

Continuous and semi-automated threat modeling of modern software applications

Håkon Nikolai Stange Sørum



Thesis submitted for the degree of
Master in Information Security
60 ETCS

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Fall 2022

Acknowledgements

I would like to thank my supervisor Associate Professor Nils Gruschka at the Department of Informatics, and my co-supervisors Cody Burkard and Ole Kristian Rosvold for valuable feedback, good discussions, and many interesting conversations while I have been working on this thesis. A special thanks to Cody for first introducing me to Attack Trees and Threat Modeling. Your detailed feedback, and structured approach to review, has always been educational and inspiring.

To Tonje, thank you for always being there with motivation and enabling me to finish this.

I'd also like to thank my family and friends for the support in this period where I have prioritized reading and writing this thesis over other activities. To my mother and father, thank you for helping me keep my priorities aligned and my motivation sustained. To Jørgen, thank you for the late push to get this done. Finally, to Henrik, thank you for proofreading. You have ensured that readers of this will have a more pleasurable time.

To anyone who reads this, thank you for taking the time. Hopefully this can contribute to you and any organization you're a part of start looking into threat modeling. Not only is it fun, but its value is also undeniable.

Contents

| | | |
|-------|--|----|
| 1 | Introduction | 1 |
| 1.1 | Modern systems development | 1 |
| 1.2 | Problem definition | 3 |
| 1.3 | Contribution..... | 5 |
| 1.4 | Research method..... | 5 |
| 1.5 | Scope limitations | 5 |
| 2 | Background | 6 |
| 2.1 | Information Security..... | 6 |
| 2.1.1 | Cyber threat intelligence | 8 |
| 2.1.2 | Attacker taxonomies..... | 8 |
| 2.2 | Cloud computing basics..... | 9 |
| 2.2.1 | Service models | 10 |
| 2.2.2 | Deployment models..... | 11 |
| 2.2.3 | IaC | 11 |
| 2.3 | Cloud Security | 12 |
| 2.3.1 | Cloud vulnerabilities and threats..... | 13 |
| 2.3.2 | Cloud threat actors | 15 |
| 2.3.3 | Cloud security tools..... | 16 |
| 2.4 | Risk management | 16 |
| 2.5 | Threat modeling..... | 18 |
| 2.5.1 | Threat modeling variations..... | 20 |
| 2.5.2 | Attack trees..... | 24 |
| 2.6 | DevOps | 26 |
| 2.6.1 | Integration, delivery and deployment in DevOps | 27 |
| 2.6.2 | Security in DevOps | 27 |
| 3 | Related work | 29 |
| 3.1 | Threat modeling overview..... | 29 |
| 3.2 | Automated threat modeling | 29 |
| 3.3 | Manual threat modeling..... | 31 |
| 3.4 | Threat modeling in an agile context | 33 |
| 4 | Description of solution..... | 34 |

| | | |
|-------|---|----|
| 4.1 | Introduction to method | 34 |
| 4.2 | Proposed method | 35 |
| 4.2.1 | Challenges | 36 |
| 4.2.2 | Algorithm | 38 |
| 5 | Development of Continuous Semi-Automated Threat Modeler | 40 |
| 5.1 | Requirements | 40 |
| 5.2 | Architecture & design..... | 42 |
| 5.3 | Considerations | 42 |
| 5.3.1 | Explanation of DFD | 43 |
| 5.4 | Cloud threat, mitigation, and remediation catalogs | 44 |
| 5.4.1 | Threat Syntax | 44 |
| 5.4.2 | Remediation and mitigation syntax..... | 45 |
| 5.4.3 | Mapping of services to service models | 45 |
| 6 | Case study | 47 |
| 6.1 | Real world scenarios..... | 47 |
| 6.2 | Case study system..... | 48 |
| 6.2.1 | System description | 48 |
| 6.2.2 | Usage..... | 48 |
| 6.2.3 | System design - IaaS | 50 |
| 6.2.4 | System design – cloud native | 51 |
| 6.3 | Threat modeling of case study system..... | 52 |
| 6.3.1 | Manual OnlinePharmacy.V1 threat model..... | 53 |
| 6.3.2 | Manual OnlinePharmacy.V2 threat model..... | 61 |
| 6.3.3 | Threat modeling of OnlinePharmacy.V1 using CSATM..... | 69 |
| 6.3.4 | Threat modeling of OnlinePharmacy.V2 using CSATM..... | 74 |
| 6.3.5 | Comparison of manual and automatic threat models | 78 |
| 7 | Discussion | 80 |
| 7.1 | Manual threat modeling of Online Pharmacy.V1 and Online Pharmacy.V2 | 80 |
| 7.2 | Automated threat modeling of Online Pharmacy.V1 and Online Pharmacy.V2..... | 81 |
| 7.3 | Considerations, discussion and learnings based on CSATM | 81 |
| 7.4 | State of current tooling and comparison..... | 82 |
| 7.5 | Advantages of manual threat modeling | 83 |
| 7.6 | Advantages of automated or partially automated threat modeling..... | 84 |

| | |
|----------------------|----|
| 8 Conclusion..... | 85 |
| 8.1 Future work..... | 86 |
| Bibliography..... | 87 |

Figure list

| | |
|--|----|
| Figure 1 - Shared responsibility model visualized [19] | 12 |
| Figure 2 - Top eleven risks in cloud computing according to CSA Survey and curation [23] | 15 |
| Figure 3 - Example of threat modeling in code using ThreatSpec [49] | 23 |
| Figure 4 - Example of threat modeling as code with Pytm..... | 24 |
| Figure 5 - Attack tree depicting how an attacker could gain access to open a safe. [50] | 25 |
| Figure 6 - Logical design of an automated cloud threat modeling engine..... | 36 |
| Figure 7 - Processing needed for desired result in threat modeling tool..... | 39 |
| Figure 8 - Requirementslist for CSATM..... | 41 |
| Figure 9 - Table describing tags used for identification of system attributes | 43 |
| Figure 10 - Generic DFD showing how output will be displayed | 43 |
| Figure 11 - Example object from the threat catalog..... | 44 |
| Figure 12 - Example object from the remediations and mitigations catalog | 45 |
| Figure 13 - example of mapping from service model to service used in CSATM | 46 |
| Figure 14 - Use Case Overview | 49 |
| Figure 15 - Architecture diagram of first iteration of cloud application..... | 50 |
| Figure 16 - Definition of cloud VPC for system..... | 51 |
| Figure 17 - System design when using cloud native services | 52 |
| Figure 18 - Risk matrix used in manual threat modeling..... | 53 |
| Figure 19 - DataFlowDiagram of OP.V1 | 69 |
| Figure 20 - Resource list from CSATM for OP.V1 | 70 |
| Figure 21 - Excerpt of threats per resource from OP.V1 threat model created by CSATM.... | 71 |
| Figure 22 - mitigations for threats in threat model for OP.V2..... | 72 |
| Figure 23 - ADTree for myRDSCluster and threat Accidental Cloud Data Disclosure | 73 |
| Figure 24 - ADTree for frontEndASG and threat Cloud Storage Data Exfiltration | 73 |
| Figure 25 - DataFlowDiagram of OP.V2 | 74 |
| Figure 26 - Resource list from CSATM..... | 75 |
| Figure 27 - Excerpt of threats per resource from OP.V2 threat model created by CSATM.... | 76 |
| Figure 28 - mitigations for threats in threat model for OP.V2..... | 77 |
| Figure 29 - ADTree for OPv2EmployeeDB and threat Accidental Cloud Data Disclosure.... | 78 |
| Figure 30 - ADTree for LogisticsFunction and threat Exploitation of serverless and container workload..... | 78 |

© Håkon Nikolai Stange Sørum

2022

Continuous and semi-automated threat modeling of modern software applications

Håkon Nikolai Stange Sørum

<http://www.duo.uio.no/>

1 Introduction

In this chapter the problem definition will be discussed, and the research questions for this thesis will be presented.

1.1 Modern systems development

The modern development lifecycle can be characterized as fast-paced, agile and lightweight. Organizations want to lower the time from inception of idea to the point the software is in use by their intended audience. Recent years we have seen developments such as “you build it, you run it” and a high adoption of DevOps-culture in organizations. The speed of development can be too high for security professionals to keep up if they are not well enough integrated in the development teams, or has enabled the development teams to do significant portions of the security work themselves.

With the introduction of cloud computing and its extensive use many organizations now deploy their IT-assets using programming code and code-like languages. This has increased the velocity one can build an IT-environment with, and many IT-departments have introduced development-team inspired workflows. Despite this security-teams are still working in older workflows, often in an ad-hoc fashion addressing what is subjectively viewed as the most pressing need.

It has long been an agreement in the industry that fixing errors in software at as early a stage as possible reduces the relative cost of the error. Often cited is the 2002 paper by the US government agency NIST: National Institute of Standards and Technology “The economic impacts of inadequate infrastructure for software testing” [1]. To enable software and systems developers uncovering and fixing security related errors as early as possible threat modeling is an often-used technique.

Threat modeling is an integral part of the security lifecycle, and is implemented to uncover the threats posed to the system in question. Threat modeling can be performed at many different layers of abstraction, such as at the application, system or infrastructure level. It is most often performed at design-time, and for applications when the development team is trying to create an architecture that most efficiently solves the problem at hand.

Every design and implementation decision taken during the lifecycle of an application can introduce new vulnerabilities, which in turn increases the risk level associated with the asset or system. Threat modeling should therefore happen at the same level of frequency as these decisions, and not as a standalone exercise.

The interest in threat modeling has increased dramatically recently. In a survey commissioned by cyber security vendor SecurityCompass [2] 79% of respondents said that threat modeling is a top priority for 2021, and 56% of respondents said the main challenge was difficulty of automating the process. 43% of respondents stated that lack of consistency was an issue, 41% said too much time consumption stopped them from performing threat modeling and 37% stated the process was too manual as challenges. This underpins the need for more research and development both in the cyber security industry as well as the research community on the issues.

Why traditional threat modeling might not scale in modern cloud-based systems

Cloud infrastructure can be more dynamic than traditional systems, additionally it can converge with the traditional infrastructure of an organization. This means the traditional- and cloud-systems are now interconnected and interface. This adds another layer of complexity, which increases the effort to secure the system. Traditional threat modeling typically relies heavily on manual labor and expertise, and requires a large investment of resources to conduct. This investment might be acceptable in a static world, but if the infrastructure and system changes at a rapid pace so must the threat model. There is a need for frameworks and tools to reduce the resources required to have an up-to-date threat model for the dynamic systems of the cloud computing paradigm.

When done right threat modeling can say something about whether the system in question is sufficiently upholding the security requirements or goals of an organization. If the threat modeling uncovers a discrepancy between the requirements and the design, then new security controls can be put in place. This can also allow an organization to base their monitoring and threat detections on the known weaknesses or dark spots in their systems.

1.2 Problem definition

The value of threat modeling is undisputed in the industry. Michael Howard, Sr Principal Consultant Cyber at Microsoft said: “About 20 years ago, I made a comment to a journalist: “If we had our hands tied behind our backs (we don’t) and could do only one thing to improve software security... we would do threat modeling every day of the week”. That comment is as true today as it was then” [3].

With distributed teams, agile-workflows and everchanging systems the current approach to threat modeling is outdated. In many organizations the security team is responsible for creating and maintaining the threat model of the organizations systems. Due to not being integrated into the development of software in a suitable manner, and the speed of change, this leads to an outdated threat model. Central entities to the threat modeling process can often become bottlenecks without any ways other than increasing the headcount to scale. The threat model should be an important factor in any security related decision to have a risk-based approach to cyber security. Organizations can have a dynamic attack surface when deploying applications to the cloud which must be continually addressed. One way to address this is with threat modeling.

While security testing is a reactive practice performed where vulnerability assessment is done after the code is written, threat modeling can be used to find design level flaws and errors, and map design level mitigations prior of the fact. Threat modeling is about thinking like the adversary, and therefor when done in a manual fashion will introduce a personal bias from the modeler. By automating parts of the process, and developing a library of threats, adversaries and vulnerabilities relevant to the organization the bias impact on the threat model can be reduced and the quality of the model will increase.

Despite the value of the practice, field is lacking standardization pertaining to threat modeling, and as a result a lot of attempts at creating “the best” threat modeling framework. This is a problem as standards allow security professionals to share, improve and streamline work across organizations. A 2021 study [4] found that security activities conducted early in the development life cycle had an outsized impact compared to the ones taken later in development. The same study found that one third of all respondents had never conducted threat modeling. This is a big problem for software security as large design flaws can go

undetected and require larger investments to mitigate or remediate later as seen by the asymmetry of impact on when security activities are applied.

In May 2021 President Joe Biden signed Executive Order (EO) 14028 “Improving the Nation’s cybersecurity”. This EO mandates US government agencies to improve their cyber security posture. Section 4 of the EO directs the National Institute of Standards and Technology (NIST) to create guidelines for US Government Agencies. In the guideline for “Software Supply Chain Security” [5] a recommended minimum on how to do verification of software code written internally or when outsourced to a commercial vendor. At the top of the list is Threat Modeling. US Government now sees threat modeling as a part of a minimum baseline that must be performed when creating systems or applications. The cyber security community must therefore provide even better and more efficient ways to threat model than we do today, that will enable those outside our community to take part in the practice without too much help.

When a system is developed following the new paradigm of writing the code, computing resources, network resources, storage resources and configuration in code-like syntax the whole program can more easily be treated as a single entity. In this thesis I will try to answer the following research questions:

- I. Can threat modeling of a whole system in the cloud be partially automated?
- II. Will the threat model be more relevant if manually refined at some point by a security professional?
- III. How does the usage of IaC impact the feasibility of continuous threat modeling?
- IV. Can a semi-automated threat modeling tool be a part of a SDLC?

1.3 Contribution

By improving the reusability and maintainability of threat models, and reducing the cost of threat modeling the practice could be adopted by more organizations. This would increase the security posture of most organizations and reduce the cyber security risk of the organization. This thesis hopes to prove the feasibility of reducing the cost of threat modeling by using the inherent traits of cloud computing.

1.4 Research method

The chosen research strategy is a case study, and the approach is qualitative. Firstly, I defined the relevant background information. Then I conducted a literature review of current research of into threat modeling, automation of threat modeling and integration of threat and risk modeling in the development lifecycle. I then defined and proposed a tool that could simplify the process of threat modeling a cloud environment. Then I compared this solution to the current way of doing this by qualitatively comparing the threat model and resources required to do it manually. Following this I concluded the thesis and answered the research questions posed earlier in this thesis.

1.5 Scope limitations

The thesis will mention some commercial offerings within the threat modeling space, but will not compare them to what is discussed in the thesis.

2 Background

In this chapter I will review the concepts and background information used in this thesis. This is key information needed to understand the problem area, and the underlying knowledge. To avoid any confusion, common concepts or terms are discussed and defined.

