Secure web development

Pankaj Pant

Haaga-Helia University of Applied Sciences
Bachelor's Thesis
2022
Bachelor of Business Administration

Abstract

Author(s)
Pankaj Pant

Degree
Bachelor of Business Administration

Report/thesis title
Secure web development

Number of pages and appendix pages
47 pages

Modern web development comes with a variety of challenges that developers must consider, such as accessibility, responsive design, performance, speed, and scalability to name a few. Unfortunately, due to a lack of time and resources, other fundamentals sometimes go overlooked or are outright ignored—especially when it comes to security.

Overlooking cyber security in an organization can turn out to be costly, with some estimates suggesting that damages related to cybercrime is projected to be $6 trillion globally in 2021. In the past two years, the coronavirus pandemic was also responsible in the uptick of cybercrime worldwide and high-profile cases such as the SolarWinds hack and the Vastaamo data breach should serve as a cautionary tale for companies.

This thesis attempts to serve as a condensed guide on current security trends and how to incorporate robust security practices into their products, for up-and-coming software developers. The thesis will also look at the common pitfalls that web developers can avoid, to create more secure products. This thesis also tries to provide an understanding of how a potential hacker acts and thinks, and on this basis create guidelines on how to defend against potential attacks.

The thesis is mostly research based, and relies on academic papers, white papers published by reputed cyber security firms, web development cyber security guidelines by reputed foundations such as Mozilla and OWASP, and on podcast materials from Cyber Security Sauna (F-Secure) and Darknet Diaries.

Keywords
Cyber security, secure development lifecycle, OWASP

Table of Contents

1 Introduction

The world wide web has come a long way since its inception more than three decades ago. In the 1990's the web was mostly used to share documents written in HTML. A typical website back then would have consisted of a simple HTML document, to present data in a purely informational manner. The websites back then, did not pay any attention to user experience, nor did they allow users to send any input back to the server which could modify the interaction flow of the website – in other words Web 1.0 could be characterized as a 'read-only" internet (Techopedia, 2020). For those interested in the history of websites, it is possible to inspect how websites looked in the past, by using the Wayback Machine (https://archive.org/web/), just type in a web address and select the timeline to see how the website appeared in the past. Even today, if one visits https://www.berkshirehathaway.com/ (the website of Berkshire Hathaway – a US company run by legendary investor Warren Buffet), visitors will be greeted by a webpage that looks straight out of the 1990's – a website whose main purpose is to only present informational data and provides no user interaction.

The early 2000's saw the advent of Web 2.0 take place (Techopedia, 2020), as websites started to store user submitted data, allow user interaction to modify the flow of the website and most importantly allow collaboration with other users of the website. These key developments combined, basically laid the foundation for social media as we know it today – as it was now possible to make interactive blog sites, media sharing sites, and have support for user accounts and data persistence. From this point forward, the web became less of a document sharing platform and moved more towards a medium for sharing web applications.

Today the web is an ever-growing universe of interconnected web pages, apps, and media files such as videos, music, photos, and other interactive content. Massive advancements in the field of computing, miniaturization and telecommunications have meant, that each year more people are able to afford a device to access the internet, with current estimates indicating that 6 in 10 people are now 'online'. In the past year alone 332 million people came online for the first time (Kemp, 2021). From the data perspective, it has been estimated that in 2020, each person on earth produced 1.7MB of data every second (Stengel, 2021). That takes a while to sink in, but the reality is that every digital interaction we make, such as a search, stream, swipe, like, click

or a purchase, generates data. Even making this thesis accessible either digitally or in print, has probably generated a few MB's worth of data if not more.

The advantages that the modern web has brought to society are plentiful. It has revolutionized the way we communicate, made access to information, knowledge and learning easy for all. Even mundane and everyday tasks such as ordering a pizza or shopping for groceries can be completed with a few clicks. Unfortunately, like with other successful entities or companies, when a web app or web service turns out to be extremely popular with users or generates significant revenue, it can attract unwanted attention from nefarious elements of society. The nature of the web - speed, convenience, anonymity, and lack of borders – can be both an advantage and a disadvantage. Cybercriminals have long been taking advantage of this fact, and cybercrime has been rising steadily in both volume and pace because threat actors no longer need to be physically present when committing a crime.

The role of the web developer in the grand scheme of things, is to build the services and apps that the world needs and improve the day to day lives of people. While a normal user of a web app might only see and interact with the user interface (UI), the developer, and other people with the technical knowledge, know that a lot more happens under the hood to power the products. For developers, creating a good product or service is not enough, since for every web app that is built, there will be someone out there looking for vulnerabilities in the app and take advantage of them for their own benefit. While it is the job of a certified cybersecurity expert to probe and test the security of web applications and services, the web developer who builds the application should also be concerned with the application security side of things and make the app as robust as possible. No developer would want to know that their code was responsible for a vulnerability and provided unauthorized users access to sensitive data.

1.1 Objectives

This thesis aims to provide a comprehensive overview of what up and coming software developers need to know about current security trends and how to incorporate robust security practices into their products. The thesis will also look at the common pitfalls that web developers can avoid, to create more secure products. This thesis will also try to provide an understanding of how a potential hacker acts and thinks, and on this basis guidelines on how to defend against attacks will be presented. The thesis will be mostly research based, and will

rely on academic papers, white papers published by reputed cyber security firms, web development cyber security guidelines by reputed foundations such as Mozilla and OWASP, and on podcast materials from Cyber Security Sauna (F-Secure) and Darknet Diaries.

Hopefully this thesis will be of use to especially junior software developers who have grasped fundamentals of software development but are generally unaware of security fundamentals in web development. With each passing day, an increasing number of digital products are being launched, but securing them is usually the last topic on the mind of the organization. This thesis will provide junior developers a glimpse into what pitfalls their products might face if left unsecured and how to avoid them.

1.2 Scope

This thesis will not produce any significant product or web application which demonstrates a secure application. Instead, code samples for each submodule may be hosted on GitHub or similar online hosting platform.

2 Cybercrime and cybersecurity: an overview

2.1 Cybercrime

Cybercrime is an umbrella term for various forms of crime which involve using information and communication technology (ICT) to target computers, networks, networked devices, website, or data or alternatively using ICT to facilitate a traditional crime (Bossler & Berenblum, 2019; UNODC, 2019, Module 1). Cybercrime is one of the fastest growing criminal activities globally, and it has the potential to impact individuals, businesses, and organizations (Sjouwerman, 2019). The key characteristics, behind the rapid growth in cybercrime, is the fact that cybercrime transcends physical and geographic boundaries and can be conducted with relatively less effort, greater ease, and at greater speed than traditional crime (Alexandrou, 2019).

2.1.1 Types of cybercrime

According to Europol, cybercrimes can be classified into two main groups: cyber-dependent crimes and cyber-enabled crimes. The first group consists of cybercrimes that can only be carried out with the help of computers and computer networks. Cyber-dependent crimes are also called 'pure cyber-crimes' since all stages of these crimes require access to digital infrastructure, and it would not be possible to carry out these attacks without access to computers or the internet for instance (Europol, 2021, Norfolk Police, 2020). Hacking of computers, networks, cloud, etc., computer malware viruses, ransomware, and distributed denial of service (DDoS) attacks are all prime examples of cyber-dependent crimes (Europol, 2021). On the other hand, cyber-enabled crimes are more 'traditional' crimes such as theft and fraud, but which have been carried out with the help of computers. The one big difference between cyber-dependent crimes and cyber-enabled crimes is that the latter can be carried out without the help of computers and networks as well. In recent years, easy access to internet and personal computing devices has transformed both the scale and form of cyber-enabled crimes. Online fraud, online romance scams, theft of banking, financial or credit card data are all prime examples of cyber-enabled crimes (Norfolk Police, 2020).

2.1.2 Who are the threat actors?

A 'threat actor' is a commonly used term used in the field of cybersecurity, and it refers to anyone who has the intent and potential to impact an organization's IT security by gaining

unauthorized access to its data, network, or systems. More specifically, a threat actor can refer to anyone who is the main driver or a participant in a malicious attack targeted at a third party. Threat actors vary in skill levels, and they can be either an individual, a group, an organization, or even a country carrying out a cyberattack (Canadian Centre for Cyber Security, 2021).

Script kiddies – this is a category of low skilled attackers. Script kiddies typically rely on tools created by other more sophisticated threat actors to carry out their attacks. Teenagers are commonly associated with this category, but the attacks they carry out should not be underestimated. An example of one of the most notorious attacks by a script kiddie occurred in 2015, when a 15-year-old exploited a SQL injection vulnerability in the website of TalkTalk – a UK telecom provider (Gallagher, 2015). The attack resulted in the exposure of banking and personal information of millions of customers.

Organized crime – the actors in this category can be quite diverse, but they all share a common factor where they are all motivated by financial gain. According to the Verizon Data Breach Report, around 40% of all attacks can be associated to organized crime (Verizon, 2021). The techniques employed by this group can range from untargeted, sophistically low attacks - such as ransomware - to more targeted and persistent attacks. One of the most prolific collections of organized crime hacking groups is Magecart, who target online shopping cart systems to steal payment card details using web skimmers. Attacks by this group have targeted businesses such as Ticketmaster, British Airways and Forbes magazine (Cimpanu, 2018).

Hacktivists – this group refers to various activist groups that utilize cyber-attacks to promote their socio-political agendas. These threat actors typically target governments or big corporations using Distributed Denial of Service (DDoS) attacks and or stealing confidential information. The most well-known hacktivist groups are Anonymous and LulzSec and they both have carried out attacks against companies such as Fox Television, Sony, Nintendo, and even taking down the FBI's websites in 2011 (Arthur, 2013).

Advanced persistent threats – this group is usually associated with nation states which carry out cyber warfare. These groups tend to be funded, trained, and maintained by government agencies and carry out attacks on their behalf (Canadian Centre for Cyber Security, 2021). Their main targets are usually key critical infrastructure of other countries – to which they try to gain unauthorized access, to carry out intelligence operations. Some examples of APT’s that

are harboured by governments include the group called Lazarus which is backed by North Korea and was behind the infamous WannaCry ransomware attack of 2017 (The United States Department of Justice, 2018). APT's tend to carry out very long-term attacks as the systems they target are very complex and well-guarded to penetrate with simple or automated techniques. APT's require a lot of planning, observation and many more stages before the end goal is achieved.

2.1.3 Motivation

It can be difficult to know with full certainty the true motivations behind every cyber-attack, but generally they can be classified in the following categories: financial, espionage, disruption, political and retaliation. In certain cases, it is quite possible for multiple motivating factors to be identified within a single attack. Figure 1 provides a general overview of the various motivations that threat actors can have.

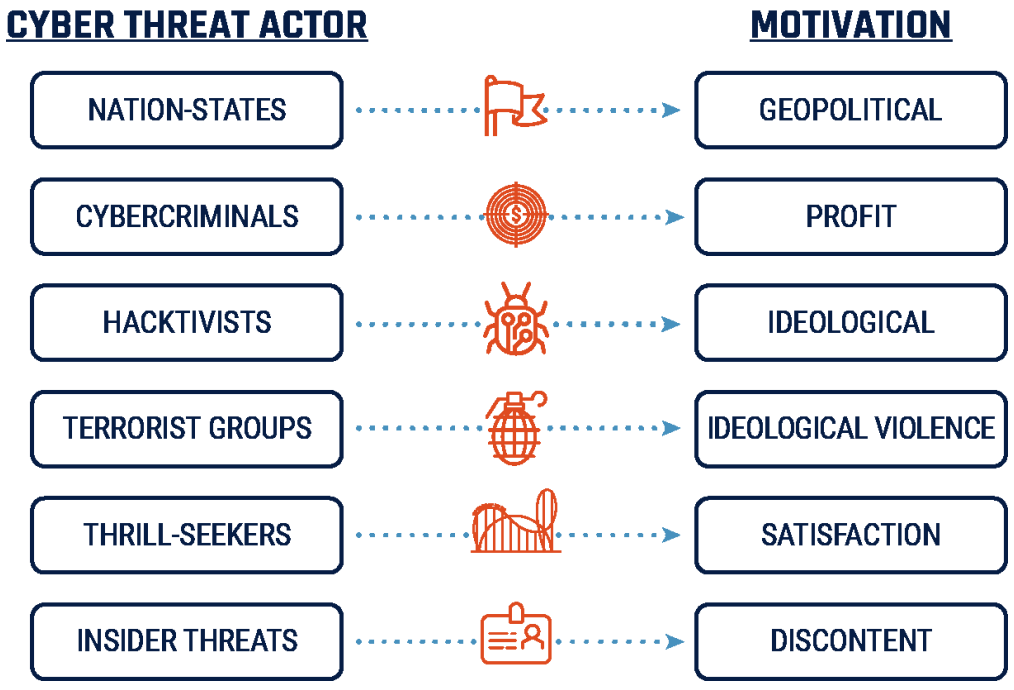

Figure 1. Motivations of cyber threat actors (Canadian Centre for Cyber Security, 2021).

