by implementing the appropriate controls, taking into consideration *defense in depth* for each vertical of confidentiality, integrity, and availability.

This chapter focuses on those topics. We will look at utilizing effective processes to implement an effective ISMS. One that ensures, through policies and procedures determined by business requirements, that risk is reduced to an acceptable level.

With that goal in mind, I think it makes sense to follow these four stages in the structure of this chapter:

- How to implement an ISMS
- The appropriate process for the identification and classification of the information assets of an organization
- Securing those assets with the appropriate controls based on their value, monitoring for changes, and adapting to those changes
- Disposal of the assets, be it through archiving or destruction

All of these points will be covered in a way that helps you avoid the common pitfalls that InfoSec professionals often run into along the way.

Now that the housekeeping is in order, allow me to proceed to the actual content.

Implementing an ISMS

Implementing an ISMS requires *structure*, *planning*, *decisiveness*, and *collaboration*. There exists an extremely important question of *"Who is responsible for what?"*, which should be asked and documented. I'd like to briefly touch on the role of **top management**, and how we might translate our findings into effective communications about risk to the appropriate audience. Improving on this should allow you to act with authority in your mitigation strategies moving forward.

Once business goals are translated into IT goals, and the appropriate level of buy-in is attained, we can move onto the actual development of the policies, which will act as the information security "rules" for your organization. This is absolutely crucial in being able to systematically define "baselines" for security, organize assets, reduce risk, and communicate the information security requirements to members of your organization.

Next, I'd like to talk about evaluating and improving this policy, keeping in mind the rule of thumb for InfoSec is to *be continually improving and optimizing*. In such a rapidly changing field, keeping on top of things and adapting to changes will pay off.

So, buckle up! We're going to talk about governance and policies! I know how exciting that sounds.

Responsibilities of top management

To begin with, it's important to remember that when it comes to anything in information security, it's ultimately **senior management** who is responsible. What does this actually mean, in practice? Well, they hire you. Hiring an information security professional to delegate the responsibilities to is the most logical thing to do as a CEO who has been hearing of their growing responsibilities in information security.

Another aspect of top management's role in information security is to effectively communicate the organization's strategic objectives. With those objectives in mind, you are able to align all information security requirements to those goals.

So, now you've been hired, and it's been decided that it's your responsibility to understand the risks associated with your specific organization and communicate this to your CEO, or CIO, or CTO, or some other **C-level** in a digestible way. This is a pretty normal situation, but having said that, I've met folks who were responsible for security at their organizations who seemingly never report their findings to the C-level, and then complain that the company doesn't take security seriously. Yeah, of course not! How could they?

It is now your responsibility to assess the level of risk that faces the organization and turn it into a legitimate business case for senior management at your organization, in a way that is easy for them to understand. Keep in mind, these people are top management! Their childhood didn't consist of browsing the web; it consisted of playing stickball near Old Man River's house and drinking seltzer at the local pharmacy. As for their day-to-day use of "computers," they've probably been limited to email and (recently) a mobile device, and it all seems pretty magical to them.

My point is: You need to *know your audience*. While communicating risks to the IT department will allow for more technical references and discussions, presenting to the other business departments requires a more translated approach, focusing on organizational impact.

Generally speaking, from my experience, far too much time is spent on a technical description rather than presenting an understandable level of risk the business faces. Properly translating IT risk findings into organizational impact will ultimately add a lot of value to your career, because as it currently stands, very few IT and InfoSec professionals do this effectively.

When you're presenting your findings to the C-level, you might be able to use this example as a template for communicating risk:

- You stand to lose this much money this year: *£100,000+*
- Unless you protect this system: *NAS Server*
- From this threat: *External malicious actors accessing it through an unknown web app exploit*
- With this mitigation: *Web application firewall (WAF)*
- It will cost this much money to do it: *£84.42 / month, or £1,013.04 / year*
- And will take this long to implement: *3 days*

Now, with these communications in mind, I would like to talk about the **ISMS** we're going to create, and how we can implement structures into our organization. Then, we can gather the type of key information mentioned in the preceding template in order to communicate the level of risk, and ensure the security of organizational assets.

Developing an ISMS

An ISMS is a defined approach to handling information security at your organization. It ensures a systematic approach to accurately discover, measure, contain, and mitigate the information security risk at your organization in order to ensure your organization is adequately protected from malicious actors, accidental loss, and regulatory impact, with risk at the heart of the decision-making process.

The ISMS contains formalized documents determined by business needs, such as operational requirements, regulations, retention periods, the ability of the staff, and the business's solutions. These documents define your organization's policies, procedures, baselines, and guidance in relation to information security at your organization. This is an extremely important step in order to achieve your goal of a risk-focused, mature *information security program*. As a result, it is the information security team's responsibility to ensure this ISMS is in place and functional.

The rest of this book is going to help you in developing and adding to the content of these documents, as well as operationalizing the content, but the structure isn't set in stone in any specific way. You might want to structure these documents in a way that streamlines compliance with specific regulations. There are many information security frameworks and standards that are available, and structuring your policies to suit one or more of those will present a logical sequence of requirements. One way, for example, is to follow the requirements of the **ISO/IEC 27001** (ISO 27001) international standard for your ISMS. In this chapter, I'm going to leverage the ISO 27001 standard's requirements to help us create documents and define policies in order to build out our ISMS, but keep in mind that it's possible to leverage other standards and guidelines, as required by your organization and its context.

Essentially, we want to create a set of documents that will come together to define the organizational requirements and serve as our ISMS. The topics of these documents can include the following:

- How the responsible people within your organization create and update the information security policies, procedures, baselines, and guidance
- How the organization identifies, classifies, monitors, and disposes of information assets
- How the responsible people within your organization identify and measure the risks facing its information assets, and choose controls to mitigate those risks
- The levels of residual and unmitigated risk that exist for your organization's information assets

Often, documents in the ISMS are referenced in the configuration of new systems, during the onboarding of new employees, or in the day-to-day undertakings of various organizational members. For example, in the previous chapter, we covered various aspects of the *Risk Assessment Methodology, Risk Treatment Methodology, Risk Treatment Plan, and Risk Assessment Report* documentation that may be required for your ISMS.

To begin creating this system, we can start with a few key documents required by the ISO 27001 standard to specifically address the following topics, which I have divided into subsections. Keep in mind that it's your responsibility to understand your organization's requirements, and align the documentation and processes with those requirements to ensure the effective implementation of your ISMS.

High-level information security documentation

When it comes to high-level policies for information security and your ISMS, there are a few key documents you'll want to consider having, depending on the level of structure your organization requires. Keep in mind that with a structured approach to handling information security, we can more effectively reduce the level of risk facing our organization.

Context of the organization

In order to understand how to build your ISMS, you should be able to outline the context of the organization for which you're building the ISMS. You'll also use this document to define what the organization requires from the ISMS. By beginning with your organization's requirements, you're creating policies and procedures that align with the business goals of your organization, as defined by top management.

You might want to address any internal or external considerations that could impact the organization's goals for the ISMS. These could include the following:

- The nature of the organization, as in the type of work the organization takes part in. These questions (and more) are likely to have information security risks associated with them:

  - Is there a political nature to the organization?

  - What are the products and services the organization offers?

  - What is the growth outlook of the organization?

  - Which threats are currently facing it?

  - Which suppliers are being utilized?

  - What legislation does the organization need to consider?
- Information or assets, such as defining the crown jewels of the organization. Consider how the following affect your organization in terms of risk and information security:

  - What type of information does your organization process or control? PII, PHI, IP, financial information, or perhaps another type?

  - Which systems are currently being used to process and store information?

- People, including the nature of how work is done. Consider how the following affect your organization in terms of risk and information security:

  - How is recruitment done?

  - How are people trained?

  - When people change roles or leave the organization, how is that handled?

Having taken these things into consideration, you are better suited to design an information security management system that caters to your organizational needs.

Scope of the ISMS

Regardless of whether you're following the ISO 27001 standard or not, you will likely want to define the scope of your ISMS. The reason being that you will want to communicate to key stakeholders (such as top management, members of your organization, auditors, or potentially even your customers) which parts of your business are covered by your ISMS.

How can you define your ISMS's scope? You should align it with the scope of the organization, and any of its organizational requirements. Ask the question *"What aspects of the business will benefit from being in scope with the ISMS?"*. Liaising with key stakeholders at your organization about this topic will help you to understand the reasoning for wanting to (or needing to) implement this system to begin with, including understanding risk, regulatory pressure, or customer requirements.

You can also consider defining specific processes (or areas) of the organization that are "out of scope" with the ISMS. This could include external third-party processing activities that are out of your control – providing you still ensure that the third-party practices are in line with your risk appetite.

Generally speaking, the certification bodies that will end up evaluating your ISMS and certifying whether your business is compliant or not will prefer to see the "entire organization" fall under the scope of the ISMS, but it's not a hard-and-fast requirement to do so. Providing you "do what you say, and you say what you do" – as in, your organization follows the requirements set out in the ISMS in practice, and the practices are in line with the standard – you will be conformant with the requirements.

Statement of Applicability

Without things getting too complicated about how to structure this document, the **Statement of Applicability** is a summary of the relationship between your organization's risk, and the various controls available to reduce the risk scores to an acceptable level, and is often displayed in table form.

Considering we're taking a risk-based approach to selecting and applying controls at our organization, we will not be able to effectively complete the Statement of Applicability before performing a relevant risk assessment for our organization. However, creating the document and including a list of available controls will help provide context to that process.

Referring to ISO 27001, we would be looking at a list of 114 controls across 14 categories in what is known as Annex A, and defining whether each is applicable or not to your organization. It is possible for a control to be *applicable and implemented*, *applicable but not yet implemented*, or *not applicable*. The Statement of Applicability document should include a justification for the decisions made.

The controls found in this document include various methods of structuring your organization's ISMS and offer a great insight into how you could go about reducing risk at your organization through policies, procedures, and definitions. Utilizing ISO 27001's Annex *A* will help in the process of developing your ISMS, and bolstering your organization's security, but will not give you specifics on how you might implement the controls.

Refer to the following table with one entry included for how you may want to structure your Statement of Applicability table:

Ref	**Control**	**Control Description**	**Applicability**	**Implementation**	**Justification**
A.5.1.1	Policies for information security	A set of policies for information security shall be defined, approved by management, published, and communicated to employees and relevant external parties.	Applicable	A formal information security policy has been implemented, approved, published, and communicated. Link: information security policy	Control has been selected after analysis based on risk assessment.
...	...	...	...	...	...

In order to clearly define how your organization is taking a risk-based approach to handling information security, you will want to add an entry for each of the 114 Annex A controls. It must be decided why each control is (or is not) relevant to your organization, and if it is, how the organization has implemented (or plans to implement) that control.

For further guidance on how to go about implementing the 114 controls found in Annex A, you can refer to **ISO 27002**, a supplementary document to ISO 27001, which provides guidelines for the selection, implementation, and management of these controls.

Information security policy

A logical continuation of the high-level information security policy documents would be the **information security policy** itself.

As we saw in the preceding table, our Statement of Applicability table, the first Annex A control for mitigating information security risks is to define a set of policies for information security, and have those policies approved by management, published, and communicated to employees and relevant external parties.

The overall goal of the information security policy is to define the objectives for information security at your organization, including measurable requirements for the confidentiality, integrity, and availability of the organization's assets. Because of this, it is crucial to ensure the information security policy is applicable and relevant to your organization. There's no point in creating documents that nobody can or will abide by; these purely exist in order to reduce the level of risk facing the organization.

Another aspect of the information security policy is committing to the ideology of continual improvement that we mentioned towards the conclusion of *Chapter 1, InfoSec and Risk Management,* in order to ensure the organization remains protected as technologies, processes, and threats change.

Other key definitions and documents

Other definitions and documents you would be best suited to include in your ISMS include the following:

- A list of *security roles and responsibilities*, defining who is responsible for what, in terms of security at your organization. We want to ensure it's clear who is going to be implementing controls, or the responsibilities of data owners and data users, for example.
- A list of *legal, regulatory, and contractual requirements*, with specific references to key documents and web pages. If your organization must comply with a certain *SLA* for a specific customer, or if your organization's processed data is subject to *EU GDPR*, it will be important to keep track and document those requirements in a specific way in your ISMS.
- *Internal audit* policies and procedures, which include the specific processes internal auditors must follow in order to effectively check the effectiveness of the organization's ISMS and implemented controls.

- Policies and procedures for *corrective actions*, which include the specific and systematic processes to improve security at the organization. It should include how (and by whom) any information security issues are discovered, reported, documented, and actioned.

Furthermore, we will want to define a way to store and present the following information in our organization:

- Business impact analysis
- Records of training, skills, experience, and qualifications
- Results from monitoring and measurement
- Internal audit results
- Management review results
- Corrective action results
- Logs of user activities, exceptions, and security events

Once we have those specific definitions, we can proceed to the other parts of the ISMS, including asset management, risk management, and business continuity and incident response. Let's begin with the key topic of asset management.

Asset management documentation

Part of the reason I've started with asset management instead of risk management is that old trope you might have heard in the past: *"How can you protect your organization's assets if you don't know what assets your organization has?"*.

It's a cliché, but that doesn't make it any less true. In your ISMS, you must define a way to understand not only the information assets your organization has, but also how information should be classified, and the way those assets should be handled securely. Only then will you begin to fully understand the risk facing your organization, and be able to apply controls to reduce that level of risk to an appropriate level.

Generally, the life cycle of managing information assets at your organization will follow these four steps. I've mapped ISO 27001's requirements to each stage in parentheses:

- Adding the information asset to the asset inventory (A.8.1.1)
- Classifying that information, based on legal requirements, asset value, criticality, and sensitivity (A.8.2.1)

- Labeling that information (A.8.2.2)
- Handling that information in a secure way (A.8.2.3)

I'll begin with the first stage, the asset inventory.

Asset inventory

You will absolutely want to have a way to effectively catalog and continually update the **asset inventory** of your organization. In the first chapter, we discussed the various categories that assets can fall into, and the reason for those definitions was in order to ensure you are considering anything that is of value to the organization that can fall into the ISMS's scope, or in plainer words: any information, as well as the devices or systems where information is stored, processed, or accessed from.

Once you have a list of assets for your organization, what you will want to do is to categorize and prioritize the assets, and then assign an **asset owner** to each asset (or asset group). It's up to you and your organization's requirements as to how you might structure this, but keep in mind that the asset owner is the one who has the responsibility (and the authority) to protect the asset. This doesn't mean daily activities can't be delegated by the asset owner, but it's their responsibility to do so.

Additionally, you might want to assign values or criticality *scores* to your assets, in order to streamline the prioritization and risk management processes. Asset criticality can allow for a better understanding of what is at stake and make for more accurate impact scores. As a result, this can help prioritize which controls get applied, in which order. With that said, asset criticality is not a requirement from ISO 27001, so ensure you're making sure it's fit for purpose for your organization, and remember that complexity is the enemy of security.

One more point to remember when it comes to your asset inventory is keeping it up to date. This inventory acts as a tool for reducing risk at your organization, and as a result, it must be continually updated to reflect any changes to assets, asset owners, values, criticality scores, or any other information that is required for a complete asset inventory at your organization.

We will delve deeper into how you might want to classify assets or information at your organization, and the roles for information assets, later on in this chapter, so let's move on to the next requirement for the ISMS documentation, the information classification policy.

Information classification policy

This policy sets out to define the requirements and processes for classifying information assets at your organization. It is a key document in the ISMS, and must (yet again) cater to your organization's needs, and be proportionate to the level of risk the organization faces.

We aim to classify information assets based on the organization's legal requirements, as well as the asset's value, criticality, and sensitivity. From general experience, I've found that most organizations will have around four classification options, to allow for flexibility without being overcomplex, but depending on the organization, there could be a wider (or narrower) range of classifications.

There are no specific requirements in ISO 27001 for what levels of classification are required, but you can align the classifications with other requirements or best practices, depending on your geographic location or industry.

A typical set of classification levels could include the following:

- Confidential
- Restricted
- Internal
- Public

In your policy, you might define that the asset owner is responsible for the appropriate classification of information, based on the definitions set out in this document.

Labeling based on classification

Your organization may commit to labeling physical and digital information assets based on their classification, and you are able to set out the requirements for doing so in the **information classification policy**.

In the later stages of this chapter (and throughout the book), I will go through various ways we can classify, handle, and protect information based on its classification and risk, but as you might have noticed, everything in your ISMS should reflect your organization and its requirements, there is no hard-and-fast way of going about this.

Acceptable use policy

Now that we have an inventory of the organization's assets, and understand how those assets are classified and labeled, we should specify how those assets should be used by members of your organization, contractors, and third parties in order to comply with the information security requirements defined in your ISMS. This acceptable use policy should be part of the information security training and education for members of your organization and should be accessible to all members who are subject to its requirements.

Some examples of what might be included in an **acceptable use policy** include any security rules applicable to employees, such as the following:

- General use and asset ownership policies
- Policies for email security
- Policies for company assets, such as laptops and mobile devices
- Policies for using third-party software
- Policies for returning assets upon termination of employment
- Policies for handling removable media, such as USB drives
- Policies for the secure disposal of media

This document should cater to your organization's needs and the nature of how members of the organization interact with information systems in their daily activities. Further, the policies should be proportionate to the level of risk the organization faces. In order to ensure the appropriate handling of information assets, the asset classifications should be taken into account, and appropriate policies for how to handle assets with each classification should be defined.

Risk management documentation

Now that we understand the organization's assets, we are able to define a policy for how we assess, treat, and report on the risks facing those assets. In order to ensure the effectiveness of your ISMS, this risk management documentation must be clear and absolutely *must cater to your organization's requirements and abilities*. If it does not, your ISMS will likely fail to protect your organization from the threats it faces. It is also crucial to remember that complexity is the enemy of security, and as a result, it is crucial to reduce any complexity in these documents and processes to the minimum amount.

I have already taken an opportunity in *Chapter 1, InfoSec and Risk Management*, to cover important risk management topics, but to skim over what we covered in the previous chapter, we want to perform risk assessments in order to understand the risks the organization faces, including determining the impact and likelihood of those risks. I assure you that I will expand on these topics throughout the book, including covering threats and control types in *Chapter 3*, *Designing Secure Information Systems*. For now, we're simply looking at documentation and processes.

In a general sense, the process that your organization will follow to manage risk can be reduced to the following steps:

- Risk identification
- Risk assessment (and prioritization)
- Risk treatment, including storing documented evidence
- Monitoring and reviewing the risks, including management reviews

To begin with, let's talk about risk identification, assessment, and treatment documentation.

Risk assessment methodology and risk treatment methodology

By creating **risk assessment methodology** and **risk treatment methodology** documents, you're able to define specifically how and when your organization performs a risk assessment, who is responsible for each step, how to calculate the risk level, and how we may reduce any unacceptable risk to an appropriate level.

You might decide that, in order to meet your organization's needs, a risk assessment is required annually, or upon significant change. That's a pretty typical choice I've seen quite often, but it's not necessarily the best choice for every organization. As we've said several times, it's all about catering each policy and procedure to your organization.

If we recall that one long sentence about risk from *Chapter 1, InfoSec and Risk Management*:

Information Security Risk is the potential for loss as measured through the combined impact and likelihood of a threat exploiting a vulnerability in one or more information system assets.

We remember that we cannot have a risk without an asset and a threat. In these documents, we can define how we identify and measure risk, and whether we perform risk assessments based on threat scenarios, or by each information asset associated with the organization. You have a list of assets at your organization, in your asset inventory, so you should leverage that inventory and reference it in your risk management documentation.

Furthermore, we can define the scale and definitions for likelihood and impact in this document. I covered an example of how you might do that in *Chapter 1, InfoSec and Risk Management,* and this is your chance to either use that example, leverage another framework, or create your own ad hoc system for assessing those levels.

You might also want to include the risk matrix, with the defined risk appetite *level* clearly displayed on it, as defined by top management.

When we're looking at risk treatment, you might remember that it's possible to do the following:

- Avoid the risk
- Reduce the risk through security controls
- Transfer the risk
- Accept the risk

You can describe these processes in your documentation, in order to ensure other members of your organization understand how and when they might choose each treatment. Keep in mind that you have a Statement of Applicability document that lists any applicable controls for your organization, and ensure that you reference it in your risk treatment methodology. You can also define how your organization accepts risks (and documents that acceptance) once the risk level has been reduced to an acceptable level.

By defining an effective risk management process for your organization, and applying the principles of continual improvement, you improve the effectiveness of your ISMS by increasing the visibility of the risks that your organization faces.

Risk assessment report

You need to be able to plan and prioritize risk treatments, as well as to communicate the results from risk assessments to relevant stakeholders. This implies that you should have various reports based on risk management activities, depending on the audience.

For example, in your regular updates to top management, you could provide a high-level overview of the risk assessment, as this audience is ultimately responsible for ensuring the ISMS is effectively implemented at their organization, but not necessarily interested in the specific details. You might include the most important assets and their inherent risk levels, the risk treatment applied (or proposed), and the resulting residual risk after the treatment.

In the more detailed report, it might make sense to provide a status for each of your organization's assets, as well as the Statement of Applicability controls and their status. Additionally, if a control is in the process of being implemented, you can report on the progress of that implementation.

Third-party security documentation

In *Chapter 1*, *InfoSec and Risk Management*, we discussed the importance of managing third-party risk at our organization, as a result of the growing reliance on suppliers to provide information systems, and cloud solutions for performing business activities.

In our third-party security policy, we want to define specifically how our organization handles the management of third-party risk, and describe how we can monitor, review, and audit a supplier's security.

This could include definitions and methodologies surrounding service-level agreements, vendor security assessments, and due care at your organization.

Furthermore, you should consider how you might catalog each asset used by your organization that is managed or controlled by a third party, and how you might structure your ISMS and information security strategy to cater to the shared responsibility model.

Incident management documentation

When we focus on incident management documentation, we want to be clear on what is required in a specific type of event. As the information security professional, you will need to work together with your organization's leadership to define the thresholds necessary to declare an information security event an incident, and the appropriate response.

The best way to document this is to create a high-level incident management policy, which defines the appropriate roles and responsibilities and includes references for any supporting documents, such as playbooks. The incident management policy should be the central document in any incident management activity, and the first reaction of any staff member in the event of an information security event should be to refer to it.

With the knowledge that this suite of documents is going to be required during incidents such as an outage, it's valuable to note that the incident management policy, and all other incident management and business continuity documents, should be stored in a way that is accessible regardless of whether availability to the information systems is affected. If ransomware spreads through your organization, you can't access the documents stored on your NAS anymore. Do you have printed copies? The same goes for physical copies: If your printed copies are now underwater, do you have a way to access those policies through cloud storage?

After the high-level incident management policy has been created, two areas in which a defined, easily-followed, structured approach to information security is incredibly valuable are **incident response plans (IRPs)** and **Business Continuity Management (BCM)**.

As we have defined how we're going to discover and classify our assets, and how we're going to perform risk assessments based on our asset criticality, we have achieved much of the progress required in order to structure these processes into our organization.

By understanding asset criticality, and the threats associated with the risks your business faces, you can ascertain and define the organizational requirements for business continuity and incident response.

We will delve deeper into various aspects of business continuity and incident response in *Chapter 7*, *Owning Security Operations*, and so I will do my best to avoid repeating myself too much here, and instead focus on the high-level requirements for the documentation and planning required to ensure your organization is prepared to respond to incidents.

Incident response and business continuity playbooks

Always remember that in the event of an information security incident, heart rates will have risen, palms will be sweaty, and there will be many people involved in the "what's next?" discussion.

Incident response playbooks should be simple, easy-to-follow for anyone, and accurate to the most recent changes in business processes or versions. This is much more difficult than it sounds. Think of it as a "picture book," where the first page is the first step, the second page is the second step, and so on. This is a highly effective method that increases the likelihood of a successful response.

You can have a playbook for the various threat scenarios, such as "Our server has been damaged," or "Ongoing DDoS attack on our website," and so on. All threats that are viable and require a response that is repeatable can be turned into playbook form, and it's not something that should be skipped over.

Furthermore, you can't just assume these playbooks are effective. We can't trust that the person who is accountable for executing the plan is going to be able to do so. It's important to run through the plan and see whether it will actually pan out the way it should, by holding incident management training, awareness sessions, and performing regular **tabletop exercises**. Tabletop exercises are simulations where members of the organization get together in a conference room and walk through realistic incidents at their organization, discussing their responsibilities and looking for gaps in the documented plans and procedures. The advantage of a tabletop exercise is that they're inexpensive and easy to facilitate, while still measuring the impact of various events. In order to make tabletop exercises more interesting, you as the information security professional should throw in curveballs occasionally. When people start following along and become bored with the activity, throw in a "The card is declined. Now what?". It keeps the members of the team on their toes and engaged, and simulates the messy real world where nothing goes to plan.

By documenting and optimizing the response process and applying findings from tabletop exercises and simulations, you are able to reduce the amount of time it would take to diagnose the cause of an incident, and the stress level experienced by the members of the response team.

IT management documentation

Due to the close-knit relationship between information technology and information security, your ISMS will likely contain several references to IT management documentation that may already exist at your organization or may need to be created with input from the relevant stakeholders.

This documentation could include the following:

- Operating procedures for IT management
- Secure system engineering policy
- Access control policy
- Bring your own device and mobile device policies
- Password policy
- Information disposal and destruction policy
- Maintenance and review plan
- Change management policy
- Backup policy
- Exercising and testing plan
- Information transfer policy
- Procedures for working in secure areas
- Clear desk and clear screen policies

It's crucial that these documents state the reality of your organization's approach, and are in line with the level of risk determined as acceptable.

Educating members of your organization

Although it's a great start to define the requirements of the ISMS in a set of documents and to use those documents to implement a systematic process for securing your organization's assets, users may feel arbitrarily restricted if a solution is rolled out that prevents them from working the way they always have, without any communication about the reasoning.

Even the most intelligent systems are able to be circumvented or destroyed, and your users might be compelled to find ways to get around the policies or controls in order to make their lives easier. Your organization has a requirement to train its users on the ISMS, and educate them on its impact on them in their role. It's important to help them see that you're trying to make their jobs easier by removing the burden of ambiguity from their daily working lives.

As we've said earlier, these policies need buy-in from top management, with repercussions for when the policies aren't followed. You are doing this to protect their organization, and so top management needs to care, and they need to be champions for your ISMS. Try to get the leadership team involved in the communications that you would normally send out yourself. By having a sponsor in the form of a CEO, MD, CTO, CFO, or similar, you will have notoriety among the staff that there is a method and a strategy, and it's not just you as a lone wolf "security dude" being paranoid and harping on at them about not installing browser add-ons, or randomly tricking them with phishing exercises. Getting the VIPs to do the heavy lifting is an especially important concept to embrace in order to increase effectiveness.

Furthermore, it's important to raise general information security awareness with all members of your organization. A good awareness initiative will enable users to identify various types of control circumvention, and potentially even prevent such activity by reporting a non-compliance scenario. That's an important aspect of ensuring the security of your organization is maintained and effective.

As with everything, optimization is key, so let's move on to evaluating your ISMS's effectiveness.

Evaluating the effectiveness of the ISMS

Evaluating the effectiveness of your ISMS, and whether it is fit for the purposes of your organization is crucial in the optimization of your organization's wider information security maturity. There should be a requirement set to review the effectiveness of the ISMS on a regular basis, such as annually, and to perform an internal audit as defined by your internal audit procedure, previously defined. Any **non-conformities** or **findings** should be subject to the corrective action procedure that we've also previously defined.

Additionally, by including findings from discussions and dialogs on a daily basis, and subjecting them to the same corrective action requirements, you enable continual improvement by defining the future changes required.

If your organization must comply with several standards and regulations from global bodies, it's important to consider the potential to leverage technological solutions to manage your requirements for your implementations. There are several tools available to align your ISMS and security controls with multiple regulations, and utilizing those can increase the coverage and effectiveness of your ISMS.

Improving the policy

The improvement phase of your ISMS is where we identify present gaps and plan for the next version of the policy. Always remember that this policy is acting as a structure for you to organize and plan your organization's information security program. It's just as important as anything else in the job, and there's a reason I started with this (extremely not-dry, but rather highly interesting and entertaining) topic.

Without fail, I'm going to mention the ideology of continual improvement with every step of this book. It's integral to keep up to date by regularly measuring risk, and applying controls to ensure the organization is not exposed to an unnecessary or unacceptable level of risk.

To summarize, in order to be able to implement an ISMS that ensures the appropriate identification, classification, and protection of the information assets of an organization, we must start with policies that reflect the organization's requirements from an organizational and IT perspective. It's critical to ensure the members of your organization are educated on the ISMS in order to avoid an increased level of unknown risk, such as **shadow IT**, which is a term to describe any IT systems that have been deployed without the IT department's oversight, and therefore circumventing the policies, processes, and controls that have been implemented to keep the organization secure.

Let's proceed toward the topic of identifying and classifying information assets next.

Identifying and classifying information assets

In your asset management policies, it's likely that you have defined the way your organization is going to identify and classify information assets based on the value, criticality, sensitivity, and legal obligations, but not specifically discussing *how* your organization might do that from an IT perspective.

While creating that policy and defining those motives is a huge leap forward in understanding the overall position your organization takes when it comes to their risk appetite and what is considered to be the most valuable information they have, remember that it's also just the "rule," and not the actual action of identifying and classifying those assets or protecting them for that matter.

In this section, I would like to detail the ways we can structure the rules, as well as the actual "doing" part of an effective identification and classification phase of an ISMS.

This includes the following topics:

- Structuring information asset classifications
- Determining the roles for assets
- Identifying and protecting information assets
- Retention policies

You're about to learn some key information security principles! Get ready!

Structuring information asset classifications

In your documentation, it is likely that you have defined classifications for your information assets, based on the value, criticality, sensitivity, and legal obligations determined by the organization.

You might have arrived at a hierarchy of classifications such as the following:

- Confidential
- Restricted
- Internal
- Public

How are you going to ensure the appropriate classification is applied to the relevant information assets, and how are you able to ensure the assets are protected and secured appropriately?

Generally, you will want to define specific roles for each of your assets. Each of these roles should have various responsibilities in the identification, protection, and disposal of information assets. The **Information Owner** could be responsible for ensuring the data is properly classified, and that the appropriate data protection has been applied, in line with regulatory and organizational requirements, for example.

Determining the roles for assets

If you want to be structured with the way your organization handles the identification and classification of its information assets, you must consider defining the roles listed in the following table for each asset. As with everything, the level of structure you create and implement should be acceptable for the organization you're working with.

Some roles to consider documenting for each asset are the following:

Information Owner	Responsible for data protection and ensuring the data is properly classified.
System Owner	Controls the software and hardware configurations for the processing and storage assets that the information lives in.
Information Custodian	Responsible for backups and restoration.
Security Administrator	Assigns the permissions for the network.
Users	Must comply with all policies and rules when sharing information with others.
User Manager	Responsible for overseeing the activity of all of the above-mentioned roles.

While it may not be feasible for a small company to have dedicated roles to accomplish each of these tasks, these roles (and several others) could exist in a large organization. Additionally, you could have several types of information owners, and each system could have a respective system owner, split, for example, by vendor.

With that said, if it's possible, use the KISS principle, which is a term recognized by the software engineering community as a design principle meaning *"Keep it simple, stupid"*. You want to ensure less work and maintenance is required to be successful in your ISMS. There's already a lot of work to be done without the extra complexity.

Methods of identifying and protecting information assets

Great! We have now defined our categories for classifying information assets. An additional consideration, and potentially the most important of all is the matter of *how* you can ensure the information is properly classified.

The manual identification, classification, and protection of information assets isn't a scalable or dependable solution. Loss, modification, or disclosure of these assets can carry significant financial and reputational risk. It makes very little sense to make non-InfoSec people responsible for appropriately identifying, classifying, and protecting the information assets of your organization during their day-to-day work.

Luckily, there exists **Data Loss Prevention (DLP)** or **Information Protection** technology to help the process of properly identifying and classifying our organizational information. We can leverage "built-in" models for discovering and classifying information, or create our own rules.

The "built-in" models can automatically identify specific types of information, no matter where it lives, or the format. This means we can configure our DLP or information protection solution to "find" credit card numbers sent in chat messages, or passport numbers detected in a `.jpeg` file on a corporate laptop's hard drive, as a couple of examples.

Beyond this, by creating our own model (or definition) of an information type, administrators are able to use regex and complex filters to locate any files that include information that is specific to the organization, such as keywords like Project Neptune, for example.

After locating this data, an automatic metadata "label" for the information classification is be applied, along with any appropriate restrictions for access, protections through encryption, as well as watermarks or notices that are visually apparent while users are accessing the information. The operating system or mail solution can even show the user a notification to explain why they aren't able to send a file outside of their organization, and link to any relevant training.

Through automation in the discovery, classification, and protection of our information assets for our organization's estate, we ensure that information is available to only those who need to access it, that the information has only been changed by those approved to, and that those changes are tracked and stored in a way that is auditable.

This type of solution is an absolute dream compared to even 5 years ago when DLP was still extremely antiquated and didn't offer anywhere near this level of automation. Now, if a user is creating a file on their laptop that discusses a sensitive topic, or contains PII, a label is automatically applied to that document, which, in turn, activates encryption and restricts access to the appropriate group of users. It should be the asset owner's responsibility to ensure the appropriate classification is automatically applied.

Administrators of this system should be able to see metrics of the users, locations, or programs and services that have each type of data associated with them. By using a technological solution to aid in this process, you can monitor the changes, detect new requirements, and measure the efficacy of your policy, based on the results from the monitoring over the entire lifespan of all of your organization's information assets.

Outside of the digital realm, in your information classification policy, you could (for example) require that the classification is indicated in the top-right corner of each physical document page and that it is also to be indicated on the front of the cover or envelope carrying such a document. As we've previously said, the policy should suit the organization, and as a result, there should be processes to follow the requirements set out from the policy, and repercussions for not complying with those requirements. Whether we can or cannot automate the process, ensuring the appropriate classification and labeling of information is usually the responsibility of the asset owner.

Luckily, we have defined a policy in plain language, to help those responsible for selecting, implementing, and configuring the systems that regulate these requirements. By setting those requirements in the information classification policy, we are able to select a technology that mitigates against the risks associated with data exfiltration, insider threats, retention policy breach, and so on, as required by your organization.

Retention policies

I'd like to briefly talk about retention policies, and considering them while developing and implementing your ISMS. There exist regulations in many jurisdictions that require that any sensitive data, when processed for any purpose, is not retained for any longer than the prescribed amount of time. What is the prescribed amount of time, you ask? Well, unfortunately, that depends on the jurisdiction, and type of data.

In our policies, it's important to consider the retention requirements for our organization, and ensure that a retention policy is defined. With these definitions, you can leverage technological solutions that exist to do the following:

- Ensure you aren't allowing users to dispose of information that must be retained due to a specific regulatory requirement.
- Ensure information that is able to be disposed of is disposed of in a timely and compliant manner.

Some of the requirements you should be looking to define are the following:

- How should we store and retain data in a way that makes it accessible whenever it is required, in a way that is easily searched and classified? This includes audio files, video files, and so on.
- What data must be retained? Business management, third-party dealings, partnerships, employment records. These are just the tip of the iceberg. Split the data into categories and consider a solution that allows tagging into these categories (along with sensitivity labeling).
- How long must the data be retained? Consider each classification or category, as they can differ.
- How do we apply ownership to information and assets? This can be done by policy only, but it would be better to automate it, as with almost anything, if you find a reliable solution.

With the same information protection technological solutions from the previous section, we are able to automate the retention policies defined by your organization and remove a file from access after 7 years of inactivity, for example.

Securing information assets

This section is all about implementing the appropriate information security controls for assets. I've been thinking about this section for a while, trying to understand how to tackle it best for you.

I know you probably have experience with choosing and implementing controls, and I don't want this section to end up being half of the entire book, just droning on and on about different types of controls or all of the great vendors out there who want to sell you a silver bullet to fix all of your issues. I'm going to go into many different controls and ideologies in the following chapters, anyway.

Instead, in this chapter, I want to make sure that we focus on heavy-hitting, effective ideologies to understand in order to select the appropriate controls, meaning that the asset is considered "secure enough" based on its criticality and classification.

There are different **classes** that split up the types of controls:

- **Administrative/Managerial Controls** are the policies and procedures I'm always talking about. They aren't as "cool" as a new software control, but they exist to give structure and guidance to individuals like you, and other members of your organization, ensuring nobody gets fined or causes a breach.

- **Physical Controls** limit the access to systems in a physical way; fences, CCTV, dogs... and everybody's favorite: fire sprinklers.
- **Technical/Logical Controls** are those that limit access on a hardware or software basis, such as encryption, fingerprint readers, authentication, or **Trusted Platform Modules** (**TPMs**). These don't limit access to the physical systems the way physical controls do, but rather access to the data or contents.
- **Operational Controls** are those that involve people conducting processes on a day-to-day level. Examples could include awareness training, asset classification, and reviewing log files.

There are so many specific controls, there's just no way we can go into each of them in this chapter. Beyond the Annex A controls from ISO 27001, further expansion on controls and the categories of controls can be found in the links on this page: `NIST SP 800-53 Rev 5 (https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final)`, including control mappings between the ISO 27001 standard, and *NIST SP 800-53*.

What I can cover are the types of controls that you'll be able to categorize and apply as mitigation against risk, depending on the threat and vertical:

- **Preventative Controls** exist to not allow an action to happen and include firewalls, fences, and access permissions.
- **Detective Controls** are only triggered during or after an event, such as video surveillance, or intrusion detection systems.
- **Deterrents** discourage threats from attempting to exploit a vulnerability, such as a "Guard Dog" sign, or dogs.
- **Corrective Controls** are able to take an action from one state to another. This is where fail open and fail closed controls are addressed.
- **Recovery Controls** get something back from a loss, such as the recovery of a hard drive.
- **Compensating Controls** are those that attempt to make up for the shortcomings of other controls, such as reviewing access logs regularly. This example is also a detective control, but compensating controls can be of various different types.

Generally, the order in which you would like to place your controls for adequate defense in depth is the following:

1. **Deter** actors from attempting to access something that they shouldn't be.
2. **Deny/Prevent Access** through a preventative control such as access permissions or authentication.

3. **Detect** the risk, making sure to log the detection, such as with endpoint protection software.
4. **Delay** the process of the risk from happening again, such as with a "too many attempts" function for a password entry.
5. **Correct** the situation by responding to the compromise, such as with an incident response plan.
6. **Recover** from the compromised state, such as a backup generator restoring availability to a server.

Furthermore, in the realm of continual improvement, we should monitor the value of each asset for any changes. The reason being that we may need to rethink our controls for protecting those assets if they become more or less valuable over time, or in certain major events at your organization.

Additionally, as a footnote, when we're looking at controls, we should also be thinking about recovery. What I mean is that we want to be able to recover from any adverse situations or changes to assets and their value. Just as examples, we're talking about backups, redundancy, restoration processes, and the like.

A concept to keep in mind, especially in the era of the cloud, SaaS, PaaS, IaaS, third-party solutions, and all other forms of "somebody else's computer" is to ensure that **Service-Level Agreements** (**SLAs**) are clearly defined, and have agreements for maximum allowable downtime, as well as penalties for failing to deliver on those agreements. This is an example of a compensating control.

As a consumer of third-party solutions, you'll want to fight for SLAs that reflect your risk appetite. Simultaneously, you'll also want to consider the idea that by chaining those assets together, you are creating a higher level of risk to availability. If just one of the services isn't online, and you can't perform a task, that's a loss of availability. If you're a vendor of cloud services, you need to consider your availability and what can be offered to your customers realistically, and what is required from a commercial perspective.

Data security

When we're looking at data security specifically, there are three key **states** of data that present unique security requirements: data in transit, data at rest, and data in use.

Data in transit

Data in transit or data in motion is data that is being transmitted either inside the network, or out of the network onto the web, either through physical cables or wireless connections, or from one application to another sitting on the same computer, or whatever other type of data transfer can occur, which I'll avoid going into before this sentence makes both of us lose our minds.

There's also the concept of **information in transit** being something like people speaking at a water cooler, or a USB drive being moved to another desk... but let's not be ridiculous. There's a reason I said "data" and not "information" in this case.

Data at rest

Without complicating things too much, let's just define **data at rest** as data that is being stored, not currently utilized. It begins to get a bit complicated when you consider databases as constantly growing and changing entities, with machine learning models utilizing the data in an evolving, living amoeba of logic. But as I've said, let's keep it simple.

Data in use

Data in use is data that is being processed by a system or user and is usually stored and referenced in a non-persistent memory form such as a CPU cache or RAM.

Encryption for data in various states

Encryption is one of the ultimate controls that we can discuss, for both data at rest, and data in transit. In order to effectively keep data secret, employing encryption is likely going to be a necessity. Actually, it must be used in order to satisfy the requirements set forth by standards such as PCI-DSS or ISO 27001.

Beyond the requirements, cryptographic controls are incredibly useful as a mitigation tactic. Make sure you're aware of the different types of encryption, and that doesn't only mean understanding "symmetric" versus "asymmetric" encryption, but also different encryption standards, and which mechanisms are used for securing data in transit, such as *TLS*, as opposed to securing data at rest, such as *AES-256*.

Hashing is a cryptographic function for integrity, which (for example) allows storing passwords in a way that ensures the input is the same as the compared data but without storing the plaintext of the input. Furthermore, hashing helps to ensure that data hasn't been changed, by using a checksum or message digest. Being up to speed with these topics should be at the top of your priorities, as you will use encryption more and more as a control moving forward.

Defense in depth

A major focus for you at your organization should be the consideration of **defense in depth**, a concept I touched on briefly in the last chapter, but wanted to re-iterate on here. We don't want a single point of failure to allow a complete loss of confidentiality, availability, or integrity. If we're protecting our assets adequately, it means that we can have a control fail, and still have the peace of mind that we have other controls in place that mitigate against the risk, (or at least some of it). It's difficult to achieve in practice, due to usability concerns, budget constraints, or a lack of options.

For an example of how defense in depth might look for a specific system, let's say your organization creates a web app allowing internal users to log in and create, read, update, or delete data in a database. How can you be sure that unauthorized users aren't able to read or update the data through a vulnerability? How can you ensure the app stays online and the data remains available?