2.1 Information Security

The main goal of security work is to preserve the confidentiality, integrity, and availability of any given system of data. To accomplish this, a set of security control are required. Many organizations or individuals responsible for the information security often use threat modeling as a part of their risk management to uncover what security controls are needed for the appropriate level of security. These security controls are put in place to manage the risk level of their systems. The term “security controls” can be seen interchangeably used with mitigation or remediation. The term cyber security is sometimes confused with information security. Cyber security is “the process of protecting information by preventing, detecting and responding to (cyber) attacks” [6]. Cyber security is a subset of information security.

Confidentiality, Integrity and Availability

Within information security, the main goals are to uphold the CIA-triad. Confidentiality is “Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information” [7]. In other terms ensuring that data and information on cyber systems are only available to authorized entities.

There needs to be trust in the data that is accessed therefore the property of integrity is important when safeguarding systems. Integrity in cyber security is “Guarding against improper information modification or destruction, and includes ensuring information non-repudiation and authenticity” [7]. Information and data must only be modifiable and destructible by authorized entities. It is also a characteristic of integrity that the entities cannot claim interaction with the data did not occur without proof.

The third property in the CIA-triad is availability, which is “Ensuring timely and reliable access to and use of information” [7]. Data and information only have value if authorized entities can access it when they request it.

Vulnerability

A vulnerability is “the existence of a weakness, design, or implementation error that can lead to an unexpected, undesirable event” [8]. Vulnerabilities are often described with an attribute that depicts the duration since the issue was discovered. A zero-day/0-day vulnerability is a vulnerability that is not known to those who would be responsible for correcting it, or mitigating the threat it poses. N-day vulnerability is often used to describe a vulnerability where there is a known mitigation, or an updated version of the software has been made publicly available. Where n represents the number of days since the discovery. A 1-day vulnerability describes a vulnerability where the vulnerability was discovered one day ago.

Design flaws

Designs that do not meet the requirements or do not serve the need of the system are flawed. They get introduced at the design, plan, or architecture phase of system development. Design flaws can result in unwanted behavior and introduce exploitable vulnerabilities in the system when implemented. Designs are conceptual ideas about how a system is to be implemented, and removing design flaws are viewed as less costly than removing the vulnerability once it is implemented. A flawed design can never be implemented securely, even if the implementation is perfect. That is why discovering these before implementation is so important.

(Cyber security) Threat

A threat is “any circumstance or event with the potential to adversely impact an organization” [7].

Threat actor/agent

A threat agent or actor is “an entity that has the motivation and capacity to take advantage of and exploit a vulnerability” [9]. Threat actors are by many divided into categories that describe their motivation and capacities. This is often depicted as a pyramid, with the higher levels of the pyramid being the most advanced actors. From the bottom of the pyramid, we have what is often referred to as “Script Kiddies”, these are threat actors that normally use off-the-shelf attack tools. Then there are criminals and organized criminals. Their motivation is solely financial, and their capabilities often range from low to very specialized. There are also

“Hacktivists”. These groups use cyber capabilities to promote their political agenda. At the top of the pyramid are Advanced Persistent Threats (APT). These groups are either nation state sponsored or directly linked to nation states intelligence services. The most well-funded APTs have close to unlimited resources, are highly technical and their motivation is clear; gather intelligence that promote the interest of their state. Among known APTs the National Security Agencies (NSA) Tailored Access Operations (TAO), Israel Defence Forces Unit 8200, Chinese groups: APT10, APT9, APT8 and Russian groups APT28 [10].

2.1.1 Cyber threat intelligence

This term is often used interchangeably with cyber threat information sharing. It is the process of collecting, assessing and analyzing data about threats and threat actors in the cyber security sphere [11]. When used in the cyber threat information sharing we also encompass how to share this information. CTI can be used to take informed decisions when prioritizing what and how our risk is to be managed. CTI is often into three levels of abstraction: tactical, operational and strategic. Tactical CTI is the information about what TTPs the threat actors use, the operational level includes the operational usage of TTPs and incoming attacks and strategic CTI is high-level information on the threat landscape.

2.1.2 Attacker taxonomies

To share insight more easily into the behavior of adversaries and define a common model of how to describe the threat landscape multiple models, taxonomies and frameworks are suggested.

MITRE ATT&CK

“MITRE ATT&CK is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. The ATT&CK knowledge base is used as a foundation for the development of specific threat models and methodologies in the private sector, in government, and in the cybersecurity product and service community” [12]. The framework can be used to annotate specific actions a threat actor will take to achieve their goals when threat modeling. MITRE ATT&CK started with being focused on “advanced persistent threats” on the Windows operation platform, but has since expanded beyond the first scope and platform. The framework also contains a database of mitigations to prevent the

described techniques and procedures from being successful. The database uses the denominations tactics, techniques, sub-techniques, and procedures to describe the actions taken by adversaries.

TTP

TTP is an abbreviation for tactics, techniques and procedures and describes the behaviors of an actor. The three describe the behavior at different abstraction layers. Tactics is the highest-level and describe the actions an actor takes to achieve a particular result. In many ways tactics is the “why” and the goal of the action. The techniques describe how the actor normally performs that action to achieve the wanted result at an abstraction level that can cover many ways on how to achieve the goal. Procedures is even more detailed and are closer to the implementation level on how the actor achieves the how [13]. TTP’s are commonly used in Cyber Threat Intelligence and can be used in threat modeling exercises to describe the goal and the tactics and techniques an adversary wants to achieve and will employ.

2.2 Cloud computing basics

Cloud computing is not a standardized concept or a strictly defined term, and there are different services and implementations. However there are certain common traits and characteristics that are defining for the paradigm. A commonly referenced definition is “Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [14]. The same authors argue that On-demand self-service, broad network access, resource pooling, rapid elasticity and measured service are the five main characteristics of cloud computing.

The benefits of cloud computing are many such as economics of scale, reduced upfront investment cost on hardware, access to almost infinite and ubiquitous compute, storage, networking, and flexibility [15].

Cloud Service Provider

The provider of a cloud service. Hosts and operates the infrastructure and services referred to as the cloud. Amazon, Microsoft and Google are the three biggest cloud service providers (CSPs) and host the three clouds AWS, Azure and Google Cloud Platform.

2.2.1 Service models

Cloud computing can be utilized in different ways by either an organization or an individual. These service models are defined by NIST, and broadly adopted by the community and CSPs, such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). This is commonly referred to as the SPI-model [16]. We also see the use of the “aaS” suffix for other service models or in marketing, such as Function as a Service (FaaS) and the all-encompassing term XaaS. The service models assign differing levels of management and responsibilities between the consumer and the CSP. CSPs offer what they call serverless-computing. This is services where the management of servers is no longer relevant as this is fully abstracted away from the consumer. FaaS is often a service model which is serverless for the consumer.

Infrastructure as a Service (IaaS)

The CSP manages the underlying services and assets such as virtualization, hardware, network, and datacenter facilities. The consumer manages everything else including OS, application(s), and data.

Platform as a Service (PaaS)

The CSP is responsible to manage the underlying OS, runtimes, and middleware. The consumer manages the data, and application(s).

Software as a Service (SaaS)

The CSP manages underlying services and assets, OS, runtimes, middleware, and the application. The consumer can manage some application specific configurations, and is responsible for the data stored and processed.

2.2.2 Deployment models

Cloud computing can be deployed in different manners. The deployment models vary in characteristics such as the location of the infrastructure, the availability to potential users, and the goal of the service. NIST SP-800-145 [14] defines the deployment models as: Public cloud, Private cloud, Hybrid cloud and Community cloud.

The focus of this thesis is primarily on the public cloud, but it can also be applied to use in certain cases for the other deployment models. The public cloud is defined as “The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.” [17]

2.2.3 IaC

Assets and their configuration in the cloud can be defined as code. This is achieved through machine-readable definition files. This text-based approach to defining and automation provisioning of assets is referred to as Infrastructure as Code (IaC). This allows for whole systems to be defined, contained, and operated as code. Using version control systems (VCS) this allows an organization to have a single source of truth on how their IT-environment in the cloud was at a point in time. This paradigm tries to use the tools and processes long used by software development to improve the efficiency of IT-operations of cloud infrastructure. Tools and languages that enable IaC can be divided into two categories: declarative and imperative.

Declarative IaC

The declarative approach is about saying what is needed as the outcome of the process. With this approach the engineer defines the desired state of the final solution, and the underlying tool or algorithm is tasked with creating the best way to achieve the desired state. Many solutions and languages that facilitate declarative IaC maintain the state of the current objects that are to be or have been created. Declarative approaches require less knowledge of the platform where the infrastructure is to be provisioned as the tool takes care of the provisioning details. If the underlying tool or language isn't changed declarative IaC is idempotent, the outcome should always be the same if the same file is run.

Imperative IaC

In this paradigm the coder defines the steps necessary to reach the final state which one desires. These steps contain the exact commands and configurations needed. It is then the tools task to simply fulfill the steps in the sequence defined. Imperative IaC is about telling the underlying tool how to create the system wanted. The coder is in this instance much more in control of how task is executed, and this allows for more optimization for specific use cases. There is no guarantee that the outcome is always the same with imperative IaC as some tasks defined might rely on the output of the previous commands, and their state might have changed.

2.3 Cloud Security

Because of the nature of the relationship between consumer and the CSP, there is a need to define the security responsibility for each party in the cloud. For each service model and differing between the CSPs, there is a certain set of expectations and responsibilities divided between the parties. A common simplification of this model is “the provider ensures the security *of* the cloud; the consumer ensures the security *in* the cloud” [18]. This model is in literature and by the stakeholders in the ecosystem, referred to as “the shared responsibility model.” Depicted in Figure 1 is the UK National Cyber Security Centers generalization of this model across cloud service providers.

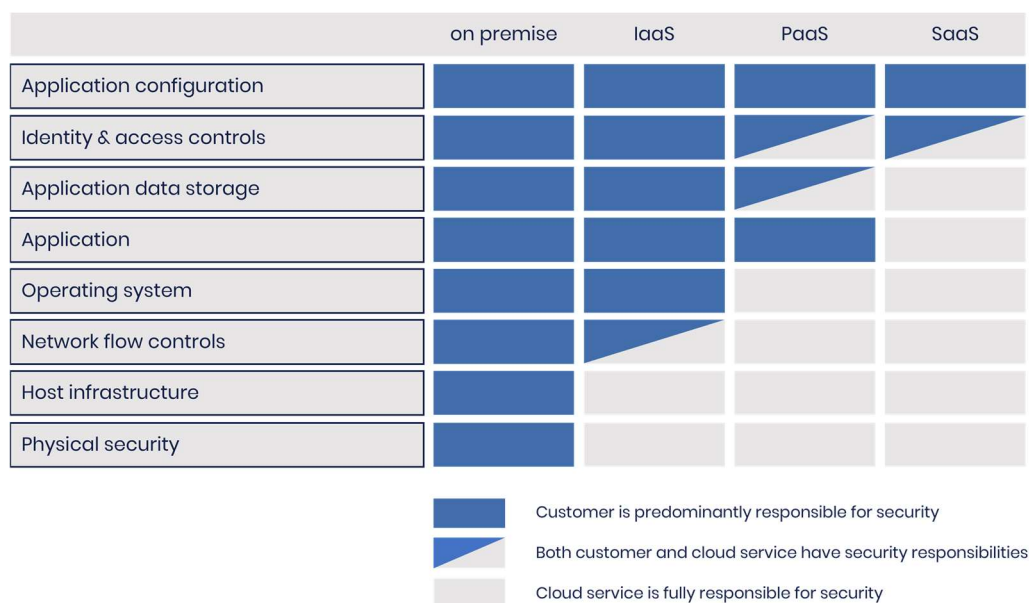


Figure 1 - Shared responsibility model visualized [19]

In the shared responsibility model the consumer will almost always be responsible for the information and data in the cloud, the IAM, platform, and resource configurations, including its own application code and logic [20]. The rapid changing offerings and systems of the CSPs needs to be monitored by the consumers. Some changes might affect what falls under the consumers responsibility, and if it doesn't respond the cyber security posture of the consumer can be adversely affected [21].

For consumers not familiar with its responsibilities in the cloud, the security of the application is a significant issue. The ubiquitous nature of cloud computing can increase the chance of exploitation and compromise as previously relied upon perimeter- and network-security measures might in many instances not be able to mitigate the vulnerabilities relevant to cloud computing. Organizations utilizing cloud computing must therefore be aware of what is expected of them regarding securing their systems by the CSP.

2.3.1 Cloud vulnerabilities and threats

Most of the common vulnerabilities apply in both a traditional on-premises computing scenarios and in cloud computing scenarios, but there are classes of vulnerabilities that are more prevalent and adapted to exploit the ubiquitous access and shared tenancy of cloud computing. A collection of these vulnerabilities were categorized and described in a "Cybersecurity Information report" from the National Security Agency [22]. Another collection of cloud threats created yearly by the Cloud Security Alliance in the series "Top Threats to Cloud Computing" [23] collects what cloud security professionals believe are the top security risks when it comes to cloud computing. Below is a summary of the vulnerabilities from the NSA report, why they are relevant, and important characteristics of the vulnerability-class. Then the CSA report is summarized.

Misconfigurations

"Is the incorrect or suboptimal configuration of an information system or system component" [24]. This includes, but is not limited to, improper configuration of data protection, data retention or encryption. Some analysts believe that through 2025 99% of all cloud security failures will be due to misconfigurations [25]. The rapid pace of innovation leads to new offerings, and adds complexity for the administrators of cloud environments. As Bruce Schneier wrote in 1999 "The worst enemy of security is complexity" [26], adding complexity

increases the likelihood of misconfigurations and increases the chance of introducing a vulnerability.

Poor access control

These are the weaknesses in the mechanisms relied upon for authentication and authorization, or the ability to bypass these mechanisms. Although not only a cloud specific vulnerability, but it is also a common weakness with cloud deployments. The ubiquitous access to cloud resources increases the risk level of a risk related to this weakness as anyone with internet access may in many cases abuse the poor access control.

Shared tenancy vulnerability

This is the weakness related to multiple customers relying upon the same hardware and software, normally also the operating system. If there is an underlying vulnerability in the infrastructure, an adversary can abuse this to move from CSP to customer, between customers or from customer to CSP.

Supply chain vulnerability

This is the presence of inside attackers, and intentional backdoors in hardware and software. CSPs are a high value target for threat actors and are at risk of being subject to a supply chain attack. CSPs are often multi-national large enterprises, and they source hardware and employees from across the globe. This can lead to both backdoors and/or inside attackers.