In most cases today, the main driving force behind cyber-attacks is financial gain (Verizon, 2021). The perpetrators typically extort money from victims directly in the form of cryptocurrency which is difficult to trace and hence easy to launder (F-Secure, 2021). In recent years, state sponsored actors have also been increasingly utilizing cybercrime as a tool to

achieve a political goal, which normally would be difficult to achieve legally. Attacks on a country' infrastructure such as the power grid, nuclear facilities, or oil pipelines, or attempts to manipulate elections fall under this category (DeCiccio, 2021). Typically state sponsored cyber-attacks are extremely sophisticated, very well planned and focused on a singular goal which can take months or even years to execute. Espionage has always been one of the top motivations behind cybercrime – and is defined as a method of intelligence collection using human sources (field agents) or technical means (by hacking networks or devices) (UNODC, 2019, Module 14). Espionage has been around for centuries, and the advent of ICT has only accelerated its frequency, intensity, and scale. In some cases, the motivation behind the cyber-attack can be to simply disrupt the day-to-day operations of the target. By doing this, attackers seek to affect the target's decision-making processes and impose additional costs. For example, a disruption to the supply chain could have major impacts for an organization, which can affect the market position and stock price of an organization. Retaliation reasons are also motivating factors behind cyber-attacks. In a retaliatory attack, the threat actor is often driven by a desire for revenge – which could be targeted at a person they know, a former employer or a business the attacker feels has 'done them wrong'.

2.1.4 The effects of cybercrime

It is a rare day when there is not some news of a company or organization which has been hacked and their day-to-day operations have been severely affected until a ransom of several million is paid. As businesses and organizations transform, they are gradually moving away from storing their data in on premise data servers and are increasingly storing more of their data on cloud platforms, along with using other services from third parties and content delivery networks (Knott, 2021). These extra complications along with many workplaces having shifted their workforce to hybrid or fully remote working models, has resulted in massive increase in the attack surfaces that a threat actor can exploit. The COVID-19 pandemic was an unprecedented event which disrupted businesses and damaged the global economy, and it also brought with it a spike in cyber-attacks which affected companies, businesses, and organizations across all sectors (Europol, 2021). A recent Interpol report highlights that since the pandemic started, criminals have shifted their focus from targeting individuals and small businesses to targeting major corporations, governments, and critical infrastructure (Interpol, 2021). This is most likely a consequence of organizations rapidly deploying remote networks and systems to support staff working from home, which presented a large and attractive target

for criminals to steal data and generate income. Leading cybersecurity research companies such as Cybersecurity Ventures predict cybercrime to cause damages of $6 trillion USD globally in 2021 alone. Furthermore, it is expected that global cybercrime costs will increase by 15 percent per year over the next five years, reaching $10.5 trillion USD annually by 2025, up from $3 trillion USD in 2015 (Morgan, 2021).

Companies in the energy, financial services, manufacturing, technology, and pharmaceutical sectors have been the main targets of cyber-attacks and they have endured the most losses as well (Kessem, 2021). Cybercrime costs include several components and are not just limited to the potential ransom the victimized organization pays or theft of intellectual property and company data. For example, cybercrime costs also include intangible costs such as lost productivity, reputation loss and loss of customers, and disruption to normal course of business (Mossburg et al., 2016), as depicted in Figure 2.

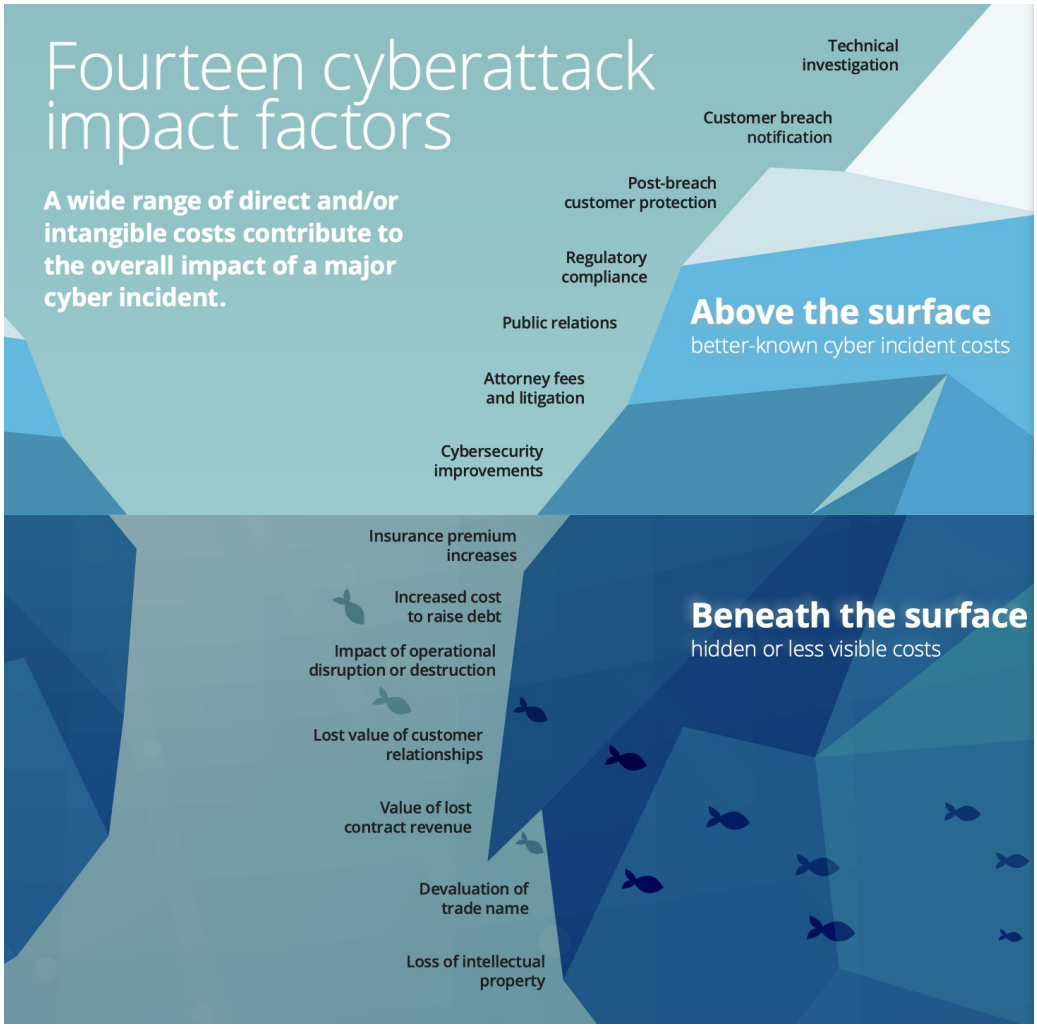

Figure 2. Potential impacts of cybercrime on an organization (Mossburg et al., 2016).

When a cyber-attack does take place, all operations in an organization can suddenly come to a halt if they are not well prepared. Operational disruption refers to major interruptions to an organization's normal course of business which results in lost revenue. An organization can be hit by any variety of cyber-attacks such as a malware infecting the systems and erasing critical information or malicious code on a server which blocks access to a critical web platform or API, which would both result in effectively bringing operations to a halt.

In most cases an organization will experience revenue losses due to a cyber-attack in the form of losing customers who act cautiously and take their purchasing power to other vendors to protect themselves and their data against cyber-attacks. In addition to this, it is becoming quite common for an organization that has been attacked to face extortion threats as well, which also leads to loss in revenue when a ransom is paid. In recent surveys it has been reported that around 83% of companies have acknowledged paying a ransom to threat actors to regain control of critical IT systems as they had no choice. (ThycoticCentrify, 2021).

Apart from operational disruption and lost revenue, the other biggest impact of cybercrime on an organization is the reputational damage it experiences. Reputational damage is an example of an intangible costs and is difficult to fully quantify in numbers. One thing that organizations that have fallen victim to cyber-attacks share, is the fact that the value of their brand falls significantly (Mossburg et al., 2016). Customers especially get spooked out and feel less secure about trusting their sensitive information in the hands of an organization whose IT infrastructure has been compromised. In some cases, this loss in brand value can also be empirically seen in numbers, as security researchers from Comparitech studied the impact of data breaches on companies listed on the New York Stock Exchange and found out that the share prices of compromised companies fell by an average of 3.5% following an attack (Bischoff, 2021).

The bottom line is that not only is cybercrime costly to an organization, but it is also costly to the end user/paying customer. As organizations are starting to realize the importance of cybersecurity, the preventive measures they undertake such as hiring cybersecurity professionals, obtaining data breach insurances, hiring legal expertise to remain compliant with cybersecurity regulations, and developing and implementing an information security program,

all increase the operating cost of an organization. These cybersecurity related costs eventually trickle down to consumers in the form of higher prices.

2.2 Cybersecurity

At the very basic level, cybersecurity is the practice of protecting networks, devices, systems, programs and data from unauthorized access and other forms of digital attacks (VISMA, 2021; CISA, 2019). It is also commonly referred to as information technology security, and the term can be used in a variety of contexts, from business to mobile computing. There are many elements of cybersecurity, which are described below as well as in Figure 3.

- Network and infrastructure security – the practice of securing computer systems, networks and other assets from intruders, unauthorized users, and attacks.
- Application security – the practice of keeping applications safe and secure, by evaluating the source code of an application to identify weaknesses and vulnerabilities. Ideally, security should be built into the application at the design stage and not as an afterthought after the application has been already deployed.
- Cloud security – the practice of securing cloud computing environments from intruders, unauthorized users, and attacks. As applications, data, and identities now reside in the cloud, and are always on and accessible to both legitimate users and threat actors alike if left unsecured. It is therefore highly important that authorized access is configured properly using tools such as two-factor authorization (2FA), the use of VPNs, security tokens, data encryption, and firewall services, among others.
- Information security – the practice of protecting the integrity and privacy of data from unauthorized access, exposure, or theft, both in storage and in transit.
- End user education – the practice of building security awareness across organization employees and other end-users. People are the most unpredictable cyber security factor and can accidentally introduce vulnerabilities by clicking suspicious links in emails, using untrusted hardware devices such as USB drives, having poor password hygiene etc. Trainings focused on improving security awareness for end users also ultimately strengthens endpoint security as well.
- Disaster recovery – the practice of setting tools and procedures in place which trigger appropriate business operations response in the face of unplanned events such as cyber-attacks, natural disasters and power or network outages. The disaster recovery

procedures basically lay out a plan on how an organization restores its operations and data and returns to normal operating capacity in the face of an event.