Let's take a look at some of the controls we might put into place for the app:

- To prevent unauthorized access, and to enforce the appropriate levels of access based on the type of user, you might want to implement a form of authentication with varying levels of authorization. To protect against the breach of passwords, they should be stored as salted hashes, created with a secure function, with no way to reverse them back into plaintext.
- To prevent unauthorized changes to the web app's source code, the code base could be stored in a source control solution, such as GitHub or GitLab. Each developer could be given a unique login and be made to choose a strong password and utilize two-factor authentication.
- To prevent backdoors and common flaws from being implemented in the source code by a developer, you could run static code analysis on the code base. To check for vulnerabilities in any imported software used in the app, you could run a *dependency scanner* for the included imports also.
- To look for vulnerabilities in the app once it's running, you could run a vulnerability scanner on it, which pokes at it for common flaws to check for issues with versions, headers, and input sanitation, among other things.
- In order to detect whether there has been a breach, and to investigate and mitigate against the vulnerability being exploited again, logging, auditing, and monitoring policies and solutions could be implemented.
- Do you know whether your users are only going to be accessing the application in a certain context, such as from a specific IP address? You can implement firewall rules to whitelist (or only allow) that access, blocking everywhere else.

- You could also use a **Web Application Firewall** *(***WAF***)* to protect against common injections and cross-site-scripting attacks, among other things.
- How can you ensure that the application is available when it needs to be? You should probably look at load balancing and deploying across multiple locations.

So, that scenario gave various different examples of technical security controls, with administrative controls as policies for how they are configured. Is this a good example of defense in depth for the web app? Are you adequately protected?

It depends on the risk appetite! You might not need all of these controls, or they might not be enough, depending on the asset classifications, criticalities, and how much risk the organization is willing to accept. That's why in the first two chapters we've focused on that so heavily: Without understanding our risk level, there is no way we could understand whether we're spending too much (or not enough) time, effort, and money on mitigating against risk.

Monitoring for changes

You should be monitoring your organization's information *assets* for changes, and adapting to those changes, following the principle of continual improvement. One way you might handle this is through a regular risk assessment process, in which you perform risk assessments annually, or upon significant change, for example.

Another way would be to implement technological solutions to keep track of configuration changes, vulnerabilities and updates, or active threats. I cover configuration management and monitoring for security purposes in *Chapter 7*, *Owning Security Operations*, and as a result, I'm going to let that chapter do the heavy lifting on this topic.

Eventually, from your monitoring activities, you will discover it is time to retire an asset and dispose of it, which I'll move on to now.

Disposing of assets

We have several different physical and digital information assets in our control at our organization, all of which are likely to reach the end of their life cycle eventually.

How do we go about ensuring the secure disposal of those assets? It's likely we're not able to just throw sensitive financial documents or the CEO's laptop into the trash, making those information assets unprotected, waiting for a **dumpster-diving** malicious actor.

ISO 27001 mentions two controls on the topic of information disposal:

- *"Media shall be disposed of securely when no longer required, using formal procedures."* (A.8.3.2).
- *"All items of equipment containing storage media shall be verified to ensure that any sensitive data and licensed software has been removed or securely overwritten prior to disposal or re-use."* (A.11.2.7).

As a result, in order to comply with the standard, you should define your organization's requirements for the secure disposal of information assets based on classification, and you should provide specific processes for how your organization recommends its members abide by those requirements. Furthermore, you should keep a record of securely disposed of information assets.

Additionally, the disposal or reuse of storage media or devices should include a process to ensure the effective removal of any data remnants from the device.

Data remnants

Data remnants are a result of deleting data in a system but the system simply marking that space as "available for use," without actually erasing it. It is very simple to recover this data, with both commercial and open source solutions available to anybody with the most basic knowledge of installing software onto a computer.

This type of threat can be mitigated by structuring some sort of **defensible destruction** in your organization, which is the controlled, compliant, legal method of destroying data through one of the following methods:

- **Overwriting**, which causes the original data to be replaced with new or random data. The US standard is a minimum of seven times.
- **Degaussing**, or removing the magnetic imprint on disk drives, which wipes the drives. This could potentially be useful for old-school HDD drives.
- **Encryption** is not only a way to protect data at rest and data in transit, but also a potential way to make data unrecoverable by using a secret key that isn't stored, as in "throwing away the key." Cryptoshredding is a term used for when you encrypt your data and then encrypt the recovery key so it can never be used again.
- And then, of course, the **physical destruction** of a disk, usually by putting it into a woodchipper-like device, sort of like Steve Buscemi in Fargo. Other examples could include incineration, shredding, disintegrating, and pulverizing.

When we talk about **data disposal**, we don't always mean defensible destruction, however. Sometimes, when we dispose of data, we archive it into long-term storage. We won't use the data, but if it's required in the future, we have it securely stored, usually in a way that is the most cost-effective for the use case. This method is valuable for data that could be required for legal proceedings, for example. The retention periods and requirements for your organization must be understood and considered in order to make an appropriate decision on how to store this data and remain compliant with local laws and regulations.

Summary

In this chapter, we focused on the various topics surrounding protecting the security of assets. We had our hair blown back with the enthralling topic of implementing an ISMS, which covered everything from the responsibilities of top management to developing an ISMS, educating members of your organization, evaluating the policy's effectiveness, and improving the policy for the next iteration.

Then we moved on to identifying and classifying information assets – from structuring the information asset classifications to determining the roles for assets, methods for identifying and protecting information assets, and retention policies.

We moved on to a high-level overview of securing information assets, data security, encryption, defense-in-depth, and monitoring for changes, before moving on to the final topic of disposing of assets and data remnants.

And, just like that, we are through this chapter. I am not even sure how you managed to take in all of that information, but somehow you did, and you're all the better for having done it. Now let's move on to the next chapter, where we'll talk about designing secure information systems.

Section 2: Closing the Gap: How to Protect the Organization

Here, we'll explore avenues we can take in implementing security controls to achieve our organization's information security goals.

This section contains the following chapters:

- *Chapter 3*, *Designing Secure Information Systems*
- *Chapter 4*, *Designing and Protecting Network Security*
- *Chapter 5*, *Controlling Access and Managing Identity*

3

Designing Secure Information Systems

In the previous chapter, we talked about how to protect assets with controls that have been applied based on risk, but there is so much more that can be done. *What more can be done?*, you ask, bursting from sheer excitement. *Well…*, I reply, and softly smile to myself: *What if we just designed the systems with security in mind from the beginning?* Your face lights up, and the stars create tracers as the world whirls around us.

No—but seriously, we should be designing our systems with security in mind from the beginning. That's what this chapter is about. By the way, I don't mean *design* the same way a man (who is still standing on a Segway scooter inside the elevator of your building with you) says that he *designed it himself*, his sunglasses still on even though you're indoors, thinking he's extremely cool. Yes, it's happened to me. I mean it in a way where we plan and **threat-model** the implementation and ensure that it is fit for purpose.

Luckily, as you may already know from studying for your fancy cybersecurity certifications, that there are certain **secure design principles** that exist, allowing for us to utilize collective knowledge in order to improve the overall security posture of the system, without causing additional overhead. By the end of this chapter, you should be able to utilize these principles and select the appropriate controls to effectively manage secure information systems.

Additionally, there are **architectural vulnerabilities** that can be avoided or mitigated against with controls, and so we will need to go over the best methods for that as well. My goal is for you to be comfortable in determining a system's security requirements in order to apply the appropriate controls. Beyond that, you will be aware of the available security capabilities for the particular information systems you are working with.

Furthermore, there is the practice of selecting the appropriate controls. We briefly touched on this topic in *Chapter 2*, *Protecting Security of Assets*, where I talked about **defense in depth** for various aspects of developing and running a web app, but I'll expand and try to help you be more comfortable with selecting and implementing controls to manage and mitigate against vulnerabilities in web-based systems, mobile systems, and embedded devices.

So, to summarize, in this chapter, we are looking at the following topics:

- Understanding the risks your organization faces
- Best practices in assessing and mitigating vulnerabilities
- Best practices in designing secure information systems

All right—now that we've got our bulleted list in the introduction, I ask: *Are we ready to start?* Yes, we are ready to start. Let's begin where we should always begin, by trying to understand the risks.

Understanding the risks your organization faces

In this section, we will focus our efforts on getting our heads around key concepts in the **threats**, **vulnerabilities**, and **methods** of exploiting **information systems**. This includes the types of systems we'll be dealing with, the threats that **information security professionals** are hired to protect those systems against, and the ways those threats exploit vulnerabilities in those systems. Only after we understand these key points can we move on to the protection section (that sounds like a *Schoolhouse Rock!* song, but do not worry—I'll keep this largely nonmusical).

Something I would like to stress is that when we are designing a new system—whether this is web-based, mobile, embedded, or what have you—there are processes in place that ensure the security of our systems by design, and then there are mitigation controls that provide defense-in-depth in the event of the failure of those processes.

First things first, let's talk about some key concepts around *threats* that exist. By understanding the threats that we are facing, we can start to apply that knowledge to understand the key concepts in the **security models**. So, *"Let's Talk About Threats, Baby"*.

Hey—I said *largely nonmusical*, not *entirely nonmusical*.

Threats, threat actors, and motivations

The **threats** against your organization can be varied, and understanding these is an extremely interesting topic to study. Most importantly, they are crucial to understand in order to fully and effectively protect your organization. Remember, we can't have a risk without a **threat** and **threat actor**, along with an asset and a vulnerability.

As the world is changing to be more and more based in the digital realm, many threats to your organization will surface in the form of **cybercrimes**. Cybercrimes are no different from non-cyber crimes, in that there are various motivations that then lead to acts of crime (providing the threat actor has the capacity to have a motivation). The reason I use that stipulation is because there are threats that are environmental. No storm has ever had a motivation to destroy a server—that I'm fairly certain of.

Now, these human motivations can vary from instance to instance and sometimes require some creative thinking and brainstorming to dream up. What I would like to do in this section is to stimulate that part of your brain, and get you thinking like a malicious actor. I want you to start to imagine how you would exploit your organization if you were to target it. What can you imagine people would want from it? What could people stand to gain, and who are those people? Which groups don't like your organization, and why is that? Maybe you work at the most benevolent organization that has ever existed, and nobody in their right mind would ever have any motivation to act against it. In that case, you still need to consider the insider risks of somebody accidently clicking a link or deleting a database. Not malicious, but still a threat actor, and still a threat that already has a certain level of access to your estate. **Insider risks** should always be considered when threat-modeling your organization, malicious and non-malicious.

As you might imagine, I can't really go into every possible motivation we might possibly see, but with that said, I can also throw some ideas your way.

One "group" of motivations your organization may face comprises those that are not financial or political in nature. They could include, for example, *thrill-seeking*, *joy*, *data hoarding*, or *mastery*. These motivations are essentially based on the idea that the malicious actor enjoys the act of hacking, or the "paydirt" of the hack itself. Some individuals are more interested in if they can get in, or seeing what's inside, rather than how much they can earn by doing so. Maybe there are images or videos on a device or server that an attacker has an interest in seeing. Maybe your organization has a major security spokesperson with a million followers on Twitter who claims your product, a hardware Bitcoin wallet, is "unhackable". Researchers around the world are extremely interested in hacking things just to hack them, and especially if a major talking head makes a claim such as that. They might find your product doesn't wipe the **random-access memory** (**RAM**) clear after being rooted, meaning the secret phrase and salt are left in the RAM for extraction, and they might post it on Twitter. It brings them joy; it hones their craft; it helps them show their mastery while proving that talking head wrong.

Additionally, there are thrill-seekers who just love the chase. It might seem strange, but a survey performed by *Thycotic Software* at *BlackHat USA* in 2014 found that 51% of the hackers surveyed said they were mostly motivated by fun or thrill-seeking. They could focus on your organization, just for the thrill of it, and the best you can do is to adequately protect your most important assets to the level of risk they pose.

Another group of motivations worth discussing centers on politics and **cyberwarfare**, such as *hacktivism*, *propaganda*, and *sabotage*. In an era of online political discourse going mainstream, with major issues in current events, including a global pandemic, US election, and the UK exiting the EU (and those are just in the final couple months of 2020), we're facing a skyrocketing level of this type of motivation. *Hacktivism* and *sabotage* can be motivations for individuals or groups, all the way up to the **state-sponsored actor** level gaining access to (or destroying) accounts, systems, and assets they are at odds with. We've also got *propaganda* as a motivation, which can yet again be distributed by **sole actors** or groups, all the way to the largest imaginable force, which aims to gain control of legitimate channels to either filter or promote certain messages.

Going further into the political sphere, looking at attacks at critical infrastructure by outside forces, we can interpret that the motivation for *destruction*, *control*, or *intimidation* exists. I am always reminded of the time the Ukrainian government, in the midst of a significant cyberattack, tweeted a "this is fine" meme in response, as seen here:

`https://twitter.com/Ukraine/status/879706437169147906`

This is a screenshot of the tweet from Ukraine's government:

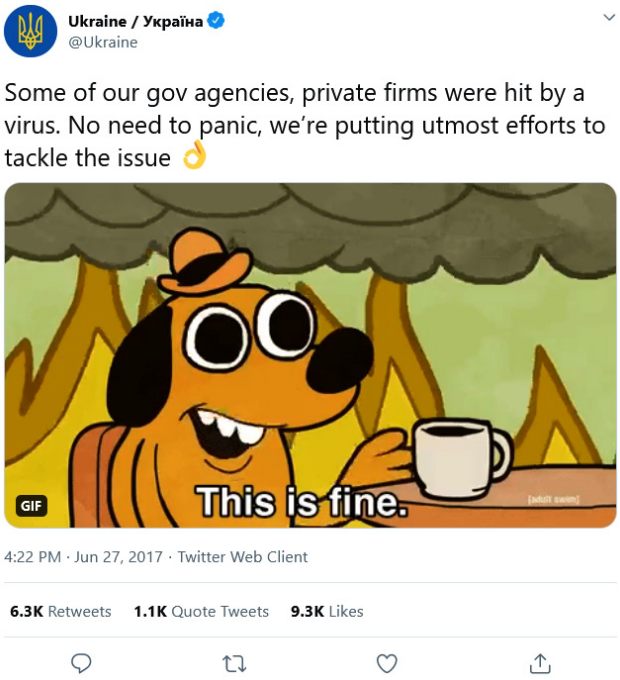


Figure 3.1 – "This Is Fine" tweet from Ukraine's government

Why were several governmental agencies and private firms in Ukraine targeted? I can't be as confident as others have been to say *who* was targeting them, but the nature of the attacks implies the attacks fit into the destruction, control, or intimidation motivation group, or the hacktivism, propaganda, and sabotage motivation group, and was likely an act of cyberwar.

Then, finally, we can look toward a motivation rooted in *monetary gain*. This could range from an individual finding a way to pilfer funds from a cryptocurrency exchange, or a mercenary hacking group spending time and effort in a heist situation, like an episode of *Mission: Impossible*. Furthermore, you have **corporate espionage**, such as what *The Ketchup Company* was starting to threat-model in *Chapter 1*, *InfoSec and Risk Management*, where a competitor tries to gain access to precious **intellectual property** your organization possesses.

There are other competition-based motivations that could be considered, such as attempting to gain research information from nonprofits, for example. Honestly, the list could just keep going, but as I previously said, that's not the point of this section. My goal was to get your mind reeling and to get you to focus on who might target your organization, and why they might do so. In that, I can almost feel the gears in your head turning, so I think we're done here.

Something you might hear about a lot lately is **advanced persistent threats**, or **APTs**. According to *NIST SP 800-39*, an APT is a threat actor that is both highly capable and highly motivated, with significant amounts of resources. This threat gains and maintains access within an organization's estate in order to exfiltrate information or manipulate operations for an extended period of time. The **National Institute of Standards and Technology** (**NIST**) says that they adapt to changes that are meant to mitigate against it. My point of view on APTs is pretty basic, in comparison to some of the things I've read on the topic. Most organizations are currently having difficulty protecting their assets against a sole 12-year-old who has watched some YouTube videos with titles along the lines of "*how to hack*", and even in those videos, the instructor will teach the viewer how to do the following:

- Gain access.
- Elevate privileges.
- Maintain access
- Exfiltrate information or manipulate operations.
- Remain undetected.

The only difference here from the YouTube *script-kiddie* and the APT is expertise, and funding. All of the other words NIST have used are just the basic steps any malicious outsider would take. I prefer to look at threats in the way we did in *Chapter 1*, *InfoSec and Risk Management*. In the case of APTs, we have malicious outsiders that are either **group actors** or **state-sponsored actors**, which means we need to consider that their ability to gain access and remain undetected is more sophisticated. Depending on your organization and its assets, this means that you might have a different level of risk that you will need to mitigate.

One final remark that I would like to make is this: *Please* keep your *risk register* up to date with the appropriate threats and motivations! This isn't only for fun; this is to appropriately protect your organization.

Vulnerabilities

Looking over what we've covered in *Chapter 1*, *InfoSec and Risk Management*, *Chapter 2*, *Protecting the Security of Assets*, and so far in this chapter, we can say that I keep harping on about how "*You can't have a risk without an asset, a threat and threat actor, and a vulnerability*". What *is* a vulnerability, though? Have we gone into that? Have I taken for granted that you're a walking encyclopedia of InfoSec, a fountain of knowledge on every topic?

The thing is that the list grows and grows when it comes to "types" of vulnerabilities if you get down to a granular level. *Meltdown*, *Thunderclap*, and *Spoiler* are some examples of recently discovered hardware security vulnerabilities, but explaining how those work will eat up my page count on this chapter, and there is simply far too much to cover already. Instead, let's take a couple steps up the ladder to view vulnerabilities from a higher level. *ISO/IEC 27005* guides us to classify vulnerabilities according to the asset class they are related to. This asset class could include *hardware, software, network, personnel, physical, and organizational* vulnerabilities, or maybe you have a different vision! Don't let me or the **International Organization for Standardization** (**ISO**) pigeonhole you with their strict Swiss guidelines—go on and fly! With that said, it *is* easier to just use their strict Swiss guidelines.

You might be totally drawing a blank. "*What kind of vulnerabilities exist for each of these, Joseph?*", you might be asking. The problem is, it's such a tremendous topic, and part of the reason I am trying to encourage you to brainstorm and consider the possibilities yourself is due to the sheer volume of known vulnerabilities in the wild. If I go down the road of listing off vulnerabilities and explaining them, this book will end up being 1,500 pages long and less entertaining that it already is, which I would never want to inflict on anybody, especially somebody who is taking their time to read my ramblings.

With that said, I *suppose* I could delve into a few examples of vulnerabilities and how to mitigate them. "*Vulnerabilities*" *is* the title of this section, after all.

If we're looking at *hardware* assets, as an example, we can try to consider the various vulnerabilities that may exist, including the following:

Physical threats such as humidity or dust; wear and tear causing failure; or over-heating all exist. A lack of controls such as ventilation, humidity control, cooling, lubrication, or quality control are all vulnerabilities that could all be exploited by a threat actor to cause a loss of availability by destroying, shutting down, or forcing the powering off of hardware.

Remember—threat actors don't need to be malicious humans; they can be environmental, such as "friction". The effective implementation of the "lubrication" or "cooling" controls (combined with testing after implementation to ensure the control is effective) would lead to the mitigation of these vulnerabilities.

When it comes to threats attempting to access storage—either hard drives or memory—you're faced with a few options. The lack of *encryption* when it comes to hard-drive storage could lead to a threat actor managing to access information by gaining physical access to a hard drive. By encrypting the hard drive, you're ensuring a level of defense-in-depth by ensuring that even if physical access were obtained, further exploitation would be required in order to gain access.

By not protecting a system's memory from being both read and written to, you're looking at a risk of loss of confidentiality, integrity, or availability. It's a highly utilized attack method, and as a result there are memory-protection controls in many modern architectures and operating systems that are occasionally broken and need to be mitigated through changes and updates to either the *kernel* or the *operating system*.

Threats may attempt to read electrical signals for reverse-engineering or eavesdropping purposes. This could lead to a loss of confidentiality or integrity if they manage to gain access to protected resources through doing so. When we talk about vulnerabilities leading to the eavesdropping of electrical signals, we're starting to get pretty intense, and we should consider the likelihood of this type of attack. Which level of threat actors is likely to perform these types of attacks? The likelihood may be higher with more sophisticated threat actors, such as state-sponsored actors. With that said, there are mitigations against leaking emanations, such as shielding magnetic fields, filtering conducted signals, and masking space-radiated signals.

Sometimes, vulnerabilities exist due to the basic function of the hardware and must be mitigate, for example, by preventing the possibility of physical access to the hardware through a **security perimeter**, although that is admittedly lacking defense-in-depth controls.

Software can have vulnerabilities exploited by a threat actor that does the following:

- Finds a way to access more information than they should have access to, either through insecure code or a design flaw
- Is able to use your software as a delivery mechanism for their malware, such as the *2020 SolarWinds hack*
- Exploits the lack of an audit trail in your systems

A lack of input sanitation can lead to information disclosure or your software being used to deliver malware through an **injection attack**. Mitigating this vulnerability is generally done through both static and dynamic code analysis, as well as automatically fuzzing your software with input that might lead to an injection, and also *penetration testing* of your software on a regular basis and upon significant change.

The lack of logging in your estate could mean that a threat actor is able to maintain access, make changes, and exfiltrate data from your organization without being detected. Mitigating this with logging is mostly a *detective control*, but with the appropriate **Security Information and Event Management** (**SIEM**) configuration, and capable analysts in your **Security Operations Center** (**SOC**), you may be able to detect an ongoing attack and react to it in real time. Even better is adding automated responses into the mix, which isn't as far-fetched and difficult as it sounds, but does require quite a lot of understanding of your existing "normal" operation.

Networks can have vulnerabilities exploited by a threat actor that does the following:

- Eavesdrops on insufficiently protected communication
- Gains access through an insecure network architecture
- Causes a loss of availability through a **Single Point of Failure** (**SPF**)

From a high-level view, *network security* is focused on ensuring the availability of network-accessible resources, along with the prevention of non-approved entities from access, and monitoring all network activity to ensure any infraction is dealt with appropriately. It's a deep topic that will require quite a bit more investigation than we currently have here, so I'm going to continue on with that topic in *Chapter 4*, *Designing and Protecting Network Security*.

Physical sites are subject to vulnerabilities exploited by a threat actor that does the following:

- Interrupts access, causing a loss of availability for your staff or systems
- Circumvents access control and gains unauthorized access to your physical estate

While you are looking at physical sites and contemplating site security, aside from considering access control, turnstiles, mantraps, and biometrics, you should also consider the effects of faults in devices or controls. Also, what occurs in the event of a power blackout or power sag (as well as brownouts and spikes)? Consider the complexity that excessive noise may bring, and ensure you think about the requirements around heating, ventilation, humidity control, and cooling. As we mentioned in the first chapter, servers and water sources don't mix, but at the same time, you might need adequate fire protection.

According to *NIST SP 800-30*, *organizational vulnerabilities* exist in the following forms:

- Governance structures
- Business processes
- Enterprise architecture
- Information security architecture
- Facilities
- Equipment
- Software development life-cycle processes
- Supply chain activities
- External service providers

Essentially, threats may see a logical flaw in the way your organization runs things. They might notice that you don't properly *background check* your job applicants, or that you use external service providers but don't perform *due diligence* on them, and so the threat actors manage to exploit that to become embedded as an insider threat themselves.

Furthermore, and by far the most likely threat in this section, threats may see a vulnerability in the lack of *security awareness* at your organization and, through manipulating or tricking personnel, manage to either gain access to your estate or have the personnel change or delete information for them, or disclose it to them.

I'm sorry to mention it again, but *Mission: Impossible* is just way too relevant to not bring up twice: all of these generic examples I've given are fine, but I want to encourage you to think outside the box. Pretend you're a member of Ethan Hunt's *Impossible Missions Force*, and consider what you would look to exploit in order to gain access to your own estate, and how you may read or write information to achieve your ultimate goal of saving the world. If you haven't seen the films, I have no sympathy for you not getting this reference whatsoever; they're all brilliant (apart from the second one).

To wrap up this section, I would like to say that it's important to consider every aspect of your business, and speak with relevant people who know the recorded processes and how real life differs from the documentation. A key question for yourself, moving forward, is: "*Do personnel do what they say, and say what they do?*". Policies and documented processes are only documentation, and if that documentation only exists to please auditors and security staff, then you are facing a huge gap between your *perceived security* and *actual security*.

The brilliant thing is, if you handle things right and show yourself as approachable and able to make a positive change, you will encourage an open, transparent, and honest working environment where people aim to involve you to ensure they have an easy way to follow the rules, and let you know when it's not possible to do so. That is an information security professional's dream scenario right there.

System exploitation methods

I'd like to look at some of the methods threat actors will use to exploit vulnerabilities in your estate. Yet again, there is a long list if we go deep and start picking out specific **keyloggers**, for example, so instead I'll go into the "category" for each exploitation method, and talk about each of them a little bit.

First, since I just mentioned keyloggers, I'd like to look at types of **malware**.

Malware is software used for nefarious purposes. It can include a wealth of different types of software, many of which have quite a lot to learn about individually, so I will split the malware section up. Something to keep in mind is that malware isn't always taking advantage of security vulnerabilities to be secretly installed on your endpoints or in your estate, which is what many definitions I've read have stated. In the late 1990s or early 2000s, I installed *BonzaiBuddy* by choice, which was a little purple gorilla in the bottom right of your screen that would sing you a song or search for things on the web. It was a marginally less lame version of Microsoft Clippy. It was later labeled by *Consumer Reports Web Watch* as **spyware**. Looking back, *BonzaiBuddy* would set your home page to `bonzi.com` without permission, which admittedly isn't very cool, and also tracked various information about the user, using that information to serve advertisements. All I can really say about that is: they were innovators! Now, almost every software does that, and it's seen as normal.

Breaking down the malware category into subcategories, we can start to understand how they work and what a threat actor might use them for.

Viruses are an interesting concept in terms of cybersecurity. What is a virus? Can you name one? Looking through *Wikipedia*, you might see that the page for *Computer virus* claims that a virus self-replicates. You might think: "*I thought that was a worm*." On the *Wikipedia* page for *Malware*, it's stated that viruses are usually "*hidden within another seemingly innocuous program*", to which you might ask: "*Isn't that a Trojan Horse?*"

What on earth is a virus, then? It seems like the community can't fully agree, but *Malwarebytes Labs* has attempted to define a computer virus once and for all as *"malware attached to another program (such as a document), which can replicate and spread after an initial execution on a target system where human interaction is required. Many viruses are harmful and can destroy data, slow down system resources, and log keystrokes."*

Now, *that* makes sense. A virus requires human interaction and a host program, and attaches itself to other files (or replaces files fully with copies of itself). A *Trojan Horse* (or *Trojan*) can be a virus; *ransomware* can be a virus; but *worms* aren't viruses.

Worms aren't that different from viruses, but just as I projected that you thought to yourself while reading the *Wikipedia* article for *Computer virus*, what differentiates a worm from a virus is that a worm does not require human interaction to spread across a system, network, estate, and so on. A worm *self-replicates*.

The name *Trojan* is a reference to *The Aeneid*, by Virgil. In the poem, Odysseus the Greek thinks of a way to get his troops inside Troy, a highly secure walled city of the enemy. He says: "*My dudes, let's construct a gigantic horse sculpture out of wood, hide inside it, and leave it at their gate as an indication of surrender.*" At no point do any of the other Greeks call him out on it, which makes me feel like they were just afraid of what his reaction might be. What I'm saying is: I think Odysseus was a bit of a prima donna. Anyway, the Trojans find the sculpture at their gate and wheel it into their city, figuring "*Yeah, we could use a gigantic wooden horse, and this doesn't seem suspicious in any way whatsoever.*" It might shock you to hear, but that night, Odysseus's men jumped out from inside the horse, overthrowing the entire city.

I'm sure you're glad that I went into the entire classical backstory for the term *Trojan*, because now you can very clearly understand that Trojans are just a way of getting malware into the walled city that is your organization's estate. When folks get the torrented, cracked version of the latest PC game, and the README tells them to turn off their antivirus when running the *keygen*, because it sets off *false positives*, and they do it? They might have just got Trojaned. The problem with knowing if the `keygen.exe` is an actual Trojan or just a false positive is difficult, because both use obfuscation to hide their algorithm and both will probably make some changes to your system, which automatically sets off antivirus software.

Very similar in nature to the term *Trojan* is the term *backdoor*. Backdoors are deliberate (likely undocumented) points of entry into systems. These backdoors may allow for the delivery of malware, or other malicious activity by threat actors. The thing about backdoors is that they could be implemented into software or hardware by the manufacturer or any of their partners or suppliers that have had a part of the design and development stages. Potentially, intelligence agencies could request that manufacturers implement backdoors to allow for easy investigations on their part. Additionally, these backdoors can be created by other types of malware, and delivered via a Trojan. Are you still with me? If this section is pretty heavy, it's worth it—keep holding on.

Another section of the overarching malware group is **adware**. Adware is often packaged with other software you install, but don't mistake it for a Trojan. It's not usually hiding; it could potentially show you a dialog in the installation process, which you haphazardly skip through like you have absolutely no time to do any reading whatsoever.

Here's an example of the **Search App by Ask** being bundled with Java:

Figure 3.2 – Search App by Ask adware installation dialog

It seems like it's just another innocuous page that you accept, because you want to get it over with and install Java already, but it's actually a **browser hijacker** that changes your default search engine to `Ask.com`. The creators make money by serving you ads and gathering marketing data.

OK—fine; your search engine has been changed in your browser—big deal. "*Why are we even worried about malware*?", people might ask. I would say the next few might change their tune.

Ransomware is a type of malware that prevents access to systems or files. It's an availability attack, usually through cryptography, and often acts as a worm, spreading through every compatible device on your network. Organizations affected by ransomware usually receive the **payload** in the form of a malicious link or an email with a malicious attachment.

Once infected, a dialog will be shown on the affected computers that attempts to shake down the victims. "*We'll give you the decryption key if you send X Bitcoin to this address…*", or similar.

Here's a screenshot of the Cryptolocker Bitcoin payment dialog:

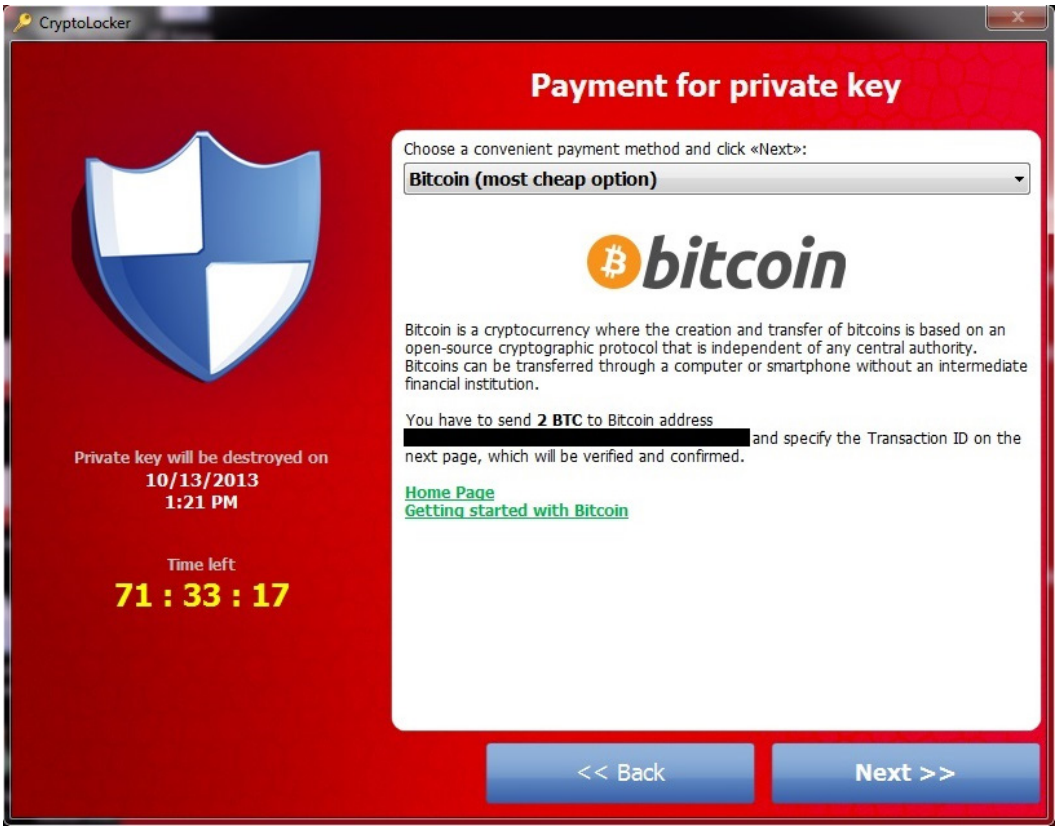


Figure 3.3 – CryptoLocker ransomware with Bitcoin payment option

The **Federal Bureau of Investigation** (**FBI**) advises people to never pay a ransom, but as with everything in InfoSec, I would say that *it depends on the risk*. If you stand to lose more than the equivalent of the ransom value by not paying the ransom, then you might be compelled to pay it. The reason the FBI has taken that position is because it creates a market, and you can't be sure that the assailant will actually give you a decryption key.

With that said, if word got out to other victims that even after payment no decryption occurs, it would ruin the revenue model of the campaign because other victims wouldn't be as likely to send the cryptocurrency to the perpetrator's addresses. As a result, it's in their best interest to decrypt your files when their victims pay, and we all know that business is booming. The *Cyber Threat Alliance* showed evidence in 2015 that the perpetrators behind a *CryptoWall 3.0* ransomware campaign had "*hundreds of thousands of dollars*" worth of Bitcoin received into their addresses, and things have only become more serious in the years since then. A cybercrime research organization called *Cybersecurity Ventures* has predicted that a ransomware attack is expected to affect a business every 11 seconds by the end of 2021, and that the annual damage will total 20 billion dollars.

Ransomware is scary, but **spyware** is creepy. Spyware is essentially a way for malicious actors to monitor your computer usage, harvest your passwords or payment information, and track what you visit online or the files you download. It can lead to identity theft or blackmail, and could have its own book dedicated to it.

Generally, spyware will end up embedding itself to your operating system and might arrive as a Trojan, or even as something you agreed to install, similar to what we saw with adware. There's generally not going to be an "uninstall" program for spyware, and you likely won't know that it exists on your PC. It may be a **keylogger**, which is a system-monitoring solution that can log keystrokes, system credentials, and pretty much any other computer activity. Keyloggers can grab screenshots and transmit the information to a remote server through a covert channel.

These types of malware generally *exploit* **vulnerabilities** in your device's software or hardware in order to get **elevated privileges**. New vulnerabilities for software and operating systems are found daily, and researchers might reach out to the creators of the software or hardware to *responsibly disclose* the vulnerability to them, in order to give them time to work on fixing the flaw and releasing a **patch** for the users currently affected by the flaw. Sometimes, the creator won't do anything or will legally threaten the researcher for that disclosure, and as a result the researcher may decide to release the information publicly, in order to try to force the hand of the creators. By drawing attention to an active vulnerability, and even providing a **Proof of Concept** (**POC**) for an **exploit**, it increases the likelihood that the vulnerability will be exploited and could result in bad press for the creators or a negative consumer response from the users.

The other side of that story is that a researcher might find a new vulnerability or explore an existing vulnerability, and craft an exploit that they then sell to malicious actors, maybe as a suite of software called an **exploit kit**, which allows nontechnical malicious actors to perform their attack and spread malware or gain access.

You may have heard of **rootkits**—or, similarly, **bootkits**—which are fancy names for malware that either escalates the privilege of an unauthorized user (potentially all the way up to that of an *admin* (or "*root*") account) or hides the activities of another malware from detection, embedding themselves into the operating system of your device. These can help a malicious actor gain full access to a computer, allowing for attacks on confidentiality, integrity, and availability, or to utilize the compromised computer to perform attacks on other networks, computers, systems, and so on.

A computer that is being used this way is called a **zombie**, and oftentimes you may find that this zombie is just one part of a much larger **botnet**, or collection of internet-connected devices that perform a synchronized attack. These botnets are especially useful in **DDoS (Distributed Denial of Service)** attacks, where the DoS sources are coming from a wealth of normally harmless **Internet Protocol** (**IP**) addresses with no pattern. As a result of their distributed nature, these attacks are difficult to create firewall rules for.

Logic bombs are a delivery mechanism that sets off malware activity once a set of conditions are met. Imagine, in a typical example, a systems administrator sets a *CronJob* that executes daily and asks them to enter a passphrase. Imagine the CronJob is set to encrypt all of the files in the network in the event that they are not performing their daily passphrase entry for 30 consecutive days, implying that they have been removed from the company. There are many other types of logic bombs, such as something as simple as: "*If this running process is closed, do this.*"

Something you've definitely read about in the news are **zero days**. These are simply vulnerabilities that were completely unknown to all aside from the malicious actor who is now exploiting that vulnerability. The term "*zero day*" is related to the fact that the vulnerability is being exploited with zero days for the developer to react and create a patch.

So, in theory, we could say the following:

"There's this new virus called "*TokTik*" that is very interesting. It is bundled with the latest version of `EssentialProgram` as a **Trojan**, initially just as **browser-hijacking adware**, but then as **malware** in the form of a **keylogger spyware rootkit** that spreads and executes a **privilege escalation exploit** to all the other devices on your network like a **worm**, adding each device as a **zombie** to their **botnet**, and performing **DDoS** attacks on businesses. If the rogue process is closed, another encrypts the hard drive and shows a dialog explaining the **ransomware** decryption process. **Patches** for operating systems are currently being developed, but many of the exploits were unknown and the **vulnerabilities** were never discovered until now, also known as **zero days**.

However, if you *were* to say that, people might not talk to you anymore because that would be incredibly odd to hear.

Beyond exploitation that occurs on your devices through viruses such as the previously mentioned malware examples, there exist other types of exploitation, such as the exploitation of organizational vulnerabilities (such as a lack of security consciousness), or exploiting software and web application vulnerabilities.

What would this section be without discussing **phishing**? As a reminder, phishing is a method of exploitation through electronic communication whereby a malicious actor is disguised as a trusted party and *socially engineers* a user to help them in their goal by clicking a link, opening an attachment, or handing over information and secrets. It's the chosen exploitation mechanism for many threat actors because of the fact it doesn't require a technical vulnerability to be successful, but rather an organizational vulnerability.

By emailing somebody in the company who might not be as *security-conscious* as a member of the InfoSec team, creating a little bit of *drama and urgency*, and guiding them toward executing a payload, the malicious actor is able to compromise your organization's system with ransomware or spyware, or initiate a bank transfer, all from their email app. It's cheap, it's fast, and it's effective.

Additionally, something worth considering is the evolving landscape of our IT systems. For example, understanding cloud service types and deployment models, or web application security, are both extremely relevant for the current time. You see, as we have moved from on-premises servers and installing programs onto endpoints toward the **cloud services** model, software (and our overall estate) is increasingly accessed via a **thin client** (which usually means they're run in our preferred browsers), transitioning into what are known as **web apps**, delivered to users as a **Software-as-a-Service** (**SaaS**). **SaaS** applications usually have a monthly fee associated with them—that is, once the users require features that are offered beyond the barebones *free* tier.

Cloud service providers, such as **Amazon Web Services** (**AWS**) or Microsoft Azure, offer **Platform-as-a-Service** (**PaaS**) and **Infrastructure-as-a-Service** (**IaaS**) in a similar way, allowing for the metered usage of cloud computing power for a monthly fee. As time goes on, more and more of our estate is being outsourced and moved toward the cloud, either in the form of PaaS or IaaS. PaaS is usually offered as a simplified way to install and run an application without further configuration, allowing users to *outsource the responsibilities* of maintaining infrastructure and regular updates. IaaS, on the other hand, is a part of a server that exists in a data center in the region of your choice and, aside from the lack of physical interaction, the *responsibilities* for maintenance and updates to the system generally remain in the hands of the customer.

The concept is referred to as the **shared responsibility model**, showing the differences in the various cloud solutions, and where separate responsibilities lie.

Now, regardless of if you deploy your web app from your own *on-prem* servers, or on PaaS or IaaS, what we can see in web apps are certain types of vulnerabilities that allow malicious actors to compromise the confidentiality, integrity, or availability of your app. The **Open Web Application Security Project Top 10** (**OWASP Top 10**) is a regularly updated list of the top 10 web application security risks, and it does a great job of explaining how each of the risks can be mitigated. In order to give **application security** the amount of attention it deserves, I will revisit it in *Chapter 8*, *Improving the Security of Software*. Furthermore, the design, governance, and secure operation of your network and assets remain your responsibility, regardless of where the servers and computers currently physically "sit". We will explore network security in *Chapter 4*, *Designing and Protecting Network Security*, and continue until *Chapter 7*, *Owning Security Operations*.

This moves us on to the next step, where we'll look at best practices in assessing and mitigating vulnerabilities.

Best practices in assessing and mitigating vulnerabilities

For some of the vulnerability concepts we've gone through earlier in this chapter, I've included ways to mitigate against those vulnerabilities. None of those mitigations takes into consideration your own organization, its threats, the value of the assets, or the likelihood of exploitation. I've just essentially listed off ways you might have seen other organizations mitigate against those risks, and potentially a few good ideas came to you that way.

In this section, I would like to continue in that fashion, but I first want to stress how important it is to apply the necessary amount of mitigations in order to reduce down to an *acceptable level*. That's the name of the game here! Save yourself the money and headaches involved with making a system completely risk-proof when it doesn't face any threats or isn't valuable to you or a threat actor.

With that said, I can get back into how we can secure our systems and choose appropriate controls.

We've already gone into how to protect the different classes of ISO-defined information assets, which were hardware, software, networks, physical, and organizational. I just want to take a second to briefly revisit each, in case there's more to be said.