Pandemic eleven

The list of threats identified in 2022 by the CSA shares a lot with the vulnerabilities in the list generated by NSA. Figure 2 displays all eleven threats.



| Survey Results Rank | Survey Average Score | Issue Name |
|---------------------|----------------------|--|
| 1 | 7.729927 |  Insufficient ID, Credential, Access and Key Mgt, Privileged Accounts |
| 2 | 7.592701 |  Insecure Interfaces and APIs |
| 3 | 7.424818 |  Misconfiguration and Inadequate Change Control |
| 4 | 7.408759 |  Lack of Cloud Security Architecture and Strategy |
| 5 | 7.275912 |  Insecure Software Development |
| 6 | 7.214493 |  Unsecure Third Party Resources |
| 7 | 7.143066 |  System Vulnerabilities |
| 8 | 7.114659 |  Accidental Cloud Data Disclosure/ Disclosure |
| 9 | 7.097810 |  Misconfiguration & Exploitation of Serverless & Container Workloads |
| 10 | 7.088534 |  Organized Crime/ Hackers/ APT |
| 11 | 7.085631 |  Cloud Storage Data Exfiltration |

Figure 2 - Top eleven risks in cloud computing according to CSA Survey and curation [23]

On the top of the list “Insufficient Identity, Credential, Access and Key Mgt, Privileged Accounts” is the same as NSA identified. With cloud computing being accessible from anywhere not controlling access properly is a big issue. Many of the other threats identified such as “Insecure Interfaces and APIs”, “Insecure Software Development”, “Unsecure Third-Party Resources” and “System Vulnerabilities” are related to both Supply Chain and Cross Tenancy vulnerabilities.

2.3.2 Cloud threat actors

Threat actors are often highly specialized, especially financially motivated actors. With the prominence of cloud computing increasing year of year for over ten years some threat actors are focusing especially on attacking cloud environments [27]. Cyber Security Research and Threat Intelligence experts in Unit 42 by Palo Alto Networks has started tracking threat actors that target cloud environments, and their TTPs. This has been made publicly available.

2.3.3 Cloud security tools

There are many tools in the market by commercial vendors. These tools are there to help organizations secure their cloud workloads. Cloud Security Posture Management is a commonly applied solution to assist security professionals to have oversight over their attack surface in the cloud. The CSPM automates security and compliance assurance. This is often done by getting the configuration of the cloud environment and then checking it up against a set of policies or frameworks. These checks results in alerts of misconfigurations being sent to the organization. Some CSPM products can, if configured, automatically mitigate the vulnerabilities introduced by the misconfiguration. Certain CSPM offerings include the capability of scanning IaC-files for misconfigurations. These alerts normally have no context and can't give a holistic view of whether the vulnerability is exploitable and the impact if it is. They are also a reactive security feature, in the sense that the IaC must have been written before we can get feedback on the security posture of the system. Risk management should be a continuous effort, and this implicates that some form of threat modeling should be done beforehand even when deploying such scanning.

The CSA has created a framework named the Cloud Controls Matrix [28] that is now in its fourth version (CCMv4). This is a control framework for cloud security. The CCM can be used by organization to identify its security posture for cloud environments. This can then be used to create a plan for how to improve the security.

2.4 Risk management

Risk management are the “coordinated activities to direct and control an organization with regard to risk” [29]. Cyber security risks are a part of the operational risks for an organization, and handling this risk resides within the overall risk management process of the organization. Organizations should have a mature, pragmatic, and organized approach to the risk management process. This would allow the organization to optimize their operation.

Risk management when applied to cyber security is the process of analyzing what can happen, and what are the possible consequences and before deciding what to do and when to do it, to reduce the risk to an acceptable level [30].

Risk

Risk is an abstract term, and is defined in many differing ways, both generally and when applied to the information security domain. In general, the commonly accepted definition is defined by the International Standards Organization as the “effect of uncertainty on objectives” [31], and when related to information security “the potential that a given threat will exploit vulnerabilities related to assets and hereby damage the organization” [30]. More specifically we can say that a risk is the relevant combination of a threat, a vulnerability, and an incident that breaches a security goal for an asset [32].

Risk level

Risk level is the product of the likelihood of an event occurring and the consequence if it occurs [33]. As we see from the definition of both risk and risk level, risk level is what is referred to as risk in common language.

Risk assessment

Risk assessment is the collection of underlying processes to identify threats and vulnerabilities. Then analyzing the threats and vulnerabilities identified and evaluating the likelihood and impact of the analyzed threats and vulnerabilities. This to uncover, quantify and manage the risks relating to information systems. The concrete implementation of these processes and procedures differ in the literature and with vendors, but commonly applied models are NIST Guide for Conducting Risk Assessment [34] and ISO 27005 [30].

Risk appetite

Organizations have differing views on risk. There is often a correlation between risk and opportunity, therefore organizations chasing bigger upside have to be willing to accept more risk. That overall attitude towards aggregate risk is the risk appetite. This pertains to a holistic view of risk, and is many times viewed to be the sentiment before risk treatment. The International Standards Organization defines risk appetite as “amount and type of risk that an organization is willing to pursue or retain” [35].

Risk treatment

After risk is assessed and documented the involved parties can take informed decisions on how to treat these risks to have a risk level that is in compliance with the desired risk level of the organization. Risk treatment can manifest as differing activities such as removing the risk source, introducing controls to reduce the impact or consequence, avoiding the risk and/or sharing the risk with other parties. Overall risk treatment is the “process to modify risk” [36].

Risk tolerance

While pursuing its goals organizations or key stakeholders will assess the inherent risk of the tasks and objectives they have. Following the risk treatment, the residual risk is the risk tolerance of an organization or stakeholder. ISO defines it as “organization’s or stakeholder’s readiness to bear the risk after risk treatment in order to achieve its objectives” [37].

Asset

An asset can be any system, person, process, information or data of value. The European Union Agency For Cyber Security defines it as Anything that has value to the organization, its business operations and their continuity, including Information resources that support the organization’s mission [8].

2.5 Threat modeling

Threat modeling is a set of processes and techniques to uncover and quantify, the threats that are relevant for the system, environment, or application that is being subjected to the modeling. It can be a part of the risk assessment lifecycle, or conversely can be a substitute for the full process as “a form of risk assessment that models aspects of the attack and defense sides of a particular logical entity, such as a piece of data, an application, a host, a system, or an environment” [38]. Another definition of threat modeling is “Threat modeling is a process that can be used to analyze potential attacks or threats, and can also be supported by threat libraries or attack taxonomies” [39]. There are many ways to approach threat modeling with both prescriptive processes on how to conduct a threat model for a system, or more overarching frameworks to do the threat modeling within.

Threat modeling is seen a part of an emerging paradigm of cyber security called “Security by Design”, which aims to ensure that software and IT systems are designed to uphold its security goals and these security goals are linked to the business or organizational context of the system owner. We use threat modeling as a tool in software security to build systems that function while under attack from malicious entities.

Threat model

“All models are wrong, but some are useful” [40] applies to threat modeling. During the threat modeling process, an abstraction of a system is created. The choices and errors that are done at implementation time are not know prior. The threat model then serves as the best representation of the inherent flaws of the system.

There is no agreed upon definition of what a threat model is. The threat model to many is what is produced during the process of threat modeling. The format this takes differs between the frameworks that are described later. For the sake of common understanding and the lack of a formal agreed upon definition I propose the following:

“A threat model is the representation of all discovered threats that can affect the security properties of the system being modeled. It can either be a textual representation, a graphical representation or both”. When using the term “a Threat Model” in this thesis, this is what is meant.

Trust boundary, data flow, data processing and data store

Key terms when threat modeling are the above-mentioned terms. Trust boundary is often used in cyber security, but can lead to confusion as the community lacks a common understanding of what it is. A good description of what a trust boundary represents: “within a boundary or a zone, there is some form of a common level of security. Within such zone, the components trust each other and do not have to question each other’s integrity and there is some sort of common control or regime” [41].

The trust boundary is the edge between areas of a system which has different levels of security due to a different set of controls being applied. These controls must strictly ensure the integrity of the entities so they can authenticate their identity to build a chain of trust within the boundary. From a networking perspective the traditional Demilitarized Zone and the

internal server network are different trust boundaries as they are under different security controls. In an operating system context, a non-privileged user cannot run certain operations while an administrator of the system can, so that if a user elevates their privileges from user to administrator, they operate within different trust boundaries.

Data flow in the context of threat modeling is the logical representation of how data moves between an entity in the system to another entity. This can be from the user of a system which enters the data, the data then flows from the user to a server and can from there flow to a database. A dataflow can cross trust boundaries or happen just inside a trust boundary. Dataflows always has a source, where the data first arrives in the system, and a sink, where the data is stored or processed in the system.

Data processing is the logical representation of data that is being consumed by any sort of logic that transforms the data, performs operations based on type of data or otherwise explicitly interacts with the data. Whilst the data store is a data sink which serves the purpose of storing data effectively.

2.5.1 Threat modeling variations

In the literature [42] we often find the various approaches divided into three variations: asset-centric, attacker-centric and software-centric. Other researchers divide the approaches into these categories: asset-centric, threat-centric, data-centric and system-centric [43].

The key difference between the variations is what actors or properties of the system they focus on. An asset-centric threat modeling technique starts with identifying assets that have security goals to be upheld. When the assets are identified the modeler considers how that asset could be impacted and the security goals be breached. After doing this for all assets in the system the threat model for the system consists of all the threat scenarios described. CORAS [44] is an asset-centric threat modeling methodology.

Attacker-centric techniques start with looking at the capabilities, motivation, or goal of the attacker and deduce what the goal is or how it can achieve its goal. In this framework it is of importance to know what threat actors might be interested in the data that will be processed. If data that is being processed have intelligence value then actors atop of the threat-actor pyramid are in the threat landscape, but if the system will mainly process data that can have

value of other threat actors their capabilities and motivation are what shapes the model. An organization that processes highly sensitive national security information[45] like a defense contractor, weapons manufacturer or governmental agencies are often targeted by highly motivated and skilled attackers. This is the same as is done in a threat-centric threat modeling framework.

Software-centric frameworks look at the system as a whole, and how data flows in the system. Based on these flows a threat model can be created. The Microsoft Security Development Lifecycle Threat Modeling approach [46] is a software-centric methodology. This methodology is what sometimes is interchanged with STRIDE. Software-centric and data-centric share many common traits, but data-centric often have an even larger focus on the data itself. In this it also shares traits with asset-centric threat modeling. Data-centric threat modeling focuses on the storage, transmission, processing, and classification of the data to be protected. By working out from the data that has some need for confidentiality, integrity and availability those frameworks then analyze what can be done to impact the security properties.

System-centric is like software-centric threat modeling, but has a broader scope as it also considers the underlying environment that is needed to operate the system, not just the software. It is a more holistic approach where flaws and vulnerabilities in the environment is included in the threat model.

All the frameworks can have overlaps in the underlying processes, threat classification frameworks used and share the same goal. The goal is always to produce an output that can be used to improve the risk understanding and the threats to the security goals of the system.

If we compare asset-centric, attacker/threat-centric and software/system-centric the asset-centric modeler starts with creating a list of assets in the system, whilst an attacker/threat-centric modeler creates or finds a list of threat actors or threats relevant for the system, and the software/system-centric modeler creates an abstraction of their system. In the next step the asset-centric modeler starts assessing how their assets can be compromised, the attacker/threat-centric modeler considers if the system is of relevance of their threat landscape and if so what data or which assets are, and the software/system-centric modeler looks for design flaws in the abstract model of the system. Following this all modelers create the artifacts that are specified in the framework or methodology they work with.

Common named threat modeling techniques and processes includes “STRIDE, PASTA, LINDDUN, CVSS, Attack Trees, PnG, Security cards, hTMM, Quantitative TMM, Trike, VAST and OCTAVE” [47]. These vary greatly in how they are conducted and the goal of technique, framework and process. STRIDE is often mistaken to be a threat modeling framework, but in reality is a threat classification framework created by Microsoft to be used in the “Microsoft Threat Modeling” [46]. Certain frameworks, such as LINDDUN, focus on the privacy aspects more than the CIA-properties of the system. Our main focus will be CIA-focused threat modeling.

Many organizations and commercial actors develop their own frameworks or approaches to threat modeling that works well within their context. These can be complete innovations, or adoptions of other known threat modeling frameworks. Autodesk, a software manufacturer, developed and open sourced a methodology called “Continuous Threat Modeling (CTM)”. It is a methodology built to enable developers to threat model their software continuously within an agile approach to development.

Threat modeling tools

Tan, Garg and Farrow [48] conducted an analysis of what open-source automated threat modeling tools are available. The selection criterion to be identified was firstly the tool had to be free and open-source. Secondly the code base had to have been contributed to the last year and lastly it had to be applicable to any system. They identified: Computer Aided Integration of Requirements and Information Security (CAIRIS), Threats Manager Studio, Threatspec, PyTM, Threat Dragon, and Threagile. I have conducted further search and found TicTaaC.

There are no concrete and agreed upon formalisms in the threat modeling community, neither in the industry nor in academia. This means there is not an agreed upon standard for the output of a threat modeling exercise. In 2022 IriusRisk, a company that develops an automated threat modeling tool, open-sourced a standard called the “Open Threat Modeling Format” which aims to define a platform independent way to threat model any system.

Automated threat modeling

Automated threat modeling is using a tool or system to create the threat model. This is often done by having some definition of the system being compared to a database of known threats,

weaknesses and other errors that can affect the security properties of the system. The definition of the system can take the form of annotating source code with properties of the following code portion. This can also be called “threat modeling from code.”

```
// @accepts arbitrary file writes to WebApp:FileSystem with
filename restrictions
// @mitigates WebApp:FileSystem against unauthorised access with
strict file permissions
func (p *Page) save() error {
    filename := p.Title + “.txt”
    return ioutil.WriteFile(filename, p.Body, 0600)
}
```

Figure 3 - Example of threat modeling in code using ThreatSpec [49]

In Figure 3 we see the developer has annotated a function with the properties `@accepts` and `@mitigations`. These properties are populated with information about the data the function can consume, and how the functions possible vulnerabilities are mitigated with implemented security controls. This is done for all functions in the system, and at the time of processing the Threatspec-tool will create a threat model of the system. Threatspec creates a diagram of the system, and a list of relevant threats and mitigations for the system.

Another way is to define the characteristics and properties of the system is “threat modeling as code.” This is also known as threat modeling with code. pytm is a python-tool that is created for this purpose. You model your system as properties such as (trust)Boundary, Server, Datastore, Actor, Lambda, Classification and Dataflow in Python code. This is then processed by the tool. In the processing the tool looks instances of threats using an extensible threat library built into the tool. The user can then request the threat model output in different formats such as a data flow diagram (DFD), sequence diagram or threat report. Because PyTm is written in Python the underlying code, which is a representation of the threat model, can be versioned in a VCS alongside the application code. This enables tracking of changes in the threat model. Figure 4 shows a minimalistic definition of a web server and a web user interacting.

```

from pytm.pytm import TM, Server, Datastore, Dataflow, Boundary,
Actor, Data, Classification

tm = TM("Simple threat model")
tm.description = "a simple threat model of a website serving static
content"
Web_User = Boundary("Web/User trust boundary")
user = Actor("Web user")
web = Server("Web Server")
web.OS = "CentOS"
web.isHardened = True
web.inBoundary = Web_User

user_to_web = Dataflow(user, web, "Get Request")
web_to_user = Dataflow(web, user, "Sends HTML-file")
tm.process()

```

Figure 4 - Example of threat modeling as code with Pytm

There are also tools that allows for modeling using YAML or JSON syntax such as Threagile. By defining the architecture of the system by properties such as data assets, data flows, trust boundaries and individual risk categories and processing that architecture with risk-rules that detect security issues it creates a threat model. This is very similar to PyTm, but the languages used to define the system, and the threat generation algorithms are different. The threat model can be displayed as a DFD, JSON-object or Excel-file. Threagile allows for risk tracking withing the model itself so the modelers can have an up-to-date view of applied mitigations. This tool also enables versioning in a VCS to track changes to the threat model.