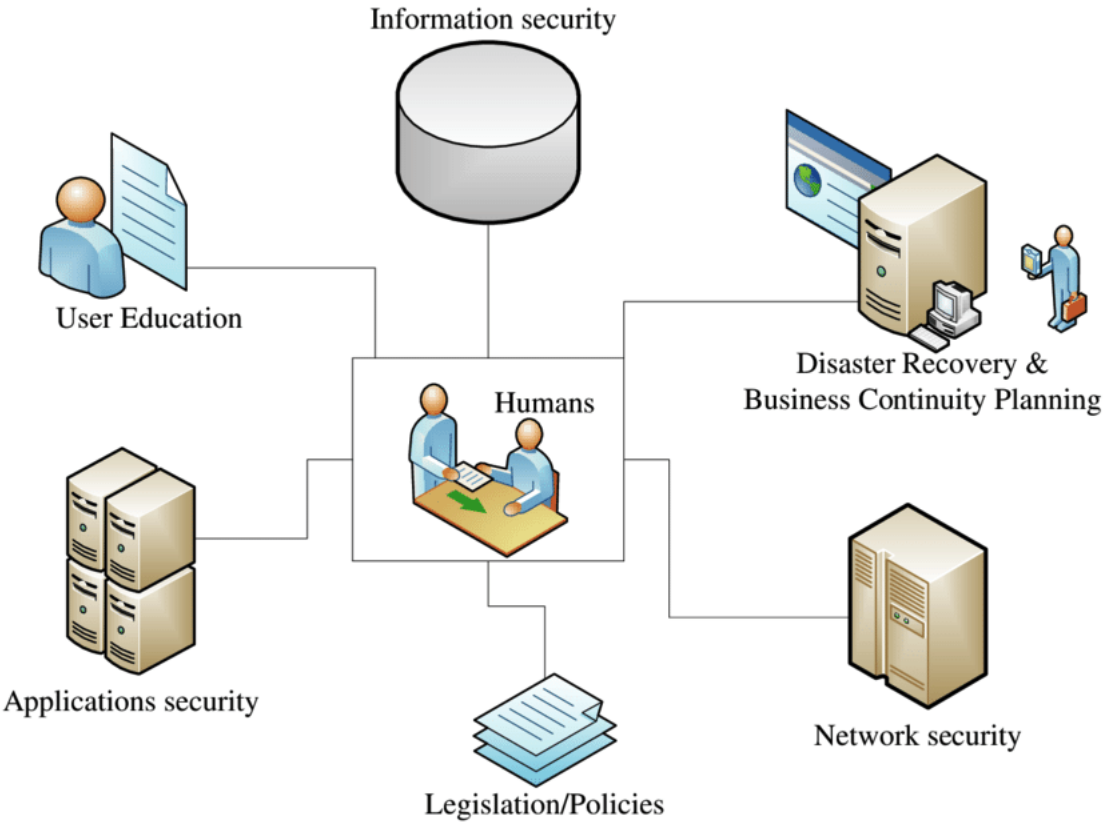

Figure 3. The various elements of cybersecurity (Nwankwo & Ukaoha, 2019).

Since cybersecurity also deals with defending data in the cyberspace, there is a bit of overlap that occurs between cybersecurity and information security. Most information nowadays is stored digitally, and cybersecurity extends the practice of ensuring confidentiality, integrity, and availability of data that is stored on a network, computer, server, or in the cloud.

Figure 4. The CIA triad (Purcell, 2018).

The CIA triad refers to an information security model which consists of three components – confidentiality, integrity, and availability, where each of these components represent a fundamental objective of information security.

- Confidentiality – refers to keeping data secure. Which data is private, and which is not private is defined by the business priorities, government regulation or industry compliance requirements. The main tenet of confidentiality is to keep data that needs to be kept private, private. This basically means that confidentiality is about making sure that only authorized users can access or modify data, and the organization is responsible for keeping their own data as well as customer's data out of harm's way (Nweke, 2017). Confidentiality can be violated both on purpose and inadvertently, either through a direct attack carried out by a threat actor or through an error by an employee or organization member.
- Integrity – refers to keeping data clean. Data integrity guarantees the fact that the data has not been tampered with, degraded, or been subject to unauthorized modification during storage or in transit, either intentionally or unintentionally (Nweke, 2017). When data is maliciously or accidentally altered, the trust between the organization and its

customers will take a hit – as customers rely on accurate, reliable, and up-to-date information.

- Availability – refers to keeping data accessible. This means that the data should be available to authorized users, whenever the user requests the data (Nweke, 2017). While the data storage unit might be accessible, but if other components of the network fail or if there is a power outage and there are no backups available – then data availability cannot be guaranteed. In a nutshell, for a system to demonstrate availability, the networks and applications should be running as they should, and the system must be resilient against interruptions caused by cyber-attacks, power outages, hardware failures and natural disasters.

The CIA triad form the foundation of all security frameworks, but it can be near impossible to strictly adhere to all the principles simultaneously (Walkowski, 2019). Depending on an organization's business priorities, its security goals, overall industry setting and applicable regulatory requirements, one of the CIA triad principles will have a higher precedence than others. For example, an e-commerce company will probably prioritize availability – as any downtime will result in lost revenue. A company in the financial sector will probably prioritize integrity as any possible corruptions in its stored financial data can be catastrophic. And finally, an organization which deals with highly sensitive or classified data such as the military will prioritize confidentiality. Finding the right balance between confidentiality, integrity and availability is often a critical step, and it helps set priorities for cybersecurity teams. Prioritizing one security principle over another also means that there will be trade-offs somewhere else. Choosing a system which does a thorough jobs of ensuring confidentiality and integrity might mean that aspects such as speed and performance can take a hit. This system might work for organizations in one industry (such as healthcare) but not for organizations in other industries (such as e-commerce). Trade-offs such as these are all part of the business, and a cost benefit analysis should be done to reach a decision. Therefore, each organization should consider their own unique requirements and decide how to apply these security principles, while at the same time ensuring they are able to provide a seamless and safe user experience (Walkowski, 2019).

2.2.1 Why is cybersecurity important?

Today, cyber-attacks can take a multitude of forms such as malware, ransomware, social engineering, and phishing, to name a few – and are ultimately meant to disrupt business

processes (Verizon, 2021). Due to the variety in cyber-attacks and the increase of attack surfaces, it is no longer sufficient to rely on anti-virus software or firewalls. With each passing day, the risk of cyber-attacks is increasing, and for organizations and businesses this means that the question is no longer "if" a cyber-attack will take place but rather "when" will a cyber-attack take place (Verizon, 2021). This unfortunate reality means that organizations and business nowadays absolutely must have cybersecurity defences and strategies in place to survive – thus making cybersecurity of such great importance (Gustafsson, 2021).

Cybersecurity is critical for several reasons:

- We have been increasingly using and relying on online platforms in our day to day lives. This means that our identities and personal data now lives in the cloud. From logging in to social media networks, healthcare systems or banking applications, if these systems don't store and transmit our data securely, then our data can potentially fall into the wrong hands. At an individual level, a data leak could lead to identity theft and extortion and as a result have serious implications on that individual (Verizon, 2021).
- Critical infrastructure all around the world such as energy, communication, healthcare, etc. among others as well as supply chains for food, agriculture and manufacturing are all managed through information technology. Cyber-attacks on critical infrastructure are becoming more widespread and can deprive citizens of key services and create massive disruptions and chaos (Verizon, 2021).
- We are also now wearing more technology and are surrounded by Internet of Things (IoT) devices. For threat actors, this boom in devices such as smartphones, smart watches, fitness trackers, smart thermostats, smart home camera setups, has meant that there are more devices to potentially attack and steal sensitive data from (ThycoticCentrify, 2021).
- As noted by several research reports from IBM and, the number of cyber incidents is increasing every year. DDoS attacks alone are estimated to occur 26,000 times per day which equates to 18 per minute (NETSCOUT, 2021). Overall, a single security incident such as a data breach cost an organization on average between $3.8 – $4.24 million (IBM, 2021).

- More importantly, it isn't only the sheer number of cyber-attacks that is increasing, but the severity of the cyber-attacks is on the rise as well. In 2021, there were two cyber-attacks which ranked very high on the severity scale. The attack on the Colonial pipeline forced the company to completely shut down its entire IT and pipeline operations in its 57-year history. The impact and severity of this cyber-attack is fully understood when one considers the fact that the Colonial pipeline is responsible for providing 45 percent of the fuel for the entire US east coast. This cyber-attack disrupted fuel supplies for several days which caused fuel shortages and fuel price increases (Osborne, 2021). The second severe cyber-attack of 2021 took place weeks after the Colonial pipeline attack, and this time the victim was the world's biggest meat producer and processor called JBS. The attack forced JBS to shut down operations in the US, Australia, and Canada for an entire day, before JBS reportedly paid the €11 million ransom to the threat actors (JBS, 2021). If JBS had not caved into the demands, the halt in operations which would have continued had the potential to disrupt food supply chains and lead to inflations in food price. Both these attacks were sophisticated and difficult to contain and in both the cases the victims ended up paying ransoms to the threat actors, to get back controls of their IT operations and infrastructure.

The rapid advancement of technology in computing, miniaturization, telecommunications, and the internet in the past few decades has placed personal computers, smartphones and other internet connected devices such as smart TVs, smart thermostats, smart locks, etc. inside consumers homes and in office spaces worldwide. Besides the silicon-based hardware, the other main component that is common to all these devices is software. Software is literally everywhere now. This explosion in software and hardware has made securing the cyberspace an extremely difficult challenge. Overlooking cyber security in an organization can turn out to be costly, and as mentioned in a recent Gartner report, 60% of organizations are increasingly looking at cybersecurity risk when making purchasing decisions with any third party (Gartner, 2021). There is a high demand for safe and secure data and to achieve this goal, a shift in mindsight is required, as cyber security can no longer be something that gets delegated only to the IT department. Instead, the approach should be that everyone in the company needs to be involved in cyber security from the board of directors to the new hire that started a day ago.

2.2.2 Why developers should be familiar with web security fundamentals

The role of a developer is to build digital services and products according to the business requirements and specifications. But unfortunately, for every product built there is a good chance that someone out there is looking for vulnerabilities to exploit for their own benefit (Knott, 2021). While it is the job of a certified cybersecurity expert, such as a penetration tester, to probe and test the security of applications and services, the developer should also be concerned with the application security and make the product as robust as possible. Ultimately, no developer would want to know that their code was responsible for a vulnerability and provided unauthorized users access to sensitive data.

For a long while there has been a mentality that cybersecurity and software engineering are two separate entities (Mouratidis & Giorgini, 2008). A lot of misguided advice refers to the fact that majority of web frameworks can handle security well and developers should not worry about the security aspect. Using a framework for both the backend and frontend is a very common practice in software development, as they abstract away a lot of the configuration, provides structure to the application and generally allow developers to focus on features. However, using a framework without fully understanding how it works and what its strengths and weaknesses, can potentially put the security of your application in jeopardy (Boyer, 2018). Using a framework does not however eliminate the fact that a developer can write defective code which can leave the application open to security vulnerabilities. In addition, a lot of organizations use the same frameworks along with various dependencies, which means that a vulnerability in either the framework or a dependency can put any application built using them at risk. This risk magnifies even more if developers do not actively update the dependencies that their applications use once the vulnerability has been reported (Contrast Security, 2014). In a lot of cases, speed and agility in product development take priority, and security is considered an afterthought, which greatly increases the attack surface of the application.

Some of the criteria that needs to be thought through beforehand include:

- What features does the application need?
- How should the application function?
- What are the public endpoints?
- What are the private/internal endpoints?
- What are the user inputs?

- What are the outputs?
- What are the potential weaknesses that can be exploited by a threat actor?

2.2.3 Integrating security into the software development life cycle

When developers and organizations become security minded – then it is possible to start integrating security into the software development life cycle (SDLC). SDLC refers to the "framework describing all activities in a software development project from inception to decommission" (Dooley 2017, Chapter 2). Over the years, many SDLC models have emerged such as waterfall, iterative, agile and CI/CD which have different ways of managing the software development life cycle, but in general share the following phases:

- Planning and requirement gathering
- Architecture and design
- Coding
- Testing
- Release and maintenance
- Decommissioning

In the past, software engineering and cyber security have not overlapped with each other a whole lot. As a result of this, implementing security into a product is usually considered after the architecture, design and coding of the product has been completed (Mouratidis & Giorgini, 2008). Implementing security measures as an afterthought can mean that the security measures are forced on to the product, which in turn usually results in placing unintended limitations on the product. In the absolute worst case scenario security testing and implementation is not carried out at all and whatever vulnerabilities exist, will remain open for exploitation.