Hardware security

When we were reviewing hardware assets, we talked about ventilation, humidity control, cooling, lubrication, or quality-control as mitigations from a loss of availability realized through the destruction or powering off of hardware. We reviewed that threat actors don't need to be malicious humans; they can be environmental, such as "friction". Something I mentioned in that section that I would like to bring up again is **testing** after the implementation of a security control, in order to ensure the control is effective.

Throughout your career as an InfoSec professional, you want to ensure that after the appropriate controls are chosen and implemented, you first "test" the system for vulnerabilities yourself and patch any holes you find. Then (if the asset is valuable enough to justify it, remember) you have an external third party "test" it. This is often called **penetration testing** (*pentest*) or **red teaming**, depending on the complexity of the assessment. You can also perform automated scans, but the effectiveness of those scans have yet to supersede having an actual *pentest*, where you pay an expert hacker to actually break into your system. It's a beautiful and heartbreaking method of checking the effectiveness of your controls. You will end up thinking to yourself a few things. First, you will think: "I can't believe I missed that!", but then, you'll think: "Thank goodness I paid one of the good guys to find it first."

You'll take their report, implement the appropriate controls to mitigate against their findings, and ask them to retest in order to ensure your new controls are effective. It's an integral part of the job and provides major firepower in management-level or board-level presentations when you're talking about the likelihood aspect of your risks. You now have an example of an actual scenario where somebody broke into your system, and you now have the attention of the people who control the funds.

We also previously reviewed protecting access to storage, either through hard drives or memory. For memory, we talked about memory-protection controls in system architectures and operating systems, which are occasionally broken and need to be mitigated through changes to either the *kernel* or the *operating system*.

Data execution prevention is an operating system protection for memory that designates sections of memory as non-executable. Making this distinction prevents code from being run from those sections by giving a memory-access-violation exception, terminating the process, and preventing from buffer overflows.

For hard drives, I briefly mentioned that by encrypting the hard drive you're ensuring a level of defense-in-depth, by ensuring that even if physical access were obtained, further exploitation would be required in order to gain access.

One other aspect of encryption that would be useful to mention in the hardware security topic of this section is a **Trusted Platform Module**, or **TPM**. TPMs are dedicated microcontrollers that, with integrated cryptographic keys, help to secure hardware. They generally include a **random number generator** (**RNG**), which is hardware-based; the ability to generate cryptographic keys; and the ability to create a **hash** of the configuration and hardware in the system for integrity purposes. When you use *BitLocker* to encrypt your Windows 10 laptop, the best way to do so is by using the laptop's TPM, which also allows for *integrity checks* at boot time.

The concept of a **trusted computing base** or **TCB** has roots in risk management, where the TCB is determined as all of the system components that are high-impact in terms of security. If a vulnerability is found in the TCB, the risk of compromise to the entire system is higher. Determining your TCB may be something you do in the risk management process, if you determine that it's important. You can enforce that every system in your TCB has a TPM to verify integrity.

Additionally, operating systems utilize an architectural concept of a **reference monitor**, which enforces an access control policy over users and processes attempting to read and write to a system. The design paradigm utilizes an acronym of **NEAT**, standing for the following:

- **Non-bypassable**, which means an attacker can't get around the reference monitor and read/write to a system without permission
- **Evaluable**, which means the results and performance should be able to be analyzed
- **Always invoked**, meaning the mechanism should always be available, and always utilized for all security-sensitive operations
- **Tamper-proof**, meaning the mechanism cannot be changed to circumvent the control

All of these ideologies are very valuable in your own decisions, moving forward. I don't want you to look at built-in controls as something that are "handled", or not relevant to your role. A lot of the controls already available contain suggestions regarding how you could design *anything* securely, from your entire system down to the most minute function. We talked about evaluation earlier when we were talking about penetration testing. We talked about tamper proofing in the TPM's integrity checks. Looking at how these paradigms work from a micro and macro scale are important in selecting the appropriate controls for protecting against attacks on confidentiality, integrity, and availability.

We then moved on to protecting electrical signals from reverse-engineering and eavesdropping. We looked at mitigations against leaking emanations, such as shielding magnetic fields, filtering conducted signals, and masking space-radiated signals. We also briefly went over the restriction of physical access to the hardware, and also how that lacks defense-in-depth.

Software security

We then moved on to software, which can have vulnerabilities such as the following:

- Giving users access to more information than they should have access to, either through insecure code or a design flaw
- Being able to act as a delivery mechanism for malware
- The lack of an audit trail, leading to maintained access, exfiltrated data, and unseen changes

I will cover more on these vulnerabilities in *Chapter 8*, *Improving the Security of Software*, but for right now we can say that mitigating these vulnerabilities is generally done through code reviews, separation of duties, static and dynamic code analysis tools, and also the *penetration testing* of your software on a regular basis, and upon significant change. The lack of an audit trail can be mitigated by implementing one! By utilizing logging solutions, you are able to track software and network-based activities, such as access, changes, and traffic leaving your organization. With SIEM combined with analysts in a SOC, organizations detect ongoing attacks in real time and potentially react to them manually or automatically. Practicing and testing this control is part of a red team, where the defending side acts as the **Blue Team**. Testing the Blue Team is highly critical for some organizations and, yet again, it's up to you to decide if that's in line with the level of risk your organization faces. I cover more on security testing in *Chapter 6*, *Designing and Managing Security Testing Processes*.

Network security

We then moved on and talked about how *networks* face vulnerabilities that are exploited by a threat actor either through eavesdropping on insufficiently protected communication or by gaining access through insecure network architecture, or by causing a loss of availability. From a very high level, to control those, we're looking at the following:

- Encryption to protect communication
- Access control to control who has access to what, for how long

- Vulnerability management to find and patch vulnerabilities in all network architecture, along with the other devices in your estate
- Redundancy to protect against a loss of availability, (hopefully) protecting against *single points of failure*

We are going to delve deeper into this topic in *Chapter 4*, *Designing and Protecting Network Security*, so I will save further explanation on how to do so for that chapter instead.

Physical security

Physical sites have vulnerabilities which can lead to either loss of availability for your staff or systems, or circumvention of access control. To prevent from unauthorized access to your physical estate, we should consider a *defense in depth* approach.

Turnstiles and mantraps are entry controls, mitigating from a threat actor *piggybacking* the entry credentials of another person or *tailgating* behind somebody else, catching the door.

Biometric locks, such as a fingerprint scanner, ensure the authorized person is in fact the authorized person, not just somebody with a key or a badge. Sometimes, they aren't effective controls, either because they respond with too many **false negatives**, locking legitimate people out, or give out too many **false positives**, giving the wrong people access. These are worth considering when selecting this control, and questions should be asked for each use case surrounding: "Which is worse, the wrong person getting in, or the right person not being able to?"

On the topic of selecting the appropriate controls for your organization and its systems, let's move on to the next section.

Selecting appropriate controls/defense against the dark arts

If we're starting by discussing the time-based control category, I'm sure you're aware of the difference between **preventative controls** and **detective controls**: they're found in the names. The first helps to prevent attacks from being successful in the future, and the second helps to alert systems and personnel while an attack is in progress. **Corrective controls** help to restore systems and reduce the effects of a *breach*.

When we're talking about the type of control without time being a factor, we've already gone into some examples of each: **physical controls**, **procedural controls**, and **technical controls**. We mentioned guards, biometrics readers, turnstiles, and so on. Additionally, **closed-circuit television** (**CCTV**), motion sensors, fences, and so on are also considered physical controls. **Procedural controls** (or *administrative controls*) are pretty self-explanatory as well, and include processes (or administration) such as training and awareness; risk management processes; recruitment policies; disaster recovery plans; and so on. You have been given a load of technical control examples in this chapter so far, such as encryption, authentication, TPMs, data execution prevention, and so on. A couple of examples of **legal and regulatory controls** (also known as *compliance controls*) are the various privacy laws and information security standards that currently exist, sometimes carrying fines for non-compliance (such as the **General Data Protection Regulation** (**GDPR**)).

The selection of controls in order to adequately defend your organization's assets against the threats it faces should be based on a combination of the following two factors:

- The level of risk that asset faces, based on the likelihood of a threat exploiting a vulnerability, and the impact of that exploitation
- Your knowledge surrounding the effective application of various controls, depending on the threat vectors for that asset

We want to apply the required controls for the required use cases, and ensure that we've covering all threat vectors that are relevant to that specific asset and threat actor, but not wasting our time on applying controls that aren't relevant. If there is no need for a control to exist, then it's important to remember that complexity is the enemy of security, and reconsider increasing levels of complexity for no benefit in terms of reducing the level of risk.

Now that we've covered how to assess and mitigate vulnerabilities in hardware, software, network, and physical systems, let's move on to the next topic, and focus on designing secure information systems.

Best practices in designing secure information systems

With all of this knowledge that we just merely glossed over so far in this chapter (and which I'm fairly certain you already had) in regard to all of the threats, threat actors, motivations, vulnerabilities, and the methods for system exploitation that exist, the clear question that remains unanswered is: *"How can I design secure information systems?"*

Finally! We're going to talk about how to design a secure information system, and you're going to learn all of the ins and outs of how to do that in the best possible way. I'm really excited, but perhaps not for the reason you might think. You see, in learning about the threats, the devices and assets, and the methods, I truly believe you've already activated many of the thought processes required and have asked yourself many of the questions required to design a more secure system.

Many guidelines exist, but since this book is (trying to) focus on the best practices, I'm going to go over some issues that we've covered, and simplify the process of designing a secure system for you.

Secure design principles

When we're looking at the appropriate ways to design our systems and the way they interact with each other, and users, we want to consider the principles of **least privilege** and **Zero Trust**, and the ideologies behind the **secure by default** and **secure by design** principles.

The **principle of least privilege** is focused on ensuring that any access to a system resource must only be for the intended purposes, for the shortest amount of time possible. If a user or asset isn't meant to perform an action then it shouldn't ever be given the ability to do so, and so rights can be reduced down to the least-possible amount.

For an example, if you have a website being served to the internet via web server software that is running as the admin user, compromise of that web server could lead to critical or sensitive files in the network being read or deleted by a malicious actor. Does that web server asset need the ability to delete files? Does it need access to all of the sensitive files? No, so reduce the privileges down to the least amount possible.

As another example, if a user (or process) needs to read the files in a certain system folder, that user (or process) should only be given read privileges, not write privileges. Additionally, they should only be able to access the files for reading purposes for the period of time in which it is required. This is aided by **Just-In-Time access**, a methodology that enforces the least-privilege principle in access to your systems. It allows for access to requested resources, by provisioning an ephemeral one-time-use account that is deprovisioned immediately after the task is complete. This is granted (or not granted) based on risk indicators, such as "Is this a normal activity for this user?", or "Does this user have **multi-factor authentication** (**MFA**) enabled?", among many other things. Gone are the days when users are always "admins". The reason for this is simple: by reducing down to least-privilege and leveraging Just-In-Time access, we diminish the impact of a compromised account and decrease the likelihood of compromise through elevation of privileges.

Zero Trust is a modern security concept focused on a few ideologies. These include the following:

- Using the principle of least privilege across your system(s) through Just-In-Time and Just-Enough access, and encrypting and protecting your data at rest to prevent unauthorized access.
- Always verifying all activities, by authenticating and authorizing based on all of those datapoints we began to mention when we were talking about risk indicators for Just-In-Time access, which include geolocation, device metrics, anomaly detection, and so on.
- Assuming breach, the major idea behind Zero Trust. You restrict any lateral movements by segmenting all assets through the methods in the preceding two bullets; you prevent any eavesdropping by encrypting end to end; you detect active threats through logging and monitoring; and you continually improve by updating, patching, and evaluating the implementation of the system.

Zero Trust sounds pretty logical at this stage, after understanding the threats, vulnerabilities, and methodologies currently seen in the world, but as I said, it's a pretty new ideology that is getting more and more traction as time goes on, thankfully. Further reading can be found in *NIST SP 800-207*, if you want to have a bunch of fun reading a NIST Special Publication.

Secure by design is very much the end goal of what we're trying to achieve in all of our systems and is the main topic of this chapter. It's the principle of building systems to be secure from the beginning. When creating a secure-by-design system, we must first consider the threats, the assets, and the vulnerabilities that are inherently present, and then we must consider all of the security principles and controls we've gone into, and more!

I just want to take this moment to harp on at you that there is a ton of further investigation and learning to do after this. The continual improvement that we're going to be putting our processes and systems through will also be applied to you and your knowledge. As threats emerge and change and new technologies are developed, new vulnerabilities and mitigation controls will crop up too, and it's your responsibility to investigate all of them. I'm sorry that I can't go into all of them in this book, but what I hope I have done is help you in analyzing controls and ensuring you are selecting the appropriate mitigations for each of your risks.

Getting back on track, among these principles and controls, the best choices are selected strategically by understanding the risks implemented and enforced by the architecture design to ensure confidentiality, integrity, availability, and non-repudiation for the entire system. This is always done by avoiding **security through obscurity**, which will eventually fail at the hands of an insider risk or knowledgeable threat actor. We should be open about the design of the system because any knowledge of how the system is designed shouldn't compromise its security, because it is designed in a way that is secure and beyond any knowledge.

When we're looking at hackers, I would like to bring up a few bullet points I mentioned earlier when I was talking about the differences between a script-kiddie and an APT. I mentioned the following sequence for hacking:

- Gain access.
- Elevate privileges.
- Maintain access.
- Exfiltrate information or manipulate operations.
- Remain undetected.

When designing your system, you will want to consider how to prevent each of these steps from happening. How can you ensure a malicious actor isn't able to gain access? How do malicious actors elevate privileges or move laterally in your estate? How are they able to maintain access? Are you able to detect when data is exfiltrated or operations are manipulated? Many of the principles we've gone into so far are applicable here, and you can bolster your system through implementing a set of defense-in-depth controls for each.

Well-known controls and their mitigations

We should probably turn the way we are looking at this around, and start talking about well-known controls and which risks they might mitigate. That could help things click, and help us analyze our existing systems.

For example, **backups** prevent availability and integrity risks. If your systems have been affected by ransomware, providing you have a recent backup that hasn't been affected, then you can always restore your systems with that data, causing a minimized amount of disruption and loss. Backups aren't the only control for **ransomware**, of course. We said that ransomware often spreads as a worm and uses elevated privileges to encrypt the data in your estate, so by following the **principle of least privilege**, you reduce the likelihood that a compromised account is able to modify the data. By updating your systems, you reduce the likelihood that a piece of **malware** is able to exploit a known vulnerability in your systems. By having **antivirus** or **advanced threat-protection** software installed on all of your compatible systems, you apply a *detective control* to prevent the spread of that malware, and mitigate against the loss.

Reducing privileges can also help protect against **logic bombs** and insider threats from exfiltrating data such as intellectual property.

Antivirus software helps protect against known threats, usually through signature-based detection, and can help detect and remove the threats I mentioned earlier under the malware category, such as **spyware**, **adware**, **rootkits**, and **bootkits**.

Vulnerability management and **patching** can ensure that various known exploits aren't able to be used in your estate, because the manufacturer of the product has released an update to mitigate or remove the risk of compromise. There are always articles and news stories about **zero days**, but in reality, most organizations are running outdated software with publicly available exploits. The "Kiss principle" is relevant here. Focus on risk, and with a high likelihood (such as if the exploit is currently available on GitHub) comes elevated risk. I will delve deeper into vulnerability management and patching in *Chapter 7*, *Owning Security Operations*.

When it comes to physical security, we can mention **uninterruptable power supplies** (**UPS**) for availability during blackouts, brownouts, and power sag. We can consider **surge protection** for spikes.

We already mentioned **heating**, **ventilation**, **humidity control**, and **cooling** as mitigation tactics from physical threats, as well as avoiding the combination of water sources and your information systems. **Fire protection** is also a topic for concern. Human life is the most important consideration (although from what I've seen, I'm not sure 100% of organizations agree!). Fire protection is a way to protect employees from harm and is potentially a compliance requirement, depending on your region or operations.

The main thing I'd like to raise when considering physical security is **outsourcing**. Many of the physical security requirements we've covered are as a result of having servers and information systems in an office location, and by outsourcing as many of those systems possible by utilizing the cloud and **shared responsibility model**, you are able to transfer the risk for physical security to a company such as Microsoft or Amazon, who have data centers with incredible physical security controls, appropriate for various levels of industrial, military, and governmental activities.

Background checks and **due diligence** for new hires and third-party vendors are preventative controls that protect from threats exploiting a gap in your organizational processes. Furthermore, we could look at **security awareness training** and **phishing exercises** as mitigation against organizational threats. By training your employees and creating an environment of *security-consciousness*, you are able to reduce the risk of a successful phishing attack or negligent practices negatively affecting your organization's operations.

By utilizing appropriate **hardware and software firewalls**, **network topologies**, and **cloud services**, you are able to reduce the likelihood of a DoS attack (or even a DDoS attack) from taking your business offline, or increasing your monthly cloud service bill to a point where it can't be paid for.

By performing active testing on all of your assets, including performing **web application testing** against your web apps, taking into consideration the **OWASP Top 10**, you are able to get visibility on present vulnerabilities, as well as prevent from new vulnerabilities being added to your existing apps and software solutions. By allowing a third-party **penetration test** on your environment, you allow for the good guys to find holes that the bad guys would have found eventually, ultimately a **preventative control** for your organization. Did your systems notice the pentest? If not, then you don't have the appropriate level of detective controls such as **logging** combined with **SIEM** and a **SOC**, which is another control that can be outsourced to specialty firms with the ability to perform the appropriate analysis on the activity in your estate.

Considering alternative devices

When it comes to alternative devices such as mobile, **Internet of Things** (**IoT**), or **supervisory control and data acquisition** (**SCADA**), different aspects will need to be considered, because of the nature of each.

Starting with **mobile**—or, specifically, *mobile phones*—let's consider what we're looking at here: We have a device that people take with them everywhere. These devices may store your organization's sensitive data, or have access to your estate… and something that people do on occasion is forget them somewhere. You've given up **physical access**, so what is it exactly you need to do to ensure you're only paying to replace the phone, and haven't (for instance) given up your intellectual property to a competitor? Well, earlier in this chapter, we already covered the idea of encrypting your hard drives to prevent physical access leading to compromise, and once you encrypt your device, you probably will have put a lock screen on that is password-protected… but as this device moves around, and you might not know the last place you left it, you might want to consider the additional feature of the **Global Positioning System** (**GPS**) **Find My Phone** or **Remote wipe** functions. Additionally, if this device is owned by the user but still accesses your resources, you can have a security baseline that is checked before it is given access your estate. This checks if the latest patches have been applied, and usually requires some sort of virus-protection solution.

When we're talking about IoT or we're looking at devices that usually have access to the outside world through the internet, which may be created by a manufacturer who doesn't specialize in making hardware or software, but rather lightbulbs or stereo systems or vacuum cleaners or security cameras, and might not regularly patch reported vulnerabilities. It sounds like a bit of a nightmare for your estate, but don't worry—there are solutions to that. First, create an IoT network for all that stuff. Anything falling under the IoT device category gets put in that segregated sandbox, mitigating against the impact of a vulnerability that is exploitable over the web and allows access into the network (also known as **remote-access vulnerabilities**). Yes, it's happened… IoT lightbulbs have given remote access into corporate networks. It actually happened all the time, and it's pretty embarrassing.

Other mitigations for defense-in-depth purposes include selecting IoT products that have a good track record for releasing patches, applying the latest said patches, restricting privileges down to the least amount possible while retaining functionality (the **least-privilege** principle), and changing default passwords that are easily guessable or (because they're the same for every device) are found online.

SCADA systems are, for example, industrial control systems and devices such as **programmable logic controllers** (**PLCs**) that interact with machinery—a human/machine interface. Imagine a **graphical user interface** (**GUI**) or dashboard that Homer Simpson stares at all day at the nuclear plant. If he clicks "open" on a valve control on the GUI, that valve opens in real life. Sometimes, these things are hooked up to the internet, which sounds cool, right? IT IS NOT COOL! The problem is that most of these systems are extremely dated, and there has been a huge lack of concern when it comes to security during the design process. Sometimes, there are simple vulnerabilities that break authentication, or sometimes there isn't even authentication to begin with. Sometimes, the only protection is **security through obscurity**, where the developers and manufacturers thought: "Well, nobody will find *this* port, or understand *this* protocol, or find *this* IP address." Unfortunately, they do, through insider knowledge, or port scanning services such as `Shodan.io`, the search engine for IoT. If you haven't looked at *Shodan.io*, you definitely should do that (as soon as you finish the *Mission: Impossible* films); it's legitimately cool. You get a world map view, with red dots on the map that show various internet-connected devices, their IP addresses and open ports, and even a screenshot of the web page that you would see if you followed the rabbit hole and accessed the device over the internet in your browser. Awesome, right? Except, sometimes, there's a *wind turbine* or *electricity plant* that is exposed to the internet with no mitigations. A malicious actor could just turn everything off, or on, or do pretty much anything an authorized user could do!

In order to mitigate against the risk of high-impact functionality combined with outdated and insecure protocols, weak (or lack of) authentication, and ineffective security-through-obscurity controls, you should consider a few things. The first thing that you need to do is to find all of the connections and create a diagram of your SCADA networks. Then, if you can, **airgap** as many of these devices as possible. No internet access; separate network; no connection to the outside world. That doesn't really protect you from all of the threats, however. Strengthen the remaining connections in the network with authentication and authorization controls; install updates if they exist; disable unnecessary services that increase your **attack surface**; perform **penetration tests** and **red teams**; and make sure to always consider SCADA in your risk management processes.

The thing is, ideally, this is how you should be treating everything in your estate! Most of the best practices for securing your devices, network, and organization apply to any of these "alternative devices", as well as the typical devices. We care about risk, and we mitigate against those risks when we design and improve our information systems.

When you read about a new control, consider what risk they are actually solving (and not just what the marketing material says). When you're designing new systems or evaluating your existing ones, consider your controls and whether there's a gap that could allow for exploitation, and if that gap should be filled, depending on if the level of risk is unacceptable according to your risk management policy.

One other point I would like to raise in regard to designing secure systems is this: in your design for your system, please ensure you are able to keep the system up to date with the latest updates and patches! So many of the issues we see today are still because systems weren't designed in a way that allows for updates, be it due to legacy software not being compatible with new operating systems, or because of a limited access to the internet as a result of **air-gapping**. Design a process that allows for the updating of all of your assets, regardless of if they need to be air-gapped or not, and create a plan to ensure that your systems work on the most up-to-date versions possible. Keep track of whether your systems are running the latest versions through your asset register or configuration management tool (which I'll talk about in *Chapter 7*, *Owning Security Operations*), and update your risk register if a known vulnerability exists in an operating system or software version that is being used in your estate. If you're able to preemptively anticipate this and include these plans and processes in your design and implementation consideration, it's going to be a huge leap forward for your organization's security in the future.

Summary

In this chapter, we covered an incredibly vast amount of information, and I must reiterate like a broken record that each and every one of these key terms in this chapter could have an entire book (or series) devoted to a deep dive into them. The purpose of this book is more focused on reacquainting you with the key points, helping you look at each of these in the right way, and applying high-level principles to your knowledge to increase your effectiveness at your organization.

From assets and their vulnerabilities, to threat actors and their motivations, to the methodologies that those threat actors use to exploit the vulnerabilities in our assets, we leapt across several major issues in information security, both from strategic and operational standpoints, in order to properly understand the landscape.

By understanding the landscape, we moved into applying our risk management principles and knowledge surrounding the impact and likelihood of a threat exploiting a vulnerability, to mitigate and control against various risks. After looking at how to choose the appropriate controls for each risk, how the controls are categorized, and how many of them work, we looked at how to design systems with these risks in mind from the beginning, to ensure a more secure design by default and see how we might contribute to improve our own organization's systems the next time there's a discussion around their design.

Now that we're done with this chapter, let's move on to the next one, which I've alluded to a few times already.

4
Designing and Protecting Network Security

In all honesty, the end of the previous chapter and the beginning of this chapter is an illusion. It's definitely the case that we are building on top of the knowledge we gained from the previous chapter, and we're going to lean on that chapter heavily in this one. The reason for this is that because we had covered so much already in the previous chapter, my publishers and I decided it was necessary to create an imaginary boundary that signaled the end of one topic and the beginning of another. I'm going to do my best to not repeat myself, unless the point is worth repeating.

Now that we've got the "ambiguous and invisible chapter boundary" problem out of the way, I can get into the housekeeping for this chapter. In this chapter, the goal is to learn how we can combine our knowledge of threats, vulnerabilities, and mitigations from the previous chapter, along with our knowledge of risk management and governance, to create secure-by-design network architectures (or at least something that is as close to secure-by-design as possible, while balancing usability).

We'll talk about securing the various network components with design choices and mitigations to bolster your organization's network security. I also aim to discuss how we might ensure we secure various network components when those components aren't under our control anymore. As we all continue our combined migration to the cloud, like a digital Oregon Trail – no, not the video game – we will venture into new challenges and reduced control. The main difference between the current cloud migration and the Americans heading west in the 1830s is that in the move to the cloud, fewer people will die of dysentery. Well, at least I hope that's the case.

After that, we're going to look at implementing secure communications between any combination of assets and users, including any combination of cloud and on-premises.

We will cover the following topics in this chapter:

- Designing secure network architectures
- Strategies for protecting network security

So, let's get started!

Designing secure network architectures

If your InfoSec learning path has been anything like mine, a load of your time has been spent learning about networking. Do **Chief Information Security Officers** (**CISOs**) talk about networking very often in the day to day? Well, yes… networking is mentioned in the LinkedIn/evening drinks/corporate "networking" type of way, but they're very rarely talking about **TCP/IP**.

Does that suggest that you shouldn't know about networking? Or that you should forget all that you've learned? Of course not. By knowing about these topics, and understanding the likelihood of a threat exploiting a vulnerability in network devices or protocols, along with the impact of that event, will enable you to make more informed decisions for mitigations, rooted in the principles of risk management.

In my opinion, it's worth learning a bit of everything in InfoSec, regardless of the path you currently have your sights set on following. Learn about something that is happening in the world of InfoSec at the moment, or something that is going on in your organization. When you start to feel a level of diminishing returns on the time you're putting into it, reassess whether the topic is something you enjoy learning about or something you need to know, or if you can focus on something else for the time being.

Of course, the concept of network security is much more than just protocols, bits, bytes, and the systems that are associated with data in motion, but let's quickly cover some of the concepts and definitions we must be aware of, and ensure we're on the same page.

Internet Protocol suite and the OSI model

There are two major models that I'd like to reference in this chapter to help you conceptualize how users and applications interface with nodes and network devices, in order to communicate with other nodes and networks, including over the internet. They are known as the **Internet Protocol suite** and the **OSI model**.

By understanding both, and how they overlap, we can get a better idea of how to categorize, and therefore protect and govern, the various devices in our estate.

TCP/IP suite

The **TCP/IP suite** or the **Internet Protocol suite** is a group of communications protocols and a model that is used on computer networks and the internet, and its importance to this topic cannot be understated. Development funding for this protocol stack came from **DARPA**, the **Defense Advanced Research Projects Agency** in the *United States Department of Defense*. This is a bit of fun trivia, if your idea of "fun trivia" includes questions such as "Which military agency developed the Internet Protocol suite?"

It's referred to as **TCP/IP** because of the **Transmission Control Protocol** and **Internet Protocol** – **TCP** and **IP**, respectively – both of which are two of the foundational protocols used in the model, and therefore two of the foundational protocols used on the internet.

TCP/IP gives us end-to-end communication, from being turned into packets, to being addressed, to being transmitted, routed, and ultimately received. In the TCP/IP model, there are four "layers" that split various protocols into classifications related to their scope:

- Communication methods for data remaining inside a single network segment is done in the **link layer** or **network access layer**.
- Communication methods for data that traverses between independent networks is done in the **internet layer**.
- Communication between hosts is done in the **transport layer** or **host-to-host layer**.
- Communication between processes is done in the **application layer**.

The TCP/IP Protocol Suite abides by standards that have been set out and maintained by the Internet Engineering Task Force.

If we wanted to make a diagram using the Internet Protocol suite layers, showing an application on a host sending data to another host's application over two routers, it could look something like this, where **Host A** has the communication travel down the TCP/IP data flow layers, and then up and down the link and internet layers of two routers, before traveling all the way back up to the application layer of **Host B**:

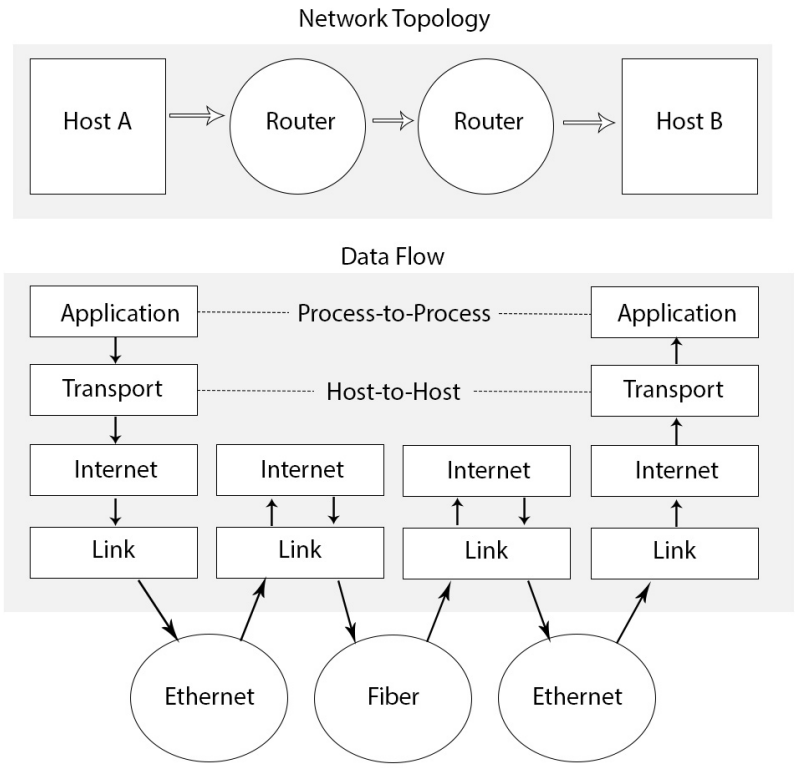


Figure 4.1 – Diagram of a TCP/IP data flow

Now that we've reviewed the TCP/IP protocol suite, it's definitely worth delving into a different way of looking at the same concept, in what is known as *the OSI model*.

OSI model

Another model for standardizing communications in a computing system is the **Open Systems Interconnection** model, or **OSI** model. It is a standardization of communications functions, regardless of any internal structure or technology. The OSI model has seven layers, as opposed to TCP/IP's four:

- **Layer 1 – Physical**: Handles converting digital bits into radio, electrical, or optical signals, with voltage levels, timing, maximum distance, modulation, physical connectors, layouts of pins, and cable specs.
- **Layer 2 – Data Link**: Handles the connection between two nodes communicating directly with one another. It can correct errors from the physical layer and is able to control the connection status, such as establishing or terminating a connection, along with controlling the flow of data.
- **Layer 3 – Network**: Handles the connection between two nodes in separate networks, potentially using routing via other nodes. It can also handle larger messages between two local nodes than the data link layer can, by splitting the messages into fragments and delivering those separately.
- **Layer 4 – Transport**: Gives us the ability to transfer messages from two nodes in the form of variable-length data sequences. It has controls in place for error handling and data flow, and can keep track of messages that succeed and fail, allowing the failed segments to be resent. It is usually agreed that the TCP and UDP protocols sit in Layer 4.
- **Layer 5 – Session**: Creates, maintains, and terminates the connections between local and remote applications from Layer 7. Usually, Layer 5 is implemented in applications that use **remote procedure calls**, or **RPCs**, thus treating programs that are on another node or network as though they were local applications. It handles requests from Layer 6 before passing them to Layer 4.
- **Layer 6 – Presentation**: Translates between network and application formats, transforming the data from the previously mentioned layers into a suitable format that is acceptable based on the destination application. For this reason, it's commonly referred to as the *Syntax* layer. It can also compress data, and can map the transfer between applications directly, preventing the ability to "pass down the protocol stack" to the previously mentioned layers.
- **Layer 7 – Application**: Handles communication with the software application that a user also interacts with. It is responsible for showing the user the relevant data that has been transmitted.

To help conceptualize what type of data is handled by which layer in the OSI model, I've included the following diagram:

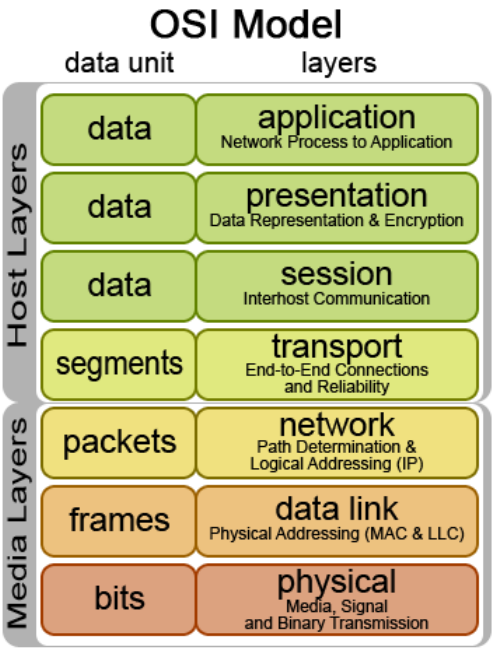


Figure 4.2 – Diagram of the OSI model

If we were to look at the OSI model and see how it might interact between applications over the internet in a similar way to how we looked at the Internet Protocol suite, we could make a diagram that looks like this:

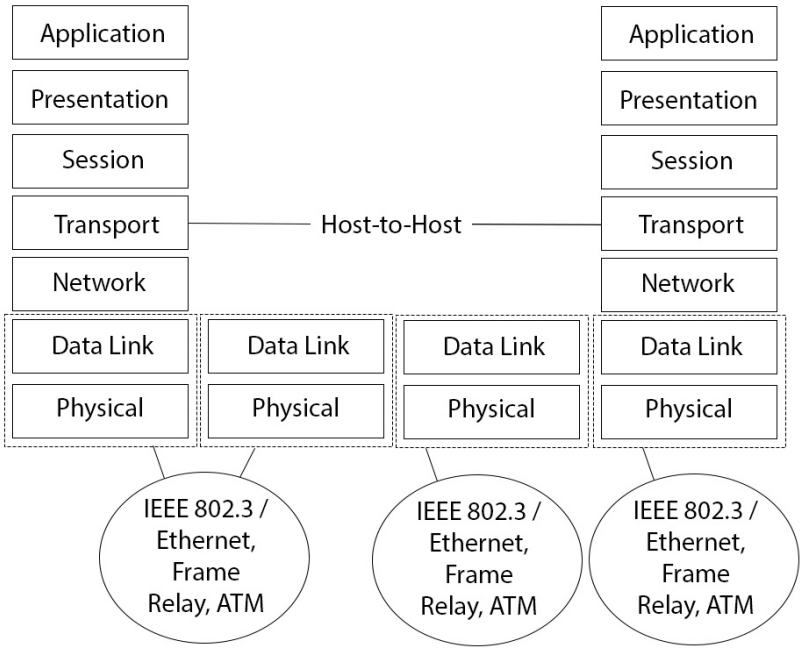


Figure 4.3 – Flow for the OSI model

So, if we were to compare the two, meaning the Internet Protocol suite and the OSI model, what would we find? Let's get into it!

Comparing the Internet Protocol suite and the OSI model

When comparing the two different models, we can see a few things:

- The OSI model separates the activities in the TCP/IP application layer into the application, the presentation, and (partially) the session layers.
- The host-to-host/transport layer in TCP/IP is handled with the remainder of the session layer, and the transport layer in the OSI model.
- The TCP/IP internet layer is only a part of the OSI's network layer, while the remainder of the network layer, along with the entire data link and physical layers from OSI, fall into TCP/IP's network access layer.

If we wanted to make a diagram to compare the two models, you might come up with something that looks like this:

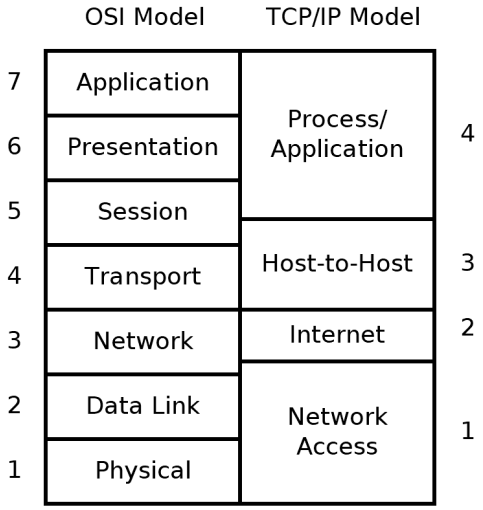


Figure 4.4 – OSI versus TCP/IP

Various concepts Internet Protocol suite and components surrounding networks exist that we haven't covered yet, such as how TCP works, or UDP, or what ports are. Let's dip our toe into that now.

Network components and protocols

TCP, or the **Transmission Control Protocol**, allows data to be delivered as a stream of bytes over a network. These bytes are sent so that they're checked for errors and ordered in such a way as to improve reliability. Its design hasn't changed that much since the 1970s, and has been used for the `HTTP/1.1` and `HTTP/2` application layer protocols for the web. Recently, it was decided that their successor, `HTTP/3`, would instead utilize the *QUIC protocol*, which has **UDP** at its heart.

TCP is known for its three-way handshake, which is used in establishing a connection. Imagine that a client wants to connect to a server that has a passively open port for connections. This is what happens:

1. The client sends a message to the server, setting a sequence number and creating an "Active Open" connection, otherwise known as a **SYN**, *or Synchronize*.
2. The server responds to the client's **SYN** message with an acknowledgment, incrementing the client's sequence number by 1, and setting its own sequence number, otherwise known as a **SYN-ACK**.

3. The client then replies to the server, with both the incremented sequence number sent by the server, along with its own sequence number incremented by 1, acknowledging that the connection has been made. This is known as the **ACK**, *or Acknowledge*.

Now that you definitely know this, you're ready for a TCP/IP joke:

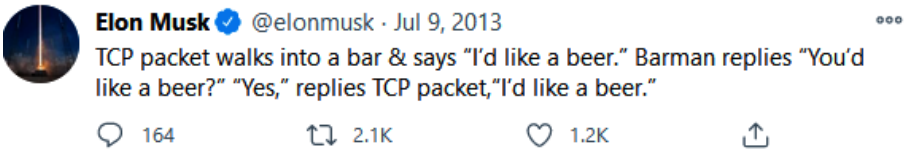


Figure 4.5 – Elon Musk tells a joke

To terminate a connection on TCP, a *four-way handshake* must occur, with each side able to initiate the termination. For an example, if a client wants to terminate a connection to a server, this is what happens:

1. The client sends a **FIN** packet, which is an indication to gracefully terminate the connection.
2. The server responds with an **ACK** packet.
3. The server sends its own **FIN** packet.
4. The client responds with an **ACK** packet.
5. The client stops accepting new connections, waits for a timeout, and then finally closes the connection.

Linux operating systems handle things a bit differently, sending an **RST** packet instead of a **FIN**, which is an immediate termination, rather than gracefully terminating the **FIN** packet.

Additionally, *steps 2* and *3* can sometimes be combined into a **FIN & ACK** packet being sent at once.

The **User Datagram Protocol** (**UDP**) is a member of the Internet Protocol suite, even if it doesn't make an appearance in the name of the protocol the way TCP does. UDP lacks a handshake to confirm a connection, and is used under the assumption that any errors or corrections aren't necessary, which removes any processing requirement from the network interface. Packets can be received in an order that isn't congruent with the way they were sent and packets can be dropped, and for that reason it is seen as less reliable, but valuable for speed. This means that UDP is faster and more lightweight than TCP, and that it is popular in online games and video chats.

The **Domain Name System** (**DNS**) is a decentralized naming system that translates *IP addresses* into human-friendly domain names, both for private networks and the internet. It's a phonebook for the internet. It's important to note that it's hierarchical in nature, and that the resolution of a domain name to an IP address is done through a sequence of queries beginning with the rightmost domain label, such as `.org`, `.com`, `.net`, `.dk`, `.wtf`, and so on.

Common free DNS resolver services include Cloudflare (`1.1.1.1` and `1.0.0.1`), Google (`8.8.8.8` and `8.8.4.4`), Quad9 (`9.9.9.9` and `149.112.112.112`), and OpenDNS (`208.67.222.220` and `208.67.222.220`), among others. These serve to route the requests that are sent to specific domain names to their IP address counterparts.

Internet Protocol (**IP**) addresses are identifiers for each device that use the IP and connect to a computer network. The format (for **IPv4 addresses**) is `32` bits, with four sets of numbers between the range of `0` and `255` (otherwise known as octets, or `8` bits); for example, from `0.0.0.0` to `255.255.255.255`, and everything in-between, such as `255.0.0.0`, `0.255.0.0`, or `108.177.8.63`.

Internal IPv4 "blocks" are also determined: `10.0.0.0` to `10.255.255.255` contains 16,777,216 addresses that have been reserved for private networks. The same goes for `172.16.0.0` to `172.31.255.255` and everybody's favorite, `192.168.0.0` to `192.168.255.255`.

IPv6 is a wild idea that hasn't really taken off. In 2074, when it's actually used, I'm sure there will be an author who is interested in going into the details of it. Until then, a casual understanding is fine, as long as it's not part of your everyday work. It's similar to IPv4 but has a larger number of possible addresses, and also has some features that help optimize routing.

Subnets or subnetworks are subdivisions of an IP network. If you split your network into multiple networks, you're subnetting. You can do this by grouping computers that belong to the same network into the same IP address group, thus improving your organization by partitioning a network address space into logical groups.

For example, you might have a network number for the first two octets, a subnet identifier for the third octet, and a host identifier for the final octet. You assign one subnet the routing prefix of `192.168.100.0/24` and another the routing prefix of `192.168.200.0/24`. This defines the first `24` bits (3 octets) as the network prefix and the final octet as the host address space. There are 256 addresses for each of the subnetworks, thus creating a more organized structure for your network.