2.5.2 Attack trees

Attack trees are a commonly used tool in threat modeling and graphical representation of the threat model, they were first proposed by Bruce Schneier in 1999 [50]. What he described is a type of and-or-tree. The nodes are connected to their children in either an AND or an OR relation. The root node of an attack tree is the goal of the attacker and results in the breach of a security goal. The goal of the attacker is considered a security event in the eyes of the system owner/defender. The goal of the attacker is then broken down into sub-goals. By completing the sub-goals, the attack/attacker can achieve the main goal. Attack trees are useful and intuitive graphical tools for threat modeling, presenting and describing attack scenarios for both experts and non-experts, much due to their graphical nature that lends itself good to communicate the threat model efficiently [51]. Figure 5 displays an attack tree for the scenario getting access to the contents of a safe.

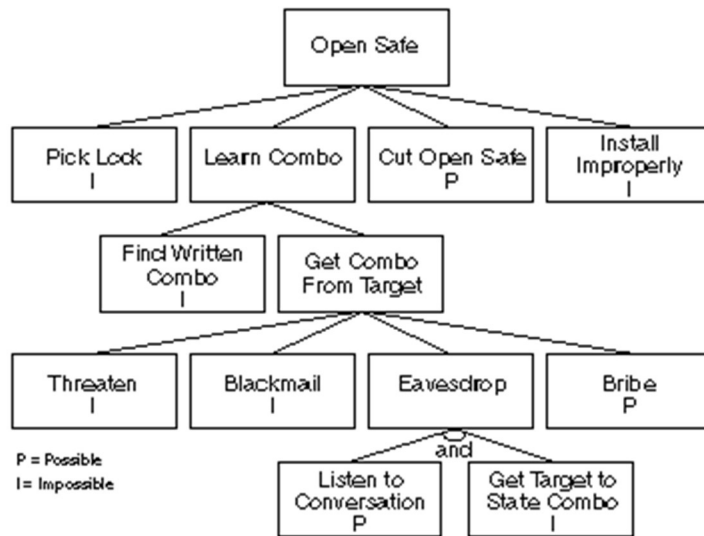


Figure 5 - Attack tree depicting how an attacker could gain access to open a safe. [50]

One of the limitations of attack trees is the complexity and manageability when applied to large and complex systems. Attack trees for one system created by two different experts that has no insight into the other have no guarantee of being the same. This may be attributed to the subjective nature of threat modeling. Additionally, manually creating and maintain an attack tree is time- and labor-intensive.

As an extension to attack trees, Kourdy et al. proposed Attack Defense Tree (ADTree). ADTrees is an attack tree where each leaf node is decorated with possible countermeasure for the action [52]. Any technique to stop the action or event that causes such a breach is a defense.

Automated Attack Tree tools

There are also tools that can assist a threat modeler with creating attack trees. When using similar selection criterion as for automated threat modeling the tools found are ADTool by the Université Du Luxembourg, AT-AT (Attack Tree Analysis Tool) and Ent. Of these only ADTool allows for annotation of defenses. Criterion for selection in this groups is that is the tool is free and open source, can be applied to any system and there is still active contribution to the project.

2.6 DevOps

DevOps, **D**evelopment and **O**perations, is a set of processes, procedures, tools and a cultural paradigm that arose from agile and lean principles [53]. It is a software development and systems operations paradigm that tries to build empathy and shared understanding in teams. These teams are often cross-functional with both business- and technology-oriented team members “working together to accelerate the rate by which businesses realize value” [53]. Davis and Daniels notes that “Devops is not so rigidly defined as to prohibit any particular methodology. [...] the details of its practice are unique per environment”.

DevOps practices and processes are often seen applied alongside agile methods of software development [54]. Agile software development was popularized in 2001 by a broad range of industry professionals who wrote the “Manifesto for Agile Software Development”. The manifesto is a set of principles to improve the quality of software produced. Agile frameworks Kanban, Scrum and Extreme Programming are both underpinned by and derive some practices from the manifesto. Agile has a focus on iterative and continuous improvement of the software, the processes around development and sustainability of developers.

Traditional security practices which have been sequential can suffer in the face of this fast-paced development. In a 2021 paper named “Security in agile software development: A practitioner survey” [4] the researcher pointed to the need for more research into better integration of security engineering practices in agile software development, methods and tools.

Traditionally development, operations and security has been separate teams in most organizations. The development teams have followed the waterfall method for project management. This is a sequential approach to writing software, and all sequences are boxed off from each other. A common list of sequences includes the following phases: requirements, design, implementation, verification and deployment & maintenance. In this methodology threat modeling might happen once in the design-phase. Security teams might also be involved in the requirement-phase to set some security requirements for the system. These requirements were then the basis for security testing in the verification phase. While all teams might have tasks and responsibility in the different sequences, there was a lacking collaboration and empathy between the teams.

In the DevOps-paradigm the sequentially is replaced with a continuous flow of overflowing tasks, often depicted as infinity-character tilted 90 degrees and annotated with the phases of a system lifecycle. A system must continuously be updated and maintained both in the functional and non-functional areas in its lifecycle. This is in DevOps a shared responsibility between the Development and Operations teams. Taking this a step further is SecDevOps/DevSecOps which I will discuss later in this section.

2.6.1 Integration, delivery and deployment in DevOps

Team members in DevOps-teams write code throughout their workday. The process of integrating this code into a mainline branch frequently is continuous integrations [53]. As a part of this process teams often include automated centralized test functionality to ensure the code integrated is functional, and up to the desired standard. The team can also include security testing such as static application security testing as a part of this process. Continuous integration is often referred to as CI.

Continuous delivery of code is an engineering principle that takes the integration one step further. This “allows for frequent releases of new software through the use of automated testing and continuous integration”[53].

2.6.2 Security in DevOps

When taking security into account of DevOps the term SecDevOps or DevSecOps is often used. It is when security is integrated into the DevOps workflow to be as seamless and transparent as possible. Security should not reduce the agility or speed of developers [55]. A foundational practice of DevSecOps is making security services and help available through APIs or code artifacts, and letting the developers decide how to use it.

One term that is often used is Security as Code (SaC). This is defined Jim Bird in the book DevOpsSec [56] as “Security as Code is about building security into DevOps tools and practices, making it an essential part of the tool chains and workflows. You do this by mapping out how changes to code and infrastructure are made and finding places to add security checks and tests and gates without introducing unnecessary costs or delays.” By utilizing security as code security, development and operations teams are enabled to speak the

same language. This underpins the cultural aspect of DevOps and is thought by DevOps-practitioners to improve security.

Security is in SecDevOps a continuous effort that as the responsibility of all involved. There are many different tasks to be conducted to safeguard the system from malicious attacks, and they can be threat modeling continuously, scanning source code and dependencies for implementation errors, security-testing the software and protecting from attacks while operating the system. As has previously discussed there are various frameworks that aim to enable continuous threat modeling such as CTM and tools to assist in the modeling such as open-source projects TicTaaC, Threatspec, Threagile and PyTm. There are also commercial offerings in that space.

3 Related work

In this chapter I have conducted a literature review of the threat modeling domain. The review has been done without a rigorous method, but using common libraries of research with keywords such as “threat modeling”, “automated threat modeling”, “Attack Defense trees”, “Automated attack trees” and the same searches with the UK preferred spelling of “modelling”.

3.1 Threat modeling overview

There is extensive research on both threat modeling, attack trees, and ADTrees, but no currently published work exists on automating the generation of ADTrees for public cloud implementations as a form of threat modeling, or a sub-process of risk assessment.

Xiong and Lagerströms [57] 2019 literature review “Threat modeling –A systematic literature review” selected 54 articles and classified three clusters applying threat modeling, threat modeling methods and threat modeling processes. This was the only SLR found on threat modeling in the literature search. The literature review found that threat modeling is a field where contributors and participants lack a common ground. There is a plethora of definitions, and these definitions are often applied in ad-hoc and very differently by the participants. Due to this many approaches to threat modeling ended up consuming too much time compared to the security value it provided, and was very error prone. This underpins the need for further exploration of the field and especially into automation. Lagerström, among many of his colleagues at KTH, continued to explore the field of threat modeling and especially with a focus on the automation aspect.

3.2 Automated threat modeling

A research group consisting of Kamongi, Gomathisankaran and Kavi from the University of North Texas published a paper in 2014 which proposed an automated framework for Risk Assessment and Threat Modeling for cloud computing [58]. This is based on the VULCAN framework previously proposed by the same researchers [59]. Their approach was to use vulnerabilities, attacks, and defenses as inputs, and using the STRIDE framework for threat modeling. The paper suggests further work be done in the area, and especially relating to

addressing zero-day vulnerabilities. This structured approach shows the plausibility of a system and automation like in this thesis.

When reviewing the current literature on automatic generation and maintenance of A(D)Trees some of the approaches suggest one or more experts create an initial attack-tree before subjecting it to automatic augmentation and refining. This process is then repeated until the tree is adequate [60]. Others have suggested multiple inputs to be put into an automated engine and then subjected to manual analysis by one or more experts [61]. This was conducted as a feasibility study and focused on systems where the architecture was poorly defined due to the early stage of the system development. The refinement by an expert as suggested and tested in this study is similar to the approach that will be included in the case study of this thesis.

In 2016 Johnson, Vernotte, Ekstedt and Lagerström created pwnPr3d [62], which is an Attack-Graph-Driven probabilistic Threat-Modeling Approach. This is an automatic attack graph generator that focused on network modeling. The work focused on creating a model that could be understood by all stakeholders in a system, and did not rely on tools such as Network scanners. The authors believe network scanners have flaws. These flaws include the lacking ability to capture all vulnerabilities in a network, and the dependency on a the CVSS scoring system. This system is not proven as a good system for capturing system level security. There is no sign that the pwnPr3d-solution is currently under active development, and no further research regarding the solution has been found. This study also shows the promising aspects of automating the threat modeling of systems. Three out of four authors of the paper were the founders of Foreseeti, a cyber security vendor founded within InnoEnergy. Foreseeti produce securiCAD enterprise, which is a commercial automated threat modeling tool focusing on attack path generation.

Välja, Heiding, Franke and Lagerström published the paper “Automating threat modeling using an ontology framework” [63] in October 2020 where they developed an ontology framework to add the context and domain knowledge a manual modeler would add to a threat modeling process. They concluded that the ontology framework improved the resulting model and encouraged the community to create similar models for other applications. The paper focused the effort on critical infrastructure. Creating such an ontology framework for cloud computing is out of the scope of this thesis. Having an ontology framework like the one described would be beneficial for the tool to be created in this thesis.

Barankova et al claim to have written an application that could perform automated threat modeling, give a risk score and a graphical representation [64] based on a threat database. The details in the paper are sparse, and no information of the implementation are known. This is similar to what is being explored in this thesis. The claims in the paper cannot be verified, but is nonetheless interesting.

3.3 Manual threat modeling

Aslanyan, Nielson and Parker presented “Quantitative verification and synthesis of attack-defence scenarios” [65] and found that ADTrees can be very useful and intuitive when studying threat scenarios from both an attackers and a defenders perspective. The paper also concludes that ADTrees are a great means to quantitatively study these threat scenarios. The graphical and intuitive nature of the ADTrees also allows non-domain experts to participate in the analysis of the threat scenarios. As this study underpins, attack-defence trees are a great way to visualize security information, even allowing individuals outside of the field to optimize the threat management. This is one of the reasons ADTrees will be used to visualize threat scenarios in this thesis.

Varghese and Buyya highlight in their article “Next generation cloud computing: New trends and research directions” [66] that they found that attack prevention, detection and mitigation for cloud environments had to be further researched to enable adoption of cloud computing. By architecting and designing more resilient and robust solutions from the start cloud environments can be more secure than traditional it-environments. When viewing this in the light of rapid innovation, agility, change and other suggested future work, the need for techniques that support risk management and threat modeling reveals the need for further study in regard to the public cloud. This thesis will try to further improve the field of cloud security computing by lowering the bar on how to identify security issues in cloud environments.

Xiong, Legrand, Åberg et al’s “Cyber security threat modeling based on the MITRE Enterprise ATT&CKMatrix”[67] uses the Mitre ATT&CK framework to design a domain-specific language called enterpriseLang based on the Meta Attack Language[68]. The language enables attack simulations on the model of an enterprise system to assess the cyber security of the system. The authors point out using more sources as input in future iterations

of the tool such as the CVE and CWE databases. The tool created in this thesis has tried to implement more inputs into the threat generation as suggested by this study.

The "threat modeling manifesto"[69] is a collection of values and principles on how to threat model. The intention of the manifesto according to the authors is to share the collective knowledge of a group of threat modeling experts. By doing this they hope to educate other practitioners in the field to adopt the practice in order to improve both security and privacy in the applications developed. The manifesto is written by a large group of industry professionals among others Adam Shostack, the author of the seminal threat modeling book "Threat Modeling: Designing for Security".

Andrei Brazhuk's "Threat modeling of cloud systems with ontological security pattern catalog" [70] point to the fact that security experts have been trying to make safe usage of cloud computing for 10 years, but it's still a large challenge. The main challenges as seen by the author is a lack of automation of threat modeling, and well-formed data sources of attacks and mitigations. In the paper Brazhuk released the "Academic Cloud Computing Threat Patterns" (ACCTP). This catalog can be used to threat model cloud deployments. The work in Brazhuk's papers is used as inspiration for the tool and the clear recommendation on automating more of the threat modeling is in direct connection with what is explored in this thesis.

Brazhuk's work is part of a larger project led by the Open Web Application Security Project (OWASP) called OWASP Ontology Driven Threat Modeling Framework[71]. The goal of the project is to "to maintain a base threat model (ontology) that enables creation of domain-specific threat models" and "to create different domain-specific threat models (for Web applications, Cloud computing, Internet of Things etc.)." This is only one of several OWASP projects focusing on threat modeling. The "Threat Modeling Cheat Sheet" [72] is a document created to help get threat modeling quickly for any one that might need to perform a threat modeling exercise.