But in most cases, security related undertakings are only considered quite late in the SDLC. Due to the relatively late stage when security testing is done, the flaws and vulnerabilities that are discovered can be very expensive and time consuming to fix. Various reports point out that the costs of fixing a bug found during the testing phase can be 6-15 times more than ones caught earlier in the design phase (Synopsys, 2020). Taking this into account, it makes sense that by integrating security testing across the SDLC, will help in detecting and reducing vulnerabilities early and as a result fixing such issues is also faster and cheaper. When security is built in to the SDLC, security related activities such as architecture analysis during design, code reviews

during implementation of features and penetration testing before product release are carried out at regular intervals (Synopsys, 2020). The benefits of these checks are immense and results in:

- More secure software products, with security baked in to the SDLC.
- All stakeholders (such as product owners, scrum master, developers, testers, etc.) are all fully aware of the security measures in place.
- Being able to detect security design flaws before implementation begins.
- Being able to save time and money by detecting faults early and not having to rectify them at a later stage.
- Reduce risks of having security vulnerabilities in software products and thus reduce business risks to the organization.

3 Common web application security risks

There are quite a few web application security vulnerabilities out there, that it would be impossible to present them all and discuss in this thesis. Vulnerabilities are also being discovered continuously, with the Log4J vulnerability being one of the latest ones to be disclosed and it also happens to be labelled as one of the 'biggest and most critical vulnerability of the last decade' (CISA, 2021). Instead, it would be more appropriate to focus on the most critical web application vulnerabilities outlined in the OWASP Top 10 document. The Open Web Application Security Project (OWASP) is a non-profit foundation dedicated to improving the security of software. The foundation has been maintaining the OWASP Top 10 list since 2003, which provides a ranking of the most critical security risks to web applications as well as remediation guidance for the top 10 security risks (OWASP, 2021). Over the years, this list has become the standard for secure development for web developers and is updated every 3-4 years. Figure 5 shows the 2021 iteration of the OWASP Top 10 list and how it has evolved since the last time the list was published in 2017. The addition of new categories and the changes in rankings of the security risks show that the landscape in cybersecurity is ever evolving.

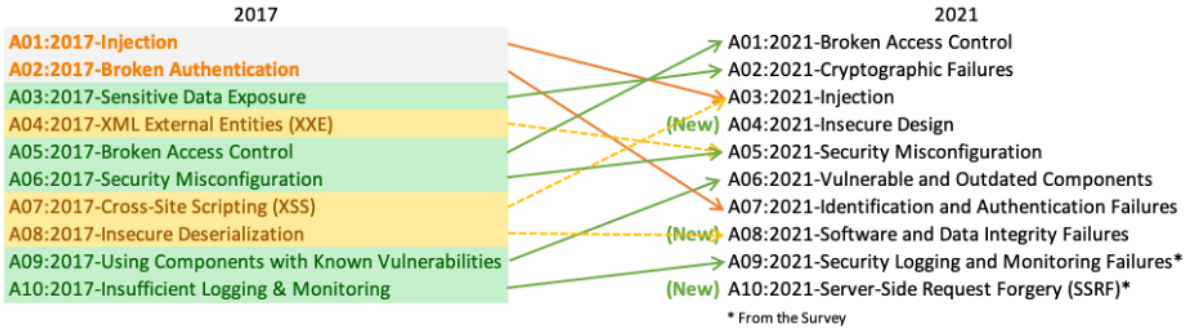

Figure 5. 2021 OWASP Top 10 list (OWASP, 2021).

3.1 Broken Access Control

Access control ensures users of a web application can access resources and perform actions only within the scope of their permissions. Failures in implementing access control properly can lead to unauthorized information disclosure and jeopardize data integrity as users might be able to modify or delete data outside of their permissions.

3.1.1 Security through obscurity

End users typically navigate a web site by clicking through links. Due to this flow, it can be tempting to believe that any hidden resources not directly linked through the webpages will remain hidden. An example of this vulnerability could be an organization releasing quarterly financial results for instance. Internally the company has prepared, proof-read, and finalized the financial report a week before it is going to be posted publicly on their website. When released the report will be accessed at:

```
https://www.finland-top-shop.com/financial/2022-Q1.pdf
```

But since the company uses automated publishing methods, which would release the financial report automatically at a pre-determined date and time (for example on midnight 14th February 2022), the report needs to be hosted at an internal endpoint first. The company chooses to host the finalized report at an internal endpoint, which no one is supposed to know that it exists and visit (since the link is not posted or shared anywhere), such as:

```
https://www.finland-top-shop.com/cms/scheduled_posts/financial/2022-Q1.pdf
```

This endpoint is not protected nor is it authenticated, and the company developers think that no one will access it as the endpoint has not been shared with anyone. This is an example of 'security through obscurity', and is generally frowned upon as there is a high chance that unprotected resources will be detected by threat actors. If resource URL paths are predictable, it can be easy to access sensitive data. Threat actors regularly scan for the existence of potential new resources using scripts before they are publicly available for instance and take advantage. Even if resource URL paths are made obscure and difficult to guess (for example with unique IDs), once the resource has been discovered it can be shared with other parties very quickly and widely. A far better practice is to implement access control for all resources on the web application, even if they aren't meant to be directly accessible by the user.

3.1.2 Failure to restrict URL access

This is a common pitfall in web applications where the security of privileged resources (for example pages that should only be accessible after logging in) has not been properly configured. In such cases, a threat actor can bypass authentication for a URL by making it seem like that the URL has been authenticated already. Threat actors can use URL pattern and URL format knowledge to figure out the URLs for sensitive resources and access pages that have not been securely configured.

A hypothetical example of this can be a web app where the URL's of logged in users are as follows:
A user called John logs in, and their main page URL looks like

```
https://www.very-secure-shop.com/john_login.html
```

A user called Allan logs in, and their main page URL looks like

```
https://www.very-secure-shop.com/allan_login.html
```

The pattern in this URL is easy to deduce.

```
https://www.very-secure-shop.com/{username}_login.html
```

It shows {username}_login.html as the main page for any logged in user. If a threat actor intends to gain access to a privileged account such as the web application's admin user, all they will need to do is visit

```
https://www.very-secure-shop.com/admin_login.html
```

Fixes for such vulnerabilities would entail requiring an authentication check for sensitive pages. Also using naming conventions which are not easy to figure out would also make the job of threat actors more difficult.

3.1.3 Insecure direct object references (IDOR)

This is a type of access control vulnerability which can occur when web applications use data supplied by the end user to access database objects directly. A hypothetical example of this can be a web app that uses the following URL to access the bank account page of a customer by retrieving information from the backend database:

```
https://insecure-bank.com/customer_bank_account?customer_number=123456
```

In this example, the customer_number is being passed as a query parameter from the frontend and is then used directly as a record index in queries performed on the back-end database. In the absence of other security measures, a threat actor can bypass access controls and repeatedly alter the customer_number query value, and view records of other customers. For example, by trying out URL's such as the one below, an attacker might be able to access an existing customer's account.

```
https://insecure-bank.com/customer_bank_account?customer_number=344812
```

or

```
https://insecure-bank.com/customer_bank_account?customer_number=772368
```

This example demonstrates how IDOR vulnerabilities can be used in horizontal privilege escalation attacks (an attack where a user gains the access rights of another user who has the same access level). IDOR vulnerabilities can also be used for vertical privilege escalation, although less common.

How to prevent broken access control?

Since implementing access control can cover a lot of vulnerabilities, there is not a one-size solution to properly implement access control. But to get the basics right, three fundamental aspects should be covered – authentication, authorization, and permission checking. Authentication entails accurately identifying users when the login to an application. Authorization on the other hand entails evaluating actions (such as viewing, modifying, deleting, etc.) that an authenticated user should be able to perform. Finally, permission checking deals with evaluating the user's permission level when the user performs some actions.

While implementing authorization, each user needs to be assigned a role. Different roles typically have different privileges and permissions, for example an administrator will have more permissions than a regular user. The more complex business rules exist in an organization regarding different levels of privileges – the more granular permission schemes need to be implemented. In addition to authorization schemes, ownership of resources also needs to be established since there might be cases where certain resources such as documents, records and other files belong to a set of users and should not be accessible by other sets of users without permission. Finally, access control schemes can be defined as a set of policies, which essentially white-list or black-list sets of users from performing certain actions.

As one can imagine, designing a proper access control while considering all the points above, can be a complex process. Some guidelines that can help in the process are:

- Deciding what are the biggest risks to your application and focusing on contingencies for those risks.

- Before starting to implement access controls throughout your application, it is recommended to first design and document the proposed access control scheme. A well thought out and agreed upon set of rules makes implementation of access controls an easier task.
- The actual implementation of the access control scheme should be standardized and versatile. There should be a standard method use throughout the application which evaluates access control decisions. This can be achieved by specialized components or a separate API which handles web access path, permission, and authentication checking. A security layer between the user interface and the database should also be implemented using a stored procedure layer or functions, which guarantees that data integrity is preserved as user queries are executed in a consistent manner.
- Finally test the access controls in your application thoroughly. The testing procedures in place should be comprehensive and genuinely attempt to find vulnerabilities in your applications access control scheme. It also helps a lot if while testing, one assumes the mentality of an attacker and try and attack the web application in a similar manner an attacker would.

3.2 Cryptographic Failures

Previously termed Sensitive Data Exposure in the OWASP 2017 list (Figure 5), which only described a symptom rather than the root cause, has now been renamed to Cryptographic Failures in the OWASP 2021 list. Since sensitive data can be exposed in several ways, by renaming the vulnerability it is now quite clear that the focus is on failures related to poorly implemented cryptography measures.

The definition of this vulnerability essentially states that sensitive data is either not protected or protected by insufficient cryptography – this applies to both data at rest and data in transit. There are three key terms in this definition – sensitive data, not protected and insufficient cryptography. Let’s look at each of them individually to better understand the overall vulnerability.

3.2.1 Sensitive Data

Any data or information that an organization would like to keep private and not accessible by unauthorized parties should be regarded as sensitive data. This can include a wide-ranging set of data such as environment variables, keys, passwords, usernames of a web platform, to user data such as email addresses, names, social security numbers, or financial data such as credit card numbers, ban account numbers, etc. Based on this list, it is obvious that some sets of data are more sensitive than others. Data such as passwords should encrypted in a manner that it is not possible to recover it in plain text. Other data such as email addresses and names, probably is ok to store as plain text, but most likely legislations such as GDPR mean that such information should also be considered sensitive and protected to some extent. The basic rule of thumb is that, if there is any doubt whether some data is sensitive or not, then it is best to provide it a level of encryption which still allows the data to be used in the application with reasonable ease.

3.2.2 Not Protected Data

Typically, once a data protection strategy is in place, any actions that diverge from the strategy can be an indication that data might exposed or mishandled. For example, in a secure messaging app the data protection strategy is probably defined by actions such as messages being encrypted at rest and transmitted over an encrypted connection – so any actions contrary to this strategy would mean data is not being protected any longer. Some common pitfalls which can lead to sensitive data being exposed and mishandled are:

- Storing data in plain text
- Logging data in plain text
- Caching data in plain text
- Transmitting data in plain text
- Transmitting data over an unencrypted connection (allows for eavesdropping or man in the middle attacks).
- Committing data into source control (for example storing application environment variables, passwords, and secrets in GitHub). Research from F-Secure has shown that exposed credentials can be abused by attackers as quickly as 9 minutes from the time the leak takes place. This is possible since attackers are constantly scanning public repositories, and other internet facing services to monitor for exposed credentials (Knott, 2021).