Let's take a look at a visual breakdown of this concept:

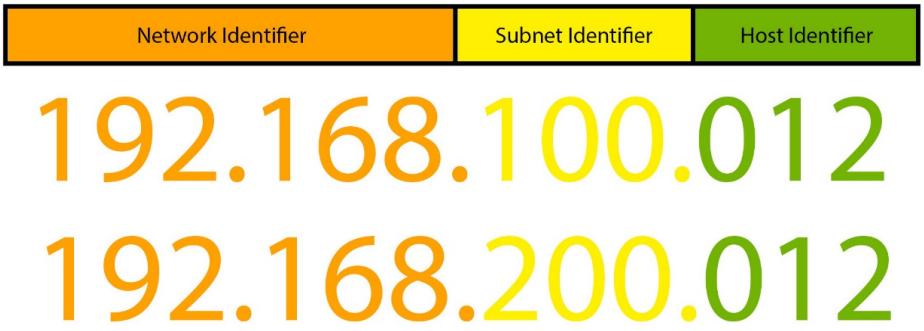


Figure 4.6 – Subnets visualized

In large organizations, or various other scenarios, 256 addresses might not be enough for a subnet, so I have only used this as a simple example.

Network ports

Communication endpoints in networks are called **ports**. Operating systems use logical "ports" to identify network service processes. Both TCP and UDP use port numbers to reach the logical endpoint. Some common port numbers that have been assigned by the **Internet Assigned Numbers Authority** (**IANA**) that are worth remembering off the top of your head are as follows:

- **Port 20**: FTP data transfer
- **Port 21**: FTP command-and-control
- **Port 22**: SSH, a secure shell
- **Port 23**: Telnet, a remote login service
- **Port 25**: SMTP, mail routing
- **Port 53**: DNS, which is usually open
- **Port 80**: HTTP, used for pages without TLS/SSL/HTTPS
- **Port 110**: POP3, for mail clients
- **Port 123**: NTP, for network time
- **Port 143**: IMAP, for mail
- **Port 161**: SNMP, for device management on networks
- **Port 443**: HTTPS, for HTTP traffic with TLS/SSL/HTTPS

This isn't to say that these port numbers are always associated with that type of protocol traffic, though. The thing is, you could have internal HTTP traffic on `8080`, `8081`, or `3000` (or pretty much any other port), depending on the web app framework you're using. It's important to note that the ports are always associated with a host IP address and the type of protocol it's using. If you wanted to reach a page on an internal Vue.JS web app that is being presented from a server with an IP address of `192.168.1.25` on port `8080`, you could navigate to it through your browser via `192.168.1.25:8080`, and it would open the web app's home page for you from that port, instead of via port `80` or `443`, which are the defaults for the browser.

What I'm getting at is that these are common port number designations, but they're not rules.

The fact that it is possible to remotely access devices through protocols such as SSH is a risk that may need to be mitigated by your organization. The reason you might want to do this is because there are automated tools that are constantly trying to log into any public-facing IP address, and then trying to SSH on port `22`. In the name of time, they will generally try port `22`, and then move on to another low-hanging fruit if it doesn't work.

In order to obfuscate the most common automated attacks on port `22`, you might decide to change your default port for SSH on your servers from `22` to a different port, for example. That way, if an automated tool is attempting to SSH into one of your servers, it has to figure out which port SSH is on. That being said, that's not enough to protect against SSH attacks, and it's definitely *security-through-obscurity* rather than a true mitigation, but it's the first step that can be taken for a *defense-in-depth* approach.

One more thing to note about network ports is that you can have multiple services on one server. You can SSH into the server, but also access a web page and handle emails, DNS, SMTP, and so on.

Next, we'll look at the various hardware devices and applications we can use for networking.

Network devices and applications

Networks have a wealth of various hardware and application-based peripherals, and it's worth looking into a few of the security-related ones here, to ensure we're on the same page. This will also help preface the next section, where we will be dealing with common attacks and tactics for defense and detection.

Network switches

A **network switch** is a piece of hardware that sits in-between your devices, through Ethernet connections typically, and forwards traffic to the appropriate device. It operates on *Layer 2 of the OSI model*, and uses the device's *MAC address* to forward data to the device. If the switch also has routing functionality, it could also operate on *Layer 3 of the OSI model*. These switches are usually called *Layer 3 switches*.

A key takeaway about switches is that they only send data to the appropriate address, rather than to every address connected to it, the way hubs do.

Routers

Thankfully, since we've already learned about ports, IP addresses, layers, and so on, we can quickly discuss how the internet works. Routers are devices that forward data between networks. In order words, routers direct internet traffic. Routers communicate with one another by forwarding data packets from their network to another network. Here, the network's router receives the packet, reads the destination node address, and by using its routing table, either forwards it to another network or to the destination node that sits in its own network. Enterprise routers connect businesses to the core routers, optical fiber, and undersea cables. These are known as **the internet backbone**:

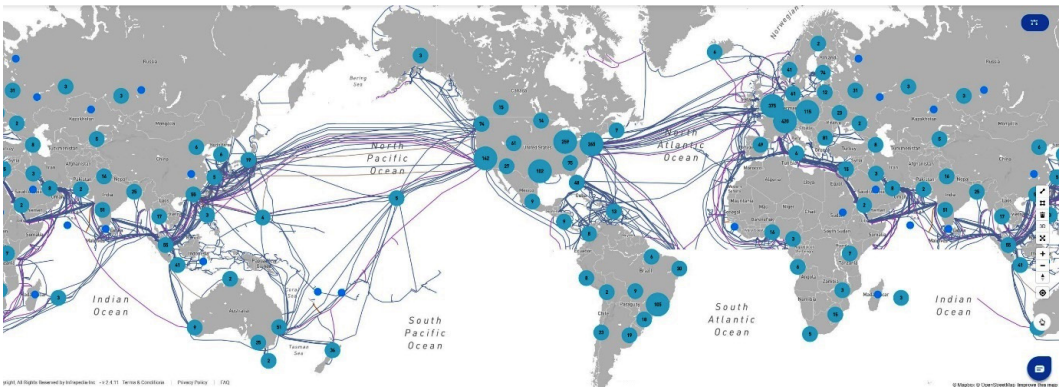


Figure 4.7 – Undersea cables and data centers, also known as "the internet backbone"

Isn't it funny that the internet's core infrastructure is cables lying at the bottom of oceans? As payments, electricity, and supply chains become more reliant on the internet, wouldn't you agree that it's crucial to ensure we reduce the risk of any **Single Point of Failures (SPFs)**? What are some of the vulnerabilities of this current setup?

Load balancers

A service that can delegate processing over an array of servers is known as a **load balancer**. This service helps improve efficiency and prevents you from over-utilizing one server, while underutilizing another that performs the same task. It helps to improve stability and increases speed.

Proxies

Proxies are server applications or hardware that help the client make requests to the server rather than letting the client make requests directly to the server, and then pass back the response. Proxy server processing can include load balancing services or security processes that simplify the request and make it fit for the server to handle, or perform error handling to get the proxied information response.

Firewalls

Another system that controls network traffic, this time based on security rules, is called a **firewall**. This includes both ingress and egress traffic, and protects your trusted network from sending and receiving traffic to ones you may not want to trust… such as the internet.

There are software firewalls, hardware appliance firewalls, and virtual appliance firewalls. Software (or host-based) firewalls are installed on the endpoint as a service or agent to control traffic and network resources.

Here's a classic diagram of a firewall, visualized as a brick wall:

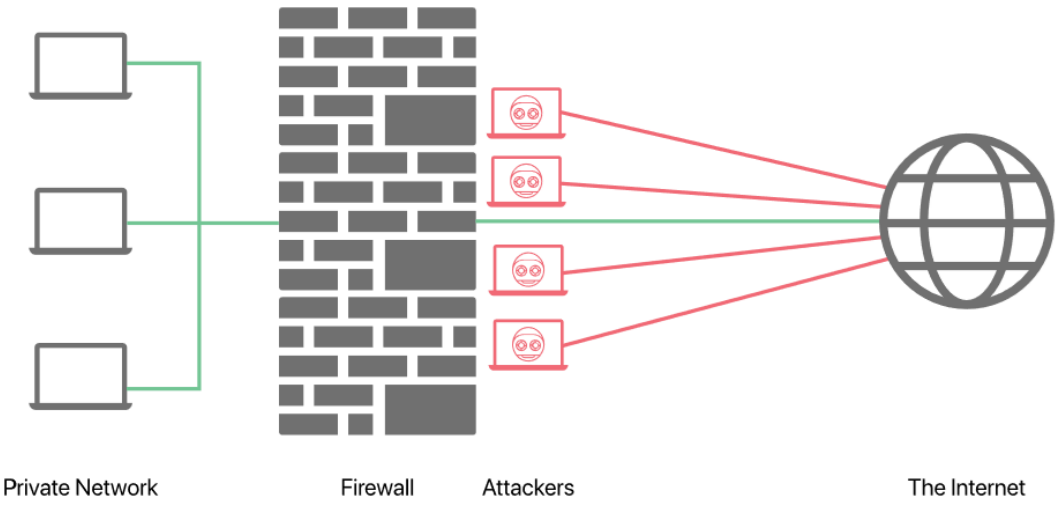


Figure 4.8 – Classic firewall diagram

Having shown that, I'd also like to go into how firewalls work, from type to type, and how they go about allowing traffic in, not like a brick wall, but more like a bouncer at that nightclub you were never allowed into.

Packet filter firewalls

Firewalls that act as proxies and inspect the data that is transmitted between endpoints are called **packet filter firewalls**. They refer to an access control list that determines the packets that are inspected, and what should be done to each. A few examples of the actions that can be taken could be to discard, forward, or send a *TCP RST* response to the sender and close the connection. You might think of these actions as firewall rules, which can filtered by source and destination IP addresses or ports, or by protocol.

Consider the benefits of this: you're able to control which protocols communicate on which ports to which devices. That's a brilliant step in the right direction for your information security posture. Your existing list of assets can include information that will allow you to fully understand what communication is required, and what communication isn't required. This is powerful because often, a "flat network" increases your exposure to ransomware, network traversal, and privilege escalation. Imagine that you have created a rule in your firewall that blocks all communications between subnets, except port `3306` (which is *MySQL*) from `192.168.100.25` (which we can imagine is an application server address) to `192.168.200.50` (which we can assume is the MySQL server).

Now, if a command is attempted from a different application, deleting all the data in the SQL database isn't effective because that command has been filtered by the firewall.

> **Important note**
>
> You should definitely also consider authentication, authorization, and preventing such actions that are out of the ordinary. This is a form of defense-in-depth.

The *Deny-All, and then Whitelist approved traffic* technique is highly effective at preventing worms from spreading in your network, as well as lateral movement. However, in my experience, it is difficult to implement in organizations with complex networks that have components and applications communicating between devices that aren't recorded. Blocking that traffic could lead to an outage and cost the organization money, so some serious preparation is needed to fully understand what *stateful firewalls* are doing at any given time.

Connection tracking firewalls

Connection tracking firewalls, or *stateful firewalls* (or *second-generation firewalls*) are similar to first-generation proxy/packet filter firewalls, but they retain information about communication between endpoints by keeping the IP addresses and port numbers from Layer 4. This allows for an overall better understanding of how the nodes are interacting. Instead of inspecting each packet, it utilizes the data it has from the context of previously open connections. As a result, they have a performance advantage over the packet filtering firewalls as the bouncer is allowing the regulars straight into the bar, while asking the people he doesn't know for ID.

Another advantage of stateful firewalls is that they protect against port scanning by only opening ports when they are required by specific incoming packets.

Unfortunately, with this power comes the ability to trick the firewall into thinking you're a trusted party by, for example, having some JavaScript on a website request information from the malicious actor's endpoint.

Application firewalls

Application firewalls (or, **Layer 7 Firewalls**) can control system calls to a service or app on a system, according to a policy that has been configured. It's called a *Layer 7 Firewall* because it is able to control all the way up the OSI model to Layer 7, which is the application layer. They can be either host-based or network-based, and are able to protect against unwanted applications that are using a port that isn't the standard IANA designation.

A **next-generation firewall** performs deep-packet inspection at the application layer in order to enable **intrusion prevention system** (**IPS**) functionality, *sandboxing*, *user identity management*, *web filtering*, and *web application firewall* tooling.

Web application firewalls

Web application firewalls can protect your web apps from malicious users. The web app firewall handles HTTP traffic as a reverse proxy before it reaches the server, inspects the traffic, and uses policies to filter activity. This activity could include common web app exploitation techniques often used by malicious actors, such as CSRF, XSS, and SQL injection attacks, and enabling *rate limiting* helps prevent DoS and DDoS attacks from being effective.

Endpoint firewalls

Endpoint firewalls are application firewalls that protect against unauthorized connections. They will generally scan a process ID for the data packet and compare that against a built-in set of rules. They exist at the application layer, but filter connections between Layer 7 and the layers below. They might also be known as *socket filters* since they filter connections via socket calls on the host.

The importance of firewall configuration

As I mentioned earlier in this chapter, misconfiguring a firewall can cause serious issues in an organization, and getting it right is a highly important task. Understanding the way traffic moves in your networks, as well as designing a way in which it *should* move, is an imperative step in ensuring proper mitigation against risks in your organization.

Tools are available to help inspect existing traffic and help you better understand exactly what the traffic might be used for. These tools are often also used by malicious actors once they're inside your network, as they are highly effective when used properly.

Sniffing tools

Tools available for inspecting network traffic, in order to better understand the nature of the communication, are considered **packet sniffers** or **packet capture** tools. A very well-known and highly utilized example of these tools would be **Wireshark**, a network protocol analyzer that can be used to understand networking issues or detect intrusions but can also be used to spy on network users, collect sensitive information, and exploit your network. Other examples include **tcpdump**, which runs on the command line, and **Ettercap**, a security tool that poisons the ARP table on the target machines, acting as a **Man-in-the-Middle**. We will discuss these terms and how to protect against these threats later in this chapter.

Network Intrusion Detection and Prevention

Intrusion Detection Systems (**IDS**) are either software suites or devices that monitor a network for policy violations. **Network Intrusion Detection Systems** (**NIDS**) monitor network traffic, while **Host-based Intrusion Detection Systems** (**HIDS**) monitor operating system activity.

Sometimes, these tools will utilize reputation-based algorithms to detect known malicious sources and destinations, or even machine learning models, to flexibly correlate based on a large dataset.

A **Security Information and Events Monitoring** (**SIEM**) system would be collecting all of the events in your network from these IDSs and IPSs, and combining all of the metrics in order to correlate activities and determine if these are actually malicious or false alarms.

Intrusion Prevention Systems (**IPS**) are devices that respond to detected threats and prevent the actor from performing the attempted action, either by reconfiguring the firewall rules or by modifying the content of the request to prevent exploitation.

VPNs and their uses

Virtual create a virtual boundary to allow users in a public network to create, read, update, and delete data in the same way they would be able to if they were in a private network. Imagine a tunnel from a user's home to a corporate office. If the user sends a request through the tunnel, the response comes back through the tunnel. Generally speaking, that communication is protected from any sniffing via encryption.

VPNs are commonly used by businesses to allow distributed teams to work in a shared environment, but other uses have appeared. For example, if a government prevents their citizen's internet connections from being able to access Facebook IP addresses, a user could, alternatively, send a request through a VPN connection to a PC in another country, which isn't owned by Facebook, which then requests and passes on the information from the Facebook servers to the end user.

Content delivery networks

Content delivery networks (**CDNs**), are designed to provide low-latency, high-availability access to your servers' data by geographically distributing various proxy servers around the world. If you have a user in Australia, for example, they'll reach out to their nearest server in Sydney, which both holds a cache for much of the content on your host server and performs a request to your server for the content it doesn't have.

Most of the web traffic that you request daily is actually served by a CDN cache, with the latest (locally unseen) content being the only thing served from the source host. Oftentimes, CDN providers supplement their service with security tools for protecting against web application exploitation tactics and denial of service attacks.

On that note, let's look into methods for attacking systems and how we, as information security professionals, may defend against those attacks.

Attacks, defense, and detection

So, if you put your hacker hoodie on and seriously think about it, you might be able to dream up a couple ways to exploit these systems. How would you go about doing that? When we're looking at methods malicious actors may use to compromise the security of your organization's assets, there are new techniques being utilized continually, but most of the classic techniques still work. I'm going to use this section to go over a few of the most common attack techniques and their mitigations.

Man-in-the-Middle attacks

Man-In-The-Middle (**MITM**) attacks are where the malicious actor secretly sits "in the middle" of your communications, passing information on after potentially reading or altering the message. If the malicious actor in the middle is reading the messages, it's also known as *eavesdropping*.

MITM defense and detection

When looking at mitigating against MITM attacks, we need to look at cryptography. Cryptography can offer authentication, proving mathematically that the source of the message was legitimate, as well as proof of integrity, which can mathematically prove that the message hasn't been altered.

In order for this to work, you'll need a way to exchange keys or certificates, for example, a public key infrastructure, as we see in TLS traffic. If it can't be proven that the identity of the source or destination is valid, the session is ended, and communication stops. The weakness in this solution is that security is lost in the event that the Certificate Authority, which issues the cryptographic certificates, is compromised.

If you are capturing and analyzing your organization's network traffic, you would be able to gather information about each server your network interacts with, such as the source IP address, DNS name, X.509 certificate, Certificate Authority, and whether other clients have the same certificate.

DNS hijacking

DNS hijacking is the act of performing DNS resolution with a **rogue DNS server**. If you, as the attacker, can get a computer, a server, or a router to perform a DNS lookup to a DNS server that has been poisoned or is under your control, rather than one of the authentic public ones we've mentioned previously, we can make it believe they're interacting with the legitimate service. Sometimes, this is called **pharming**, but the term didn't really catch on the way **phishing** did.

DNS hijacking defense

Using the previously mentioned DNS servers, such as those from Cloudflare, Google, and OpenDNS, with **DNS-over-HTTPS (DoH)** or **DNS-over-TLS (DoT)**, you can prevent returning spoofed results. DoH/DoT mitigates against both eavesdropping and spoofing, but unfortunately, it can impede monitoring DNS traffic for anomalies and malicious activity. The DoT server eventually reaches the authoritative DNS servers using port `53`, unencrypted, so it's not end-to-end encrypted, and since TLS may not implement the entire route properly, it doesn't guarantee privacy.

Additionally, webmasters can create certain entries for their website's **DNS records**, which protect against hijacking on their domain, but that's not a very scalable approach and relies on the providers being proactive.

ARP spoofing

With **ARP spoofing**, a mapping between a **MAC address** and an **IP address** in the **ARP table** is changed, but the table continues to act as a validator and as a facilitator for connecting a source to a destination. **ARP spoofing** is either done to perform a *DoS* attack by directing all traffic requested from several IPs to one device MAC address, or a *MITM* attack, such as modifying traffic, stealing session IDs, and more.

Sometimes, ARP spoofing is done for redundancy purposes, to ensure that when a server goes down, another is able to step in. Another legitimate use case for this technique is being able to debug and analyze traffic between two hosts while using a switch, as switch traffic is not transparent/replicated across all the endpoints on the network.

Imagine that a user is monitoring **Host C**. If they want to watch traffic go from **Host A** to **Host B**, they can set **A's** ARP to consider **B's** MAC address as **C's** actual MAC, and then set **B** to consider **A's** MAC address as **C's** actual MAC, and allow for **C** to forward the packets. At this point, they have a MITM situation.

This is really hard to imagine through my description, I concede. Here's a diagram that should help you out:

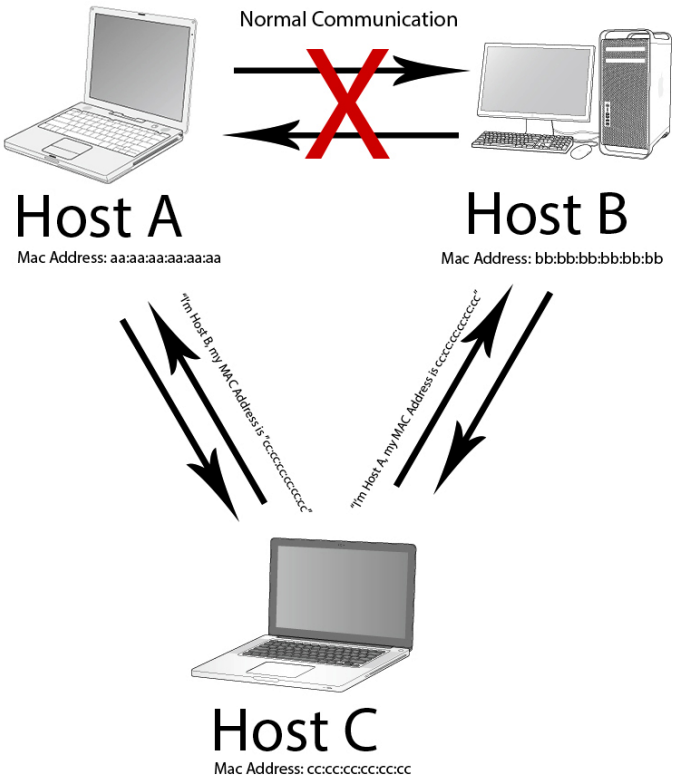


Figure 4.9 – A diagram of an ARP spoof

As you can see, tricking computers to believe one device is another one is a pretty simple way to eavesdrop and requires adequate protection.

ARP spoofing defense

You can set up your ARP cache for critical servers and endpoints so that they're *static* and *read-only*, which means they don't allow updates to be made. This is extremely simple, but adds to the overhead for IT. It's also very difficult to scale because each machine must be mapped accordingly, meaning you're looking at *n^2-n* ARP entries, where *n* is the number of endpoints on the network. If you have 1,000 endpoints, you have 999,000 ARP entries.

Alternatively, you could use a software solution that provides a certification for ARP responses and can be built into the DHCP server to ensure all IP addresses, either static or dynamic, are handled properly. Alternatively, you can just send a notification to an admin in the event that an ARP entry changes, but that's not really a preventative control.

You can also configure your hosts operating systems so that they gracefully handle unsolicited changes to the ARP cache.

DNS spoofing

DNS spoofing/DNS cache poisoning is similar to an **ARP spoofing attack**, and is where a trusted source of information is poisoned with malicious data. With **DNS spoofing**, the victim is requesting a response from what looks like a legitimate source (which they always take for granted) in their network's DNS cache. In reality, the DNS server's cache data has been poisoned by the malicious actor and the request is being sent to a separate destination, instead of the legitimate service. These types of attacks fall under the **spoofing** category.

DNS spoofing defense

In order to avoid *DNS spoofing attacks*, you need to reconsider your organization's trust. You want to authenticate, you want to verify, and you want to make it as streamlined as possible. Cryptography protocols such as TLS/HTTPS and SSH help with preventing this activity by verifying the source, and source port randomization reduces the likelihood of a DNS spoof attempt via a race condition flaw being successful.

Additionally, DNSSEC or secure DNS is a method of using cryptographically signed certificates to prove the authenticity of DNS data. All of the original top-level domains (and many country-code TLDs) support DNSSEC.

Sniffing

Sniffing attacks, as we discussed in the *Sniffing tools* section of this chapter, is when network traffic is captured by utilizing a sniffer such as *Wireshark* or *Ettercap*. Wireshark is a commonly used tool in IT, with excellent features for diagnosing connectivity or web application issues. However, when a malicious actor utilizes a sniffer (providing the traffic is unencrypted), network traffic can easily be read, which could lead to credentials being captured and network exploitation.

Sniffing defense and detection

Typically, when we're looking at sniffing attacks or network monitoring, the mitigation is centered around utilizing the appropriate protocols that are protected with encryption. Rather than use HTTP, use HTTP over TLS. Rather than use FTP, use SFTP. Instead of Telnet, use SSH.

If you're forced to use the old/insecure protocols, you could potentially route them over a VPN and wrap them in an encrypted tunnel.

Additionally, to prevent malicious actors from being able to utilize these tools on your network, you may want to set up a prevention/notification for any time a known network monitor is installed or executed on a system. Furthermore, utilizing switches instead of hubs can prevent traffic from being sprayed all over your network, and mitigate against a non-admin user device becoming compromised.

DoS attacks

Have we covered these? I'm even starting to lose track here! A **DoS** attack is a way to overwhelm a system or network by creating more requests than it can process. It's achieved either through a flood of traffic, or though very clever timeout attacks.

Think back to what we learned about the three-way TCP connection handshake and how that works, while using Elon's joke:

```
You: "Hi, I'd like a beer." (SYN)
Barman: "You would like a beer?" (SYN-ACK)
You: "Yes, I would like a beer." (ACK)
```

So, after your initial `SYN`, what would happen if you never confirmed the server's `SYN-ACK` response with an `ACK`, but instead you opened another connection with the first SYN request again? And again, and again, and again? Well, the server waits for a response for a set amount of time before closing the connection, and eventually, you would make the server fill up their potential connections. This is called a **SYN flood**, or a **half-open attack**.

There are DoS attacks for the application layer, along with DoS attacks for the network, but generally, a DoS attack is successful when it causes a system that is serving legitimate users to stop doing so.

You might be targeted by a DoS attack for political purposes, chaotic reasons, blackmail, revenge, hacktivism, or because a group thinks it's funny, also known as *"for the lulz"*.

DDoS attacks

Distributed Denial-of-Service (DDoS) attacks are DoS attacks that come from various sources, making it very difficult to understand which traffic is legitimate and which traffic is part
of the botnet.

The number of nodes that constitute a DDoS rather than a DoS isn't as important as whether you are able to discern between the nodes that are legitimate users, and those that are contributing to the malicious traffic taking your network or systems down.

Previously, GitHub has been subject to a 1.35 terabit-per-second DDoS attack, which was achieved through *amplification* by "bouncing" traffic from a separate service (memcache, running on multiple servers in this instance) to the GitHub destination, creating up to 51,000 packets per source request. In order to get to 1.35 terabits-per-second, you're looking at around 130 million packets per second.

DoS defense

When we're looking at a DoS, we're generally looking at a single source of requests, so we could potentially do some of the following:

- Filter traffic from unwanted sources, such as those trying to DoS your systems.
- Increase the number of requests your servers can put into their backlog instead of handling them in real time.
- Reducing the timeout duration, for which your server waits for the final ACK response.
- If the server isn't able to handle many more requests before becoming overwhelmed, you can configure it to recycle the oldest request that hasn't been confirmed.
- You can increase the cache.
- You can utilize firewalls.

DDoS defense and detection

Similarly, when defending against a DoS, you want to prevent illicit DDoS traffic from reaching your server to begin with. In my previous example, the memcache DDoS at GitHub could be blocked by simply blocking port `11211` on an **access control list** (**ACL**) since port `11211` is the memcache source port, making for a simple mitigation.

However, how can your system discern legitimate traffic from illegitimate traffic when it's distributed, and not based on a pattern or from a port that can easily be blocked?

You want a system that can detect, classify, and respond to traffic requests that are seen as illegitimate, either through hardware or software solutions. For example, an application frontend hardware solution can be used along with routers and switches to analyze data packets and handle the traffic based on insights. This could include updating the ACL or rate limiting certain sources. Your IPS/IDS are also able to provide valuable insight into your network traffic, in order to identify real and illegitimate traffic and react with firewall rules in a reactive way.

Web application firewalls can identify legitimate traffic to your web app based on insights and intelligence from the vendors' other customers, and how often the request has been seen previously as a malicious source, not completing requests.

Something you would like to avoid is your cloud service, with elastic load balancing, from being vulnerable to a massive surge of traffic, thus leading to your wallet being heavily impacted by the upcoming cloud service provider bill. This is also known as a **Denial-of-Wallet** attack. There are various solutions for this that are generally offered by the cloud service providers themselves, and they can be part of your defense in depth strategy. It could be worth having a discussion with them, or reading their documentation on the topic.

Furthermore, there are DDoS protection providers, such as Cloudflare and Imperva, that provide CDN and threat intelligence solutions to help your organization withstand increased levels of traffic.

Typosquatting

Typosquatting is where you buy a domain that is a common typo for another domain, and configure the domain so that it points to your own server. In the past, this has happened to major companies, such as AirFrance being typosquatted by `www.arifrance.com` (note the misspelling of "air").

Typosquatting defense

Unfortunately, there are a limited number of defenses against typosquatting. You can preemptively purchase similar domains to your business in order to reduce the likelihood that a malicious actor can buy one, which is tricky enough to fool your user base. You can also file a case with the **World Intellectual Property Organization** (**WIPO**), to complain that your registered domain name is confusingly similar to the malicious actors' registered domain, and that the malicious actor has no legitimate interest in that domain. It's not ideal, but that's the situation.

Strategies for protecting network security

Now that we have gone through the TCP/IP suite and OSI model, network components and protocols, network devices and applications, some of the attacks, and how to defend against and detect those attacks, I would like to discuss some strategies you can use to actually protect your network and prevent malicious actors from having a field day in your estate.

Creating a policy

What we will want to do, in terms of a best practice, is define our strategy as a network security policy. This is a document that describes the rules that your organization will need to abide by in the realm of how the organization's environment is designed, maintained, protected, revised, and deprovisioned.

This document doesn't need to be complex, but can be as detailed as required to ensure the organization's network is effectively protected. How passwords or email attachments are handled in-transit can be included in this policy, or preferred encryption methods for various communications, if that's important to mitigating the risks inherent to your organization. Most importantly, you should highlight the change control process for your organization's network. If a change needs to be made, those responsible will need to abide by this policy.

In order to draft this policy, you will need a few things:

- Top management buy-in. There's no way you're going to successfully change the actions of individuals in your organization, and control their access to services online, without the blessing from top management. If you don't have this, handling any pushback will be impossible.
- Understanding your organization's requirements from a regulatory and compliance perspective, in order to effectively avoid compliance risks.
- Understanding your organization's network, information, and services. Without this, how can you create an effective and relevant policy?
- Understanding your users and their access requirements in their daily work. Yet again, you need this document to reflect your reality; otherwise, it's just a sheet of paper with nonsense written on it.

You will want to keep this document up to date, and update it any time any of the following occur:

- Technological changes
- Changes to regulatory requirements
- Changes to terminology
- Changes to your asset impact
- Changes to the threat landscape

Once these processes and policies have been established, you can execute your plan for implementing the appropriate controls to protect your organization.

Keep it simple

You don't want undue complexity in your network, and you don't want to make the security complex either. You might have a hybrid environment, where your on-premises and cloud assets are communicating with each other, or a network where the scope is unimaginable to our human brains… this is all absolutely fine, as long as you keep it as simple as possible.

What you need is visibility into your assets, clarity in your architecture and design, and a fantastic understanding of your organization's activities. You need an idea of what the core function of each asset is, and (for example) how new firewall rules may affect each asset. This should be documented, and optimizations should be continually designed, planned, implemented, and tested.

Business continuity and disaster recovery

As we've previously discussed, it's highly valuable to have a plan for when things go wrong. Regardless of the level of risk your organization faces, I'm willing to bet that it's important that you are able to restore business operations quickly and effectively in the event of a disruption. Disruption can often happen at the network level, so having the appropriate **Business Continuity Plan** (**BCP**) and **Disaster Recovery Plan** (**DRP**) in place that takes this into consideration is paramount to your organization's success.

Backup and restore procedures

We've mentioned this one before as well, but it's highly important for keeping your network alive: you need to have a plan and procedure for backing up, testing, and recovering the systems that perform your critical business functions, while focusing on risk mitigation and improving recovery times.

Insider threat mitigations/third-party threats

An insider threat has access to your network assets and can escalate their privileges more easily than an external threat can. In order to mitigate against these threats, either from intentional or accidental sources, we should consider employing methods from the Zero Trust approach.

Zero Trust

You can't expect that just because you have a firewall, a segregated network, and mitigations against sniffing, spoofing, or DNS hijacking that your network is safe and secure from threats that have gained access to (or naturally have access to) your internal estate.

If internal users always have elevated privileges, regardless of context, a compromised account in the wrong hands is going to cause major issues. As a result, we could move to a *least privilege* model, where we only allow a user, device, or account to access what it needs in order to complete the task specified. Unfortunately, that still doesn't fully mitigate against a compromised admin account.

What we'd be more interested in is a **Zero Trust** model, granting privilege based on context, such as the requester's identity, device, and location, along with least privilege access, and just-in-time access to mitigate the harm that can be done. Any request for elevated privileges can be accompanied with *Two-Factor Authentication (2FA)*, abiding by the "*Never trust, always verify*" mantra.

Additionally, we will want to recognize and report suspicious behavior and act on it automatically to reduce the impact of a compromise.

Rather than the classic "*castle-and-moat*" style network perimeter, where anybody inside the network is considered to be a trusted user, a major concentration point here is that we consider devices, both internal and external to the network perimeter, in the same way: with *Zero Trust*. We need proof, not faith. It's an ideology rather than a technology stack, and it can be achieved with some very basic rules.

Another ideology of Zero Trust networks is to microsegment your estate into several different secure "zones" for specific groups. If a user doesn't need to access certain data, then they don't have access to that zone. It requires an understanding of your organization's data, as well as a technological solution for preventing access. Data loss prevention tools can work together with labels to protect various types of sensitive and confidential information access by the wrong users, utilizing encryption and parsing the data of the file itself to detect known patterns that may indicate a file contains such information.

One more catchphrase from the Zero Trust school is to "*assume breach*". Essentially, you must assume that your network is compromised. This is how we mitigate against insider threats with Zero Trust: by reducing the ability for a compromised account to perform overarching actions across the entire estate. Utilizing segmentation of user privileges, along with the **two-man principle**, where you need more than one person's approval to be able to perform an action, and **segregation of duties** where a single action isn't completed by a single person from end-to-end, you are able to reduce the likelihood of a sole actor being able to silently make changes to your network.

The fantastic thing about this ideology is that it allows for a distributed team, with a less-defined network boundary, which is very relevant in the modern era where organizations are dealing with multi-cloud and on-premises hybrid environments, combined with users who are working from home and using their own devices.

Software and firmware updates

Patching your systems with security updates will have an enormous impact on the overall health of your estate, and will bolster your information security. Additionally, up-to-date systems are more agile and better at processing information. Yet, with all of these advantages, most organizations that I've worked with have major issues with their legacy systems; they're outdated, past the manufacturer's *end of life*, unable to be renewed due to incompatible software, or have an unknown level of risk in the event of an update.

As a result, it's highly important to define an **asset life cycle**, including the provisioning, updating, maintenance, and disposal processes. In addition to this, **vulnerability management** should be considered and defined, with a clear plan for identifying, classifying, remediating, and mitigating against known vulnerabilities in your estate. This is to ensure that, in 10 years, you don't end up with *legacy servers* running the equivalent of Windows XP or Windows Server 2003 today, which is extremely vulnerable to ransomware, highly incompatible, slow, and brittle. More information on how we can handle updates and vulnerabilities can be found in *Chapter 7, Owning Security Operations*.

Ensuring secure communication

The communication that occurs to and from your estate and inside your networks is vulnerable to being eavesdropped on and tampered with. Data encryption is a key part of ensuring your network is secure and protecting your organizational assets. *Data at rest* can be encrypted, as previously discussed, but encrypting *data in transit* should also be a key focus for security professionals looking to mitigate against eavesdropping and tampering threats.

Encryption is not only able to help protect the data by making it unreadable to unauthorized entities, but it can also confirm the *integrity* of the information, and also confirm the *identity* of the sender and recipient of a message.

The downside to this is that you need to trust that the entities managing the keys in the encryption process, such as the Certificate Authority, are secure. Performing **due diligence** and having a good **vendor assessment program** in place can help you understand the risk of this happening, and potentially cause you to reconsider the vendor you are currently using, in favor of one with a better security posture.

Another aspect of secure communication is that communications can occur outside of the realm of your estate. Your users may communicate with third parties by using a cloud service or text messages on their mobile device, and it's difficult to keep on top of this activity. Proper education in your organization about your policies regarding these practices can reduce the likelihood of an accidental breach. When it comes to malicious actors, you might want to think about a **Cloud Application Security Broker** (**CASB**) for your network, which blocks certain actions being performed on cloud web apps. It could, for example, prevent a file with a certain type of label from being uploaded to *Dropbox* or *Twitter*.

Cloud network security

What I am seeing more and more of is the transition toward the public cloud, which leads to relinquishing responsibility when it comes to the physical (and sometimes architectural) aspects of network security. Does the fact that you've outsourced your network security mean that you're not responsible for it anymore? Who is responsible?

The answer to that question is always that *top management is ultimately responsible for security*. As we've said previously, they have hired you to help them reduce their risk, and part of that undertaking is understanding the level of risk that exists, comparing that risk level against your organization's risk appetite, and mitigating it until the risk is below that line.

So, outsourcing physical and architectural aspects of network security to a **cloud service provider**, or **CSP**, has risks and opportunities, and it's your responsibility to understand those.

Often, using due diligence and performing vendor security assessments will help you understand how your suppliers are handling these aspects of security for your outsourced datacenters. In addition to understanding the security controls that the service provider has put into place, you can get a better understanding of the products that the providers offer, such as data encryption, monitoring and logs, access control, key vaults, and so on.

It's important to understand what falls under your organization's responsibility, rather than the CSP, in the cloud's **shared responsibilities model**.

Being able to quickly make changes to your infrastructure in the cloud means that change control processes may not be considered, leading to misconfiguration. It's pivotal to include the cloud in the wider scope of your network security policy's *Change Management* section. Changes that can be made very quickly have the potential to bring your network to its knees. Additionally, changes that have been made without oversight could include misconfigurations, leading to easy breaches from malicious actors.

Along with the change management policies comes proper testing and implementation of security controls for new or changed infrastructure before it goes live into production. Additionally, a deprovisioning procedure should be included to complete an instance's life cycle. It's important to keep an eye on this and control it, both in terms of policy and automation rules, otherwise your estate will continue to grow with more and more insecure instances, with old instances not being deprovisioned properly, giving easy access to confidential information.

Education and awareness

You might determine that educating your organization's users and administrators about how information is handled is necessary for success and risk mitigation. To do so, you can create an information security awareness policy, which will dictate the organization's requirements on this topic.

The requirements aren't set in stone, so you may want to ensure that all employees that access organization data must complete training courses provided by your organization on how to securely handle such data.

You might be able to educate users on phishing, and what constitutes **Personally Identifiable Information** (**PII**) or **Protected Health information** (**PHI**). You might also be able to create security champions, as we've mentioned previously.

Security Operations Center

Finally, we have a **Security Operations Center** (**SOC**), which is a team of security analysts, engineers, and managers that monitor the network for anomalies and issues and react accordingly, in real time.

Oftentimes, in order to do this, they must utilize SIEM systems and aggregate signals from all of the network hardware and software they can in order to detect outages or intrusions. This includes what we've talked about earlier in this chapter (and in the previous one) regarding IDS, IPS, firewalls, CDNs, antivirus, and vulnerability management systems all sending data to the centralized *single-pane-of-glass* observation solution.

Further reading on security monitoring best practices can be found in *Chapter 7, Owning Security Operations.*

Summary

What an absolute behemoth of a chapter! Are you okay? Do you need a second? I think I may…

It's so great to be finished with this chapter, having covered as much as we have. At the start of this chapter, we talked about the Internet Protocol suite and OSI model, before moving on to the various devices and applications we can use to set up a network, such as switches, routers, firewalls, and CDNs.

After that, we looked at some common network attacks, as well as how to defend against and detect them. After that, we talked about typosquatting, DNS hijacking, and MITM attacks, which helped us think in the way a malicious actor might.

From there, we looked at the strategies for protecting network security, including making policies, reducing complexity, business continuity, mitigating against insider threats and Zero Trust, updates, and cloud security, among other things.

Next, we will move on to a topic that I'm sure will float your boat: controlling access and managing identity. Let's get started!

5
Controlling Access and Managing Identity

Well, well, well… look who we have here. It's you! Not to mention the thirst for knowledge about controlling access and managing identity that you brought along. That's so great, and quite a coincidence, as this chapter actually covers those topics.

Identity and Access Management, or **IAM**, helps with understanding both the people and the automated services that are doing all of the **CRUD** (short for **Creating, Reading, Updating, and Deleting**) in your estate. That sounds important because it is important. How else can you ensure an entity is actually the approved user they claim to be?

Also, based on previous chapters, we have described certain levels of permission based on various aspects of privacy and confidentiality, but how is it that we can enforce those rules in an automated way in our digital environment? How do you walk the tightrope of the right amount of access? Too much access for your users and you've got a breach on your hands, but not enough access and your users aren't able to be productive in their day-to-day work. The aim of this chapter is to help you answer these questions for you and your organization.

Essentially, what I want from this chapter is to discuss the following bullet points:

- Access control models and concepts
- Selecting and implementing authentication and authorization mechanisms for people, devices, and services
- Identity and access management
- Controlling physical access to assets

I know how exciting this all sounds for you because just writing this is making the hair on the back of my neck stand up straight. So, with that said, enough stalling; let's just get started!

Access control models and concepts

When it comes to information security, the same idea applies from the physical security realm; *restricting access* to a location or asset is referred to as **access control**. When we say access, we could mean physically entering a space or digitally accessing a folder. We could mean reading a printed document in an office, but we can also consider the CRUD possibilities in a digital estate as well.

The classic (or retro?) access control is a **lock and key**. If somebody has the key, they're able to access what the lock is preventing access to. And if they have the key, they're allowed to access it, right? Is it that simple? Of course not. What we want is the ability to ensure that the people with the key are the people with permission, and try our best to ensure that even *with* the key, the wrong people aren't allowed to (or **authorized** to) access what they shouldn't.

Before diving into the models, I'd like to cover four key definitions first:

- A **subject** is a user or program that manipulates *objects*. Subjects can act.
- An **object** is passive data that is manipulated by *subjects*. Objects cannot act.
- **Clearance** is assigned to a *subject*, which dictates their access level, for instance, *Confidential*, *Secret*, or *Top Secret*.
- The **classification** or **confidentiality/integrity** level is assigned to an *object*, depending on its nature or sensitivity.

Now that those are covered, let's begin with the **state machine model**.

State machine model

The **state machine model** is a concept of preventing a system from becoming insecure by the continual monitoring of its status. All of the potential *states* of the system and the ways to transition from one state to another are defined and regulated, along with any other actions. This way, the current state of the system is able to be predicted and compared against.