Toreon, a Dutch security company, has open-sourced their threat modeling approach through OWASP as the OWASP Threat Modeling Playbook (OTMP) [73]. This is a manual approach to threat modeling based on experiences from industry professionals. The OWASP Threat Model Cookbook is another project from OWASP. It's a collection of threat models to assist and give inspiration to others that are operating within the field. OWASP also supports two

tools to assist threat modelers. OWASP pytm [74] (previously described in section 2.5.1) is a tool that creates a data flow diagram (DFD) based on attributes written directly in python code. Using the (DFD) and a threat database it creates an automatic threat model. OWASP Threat Dragon [75] is a modeling tool used to create threat models. It supports STRIDE, LINDDUN and CIA and follows the threat modeling manifesto. Both of these tools have inspired the tool in this thesis, and shows the large need for better and more robust tooling in the threat modeling space.

3.4 Threat modeling in an agile context

As discussed previously iterative, dynamic, and agile software development does not lend itself to the rigorous nature of traditional threat modeling. Especially when iterations, often called sprints, are short and intensive. A paper by Bernsmed et al [76] collected results from four studies that looked at how threat modeling had been implemented in agile organizations and what challenges arose from this. Some of the challenges the authors found were that most development teams preformed threat modeling as a compliance activity and not due to the security impacts it can have. They also observed that following up on the findings were harder than expected. Most organizations tracked findings in a bug tacking system, but issues kept getting ignored and eventually was no longer relevant.

The same paper suggested “Further development of tools for threat modeling that incorporate the trends in technology that the software products are being developed upon” which this thesis tries to support.

4 Description of solution

In this chapter I will present the approach I propose to improve threat modeling for cloud based systems. I will then describe the design of the implementation.

4.1 Introduction to method

Threat modeling has many benefits. The primary benefit is the threats identified and the mitigations and remediations of those threats, often called the threat model. Threat modeling also generates intimate knowledge of the system modeled in the group modeling the system. Threat modeling can often bring developers, architects, product managers and security professionals together. This creates professional relationships that benefit the system being modeled in other ways than improving the security posture by removing design flaws. This cannot easily be quantified, but should nevertheless not be forgotten when comparing the approaches.

Automated or semi-automated approaches are easier to scale and consume less time from the parties involved. They can give actionable advice quickly and the involved parties can focus their time on improving the security posture of the system in question, rather than spending time on finding the issues. For non-security professionals the overhead of starting with threat modeling can be large, as there is a lot of learning involved. That means automated or semi-automated approaches can be more efficient.

There are resources aimed at these individuals such as Threat Modeling – A Practical Guide for Development Teams [77], but usage of such books or instructions still requires the development teams to have time for learning. Not all organizations can or are willing to invest in that.

This thesis will explore a semi-automated approach to threat modeling as there is currently already a relatively large amount of research on manual threat modeling. These compare the different methodologies and applications of threat modeling. Research on fully automated threat modeling is also easy to find. There is little research found on using semi-automated tools to support both security professionals and other stakeholders when threat modeling. The research I have found on this topic is discussed in the related work chapter.

4.2 Proposed method

Threat modeling, as seen in chapter 3.1 is often a manual, error-prone, time consuming and demanding task. The goal of this chapter is to suggest a methodology that is based on the use of the key benefits of cloud computing. These benefits include “Everything as code” and community projects to create threat databases. Using these a way to automate a recurring threat model every time the system, environment, and application, is built or changed will be developed. This can help the organization prioritize security efforts for the next iteration of the system.

Threat modeling should be performed before any code is written so I will also explore the possibility to evaluate design, architecture and/or documentation before the first build, but the focus will be on continual threat modeling of an existing, in-development system. For a holistic approach the tool will output both in-depth information for system experts, and graphical representation of the threat scenarios for key stakeholders without domain knowledge. Based on CTI and other context that is missed by the method an expert should be able to remove certain attack-paths from the model. There could be security controls that are not covered in the input that would render a threat less relevant, or it could be that the involved parties see that certain threats are not in the threat model.

The primary basis for threats will be the CSA Pandemic Eleven [23] and NSA Mitigating Cloud vulnerabilities [22]. These are threat and vulnerability libraries written specifically for the public cloud. Enriching the model with data from the MITRE ATT&CK framework, Common Weakness Enumeration (CWE) and Common Vulnerabilities and Exposures (CVE) databases will be explored, as suggested by Xiong, Legrand, Åberg et al [67].

To build the infrastructure model I will use the IaC-files that define the infrastructure. This should give close to a complete picture of the system. I will assume the prerequisite that the system is only built using IaC and not using API- or GUI-endpoints. It could be considered using services such as the AWS IAM Policy Simulator API [78] to enrich the model with the current access the actor has, but that is beyond the scope of the thesis.

I want to model the whole system, and not only the infrastructure or environment. The tool could use common open-source security software such as Software Composition Analysis (SCA), Static Application Security Testing (SAST) and Container Vulnerability Scanning.

This could then scan more of the components that make up the system. This data could be input that affect the probabilities for successful execution of adversarial behavior in a threat model, but this will not be within the scope of the work.

There are formalisms [79-81] to quantify the risk when using attack trees in as a part of threat modeling, but this is not a in the scope of the thesis.

The tool is not meant to be a replacement for a threat modeling framework or methodology, but rather as a tool to support within a framework or methodology. As it is aimed at agile teams working in DevOps-cultures on cloud-infrastructure it could fit well within an assimilation of the CTM methodology.

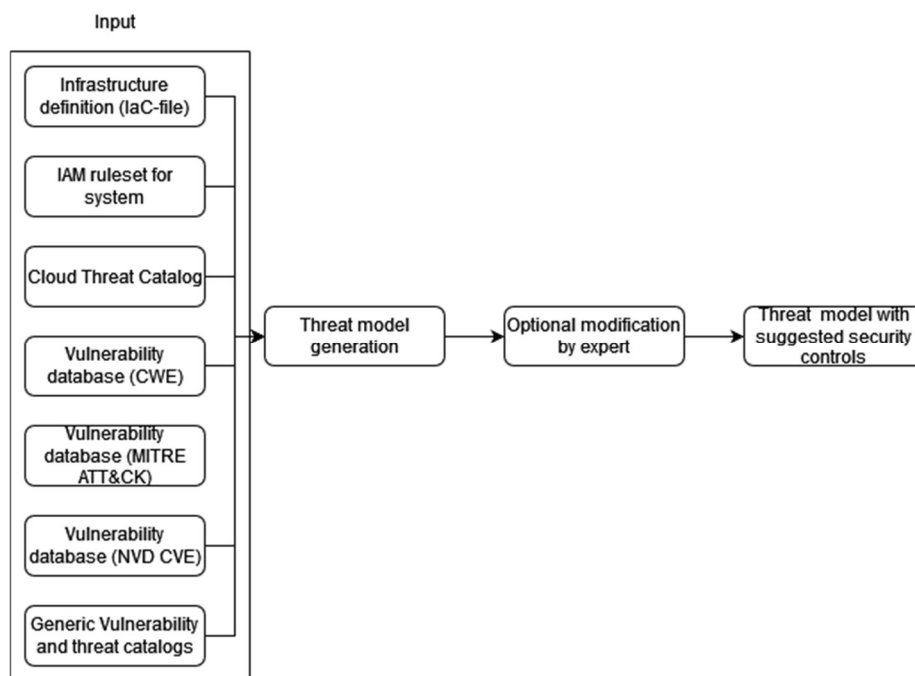


Figure 6 - Logical design of an automated cloud threat modeling engine

Figure 6 displays an overview of input and steps in the algorithm. A simplified version of this approach will be implemented and tested in the development and case study sections of this thesis to allow for feasibility study.

4.2.1 Challenges

IaC alone doesn't provide the answer to where data flows, is processed or is stored. This cannot automatically be deduced with high accuracy from IaC itself. To support data flow representations, the IaC can either be annotated via comments or metadata attributes that

define these properties. If one takes in the source code of the application that were to be operated in the environment this could instead be annotated with the same characteristics. This would be advantageous as it forces the software developers to be more aware of data flows and stores, but the cost of implementing such a procedure in an organization is high. There are also similar tools available, and previously discussed in this thesis, as this pertains to “Threat Modeling in code”. As what is researched in this thesis is the feasibility of semi-automated threat modeling of cloud hosted environments this challenge is abstracted away by adding this information in the IaC-templates. How this is done is described in the next sub-chapter. Automatically determining dataflows and processing is out of scope.

Another approach would be to build a data flow representation based on the code. This could also be done across applications such as a front-end application and an API that consumes and fulfills the request of the front-end. This would allow for even more automated threat modeling, and is similar to what CodeQL by GitHub does when scanning for vulnerabilities in source code. This approach is using an Abstract Syntax Tree and following execution from source to sink.

Similarly, this could be solved by looking at the networking of the environment. For communication to be possible there needs to be a route from the initiator of traffic to the recipient of the traffic. Depending on the infrastructure and the default processing of the network a network access control list-rule and a firewall or security group-rule must be in place for traffic to be allowed. The problem with this approach is that it would only tell us if this traffic is possible, not whether that is the intention of the creators. In threat modeling we are trying to determine the delta between the intended behavior and how we believe the system should be built, while this approach is a check on the concrete implementation.

We have no guarantee that all information needed is contained within one single IaC-file. Many organizations and individuals create multiple IaC-files that fulfill different requirements. A normal architecture for organizations is to define the “platform”, a set of components and policies, that enable other teams to do less infrastructure provisioning and helps ensure that the resources are compliant with company policy, applicable laws and regulations.

On top of this platform the different teams deploy their environments. These are the operating environments for the applications to be deployed. Then the components needed for the

application can be deployed on top of this again. These can be, and are often, defined in separate files per cloud service. This can be a complex structure to process for a threat modeling tool.

Lastly, even if the underlying components are dynamic and they can often change the trust boundaries, data processing, flow and storage doesn't necessarily change. New features, major refactoring of logic in code and changes in those four attributes are often triggers for threat modeling. Changing the underlying cloud-service that the system runs on doesn't always result in any of that changing. In some instances whether a system is run on-premises, in the public cloud or hosted by a third party in their traditional datacenter isn't relevant for the threat model. The operating model for hosting is more an aspect of supply-chain security and that is arguably not a part of the technical threat model.

The cloud-services the vendors offer has different technical attributes, even if they serve the same use case. An example is in AWS where you could choose to store data in the AWS S3 service, which is an object storage service, or the same data in an SQL-server running on an AWS EC2 instance, which is a virtual machine service offering. In this scenario the two services have different known technical risks associated with them and the threat model changes. The aspect of the shared responsibility model is also important in this regard. While with an on-premises deployment you assume all responsibility of the security posture of the system, in the public cloud this responsibility is shared with the vendor. This affects what security controls are needed to implement, and how to prioritize them

As earlier mentioned, compute and storage are only getting cheaper due to the competition in the cloud space and technical advancements. The downside to running an automated threat modeling tool is therefore negligible in cost, and the upside in threat awareness is large over time.

4.2.2 Algorithm

Graphical models are valuable to create a common understanding of what is expected behavior of a system. When facing security questions knowing where data resides, where it flows and how it is processed is a good graphical model. The first thing the tool will do is to translate the IaC to a DFD. This DFD must contain trust boundaries, data flows and data sources and sinks.

Based on the DFD we run a threat generation algorithm looking for design flaws in the system. These design flaws are the source of threats to the system. The impact of a threat materializing, the objective the threat actor seeks to obtain, will be the basis for an attack tree that is then generated. This is then the root node of the attack tree. We cannot have full insight into all security controls in place in the whole environment, as the system being modeled isn't necessarily a complete picture. Therefore at this time the analyst, security professional or developer can choose to mark certain attack paths as unobtainable. That can deem other paths as invalid as well and the tool reruns with the new data.

| Step | Processing | Output |
|-------------|------------------------------|--|
| 1. | Parse IaC-file | Data Flow Diagram, Resources in system |
| 2. | Analyze DFD-file | Threats for the system |
| 3. | Analyze threats | ADTrees for all threats |
| 4. | Manual refinement by analyst | None |
| 5. | Reprocess steps 1-3 | Threats and ADTrees |
| n. | Reprocess steps 1-3 | Threats and ADTrees |

Figure 7 - Processing needed for desired result in threat modeling tool

Figure 7 displays the order of execution, the processing needed and the desired output. The analyst should be able to refine the model as many times as they believe is necessary.

5 Development of Continuous Semi-Automated Threat Modeler

This section describes the development process, and the artifacts produced in the process, to develop a tool that supports the suggested method to compare alongside other approaches of threat modeling. The tool is named Continuous Semi-Automated Threat Modeler (CSATM). The tool is written in the Python programming language. The code can be found at <https://github.com/hsorum95/CSATM>.

Following the standards in software development I have tried to find open-source components with permissive licenses to reduce the development efforts for already solved problems. Where this isn't possible, I have implemented logic to fulfill the requirement. The implementation will only be for threat modeling AWS-environments where CloudFormation is used as the IaC-service. The implementation will only allow for the IaC to be written in one file. The results of this will give enough information about the feasibility focus of the thesis, while not increasing the implementation scope.

5.1 Requirements

To fulfill the goal of the tool list of requirements was created prior to implementing the tool. These requirements are the functional and non-functional properties that must be implemented in the system for the system to have the features and functionalities described in chapter 4. No specific method or framework were used to create the requirements. Neither were the requirements formulated using any specific formalisms.

As some requirements arise while implementing the system these have been captured and added to the list under a separate heading. This occurs when a feature is not possible to implement if other processing has not taken place. Software development is a creative and iterative process. Some ideas to enhance the functionality of the program arise while creating it. These are also added to the list under the same heading.

Requirement list:

| Requirement number | Requirement |
|---------------------------|--|
| 1 | The tool accepts a file-location where the textual representation of the system resides |
| 2 | The tool creates a data flow diagram of the infrastructure |
| 3 | The tool consumes a design flaws and vulnerabilities from an external source |
| 4 | The tool must accept a parameter that dictates what iteration of processing it's currently running, and if no input is given it assumes it's the first. |
| 6 | The tool must contain logic that detects bad configuration |
| 7 | The tool must contain logic that detects bad design |
| 8 | The tool outputs attack and defense trees that visualizes the threat model of the system |
| 9 | The tool must allow interaction from an entity to choose what paths to focus on |
| 10 | The tool outputs a list or a text file of the threats the system is facing |
| 11 | The tool further annotates and enriches on each iteration such that more details and more threats can be uncovered as number of attack paths are reduced |
| Requirement number | Requirements that arose after commencing development |
| 12 | The tool creates a text output of all assets in the system |

Figure 8 - Requirementslist for CSATM

Figure 8 are all requirements that were found prior to and during development.

5.2 Architecture & design

CSATM is a standalone program, with some system dependencies. It's written to be run on machines running Debian-based [82] Linux. It can be run on any environment that has Python 3.8.10 and graphviz [83] installed. Graphviz is the tool used for rendering any graphical representation that is created, and is a system dependency. CSATM uses no databases and saves all output to the local machine where it is run. It can also be packaged to run in CI and CD pipelines. This is to facilitate for the continuous threat modeling of systems in development.