3.2.3 Insufficient Cryptography

The basic concept behind incorporating cryptography, is to utilize ciphers that are impossible to crack within a reasonable amount of time using the computational power available today. This does not necessarily mean using ciphers which are impossible to crack in absolute terms, but rather using ciphers that make efforts to crack the cipher difficult and impractical in terms of time and effort required. So, by definition insufficient cryptography is cryptography that does not provide sufficient levels of security as it can be easily compromised by an attacker with reasonable time and effort.

The topic of cryptography is a research field in itself and creating cryptographic cyphers should be left to the experts. Cryptographic functions require a lot of research and design by specialists, and it is extremely difficult to create one. Any product flaunting cybersecurity features built in house (especially companies which do not specialize in cryptography) should probably not be trusted to be secure – as there is a good chance that the cryptographic functions have some fundamental flaw and are not secure by design. So, in short when it comes to picking cryptographic functions, it is best to trust the expertise of specialists and not invent your own cryptographic functions.

Even when an appropriate cryptographic function has been identified and taken into use in your product, it is highly advisable to stay up to date on news related to cryptography to monitor when an algorithm is no longer considered secure. Considering Moore's law which states that - the number of transistors on a microchip doubles every two years – the undeniable trend is that over time computers have become faster and more efficient. Because of this trend of computers becoming more powerful, the list of the most secure cryptographic functions keeps changing occasionally. For example, in the 1990's, the DES algorithm (Data Encryption Standard) was widely used and considered secure, but modern computers can crack this cryptographic function with ease. The DES was phased out in early 2000's and was superseded by the AES family of algorithms (Advanced Encryption Standard). The AES is currently considered to be secure, but with the rise of quantum computing and other advances in hardware this might not be the case in the near future.

There are also other important considerations while using a cryptographic function correctly so that security is not compromised over time. Aspects such as using unique salts while hashing

data, having strong encryption keys and regularly rotating encryption keys need to be strictly followed.

- Using unique salts – A salt is a piece of random data that is added to sensitive data (such as passwords) before the sensitive data is hashed. For example, the password

  `password`

  when hashed using the MD5 algorithm produces the hash

  `5f4dcc3b5aa765d61d8327deb882cf99`

  Although it is not possible to decrypt a hash back to the original input, it can be cracked, simply by comparing the original hashed password with hashes of sample passwords obtained from a password leak of another compromised website for example. Password hashing can be made more secure by using a randomly generated salt for each user. So, in our example – a salt would mean appending or prepending a string such as

  `S3cret`

  to the original password

  `S3cretpassword`

  and then hashing it. This adds a layer of complexity to the hash and an attacker will have to crack hashes one at a time using the respective salt, as compared to calculating the hash once and checking against every stored hash. This makes cracking hashes much more time consuming and difficult.
- Using strong encryption keys – A weak encryption key is just as bad as a weak password. A weak encryption key looks like:

  `ENCRYPTION_KEY='SECRET'`

  Whereas a strong encryption key looks like:

  `ENCRYPTION_KEY='NHVvX4YMuDd4PESGh7szr2KfZCSYYTt78SnnRTRTSq'`

  When attackers get their hands on encrypted data, they carry out a dictionary attack on the data which essentially uses a list of commonly used terms to try and decrypt the data. A dictionary attack can be carried out much faster compared to a brute force attack which is required when the encryption key is much harder to guess.
- Encryption key rotation – Encryption keys are prone to being compromised. Accidentally sharing them or committing to git can be one of the reasons. Additionally, when team members leave a project, then too is a good time to be proactively rotating encryption keys. The basic rule of thumb is that whenever an encryption key is

suspected that it is known by some unauthorized party, then the encryption key should be rotated and updated. The rotation of encryption keys should be an automated task, where a backup of the encrypted data should be backed up, perform the rotation, encrypt the data with the new key and then finally delete the backup data.

3.3 Injection

Injection attacks refer to an attacker's attempt to supply untrusted input to an application, which then gets processed by the application's interpreter and alters the execution of code in a manner which can lead to unauthorized data access and in the worst-case full system compromise.

Injection attacks have been around for a long time and in the previous iteration of the OWASP 2017 list, injection attacks were categorized as the most critical and dangerous types of attacks for web applications.

The main factor behind injection attacks is insufficient validation of user input. Since most web applications accept some sort of user input – the default strategy should be to treat all input as "untrusted" until proven otherwise.

Some common types of injection attacks are:

- SQL injection – SQL injection vulnerabilities arise when database queries are constructed in an unsafe manner and untrusted data is interpreted as part of the SQL query. For example, in a Java application that lets us fetch users based on their email, the following scenario where we construct a query with string concatenation is highly dangerous:

  ```
  String sql = "SELECT * FROM users WHERE email = '" + email + "'";
  ```

  In the example above, the concatenated email can refer to unsafe user input, which means that malicious parameters can easily be passed through. Instead, it is much safer to use parametrized statements, which ensure that user inputs (i.e., parameters), passed into SQL queries are treated in a safe manner.

  ```
  // Prepare the SQL statement and specify the parameter.
  String sql = "SELECT * FROM users WHERE email = ?";
  // Generate a prepared statement with a placeholder parameter.
  PreparedStatement stmt = conn.prepareStatement(sql);
  // Pass email value into the statement as the first parameter.
  stmt.setString(1, email);
  ```

- Cross site scripting (XSS) – This vulnerability abuses HTTP parameters to inject malicious JavaScript code in a web application.
- Command execution – Similar vulnerability to XSS scripting, which abuses HTTP parameters to inject shell commands that run on the server.

3.4 Insecure Design

When it comes to designing secure web applications, a lot of planning and careful thought is required before code implementation even starts. In this phase the blueprint of the application architecture is created. Once that is done, one also needs to consider what potential threats the application will face, what the attack surface looks like and how to mitigate the threats. This step is called 'threat modelling' and it helps identifies what threats the application will face, understand how and where malicious input might enter the application, anticipate conditions under which the application can fail, and consequently plan how to fail in a controlled manner. Typically threat modelling is kicked off by drawing a sketch of a data flow diagram which illustrates how data is stored, processed, and flows through the application along with interactions and trust boundaries that make up the application.

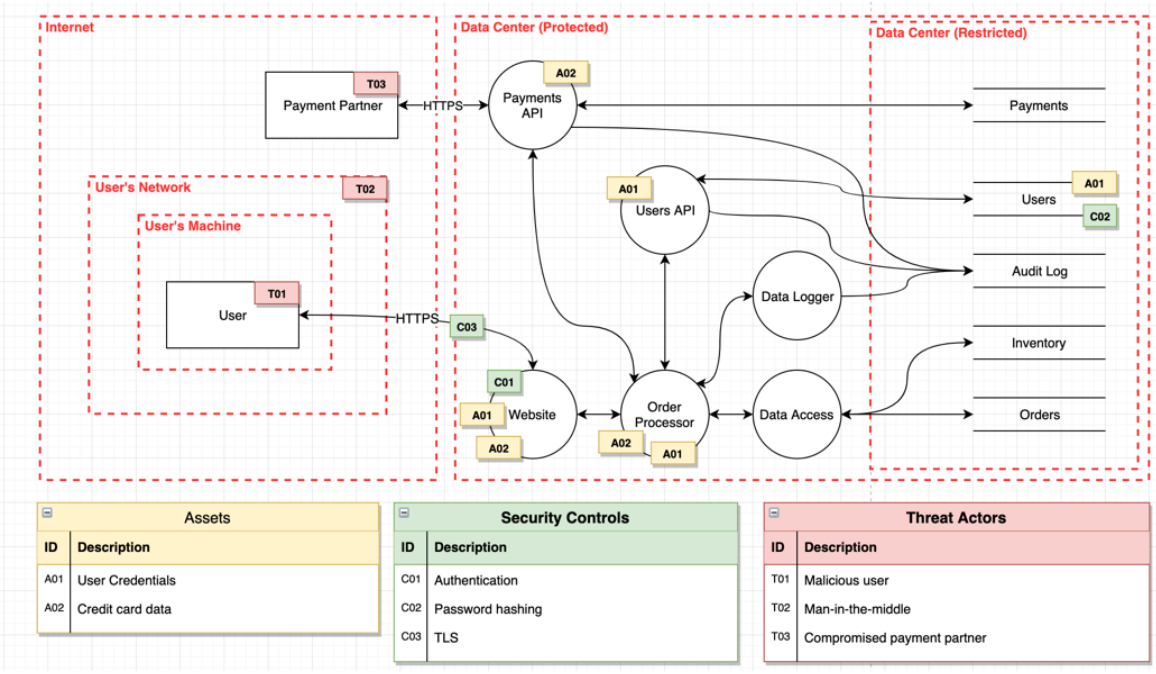

Figure 6. Threat modelling data flow diagram (Diagrams.net, 2019)

With the help of a data flow diagram, it is easy to point out features in the application that might act as access points for attacks and detect instances of safeguards that are missing. An accurate data flow diagram attack vectors and structural vulnerabilities in the application's

architecture can be detected beforehand and immediate risk mitigations can be carried out. Based on the findings of the 'threat modelling' phase, concrete security requirements for the application can be defined. In most cases, the key to adhering to security requirements is a well-functioning SDLC – one which has issue tracking (JIRA, Azure DevOps Board, etc.), source control (Git), a build process, CI/CD pipelines with unit and integration tests, and finally a good code review system. It is a good idea to automate code deployment as well and always having the ability to roll back the latest code release in case of issues should also be a default strategy. There are certain tenets of secure design which should be followed as closely as possible:

- The principle of least privilege should be applied, and each process, program and user should only operate with the minimum amount of privilege required for their tasks. This strategy will ensure that an attacker is only able to do limited damage in case they compromise the application.
- Data should be encrypted with strong cryptographic functions both in rest and in transit. This strategy will rule out man-in-the-middle attacks and provide an extra layer of protection for your data.
- All user input should be treated as untrusted by default and should always be validated to ensure that it matches the expected format. If the user input does not match the expected format, then it should be rejected immediately.
- When the application eventually fails, it is critical that it fails securely. This means that at the time of failure, error messages need to be consistent and uniform and more importantly not divulge any internal architectural details. It is a common tactic for attackers to try and cause an error in the application only to read the output of error messages, since in many cases internal details and mechanisms of the application are leaked in this manner.
- It is also highly advisable to implementing real time monitoring and event logging into your application. This will help you get an overview of the type and volume and traffic reaching the application servers and help distinguish between real users and automated traffic generated by potential attackers and provide you a prompt to take appropriate actions to protect the application.
- Finally, the most important thing is to keep learning from mistakes that eventually happen. Whenever a vulnerability in the production environment is detected, it should

be carefully analysed to figure out what safeguards and processes were overlooked, to make sure they are not repeated.

3.5 Security Misconfiguration

Improper security settings are a major cause for vulnerabilities. At the end of the day, even the most securely coded application can be vulnerable if the security configurations are not defined well enough. In fact, misconfigured security settings are an extremely common cause of security vulnerabilities and are quite easily detected and taken advantage of by attackers. A lot of databases, servers and content management systems come with default accounts – which allows developers to get started quickly with development. But this can also cause accidentally shipping software with default accounts enabled, which attackers are quick to sniff out and take advantage of. An example of this was version 3.23.2 of MySQL which had a default root account that did not require a password.

Developers heavily rely on tools such as interactive consoles and debugging tools, as they make errors in code easier to track and catch. But after development has been completed and the code is ready to be deployed, it is a really good idea to disable all development tools and turn off client-side error reporting. The rule of configuring production releases more securely than pre-production releases should be strongly followed. In some cases, it is advisable to securely configure pre-production environments as well, if these environments contain sensitive client data.

If your application has administrative or management interfaces, these need to be properly configured as well and should not be left open to the world. For example, if authorized users can manage production data through these interfaces, consider adding extra layer of security such as two factor authentication and restricting access to internal networks only.