A lot of these concepts are going to be a bit "wacky" to write out, and I think it might make more sense to use diagrams to help with conceptualizing their usefulness. Essentially, what a **state machine diagram** aims to do is to map out the various actions that can occur on a system, in order to prevent misuse:

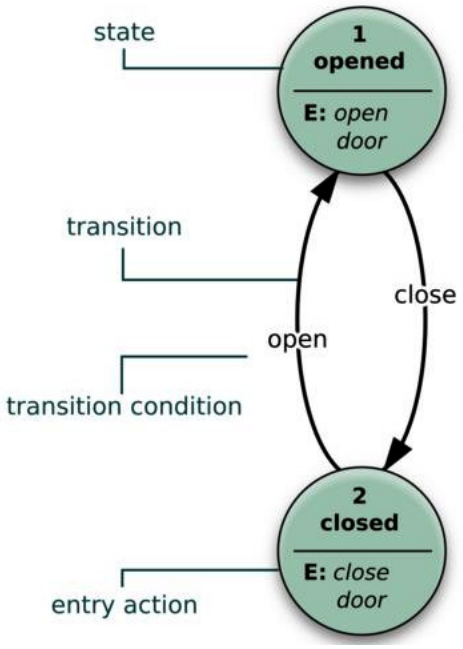


Figure 5.1 – State machine model diagram

(source: https://commons.wikimedia.org/wiki/File:Finite_state_machine_example_with_comments.svg)

In this example, we have a very simple system consisting of a single object, a door, that has two states, open and closed. The *transitions* that can occur are that a closed door can be opened and an opened door can be closed. These are triggered by the actions of *subjects*. Please, wake up, I can't have you fall asleep now; there are still three chapters after this one. Let's move on.

Information flow model

Similar to (and extending on from) the state machine model is the **information flow model**, consisting of **objects**, **state transitions**, and **flow policy states** (or **lattice states**). It aims to prevent any insecure or unauthorized communication, no matter the direction of the flow. This model lends to the **Biba** and **Bell-LaPadula** models, which will be covered later on in this section.

Confidentiality models

From the information flow and state machine models, we can transition into models that are focused primarily on confidentiality. Governments categorize information to be able to streamline the process of managing access to it, such as *Top Secret*, *Secret*, *Confidential*, and *Sensitive but Unclassified*. Let's look at how the Bell-LaPadula model, the Take-Grant model, and the Brewer and Nash model can help an organization manage access to information.

Bell-LaPadula model

By using **Mandatory Access Control** (**MAC**) to enforce a security policy across multiple levels, you enable the ability to manage access to information based on a subject's *need to know* and providing they meet or succeed the required clearance.

When considering the Bell-LaPadula model, you should keep in mind the following properties:

- **No read up**: Subjects aren't able to read information at a higher confidentially level than they have clearance to. This is pretty self-explanatory. In order to keep things confidential, you prevent people who are below a certain "level" from being able to know what's going on. It's standard and is appropriately called the **simple security property**.
- **No write down**: Subjects aren't able to produce or contribute information that is classified at a lower confidentially level than they have clearance for. This is to prevent information that is classified at a higher "level" from being "written down" to a lower confidentiality-level document, either explicitly or through inference.
- **No read up/down and no write up/down**: Also known as the *strong star * property*, this essentially enforces that in order to read or write to an object, the subject's clearance and the object's confidentiality must be equal. This is a strengthened version of the Bell-LaPadula model, which prevents the write-up operation from occurring.

Take-Grant model

The **Take-Grant model**, dating back to 1976, is a way to represent a system in a directed graph of nodes and connections, rather than a matrix, with the aim to easily be able to determine the safety of the system, even if it's complex.

The basics are as follows:

- *Subjects* can have the standard *read and write access rights*.
- *Subjects* can also have state transition rules associated with them, such as **take**, **grant**, **create**, and **remove**.
- A *subject* with the **take** right can take the rights of another *object* or *subject*.
- The **grant** rule allows a *subject* to grant their own rights to another *object* or *subject*.
- The **create** rule allows a subject to create new *objects*.
- The **revoke** rule allows a subject to remove rights it has over another *object*.

A Take-Grant diagram representation example is shown as follows. Keep in mind that **D** is a **directory** and **F** is a **file**, both being **objects**.

P1 and **P2** are **subjects**.

You might be able to decipher from the diagram that when a subject or an object has the **take** right for an object (or **t**), it can gain any rights the object has. If it has the **grant** right for an object (or **g**), it can pass any of its rights to that object.

See the following diagram for further investigation, which is exactly the point:

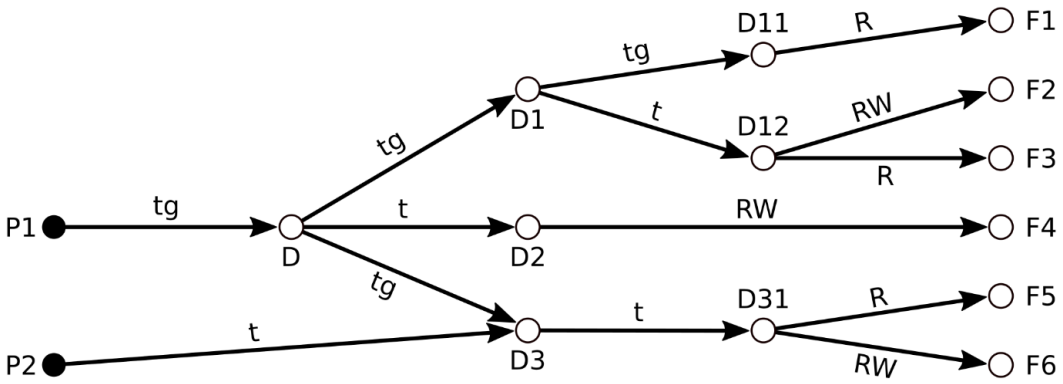


Figure 5.2 – A Take-Grant directed graph from https://commons.wikimedia.org/wiki/File:Take-grant_representation.svg

Now let's look at the Brewer and Nash model.

Brewer and Nash model

The **Brewer and Nash**, or **Chinese wall**, model is focused on segmentation. Essentially, we want to prevent *conflict of interest* by allowing access to the group of information that the user needs, but nothing more. The goal of this model is to ensure that no information is able to flow in a way that would allow a conflict of interest to arise.

It was initially created for the UK's finance sector and can be conceptualized by imagining a consultancy firm that provides services to several different companies. The employees at the consultancy firm who help with one client shouldn't have access to another client's information, as that could lead to issues around confidentiality.

Integrity models

We have covered integrity previously, so I'll spare us both from going into it again. Keep in mind that certain organizations might be more focused on integrity loss than confidentiality loss.

I'd like to go into the Biba and Clark-Wilson models, which address integrity.

Biba

Biba, or rather **the Biba model**, is conversely an access control model that focuses on addressing the concerns of *integrity*.

The Biba model was first published in 1977 and is another *lattice*-based model, like *Bell-LaPadula*. The defining properties of the Biba model for access control are as follows:

- **No read down**: Subjects are not able to read objects at a lower classification than their clearance. The opposite of the Bell-LaPadula's *no read up* principle, Biba's *simple integrity property* ensures that subjects that are at a higher rank aren't reading documents that may be "tainted" with misinformation.
- **No write up**: Subjects are not able to write to objects at a higher classification than their clearance. Also known as the *star * integrity property*, it prevents breaches to integrity in objects that shouldn't be "tainted" by contributions from below a certain clearance.
- **Invocation property**: This property aims to prevent a subject from invoking another subject who has higher clearance.

As you might have been able to tell, the Biba model doesn't address availability or confidentiality but focuses on integrity entirely.

Clark-Wilson

Next, we can speak about the **Clark-Wilson model**, dating back to 1987. Like most things from the 80s, it had commercial activities as the foundation of its creation. This model, like Biba, focuses on integrity.

The key takeaways from the Clark-Wilson access control model are the separation of duties principle and that data must be accessed through an application that allows logging, and the auditing of those logs is imperative.

Authorized users aren't able to change (or perform a **transformational procedure**, in this case) **constrained data items** in a way that is deemed inappropriate. The logged statuses for constrained data items are *tampered*, *logged*, and *consistent*.

In Clark-Wilson, users only have access to the data at their clearance level, not higher or lower, meaning each clearance level has its own set of data entirely.

With that, to prevent this section from becoming overly long, I'd like to move on to briefly discuss how this applies to the real world.

Real-world access control models

So, going back to our definitions from earlier, from a real-world perspective, what we're saying is the following:

- *Subjects* are entities that perform actions on the system. Some systems give *subjects* a **user ID**.
- *Objects* are the resources on the system. The *subjects* are potentially accessing them, and therefore access to *objects* should be controlled.
- Generally, we're looking at two ways to handle access control: **capability-based models**, and **access control list (ACL)-based models**.

In **capability-based models**, subjects that possess a **capability** to an object (similar to ownership) are the subjects that have access to an object. They can transfer that capability to another user.

In **ACL-based models**, subjects have access to an object if their ID is on a list of those with access to the object. If they don't, then "they're not on the list," and they get thrown into the street and a taxi splashes water all over their face and their cool red velvet suit is completely ruined.

Both model types have the ability to treat a **group of subjects** as a *subject*.

From there, let's dig further and investigate the real-world access control models that we see regularly:

- **Identity-Based Access Control** (**IBAC**) is based on individuals and allows a more granular but more difficult-to-manage approach to access control.
- **Lattice-Based Access Control** (**LBAC**) is a way of describing the rules for accessing an object or the rules for which objects a subject may access. For example, in a lattice approach, let's suppose you create rules that state the following:

  - If the *classification* of the *object* is less than or equal to the *clearance* of the *subject*, then they are allowed to read it.

  - If the *classification* of the *object* is less than the *clearance* of the *subject*, then they aren't able to write to it.

  That's *Bell-LaPadula* in practice. I knew it wouldn't be a complete and utter waste of time to describe those models earlier! No read up, no write down, BOOM!
- **Role-Based Access Control** (**RBAC**) is where we create clearance groups based on roles and put users into the various role groups. As a result, for access to *objects*, we eliminate the requirement for ad hoc decisions. If you have the role, you can access this *classification* of an *object*; if you don't, you can't.
- **Rule-Based Access Control** (**RAC**) is associating rules with access, such as dictating that an *object* is only available from 09:00 to 17:00 on working days.
- **MAC** is what we've been describing thus far, in the fact that we're using the *classification* of the *objects* to determine access, rather than allowing users to determine who gets access to what.
- **Discretionary Access Control** (**DAC**) is where a data owner determines which subjects can access specific objects. SysAdmins create a directory structure that has various permissions, and the appropriate data is stored in the appropriate location.
- **Attribute-Based Access Control** (**ABAC**) is where access is given to *subjects* through policies that check *subject* and *object* attributes.
- **Graph-Based Access Control** (**GBAC**) uses graphs and a query language to define access based on organizational diagrams, which sets it apart from RBAC, for example. Doesn't this diagram remind you of Take-Grant directed graphs?

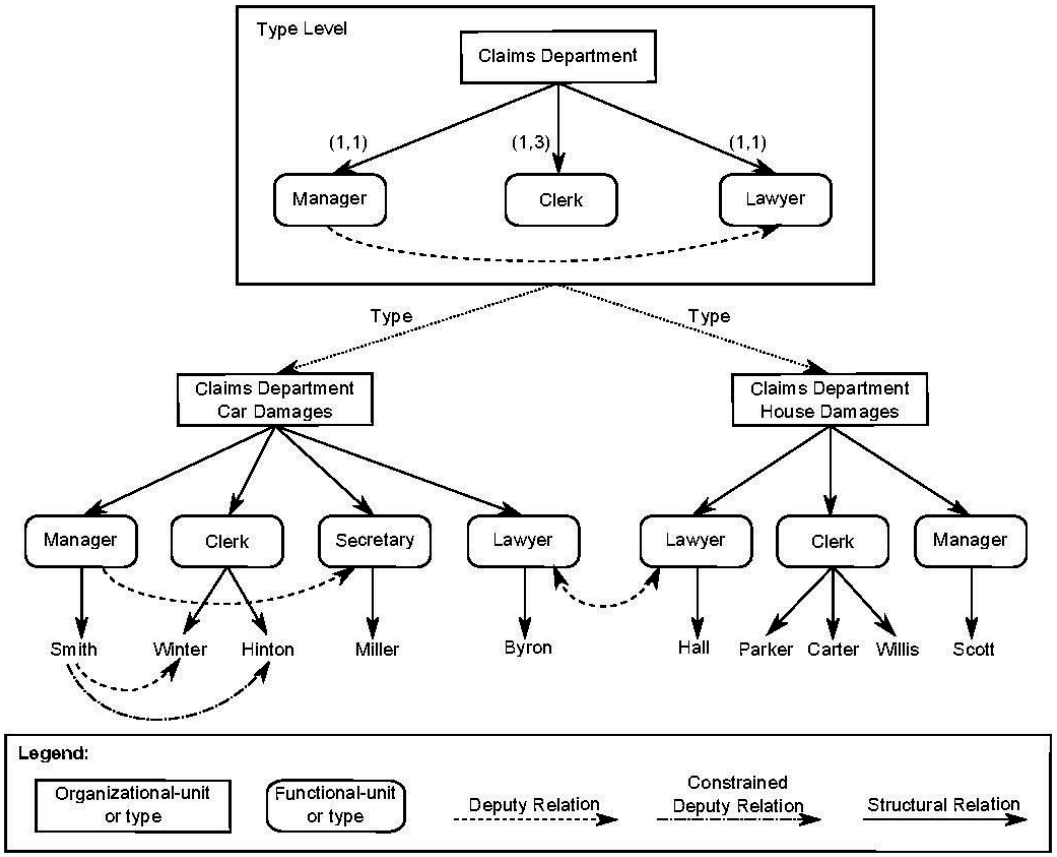


Figure 5.3 – A GBAC organizational graph (source: https://commons.wikimedia.org/wiki/File:GBACOrgGraph.pdf)

- **History-Based Access Control** (**HBAC**) is where access is determined after assessing the activities of the *subject*. This could include their behavior in the system, the amount of time between their requests for access, and the content of that request.
- **History of Presence-Based Access Control** (**HPBAC**) is an analytical approach to access control, taking into consideration the occasions for access. You might set a policy to state that users have access to an object if last week they have accessed the same object four or more times. It's not a popular choice, I'll be honest.

Now that we have those covered, let's delve further into the real world and talk about selecting and implementing authentication and authorization mechanisms.

Selecting and implementing authentication and authorization mechanisms

Thankfully, we have actually covered some of the aspects of this topic previously. Just as a refresher, I think we should touch on authentication versus authorization before moving on to how we might select and implement the various IAM solutions to ensure we keep unauthorized users from gaining access to resources they shouldn't, and ensuring that the authorized users are able to access what they need to get on with their miserable lives.

Authentication versus authorization

Authentication is focused on the identity of the user, and ensuring they are *authentic*. Not in the way that they compliment you and actually mean it, but rather that they are who they say they are.

Authorization is focused on what that user is allowed to do (or authorized to do) when they are *authenticated*.

A user can be *authenticated*, and due to their role as a data analyst have *authorization* to access the records they are currently working on. They might not have *authorization* for the new quantum levitation device that the company is currently prototyping in order to build the skateboard from Back to the Future: Part II, because the company is focused on confidentiality.

We are able to associate authorization clearances with the sensitivity labels or classification of the organization's data, which we have previously mentioned.

Authorization could be conceptualized as follows:

"If a document has this classification, then users with these clearances are authorized to read, but not write or delete".

Of course, these decisions are based on your organization's policies and vary heavily based on risk and requirements.

How is it possible to confirm the identity of the user? Where do we store these properties and how are we able to automate the process using computer systems? Those two questions lead into the next section beautifully.

Authentication and security

A crucial aspect of information security is **authentication**. It's at the center of many things we've covered thus far in this book. We want to allow known people to work on what they need to work on (but nothing more) without a hitch, but at the same time, we want to prevent unknown people from accessing anything they shouldn't.

There are many ways to tackle this issue, and for many solutions, the goal can be the same but apply to different components inside the network architecture or system.

Let's have a look at a few examples. We can, for example, authenticate a user through the following:

- **Password authentication**, an example of "*something you know*."
- **Smart card authentication**, an example of "*something you have*."
- **Biometric authentication**, also known as "*something you are*," an example being **fingerprint scanning**
- A combination of any of the preceding (or other) methods, known as **multi-factor authentication** (**MFA**)

When it comes to MFA, we need to remember that entering a password and then "the name of your first dog" is not an example of *MFA*. Those are two things that you know. It might strengthen the authentication process, but it's not truly MFA.

Password authentication

The advantages of **password authentication** are pretty obvious: We have trained people around the world to know how to authenticate themselves using passwords. It's easy for users (unless they're my mother-in-law). If they forget their password (like my mother-in-law does), then they can reset it simply by having access to their email account (which my mother-in-law has forgotten the password of).

Generally, this password is stored as a **salted hash**, and a service such as **Microsoft Active Directory** handles the secure storage of these credentials.

Passwords on their own need to be difficult to guess, and the number of attempts at guessing a password should be *rate-limited* or blocked, to prevent **dictionary attacks** or **brute-force attacks**.

Another mitigation against these attacks are **password complexity requirements**, where a password should not contain dictionary words, must have a certain number of characters, and must be a mix of letters, numbers, and symbols. Additionally, reusing the same password in multiple places allows for a breach in one service leading to account compromise elsewhere… but now we're canceling out the upside of passwords because they're supposed to be easy. **Password managers** give us a technological solution to this issue by automating the creation and utilization of complex passwords, provided a "master password" is entered, generally with MFA enabled.

Smart card authentication

By using a smart card, you're utilizing the "something you have" factor of authentication. The card has a chip that stores keys to identify a person and verify this information to the authentication process.

Generally, these cards also require the use of a PIN for access to be granted, which provides the second factor of "something you know." It's a bit like the way we used to use ATMs to take cash out, back when cash was a thing. Generally, the PIN is a very weak password, but the number of attempts is limited to prevent attacks surrounding complexity. The PIN requirement also prevents somebody from just *finding* a smart card and being granted access to your system.

Biometric authentication

Another form of *authentication* is through **biometrics**. Scanning your retina, listening to the sound of your voice, reading your fingerprint or palm veins, or dripping your blood into a witch's copper bowl, these sorts of things.

The advantages of biometrics are that there aren't any passwords to remember; you can't forget your fingerprint at home, and the likelihood of somebody having the same biometrics as another person is very slim – unless you are using something like Face ID on your iPhone, which occasionally doesn't work when it is actually you attempting to unlock your phone (**false negative**), or it works to unlock the screen of a similar-looking person (**false positive**).

The negatives of biometrics are also obvious:

- You can't change your fingerprint.
- If somebody needs your fingerprint, they might just chop off your finger (in this situation, you can only hope it's for a fingerprint biometric).
- They cost way more to implement than passwords do. Before phones came out, nobody had a fingerprint scanner in their home.

Single sign-on

Furthermore, entering in passwords is a bit of an annoying experience, especially if you're following best practices and using different passwords for each service, but not using a password manager.

Instead of prompting the user to enter in a password each time they try to use a different service, there are solutions that allow for a more unified approach to authentication, such as **single sign-on**, or **SSO**, where a user is prompted to prove their identity once, and then give access to all of the resources they need. This usually requires a bit of configuration but can lead to reduced risk from password reuse and easier revocation of privileges, among other things.

Authentication protocols

Thankfully, authentication in the digital estate has become somewhat standardized through a few different protocols. Over time, these protocols have been improved from a security perspective, and their compatibility has expanded to include more use cases.

As a result, it's worth going into the most common authentication methods and protocols that are utilized by organizations around the world currently, such as **NTLM**, **Kerberos**, and **PKI**.

Microsoft NTLM (NT LAN Manager)

NTLM, or **NT LAN Manager**, is a revisioning of how authenticating in Windows works and an upgrade from LM, or LAN Manager, which merely passed the (weakly hashed) password over the network to the domain controller, which stores the authentication information for each user. With NTLM, now the hashed password is never actually sent over the network, but rather a message that has been encrypted with a hash of the password as the key.

It is considered an outdated authentication protocol, but it is still widespread in IT systems because of its deep connection with **Windows**, **Exchange**, **Active Directory**, and **Windows Server** systems.

Here's a diagram of how NTLM works:

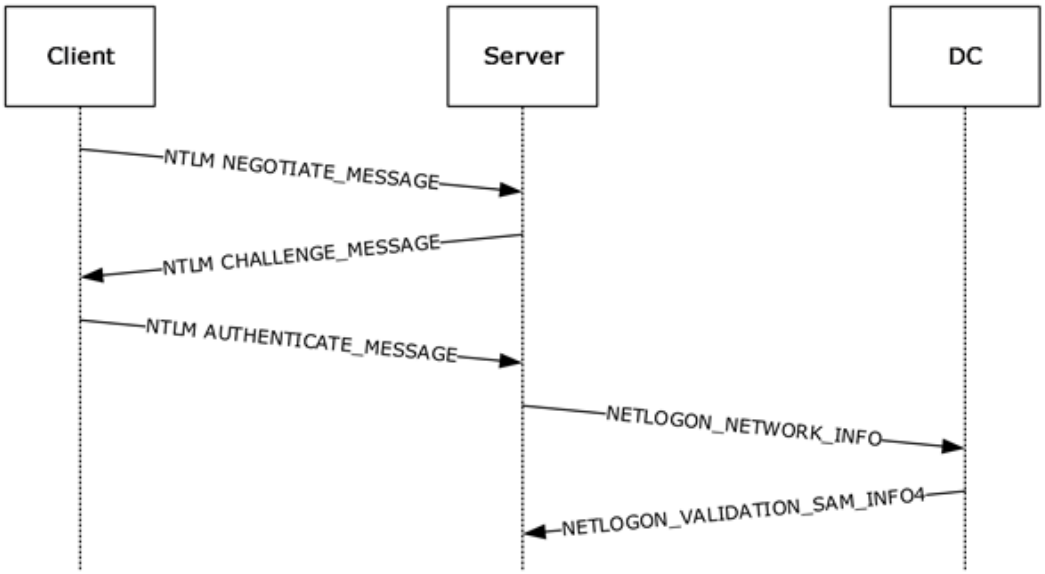


Figure 5.4 – NTLM authentication diagram
(source: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-apds/5bfd942e-7da5-494d-a640-f269a0e3cc5d)

To summarize, here are the steps:

1. A user signs in to the **client** PC, which sends a message to the **server** requesting authentication.
2. The **server** responds to the **client** with a challenge message as a response.
3. The **client** encrypts their password with a 56-bit segment of an MD4 hash of their password, encrypts the server's challenge with it, and sends it back to the **server** as a response. This is part of the weakness of the protocol.
4. The **server** passes the response through to the **domain controller**.
5. The **domain controller** checks its records for a match and notifies the server whether the user can be authenticated or not.

Because of flaws in the logic of NTLM, various issues have cropped up. For example, by relaying the NTLM `AUTHENTICATE_MESSAGE` to a rogue server with a dictionary, a **rainbow table** (pre-computed hashes of the most popular passwords), or the *hashcat* tool, they are able to quickly solve for username/password combinations. Currently, a modern GPU like the GTX 1060 is able to solve the NTLM for seven-character passwords in 1 second.

Even worse, malicious actors are able to simply sniff the network for the NTLM hash from another client, and then **pass-the-hash** and authenticate as that user, because the *hash* is **password-equivalent** thanks to a lack of *salting*.

Additionally, **Remote Code Execution** (**RCE**) is possible on machines that have NTLM enabled and access to administrative accounts on Exchange and Active Directory servers.

Unfortunately, NTLM authentication is still used for local log-on authentication, and for any system that is part of a **workgroup**. Many applications use NTLM for authentication instead of the more recent **Kerberos** protocol (it's been 20 years, people! COME ON ALREADY!).

If you find yourself with the requirement to keep NTLM enabled on machines and servers at your organization, there are a few mitigating steps you can take, but keep in mind that your team will be busy each time a new flaw is found, which is quite often.

You can enforce **SMB signing** to prevent **NTLM relay attacks**.

You can block the older version of **NTLMv1**, which is beyond saving.

You can enforce **LDAP/S signing** and **channel binding** to prevent **LDAP relay attacks**.

You can only accept requests with **Enhanced Protection for Authentication** (**EPA**) to prevent **NTLM relay attacks** on web servers.

Kerberos

To remedy many of these flaws from **NTLM**, **Kerberos** was introduced into **Windows 2000** onward for authentication. The three parts we'll include in this overview of Kerberos are the **client** (the user's PC, generally), the **server**, and the **Key Distribution Center** (**KDC**).

Before we get into the process, let's have a look at the diagram here:

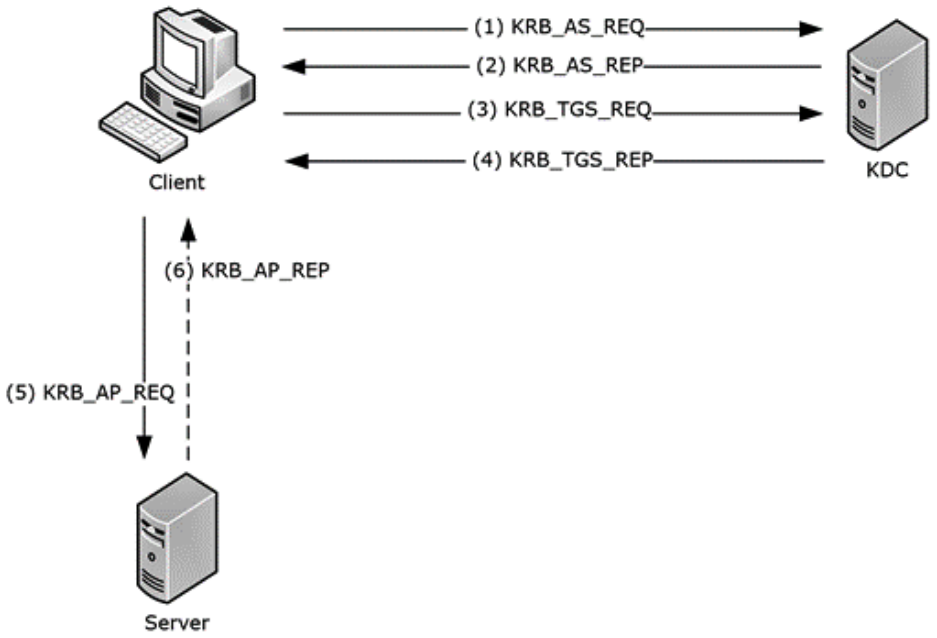


Figure 5.5 – A diagram of Kerberos authentication
(source: https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-kile/b4af186e-b2ff-43f9-b18e-eedb366abf13)

So, one of the first things we can see from this diagram is that the KDC is completely separate from the server, and there's no pass-through authentication happening as there was with NTLM.

Let's discuss what is occurring and how it's a secure method of authentication:

- **Kerberos Authentication Service exchange**: The **client** sends a request to the **KDC** for a **ticket-granting ticket** (yes, that's an incredibly awful name, let's instead call it the **TGT**). The client presents its principal name and can present pre-authentication information. This is represented on the diagram by **(1) KRB_AS_REQ**.

  The **KDC** returns a **TGT** and a **session key** that the client can use to encrypt and authenticate communication with the **KDC** for **Ticket-Granting Service** (**TGS**) requests, without reusing the persistent key. This is represented on the diagram by **(2) KRB_AS_REP**.

- **Kerberos TGS exchange**: The **client** then sends a request to the **KDC** for a **ticket** for the server. The client presents the **TGT**, a **Kerberos authenticator**, and the **Service Principal Name** (**SPN**), which is the name the client uses to identify a service for authentication purposes. A **Kerberos authenticator** is the client ID and a timestamp, encrypted with the **session key**, and helps the server detect **replay attacks** by proving that the **authenticator** is recently constructed. This is represented on the diagram by **(3) KRB_TGS_REQ**.

  The **KDC** validates the **TGT** and the **authenticator**. If these are valid, the **KDC** returns a **service ticket** encrypted with the **server**'s **long-term key** (which is already stored in the **KDC**) and a session key, which the client can use to encrypt communication with the **server**. This is represented on the diagram by **(4) KRB_TGS_REP**.
- **Kerberos client/server authentication protocol exchange**: From there, the **client** is able to request access to the **server** by providing the **service ticket** and a newly generated **authenticator**. The **server** then takes the **service ticket** and decrypts it with its long-term key, checks the **authenticator** is valid, and then uses the authorization data to dictate access control for the **client**.

To ensure the **server** is genuine, the client is optionally able to request verification by having the **server** send the **client's timestamp** extracted from the **authenticator** (which was previously encrypted with the session key). By returning the timestamp, the **client** has proof that the **server** can decrypt the **authenticator**. Otherwise, the server could be just gathering information without sending anything back, leaving the client in the dark.

PKI and digital certificates

A crucial part of **Public Key Infrastructure** (**PKI**) is **digital certificates** (or **public key certificates**), which prove the ownership of a *key*, which is tied to identity –of either a user or a device.

Certificate services are used in the **TLS protocol** (which protects data-in-transport on the web), as well as in email encryption schemes such as **S/MIME** and digital signature solutions. The advantage of **public key cryptography** (or **asymmetric cryptography**) is that, depending on which of your two keys you use to encrypt a message, you're able to ensure integrity, non-repudiation, and confidentiality.

One of the most popular standard formats for *public key certificates* is called **X.509**, which is split into several different parts because of the wide range of use cases for this service. Essentially, this certificate sits on a device and silently provides authentication to servers behind the scenes.

One major advantage of using an **X.509 certificate** is that we lose the requirement for a username and password combination, which we've talked about the weaknesses of previously. Using a *digital certificate* would ideally be done in combination with another factor of authentication, such as a PIN or an **MFA app** on the user's mobile phone that requires a fingerprint.

By acting as a **Certificate Authority** (**CA**) in your organization's PKI, your *systems administrator* is able to distribute and revoke digital certificates at scale and utilize those certificates to meet the organization's **access control policies**. This is compatible with existing systems such as Active Directory or LDAP and provides a much more streamlined way of managing authentication in an organization.

From here, let's move on to **authorization**.

Authorization

Often paired together with **authentication**, **authorization** is the level of access that is given to a **client** or **subject** after authentication. Authorization isn't always required to access a resource, for example, searching on Google. You don't need to authenticate in order to perform a search on Google (although you can), and anybody that can visit `google.com` in their browser has the authorization to use that resource.

However, in order to access your Facebook messages, or to be able to change your profile picture, you need to *authenticate*, and then you have *authorization* to create, read, update, and delete content that you have ownership of.

Essentially, computer systems allow a *programmatic decision-making process* to be implemented, which *provides authorization* based on information derived from a combination of the *authentication* process along with the *defined requirements for security and privacy*.

Focusing on *authorization* at your organization is a crucial step in ensuring the appropriate access is given to the appropriate people, and nothing more. Yes, we're talking about *least-privilege* again, and there's a reason for it. Least-privilege is a huge part of ensuring your organization is secure, and although sometimes implementing the principle of least-privilege seems like a lot of work, the benefits are generally able to recoup that cost very quickly.

We spoke about real-world access control models earlier. Leveraging them is a best practice in order to ensure you don't run into (or completely miss) the issues often seen in security breaches. Utilizing either **MAC**, **DAC**, **RBAC**, or **ABAC**, or a combination of those examples, is standard and seen often in information systems.

As an example, you might combine an ACL with RBAC to create a set of rules for each role in your organization, allowing access to view the resources associated with their role, and potentially allowing editing, updating, or deleting those resources, as required.

The best practice for managing these principles and ensuring your organization's users have the access they need, no more and no less, is to utilize an IAM tool suite. We are going to dive deeper into how to leverage those systems now.

Identity and access management (IAM)

When we're looking into how we can possibly manage all of the authentication and authorization principles that we've gone into so far in this chapter, we can simplify the process by leveraging what are known as IAM tools. These tools help organizations give the right access to the right resources at the right time and generally help with maintenance by giving each user a single identity that is then maintained and monitored throughout their time at the organization. This could include employees, customers, or third parties. If a user changes roles or leaves, that is reflected in their access through the *IAM*.

A *centrally managed identity repository system* for managing identity and regulating access leads to greater agility, better security, and higher levels of productivity, but requires technically experienced people to manage their administration. Furthermore, policies and procedures surrounding the regular administration and auditing of such systems are crucial to avoiding downtime and ensuring that security violations don't go unnoticed.

Keep in mind that when we're looking at compliance requirements of the likes of **HIPAA**, **SOX**, or **GDPR**, or almost any other regulation or standard for information security or privacy, we'll see a requirement for *controlling access* to employee and customer information. IAM tools provide a powerful solution to ensure your organization is complying with those requirements.

These systems are generally compatible with the previously mentioned protocols and processes, such as PKI, Kerberos, SSO, and password authentication, and allow for various implementation models, enabling users to work from home, authenticate to *SaaS* solutions from their mobile devices, and other non-traditional methods of working.

It would be pointless to discuss managing identity at an enterprise level and not mention **directories**, such as **Microsoft Active Directory**. Directories are different from IAM tools but are a crucial part of their functioning. IAM tools manage information coming from multiple sources, such as HR tools or Active Directory, for example.

Essentially, what we want from an IAM system is to automate the business processes surrounding the managing of these tools in your organization. Somebody new joins the company and a software solution for HR triggers the creation of a new user in the directory, which then triggers other actions based on what they might need on their first day.

Third-party identity services, including **Identity-as-a-Service** (**IDaaS**) cloud subscriptions, are increasingly popular and aim to reduce overhead and improve connectivity in this regard.

With the automation available to reduce manual IT management tasks, a few things are required in order to properly utilize these services. Let's take a look at a few.

Leveraging identity services

To begin with, *your organizational policies and procedures must be in place*. That's what I always say, and I always say it because it's always true. How could you possibly manage something as complex as the access control for your organization without writing down the rules?

These policies and procedures will aim to answer the following questions:

- How is your organization going to handle access control? Is the focus on a *role-based approach*? Are we looking at *MAC* instead?
- How are *new employees provisioned*? With each new employee, how do we define their access? What occurs to ensure they are able to access everything from their first day at work?
- Which roles or employees have which level of authorization? This is likely to be connected to your approach and whether it's *RBAC*, *MAC*, *DAC*, and so on.
- When an employee *leaves the organization*, what is required to ensure they don't have access anymore?
- If a user requires an *exception* to the rules, how is that escalated, approved, and administrated?
- When is access reviewed? How do we ensure the access to resources isn't growing with each role change for a user at your company?
- How far can automation go? Can we automate *password resets*? What about the revocation of privileges based on changes perceived in HR systems? For example, say a user hands in their resignation.

- If an external consultant joins the company or a third-party needs temporary access, how is that handled? Is *federated identity management* employed?
- Which services are covered by *SSO*?
- How do you teach new users how to use these systems? Is there a printed handout? A web portal? Is there a person at the company who sits with them? Create a normal starting procedure and try to walk through the steps of what life is like for a new starter, in order to test your plan.

Now that we have covered the concepts of IAM, I would like to move on to controlling physical access as part of our access control chapter.

Controlling physical access to assets

When we're looking at access control in information security, sometimes we neglect the physical aspect in favor of the digital one. *Physical access to hard drives, machines, folders, or printed documents* has the potential to be highly valuable to a malicious actor and should be controlled. In this section, I will briefly discuss various concepts of **physical security** to consider.

Physical access control

If we're looking at restricting access or entrance to a building or room and only allowing authorized individuals to enter the restricted area, we might look at a few solutions:

- **Human access control**, such as *guards*, *receptionists*, or *ticket-checkers*.
- **Mechanical access control**, such as *locks* and *keys*.
- **Technological access control**, such as *mantraps*, *turnstiles*, *fingerprint or retina scanners*, *fob-based access control*, or *exit barriers*.
- **Fences or perimeter barriers** may prevent individuals from being able to avoid the *access controls*.

Access control systems can help manage and monitor the access to areas and log who, when, and where the access was attempted, and whether it was accepted or not.

Physical locks and keys don't provide the ability to restrict access based on time or provide any records of entry. Additionally, there's little in the way of proving the keyholder is the individual who is supposed to have the key.

Additionally, it's difficult to revoke access in the event of a lost or stolen key, and it's impossible to know whether a key has been duplicated in secret.

In walks **electronic access control**, and just in time.

Electronic access control

The advantage of **electronic access control**, or **EAC**, is that we can solve some of the previously listed shortcomings of the lock and key combination… or the combination lock, for that matter.

With computers, we can do the following:

- Better restrict access by time
- Log entry attempts and successes
- Easily revoke access on a granular level, rather than having to replace the entire lock
- Gain other information to prove the individual has authorization rather than just the key

On the topic of gaining other information, that's when we can get into MFA, asking the user for two or more of the following factors of information for authentication:

- Something they know, which could be a password or PIN
- Something they have, which could be a key fob or smart card
- Something they are, which could be a retina or fingerprint scan

If a user is digitally accessing something, rather than physically, we might add one other "factor" type or "*somewhere they are*," which is a GPS location that proves the user is in a place deemed to prove their identity. For obvious reasons, that doesn't work for physical security access controls.

Preventing exploitation

When it comes to physical security, we could be looking at various risks, such as the following:

- Tailgating, when an unauthorized person follows an authorized one into a restricted area, sometimes being let in through politeness of holding a door open. This can be mitigated with mantraps and turnstiles.
- Destruction, such as crashing a car through a wall in order to gain access, or prying a door open with a prybar. Detection of this activity and defense-in-depth mitigations such as bollards or steel bars may be required for the risk of this to be reduced to an acceptable level.

- Technological attacks, such as cloning smart cards or brute-forcing pins, which can be mitigated through MFA, rate-limiting, and so on.

Among other things, it's worth considering the risks surrounding your physical access control and adding them to your risk register.

Summary

Holy cannoli! We managed to cover so much in this chapter; it's really worth patting yourself on the back.

We reviewed many of the various access control models and concepts, including classics such as *Bell-LaPadula* and *Biba*. Then, after those concepts were established, we looked at real-world examples, including *RBAC* and *DAC* models, and what those actually mean.

We then proceeded on to the topic of how we might select and implement authentication and authorization mechanisms for people, devices, and services. We covered the difference between *authentication* and *authorization*, and how they work together to provide *access control* for security and privacy purposes. We covered *passwords*, *smart cards*, and *biometrics*, as well as some of the most-used protocols for *authentication*.

Then we dove into what *IAM* is, and how you might utilize *identity services* at your organization to scale up and ensure your IT team isn't overwhelmed with menial access management tasks.

We dipped our toe into how to control *physical access to assets* as well, which is often overlooked in this topic.

With all of these topics covered, it looks to me as though you are better acquainted with the ideas that can help you to answer the questions we asked at the start of the chapter:

- How can you ensure an entity is actually the approved user they claim to be?
- How is it that we can enforce those rules in an automated way in our digital environment?
- How do you ensure the right number of permissions to enable productivity?

Since we've covered those points, I'd say we've reached a major milestone in this book! We have covered enough information security topics to finally get into the topic of *Chapter 6*, *Designing and Managing Security Testing Processes*. Without further ado, let's get started!

Section 3: Operationalizing Information Security

In this final section, we'll implement measures to ensure systems and software are secure, that visibility on security events is maintained, that reactions to security events are adequate, and that processes are continually improved.

This section contains the following chapters:

- *Chapter 6*, *Designing and Managing Security Testing Processes*
- *Chapter 7*, *Owning Security Operations*
- *Chapter 8*, *Improving the Security of Software*

6
Designing and Managing Security Testing Processes

Now that you have a good understanding of the controls that can be put into place so that you have a functioning security strategy in your organization, my guess is that you've implemented several of them, and now, you're free to just chill out, browse LinkedIn, and watch Netflix on your work computer for the next few years until your retirement. Congratulations! You've earned the gold watch.

Oh, you just got *pwned*. Whoops. How did that happen? Well, there was a huge gap that you didn't consider in your design, and you didn't have anybody else *sanity check* the architecture, nor did you have any internal or external team perform a *penetration test* on your environment. You want a second pair of eyes, and often, you'll want that second pair of eyes to be a technical wizard – one with no ties to your organization, who isn't concerned about offending their colleagues or bosses with negative observations.

You want answers. YOU WANT THE TRUTH! Do your security controls work? Are there ways around the controls? Does your staff find the implementation of the controls annoying or difficult to work with?

When I'm talking about InfoSec, it will often come back to adopting a mindset of continuous improvement, and that will, of course, include testing your existing implementations and improving on any findings for smoother-running, optimized information security programs. Oh, and of course, it goes without saying that you prioritize actions based on… Do you know what I'm going to say? I hope so: risk.

In this chapter, we're going to cover the following topics:

- Preparing for security assessments
- Understanding the different types of security assessments
- Best practices in performing security assessments
- Interpreting results from security assessments

By the end of this chapter, you will be able to understand security control testing, including scheduling for external and internal security audits, how to analyze reports and review the test results, and also how to manage the actions that arise from each assessment.

Let's get started! First of all, let's look at what we need to test, and how we are going to test them.

Preparing for security assessments

How do we know our systems are secure from attackers? How do we know that our systems are resilient? How do we know that our systems are appropriately redundant?

Unfortunately, the answer simply cannot be "*because I designed them myself*" (sunglasses on, standing on a Segway in the elevator). A second pair of eyes is important, at the very least. Why do I say, "at the very least"? Sometimes, your second pair of eyes is a colleague who either doesn't have the appropriate knowledge, or the appropriate segregation from the organization, in order to make the observations they might make if they didn't have a boss that was going to be offended by their input.

First of all, you need *buy-in from Top Management*. This is another classic theme in this book, and a classic theme in Information Security that you'll see in various standards. For example, in *ISO 27001*'s requirements (`https://www.iso.org/standard/54534.html`), *Clause 5* talks about *Leadership*, while *Clause 9.3* is about *Management Review*. *Clause 5* states that Top Management is required to be involved in creating the Information Security Policy. *Clause 9.3* is set on having Top Management be continually involved in evaluating your Information Security Management System to ensure that it is effective.