Each functional requirement is implemented as a separate module. This allows anyone with access to the source code to only use the parts of the program they find useful. Such as only rendering a DFD from the IaC and not running the threat modeling-engine.

The code is written to follow the principles of loose coupling. This involves ensuring that each module needs no insight into the intricacies and implementation details of the other modules. By doing this we can change the behavior in one module, if we ensure that we still send the expected data to other modules.

By modularizing the functionality, CSATM is also following the principle of high cohesion. Code that is related to only one task, such as parsing IaC into DFD, is contained in the same module.

5.3 Considerations

The IaC-templates will need to contain information about trust boundaries, data flows and where data is processed. They will use the key-value pairs named in the code to enable the tool to evaluate these attributes.

Figure 9 is a table containing all keys, what is a valid value for the key and if they are required. This approach was chosen to reduce the need to reimplement CSP logic when interpreting the IaC-file for deployment. This is a variant of threat modeling in code, but

doesn't affect the main purpose of the thesis which is to study the feasibility of automated threat modeling of cloud deployed systems.

| Key name | Values permitted and or expected | Required |
|------------------|---|----------|
| trust_boundary | Any string | Yes |
| data_flow | The UID of the entity which the resource communicates with | No |
| data_flow_source | The name of a resource outside of the system the resource itself must accept communication from | No |
| data_flow_sink | The name of a resource outside of the system which the resource itself sends data to | No |
| encrypted | We assume all traffic is encrypted, so if a dataflow is unencrypted this has to be false. Otherwise, the tag is redundant | No |
| datastore | True if the resource stores data, false otherwise | No |
| public | True if the resource is available for anyone on the internet, false otherwise | No |
| data_processing | True if the resource does any computation or modifications of data, false otherwise | No |

Figure 9 - Table describing tags used for identification of system attributes

5.3.1 Explanation of DFD

CSATM will output a dataflow diagram. In this subchapter the legend for how the DFD can be interpreted is described.

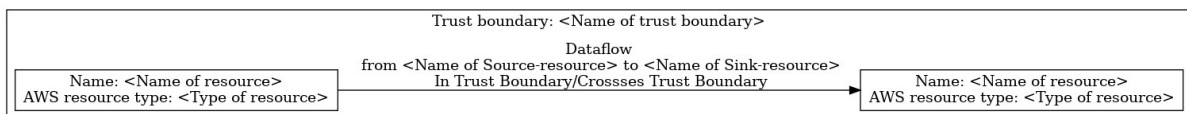


Figure 10 - Generic DFD showing how output will be displayed

Figure 10 shows the syntax for the outputted DFD. A box annotated with Name: <value>/ and AWS resource type: <value> represents a resource such as a server, database or cloud service. Boxes notated with a "Trust boundary: <value>" must contain one or more entities to be rendered. Lines with arrows represent dataflows. These are represented from source to sink, and the arrow is on the sink-side of the interaction. Any dataflows that cross a line forming a trust boundary equates to data moving between trust boundaries.

5.4 Cloud threat, mitigation, and remediation catalogs

To enable threat model generation described in the methodology chapter, both a threat catalog and mitigation catalog had to be created. The threat catalog is similar to what was described in chapter 3.3 named the Academic Cloud Computing Threat Patterns (ACCPT) [70]. The threat catalog used for this thesis is based on the threats described in the Pandemic Eleven [23] and Cybersecurity Information Report – Mitigating cloud vulnerabilities [22].

The threat “Organized Crime, Hackers & APT” has purposefully been removed from the catalog as that is defined as a threat actor, not a threat, in this thesis. The threats are written in the JSON-format and the syntax is heavily influenced by the one used in ACCPT. The mitigation and remediation catalog are based on the CSA CCM Controls Version 4.0 mappings done in the Pandemic Eleven report. It is slightly altered as there were too many mappings per threats to fit the feasibility approach taken in this thesis.

5.4.1 Threat Syntax

The format chosen for threats is JSON. The syntax for the cloud threat catalog uses attributes, and follows the JSON-standard. This allows for adding sub-objects to the main JSON-object. This enables the format to be extensible if more attributes are required per threat.

```
"Unsecure Third Party Resources": {
  "id": "6",
  "name": "Unsecure Third Party Resources",
  "stride": [
    "Non_Repudiation",
    "Information_Disclosure",
    "Elevation_of_Privilege"
  ],
  "description": "Third party resources used in the system
with vulnerabilites or flaws",
  "impact": "7.21"
}
```

Figure 11 - Example object from the threat catalog

Figure 11 is a threat taken from the threat catalog used. All objects loaded into the tool must have a unique ID to enable mapping between threats and remediations and mitigations.

Duplicate IDs will throw an error and the threat loaded into memory last will be discarded.

The “stride”-array contains what negative consequence a threat materializing can cause. The

values of the attributes are the results of merging information from the two previously discussed cloud threat and vulnerability reports.

5.4.2 Remediation and mitigation syntax

The format chosen for remediations and mitigations is JSON. The syntax for the cloud remediation and mitigation uses a similar syntax to the threats.

```
{
  "id":1,
  "name":"Least Privilege",
  "CCMV4_ID":"IAM-05",
  "remediation": true,
  "description":"Apply the principle of least privilege for all
roles in the system",
  "related_to":[1,5,6]
}
```

Figure 12 - Example object from the remediations and mitigations catalog

As can be seen in figure 12 there are attributes seen in these objects that are not in threats. The “related_to” array-attribute allows mapping between threats and remediations as described in chapter 5.5. The “remediation” attribute is a Boolean value and if set to true remediates the threats in the “related_to”. If it is set to false then the object is a mitigation. The description is taken from the CCM, but is altered for brevity.

5.4.3 Mapping of services to service models

To ensure that the threat modeling takes the shared responsibility model into account a mapping of the cloud services used and what service model they belong to has been performed. As there is no definitive mapping of this done by the CSPs any user of CSATM can easily change this based on their own interpretation. This mapping is kept in a YAML-file in the code. There are four YAML-keys in the mapping: external, iaas, paas and saas. Figure 13 is an extract from the file used by CSATM for this thesis.

```
---
external:
  - External
iaas:
  - "AWS::ElasticLoadBalancingV2::LoadBalancer"
  - "AWS::AutoScaling::AutoScalingGroup"
paas:
  - "AWS::ApiGateway::RestApi"
```

Figure 13 - example of mapping from service model to service used in CSATM

6 Case study

In this chapter I will present a hypothetical system that I will use as a basis to test the threat modeling tool presented in previous chapters. I will then conduct a manual threat modeling using a traditional framework on this system. Following that I will apply the threat modeling tool on this system to create a basis for comparison. To facilitate for feasibility study of agile development the system will first be designed based mostly on IaaS-services, and then redesigned to use mostly serverless services in the public cloud. The system will be continuously threat modeled while this migration is done using the tool.

6.1 Real world scenarios

Finding and choosing one or multiple systems to use as testing data in the thesis on resulted as more difficult than expected. Few open-source projects publish their threat model publicly, and the two found, Kamus and Kubernetes, doesn't fit the other criterion for selection. Kamus and Kubernetes are not applications with available IaC for deployment of a whole system. Kubernetes is a solution for managing and orchestrating containers at scale[84]. Kamus is an open-source key management solution for Kubernetes applications[85]. Kamus comes with a Helm-chart, which is a way to define, deploy and update Kubernetes-based applications using the Helm package manager. This would add the complexity of Kubernetes and containers into our threat modeling. In addition to this there was a possibility to lose focus on what this thesis wants to achieve. Which is to understand the feasibility of partially automating the threat modeling process and how the public cloud paradigm affects this. As the publicly available data, IaC and Threat Models, isn't sizable the thesis won't be able to answer the research questions.

The lack of publicly available data can be due to the fact some organizations and individuals view either the threat model, the IaC or both to be security sensitive and wouldn't want that accessible to others. That would then affect whether this thesis could be public or not. To contribute to the improvement of the field of threat modeling contacting organizations to use their data as the baseline for comparison that approach was rejected. It would also increase the scope as interactions with these organizations would require effort and time, that instead could be spent on the tool and thesis.

6.2 Case study system

The application to be used as the basis for the case study is a three-tier web application. It is hosted in a public cloud and its infrastructure is written in YAML and provisioned through the automated deployment. The public cloud used for the case study will be Amazon Web Services and the IaC-tool will be AWS CloudFormation service. This is a declarative IaC-tool. The whole system will be written in one single YAML-file.

6.2.1 System description

The application is an online pharmacy which sells both non-prescription and prescription pharmaceuticals. The application will have two types of users: employees and customers. Authentication will happen through an external provider of secure identities. The identities used in the system are linked to the identities of the actual real-life identities of the users. Prescription records is also provided as a part of the claims from the external identity provider. These are then stored in the system. The application will contain a database of employees for which is used to verify when a user logs in whether it is a customer or an employee.

The pharmacy will not manage logistics such as shipping or inventory. An external party provides this. The orders are sent from the application to a dedicated API provided by the external party. As such this API is viewed as a part of the system and should be a part of the threat modeling.

Customers cannot cancel or modify orders after they are submitted to the system. The system will keep the orders of the users in its databases after they are shipped. The shipping status is sent from the API of the external provider to an internal API that updates the database.

6.2.2 Usage

The application is going to support four use cases, enough to create a realistic threat model. These four use cases are core functions for this to be a realistic system. The use cases are presented in the Figure 14 below.

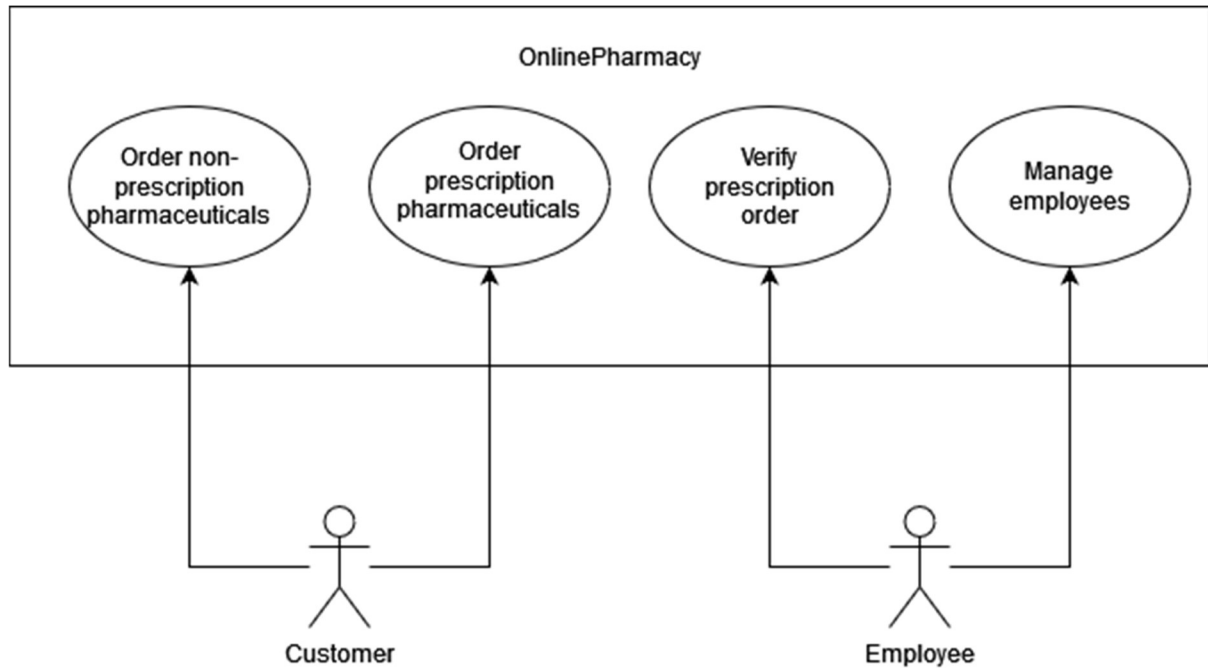


Figure 14 - Use Case Overview

Order non-prescription pharmaceuticals

A customer of the system can order non-prescription pharmaceuticals both as a guest user and a logged in user of the system. As a guest user they will have to provide shipping and billing details at checkout.

Order prescription pharmaceuticals

A customer of the system can order prescription pharmaceuticals. The customer will have to be logged in and a manual verification of all orders will be done by an employee of the pharmacy.

Verify prescription

An employee will log in and verify fulfillment of all prescriptions. The employee will log in using the same external party identification service. This identity is then checked against an internal database to verify that the user matches against the employee database. The employee cannot verify any orders they might have submitted themselves.

Manage employees

Already verified employees that have authenticated sessions can add new employees to the employee-database. Already verified employees can also remove employees that no longer work with the pharmacy.

6.2.3 System design - IaaS

The first iteration to be threat modeled of the systems is built using primarily IaaS-components and some AWS native services. This is a common approach for many organizations when they first start using the public cloud.

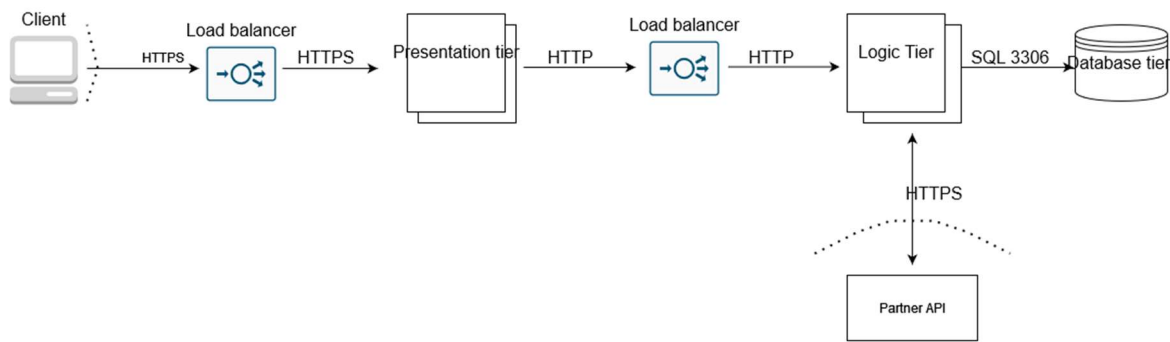


Figure 15 - Architecture diagram of first iteration of cloud application

The system based on IaaS will be referred to as OnlinePharmacy.V1 (OP.V1) going forward in this thesis. The system design is illustrated in figure 15. The clients connect to the front end through a load balancer. The load balancer distributes requests to the presentation tier servers. The front end of the pharmacy is hosted on the servers in the presentation tier. There are multiple servers, which can scale horizontally. This is to allow for high availability and scalability. The presentation tier runs web servers which serves the static content to the clients. The presentation tier also forwards requests which needs processing to a load balancer which routes the requests to the logic tier. This load balancer also distributes requests to logic tier servers. The logic tier also consists of multiple servers, and they can scale horizontally. If the logic tier needs data for processing it talks directly to the database tier. The logic tier is also responsible for getting data from the database tier and communicating it with the external API of the logistics vendor. The database tier is used for hosting a database with multiple tables. Each part is separated logically into different subnets and there are ACLs and security groups that ensures only allowed traffic to flow.


```
Resources:
  PharmacyVPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 192.168.0.0/16
      EnableDnsSupport: true
    Tags:
      - Key: trust-boundary
        Value: cloud-network
```

Figure 16 - Definition of cloud VPC for system

Figure 16 displays how a Virtual Private Cloud (VPC) is defined when using the Cloudformation service with YAML-syntax. “Resources” is an object which contains all resource-objects of the IaC-template. This is one of ten template sections, but it is the only one that is required by the syntax of Cloudformation. The first keyword “PharmacyVPC” is the name of the object within the definition. This can be used to reference the object and its attributes in the IaC-template. Then we declare what type of instance or service we want. This is done on the “Type”-attribute and it follows a syntax of AWS::X::Y where X is the ProductIdentifier and Y is the ResourceType. Then the properties of the object is declared. Any ResourceType has from 0 to n required properties. In this case we create a network with the RFC1918 IP address-space with CIDR notation 192.168.0.0/16 and we enable DNS-resolution within the VPC. The Tags property allows us to set tags formatted as key-value pairs on resources. These will be used for the automated threat modeling. The full template for the system can be found with the code on <https://github.com/hsorum95/CSATM> .