3.6 Vulnerable and Outdated Components

Modern web development relies heavily on open-source frameworks, libraries and tools created by other developers. The solution to common problems in development is to simply take in to use a tool that has already been developed, rather than implementing the solution from scratch. Pretty much all the major programming languages and frameworks make it easy

to install and run third-party code. For example, in Python one can get packages using the pip tool, for Ruby there is Gems, and for JavaScript there is the npm tool.

Typically, when selecting libraries and packages, developers pay attention to how old and mature the library is, how actively developed it is, its popularity rating (the number of stars it has gained), and the number of weekly downloads it has. All these aspects combined paints a picture of how reliable and trustworthy the library is. Despite these superficial checks, rarely do developers check the source code of libraries to make sure that the code does not contain vulnerabilities or in the worst-case scenario - that the code might be hiding something malicious. Since these libraries are open source, developers tend to rely on the fact that some other user of the library would have detected something amiss if indeed there was some vulnerability. However, threat actors work in very sophisticated ways and have developed methods to avoid detection. Attackers especially find popular libraries very attractive, since it provides a fairly easy way to impact many applications. One of the more recent cases of vulnerable components was the Log4J logging tool used in Java-based applications. Log4J is a very popular tool and is used by over 35,000 Java packages, which represents 5% of all packages in the Java ecosystem. Towards the end of 2021, a vulnerability in the Log4J package was discovered which allowed attackers to perform remote code execution by exploiting JNDI lookups in the library (CISA, 2021). This major and very critical vulnerability affected millions of applications, and everyone affected had to immediately patch their applications. In 2020, a major supply chain attack on SolarWinds (the producer of a network management tool called Orion), attackers created a malicious security update which would install a backdoor in everyone's system who downloaded the update. Eventually this updated was installed by more than 18,000 users many of them in the government sector and large corporations. Further instances of compromised software include the 2015 discovery of a modified version of XCode – the default development tool for the Apple ecosystem. The modified version of XCode was designed to steal system information and inject malicious payloads into all apps built using the infected tool. Some infected apps built using the modified version of XCode were published in the Apple App Store and subsequently downloaded by users.

3.7 Identification and Authentication Failures

This vulnerability is related to improper implementation of authentication and identification related components in an application. These components are used in identifying user logins,

user authentication, user session management, among other things. If these components are not properly implemented, then attackers may be able to exploit user passwords or session tokens and take over user accounts and identities temporarily or permanently. An application that has authentication flaws would exhibit some of the following bugs:

Password mismanagement

- Application allows users to use weak passwords
- Application user passwords can be cracked using brute force or other automated attacks
- Application uses weak or ineffective user account credentials recovery processes
- Application stores passwords in plain text, or passwords are weakly encrypted
- Application does not offer multi-factor authentication

Session mismanagement

- Application exposes session IDs in the URL
- Upon successful login the application does not rotate session IDs
- Application does not properly invalidate session IDs

How to prevent identification and authentication failures?

Safe password management is an essential aspect for any application, but it is quite easy to get it wrong. Based on the business rules of an application, if allowed one should offload the responsibility of authentication management to trusted third parties such as Google, Facebook, GitHub, or LinkedIn for example. This makes the login process also very convenient for users, since they can now use the credentials of the third party (e.g., Facebook) to log in to your application, and subsequently don't need to remember another set of credentials. Authentication using third parties isn't always suitable depending on the business context. In cases where a development team needs to implement their own authentication system, there are a few guidelines that should be followed:

- Require a minimum password length of 10 characters
- Allow users to have long passwords (e.g., up to 64 characters if the user wants). Some applications cap the password length at 18 characters for no good reason at all. A long simple password (e.g., a password containing only lowercase letters) is more secure than a short complex password (e.g., a password containing numbers, upper and lowercase letters, and symbols). Figure 7 shows the time it takes to brute force various passwords.

Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	Instantly	Instantly
7	Instantly	Instantly	2 secs	7 secs	31 secs
8	Instantly	Instantly	2 mins	7 mins	39 mins
9	Instantly	10 secs	1 hour	7 hours	2 days
10	Instantly	4 mins	3 days	3 weeks	5 months
11	Instantly	2 hours	5 months	3 years	34 years
12	2 secs	2 days	24 years	200 years	3k years
13	19 secs	2 months	1k years	12k years	202k years
14	3 mins	4 years	64k years	750k years	16m years
15	32 mins	100 years	3m years	46m years	1bn years
16	5 hours	3k years	173m years	3bn years	92bn years
17	2 days	69k years	9bn years	179bn years	7tn years
18	3 weeks	2m years	467bn years	11tn years	438tn years

Figure 7. Time it takes to brute force a password (Neskey, 2022)

- Avoid prompts for periodic password resets as it is much better to have one good secure password than changing less secure passwords frequently.
- Do not allow users to select passwords which are already listed in data breaches. One can use various open sources such as Daniel Miessler's security lists which contain over 14 million entries of leaked credentials (https://github.com/danielmiessler/SecLists). Credential stuffing with stolen passwords can be used to gain unauthorized access to user accounts, simply because users reuse the same username and password combination in several applications. So once one credential pair is known, then it can "sprayed" on to other applications using automated tools to check if the credentials work on another application as well.

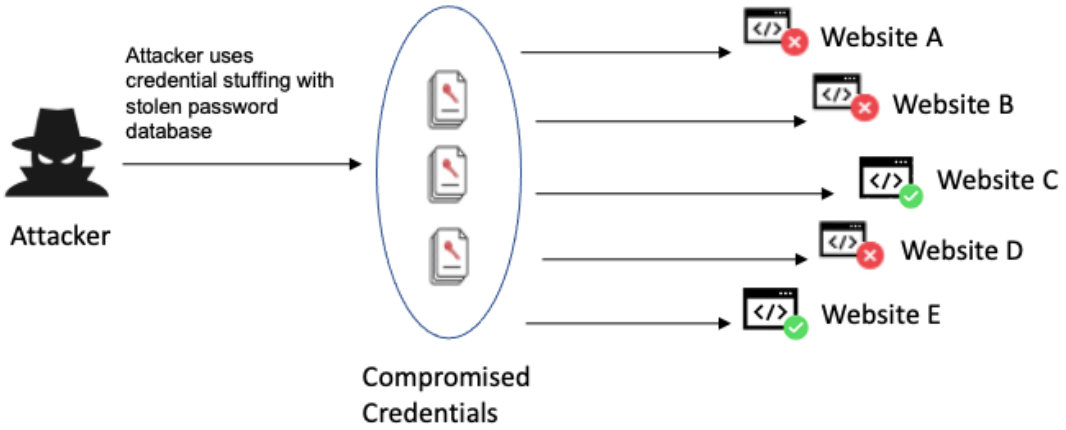

Figure 8. Credential stuffing using a database of leaked credentials (F5, 2022 Chapter 7)

- The application should also provide some options on password fields so the user can view the password that they have typed, and users should also be able to paste passwords into the password field.
- Passwords should be hashed with bcrypt hashing function, and it is also recommended to use a salt phrase while hashing.
- Password brute forcing should be avoided by implementing a periodic lockout of the account after a certain number of incorrect login attempts.
- Login attempts from new or untrusted devices should require an additional check.
- When implementing the "Forgot Password?" functionality, return an identical message for both existing and non-existent accounts. So, instead of indicating to the user that their email was or was not found during the password reset phase, the application should instead show an identical message such as 'Password reset initiated, and email will be sent to the email address provided if found'. This way the application does not provide any information about existence of the account in question and whether the attacker should try another email address.
- The URL for resetting the password contains tokens or code identifying the user account. These tokens should be generated using a secure cryptographic algorithm, tamper proof against brute force attacks, single use and be valid for a short amount of time only.
- Enabling two-factor authentication (2FA) should be highly recommend to an application's users as it adds another layer of authentication. In contrast single-factor authentication means only having a username and password. In 2FA, the user needs to be able provide credentials from two of the three categories 1) PIN or password 2) phone or authenticator device and 3) fingerprint, voice or facial biometric.
- When discussing session ID's, the application should not be passed in query strings or in the body of POST requests. Session IDs are recommended to be passed as HTTP cookies.
- The application should only accept server generated session ID's and they should also be regenerated at every successful log in instance.
- The application should periodically timeout and replace old session ID's, as a second layer of security in case the session ID's get leaked.
- When a user logs out, the application should make sure that the previous session ID is marked as obsolete and is no longer usable.

3.8 Software and Data Integrity Failures

The official definition states that software and data integrity failures relate to "code and infrastructure that are not protected against integrity violations" (OWASP, 2021). The typical things that could go wrong under this vulnerability category are:

- The software application relying upon third party plugins, libraries, or packages from untrusted sources, repositories, and content delivery networks (CDNs). There is always a chance that even the most widely used and trusted repositories can become compromised.
- Insufficiently secured CI/CD pipeline's can potentially be accessed by unauthorized parties and be an entry point for malicious code, or full system compromise.
- Leaving settings on default for features such as auto-update functionality can also pose a risk as downloading and applying software updates without sufficiently verifying the data integrity of the software update is a big red flag for critical systems. Threat actors also have the ability to upload malicious updates which are then distributed and run on all installations.

Agile software development has undoubtedly sped up the software development lifecycle. It is nowadays very common for products to be releasing regular updates (which include new features, bug fixes, etc.) on a weekly basis. Although the end user gets a better product at the end of the day, there is also a potential drawback from a security standpoint of such a fast pace of software development. Firstly, software developers are increasingly relying on third party plugins, libraries or modules that can potentially come from untrusted sources or contain malicious code. Secondly, security teams also need to function inside a short timeframe, and their ability to check the quality code in pull requests, configuration changes, access role changes, auditing of vulnerable components, or detect identification and authentication failures built into the application eventually suffers. With companies focusing more and more on agile development, it is easy to overlook the time required to fully verify a software release, and due to this phenomenon, many development teams do not have sufficiently good integrity processes that allow them to analyse and detect security failures.

The SolarWinds Orion attack is also one of the notable examples of software and data integrity failures. Threat actors were able to gain access to the SolarWinds backend by means of

password spraying, after which they proceeded with inserting malicious code into the SolarWinds CI/CD pipeline. The malicious code that the threat actors eventually introduced into the CI/CD pipeline was a component called *SolarWinds.Orion.Core.BusinessLayer.dll*. The component was masked extremely well, and it resembled an authentic software component and due to security gaps in the pipeline, it was signed off as an authentic SolarWinds software update with legitimate digital signatures. Due to the nature of the vulnerability, the resulting software update was considered a legitimate update as far as the SolarWinds customers were concerned, and once installed the threat actors were able to gain access to the customer's networks. This security incident would eventually turn out to be one of the major security incidents of the year, and it shows that simple things such as failures in monitoring the CI/CD pipeline can lead to critical systems being compromised and data being leaked. Figure 9 below shows an attack scenario in which insufficiently secured CI/CD pipelines can be an entry point for threat actors.

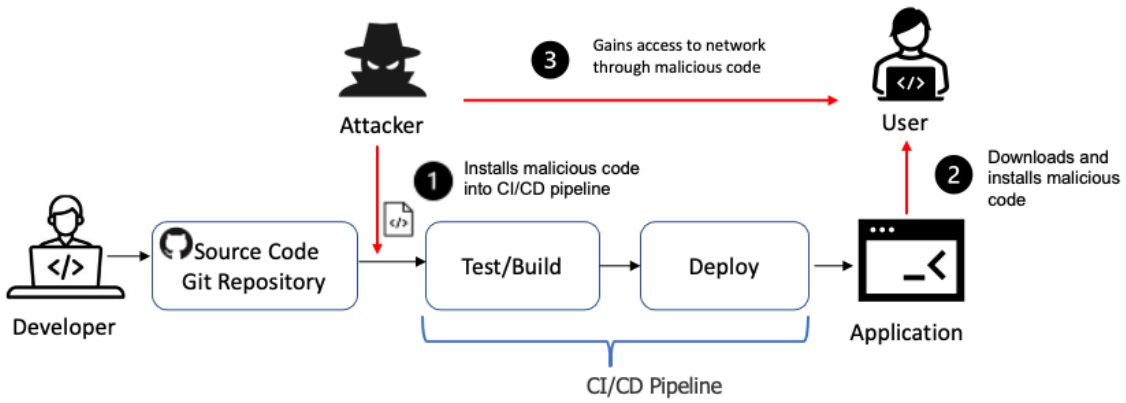

Figure 9. Attack involving insufficiently secured CI/CD pipelines (F5, 2022 Chapter 8)

How to prevent software and data integrity failures?