Like we've said previously, how is some crusty old person from Management going to do that, when they still haven't picked up on how to use formulas in Excel? It's pretty simple: they greenlight the spending and formally react to the findings, either greenlighting the expenses required to remediate or accepting the risks. Of course, as the Information Security Professional in your organization, you are going to have to translate the findings for Top Management to understand, and it's important to do so in a way that is easy to understand and highlights any risks.

Let's look at how we can choose the right assessment solutions, based on organizational requirements.

Defining your requirements

You have a plethora of information, systems, buildings, and other sensitive or confidential assets that you have implemented controls to protect. In order to say they're secure (and that the *CIA Triad* is intact), you need to ensure those controls are effective in preventing unauthorized access, unauthorized modification, and ensuring availability. Right? Right.

It's a simple thing to write, and less simple to execute, especially if your organization does a million different things. If you reach out to an Information Security consultancy, you might be lucky and get in touch with a salesperson who truly cares about your organization, and then refuses to sell assessments that aren't appropriate or required. That would be a great situation for you, but at the same time, it's important to ensure you know what you need before buying something, and also, how do you know that's the type of salesperson you're in touch with?

Additionally, do you want to pay a consultancy to assess your systems before you've had a chance to properly assess them internally? Not only does it sound like a waste of money, but it also seems slightly irresponsible to immediately hand it over to somebody else without any knowledge. I mean, what are we? Top Management?!

First, we want to refer to any of the standards or regulations that we need to abide by and document if there are any requirements for security assessments as a result of those. There are two reasons why I suggest this:

- First, we can accept that the standards and regulations exist for a reason and that they contain best practices for all organizations, including ours.
- Second, you stand to be fined, or lose your certification against the standard in a future audit if you haven't complied with the requirements. Sometimes, your organization has business operations and relationships that are tied to the maintenance of various standards, such as **Payment Card Industry Data Security Standard** (**PCD-DSS**), and being able to process card payments.

Sometimes, such as with PCI-DSS for example, standards have different assessment requirements based on the activities of your organization. For low-volume processing, you might only need to fill in a **Self-Assessment Questionnaire** (**SAQ**), rather than have a **Qualified Security Assessor** (**QSA**) evaluate your systems.

Let's look a bit closer at PCI-DSS's 12 high-level requirements as an example:

1. Implementing and maintaining effective firewalls
2. Changing all default passwords
3. Encrypting cardholder data at rest
4. Encrypting cardholder data in transit
5. Implementing and maintaining malware protection
6. Updating and patching systems against vulnerabilities
7. Access control for cardholder data
8. Access control for system components
9. Access control for physical access to cardholder data
10. Monitoring all access to cardholder data and network resources
11. Testing systems regularly
12. Maintaining an Information Security Policy for all employees

Now, let's imagine your organization requires a QSA due to the amount of *payment processing* they do. These QSAs are going to assess if your organization meets the requirements of these 12 high-level requirements (and the 220+ sub-requirements in total). If you aren't able to demonstrate these, your organization faces the risk of fines, or not being able to take certain types of payments.

When we're looking at the testing that we will need to do to prepare for the *PCI-DSS assessment*, and to ensure your organization complies with the standard before the QSA turns up at your door, it's important to schedule testing that documents the findings of each of these points. We do this to make sure our organization is able to take payments and doesn't get fined. We have a list of requirements, including *Requirement 11*, which is to *test all systems, processes, and software* to gain visibility on any vulnerabilities that can be exploited by malicious actors, and then applying the appropriate controls.

There are other reasons an organization might want to perform a security assessment on various systems in their organization, even if they don't have specific regulatory requirements to do so. This could include being unsure if your organization's intellectual property is appropriately protected, or to check that your implemented controls are effective in protecting against threats. Perhaps you think that the security of your organization is poor, but nobody is listening to you. In that circumstance, you likely don't have buy-in from Top Management, but might be able to get more attention and/or budget by having professionals demonstrate how a threat might exploit any vulnerabilities. The decision to do this isn't something I can advocate, but it could prove effective. It could also get you into hot water, so try your best to get that buy-in from whomever you report to.

When we're looking at which assessment we might choose to undertake, it depends on what your requirements are, and on your strategy. Some examples of things you might be able to define before making your decision could include the following:

- Why is the assessment being performed?

  — **Compliance**: Sometimes, organizations just need to tick a box. The reality for some consultants is that they perform compliance assessments for the vast majority of their work lives. If this is the main driver for your organization, then it might not be worthwhile to pay for a higher-value assessment.

  — **Striving for Improvement**: In this circumstance, the organization might be compliant with requirements, but wants to be better protected than they currently are. In this circumstance, you might be looking for a higher-value assessment, and therefore something a bit more expensive, or from a more well-known firm.

  Additionally, security assessments are useful in reducing costs through finding errors quickly, in a private setting that allows for the team to revise any findings, as well as protect them from any reputational damage that would occur from a breach.

- What is being tested? What is the scope?

  — **Software**: Do you want to see if the software your organization has created is secure from infiltration or misuse? Do you want to check if your fileshare software or operating system implementation is appropriately configured? There are various assessments for software, such as web app penetration tests, security scans, internal penetration tests, external penetration tests, and so on.

  — **Hardware**: Has your organization created or purchased a physical device, such as an ATM, lock system, or cryptocurrency wallet? You will want a hardware security tester to assess if it's secure enough to protect whatever it needs to protect.

  — **Networks**: Mentioned previously to some degree, you might want to see what a low-privilege user inside your organization would be able to achieve with their own credentials. You might want to see if somebody can enter your network and escalate their privileges from the outside world. Do you want to include social engineering in this? Do you want to test your physical security along with this?

  — **Employee Awareness and Physical Security**: Social engineering assessments such as phishing campaigns, which email users with fake tracking links, testing their ability to spot social engineering attacks.

  — **Physical Security**: Are you interested in a "Black Team," where security assessors physically (and digitally) infiltrate your premises? Do employees let strangers into restricted areas? How much information does somebody have access to in that circumstance?

- How much time will you allow the assessor?

  — Sometimes, a well-planned campaign could take months to properly plan and execute. What type of threat are you trying to simulate?

- When do you need the assessment completed by?

  — Information Security is booming, but there is a talent shortage currently, which could lead to delays in having an assessment be completed. Make sure to plan ahead of time, and speak to various businesses to see what their lead time is like.

- What is your budget for this particular assessment?

  — If you want a "Red Team" assessment, where a team of assessors plan for a long period of time and simulate a real threat over a long time period, but you only have $5,000 and no monitoring or "Blue Team," you might need to reconsider what you're looking for.

When you speak with various third parties that offer security assessments, it's worthwhile to ask them what their methodologies are, and if they have any references for the specific type of task you're looking to undertake. Additionally, it might be worthwhile to speak with the assessor(s) prior to the engagement and/or see their CVs.

Another important requirement is your terms. For one, you don't want the assessment to take down your *production systems* and cost your organization money in lost revenue. That's not good for anybody, and you want to work with a third party that is experienced in preventing that from happening. Furthermore, you don't want your weaknesses shared, especially not before you have the chance to remediate them. One way to prevent unauthorized sharing is to set up various terms, including **Non-Disclosure Agreements**, or **NDAs**.

Now that we have covered the high-level ideas surrounding the questions you might like to ask yourself and the members of your organization before getting involved with a security assessment, I think it's a good idea to move on to the types of security assessments we might see.

Understanding the different types of security assessments

There are various security assessments available that can help you understand any of the following:

- Your entire organization's security posture
- Your hardware product's security control effectiveness
- Your software product's security control effectiveness
- Your building's security
- Your staff's security awareness
- Your security staff's ability to respond to an active threat

Beyond that list, there are various methods in performing these types of assessments, including having your own staff members perform an **internal review**, or having another organization perform a **third-party review**. Furthermore, automated testing can be performed.

Internal reviews are effective in reducing costs, and in some circumstances, they can save time, providing the internal team is well-acquainted with the processes they are reviewing. However, internal reviews lack objectivity and may suffer from a lack of "fresh eyes" on the target.

Third-party reviews are performed by industry experts who break into software, hardware, buildings, and organization's information systems every day for a living. Their findings are generally provided in a fairly standardized report, which may also provide remediation actions and strategic advice for your organization on the next steps that can be taken. Additionally, third parties benefit from being honest and open with their customers about their flaws, providing it's not simply a compliance box-ticking assessment.

On the other hand, *third-party reviews* can be expensive, time-consuming, and can lack relevance if they aren't well-planned and well-defined in collaboration between the assessors and your organization.

Overall, security assessments have both upside and downside potential. They run the risk of costing the organization time, money, and reducing available capacity to create new value, but they provide the opportunity to shed light on security issues that could cost the organization more money in the long run through reputational, regulatory, or theft-related damages.

Let me define what each of the assessments might entail.

Automated assessments and scanning

In automated assessments and security scans, you are relying on software to detect *known security vulnerabilities* based on various pieces of information that's been gathered and compared against a vulnerability dataset. Let me briefly explain a few times where automated scans can be performed.

Web Application Vulnerability Scanners

Generally, **web application vulnerability scanners** can be pointed toward an IP or URL for your organization's web application and told to "scan" for vulnerabilities by interpreting *HTTP responses* to gain information about your software. Then, it can compare that information against a dataset of known vulnerabilities.

Beyond that, they may check for exposed files that could be used to access confidential information or escalate privileges, such as finding `/readme.html` in a WordPress site, which then reveals the WordPress version to the software.

Many of these scanners will include **fuzzers**, which check for **SQL injection** and **Cross-Site Scripting** (**XSS**) in order to detect previously unknown vulnerabilities. Fuzzers automate the initial process that a penetration tester (or malicious actor) would undertake to determine if a user input is vulnerable, and once that vulnerability is found, the tester would take time to modify and utilize the vulnerability to gain information or access.

Furthermore, if your web app allows login capabilities, most **web application vulnerability scanners** will allow you to provide credentials for authentication, which will simulate an attack from a malicious actor that is already "inside" the login barrier.

Many of the paid-for proprietary web app vulnerability scanners will have built-in "profiles" that are catered for different types of applications, in order to reduce scan time, or to simulate different degrees of attack.

Once the scanner has been configured properly, you can run the scan against your organization's web app, and upon completion, the software will generally package the findings in a report or web dashboard with explanations of each finding, vulnerability scores based on risk, and potential next steps for remediation.

If you have a web application facing the outside world, it's likely that malicious actors have already run automated scans against your server and application in order to check for vulnerabilities, often by one of the **open source tools** listed further down. It's difficult to **IP Blacklist** these attacks, as they're often undertaken from IP addresses of compromised machines, or simply don't follow a specific pattern. Activities such as these are occurring every moment, across the internet. These scans are from groups that are likely looking for "low-hanging fruit," so it's important to run these scans against your own apps and interpret if the results would lead to an opportunity. **Web application firewalls** are able to detect these activities and prevent too much information from being divulged, but your best defense in this regard is to ensure that security updates for your server's OS and web app's software packages are implemented promptly.

Here are some examples of web application vulnerability scanners, split into proprietary and open source options:

- **Proprietary**:

  a) Tenable.io/Nessus

  b) Rapid7 InsightAppSec

  c) Acunetix

  d) Detectify

  e) WP Scan

- **Open source**:

  a) Nikto

  b) OWASP **Zed Attack Proxy** (**ZAP**)

  c) w3af

  d) OpenVAS

If you are planning on setting up your own web application vulnerability scanner, either open source or from a vendor, you should consider a couple of common questions surrounding the implementation of vulnerability scanning activities.

Do you have the resources to maintain an internal system?

If you don't have dedicated staff who have maintaining internal vulnerability scanners in their job description, it might be useful to outsource that requirement to a third party who offers a managed service, maintaining and updating the scanner to ensure you have up-to-date and relevant results.

Another question worth asking is as follows:

Are you planning to scan against your production web application?

If your organization relies on your web app to make money and has SLAs with customers to ensure uptime, does it make sense to run a scan against it that has the potential to either slow it down or take it down?

An alternative solution is to set up a *pentest* or *scanning environment*, which is a *1:1 clone* of your production environment in terms of software but has a dummy dataset, usually running on completely separate (much less expensive) hardware. There are many advantages to this solution:

- You can **IP Whitelist** to the scanner's *IP range*, which is generally provided by the vendors in their documentation. If you have set up a web app vulnerability scanner yourself, you'll know the IPs. This gives you the ability to limit the amount of work the server will need to do, ensuring that the requirements for that environment are less expensive than the production environment, and therefore will cost significantly less money than a true replication would.
- You avoid the risk of taking your production systems down from a scan, because the test environment doesn't share any resources with the system that is making you money.
- You might have a **web application firewall** (**WAF**) on your production environment, but in your pentest environment, you are able to test your application specifically, rather than testing the WAF itself. In this circumstance, you can whitelist the scanner IP through the WAF, in order to pass any requests through without inspection.

It's a good idea to use different credentials and accounts on your testing environment, and to use a dummy dataset rather than your production dataset for privacy and confidentiality purposes. Additionally, it will save money to have a "pentest" dataset that occupies less storage than your production dataset does.

Network Vulnerability Scanners

Network vulnerability scanners are very similar in their functionality and operation to **web application vulnerability scanners**, but instead focus on your organization's network environment, including the devices and services used by your employees.

Many of the vulnerability scanner vendors are incorporating both web app and network scans into their solutions, offering an all-in-one or "single pane of glass" unified approach to automated vulnerability assessments.

Generally, what you will want from a network vulnerability scanner is for it to perform the following tasks:

- Discover both devices and software assets inside the network.
- Determine versions of software and operating system versions of devices.
- Check for known vulnerabilities from the information gathered.
- Report on and prioritize remediation steps based on risk/vulnerability scores.

Many solutions offer extra functionality, including the following:

- Dark web monitoring, to check if credentials for your organization have been detected on marketplaces
- Logging network events such as user activities and correlation with alarms
- An **Intrusion Detection Systems** (**IDS**) for **Network** (**NIDS**), **Hosts** (**HIDS**), and the cloud
- Endpoint detection, likely from network event heuristics

Essentially, what this means is that the **network vulnerability scanner** software is able to collect information about your internal network environment, and then determine if vulnerabilities exist that a malicious actor could exploit in order to enter your estate and subsequently "pivot" to escalate their privileges while inside the network, resulting in compromise.

If you're running outdated operating systems in your organization, such as *Windows 2008 or 2012 Server* or *Windows 7 Desktop*, your network vulnerability scanner is going to light up like a Christmas tree, and you're going to end up with a massive amount of work to complete in order to remediate these issues. That being said, it's much, much cheaper to run this scan than it is to pay thousands of dollars to a security firm, and then have a *penetration tester* own your network in a couple of hours. By remediating the findings in the network vulnerability scanner, you're *removing the low-hanging fruit* and increasing the amount of skill required in order to compromise your organization's network.

Some examples of network Vulnerability scanners, split into *proprietary* and *open source* options, could include the following:

- **Proprietary**:

  a) Microsoft Security Suite

  b) Tenable.io/Nessus

  c) Rapid7 InsightVM

  d) AT&T Cybersecurity USM Anywhere

  e) Acunetix

  f) Detectify

  g) WP Scan

- **Open source**:

  a) Nikto

  b) OWASP **Zed Attack Proxy** (**ZAP**)

  c) w3af

  d) OpenVAS

Further to doing vulnerability scans in your environment, it's also important to consider vulnerabilities in any software that your organization develops, so let's look at this scenario in more detail.

Software Development Life Cycle (SDLC) or DevSecOps Scanning

If your organization develops software, either for internal use, customer use, or use by the general public (including a website), then it is imperative to implement secure practices in the **Software Development Life Cycle** (**SDLC**).

There are many opportunities and stages in which vulnerabilities can be implemented in the process, and in *Chapter 8*, *Improving the Security of Software*, we will be going into how to improve the security of software and enforcing a secure software development life cycle. Until then, it's still worth looking at the scanners that can be implemented along the way to check for vulnerable code or software packages, including the following:

- **Static Application Security Testing** (**SAST**) tools scan through the source code of your software. They're usually either implemented in your developers' **Integrated Development Environment (IDE)** or set up to automatically scan new entries or pushes to your code repository, such as GitHub, BitBucket, or GitLab. By scanning the source code, the software might be able to find common pitfalls and mistakes that lead to software vulnerabilities, such as *XSS* and *SQL Injection*, after the app goes live. Common examples of SAST tools include GitLab's integrated SAST tool and *SonarQube / SonarCloud*.
- **Dynamic Application Security Testing** (**DAST**) tools are essentially the same as the previously discussed web application vulnerability scanners. They allow you to scan the software via information gathering and fuzzing after it is compiled and running. Usually, DAST is mentioned in relation to the software development life cycle, with the ideology that you can automatically trigger a scan of an ephemeral test environment for every new change that's made to the application source code by a developer, and then reject the change from being *merged* into a production codebase based on if it doesn't meet a baseline requirement for security, as determined by the results of the scan. Essentially, what I'm saying is that web application vulnerability scanners are generally able to be configured as DAST tools in the software's *CI/CD pipeline*, or on a regular basis.

  Examples of DAST tools include **Acunetix**, **Rapid7 InsightAppSec/AppSpider**, **Tenable.io/Nessus**, and OWASP's **Zed Attack Proxy** (**ZAP**).
- **Dependency scanning** tools allow you to scan dependencies brought in by software imports. Often, developers leverage software that has already been created by another party, which avoids the time-consuming development process and reduces the complexity of implementation, but also increases the likelihood of introducing *known vulnerabilities* into the software. Dependency scanners will check the imported modules and versions of those modules against a database of known vulnerabilities to ensure that your software isn't utilizing a vulnerable version of third-party code.

Examples of *dependency scanners* include the following:

a) **Snyk.io** assesses the imports for several languages, including JavaScript, Java, .NET, Python, GoLang, PHP, and others.

b) **Node Package Manager** (**NPM**) has some built-in dependency scanners for JavaScript.

c) **OWASP Dependency-Check project**.

Now that we've covered various methods of performing automated scans on your organization's digital estate, I think it makes sense to look into other *internal assessments* that can be performed by employees inside your organization.

Internal assessments

Internal assessments are those that can be performed without help from external (third-party) experts. Let's look at a few different types of internal assessments that you can undertake.

Employee security awareness assessments

We've spoken about giving our employees at our organization **security awareness tests** after training, to ensure that they have taken in the information provided, and as evidence of *due care* and *continual improvement* for your *compliance audits*. One method is to have a standardized test that the employees need to answer, in order to display knowledge in a question-and-answer format.

Another way is to leverage tools that can perform **phishing training** on your employees, for example, by sending emails that they have been trained to spot and recording metrics around their responses. This can also be done by third parties as a *managed service*, if the overheads of maintaining internal software and creating new templates for emails isn't manageable internally.

Compliance audits

If your organization is subject to regulations such as the **General Data Protection Regulation** (**GDPR**) or the **California Consumer Privacy Act** (**CCPA**), or standards such as *ISO27001* or PCI-DSS in order to perform its core business objectives and serve its customers, it's crucial to understand the effectiveness of your controls and find any gaps between the compliance requirements and reality.

To do so, **internal compliance audits** can be undertaken, as preparation for any future audits, by the regulating body or a third-party that performs the checks on their behalf. Simulating a third party, and investigating whether evidence exists on the effectiveness of a control implementation against requirements, is a great way to reduce the amount of time and effort required when you're actually paying expensive auditors to come and perform the assessment on your organization.

You must make sure you keep notes and ensure that all the requirements are understood by the appropriate members of staff, so that the appropriate controls are implemented and compliance is maintained. Software solutions such as *Microsoft Compliance Manager* exist to track, assign, schedule, and record observations around compliance to over 300 global regulations, with explanations and guidelines on implementation methods that could be effective in your organization.

Red and Blue Teams

If you have a well-qualified internal security team, you may be able to split the team into a **Red Team (offensive team)** and a **Blue Team (defensive team)**, to simulate a malicious actor's methods into attacking your organization, test the defensive team's ability to respond to those attacks, and mitigate the harm that may result from this.

Smaller organizations will have much more difficulty in this undertaking than mega-corporations with very large security teams and large budgets, but even large organizations may find that their team is better suited to undertaking defensive measures than offensive measures that have been mastered by penetration testers, all of which can be utilized through third-party engagement.

With that, let's look at the third-party assessments that can be undertaken by your organization.

Third-party assessments

When you want to assess if your control implementations are as effective as you estimated during your risk assessments or internal assessments, you will want to bring in a **third-party assessor**. When you want to test your *Blue Team's* effectiveness, you will want to bring in a third-party assessor. If you want to be able to show both prospective and current customers your security effectiveness, you will want to bring in a third-party assessor. When you want to comply with standards and regulations, you will need to bring in a third-party assessor.

There are many reasons why you may want to, or need to, bring in a third party. It boils down to two specific ideologies: *objectivity* and *skill*.

The penetration testers working at security firms get paid on a daily basis to "own" customer systems, both those that are highly advanced and "normal" to all extents. They use the tools the malicious actors will use, and they are efficient and skilled at offensive security. Furthermore, they don't need to appease anybody at your organization due to office politics or vested interests; they're objective, and their reports will be clear, generally standardized, and actionable.

There are a wide variety of assessments that can be undertaken by third parties, and new "flavors" are being created all the time. I'll offer a few examples.

Consider that we could (potentially) run any of the following security assessments as an internal review instead of having a third-party organization perform the test.

Technical assessments and penetration tests

Technical assessments and **penetration tests** can come in different forms, but generally, the focus of the assessment is to see if vulnerabilities can be exploited in order to gain access, and then escalate privileges until access to confidential or sensitive data is attained:

- **Web application penetration tests** are tests that are performed against a *web application* by a *penetration tester*, or multiple *pentesters*. It simulates a malicious actor that is trying to gain access and escalate privileges through functionalities and/or bugs in your organization's web application.
- **Network penetration tests** are tests that are performed against an organization's network estate by a *penetration tester*, or multiple *pentesters*. It simulates either a malicious actor that is trying to gain access and escalate privileges remotely without any initial access through security vulnerabilities, or an internal threat that has limited access, and then tries to escalate privileges and access confidential or sensitive information through security vulnerabilities.
- **Red Teams and Black Teams** are **threat simulations** that replicate a malicious actor attempting to infiltrate your organization by any means necessary. **Red Teams** are generally focused on Information Security, while **Black Teams** are generally focused on Physical Security.

  Both Red Teams and Black Teams take much more planning and *reconnaissance* work than a traditional penetration test, but they represent a much higher-quality test overall. These are valuable for organizations who have good Information Security practices and procedures in place, with proven control effectiveness and a high level of security awareness across their employees. Generally, a **Blue Team** will be aware of the engagement and stand ready to defend the estate, which takes planning and preparation from your organization's perspective.

Moving on from *penetration tests*, next, we will look at Risk Management and Governance Assessments.

Risk Management and Governance assessments

Moving away from the technical assessments, we have to the "boring" box-ticking world of compliance audits and Certifications Review. Here, business processes and policies are reviewed and tested for their effectiveness, with evidence requirements to satisfy the auditor. I've mentioned these audits before; they can include *PCI-DSS Compliance assessments* or *ISO27001 Certification assessments*, for example.

These assessments have the potential to carry great value, provided that (the standard clause) the auditor is skilled, and you are being open and honest with the assessor. These assessors can spot gaps in your processes, procedures, and identify risks that you might not have thought of, in a way that can be easily translated by business stakeholders in your organization.

Unfortunately, assessors can miss vulnerabilities and gaps if they aren't engaged for enough time, or don't have enough access to fully assess the company from start-to-finish on a regular basis. This is why it's important to be honest and open with the assessors, and ensure that your organization is getting their money's worth from the assessment.

Considering that third-party security assessments can be split into those two "styles", we can understand the following:

- An assessment that covers your entire organization's security posture is generally going to be a mix of technical and Risk Management assessments.
- An assessment of your hardware product's security control effectiveness is generally going to fall under a technical Assessment.
- The same goes for assessing your software product's security control effectiveness.
- Assessing your building's security can be done through either technical or Risk Management assessments, but generally, we'd be thinking "Black Team."
- A third-party assessment of your staff's security awareness could include phishing exercises or Red/Black team engagements.
- Your security staff's ability to respond to an active threat is handled through a Red Team assessment, where your staff acts as the Blue Team, defending your organization's estate.

Let's move on to the topic of best practices while performing security assessments.

Best practices in performing security assessments

Regardless of whether your organization is undertaking an internal assessment or has engaged a third party to assess its Information Security posture, there are a few best practices that are effective in ensuring that the value of the assessment being undertaken is maximized.

The key takeaways are as follows:

- Ensure the engagement has enough time to be completed thoroughly. There is no point in rushing, but additionally, it's important to ensure the test doesn't drag along and that you keep your valuable IT and security staff occupied on the assessment rather than their day-to-day requirements.
- Ensure the test scope is defined to avoid irrelevant assessments.
- Ensure the tester and internal staff assisting the tester are well-trained, prepared, and knowledgeable to increase the value of the assessment.

There are also some processes that can be utilized in order to effectively choose the appropriate assessment outside the tests and audits that are required purely from a compliance perspective.

These processes can include the following:

- Defining business requirements.
- Defining Information Security goals and strategies.
- Reviewing perceived strengths and weaknesses from an internal point of view.
- Performing risk assessments for assets, including known threats and vulnerabilities.
- Performing and documenting results from network and web application vulnerability scans.
- Reviewing and documenting the security control effectiveness of systems, including firewalls, VPNs, clients, access control mechanisms, and other information systems.
- Reviewing and documenting findings from access control lists and access logs.
- Reviewing and documenting the security awareness of staff members.

Once you've gathered as much information as you can about your organization (a lot of which has been performed during the risk management phases, which we've covered several times), you will be able to make risk-based decisions on the assets that you would like to be assessed, which helps stretch your budget further than if you were to simply test anything and everything fully.

Another quick note when it comes to best practices is something I mentioned earlier: I believe it's worthwhile to set up a dummy environment for automated scans and (some) pentests. It reduces the risk of taking down production systems, and also doesn't expose real-world customer data or confidential business information by utilizing dummy data instead. It's a cost-effective solution to mitigate against the risk of downtime as a result of security assessments. The trade-off is that some certification bodies, regulatory authorities, and customers will not see a test against a clone the same way as they would see a test against production, but those circumstances can be dealt with when they arise.

At the end of a third-party test, the assessor will take time to write a report and provide you with results. These results are what bring value to your organization, rather than it paying to tick a box and carry on working with an unnecessary level of risk. This is why honesty leads to better, actionable improvements. You're paying professionals to tell you where you're weak, so be honest and get real results instead of paying money to a third party and then misleading them from the reality of your processes and practices, which allows your organization to continue making the same mistakes repeatedly.

Let's have a look at the way these results are laid out.

Interpreting results from security assessments

When we're looking at the results from security assessments, there are two types, similar to how we split the third-party assessments in the previous section. These two types are technical assessments and Risk Management and Governance assessments.

Both types of reports will generally try to quantify the level of risk posed by each of the vulnerabilities found by assigning a score (either 1-5, 1-10, or some other scale) to them. It's important that you consider the value of your assets from your risk assessment proceedings, as sometimes a vulnerability could be seen as highly exploitable, but perhaps isn't worth mitigating, because it wouldn't stand to protect anything of value. In other words, the level of risk that's presented is below the risk acceptance level.

Often, the technical reports will include a narrative of how the penetration test was undertaken on a step-by-step basis, with screenshots, commands, and other information to help the customer replicate the findings and test their remediations. These are an invaluable contribution to the results as an Information Security Professional, as not only do they serve as tutorials for the IT staff in your organization, but they also help you conceptualize the threats to non-technical people and senior management. It gives a great first-person view into a malicious actor's mind and helps make the threat real.

When it comes to third-party reviews, regardless of whether the reports are technical or non-technical, they will generally include charts, graphs, and presentation material that will be delivered by the third party in a "findings presentation." It is important to include the company stakeholders in this meeting, as many people won't read the PDF report. Let's be honest: most people have thousands of unread emails in their inbox, so why would your boring PDF about how their business got absolutely *pwnd* catch their attention?

Beyond these findings, there will be suggestions for remediation steps. The value on these pages is incredible. You essentially have "the bad guy" telling you how you could stop them (or other people with their skillset) from getting in next time! So, as a result, my suggestion is to follow their guidance. You are going to have to plan how you can update and restructure systems, change passwords, and hire staff, so the first step is to ensure you have the resources and signoff required. Remember: the findings presentation is going to wake senior management up and serves as a great opportunity to ask for approval for that resource while the issue is burning in the back of their mind.

The same goes for *internal reviews*, including the results from *automated assessments and scanning*. You have metrics, as well as actions that can be taken to control the risk, and it's been quantified and presented in an attention-grabbing way in order to help you deliver your own findings meeting internally to senior management. Don't miss the opportunity to present the value of your *Information Security program* to your bosses – it's what they pay you for.

During the remediation phase, it's important to validate internally. It can serve as a beneficial opportunity to train IT and developer staff as to the dangers of security vulnerabilities, and it also improves the overall security awareness of your organization. Furthermore, internally validating remediations prevents embarrassment, because the last thing you want is to schedule a follow-up test with the third party, only to have them do the exact same thing again, thus wasting time, money, and resources.

This brings me to my final point: continual improvement. You want to schedule a follow-up to check that your remediations were effective, and then you'll want to schedule another test for a future date to see if the assessors can find other weaknesses.

Summary

Now that we've reached the end of this chapter, I know for a fact that you're able to understand security control testing, including scheduling for external and internal security audits, how to analyze the reports and review the test results, and also how to manage the actions that arise from each assessment.

You now know that there are various different security assessments you can undertake to understand how you can protect and improve your entire organization's security posture and your network and digital estate, from the hardware and software your organization has created to physical security and staff awareness.

Furthermore, you're now aware of the different ways you can perform these types of assessments, such as having your own staff members perform an *internal review*, or even getting another organization to perform a *third-party review.* Furthermore, automated assessments can be performed, and you know about several different methods of automated security testing.

Now that we've gone over how security assessments can help your organization improve its security and streamline its operation, I think it makes sense to move on to *Chapter 7*, *Owning Security Operations*.

7
Owning Security Operations

You've implemented your vision of an excellent, continuously improving information security strategy, with the appropriate controls to ensure your organization is secure and compliant. Nicely done! How are you going to stay on top of things? In the event of a security event or incident, how will you investigate? What is required to ensure we stay online, keep the organization's assets available, and enable people to work on their day-to-day tasks without interruption? This chapter is devoted to the never-ending task of security operations and how it aligns with the business' vision for the organization.

We'll look at the following topics in this chapter:

- Effective strategies in provisioning resources and maintaining assets
- Focusing on availability, disaster recovery, and business continuity activities
- Managing upgrades, patching, and applying security controls
- Investigating events and responding to incidents
- Implementing and utilizing detective controls
- Using security monitoring to improve visibility
- Security monitoring best practices

Let's get started! First of all, let's look at some effective strategies in provisioning resources and maintaining assets.

Effective strategies in provisioning resources and maintaining assets

There was a poll conducted by Automox in companies with between 500 and 25,000 employees, with over 500 IT and InfoSec professionals answering the questions (`https://www.darkreading.com/vulnerabilities---threats/missing-patches-misconfiguration-top-technical-breach-causes/d/d-id/1337410`). Automox found that over 80% had been breached over the past 2 years, and the following numbers show the vulnerability type that was exploited in those breaches:

- Missing operating system patches (30%)
- Missing application patches (28%)
- Operating system misconfiguration (27%)

The chain of events that occurs after a successful phishing attempt or credential harvest generally involves exploiting one of the preceding vulnerabilities. The brilliant thing is that we can prevent this with the provisioning of approved configurations and ensuring our assets are maintained, with both operating system and application patches and updates.

This section is all about how we can ensure that the assets in our estate are maintained and up to date for their entire life cycle. It's a continual process but saves time and effort and ensures the organization is innovating their offering instead of purely reacting to information security threats and racing to update their legacy software.

Provisioning resources

Within a digital estate, information systems are constantly being added, removed, and changed. Unfortunately, this is often done with very little oversight or without any guidance on best practices.

When we add new assets to our estate, what kind of process is involved? Do we "just do it," without a wider strategy defined? What kind of visibility do we have on our asset versions or operating systems?

Furthermore, in the era of cloud computing, with provisioning being as easy as a click of a couple of buttons in a web browser, how can we avoid **shadow IT** (the use of systems, devices, software, or other IT solutions without the approval of the IT department), and be sure that when employees at your organization "spin up a new server," our process is followed, and that our InfoSec strategy is considered?

Policies

As you've progressed through this book, you'll have become accustomed to my view on policies: the power of defining your approach in your information security policy documents is paramount. The same is true for the provisioning and maintenance of information systems. We need to be specific in guiding users toward the right answers. You don't necessarily need to define the "approved" versions of operating systems or which features should be disabled directly inside the policy document; in fact, I would recommend against doing so, as it will require continual updates in order to stay relevant.

Instead, you can point toward another resource that is updated continually, stating something along the lines of "approved operating system versions and configurations can be found here: <link>." Then, in the linked document, it's possible to update and cater the information to match your strategy, without the document upkeep, while ensuring everybody has the correct version of the information security policy available to them.

Configuration management

When we look at the aforementioned link that contains the "approved" configurations, how should that be handled? Where do we get the "good" configurations from, and how can we define what we need in our organization in order to ensure operations are maintained?

Thankfully, there are tools for **software configuration management**, or **SCM**. Popular choices include the following:

- Microsoft **System Center Configuration Manager** (**SCCM**), or **Configuration Manager**
- **SolarWinds Server Configuration Monitor**
- **Ansible/Ansible Tower**
- **Puppet**
- **Chef**

These tools provide various features, including the following:

- Remote management
- Patching
- Software distribution
- Operating system deployment
- Configuration management
- Maintaining software and hardware inventories of each asset

This can help you as an information security professional in several ways. Consider the fact that you now have a list of assets and the software used by each of those assets. This can help feed into your asset and risk registers and provide accurate real-time information about each asset, along with streamlining point-in-time assessments such as audits. With configuration management, we also ensure to document all changes, in order to easily troubleshoot misconfigurations, outages, performance degradations, and sources of non-compliance to standards, and increase the auditability of an estate.

Additionally, we have governance structures available to us in order to set baseline configurations and implement a change control process to ensure that any changes to production systems are done in a systemized, planned way. By implementing these governance controls, such as requiring that an administrator verifies the changes made to any system or network configuration before proceeding (*separation of duties*), you reduce the risk of misconfigurations leading to either downtime or security vulnerabilities.

There may be pushback from operational employees on setting up governance structures, with complaints such as "it goes against our way of working previously," or "it will slow us down." Keep in mind that from the organization's perspective, the time saved from having to restore systems, the money saved from not breaking your **SLAs**, the prevention of reputational damage, and the peace of mind to move ahead knowing that the estate is properly maintained with checks and balances in place will be worth the slight loss of velocity in making changes (if there even is a loss in velocity in the first place). Automation of the processes can help lift blockers and prevent the types of headaches many employees may predict.

When we look at effective use of configuration management tools, we should consider the following practices:

- Ensure to categorize and group systems into a taxonomy that makes sense to your organization. These categories and subcategories are able to be batch-changed, so ensure you and your fellow architects take that into consideration in the design.

- We need to define base configurations that are "approved" and document the reasoning and labeling for each of these configurations. There are organizations such as the **Center for Internet Security** (**CIS**) that offer "hardened images." It's likely that these operating system and application images are aligned with secure configuration guidelines, such as the CIS Benchmarks, enabling quick and easy deployment of secure systems across cloud and on-prem environments, without having to define the requirements yourself. Further reading can be found at `https://www.cisecurity.org/cis-hardened-images/`.
- When making changes to systems, all information about the change (user, time, changes made, systems affected) should be documented and tracked, and the process should employ an effective *separation of duties* principle to ensure breaking changes aren't able to be made by a sole individual.
- Ensure that there is a match between the testing and production environments, from operating systems, software versions, networking configurations, and so on. The better aligned your environments are, the less likely you are to see downtime or unnoticed vulnerabilities when the time comes for moving changes to production.
- It's also possible to automate the process of rolling back or remediation efforts in the event of a perceived fault.

With that said, many organizations are moving toward an idea called **Infrastructure as Code** (**IaC**), which closely aligns with the **DevOps** and **Continuous Integration/Continuous Deployment** (**CI/CD**) methodologies of IT management and software development. Let's have a look at what IaC offers.

Infrastructure as code

There is a concept called IaC that is a hugely beneficial approach from both a security perspective as well as an IT operations perspective. The basic idea of IaC is that your estate is moving more and more away from being physical servers in a specific location and toward a fully virtualized, distributed cloud environment. Therefore, instead of provisioning, managing, scaling, and de-provisioning physical hardware or utilizing the traditional, interactive configuration management tools that are catered to those types of hardware, your estate and all of the information processing facilities within can instead be defined in *machine-readable files*, which are *versioned* and updated the same way software code is updated, to be referenced by automation software in a hands-off provisioning/scaling/de-provisioning process.

This idea is tightly coupled with **DevOps** and **CI/CD** practices, rather than the more traditional split between IT operations and software developers. The aim of *IaC*, and most *DevOps* and *CI/CD* practices in general, is to increase the velocity of improvements and speed up and improve the efficiency of deployments, with higher-quality software being deployed, by reducing error-laden manual processes.

Before we touch on IaC tooling and formats, I would like to mention that IaC paradigms often involve leveraging containers and orchestration software, such as **Docker** and **Kubernetes**, respectively.

Docker builds *container images* based on the instructions defined inside a **Dockerfile**, which is a text file containing all of the commands required for building an image for a system (such as an application and its dependencies) as what is known as a **container**. Containers are a lightweight alternative to VMs, having been decoupled from the underlying infrastructure of the server it runs on. With that, one server can be running containers based on thousands of different configurations, with different operating systems and software packages in each. Further reading on how Docker works can be found here: `https://docs.docker.com/get-started/overview/`.

Further reading on how *Dockerfiles* are defined and written can be found here: `https://docs.docker.com/develop/develop-images/dockerfile_best-practices/`.

Orchestration software such as **Kubernetes** helps to automate the deployment of these *containers* across distributed environments, and aids in the operational aspects of deploying, managing, and scaling containers. The baseline configuration for the way the orchestration software will behave is usually defined in a **YAML** text file, with further changes being made through either a frontend web app interface or command-line interface tools. Further reading on Kubernetes can be found here: `https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/`.

What's the reason I wanted to briefly delve into these, you ask? I wanted to highlight the requirement from an information security perspective to keep on top of the way your organization operates. IT changes incredibly fast, and it's important to understand the concepts surrounding new tech in order to ensure the processes and solutions are appropriately secure. In this example, consideration should be made on how to ensure these tools and text files are checked for security concerns, as well as version-controlled. These new ways of packaging and deploying software are increasingly becoming the way your organization's estate is built, and without the proper governance and procedures defined by your organization, they could present new opportunities for malicious actors to exploit vulnerabilities.

When it comes to IaC, after creating the Dockerfiles/YAML required for packaging and deploying applications in your organization, you will need to ensure the infrastructure the software is deployed to is configured correctly. Enter **Continuous Configuration Automation** (**CCA**) tooling. CCA utilizes the principles of IaC and references IaC definition files in order to automate the configuration management aspects of IT we previously referenced.

You might ask, "Well, why did you go into configuration management tools earlier if you were just going to tell us that that is the old way of doing things, and there's this new, better way?!"

The grand thing is, you're generally looking at the same group of tools as before! Some popular CCA tools include the following:

- **Ansible/Ansible Tower**
- **Chef**
- **Puppet**
- **SaltStack**

Compared to the previous list, we can see that there is an overlap between the traditional configuration management tools and CCA tools, in Ansible, Chef, and Puppet, among others. These give a traditional "frontend" configuration dashboard, accessible via a browser, and provide the graphical inventory management and reporting we might expect and require from our traditional configuration management tools.

These *DevOps-focused* ideologies, tools, and solutions give developers and system administrators the opportunity to simply "declare" the infrastructure that is to be deployed inside their code, and automation tooling takes that declaration and then references the predefined "approved" blueprints, which can be configured to ensure compliance with security and regulatory requirements, (oftentimes called **policy as code**).

So, to the same extent that we're able to create our policies for humans to read, we're able to define those same policies into a *machine-readable format* and ensure the appropriate configuration is deployed every time, in an automated, error-free way. We now have an auditable, measured, predefined set of "approved" infrastructure and deployment scenarios, which are continually monitored for unauthorized changes, and any circumvention or noncompliance is torn down while simultaneously being replaced with a compliant version. Taking your risks into consideration, this is absolutely beautiful. Additionally, we've sorted many *integrity and availability* issues out in the process, providing the tooling is reliable and the solution is well-architected for your organization.

From a *confidentiality* perspective, we're able to automate the creation and declaration of super-strong secrets, tokens, passwords, and keys, which are never handled by individuals, by utilizing a **secrets management** automation tool such as **HashiCorp Vault**.

With *secrets management*, you are able to programmatically create, revoke, and rotate your organization's secrets in a dynamic way, responding to changes or requirements as set out by policy definitions or detected activity. By doing so, you can automate the efficient encryption of data at rest or in transit, with zero knowledge of the keys, and audit logs of all interactions. Furthermore, automation of the secrets management process can lead to a highly efficient *just-in-time* solution for authorization.

Of course, there are some downsides to the IaC CCA approach. Community repositories with pre-made container configuration files such as **Docker Hub** have hundreds of thousands of projects that have been left to "rot." These files contain out-of-date definitions, poor security practices, or back-doors and malware written into the definition on purpose, waiting for somebody who is in a rush (or unable to properly read and understand the file) to deploy a vulnerability into their estate.

Keep in mind that this isn't an edge case, either. The threat intelligence team at Palo Alto Networks previously discovered that 65% of the publicly disclosed cloud security incidents in 2019 were the result of misconfigurations by the organization's staff. When they investigated why that number was so high, they uncovered a culture inside the companies utilizing the DevOps methodology that values the velocity of creating new features over governance and security. Combined with that push for velocity, there sit hundreds of thousands of risky templates, available for free, in trusted providers' (such as Docker Hub's) open source, searchable repository libraries. Further reading on the topic can be found here: `https://www.paloaltonetworks.com/company/press/2020/palo-alto-networks-report-finds-poor-security-hygiene-leads-to-escalating-cloud-vulnerabilities`.