6.2.4 System design – cloud native

The design will rely heavily on serverless computing and cloud native service offerings. This is to assimilate how many organizations are utilizing the capabilities of the public, and see if these deployment models affect the threat modeling. The system shall still serve the same use cases as described in chapter 6.2.2, but how it fulfills the requirements is engineered differently. This design alleviates our security responsibility according to the shared responsibility model as all services used are PaaS-services. This system design will be referred to as OnlinePharmacy.V2 (OP.V2) going forward in this thesis.

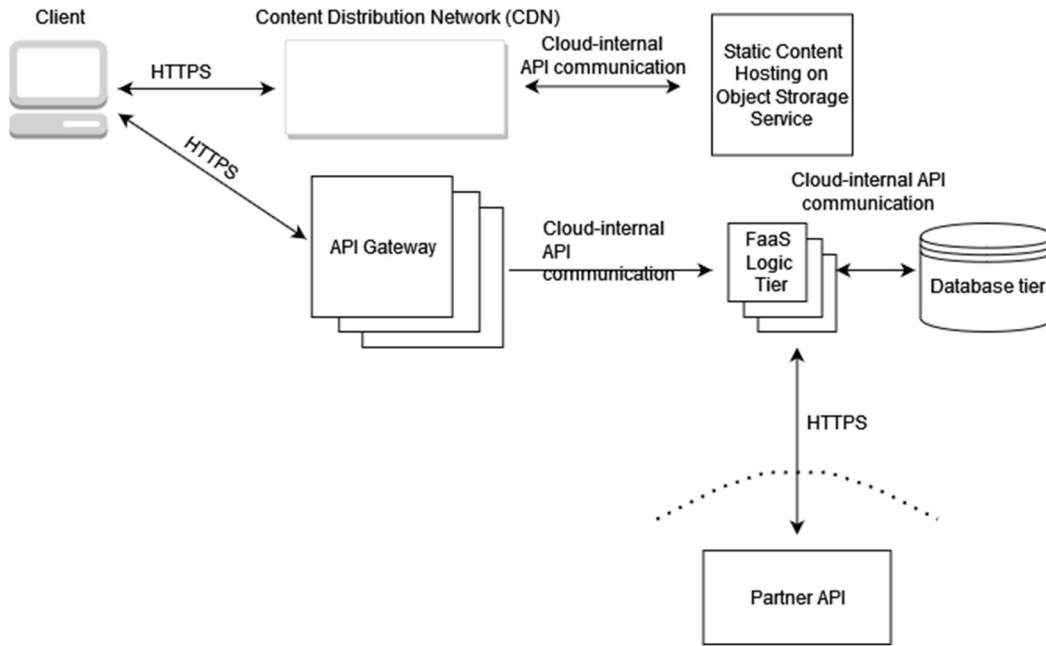


Figure 17 - System design when using cloud native services

Figure 17 depicts the system design of OP.V2. The clients connect to the service through the content delivery network which serves the static front-end code to the browser of the client. When a user performs any action in the browser request are sent to the API Gateway which hosts the API, and the gateway then distributes these requests to the functions that process those requests. Each use case described above has a separate function serving it. The use case to send to partner is also handled by a separate function. The logic tier connects to the database to either put data in the database, fetch data for processing or to send to the external partner. The cloud-native architecture doesn't rely on traditional network-communication for the internal communication, as this is taken care of by the CSP.

6.3 Threat modeling of case study system

To compare the resulting threat model of the case study system different types of threat modeling has been conducted. Both manual and semi-automated threat modeling has been performed based on the information in this chapter. The manual threat modeling of both OP.V1 and OP.V2 was conducted before the semi-automated threat modeling using CSATM.

The threat modeling will follow the prescription of the threat modeling manifesto [69]. First asset identification will be performed. This includes data assets and computer assets in the system. Then we define the trust boundaries and the data flows. Using this information and

the threat-mnemonic STRIDE the threats will be identified, and the corresponding risk level will be quantified with the labels informational, low, medium, high, and critical. Lastly design changes to remediate or mitigate the threats will be documented. In a real-life scenario these would then be ranked and applied, but that is not within the scope of this thesis.

To quantify the risk level of the threats discovered the risk matrix in Figure 18 was created. The risk level is found at the intersection of the chosen likelihood and impact. The matrix is used for both OP.V1 and OP.V2.

| | Impact | | | | |
|---------------|------------|----------|----------|-------------|----------|
| | Negligible | Minor | Moderate | Significant | Severe |
| Very likely | Low | Moderate | High | High | Critical |
| Likely | Low | Moderate | Moderate | High | Critical |
| Possible | Low | Low | Moderate | Moderate | High |
| Unlikely | Low | Low | Moderate | Moderate | Moderate |
| Very unlikely | Low | Low | Low | Moderate | Moderate |
| Likelihood | | | | | |

Figure 18 - Risk matrix used in manual threat modeling

6.3.1 Manual OnlinePharmacy.V1 threat model

Using the use cases from chapter 6.2.2 and the design diagram of the system in chapter 6.2.3 in this thesis the assets, trust boundaries, data flows, threats and remediation/mitigations has been identified. b

Assets of OP.V1

| Name of asset | Type of asset | Nature of asset |
|---|---------------|------------------|
| Order – prescription pharmaceuticals | Data | Highly sensitive |
| Order – Non-prescription pharmaceuticals | Data | Sensitive |
| Shipping information | Data | Sensitive |

| | | |
|----------------------------------|----------------|------------------|
| Prescription information | Data | Highly sensitive |
| Database | Computer asset | Highly sensitive |
| External Load Balancer | Cloud service | Negligible |
| Presentation Tier Servers | Computer asset | Sensitive |
| Internal Load Balancer | Cloud service | Negligible |
| Logic Tier Servers | Computer asset | Highly sensitive |
| Partner API | Computer asset | Valuable |
| Client | Computer asset | N/A |

The asset list is created by viewing the system design documents and the use cases described earlier. Data assets are classified based on their nature. That is data which contain highly valuable information regarding one individual, or valuable information about a larger population of individuals has a higher sensitivity than less valuable information. Computer assets that are processing information with higher sensitivity is classified more sensitive than computer assets that only forward that information. Assets that store data are classified based on the sensitivity of the data they store.

Trust boundaries OP.V1

| Name of trust boundary | Computers in boundary | Description |
|-------------------------------|------------------------------|--|
| External | Client | There is no possibility to control devices in this boundary. |
| External partners | API Service | We do not control the service, but have a level of trust with the organization that does control it. |

| | | |
|-------------------------------|--|---|
| External Cloud Network | External Load Balancer, Presentation Tier | We serve content from this zone. Devices in this zone actively listen for request from any destination in a non-trusted zone. |
| Internal Cloud Network | Internal Load Balancer, Logic Tier, Database tier | All entities in the zone only listen for specific traffic in a somewhat trusted zone. |

The trust boundaries are generated by reviewing the system design document. This is typical for a manual threat modeling process. The group of modelers has intimate knowledge of the workings of the system and should know what assets are within the same trust boundary.

Data flows OP.V1

| Name of data flow | Source | Destination | Description |
|--------------------------|-------------------|--------------------|---|
| Client-FrontEnd | Client | Presentation Tier | Crosses a trust boundary “External Cloud Network” |
| FrontEnd-Backend | Presentation Tier | Logic Tier | Crosses a trust boundary “Internal Cloud Network” |
| BackEnd-Datasbase | Logic Tier | Database | Does not cross a trust boundary. |
| APICalls | LogicTier | API Service | Crosses a trust boundary “External Partners” |

The dataflows identified in the system with a description explaining whether the flow is within a trust boundary from the source to the sink. Four data flows were found in the system, and three of the crosses a trust boundary.

Threats for OP.V1

| Number and name of threat | Description | Assets impacted | Impact | Likelihood | Risk level |
|--|---|--|--------|------------|------------|
| Spoofing | | | | | |
| 1) Clear-text traffic (HTTP) crosses trust boundary | An attacker that has gained access to any computer asset or can act as a man in the middle can read all traffic and see all data. Attacker could change parameters in requests and gain credentials to spoof another actor in the system. | Presentation, tier, internal load balancer, Logic Tier Server, Database. | Severe | Likely | Critical |
| 2) Clear-text (HTTP) traffic within trust boundary | An attacker that has gained access to a computer asset in the internal-cloud network can spoof other | Internal load balancer, Logic Tier Server, Database. | Severe | Likely | Critical |

| | | | | | |
|--|---|-------------------------------|-------------|-------------|----------|
| | assets as the traffic is unauthenticated | | | | |
| Tampering | | | | | |
| 3) Manual verification of orders | An attacker could tamper with requests and get orders fulfilled with access to the logic tier | Logic Tier, partner API | Severe | Possible | High |
| Non-repudiation | | | | | |
| 4) No centralized verifiable log collection | No logging of actions and orders fulfilled | Whole system | Significant | Very likely | Critical |
| Information Disclosure | | | | | |
| 5) Injection attacks on application presentation tier | An attacker abuses implementation flaws that allow injection attacks such as XSS, SQLi and CRLF to gain unauthorized access to confidential information | Presentation tier, Logic Tier | Severe | Possible | High |

| | | | | | |
|---|---|-------------------|-------------|----------|----------|
| 6) Cryptographic failure allows attacker to downgrade encryption scheme in use | An attacker could abuse flaws in outdated cryptographic functions to read encrypted traffic | Public ALB | Significant | Possible | Moderate |
| Denial of service | | | | | |
| 7) Volumetric (D)DoS attack on application front end | An attacker floods the presentation tier with requests making it unavailable or the cost of compute too large | Presentation tier | Moderate | Possible | Moderate |
| 8) Protocol-based (D)DoS attack on application front end | An attacker abuses inherent web-protocol traits to make the application unavailable | Presentation tier | Low | Possible | Moderate |
| 9) Volumetric (D)DoS Attack on application logic tier | An attacker floods the presentation tier with legitimate requests which overwhelms the logic tier or the | Logic tier | Minot | Unlikely | Low |

| | | | | | |
|---|---|----------|--------|--------|----------|
| | cost of compute too large | | | | |
| Elevation of privilege | | | | | |
| 10) Clear-text traffic (HTTP) crosses trust boundary | An attacker with network access in the external cloud-network can alter requests of authorized employees and add themselves to the employee-table in the database and gain persistence. | Database | Severe | Likely | Critical |

The threats are identified by using the STRIDE-mnemonic for threat enumeration, reviewing dataflows and assets that are candidates for abuse, and then creating relevant threat scenarios. As with all threat modeling the list is influenced by the biases of the modeler(s) and isn't an exhaustive list of all threats relevant to the system.

Remediation and mitigation

| Number of threats | Remediation or mitigation | Description |
|-------------------|---------------------------|---|
| 1) | Remediation | Ensure that all network traffic is encrypted and authenticated by using HTTPS on traffic crossing trust boundaries. |
| 2) | Remediation | Ensure that all internal traffic is encrypted and authenticated using HTTPS. |

| | | |
|--------------|-------------|---|
| 3) a | Remediation | Create separate system and infrastructure for administrative and employee actions that can reduce the integrity and authenticity of the system. |
| 3) b | Mitigation | Create an automated system to do verification of orders or require multiple employees to verify all orders |
| 4) | Remediation | Create or ensure that all security relevant logs are stored for an adequate period of time in a tamper proof system located in another trust boundary than the system itself. |
| 5) a | Mitigation | Enable security capabilities that filter and alert on requests that look adversarial using web application firewall on the public load balancer |
| 5) b | Mitigation | Ensure proper validation of all data consumed by the application, use of web application frameworks with built-in mitigations and escaping of special characters |
| 6) | Remediation | For a sensitive application as this handling PII only enable modern cryptography and use modern TLS ciphers |
| 7) | Mitigation | Enable security capabilities that offload traffic when unexpected peaks occur from unexpected locations. Often called DDoS-protection. |
| 8) | Mitigation | Enable security capabilities that offload traffic when common traits of protocol-based attacks are seen. |
| 9) | Mitigation | Enable security capabilities that offload traffic when unexpected peaks occur from unexpected locations. Often called DDoS-protection. |
| 10) a | Remediation | Ensure that all network traffic is encrypted and authenticated by using HTTPS on traffic crossing trust boundaries. |

| | | |
|--------------|-------------|--|
| 10) b | Remediation | Create separate system, infrastructure and accounts for administrative or employee actions so compromise of an asset in the current system can't lead as easily to full system compromise. |
|--------------|-------------|--|

A remediation is removal of the threat scenario while a mitigation reduces the either the likelihood or impact should the threat scenario occur. The list is created by taking an threat from the **Threats for OP.V1**-list and using common security controls to better uphold the security goals. The remediations and mitigations are design patterns or architectural decisions that if implemented correctly treats the risk, but if implemented poorly might not result in the intended security control.

6.3.2 Manual OnlinePharmacy.V2 threat model

Using the use cases from chapter 6.2.2 and the design diagram of the system in chapter 6.2.4 in this thesis the assets, trust boundaries, data flows, threats and remediation/mitigations has been identified.