There are a few guidelines that one can follow to reduce the likelihood of software and data integrity failures. These include:

- Verifying modules, packages, software updates with digital signatures to make sure that the data is from the source that you expect it should come from and has not been tampered with. This is not always fool proof since even trusted repositories can be corrupted, forcibly taken over or the original author deliberately sabotages the library for nefarious purposes.
- Hosting an internal known-good and vetted repository as a proxy

- Using yarn audit, OWASP dependency check or other software supply chain security tools to check for known vulnerabilities in the components that your application is using.
- Having an established code review process in place within your infrastructure management team, will most likely reduce chances that malicious code or configuration changes making their way into the CI/CD pipeline.
- Having granular control over the CI/CD pipeline with proper access controls, segregation, and configuration, is also recommended to ensure the integrity of the software that is built and deployed through the pipeline.

3.9 Security Logging and Monitoring Failures

Logging refers to documenting an application's state at any given time, and can provide information such as user activity, system performance, and errors that occur within the application. Logs are primarily used for keeping track of how the application is being used, application improvement and recovery. For example, in a web server, logs are kept for each HTTP request the web server receives, timestamp of the request, the HTTP method and the HTTP response code that the web server sends back.

In most cases, the probability of major security incidents occurring drastically increases due to insufficient logging and monitoring. Insufficient logging and monitoring, makes it difficult to detect suspicious behaviour (such as repeated multiple failed login attempts or automated requests to certain endpoints) and ultimately increases the chances that a threat actor will be able to exploit an application. An attack scenario that involves insufficient logging and monitoring, could potentially look like – an attacker gaining access to an internal network, and once inside the network the attacker starts scanning for vulnerable components and unencrypted sensitive data. Since there is no logging and monitoring taking place, the attacker can keep obtaining information over a long period of time and the breach continues undetected. Typically, breaches are only detected when the attacker makes the data leak public by selling the data on the dark web or contacting the data owner directly for a ransom. In some cases, the initial target system is only an intermediary step of a large scale attack whose ultimate target is a totally separate system, and insufficient logging and monitoring ultimately makes the life of an attacker simpler as they can then carefully plan and execute their next steps. The fact that many security incidents are detected after a significant amount of time (~200 days), shows the

importance of proper security logging and monitoring. Again, the SolarWinds Orion security incident can be highlighted as investigations have shown that the breach could have started as early as October 2019 (SOURCE).

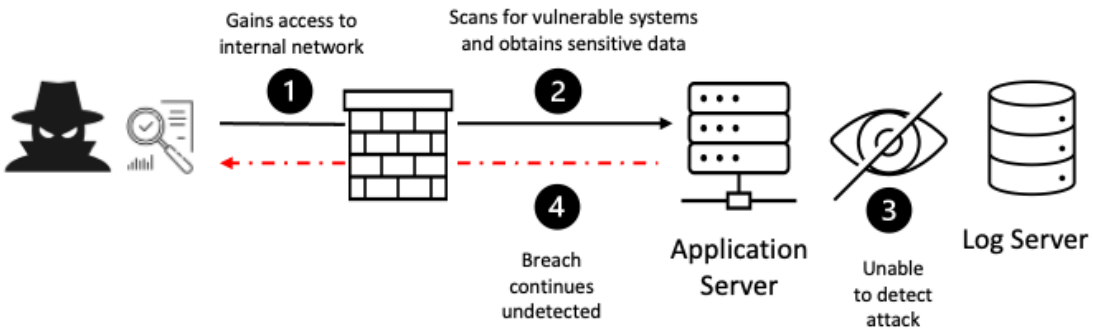

Figure 10. Attack scenario involving insufficient logging and monitoring vulnerability (F5, 2022 Chapter 9)

How to prevent security logging and monitoring failures?

There are a few guidelines that one can follow to improve the security posture of an application by properly implementing logging and monitoring. These include:

- Adding logging statements (console.info, console.warn, console.error) to important parts of your code. To make each logging statement more informative and traceable one can add a timestamp, file name and line number.
- Make use of logging packages (although the popular Log4J package caused a major security incident at the end of 2021), as they are more powerful and provide a lot of features such as filtering, consolidating logs and providing a dedicated dashboard for logging and monitoring.
- Retain log file for as long as possible. Old logs can be used for troubleshooting purposes and for analysing attack patterns. In some industries (for example healthcare), keeping a record of all past logs is required by law.
- Since logs files can potentially be a target for threat actors, avoid logging sensitive information such as passwords, SSN's (social security numbers) or other personally identifiable information. If sensitive information does indeed be logged, make sure to either pseudonymize the sensitive information or store your log files in an encrypted manner.

- Avoid being drowned in log files and avoid attempting to read all the logs in raw format. Instead filter or monitor logs to detect trends in the log output.
- Monitoring should be used as an indicator for ongoing or attempted cyber-attacks. Look out for suspicious errors or automated web server requests.
- It is also a good idea to have a rotating role within the development team, where one member has the monitoring shift per week and they should check once a day the monitoring dashboard and report normal (number of users, response time and general health of the application) and abnormal activity (unusually high number of requests, unauthorized access attempts, errors) to the team.
- For monitoring of critical components of the application such as data pipelines, user authentication components, payment portal, etc. it is recommended to set up monitoring alerts. When the correct condition is met, then an alert will arrive by email, SMS or to a messaging platform of your choice such a Slack or MS Teams for instance. This helps detecting component failures or attack attempts promptly, rather than relying on 24/7 manual monitoring of logs.
- Finally, make sure that the development team has an internal wiki page for documentation related to a response plan (which include trouble shooting steps for restarting servers or adjusting load balancers) when security logging and monitoring alerts arrive.

3.10 Server-Side Request Forgery

Server-Side Request Forgery (SSRF) is a vulnerability which occurs when a web application server is tricked in to providing access to unauthorized resources (for both viewing or modifying purposes) to an attacker. Applications that support reading user supplied data are the primary target of this type of attack. In this attack, the attacker crafts a malicious request containing the details of the target (for example through a URL), which is initially sent to the main web server of the application. The server-side code then handles the attacker's request and triggers a secondary request to the target which are typically internal services, databases, or other internal servers on the network which are not facing the public internet. The response from these internal services, which contains unauthorized data, is then relayed back to the attacker completing the attack.

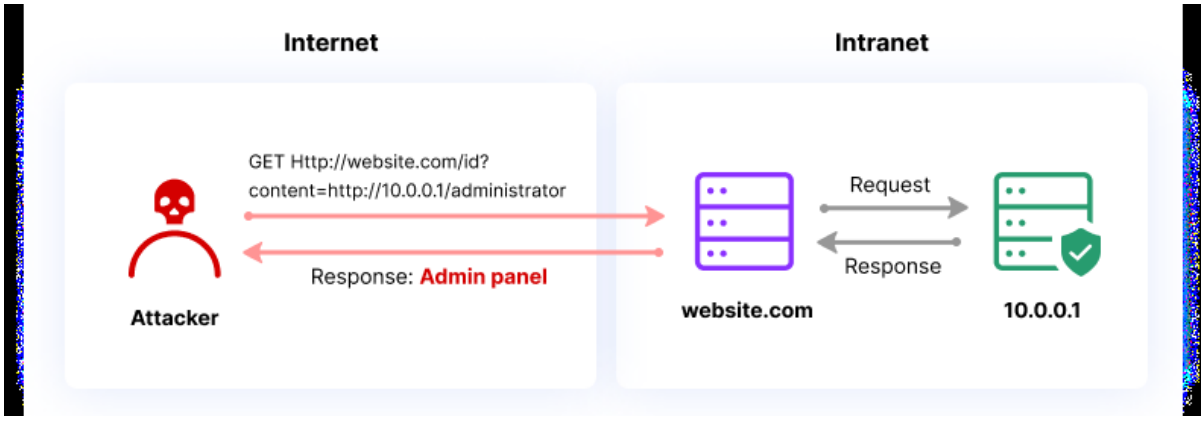

Figure 11. Attack scenario involving a SSRF attack (Imperva, 2019)

In the attack scenario depicted in Figure 11, the website.com application serves content based on the 'content' query parameter. A normal request could look like:

```
GET http://website.com/id?content=10
```

The website.com server would respond to the above request with the data related to the query. But hidden behind the network is also another server with the address 10.0.0.1 which is not directly accessible from the internet, and also happens to be the server responsible for generating the admin panel view. This internal server normally only communicates with the website.com server. Now, the attacker modifies their initial request to:

```
GET http://website.com/id?content=http://10.0.0.1/administrator
```

The website.com server receives this request and tries to resolve it by forwarding the request to the internal server. The internal server itself does not to do any authentication checks, and trusts that the main website.com server has handled authentication. Now the internal server generates the admin dashboard view and responds back to the website.com server, which in turn serves the response to the attacker.

The main risks that SSRF poses are data exposure, Denial of Service (DoS) attacks, and the attacker carrying out reconnaissance and fingerprinting of services. One of the reasons why SSRF related attacks are on the increase is due to the higher usage of cloud services in modern web applications. A SSRF attack to gain access to cloud service credentials (and consequently data stored in that cloud service) is one of the more common motivations for the attackers. SSRF attacks are also used for reconnaissance purposes focused on services running on internal networks. Since established security practices recommend minimizing the attack surface area, only a handful of an application's servers are truly public facing, and the remaining servers reserved for internal functioning and communication are not accessible from external networks.

So, with SSRF attacks it becomes possible to figure out what services might be running on internal networks and based on this information, attackers are able to device further attacks on these internal services to steal credentials or data. DoS attacks on these internal services can also be triggered by SSRF attacks. Since internal services typically receive much lower traffic than public facing services, the resources they are provided to handle traffic is also much lower. So, in a sense it is much easier for an attacker to taken down an application by flooding the internal servers with traffic, rather than directing the traffic to the public facing services. This attack is called an internal DoS attack which can be triggered through SSRF attacks.

How to prevent server-side request forgery?

SSRF attacks can be minimized by following guidelines related to both application security and network security.

Application level:

- Always sanitize and validate all user-supplied input data
- Enforce a certain pattern of URL schema, port, and destination with whitelists.
- Avoid sending raw responses to users
- Disable HTTP redirects from one service to another

Network level:

- Segment remote resource access functionality in separate networks
- Enforce “deny by default” firewall policies or network access control rules which block all but essential intranet traffic
- Log all accepted and blocked network requests on firewalls (related to security and monitoring)

4 Conclusion

The starting point for this thesis was to create a condensed guide on incorporating security best practices in software development and to learn more about the fundamentals of secure web development in the process. In the first half, this thesis attempts to provide a baseline of what cybercrime and cybersecurity are, the actors involved on both sides of cybersecurity, why it is important for developers to be familiar with web security fundamentals, and why organizations need to create a culture of security from the ground up. The second half of the thesis deals with the top 10 web development security risks as identified by OWASP. In this section, theoretical rundowns for each risk are provided, which covers how each risk manifests, the threats each risk poses to an organization, and finally steps to mitigate each risk.

Hopefully this thesis has been able to bring to light the importance of cybersecurity and the importance of building secure web applications. Due to the nature of digitalization and modern society's reliance on web services, cyber threats pose a direct threat on economic, privacy and infrastructure grounds. One could say that these days cyber threats impact pretty much all aspects of modern life. Despite this fact, security is one of the most overlooked aspects of software development. Companies and organizations put a lot of emphasis on rapid product development, faster time to market, lowering development and operation costs – but the outcome of all these business decisions usually turns out be a product with a lot of security gaps, since security is almost an afterthought. Nowadays, with automated tools which probe for vulnerabilities in applications, it is only a matter of time that these products suffer an attack or suffer a data breach, leading to loss of customers and market position.