With that taken into consideration, we can agree that IaC doesn't remove the need to implement and follow change control and approval processes, even with a heightened level of speed and automation at the core of things. We have to remember that eliminating these governance steps increases the risk of implementing changes that introduce unknown risks to the organization.

A few appropriate ideologies for the processes and procedures in an IaC deployment pipeline could include the following:

- Ensuring you have employed the principle of *least-privilege* for your users. You don't want to give them the keys to the castle or the opportunity to turn off features in the pipeline in order to circumvent certain protections and checks along the way. Furthermore, there shouldn't be a way for your users to be able to gain access to the production server they have deployed their code to.
- Security checks and scans can be undertaken for the machine-readable text files that define the infrastructure at multiple stages of the development and deployment process:

  – DevOps engineers or developers can have their text files scanned right in their IDE, where they write their code, checking for known bad configurations and images.

  – The Git repository, such as GitHub, GitLab, or Bitbucket, can integrate with a security scanner to check for known misconfigurations or malicious items inside the files. Solutions such as **Snyk.io** can be involved in the CI/CD process to ensure the IaC definitions that are being utilized are secure.

  – Tools such as Snyk are able to scan the IaC configuration files to report on vulnerabilities such as the containers running in privileged mode/as a privileged user or not having a memory and CPU limit set.

  – These scans can also check to ensure the infrastructure is running at the most up-to-date and secure operating system and application versions.
- The IaC tooling itself should be maintained and we need to ensure that we're running the latest, most secure versions.

To summarize, with IaC, we are able to implement automated methods for reducing the risk of misconfiguration and outdated infrastructure, including pipeline-based security scans for infrastructure occurring through the change control stages.

With automation and the appropriate level of testing, you reduce the risk of errors by removing the human element from your deployments, updates, and changes. A good script will execute effectively every single time it's run and requires less hands-on time and effort from your employees.

These automated solutions should prevent infrastructure from being deployed in the event of an unacceptable level of risk being detected, and it's your job to determine that level of risk as suitable to your organization, and communicate that to the appropriate staff.

Additionally, with the appropriate level of monitoring and detection of changes, we can ensure systems recover predictably after an outage or similar event. If, for example, only a single server goes down, we can create rules to automatically deploy a mirror of that server in its place, instead of waiting for a staff member to notice the outage, diagnose the cause for the outage, and remediate the flaw.

On the topic of outages, I think it's about time to move on to talking about focusing on availability, disaster recovery, and business continuity.

Focusing on availability, disaster recovery, and business continuity

We need to be honest when we talk about information security; it's not always about defending against malicious hackers trying to grab our organization's precious intellectual property. In reality, many occasions you will face in your career will be surrounding an outage due to a power cut or a simple software update gone wrong, taking down a production system.

Just because these threats aren't adversarial, does not imply that they're any less real. The opposite may be true, in fact: we might face outages from non-adversarial events more often than we face malicious actors. Therefore, we need to ensure that we have effective change control policies in place and follow them, and we need to define and rehearse disaster recovery and business continuity processes in order to ensure maximum uptime.

Defining, implementing, and testing disaster recovery processes

Disaster recovery is all about establishing the appropriate documentation, planning, and practice required to ensure we can recover from either natural or human-caused disasters and restore critical business functions. It differs from business continuity in that business continuity is focused on ensuring all critical business functions remain online and function regardless of a disruptive event or disaster, while disaster recovery is about restoration.

For example, as I've discussed in previous chapters, your organization can face natural disasters such as floods, earthquakes, and tornadoes. We have pandemics and warfare activities ongoing globally at the time of writing. You might face a hazardous waste scenario, explosions, or even cyber-attacks that lead to your business either being taken offline or your personnel having to get away from a typical workspace.

For all of these disasters, we face the potential for loss. In 2015, the IT **Disaster Recovery Preparedness** (**DRP**) Council reported that small businesses could face up to $8,000 of loss from every hour of downtime. Mid-size companies in the same report faced a loss of up to $74,000 per hour, and large enterprises were reported to face up to $700,000 loss for every hour of downtime in the event of a disaster. Further reading can be found here: `https://www.researchgate.net/publication/343063858_PLANNING_FOR_DISASTER_RECOVERY_CAN_LEAD_TO_AVOIDING_DISASTER`.

In order to prepare the appropriate disaster recovery plans, we need some documentation that will seem remarkably familiar, as we have previously mentioned the same things over and over again in this book:

- Clear, straightforward playbooks that can be followed by anybody without distraction. This means clear labeling and ensured availability (even during an outage or offsite scenario).
- Policy documents that clearly state the roles and responsibilities for your organization, in addition to requirements from a business perspective, which could include compliance requirements from regulatory bodies or SLAs.

Once these documents have been created, it's priority *numero uno* to go over these plans – not only as a tabletop simulation but also as a real-life scenario, with actual evacuations and infrastructure and system restorations performed, in order to find any logical or technical gaps from the theory to the reality. If any gaps are found, the remediations must be planned and implemented, and then the test scenario should be recreated to establish the effectiveness of the changes.

Of course, this is all centered on risk. You focus on your most critical assets and systems and prioritize the restoration activities based on what is critical to the operation of your business.

Connected closely to disaster recovery, so much that I didn't know which one to speak about first, I would like to very briefly talk about business continuity, even though I've touched on it in previous chapters.

Managing business continuity design, planning, and testing

A closely related function to disaster recovery is business continuity. Business continuity is focused on ensuring all critical business functions remain online and functional regardless of a disruptive event or disaster. In a way, you might see that disaster recovery is merely a subsection of business continuity. Through establishing all the appropriate documentation, planning, training, and testing required, we're able to ensure our organization is resilient in the face of adversity or in the event of a disruption.

Something worth considering for both disaster recovery and business continuity is the ideas surrounding hot sites, warm sites, and cold sites.

Hot sites refer to active infrastructure serving as an available and fully redundant mirror of your company's production systems, decoupled from the original infrastructure to reduce the likelihood of both systems being taken down simultaneously. If you have a hot site ready, it's a matter of lifting and shifting operations over. This is your most expensive option, but the one that is most ready to be engaged in the event of a disruption.

Warm sites are a synced version of your production systems that are ready to be deployed at a moment's notice, with some delays expected in the provisioning process. The benefit of a warm site is that it's much less expensive to have prepared.

Cold sites can be utilized for non-critical business operations or if your organization either doesn't face the level of risk to require a hot or warm site or doesn't have the budget to maintain either. Cold sites require provisioning from the bottom-up, but still offer an opportunity to recover quickly from failure or disaster.

Implementing and managing physical security

As always, I'd like to briefly mention that physical security should be taken into consideration as an aspect of your responsibilities as an information security professional. Since the beginning of this book, we've talked about the risks facing your organization in the physical realm, from losses of availability due to severe weather or power outages to malicious actors physically breaching the perimeter and gaining access to confidential information.

Plans and procedures for the various physical security events must be defined, workshopped, and practiced in order to ensure that your team is able to respond appropriately. Documentation must be held in a place that is easily and readily accessible by any person who might find themselves responsible, and the instructions must be clear and concise, as heightened levels of stress can lead to misinterpretations and confusion.

As we have previously discussed various aspects of physical security throughout this book, I will keep this section concise and proceed to the summary of this section.

In this section, we covered ways we can focus on increasing our systems' availability through defining, implementing, and testing disaster recovery processes, as well as managing business continuity design, planning, and testing. Now that we've covered availability concerns, I would like to proceed with managing upgrades, patching, and applying security controls.

Managing upgrades, patching, and applying security controls

Your systems will need updating and patching continually. In a previous section, we looked at how IaC can help us with the automation of those processes, which is highly beneficial but not always practical.

Sometimes, we need to take a hands-on approach to provisioning, auditing, updating, and de-provisioning systems, for example, when we transition to newer technology or when we have findings from a penetration test that was undertaken by a third-party.

Additionally, in *Chapter 6*, *Designing and Managing Security Testing Processes*, we covered *web application vulnerability scanners*, *network vulnerability scanners*, *SAST/ DAST*, *dependency scanning*, and similar automated scans, which can provide actions into changes that need to be undertaken and upgrades that need to be performed.

The problem is, how do you ensure that the IT team understands the risks facing the organization and action the list of changes that need to be implemented in order to ensure a more secure estate?

Education

It's part of our responsibility as the information security experts at the organization to ensure that the appropriate people are aware and knowledgeable about the risks that they're able to take action against. This can include arranging for security training, but it could also mean scheduling a regular meeting on a weekly, fortnightly, or monthly basis with your IT team(s), in which you present information about a specific risk or pitfall or cover the overall risk posture facing the organization. Take time to explain and help them understand why it matters. This is a key part of your knowledge; otherwise, you're just the grump in the office that makes people waste their time on useless things and doesn't let them plug USB drives into their company laptops.

Furthermore, these sessions can help you in your understanding of the estate. You can be involved in a dialog where you get useful information about why it might be logical to apply a certain change before another one. You have a wealth of information available to you from the team that manages the day-to-day operations of your assets, and this dialog is incredibly valuable in the quest for continual improvement, which is not only an effective way to be proactive about your digital estate's health but is also a requirement from most of the standards and obligations your organization will be subject to.

This kind of knowledge-sharing and openness could be the difference between a successful change or a failure leading to a breach or an outage. With that, let's look at best practices for making changes.

Change control

As we've previously mentioned, it's crucial to follow the governance structures and **change control** processes that have been defined to reduce downtime and improve the ability of your organization's staff to diagnose outages or errors.

We could delve into this topic for thousands of pages, but I think it's more relevant to focus on security-specific areas of IT and allow you to investigate further into change control processes as a wider IT or organizational control. When we're looking at *change control* from an information security perspective, we're looking to avoid misconfigurations, maximize uptime and availability, and document any changes with the appropriate information.

Valuable steps in a change control process when patching, upgrading, or applying security controls will include the following:

- Identifying and describing the needed change in documentation
- Justifying the change and why it is required from a business perspective
- Estimating the impact that could be seen on the organization and documenting ways to mitigate any negative impact
- Describing a roll-back plan in the event of failure
- Documenting any approvals that are required and recording those approvals
- Planning for communication with staff, partners, customers, or similar parties
- Effective communication surrounding the planned changes, including the time window and any expected outages
- Implementation and evaluation of the changes

A great change control process doesn't need to be slow or prohibitive. It's a matter of ensuring we have a plan in place in order to reduce the likelihood of an outage, misconfiguration, or similar information security incident. By ensuring we have the appropriate documentation and planning, we're also ensuring we have all the appropriate resources available to us.

Large projects will benefit from a defined change control process, such as a security improvement program, for example, which I will briefly cover.

Security improvement program

Beyond the continual improvement efforts after a gap analysis or third-party/internal penetration test, it might be revealed that there are aspects of your information security program that are lacking the maturity or effectiveness that is required in terms of risk. A **security improvement program** is a project of defined information security improvements, which are scheduled, implemented, and tested for their effectiveness over a period of time, taking leaps and bounds toward being a more secure organization with more resilient operations.

Security improvement programs can cover as many topics as we've covered in this book and beyond, and are an organization-wide effort toward implementing and bolstering controls and reducing risk.

The idea behind a security improvement program is that we're not simply "sprinkling in" some security-based tasks into the daily, weekly, and monthly tasks that we're always focused on. It's an active decision to *de-prioritize new features* and instead focus on raising the level of protection and risk mitigation. Of course, decisions such as this require buy-in from senior management, which will require a defined plan, with estimates for time, cost, risks, and rewards.

Take your time in planning and estimating for this, because oftentimes it's a difficult decision for leadership to make. However, once you have that buy-in, you will be able to present the plan to the teams, which will help you with each step of your security improvement program, and making it known to those teams that this directive is coming "from the top" is a highly effective method in prioritizing these tasks and gaining buy-in from the boots on the ground.

Now that we've briefly spoken about provisioning resources, focusing on availability and business continuity, as well as managing upgrades, patching, and applying security controls, I believe we should move on to the next section, which is about investigating information security events and responding to information security incidents.

Investigating events and responding to incidents

I think you'd all agree when I say "by failing to prepare, you are preparing to fail" in terms of incident response and investigations. Without adequate response plans, playbooks, and documentation, we're destined to be scrambling without any proper direction when we face a breach, outage, or some other information security event.

We can have incredible software solutions to help identify malicious and risky activity inside our estate, but it's all pointless without an adequate plan for what to respond to and how to respond.

We should begin the process of defining our incident response plan by defining what constitutes the initiation for a response. What types of information security events will lead to a member of your organization investigating or responding?

Not all information security incidents are information security breaches, but all information security breaches are information security incidents and information security events. Let me take some definitions from ISO 27001:

- An **information security event** is any activity inside an estate or environment that is indicative of a security policy being violated or a control failing or an unpredicted scenario that has the potential to impact security.
- An **information security incident** is one or more **information security events** in which information security or business operations are compromised.
- **Information security non-compliance** is where a requirement from a policy, standard, or regulation is not fulfilled.

So, an **information security event** has the potential to affect risk, even momentarily. An **information security incident** is something we know has negatively affected operations or security, such as an outage or confirmed breach, and **information security non-compliance** is finding that a control hasn't been implemented correctly or a policy is being disregarded in practice.

Allow me to cover an example of an information security event leading to an information security incident:

- **Event**: A user at your organization reaches out and says they can't open any of their files and has a notice on their screen.
- **Incident**: After investigation, your team discovers that the user's laptop has been infected with ransomware. As a result, the incident response playbook for ransomware is initiated and the appropriate team is engaged to respond.

- **Non-compliance**: After investigation and remediation, it was discovered by the incident response team that the staff member opened an email attachment from an unknown party and that an outdated SMB service wasn't disabled on some machines, due to an oversight by the IT team.
- **Actions**: Phishing and information security training for staff members has been prioritized and IT has added a roadmap item to implement secure configurations for workstations, including disabling outdated SMB versions.

As you can see, if we use these definitions for events and incidents, while taking our risks into consideration, we're able to create procedures for investigating specific types of events and responding to incidents in a uniform and effective fashion. To do so is about defining your incident response plans, which I would like to go into next.

Defining your incident response plans

In order to understand the workflow and requirements, it's integral to create an incident response plan that suits your organization. If you don't create a feasible response to the alerts generated by your organization's **Security Information and Event Management** (**SIEM**) solution, for example, you've essentially wasted money on a tool by not being able to derive value from its use.

I would like to take this opportunity to discuss what might be some key questions that need answers before you begin your security monitoring activities.

Some questions we would like to answer can include the following:

- *What are the defined roles relating to information security events?*

  Some examples of roles may be the CEO, CISO, CTO, MD, SysAdmin, HR manager, and so on.
- *What are the duties of each role, for each type of event?*

  Some members may be required to communicate with customers, others with law enforcement, customers, or business stakeholders. There may be roles for investigating the event; there may be roles for restoring processes. Tying the roles to specific skillsets is a beneficial solution, and tying those roles to specific job titles and job descriptions allows an approach that makes individuals aware of their responsibilities when they move into a new role.

- *What is the method for prioritizing security events?*

  Priority should be based on risk, but it's up to you to define these formally in a way that is easily followed by anybody who may find themselves in a situation where they are the "first responder." Educating staff on how to respond to security events and where the documentation may lie is part of incident response, which we have previously touched on in other chapters.

- *What is the method for documenting security events?*

  How we document security events from discovery to closure is another facet of incident response that we need to define and train our staff effectively for.

- *What is the method for notifying the appropriate members of the incident response team of an event?*

  Some examples may include email, phone, SMS, or WhatsApp.

When we define a process, we will want to consider guidance for the following steps:

- Investigating an event, including what information to observe and record
- Confirming an incident, including communications and next steps for the appropriate response
- Responding to incidents, which includes isolating or containing the threat and reducing the damage that can be done to the organization
- Restoring operations, ensuring the impact of the incident is minimized to the lowest level possible
- Investigating and reporting, potentially using forensic capabilities, and taking actions to protect against the perpetrator or taking legal action (if applicable)

If we want a simple set of guidelines for how to respond to information security incidents, we might want to see the following few guides, provided by various governmental organizations:

- The UK **National Cyber Security Centre** (**NCSC**)'s *Small Business Guide: Response & Recovery* (`https://www.ncsc.gov.uk/collection/small-business-guidance--response-and-recovery`)
- The European Network and Information Security Agency's *Good Practice Guide for Incident Management* (`https://www.enisa.europa.eu/publications/good-practice-guide-for-incident-management`)
- The National Institute of Standards and Technology's *800-61 Rev. 2* (`https://csrc.nist.gov/publications/detail/sp/800-61/rev-2/final`)

You can also leverage help from an information security consultancy company that offers incident response services. Oftentimes you will pay in advance in order for these organizations to be ready to respond in the event of an incident, and the amount of time utilized from each incident is deducted from your "balance."

Furthermore, it's important to note that support is available from various governmental departments, depending on the country your organization is operating in.

Other organizations might feel comfortable with and capable of performing the investigations and responses themselves, without external help. Let's look at a few points surrounding performing security investigations next.

Performing security investigations

I'm going to use this section to discuss how we can manage and conduct security investigations on incidents, and how we might go about reporting incidents. By using a consistent, structured approach, we can ensure a systematic and effective approach to most circumstances.

To begin with, let's remind ourselves that not all security incidents are significant enough to require an investigation. It's important to document individual events or combined events that have led to an incident, but we don't necessarily need to escalate every documented occurrence and engage the wider organization or contact law-enforcement. Guidance for what you should report can be found in your local regulations or in resources provided to your organizations from police, national defense forces, cybersecurity departments such as the UK's NCSC, and other similar agencies.

As an information security professional, it's your responsibility to do the following:

- Identify what constitutes a **minor incident** and a **major incident**.
- Identify the responses to minor incidents and major incidents.
- Ensure all national or regional laws and regulations are complied with, including ensuring that incidents are reported to the appropriate law enforcement parties.

Some examples of a *major incident* could include the following:

- Crimes such as burglary or damage to organizational property
- Floods, storms, and other weather damage that leads to compromised security or operations
- Unauthorized access to information that is classified as confidential or protected

Oftentimes, a major incident will require contacting insurance providers, law enforcement, governmental agencies, customers, and other groups. It's important to define and maintain a procedure for these situations and keep them accessible, most likely inside your playbooks and policies for incident response and disaster response.

Along with the processes, it is important to define roles related to security investigations. Some examples for roles and their responsibilities could include the following:

- The *CEO* and *C-suite*, who are ultimately responsible for responding to security incidents. This generally includes ensuring there are adequate processes for staff and contractors to report and respond to such incidents, as well as ensuring there are records of past performance and requirements. This is generally done by appointing the appropriate members of staff to focus on such efforts.
- The *CISO*, who is the person appointed by the C-suite to handle the efforts mentioned previously, and is responsible for the following:

  – Handling processes related to information security incidents

  – Managing information about security incidents, including the outcomes from internal investigations as well as investigations from law enforcement

  – Reporting that information to the C-suite and the senior management team(s)
- *Senior management* team members will be leveraged in the incident response process and will utilize their team of staff members in the event of a security incident to perform their relevant duties. This could include the head of legal preparing information relating to litigation, the head of HR responding to insider risks, the head of communications preparing statements for the public, and so on. They will generally also be responsible for an area of reporting related to their departments and presenting that information on a regular basis to the CISO.
- *Managers* have a closer view of what is happening at the operational level of business activity. They ensure that security incidents are appropriately reported and that employees are prepared for their responsibilities in the event of a security incident requiring a response.
- *Employees* are the group that is most likely to notice an information security event or incident. They should be required to understand their responsibilities and the processes they have available to them in the event of detecting a security incident. This can be achieved through education and training.

I would like to suggest reviewing a document that the international not-for-profit body CREST released. It's a guidance for incident response processes called the *Cyber Security Incident Response Guide* (`https://www.crest-approved.org/wp-content/uploads/2014/11/CSIR-Procurement-Guide.pdf`).

Inside the linked PDF guide, you will find guidance on the following:

- Understanding information security incidents, including the following:
  - How to define an incident
  - How various incidents compare to one another
  - The phases we would expect from an attack
- Preparing for information security incidents, including the various challenges we may see in an incident response undertaking, ways to respond, how to leverage experts in response, and how to build an internal response capability. Beyond that, the document discusses five steps in preparing your organization:
  - Conducting a criticality assessment for your organization
  - Carrying out threat analysis
  - Considering the implications for your organization, including people, processes, and technologies required
  - Creating a secure environment, including implementing the basic controls required to prevent the most common types of incidents from ever happening
  - Reviewing your posture and testing your readiness for incident response
- Responding to information security incidents, including the following:
  - Identifying incidents
  - Defining objectives
  - Taking action to contain the incident
  - Recovering systems
- Following on from information security incidents, including the following:
  - Investigating the incident further
  - Reporting the incident to stakeholders
  - Carrying out a post-incident review of the investigation
  - Communicating findings and building on findings
  - Updating information and controls based on the findings
  - Performing analysis to uncover trends
- Finally, choosing suppliers to help with handling information security incidents.

This document will serve as a great resource to you and your organization in the future, and I hope you take my advice and give it a read.

Now that we've touched on investigating events and responding to incidents, I would like to go into the implementation and utilization of detective controls in your organization.

Implementing and utilizing detective controls

As is the nature of IT, we have a constantly mutating landscape of technologies, threats, and techniques to be able to identify and protect against, including ransomware, DDoS, and infiltration attacks. We need to know what is happening in our estate and implement automated controls for notification in the event of an attack, as well as the active prevention of those threats from being exploited.

In order to achieve this level of visibility, we need a set of processes and structures surrounding **security monitoring** and **security investigations**, and we need to test those processes with staff members and third parties that are responsible for protecting our estate.

Ultimately, senior management is responsible for allocating the appropriate resources toward an internal **Security Operations Center** (**SOC**) or a **SOC as a service** from a third party, as well as having an **incident response team** ready to be engaged. Once senior management has recognized the importance of these aspects of information security, the responsibility for implementing and maintaining those controls is likely to be delegated to you, the information security professional, at your organization.

With that responsibility, we need to ensure we take a risk-based strategy, including the following:

- Asset prioritization
- Threat modeling
- A detection budget
- A response budget

Beyond that, we need to set our strategy for the future and define what types of education and improvements should be undertaken to ensure the organization will remain protected, even in the face of constant change.

In terms of the present, one of the most effective measures available to information security professionals in achieving these heightened levels of visibility and enabling the SOC and information security team to be effective in their endeavors is through security monitoring. Let's discuss what may be involved with that next.

Using security monitoring to improve visibility

A specifically popular control for increasing visibility into the ongoings of your digital estate is the implementation of logging and the aggregation of those logs into a SIEM system.

SIEMs not only provide specialists the ability to investigate logs from your assets in your organization but also, lately they have been able to leverage IPS/IDS features, along with machine learning algorithms to enrich the log data and actively protect against compromise. This shifts the SIEM from being solely a *detective control* to being both a *preventative and detective control*, giving context and visibility into the previously dark alcoves of a network, along with reactions to mitigate the effectiveness of a threat actor.

This could include recognizing patterns and actions such as the following:

- **Account compromise detection** through authentication logs.
- **Malware detection** through system activity, firewall, IDS/IPS, and **CASB** *(***Cloud Application Security Broker***)* logs.
- **Intrusion detection** through contextual referencing, such as comparing IDS/IPS and vulnerability databases.
- **Malware host detection** through outbound connection logs from firewalls and CASBs.
- **Policy violation detection** through administrative activity, user activity, and change logs.
- **Detecting attacks on external-facing applications** through web application firewall logs, web or cloud server logs, or application and database server logs.

In order to understand whether these detections are valid and credible threats, it will be important to prepare certain information to enable your *incident response team* and *SOC* to compare and reference.

This information could include the following:

- *A list of hosts* that are expected in your environment. This helps team members or automated systems to detect unwanted hosts or endpoints in your environment.
- A *baseline log of network activity* over a specific duration that could be seen as "normal." The issue with gathering this information is answering the question "What if we're already breached?". The principle of "assuming breach" has become popular across the Zero Trust network idea, but many organizations aren't prepared to take those steps as rapidly as we should hope.

- *A list of operating services on each host* to serve as a baseline. It can build on your host list mentioned previously and should include the port numbers and protocols that are expected to be communicated on for that specific host.
- *An interactive diagram or visual representation of network functions* aligned with the OSI reference model. As previously mentioned, there is a risk of providing tainted data in this process, which actually justifies the existing threats in your network, because they're assumed to be "normal."

To avoid the pitfalls of providing garbage information that masks the threat already living in your estate, you should consider a project that tackles the following:

- Identifying *all processes and services existing on all servers* and specifying what they do
- Identifying *the types of communications* that those processes and services would produce and/or receive
- Determining the *criticality of those services*, and with that their vulnerabilities and the level of risk associated
- Determining whether any *services are redundant* (as in, they are not required)
- Determining whether any *critical services lack redundancy* (as in, they are critical, but don't have any fallback measures in-place)

"Ah, here we go!", you're thinking right now. "Asking us to do the impossible again!"

For some, trying to get an accurate representation of all the network activity in the estate seems impossible… which is why automated systems are starting to offer the solution instead of asking for teams of people to put aside their daily tasks to dedicate time and effort toward tasks that are as difficult as what we've seen previously.

Instead, these automated systems are able to "scan" and "detect" the type of activity that has been found in your estate, and enrich that data with intelligence surrounding malicious activity occurring across other estates globally, and then use that information to notify your SOC and respond to any active threats.

These solutions may be a more tenable solution for your organization and can help navigate the complexity of your estate, filtering out all of the noise to highlight only the most valuable insights and risk-based detections. These insights can even be turned into reports or "management-style" reports based on key point summaries.

Some examples of this software can include the following:

- Microsoft Azure Sentinel
- Splunk Enterprise Security
- IBM QRadar Security
- LogRhythm NextGen SIEM Platform
- AT&T Cybersecurity AlienVault USM
- Graylog

Regardless of the decision you make for which SIEM and intelligence solution is right for your organization, there are a few best practices that might be worth noting and applying in the early stages of onboarding the solution into your estate and creating processes that are best suited.

Security monitoring best practices

Security monitoring and SIEM tools are highly beneficial in both the detection and prevention of serious threats attempting to exploit your estate and help to ensure your organization is compliant with regulations and requirements while giving measurable insights into where your security posture can improve.

For systems like these, we want to collect as much information as we can, from both a technical and legal or regulatory perspective, to ensure the highest level of visibility and aggregation and derive meaningful insights from the activity in our estate. This could include *network device and server logs*, *Active Directory/IAM logs*, *vulnerability scanner and configuration management tool metrics*, and so on.

Additionally, it's important to implement redundancy in this aggregated data and treat it as you would any critical data, where the risk is mitigated with backup and recovery strategies to align with your information security policies.

There also exist some limitations in *SIEM and security monitoring*. Sometimes it isn't feasible to collect certain information or the network and computational throughput isn't able to handle the amount of data in your estate. It's important to note the amount of data you'll be expecting the system to handle before making a decision on the specific solution and vendor appropriate for your organization.

I would like to cover some particularly important steps to take before leaping headfirst into security monitoring that can help you and your organization avoid common missteps and mistakes.

Establish requirements and define workflows

Let me ask you this: how can you weigh up the features of the solutions available if you don't know what you need from your SIEM?

First, as with everything, before even deciding on a solution, it's important to do the following:

- Establish your requirements.
- Establish your timeline objectives.
- Define what the ideal workflow looks like to you and your organization.

Sometimes, vendors will try to highlight features that are of no value to you and only exist to dazzle the unprepared. On the other hand, sometimes vendors will come to you with a better solution that solves more problems than you were expecting or solves your problems in a clever way, which is a great situation to be in.

When we're establishing our requirements, let's consider the *KISS principle* again. Define a reasonable scope to pilot your solution. You don't need to boil the ocean on the first day of having your SIEM implemented, and there's definitely room to roll out your system gradually, allowing for troubleshooting and testing of defined processes. By doing so, you're providing a proof of concept, reducing the likelihood of edge cases, while also demonstrating the benefits of a SIEM system to key decision-makers.

With that in mind, make sure to keep your end goal in mind, to ensure the solution is able to scale to the level of data you would like to cover. This might be measured in **Events Per Second** (**EPS**), or maybe **Gigabytes Per Day** (**GB/day**).

While you're considering your requirements, objectives, and workflow, you want to consider your compliance and regulatory requirements. *SIEM* is able to help demonstrate the effectiveness of your other security controls to regulators and auditors. In previous chapters, we discussed creating a list of regulations and standards, as well as the specific requirements for each of the regulations and standards your organization is obligated to comply with, so you'll already have this prepared, right? Right?!

With this list, you're ready to mention your compliance requirements to the vendor during presentations and demos, ensuring that you are able to collect, retain, and remove information that is required in order to stay compliant. It can also help with estimating the amount of storage required for retention purposes, helping streamline the pricing discussions.

Define specific rules and ensure their effectiveness

Moving on from establishing your requirements and defining your workflows for *SIEM* and *incident response* processes, you are able to define the types of data that are useful to your organization and what they may indicate. Many solutions have various "built-in" indicators, such as the following:

- Authorization failed attempts
- Authorization successes
- User privilege changes
- Application server errors
- Application server performance issues
- Administrator activities

Furthermore, it's important to define the information you don't want to store or perform SIEM activities on, or perhaps would like to restrict the type of activities on, such as the following:

- PII
- Information that is illegal to collect
- Credit card and banking information
- Passwords
- Encryption keys

From these definitions and the rules created from the definitions, it's important to regularly test the effectiveness of the solution, to ensure that the expected outcome does indeed occur. This includes *exact-data matching* or *near-matching*, or *thresholds* that need to be passed (for example, this needs to happen more than three times in 1 minute before an alert is created), and so on. It also includes the communications and notifications created from the solution and whether the appropriate people are notified as expected in the defined requirements.

I'm fairly certain that during the process of workshopping and defining your requirements and plans for both *incident response and security monitoring*, you'll notice theories that you thought would work but don't. It's your opportunity to start acting on the next best practice, which is to… you guessed it, continuously improve your policies and configuration.

Continuously improve your SIEM configuration and incident response policies

The IT landscape is changing faster than nearly anything else ever has in history, and its pace is only accelerating further. What that means is that solutions become outdated, information about solutions or assets previously deemed as high-quality becomes either difficult to utilize or useless, and new threats appear to exploit or avoid configurations that were previously seen as being bulletproof.

As a result, we need to set a recurring date in our calendars that focuses entirely on refining the SIEM configuration and incident response policies (among everything else, but let's just talk about SIEM and incident response). We can't just let it rot, and even if we're leveraging a *SaaS* solution that gets automatic updates from the vendor, we still need to ensure the current reality of our estate is reflected in the system.

Now that we've concluded this section about security monitoring, I would like to wrap up with a summary of this chapter.

Summary

So, now that concludes *Chapter 7*, *Owning Security Operations*. I believe we've covered the high-level requirements of information security professionals to handle security operations. As with every section of this book, we have covered many topics, but each one very briefly. Full books can be dedicated to any of this chapter's sections, or even individual paragraphs on their own. I hope this serves to raise interest in learning more about each subject area in the future!

In this chapter, we covered various important aspects of the daily requirements from an information security operations team.

First, we looked at effective strategies in provisioning and maintaining assets, including the policies and configuration management steps required, as well as touching on the concept of IaC to streamline and automate these processes.

Then, we moved on to focusing on availability, disaster recovery, and business continuity activities, including the processes of defining, implementing, and testing disaster recovery and business continuity plans.

From there, we progressed toward the topics of managing upgrades, patching, and applying security controls in your organization, which included information about education requirements, change control processes, and security improvement programs.

Then, we delved into investigating events and responding to incidents, including defining incident response plans and performing security investigations.

Then, to close out the chapter, we looked at implementing and utilizing detective controls, including how to use monitoring to improve visibility inside your estate, leveraging increased levels of logging, aggregating those logs into a SIEM, applying automated logic and machine learning insights to the data, and having that data reviewed by members of a SOC.

Finally, we looked at security monitoring best practices, including establishing requirements and defining workflows, defining specific rules and ensuring their effectiveness, and continuously improving your SIEM configuration.

Now, it's only appropriate to move on to our final chapter, *Chapter 8*, *Improving the Security of Software*, my personal favorite topic.

8

Improving the Security of Software

Software has enabled organizations around the world to increase their productivity to unforeseen levels of efficiency, helping to automate previously manual and menial tasks. By looking at your organization's software assets (and updating your risk register as you do it), you have become more and more aware that nearly every business process is aided by at least one software solution, and that the more resilient, secure, and available the software is, the more benefit is seen by the organization.

Some of your software has probably been developed in-house, and other software has been purchased or licensed from third parties. These systems often present huge attack surfaces, with many moving parts that are ready to be exploited, and since they process confidential and sensitive information and store business-critical data, unauthorized access to (or destruction of) these systems can lead to either permanent loss of critical data or the loss of confidentiality or integrity of intellectual property, company secrets, and customer data. Furthermore, these breaches can lead to colossal fines from regulators, adding insult to injury.

The terrifying thing is, even with all of the reasons to care that I've just listed, the procurement, development, and use of various software systems are usually undertaken with absolutely zero security-focused oversight. This is exactly why understanding and enforcing improved software security is crucial in order to ensure your organization mitigates the risk presented by the software it uses.

In this chapter, I'm going to delve into some of the topics that can help you ensure a higher standard for software security in your organization. This includes topics such as the following:

- Exploring software security paradigms
- Understanding the secure development life cycle
- Utilizing the OWASP Top 10 Proactive Controls
- Assessing software security

Overall, what we want to learn from this chapter is how to establish requirements for software, regardless of whether it has been developed by a third party or an in-house development team. I'll cover the methods we can use to understand the risk profile for software systems developed by a vendor, how to reduce the likelihood of vulnerabilities and errors in your organization's in-house development activities, and methods to mitigate against security risks, focusing on the CIA triad.

Without further ado, let's proceed on with this chapter.

Exploring software security paradigms

I'd like to take you on a trip down memory lane for a moment, and remember April 2014, an important moment in the general history of InfoSec; the world was blindsided by the disclosure of the *CVE-2014-0160* vulnerability, given the moniker of **Heartbleed**. Now, when I use the term *the world*, I mean it. *Heartbleed* was the *Jaws* of software security blockbusters, getting a website of its own (`heartbleed.com`), and even its own logo:

Figure 8.1 – The Heartbleed vulnerability's logo

In the disclosure was information about how the **OpenSSL cryptography library** contained a vulnerability related to a **buffer over-read**, allowing a malicious actor to access cryptographic keys and login credentials, along with various other pieces of confidential information. It sounds bad, but it gets worse: the OpenSSL cryptography library is used in the **OpenSSL version of the TLS protocol**, widely used globally for securing data in transit, pretty much everywhere. To put it plainly: over half of the top 1 million most-popular **TLS-/HTTPS-protected sites** were affected and able to be exploited.

Even though a fix was released for Heartbleed on the same day as the disclosure, it didn't prevent devices such as firewalls, Android phones, and other trusted hardware, as well as software and websites, from being left vulnerable until the patches were implemented and updates were rolled out to align with the most recent version.

Furthermore, let's be clear: the vulnerability had existed in the code since 2011; we know this because it was viewable to anybody who cared to look through and read the source code. This was as a result of the fact that OpenSSL was software created in an **open source** development context. Open source means that the source code of the software is freely available to be read, modified, and redistributed. A perceived advantage of the *open source software movement* is the idea that transparent software prevents security-through-obscurity among other vulnerabilities in the source code, as it would be discovered by anybody who cares to look. Unfortunately, in the case of OpenSSL's cryptography library, it seems as though either nobody noticed or nobody was willing to shout about the vulnerability until 3 years later.

That might seem strange, right? Wrong. Up until recently, it's been the default. Security has been an afterthought to any organization looking to create new features and roll out new products, but as time has gone on, with blockbuster-style disclosures such as Heartbleed entering into the lexicon of the average business owner and IT professional, we've seen a paradigm shift toward giving security a bit more consideration.

So, how can we ensure we avoid implementing vulnerable software into our organization, regardless of whether it's a vendor-created productivity suite, an open source cryptography library, or a tool developed by an in-house team for your employees? You're not going to catch every issue, so it's important to manage your expectations on the matter. Just like with anything in information security, it's about risk management and setting up the appropriate policies and procedures to ensure the necessary steps have been taken, in line with the value of the assets in question.

To begin with, let's look at an idea that can be applied to more than just InfoSec, to any time we buy anything.

Buyer beware

Have you ever heard the term *let the buyer beware*? I'm betting you've heard the Latin version, *caveat emptor*, right? How about we use this opportunity to do what every person reading this chapter focusing on software security was hoping for and take a quick dip into Latin terms found in contract law:

Caveat emptor, quia ignorare non debuit quod jus alienum emit

Loosely translated to English, this means the following:

Let a purchaser beware, for he ought not to be ignorant of the nature of the property which he is buying from another party.

I'm surprised it's caught on so well, to be honest. Usually, if you need to take a breath in the middle of reciting an adage, it doesn't really last. Despite being quite long-winded, this idea has stood the test of time and must be considered when procuring new software.

It's up to the buyer to perform due diligence on software systems that are going to be utilized in their estate. It's your responsibility to ensure your organization has undertaken that due diligence appropriately and stores that information for future use in the event of needing to prove to regulators or auditors that the appropriate precautions were taken and that due care was applied in the process.

"But how?", you ask. "How can I make sure?".

Legal documentation

"Oh great," you say after reading the heading of this section, "he's going to talk about paperwork next." Well yes, I am, unfortunately.

The fact of the matter is, you can't be 100% certain that the software you've purchased has been developed securely. You can, however, try to implement terms into the contractual agreement you have with the vendor to ensure they understand your requirements and are responsible for ensuring their developed solutions are created with security in mind.

Contracts that define *liabilities* associated with the vendor are a form of risk mitigation. What I mean is that in the event of a breach that leads to the disclosure of either your organization's confidential information or sensitive information related to individuals, are you able to be reimbursed by the vendor based on the level of loss your organization faces from fines and reputational damage?

Whether or not the vendor accepts those terms depends on various factors, including the nature of the terms, the size of the vendor, the size of your organization, how big of a customer you are for them, and so on.

Sometimes, you'll just need to implement other types of mitigation and accept that the vendor isn't accepting liability in the event of a breach, regardless of whether a backdoor or malware was introduced from their side. It's your responsibility to keep the agreements documented and reference them when you're performing risk assessments, both before and after the software has been implemented.

Furthermore, the ideologies relating to InfoSec for your assets that we've covered in this book still stand. We must allow least-privilege access to the software, we should monitor the software's activity and how it interacts with our other assets, and we will continually review and assess the software's suitability and risks over its entire life cycle in our organization.

Speaking of life cycles, I figure we should discuss the idea surrounding the **Secure Development Life Cycle** (**SDLC**), as knowledge surrounding this topic relates to both software developed by third parties and your organization's own undertakings in developing software.

Understanding the secure development life cycle

The SDLC is all about baking security into the development of software through a set of processes. When you ask your vendors about their SDC, you're going to want to understand the methods that they are employing to ensure the software they're selling you is secure enough for your organization.

The same company that offers the *CISSP*, known as *(ISC)²*, also offers the **CSSLP**, or **Certified Secure Software Lifecycle Professional**, which covers eight domains that you need to understand in order to pass the exam:

- Secure Software Concepts
- Secure Software Requirements
- Secure Software Architecture and Design
- Secure Software Implementation
- Secure Software Testing
- Secure Software Lifecycle Management
- Secure Software Deployment, Operations, Maintenance
- Secure Software Supply Chain

Obviously, going into each of these topics in any sort of depth is going to occupy more than the 30-page limit I have on this chapter, and if I did so, it would repeat much of the concepts I've previously touched on in this book, including labeling the types of data or implementing the principles of least-privilege and separation of duties, or defense-in-depth, but it's still worthwhile to highlight some of the key takeaways unique to the SDLC and pique your interest for further investigation.

We can essentially split the SDLC into five stages, each with its own sub-groups:

- Defining business and security requirements
- Designing secure software
- Test plans for secure software
- Secure software development
- Testing the software

Shall we delve briefly into each of these bullets to discuss the process at a slightly more detailed level? Let's do it, but first I would like to address the elephant in the room: the idea that *nobody develops code this way anymore*. Let's talk about the various *software development methodologies* and how they might work with the SLDC.

Compatibility with various software development methodologies

Over time, the most widely chosen methods of developing software have transitioned from a stage-by-stage, project-managed, "release major updates twice a year" **Waterfall** type of approach to software development toward a more rapidly iterative approach, often referred to as **Agile**, where the solution is broken up into small cycles, attempting to deliver a working product early and improving incrementally and often.

Furthermore, there might be a split between the developer team at an organization, and the IT operations team, or there might be a more integrated approach to developing and deploying software, such as the **DevOps** approach, for example:

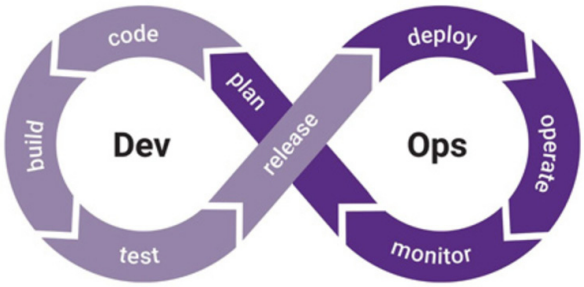


Figure 8.2 – The DevOps life cycle diagram

Regardless of which method(s) your organization develops software with (and there are many different styles inside those methodologies), the SDLC process can still be utilized and employed as a baked-in approach. Notice in the preceding diagram of the DevOps life cycle that it aligns with the five steps of the SDLC I mentioned previously, as long as you find a place to ensure you define the business and security requirements for the software in the planning phase!

Okay, let's take a closer look at each of the five steps of the SDLC, starting with defining your business and security requirements.

Defining business and security requirements

You see, in order for good software to be created, requirements must be defined at the beginning of any iteration. It rarely is, but then again, software is rarely what we would consider good. Inside those requirement definitions, along with usage scenarios and solutions, we need to ensure we define the security and privacy requirements for the software, and in the process of development, the software must be tested against those security and privacy requirements.

Beyond that, we need to consider the **abuse cases** or *ways a malicious actor may misuse the software*, essentially performing **threat modeling** with consideration into the **attack surface**. As a result of the findings from the abuse cases, we need to ensure that the appropriate mitigations have been put into place (with expenditure in line with the level of risk) to prevent or decrease the *likelihood* of those attacks from being successful and find a way to reduce the *impact* if the attack is successful.

I will discuss how we can do this in a more detailed fashion in the *Utilizing the OWASP Top Ten Proactive Controls* section.

Designing secure software

With the abuse cases and requirements in place, we want to design the software architecture to have the appropriate controls (preventative, detective, and responsive) in place, and ensure the overall structure of the application and the way it interacts with other systems and individuals are *secure by design*.

Testing plans for secure software

Along with *unit testing*, *integration testing*, *system testing*, and *acceptance testing*, *security testing* must be implemented into the SDLC in order to catch security vulnerabilities through various methodologies, including both automated and manual processes.

Prior to the development of the software, we want to plan and define the tests that must be done in order to ensure the software development process takes these tests into consideration.

The **ISO/IEC/IEEE 29119 standards** for *Software and Systems Engineering – Software Testing* are a set of standards comprising five different parts:

- *Part 1: Concepts and definitions*
- *Part 2: Test concepts*
- *Part 3: Test documentation*
- *Part 4: Test techniques*
- *Part 5: Keyword-driven testing*

There was a bunch of pushback over the *29119* standard, where several groups claimed it's unnecessary to standardize testing. I don't think this should matter to you in this chapter; it's up to you to determine whether any of these ideologies is applicable in your organization. The bottom line is, the standard's *Part 3* offers template documentation examples for your test plans, and if we're following the SDLC, then it might make sense to lean on those rather than spending time reinventing the wheel.

The templates from ISO/IEC/IEEE 29119:-3:2013 (yes, I know, what a name) cover the organizational-level, project-level, and dynamic test documentation you might like to use, including the following:

- **Organizational test process documentation templates**:

  - *Test policy*

  - *Organizational test strategy*

- **Test management process documentation templates**:
  - *Test plan (including a test strategy)*
  - *Test status*
  - *Test completion*
- **Dynamic test process documentation templates**:
  - *Test design specification*
  - *Test case specification*
  - *Test procedure specification*
  - *Test data requirements*
  - *Test data readiness report*
  - *Test environment requirements*
  - *Test environment readiness report*
  - *Actual results*
  - *Test result*
  - *Test execution log*
  - *Test incident report*

This means you'll have the appropriate suite of templates to help your product team(s) and engineering team(s) quickly and systematically document and meet the requirements set out for your software. If it seems like all of this process and documentation will get in the way of getting *actual* engineering work done, you're probably a software engineer.

With that said, I'm well aware of the complaints and concerns that are raised by developers in this circumstance, especially the faster we develop things and the more rapidly we integrate changes into our production environment. If your organization is utilizing these principles, then I'm not going to advise you to bog down your team with a bunch of paperwork.

Our job as information security professionals is to understand the risk and reduce it to an acceptable level. Part of this process is ensuring that the software that is being created by the development team is fit for purpose and meets the requirements defined, and showing that without these processes in place is a difficult task.

With the documentation I mentioned from ISO/IEC/IEEE 29119, we aim to do the following:

- **Analyze the software**, where we specify the users, use cases, use scenarios, and resources utilized by the solution.
- **Design the test strategy**, where we define the objectives for the tests and estimate the resources required and costs associated with the testing phase. We also define what is in scope and what is out of scope.
- **Define the test criteria**, where we create a flow that defines when we stop testing in the event of a failure and what constitutes a successful test completion.
- **Define the test environment**, where we specify the user and business environments that we will test in.
- **Define the schedule**, including deadlines, estimations for each resource, and any blockers or dependencies that might hamper the ability to complete the tests.
- **Determine test deliverables**, including test results and reports, test procedure documentation, and release notes in the event of the software meeting the test requirements.

So, it's not like I'm asking you to write *War and Peace* here, and if we're following the *Agile* process, each small improvement can have a truly short definition for each of the six previously listed concepts, potentially right after the **user story**. The user story is a way to define software requirements to developers in a structured way:

```
As a <insert role>, I want <insert requirement>, so I can
<insert use-case>.
```

That covers most of the first step, analyze the software, and after that, we can include our test objectives, resources required for the tests, acceptance criteria, test environment, and test schedule.

If errors are found, they can be solved quickly and easily in the same iteration or sprint.

So, how might you want to test the software in order to ensure security vulnerabilities are taken into consideration and avoided?

I'm going to go into that, but first I'd like to talk about the process of *securing software development*, the next phase of the SDLC.

Securing software development

In the process of developing the code, we want to ensure our developers are following certain ideologies related to secure code. We can create training and awareness programs and provide documentation on how to ensure they are following secure software development ideologies, including but not limited to the following:

- **Input and output sanitation**, to prevent *injection* vulnerabilities and *denial of services* from malicious or negligent user input
- Appropriate **error handling**, and ensuring the user isn't given too much information in the event of an error occurring
- **Resource management** and **concurrency**, ensuring processes are scalable
- **Isolation**, ensuring processes are segregated through *sandboxing*, *virtualization*, *containerization*, and related technological concepts
- **Cryptographic control** selection and implementation, ensuring the appropriate protocols are implemented correctly, with cost-benefit analysis taken into consideration
- **Access control structures**, including ensuring *trust zones*, *least-privilege*, and **Role-Based Access Controls** (**RBACs**) are considered and appropriately applied

In order to avoid repeating ourselves, I think it might be more useful to cover these ideologies in further depth by having a look at the types of tests we may implement to check for well-known security flaws in software and explaining how those flaws can be avoided or mitigated against.

Testing the software

Considering we have already defined the appropriate test processes, relevant stakeholders, environments, and acceptance criteria, now is the time to execute those plans, with both automated and manual processes being undertaken to ensure any changes that have been made to the source code, environment, or usages of the software are appropriately effective, efficient, and secure.

This is not to say we perform all these tests after the software has been developed. If we're using more rapid development methodologies, we must leverage more automated systems and create rapid notifications for any manual testing processes required in order to ensure there isn't a bottleneck at the testing phase of the SDLC. This *near-real-time* approach to testing new software results discovers issues quickly and ensures the developers are adaptive and *fail fast*, discovering their errors quickly and remediating before too much time has been put into a flawed approach. By doing so, we're helping the developers create more secure software, quickly.

Looking back at how we might have defined our test plans and designed our test processes, a few examples of what we would like to implement along the process in order to increase security and speed up testing processes could include the following:

- **Architecture analysis**, including end-to-end process and design examination.
- **Dependency checks**, which automate the process of checking for vulnerabilities in imported third-party software dependencies.
- Automated **SAST**, or **static application security testing**, which we've mentioned in previous chapters.
- Automated **DAST**, or **dynamic application security testing**, which we've also mentioned in previous chapters.
- **Code review**, where one developer reads and critiques the code of another developer. This helps with knowledge-sharing, as well as implementing the *two-man principle*, which means the two parties would need to collude in order to implement a *backdoor or malware*. It's not fail-safe, but it's part of a *defense-in-depth* approach.
- **Penetration testing**, which we've spoken about multiple times, could occur upon significant change, as well as on a regular point-in-time basis.

Inside these processes, we will as a bare minimum want to check to ensure that the OWASP Top 10 Proactive Controls (`https://owasp.org/www-project-proactive-controls/v3/en/0x04-introduction.html`) have been implemented. This means putting documented procedures into place for both automated scans as well as manual code reviews, and training developers on all of these steps and controls. It's crucial for the success of your SDLC, so don't neglect training, awareness, policies, and procedures!

I'd like to go into the Top 10 Proactive Controls from OWASP in the following sections.

Utilizing the OWASP Top 10 Proactive Controls

Let me briefly cover each of the OWASP Top 10 Proactive controls for improving the security of software. Each control has its own section.

Define security requirements

As we've previously discussed, the ability to articulate and document the requirements expected to be fulfilled by a software solution is highly beneficial to the organization for various reasons, including *cost-savings* and *useability improvements*, but another business requirement that must be fulfilled by either software development work or purchased/open source software are those requirements surrounding information security.

Creating standard security requirements based on best practices and industry knowledge helps developers and procurement staff reuse the knowledge they gain from previous iterations, so it's highly recommended to define the requirements in a way that will be uniform and stand the test of time.

Inside the *OWASP Top 10 Proactive Controls* section on defining security requirements, they reference the **OWASP ASVS**, or **Application Security Verification Standard**, which is a collection of security requirements, and the various criteria required for verifying that those requirements have been met. As a sample, it includes categories of best practices for various information security functions, such as access control, error handling, authentication, and so on. It's an excellent resource that should not be ignored.

As we previously spoke about, utilizing *user stories*, as we would see in Agile software development processes, as well as *misuse user stories*, can help teams operating in these functions to verify that the security requirements have been met.

As an example, we can reference the *ASVS* requirements that you could leverage for authentication, such as verifying that there are no default passwords in use, as referenced here:

> *Verify there are no default passwords in use for the application framework or any components used by the application (such as "admin/password").*
>
> *– ASVS 3.0.1, requirement 2.19*

To convert that into a **user story**, use the formula I mentioned previously:

```
As a <insert role>, I can <insert requirement>, so I can
<insert use-case>.
```

Let's do it for ASVS 3.0.1, requirement 2.19:

"As a user, I can enter my username and password, so I can gain access to the application."

Or:

"As a user, I can enter a long password that has a maximum of 1023 characters, so I can use a strong and unique password."

A **misuse user story** is told from the perspective of the malicious actor:

"As a malicious actor, I want to be able to enter the published default username and password, so I can gain access without permission."

It's up to you whether you want to utilize one or both of these in your user stories, but just be careful that your developers understand what they are, and don't misread the misuse user story as a requirement!

Leverage security frameworks and libraries

Software development has a wealth of resources available online, from free online courses to learn various languages to documentation for web app frameworks, and the list goes on. That culture has extended into the world of security in software development, with standards, frameworks, libraries, and other highly documented resources available to anybody interested enough in learning. Sometimes, it's as simple as importing a few packages into a development project.

With that, it's still important to remember a few key ideologies to ensure the risk of utilizing third-party resources is safe and secure:

- Using trusted sources that have a track record for maintenance activity. If a framework or resource isn't actively updated and maintained, it will present challenges and has a high likelihood of vulnerabilities.
- Ensure we document all third-party libraries and software inside our risk management asset catalog. This may be difficult, but there are services such as Snyk (`https://snyk.io/`) that automate the process of gathering the list and applying a risk score to that list.
- Ensuring that updates are applied before the software is facing breaking changes due to lack of maintenance. Aside from the previously mentioned Snyk, OWASP themselves offer a tool called **OWASP Dependency-Check** to look for publicly disclosed vulnerabilities in your third-party library list. NPM has a similar service available, and more options appear regularly.

- Ensure developers maintain the *principle of least-privilege* any time they can. Allowing increased access increases the attack surface, and therefore will likely increase risk.

You know, the basics that we've been harping on about for this entire book now!

Secure database access

When software interacts with data stores, such as SQL databases, a number of key ideologies can be upheld in order to increase the security of the software and reduce risk to an acceptable level. As an example, we don't want users to be able to input anything into a query that is then interpreted as a command without being parsed, checked, and sanitized first:

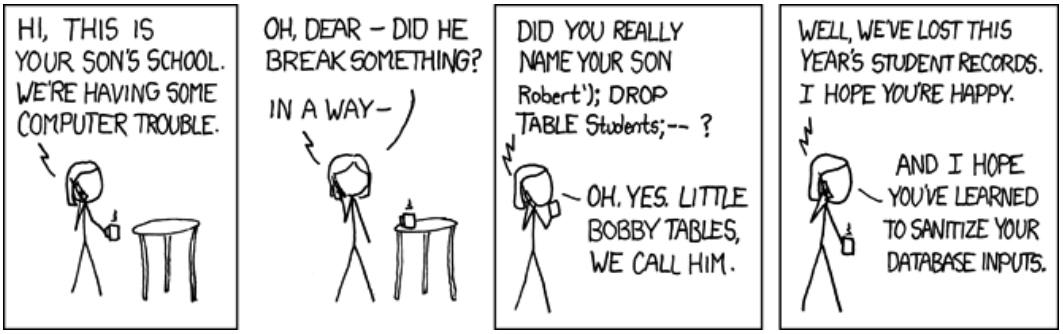


Figure 8.3 – XKCD #327 – sanitizing inputs

Ideally, we won't even give the users the ability to enter queries themselves, but that depends on the use cases. By using protections such as **query parameterization**, you can create *secure queries* that reduce the likelihood of an injection attack, such as SQL injection.

Read more about *query parameterization* from the *OWASP Query Parameterization Cheat Sheet* (`https://cheatsheetseries.owasp.org/cheatsheets/Query_Parameterization_Cheat_Sheet.html`).

Furthermore, you'll want to ensure your database and any computing platform used to run the database are properly set up with a **secure configuration**, preferably leveraging the baseline configurations we referenced in previous chapters.

Secure authentication means that authentication is protected through various means. For example, it's important to ensure that authentication is performed through a **secure channel**, protecting any credentials from being exposed. When credentials are stored at rest, they must be protected with *defense-in-depth* for the various threats, including *access control and encryption-at-rest* for protection against attacks on confidentiality, and *redundancy* for attacks on availability.

Finally, while the software and database communicate back and forth, it's important to ensure that the communication takes place over a *secure channel*, utilizing *encryption-in-transit*, as well as preventing the user from gaining too much information from error messages and logs.

You can read further into database security from the *OWASP Database Security Cheat Sheet* (`https://cheatsheetseries.owasp.org/cheatsheets/Database_Security_Cheat_Sheet.html`).

Encode and escape data

Another way to prevent *injection attacks* is to **encode and escape** any special characters with an escape character (such as adding a backslash character before a double quote character, `\"`, to prevent the interpreter from closing a string) in order to reduce the risk of the interpreter or browser outputting dangerous content as a result of bad input.

Encoding output data helps protect against **Cross-Site Scripting** (**XSS**) attacks, among other flaws such as operating system command injection or the time somebody's emoji use ended up crashing an entire banking system (`https://www.vice.com/en/article/gv5jgy/iphone-emoji-break-apps`).

Encoding and escaping can occur at various stages in the process of accepting and outputting data and should be considered any time *user input* or other *untrusted data* is interpreted or dynamically output back onto the interface.

You can read more about how to prevent XSS attacks from the *OWASP Cross-Site Scripting Prevention Cheat Sheet* (`https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html`), and how to prevent injection attacks from the *OWASP Injection Prevention Cheat Sheet* (`https://cheatsheetseries.owasp.org/cheatsheets/Injection_Prevention_Cheat_Sheet.html`).

Validate all inputs

Closely related to the last point is the idea of *validate all inputs*. We want to ensure all inputs are appropriately validated and properly formatted before they're stored or interact with any part of the software or system.

That includes the classic syntax checks, such as if a user is entering a credit card number, we want to make sure that it follows the format of a credit card number.

This includes preventing any blacklisted characters or terms from being accepted, as well as having a whitelist if the input can only be a certain number of things. For example, if the input is asking for a two-letter abbreviation for a US state, we can have a whitelist of the 50 "approved" inputs and prevent any other input from being approved.

These checks should follow the *defense-in-depth* idea and shouldn't rely on a **happy path** for the user to follow, as malicious actors like to find ways around the frontend controls.

Leveraging validation functionality in various security libraries and frameworks can simplify this process for your development team, but they should always be tested to ensure they're fit for purpose on your project.

Treating all inputs as though they are malicious is the way forward, so let's just agree to add that to all of our SDLC policies moving forward, alright? Further reading on input validation can be done with the *OWASP Input Validation Cheat Sheet* (`https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html`).

Implement digital identity

We need to ensure that our individual users are given an ID while interacting with our software, in order to both offer a streamlined experience during their session, as well as providing ourselves the ability to understand errors and track misuse.

NIST released a special publication, *800-63B, Digital Identity Guidelines – Authentication and Lifecycle Management* (`https://pages.nist.gov/800-63-3/sp800-63b.html`), and it's worth referencing in the process of implementing and leveraging digital identity.

Inside the *NIST Special Publication 800-63B*, various controls and their proper implementations are detailed. This includes requirements for various "levels" of applications, depending on the information contained and processing performed.

For example, a *Level 1* application is one that is considered low-risk and doesn't contain any private data or PII. As a result, only password authentication is required, with checks against commonly used passwords and *password length requirements* of 10 characters or more being suggested for users not utilizing **multi-factor authentication** (**MFA**). I highly recommend reading the publication for further guidance.

The process for users of your software to reset their *forgotten passwords* should include MFA methods to prove their identity. A typical way to achieve this is to utilize a side-channel such as email and send a password reset email containing a unique link to the email address associated with the account. More information on the suggested process can be found in the *OWASP Forgot Password Cheat Sheet* (`https://cheatsheetseries.owasp.org/cheatsheets/Forgot_Password_Cheat_Sheet.html`).

When we talk about session management and cookies, a few steps can be taken toward better security for our users. For example, expiry for the cookie should be set. Setting an `HttpOnly` attribute on a cookie prevents the cookie from being referenced and utilized in JavaScript code. Any transfer should be protected with TLS encryption to prevent man-in-the-middle attacks from grabbing a session ID, by setting the `Secure` attribute.

Further reading on session management, web authentication, and access control can be found in the *OWASP Session Management Cheat Sheet* (`https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html`).

Enforce access controls

Similar to the previous point on digital identity, access control and authorization are about granting and revoking various permissions or privileges from users. By leveraging known paradigms, such as RBAC, for example, which we've gone into previously in this book, and by building out strong access controls up front that prevent vulnerabilities and flaws such as *access creep*, you're ensuring that your users have access to what they need, nothing more and nothing less.

In order to enforce the access controls implemented, all requests must be passed through an access control check, with the default setting being deny. As a result, rules shouldn't be hardcoded, and instead should be dynamic based on the access level granted to the present user interacting with the software. Again, it's a matter of defense in depth.

Finally, ensure you keep track of all events related to access with logging and pass those logs into security monitoring resources. We will discuss that later.

Further reading can be found in the *OWASP Access Control Cheat Sheet* (`https://cheatsheetseries.owasp.org/cheatsheets/Access_Control_Cheat_Sheet.html`).

Protect data everywhere

The basis of the information security profession is to ensure the security of information. When we transfer or store data in our applications, we want to make sure that it's adequately protected from threats and take a risk-based approach to applying controls, with the ideas of least-privilege, defense in depth, and other approaches we've discussed in this book taken into consideration.

In terms of data security, it includes data classification, where we apply a sensitivity level label to each piece of data, and map those labels to the adequate control requirements defined by you and your organization, based on regulations, reputational risk appetite, and confidentiality, integrity, and availability risk appetite.

An easy win in terms of protecting data is to implement encryption for data in transit, generally TLS. This protects against **man-in-the-middle** and **side-channel** attacks occurring on either the communication between the user and the frontend application server or between the frontend application server and the backend.

Encrypting the data at rest reduces the risk related to loss of confidentiality, but the complexity and technical skills required might slightly increase the potential for misconfiguration leading to a loss of availability. Encrypting data at rest is an important step, but must be carefully planned and implemented in order to be effective.

Leveraging cryptographic libraries can be a good starting point, regardless of whether my horror story at the beginning of this chapter was entirely based on organizations experiencing an information security incident due to using a third-party cryptographic library.

Secrets such as credentials, certificates, SQL passwords, and so on should be managed and protected as though they are the crown jewels of your application. Don't store them in plaintext in the databases, don't hardcode them, and rotate/de-provision the keys associated with users when they leave the organization. *Secret vaults* such as **HashiCorp Vault** offer the streamlined automation of this process but require a technical skillset.

Further reading can be found at the following links:

- *OWASP Transport Layer Protection Cheat Sheet* (`https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.html`)
- *OWASP Cryptographic Storage Cheat Sheet* (`https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html`)

- *OWASP Password Storage Cheat Sheet* (`https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html`)

Implement security logging and monitoring

As we've previously spoken about security monitoring in previous chapters, I don't believe I need to delve deeply into how that might help your organization. The bottom line is: more information can help you and your colleagues detect security incidents with better accuracy and improved response times.

You can increase the level of information that your organization's developed applications feed into your monitoring solution, but remember that sensitive information such as PII or confidential information is sometimes included in diagnostic and debugging information, and might need to be removed before reaching the monitoring solution. Additionally, make sure that your timestamps align across your nodes or your data will be inaccurate and difficult to use.

In order to prevent any log forging or injection attacks, you should consider the encoding principles and validation exercises we mentioned in *#4* and *#5* of this list. Another protection against tampering is to set up access control and authorization principles for your logging solution. In order to avoid loss of availability, your solution should have some form of redundancy in the dataset, such as storage in multiple locations and backups.

Further reading into how you might use security logging and monitoring can be found in the *OWASP Logging Cheat Sheet* (`https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html`).

Handle all errors and exceptions

Ensuring your application is resilient and able to handle errors and exceptions is an important property of ensuring all forms of the CIA triad. Bad error handling can lead to the disclosure of confidential or sensitive information, disruptions to the application's stability, or even modification to important data that you rely on for various monitoring purposes.

Bad error messages can give a malicious actor more information than acceptable as well. Have a look at this signup page that I need to believe is a joke, or else I've lost all hope:

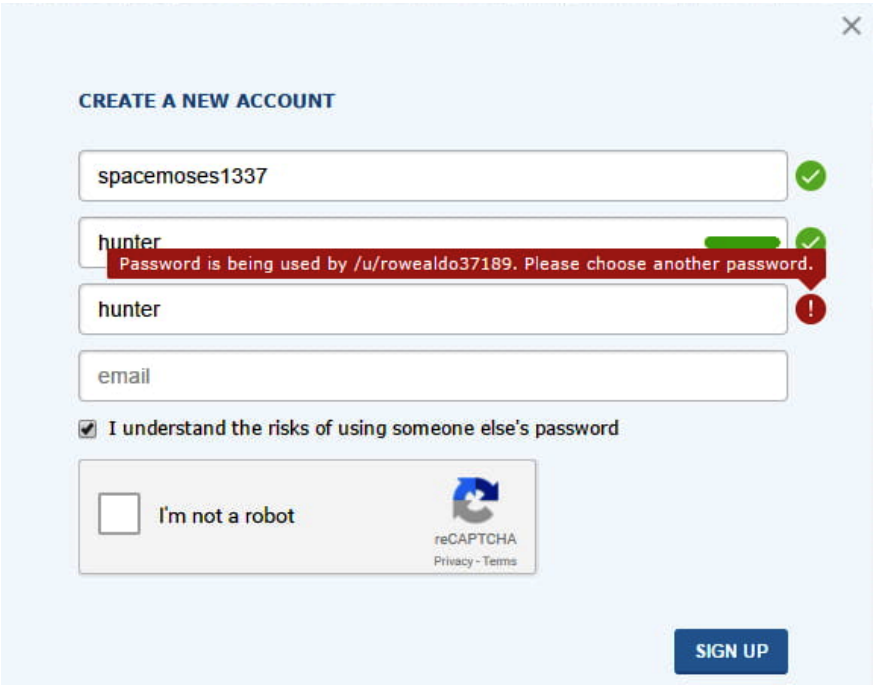


Figure 8.4 – The dumbest signup page error message ever

I can only guess the user decided to go with `hunter2` as the password after that error message.

When we are creating errors, make sure the user isn't able to get more information than they need, but give them enough to find the answers themselves or to reach out to a member of support for help. Internally, we want to log enough to help with support, testers, forensic investigations, and incident response teams. Remember that if team members from the software developing team actively collaborate with the security, testing, and business team members, it will lead to better software overall.

Finally, good tests must be created in order to catch these exceptions before users do. Great testers, combined with some automated tooling for finding errors both statically and dynamically, will help reduce the overhead required for support, as well as reducing the risk of the application suffering a loss of one or more of the CIA triad principles.

Further reading can be found in the *OWASP Error Handling Cheat Sheet* (`https://cheatsheetseries.owasp.org/cheatsheets/Error_Handling_Cheat_Sheet.html`).

If you enjoy this topic as much as I do, I think some further reading might be of interest to you, and so I've compiled a few different links to frameworks and processes for you to investigate. These are useful in implementing security in the SDLC:

- **MS-SDL**, the **Microsoft Security Development Lifecycle**: `https://www.microsoft.com/en-us/securityengineering/sdl`
- *NIST 800-160*, *Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems*: `https://csrc.nist.gov/publications/detail/sp/800-160/vol-1/final`
- The Cybersecurity and Infrastructure Security Agency's **CLASP**, or the **Comprehensive Lightweight Application Security Process**: `https://us-cert.cisa.gov/bsi/articles/best-practices/requirements-engineering/introduction-to-the-clasp-process`

Okay, fine, we've gone through the SDLC, so now we can go into detail about how it might help us with assessing software next.

Assessing software security

Moving forward, I would like to discuss the methods we might utilize in order to assess the security of software. In previous chapters, we've looked into the importance of regular testing of software and systems, including *penetration testing* and *vulnerability scanning*, and the remediation of any of the findings. I've encouraged the implementation of *configuration management systems* that can help keep your organization's assets up to date, and *monitoring solutions* to uncover performance issues, misuse, errors, or malicious activity. I've also talked about *resilience and redundancy*, and how expensive it might be for each hour that your organization loses access to one of their systems.

Now, with that all said, if we're going to go deeper, I think it's relevant to split this topic into two sections because the methodologies and approaches are different depending on who has ownership of the software, taking into consideration the *cloud operating model* and the shared responsibilities related to utilizing **Software-as-a-Service** (**SaaS**) products.

On one hand, we have third-party vendors who develop and sell software to companies. Their focus is creating a tool that is valuable to your organization, such as **Customer Relationship Management** (**CRM**) systems, HR software, productivity tools, web analytics software, cloud storage solutions, and so on.

On the other hand, we can also leverage an in-house development team to create software for our organization to either solve problems that aren't currently offered by a vendor or to save money based on the cost presented by the current offering from third parties. Additionally, we can leverage outsourced development teams to create our in-house software or leverage the power and convenience of open source software.

Let's delve into how to improve the security of third-party vendor software.

Reducing the risk from software developed by a third-party vendor

As we have now covered the various controls and processes that can be put into place to reduce the likelihood of software being created with vulnerabilities in the code itself, it must be noted that many of your organization's software solutions are going to be either purchased from a third party or pulled from an open source software repository resource such as the public repositories on GitHub.

Policies and procedures need to be put into place to ensure that you have visibility on the procurement process, regardless of which scenario is occurring, and have oversight on the software assets being utilized in your estate.

Here's the issue: what can you do about it? When we're looking at the major software that I see used by nearly every organization, such as those developed and sold by Microsoft, Salesforce, Atlassian, and so on, how can you ensure they're employing the appropriate controls for their solution, as well as following an SDLC approach?

Cloud-based software

The massive paradigm shift we've seen over the past decade toward the SaaS model has increased the pace of implementing new software into an organization, where a user is likely accessing the solution through their web browser. This model for using software has led to the responsibility of handling servers and databases shifting away from the customer and onto the vendor.

That isn't to say we absolve ourselves of any responsibility when it comes to the privacy and security of the data processed and stored in these SaaS applications, however; in fact, it is absolutely our responsibility as customers to ensure that the solution we choose is *secure enough* for our organization.

As I mentioned previously, it might be difficult to arrange for your own testing to be performed on the infrastructure and software currently being offered by your SaaS vendors. The reasoning behind that is simple: that type of testing activity could present a level of risk that is deemed unacceptable by the vendor, as you could accidentally disrupt their service for millions of other users. You can't perform any tests on infrastructure without permission either, or you'll be putting yourself and your organization at risk for legal liability.

So, how can we go about asking our vendors about their software's security?

Understanding third-party software risk

We've previously mentioned in this book that we might – from the perspective of being an information security professional working at a software vendor – create an *information security whitepaper* to detail security paradigms and controls utilized by your organization. With that said, now that you're the potential customer, how do you ask the appropriate questions to ensure the software your organization is in the process of procuring is appropriately secure and controls are in line with the level of risk?

Well, we perform *due diligence* and produce *legal documentation* defining liability and the agreements between your organization and the vendor. I already covered this in the preceding thrilling section where we talked about Latin terms found in contract law… and how could you not remember that?

In these agreements, you might be able to arrange for documentation from the latest penetration test performed on the software to be shared for your review, or for your own security team to be able to perform their own testing as a form of due diligence. It all depends on what both parties are willing to agree to and what will give you the assurance required to help meet your level of risk tolerance. As we've said, we have to be aware that it's not always possible to penetration test or even to vulnerability scan a software solution, but even if you were to find a vulnerability in a software solution that has been developed by a third party, how are you going to notify the vendor, and do you expect them to fix the vulnerability in a timely manner?

In your legal documentation, you might also want to try and define the *SLA*s and procedures for the remediation of vulnerabilities, but we still haven't covered how to get a better look into our vendors' security posturing. How do we know that they're handling security in a way that we deem is acceptable?

In this circumstance, we often send a set of documents known oftentimes as the **Vendor Security Assessment Questionnaire** (**VSAQ**) to software vendors for them to complete before an agreement is made. Most organizations send out some Excel spreadsheets with rows and rows of hundreds of questions, along with drop-down or free-text cells to input answers, and/or evidence and justifications for answering the way they did. If you've ever worked as a security professional at an organization that develops software to be offered to customers, you've had to fill these assessment spreadsheets out, and they're never the same, unfortunately. Google tried their best to standardize these questions into an open source, interactive, and dynamic solution on their GitHub page (`https://github.com/google/vsaq`), but from what I've seen, it still hasn't caught on. If we can make this happen, it would really save us all a load of time and effort, while delivering a high-quality result. Can we please make this happen?

Upon receiving the completed VSAQ back, we shall document the residual risk that is unmitigated by the vendor, and measure that residual risk to see whether it's below the threshold level of risk acceptance or not. In the event that the risk is above the acceptable level, we need to make the decision to either mitigate it, accept it, or avoid it. I should hope, considering we're nearly at the end of the book now, that it's all awfully familiar at this point.

Once you've completed your due diligence and communicated your findings to the business decision-makers, and the agreement is secured and finalized, the implementation of the software will begin. It might be on-prem installed software, or it might be cloud-based SaaS.

SDLC due diligence for software vendors

Since we covered the various steps and requirements for an SDC, you now know the basics, and you have much of the information you need to ensure you can perform the appropriate due diligence surrounding their development life cycle.

First of all, it's important to not only ask whether the vendor follows processes surrounding the SDLC, but also to have them detail how exactly they do so. You could also ask for evidence as a validation method. If they answer "no," then you need to highlight the risks associated with utilizing this software that has been developed without regard to the SDLC to the appropriate decision-makers in your organization.

Penetration testing and bug bounties

We might be able to arrange to penetration test the software, but if it's a SaaS, as we've said previously, don't get your hopes up, as most vendors will not allow it. Besides, performing that sort of in-depth, technical activity on all potential vendors will become costly for your organization very quickly. A novel solution for both vendors and customers that has recently become popular is the implementation of a bug bounty program, where a software vendor opens their product up to be penetration tested by bug bounty hunters, who are trying to exploit security vulnerabilities for a reward offered by the vendor. Companies such as Bugcrowd (`https://www.bugcrowd.com/`) and HackerOne (`https://www.hackerone.com/`) are offering organizations such as yours (and your vendors) the opportunity to have their software "hacked" by security professionals globally. If your vendor has a bug bounty program, it may bode well for where they stand in their security posture, especially if you are able to review the bugs that have been found and the current open bounties.

If the vendor claims to be secure but doesn't want to provide penetration test reports and doesn't take part in a bug bounty program, you might want to ask them how they process and prioritize vulnerabilities that are coming from their penetration tests, or which tools they use for DAST, SAST, or dependency scans. Use your knowledge of the landscape to see whether they know what they're talking about, and try your best to get evidence where you can. It is the security of your organization on the line, after all.

On-prem third-party software

When we're looking at software that is installed onto your company's servers or endpoints, we have a few simple mitigations that we can employ to ensure your environment's security isn't compromised due to a vulnerability in this software. We want to reduce our attack surface by following the foundational concept of least-privilege we've covered.

"*Sorry, our software needs to run as admin*," the vendor tells you.

Your first question back to them should be "why?". You know too much to be told that and not question it further. Find out what the software does and why it needs admin access, and find out whether there's a way to mitigate against that risk.

Furthermore, we've discussed vulnerability scanning at your organization in previous chapters, and your new on-prem software should be included in the scope. If you find a vulnerability through these means, it is worth starting a dialogue with the vendor to understand the vulnerability and to see about patching it. They might react poorly as a result, but in the near future, I am absolutely positive that it won't just be you that pushes back about bad security in their software. Together we can move mountains and force vendors to create software that follows best practices.

What about on-prem software that is open source?

Open source software

When we talk about open source software, what we're generally referring to is **Free and Open Source Software** (**FOSS**). With FOSS, you're able to download software that has been created and hosted on various platforms such as GitHub or SourceForge in a way that allows the potential user to actually see the underlying code.

The idea behind open source is that we get software that is transparent, peer-reviewed, and free created by people looking to solve technical (or societal) issues without further incentive to do so apart from altruism and potentially demonstrating mastery.

With that said, the events surrounding Heartbleed, which we investigated earlier in this chapter, were a situation stemming from the use of insecure open source software being widely utilized. How can we ensure the appropriate mitigations have been implemented during the development process of the open source tools and pieces of code – also known as **snippets** – that are either currently being used at your organization or are under consideration?

Additionally, we need to remember that many of our vendors are leveraging open source software in the process of developing their proprietary software. Let's keep that in mind when we're developing our VSAQs for the due diligence process.

On the plus side, we have the ability to review the code. There should be requirements in our information security policies to ensure that the same security steps that we have put into place for our own software development are also applied to the open source software we use. This means performing dependency checks, SAST, DAST, vulnerability scans, and so on. Oftentimes, we'll create a security pipeline that automates the defined processes and security scanning steps required on any code or software that is deemed in scope.

Improving the security of in-house software

Let's say in-house software can include software that has been developed by either your organization's in-house development team or an outsourced development team, as well as the open source software that your organization is leveraging. Although that open source software was (likely) developed by a third party, you have the ability to treat it as in-house-developed code and put it through the same assurance processes.

I would like to discuss how we might implement security controls into development environments and deployment pipelines to ensure any risks have been identified and actioned before the code reaches the *production stage*.

Both in-house and outsourced development teams should be able to follow your direction when it comes to the policies you define to increase the security of your organization's developed software. Sometimes, outsourced teams have their own processes and solutions in place to provide the same level of assurance that you require, and other times you might need to reach an agreement to mitigate against any gaps.

With that, let's look at ways we can improve the security of our in-house software.

Software development life cycle

First, we want to implement the controls previously discussed in this chapter for the secure SDLC. These policies, procedures, and tools are able to ensure the efficient and secure development of high-quality software by your teams. You can design the SDLC to reflect your organization's *risk tolerance* and the idiosyncrasies of your existing processes, but staying in line with the general ideologies will help you in your quest.

With that, I don't believe it's necessary to cover all of the steps we've already covered earlier in this chapter, so instead, I would like to go into how we might implement practical controls to achieve the high-level goals we've previously covered.

Code repositories

Your organization likely uses a code repository solution to store and version-control any developed software. Examples of *code repositories* could include GitHub, Bitbucket, or GitLab, among others.

Inside these tools, we have various available features and controls that ensure our requirements for the SDLC are met. For example, **issue trackers** allow a transparent discussion between developers, product team members, and other stakeholders on software bugs, feature requests, vulnerability discoveries, and so on. These issue trackers allow teams to estimate requirements, assign priorities, and give complexity scores to each issue, to help organize how they might be handled.

There are what are known as **branches**, which allow developers to actively develop and make changes to software in a development context, without affecting the main branch, which is used for production, or even potentially a development branch for the current sprint. You want to protect this branch from change without various forms of sign-off, which you can arrange to include code reviews, testing, and ensuring the merge is following the separation-of-duties and two-man principles.

Each change to the code is documented and attributed to the associated software engineer, allowing transparent audit processes and clear documentation of all approvals. All commits can be signed using public-key cryptography, providing non-repudiation and integrity for each engineer's incremental contribution to the codebase.

Additionally, if we take into consideration the ideology of infrastructure as code that we previously covered, it's possible to store the configuration files for your organization's servers and tools in repositories and apply the same level of scrutiny as you would over the software that is being developed.

DevSecOps pipelines

Continuous Integration and Continuous Delivery (**CI/CD**) pipelines provide the automation flow aspect to ensuring only software that meets the specified requirements is able to be deployed to the next stage of the process, and by relying on the pipeline you are able to reduce developer and systems administrator access down to least-privilege, meaning they are not able to access or change production systems without following the pipeline and getting appropriate approval.

Quite often, these pipelines are related to the term **DevSecOps**, in which we bake security into the DevOps life cycle through automation and pipelines.

Let's imagine this pipeline flow for your organization's software development process:

1. A software developer uses their SSH key to access a project's Git repository, creates a "feature branch" off from the development code, and clones the code to their workstation for local development.
2. Their **Interactive Development Environment** (**IDE**) references secure coding paradigms, as well as performing regex checks for plaintext secrets and passwords, immediately informing the developer if any flaws are found in real time, allowing them to remediate before committing their code back to the repository and triggering a pipeline process.
3. The developer makes a change and attempts to commit that code back to the Git branch they created. Their access is checked to ensure they're allowed to do so, their change is logged, and differences are highlighted in the Git solution.

4. The CI pipeline runs and performs the following checks:

   - A *dependency scanner* assesses the risk profile of any imported software.

   - *Static code analysis* is performed by a *SAST* to check for the code's quality and any security risks.

   - A *secrets checker* looks for secrets and passwords stored in the code.

   - A *container auditor* checks for any vulnerabilities in the defined *infrastructure as code*.

   - An ephemeral version of the updated software is deployed into a test environment, and a *DAST* performs the appropriate checks on the software, looking for errors, security vulnerabilities, and performance issues.

   - If all automated systems pass, then another developer is notified via email or chat, and they perform a *code review*.

   - If the code review is approved, a tester is notified, and the feature is tested by a human.

   - If all steps are approved, the *code is merged* and the *CD pipeline* runs to implement the change into the next environment.

This process requires careful implementation and defined processes and expectations for the various team members involved but could lead to a streamlined, secure development process with automation at its core.

Summary

In this chapter, we covered the highly interesting topic of improving the security of software and highlighted various methodologies we could use to ensure the software used inside our organization's estate is "secure enough" from a risk perspective.

To begin the chapter, we went into a few universal paradigms for software security, including the SDLC, and the steps required for that process to be an effective undertaking.

After that, we highlighted that we put a lot of faith into software systems developed by third parties, especially when the processes under which they are developed are opaque, such as when we procure software from a vendor that doesn't disclose their approach when it comes to security. We delved into how we might better understand the risk presented by third-party software, either as proprietary solutions or in the context of the open source model.

From that, we went into how we can utilize our knowledge of the SDLC to produce better software in-house, with testing and automation at the heart of the solution to ensure scalability and efficiency in finding vulnerabilities, which could lead to a loss of confidentiality, integrity, or availability.

Overall, what we learned from this chapter was what to look for when you are establishing your requirements, policies, and procedures surrounding software development and use in your organization. With this knowledge, you're able to accurately measure the risk presented and respond to any residual risk that is deemed to be above the acceptable risk tolerance.

With that, you have now completed this book. Thank you so much for taking the time to read it! I hope you had fun and learned a few things along the way. If I was able to help you with your organization's security posture, I am delighted.

I would like to take this opportunity to thank my wife, Helen, for her support and care along the way. Furthermore, I would like to thank all of the members of the Packt team who have supported me in the creation of this book, from beginning to end.

Packt.com

Subscribe to our online digital library for full access to over 7,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Fully searchable for easy access to vital information
- Copy and paste, print, and bookmark content

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at packt.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at customercare@packtpub.com for more details.

At www.packt.com, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

Other Books You May Enjoy

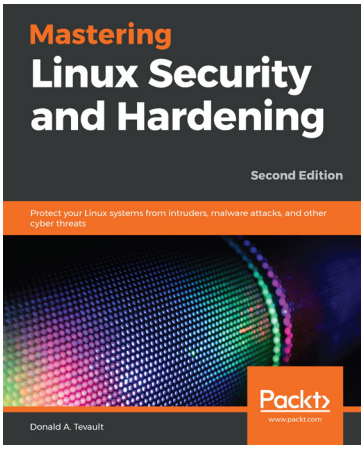
If you enjoyed this book, you may be interested in these other books by Packt:

CompTIA Security+ Practice Tests SY0-501

Ian Neil

ISBN: 978-1-83882-888-2

- Understand how prepared you are for the CompTIA Security+ certification
- Identify different types of security threats, attacks, and vulnerabilities
- Explore identity and access management in an enterprise environment
- Protect your business tools and platforms from cyberattacks
- Create and maintain a secure network



Mastering Linux Security and Hardening - Second Edition

Donald A Tevault

ISBN: 978-1-83898-177-8

- Create locked-down user accounts with strong passwords
- Configure firewalls with iptables, UFW, nftables, and firewalld
- Protect your data with different encryption technologies
- Harden the secure shell service to prevent security break-ins
- Use mandatory access control to protect against system exploits
- Harden kernel parameters and set up a kernel-level auditing system
- Apply OpenSCAP security profiles and set up intrusion detection

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit `authors.packtpub.com` and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Leave a review - let other readers know what you think

Please share your thoughts on this book with others by leaving a review on the site that you bought it from. If you purchased the book from Amazon, please leave us an honest review on this book's Amazon page. This is vital so that other potential readers can see and use your unbiased opinion to make purchasing decisions, we can understand what our customers think about our products, and our authors can see your feedback on the title that they have worked with Packt to create. It will only take a few minutes of your time, but is valuable to other potential customers, our authors, and Packt. Thank you!

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z