The security risks and their mitigations covered in this thesis, are considered to be the most critical risks to web applications. The OWASP Top 10 document provides a good basis to start one's journey in secure web development. It must be stated that this document alone will not guarantee that your application will be free of security gaps. With enough time and pressure, any application is hackable and perhaps the only application that has the best shot of not being hackable can be found at - https://github.com/kelseyhightower/nocode
But by covering the bases with the OWASP Top 10, it will guarantee to make the work of threat actors more difficult in breaking into an application. In most cases, having basic security bases covered is enough of a deterrent for threat actors to give up hacking an application.

The target audience of this thesis are software development students, who have recently entered the IT work force or will join soon. In the university environment, the focus has always been on laying a good foundation in software engineering, programming, and product development. But very few courses touch up on the fundamentals of cybersecurity. So where can up and coming developers discover more about web security? Here are some tips:

- Cyber Security Base is a free course series by University of Helsinki and MOOC.fi in collaboration with F-Secure - https://cybersecuritybase.mooc.fi/
- Haaga-Helia UAS also has a few courses covering infrastructures, configuration management, managing Linux servers, data security and pen-testing. - https://terokarvinen.com/courses/
- Great writeup on 'Cracking passwords with Hashcat' - https://terokarvinen.com/2022/cracking-passwords-with-hashcat/
- Platforms to learn web application security such as TryHackMe - https://tryhackme.com/, HackTheBox - https://www.hackthebox.com/ and PortSwigger's Web Security Academy - https://portswigger.net/web-security.
- Physically one can also join local cybersecurity minded communities such as HelSec Ry - https://helsec.fi/, or TurkuSec Ry - https://turkusec.fi/. Both these communities organize regular workshops and monthly events with great topics and speakers.

As a recommendation to educational institutes, it is now a good time to start offering degree programs geared towards the cybersecurity industry, as there is a very big skill shortage the industry is facing. This gap will probably widen as more digitalization takes place globally. Cybersecurity is such a big sphere of competencies and roles, that within the degree program there can be many specializations such as web security, network and system administration, cryptography, compliance and management, pen-testing and many more.

-- To a more secure tomorrow.

5 References

Alexandrou, A. (2019). Cybercrime. In M. Natarajan (Ed.), *International and Transnational Crime and Justice* (pp. 61-66). Cambridge: Cambridge University Press. doi:10.1017/9781108597296.010

Arthur, C. (2013). LulzSec: What they did, who they were and how they were caught. Retrieved December 07, 2021, from https://www.theguardian.com/technology/2013/may/16/lulzsec-hacking-fbi-jail

Bischoff, P. (2021). How data breaches affect stock market share prices. Retrieved December 08, 2021, from https://www.comparitech.com/blog/information-security/data-breach-share-price-analysis/

Bossler, A. M., & Berenblum, T. (2019). Introduction: new directions in cybercrime research.

Boyer, J. (2018). How secure are popular web frameworks? Retrieved November 21, 2021, from https://www.veracode.com/blog/secure-development/how-secure-are-popular-web-frameworks-here-comparison

Brooks, C. (2021). Alarming cybersecurity stats: What you need to know for 2021. Retrieved December 01, 2021, from https://www.forbes.com/sites/chuckbrooks/2021/03/02/alarming-cybersecurity-stats----------what-you-need-to-know-for-2021/

Canadian Centre for Cyber Security. (2021). Cyber threat and cyber threat actors. Retrieved December 07, 2021, from https://cyber.gc.ca/en/guidance/cyber-threat-and-cyber-threat-actors

Cimpanu, C. (2018). British Airways breach caused by the same group that hit ticketmaster. Retrieved December 07, 2021, from https://www.zdnet.com/article/british-airways-breach-caused-by-the-same-group-that-hit-ticketmaster/

CISA. (2019). What is Cybersecurity? Retrieved October 19, 2021, from https://www.cisa.gov/uscert/ncas/tips/ST04-001

CISA. (2021). *Apache LOG4J vulnerability guidance*. CISA. Retrieved April 21, 2022, from https://www.cisa.gov/uscert/apache-log4j-vulnerability-guidance

Contrast Security. (2014). The Unfortunate Reality of Insecure Libraries. Retrieved October 27, 2021, from https://cdn2.hubspot.net/hub/203759/file-1100864196-pdf/docs/Contrast_-_Insecure_Libraries_2014.pdf

DeCiccio, E. (2021). Former NSA Hacker argues Russian government connected to colonial pipeline attack. Retrieved October 27, 2021, from https://www.cnbc.com/2021/05/12/former-nsa-hacker-argues-russian-government-connected-to-colonial-pipeline-attack.html

Diagrams.net. (2019). *Analysing vulnerabilities with threat modelling using diagrams.net*. Blog - Analysing vulnerabilities with threat modelling using diagrams.net. Retrieved April 21, 2022, from https://www.diagrams.net/blog/threat-modelling

Dooley, J. 2017. Software Development, Design and Coding: With Patterns, Debugging, Unit Testing, and Refactoring, Second Edition. Retrieved October 19, 2021.

Europol (2021), Internet Organised Crime Threat Assessment (IOCTA) 2021, Publications Office of the European Union, Luxembourg

F5. (2022). *Secure against the OWASP Top 10 for 2021*. Support.f5.com. Retrieved April 21, 2022, from https://support.f5.com/csp/article/K45215395

F-Secure. (2021). What is... crypto-ransomware: F-secure. Retrieved November 09, 2021, from https://www.f-secure.com/v-descs/articles/crypto-ransomware.shtml

Gallagher, P. (2015). 15-year-old boy arrested in Northern Ireland in connection with TalkTalk Hack. Retrieved November 07, 2021, from https://www.independent.co.uk/news/uk/home-news/talktalk-hack-boy-15-arrested-in-northern-ireland-over-attack-a6709831.html

Gartner. (2021). The top 8 cybersecurity predictions for 2021-2022. Retrieved November 09, 2021, from https://www.gartner.com/en/articles/the-top-8-cybersecurity-predictions-for-2021-2022

Gustafsson, P. (2021). 4 cybersecurity strategies for small and midsize businesses. Retrieved November 19, 2021, from https://hbr.org/2021/09/4-cybersecurity-strategies-for-small-and-midsize-businesses

IBM. (2021). Cost of a Data Breach Report 2021. Retrieved November 21, 2021, from https://www.dataendure.com/wp-content/uploads/2021_Cost_of_a_Data_Breach_-2.pdf

Imperva. (2019). *Server-side request forgery (SSRF): Common attacks & risks*. Learning Center. Retrieved April 21, 2022, from https://www.imperva.com/learn/application-security/server-side-request-forgery-ssrf/

JBS. (2021). JBS USA Cyberattack Media Statement - June 9 - JBS foods. Retrieved October 11, 2021, from https://jbsfoodsgroup.com/articles/jbs-usa-cyberattack-media-statement-june-9

Kemp, S. (2021). *Digital 2021 April Statshot report - datareportal – global digital insights*. DataReportal. Retrieved November 5, 2021, from https://datareportal.com/reports/digital-2021-april-global-statshot.

Kessem, L. (2021). Threat actors' most targeted industries in 2020: Finance, manufacturing, and Energy. Retrieved October 08, 2021, from https://securityintelligence.com/posts/threat-actors-targeted-industries-2020-finance-manufacturing-energy/

Knott, J. (2021). Managing your true attack surface. Retrieved October 18, 2021, from https://www.f-secure.com/en/consulting/our-thinking/managing-your-true-attack-surface

McAfee. (2020). *The Hidden Costs of Cybercrime*. Retrieved November 5, 2021, from https://www.mcafee.com/enterprise/en-us/assets/reports/rp-hidden-costs-of-cybercrime.pdf

Morgan, C. (2021). 2021 Report: Cyberwarfare in the C-Suite. Cybersecurity ventures. Retrieved October 08, 2021, from https://cybersecurityventures.com/wp-content/uploads/2021/01/Cyberwarfare-2021-Report.pdf

Mossburg, E., Gelinne, J., & Calzada, H. (2016). Beneath the surface of a cyberattack: A deeper look at business impacts.

Mouratidis, H., & Giorgini, P. (2008). Integrating security and software engineering: an introduction. In *Information Security and Ethics: Concepts, Methodologies, Tools, and Applications* (pp. 200-210). IGI Global.

Neskey, C. (2022). *Are your passwords in the green?* Hive Systems. Retrieved April 21, 2022, from https://www.hivesystems.io/blog/are-your-passwords-in-the-green

NETSCOUT. (2021). NETSCOUT Threat Intelligence Report Issue 6: Findings from 2H 2020. Retrieved November 15, 2021, from https://www.netscout.com/sites/default/files/2021-04/ThreatReport_2H2020_FINAL_0.pdf

Norfolk Police. (2020). What is cybercrime? Retrieved December 01, 2021, from https://www.norfolk.police.uk/advice/cybercrime/1-what-cybercrime

Nwankwo, W., & Ukaoha, K. C. (2019). Socio-Technical perspectives on Cybersecurity: Nigeria's Cybercrime Legislation in Review. *International Journal of Scientific and Technology Research*, *8*(9), 47-58.

Nweke, L. O. (2017). Using the CIA and AAA models to explain cybersecurity activities. *PM World Journal*, *6*, 1-2.

Osborne, C. (2021). Colonial Pipeline Attack: Everything you need to know. Retrieved October 01, 2021, from https://www.zdnet.com/article/colonial-pipeline-ransomware-attack-everything-you-need-to-know/

OWASP. (2021). *Owasp Top Ten*. OWASP. Retrieved April 21, 2022, from https://owasp.org/www-project-top-ten/

Purcell, A. (2018). 3 key ideas to help drive compliance in the cloud. IBM. Retrieved October 21, 2021, from https://www.ibm.com/blogs/cloud-computing/2018/01/16/drive-compliance-cloud/

Sjouwerman, S. (2019). Seven reasons for Cybercrime's meteoric growth. Retrieved December 06, 2021, from https://www.forbes.com/sites/forbestechcouncil/2019/12/23/seven-reasons-for-cybercrimes-meteoric-growth/

Stengel, R. (2021). *Data Drives the World. You Need to Understand It*. Time. Retrieved December 5, 2021, from https://time.com/6108001/data-protection-richard-stengel/.

Synopsys. (2020). What is the Secure Software Development Life Cycle (SDLC)? Retrieved November 18, 2021, from https://www.synopsys.com/blogs/software-security/secure-sdlc/

Techopedia. (2020). *What is web 2.0? - definition from Techopedia*. Techopedia.com. Retrieved December 5, 2021, from https://www.techopedia.com/definition/4922/web-20.

The United States Department of Justice. (2018). North Korean regime-backed programmer charged with conspiracy to conduct multiple cyber attacks and intrusions. Retrieved November 07, 2021, from https://www.justice.gov/opa/pr/north-korean-regime-backed-programmer-charged-conspiracy-conduct-multiple-cyber-attacks-and

ThycoticCentrify. (2021). 2021 State of Ransomware Survey and Report. ThycoticCentrify. Retrieved October 05, 2021, from https://thycotic.com/resources/ransomware-survey-and-report-2021/

UNODC. (2019). University Module Series: Cybercrime. Retrieved November 17, 2021, from https://www.unodc.org/e4j/en/tertiary/cybercrime.html

Verizon. (2021). 2021 Data Breach Investigations Report (DBIR). Retrieved November 07, 2021, from https://www.verizon.com/business/resources/reports/2021-data-breach-investigations-report.pdfx

VISMA. (2021). What is cyber security. Retrieved October 09, 2021, from https://www.visma.com/cyber-security/what-is-cyber-security/

Walkowski, D. (2019). What is the CIA triad? Retrieved October 14, 2021, from https://www.f5.com/labs/articles/education/what-is-the-cia-triad