Assets of OP.V2

| Name of asset | Type of asset | Nature of asset |
|---|---------------|------------------|
| Order – prescription pharmaceuticals | Data | Highly sensitive |
| Order – Non-prescription pharmaceuticals | Data | Sensitive |
| Shipping information | Data | Sensitive |
| Prescription information | Data | Highly sensitive |
| Static web site content | Data | Negligible |

| | | |
|-------------------------------------|----------------|------------|
| API Gateway (AWS API GW) | Cloud Service | Negligible |
| Database Tier (AWS Aurora) | Cloud service | Important |
| FaaS (AWS Lambda) logic tier | Cloud Service | Sensitive |
| Object Storage (AWS S3) | Cloud Service | Negligible |
| CDN (AWS Cloudfront) Service | Cloud service | Sensitive |
| Partner API | Computer asset | Sensitive |

The assets list is created by viewing the system design documents and the use cases described earlier. Data assets are classified based on their nature. That is data which contain highly valuable information regarding one individual, or valuable information about a larger population of individuals has a higher sensitivity than less valuable information. OP.V2 has no computer assets in the same way OP.V1 has due to serverless cloud services (Paas and FaaS) being utilized. The cloud services that are processing information with higher sensitivity are classified more sensitive than cloud services that only forward the same information. Assets that store data are classified based on the sensitivity of the data they store.

Trust boundaries OP.V2

| Name of trust boundary | Computers in boundary | Description |
|-------------------------------|------------------------------|--|
| External | Client | There is no possibility to control devices in this boundary. |

| | | |
|-------------------------------|--------------------------|--|
| External partners | Partner API | We do not control the service, but have a level of trust with the organization that does. |
| Public Infrastructure | CDN Service, API Gateway | Assets in this boundary are there to facilitate connection from the internet to the application. Devices in this zone actively listen for request from any destination in a non-controlled boundary. |
| Private Object Storage | Object Storage | Only used for hosting web content, and is segregated with IAM policies. |
| Private Logic Zone | Logic Functions | These functions accept only request via the provider API from sources with invocation right. |
| Private Employee Zone | Employee Function | This function can only be invoked by employees. |
| Private Logistics Zone | Logistics Function | Runs on a schedule and doesn't accept invocation from anyone. |
| Orders Database | Order DB Table | Accepts put, get and delete from all relevant logic functions based on data access rights. |

| | | |
|--------------------------|-------------------|--|
| Employee Database | Employee DB Table | Accepts put, get and delete from Employee Function based on invocation rights. |
|--------------------------|-------------------|--|

The trust boundaries are generated by reviewing the system design. This is typical for a manual threat modeling process. The group of modelers have intimate knowledge of the workings of the system and should know or decide what assets are within the same trust boundary. OP.V2 has more trust boundaries than OP.V1. This is because the system allows for more tightly scoped permissions and security controls on the flow of data.

Data flows OP.V2

| Name of data flow | Source | Destination | Description |
|---|----------------------|----------------------|---|
| Client-Static | Client | Object Storage | Crosses trust boundaries “Public Infrastructure” and “Private Object Storage” |
| Client-Function logic tier | Client | Functions logic tier | Crosses trust boundaries “Public Infrastructure” and “Private Logics Zone” |
| Functions logic tier-Employee database | Functions LogicTier | Employee DB | Crosses trust boundary “Employee Database” |
| Function logic tier-Orders Database | Functions logic Tier | Orders Database | Crosses trust boundary “Orders Database” |

| | | | |
|------------------|-----------|-------------|--|
| API Calls | LogicTier | API Service | Crosses a trust boundary “External Partners” |
|------------------|-----------|-------------|--|

The dataflows were identified by looking at the system design. They are described with the name where the dataflow logically starts and the name where it ends. The table also contains a description explaining whether the flow is within a trust boundary from the source to the sink.

Threats for OP.V2

| Number and name of threat | Description | Assets impacted | Impact | Likelihood | Risk level |
|---|---|-------------------------|---------------|-------------------|-------------------|
| Spoofing | | | | | |
| 1) External API is only authenticated based on TLS Certificate | We must ensure that the External API is authenticated properly, not only based on authenticity of TLS Certificate and network traffic | Partner API, Logic Tier | Significant | Possible | Moderate |
| Tampering | | | | | |
| 2) Injection attacks on logic tier | An attacker could tamper with data of different sensitivity as they are stores in | Logic tier, DB Tier | Significant | Possible | Moderate |

| | | | | | |
|---|---|--------------|-------------|-------------|----------|
| | the same data store | | | | |
| Non-repudiation | | | | | |
| 3) No centralized verifiable log collection | No logging of actions taken in the application, events occurring in the system and orders fulfilled | Whole system | Significant | Very likely | Critical |
| Information Disclosure | | | | | |
| 4) Cryptographic failure allows attacker to downgrade encryption scheme in use | An attacker could abuse flaws in outdated cryptographic functions to read encrypted traffic | APIGW | Significant | Possible | Moderate |
| Denial of service | | | | | |
| 5) Volumetric DoS attack on logic tier | An attacker could send a large amount of legitimate traffic to be processed by our logic tier. This could result in a huge cost or processing limits. | Logic Tier | Moderate | Possible | Moderate |

| Elevation of privilege | | | | | |
|---|--|--------------|--------|--------|----------|
| 6) Employees and user use the same webpage | There should be separation between administrative work such as verification and employee modification, and normal users. Any vulnerability could potentially lead to full compromise | Whole system | Severe | Likely | Critical |

The threats are identified by using the STRIDE-mnemonic for threat enumeration, reviewing dataflows and assets that are candidates for abuse then creating relevant threat scenarios.

We see that the threat list for OP.V2 contains fewer threats than the one of OP.V1. This could be caused by many factors. One is that since the design uses services where the CSP assumes more security responsibility there are less threats relevant for us when modeling. Another could be that using these types of cloud services are less prevalent, and a newer concept than older models of hosting, less threats are spotted because of lacking knowledge. As with all threat modeling the list is influenced by the biases of the modeler(s) and isn't an exhaustive list of all threats relevant for the system.

Remediation and mitigation

| Number of threats | Remediation or mitigation | Description |
|-------------------|---------------------------|---|
| 1) | Remediation | Implement system for two-way authentication for service-to-service communication. |
| 2) | Remediation | Create separate datastores with different privileges required to store data of different sensitivity. |
| 3) | Remediation | Create or ensure that all security relevant logs are stored for an adequate period of time in a tamper proof system located in another trust boundary than the system itself. |
| 4) | Remediation | Allow only modern and up-to-date encryption mechanisms and cipher suites for communication with the system. |
| 5) a | Mitigation | Enable security capabilities that offload traffic when unexpected peaks occur from unexpected locations. Often called DDoS-protection. |
| 5) b | Remediation | Ensure rate-limiting for authenticated actions so legitimate users cannot overly consum resources and make the application inaccessible for other users. |
| 6) | Mitigation | Scope roles, permissions and privileges tighter so the likelihood of an attacker being able to perform such an attack lower. |
| 7) | Remediation | Create separate system for administrative or employee actions so compromise of a service in the current system can't lead as easily to full system compromise. |

The list is created by taking an threat from the **Threats for OP.V2**-list and using common security controls to better uphold the security goals.

6.3.3 Threat modeling of OnlinePharmacy.V1 using CSATM

Using the tool developed, CSATM, a threat model for OP.V1 has been created. CSATM creates a lot of artifacts as output, some of which have been redacted for brevity. Where only an excerpt has been used in this thesis the full artifact can be found with the code on <https://github.com/hsorum95/CSATM>. CSATM was run with the CloudFormation-template three-tier-web-app.yml as the input file.

Data flow diagram

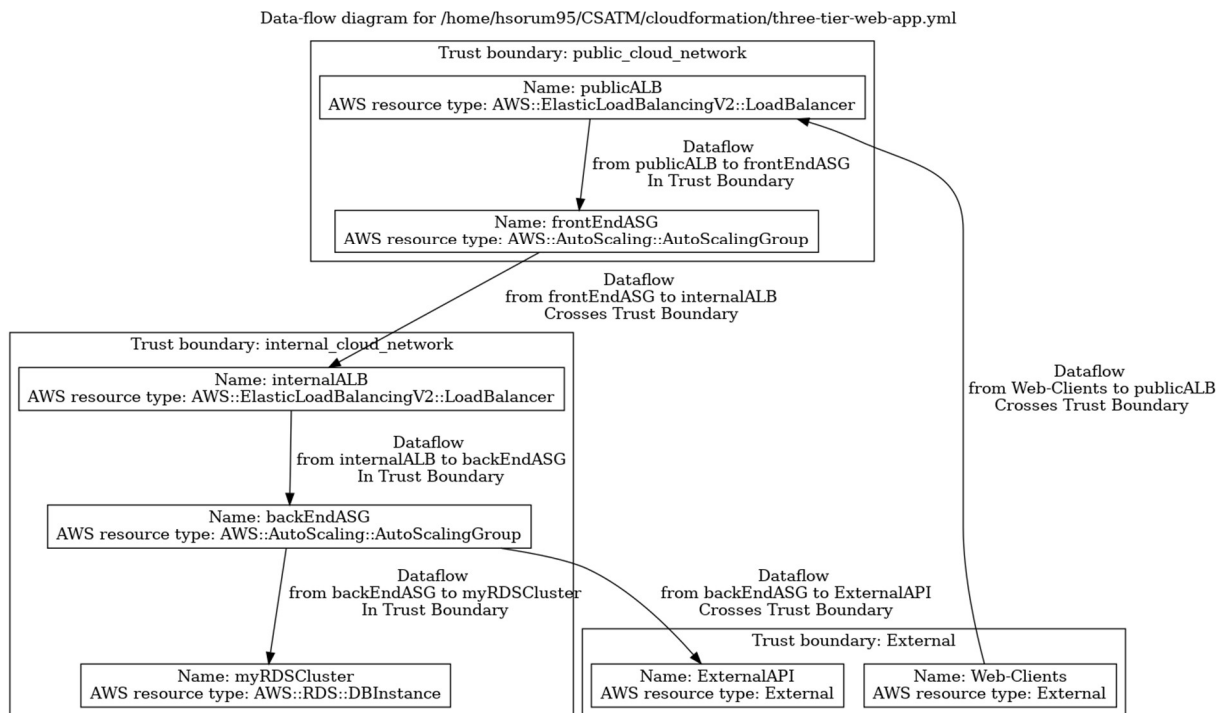


Figure 19 - DataFlowDiagram of OP.V1

Figure 19 displays the DFD for OP.V1 created by CSATM. We can clearly see whether data flows within a trust boundary or crosses a boundary. The CSATM has created more dataflows than manual modeling as CSATM treats each flow as a separate flow, while the manual modeling abstracted away hops on the network layer when the traffic flowed through the asset. This could be advantageous for detecting more design errors. Some modeling-time abstractions made unconsciously by the modelers could cover over flaws relating to the infrastructure it is deployed on. CSATM could help combat such event.

Resources

The tool also creates a file with all resources which are within a trust boundary. This file could be used for purposes such as an asset database, a sanity-check of the CSATM-outputs and simply another type of abstraction of the system that is more pleasant for some users. The file is formatted for easy reading with information about each asset, the asset type, and dataflows where the asset is either a data sink or source is displayed in the file. The text-file format also allows for versioning of the file. This enables users of the tool to compare the changes of the system during its lifetime. Figure 20 is an excerpt of the resources found by CSATM in OP.V1.

```
Name: publicALB
Resource-type: AWS::ElasticLoadBalancingV2::LoadBalancer
Resource-TrustBoundary: public_cloud_network
DataFlows:
- Source: publicALB
  Destination: frontEndASG
  DataSensitivity: In Trust Boundary
- Source: Web-Clients
  Destination: publicALB
  DataSensitivity: Crosses Trust Boundary
Name: Web-Clients
Resource-type: External
Resource-TrustBoundary: External
DataFlows:
- Source: Web-Clients
  Destination: publicALB
  DataSensitivity: Crosses Trust Boundary
```

Figure 20 - Resource list from CSATM for OP.V1

Threats

Using CSATM a threat model for OP.V1 has been created.

```
{
  "publicALB": [
    "8, Accidental Cloud Data Disclosure/ Disclosure",
    "9, Misconfiguration & Exploitation of Serverless & Container Workloads",
    "1, Insufficient ID, Credential, Access and Key Mgt, Privileged Accounts",
    "3, Misconfiguration and Inadequate Change Control"
  ],
  "frontEndASG": [
    "8, Accidental Cloud Data Disclosure/ Disclosure",
    "1, Insufficient ID, Credential, Access and Key Mgt, Privileged Accounts",
    "3, Misconfiguration and Inadequate Change Control",
    "7, System Vulnerabilities",
    "5, Insecure Software Development"
  ],
  "myRDSCluster": [
    "8, Accidental Cloud Data Disclosure/ Disclosure",
    "9, Misconfiguration & Exploitation of Serverless & Container Workloads",
    "3, Misconfiguration and Inadequate Change Control"
  ],
  "ExternalAPI": [
    "2, Insecure Interfaces and APIs",
    "6, Unsecure Third Party Resources"
  ]
}
```

Figure 21 - Excerpt of threats per resource from OP.V1 threat model created by CSATM

In figure 21 we see an excerpt of the threat model. The excerpt contains one of all resource types that is in the system.

Mitigations

Using CSTAM, mitigations for the threats found has been identified. Figure 22 contains two resources from the resource, all threats found for that resource, and all mitigations found for that threat. The list of mitigations found is sizable. No good solution to determine what is the most appropriate mitigation for each threat, while keeping the tool extensible was found.

```

    "publicALB": [
        "8, Accidental Cloud Data Disclosure/ Disclosure, Mitigation id: 7 & Mitigation: Data Protection by Design and Default",
        "9, Misconfiguration & Exploitation of Serverless & Container Workloads, Mitigation id: 9 & Mitigation: Vulnerability Identification",
        "9, Misconfiguration & Exploitation of Serverless & Container Workloads, Mitigation id: 3 & Mitigation: Automated Secure Application Deployment",
        "9, Misconfiguration & Exploitation of Serverless & Container Workloads, Mitigation id: 5 & Mitigation: Automated application security testing",
        "9, Misconfiguration & Exploitation of Serverless & Container Workloads, Mitigation id: 6 & Mitigation: Quality Testing",
        "9, Misconfiguration & Exploitation of Serverless & Container Workloads, Mitigation id: 10 & Mitigation: OS Hardening and Base Controls",
        "9, Misconfiguration & Exploitation of Serverless & Container Workloads, Mitigation id: 1 & Mitigation: Least Privilege",
        "9, Misconfiguration & Exploitation of Serverless & Container Workloads, Mitigation id: 7 & Mitigation: Data Protection by Design and Default",
        "9, Misconfiguration & Exploitation of Serverless & Container Workloads, Mitigation id: 11 & Mitigation: Application Vulnerability Remediation",
        "9, Misconfiguration & Exploitation of Serverless & Container Workloads, Mitigation id: 4 & Mitigation: Data Encryption",
        "1, Insufficient ID, Credential, Access and Key Mgt, Privileged Accounts, Mitigation id: 2 & Mitigation: User access review",
        "1, Insufficient ID, Credential, Access and Key Mgt, Privileged Accounts, Mitigation id: 1 & Mitigation: Least Privilege",
        "3, Misconfiguration and Inadequate Change Control, Mitigation id: 9 & Mitigation: Vulnerability Identification",
        "3, Misconfiguration and Inadequate Change Control, Mitigation id: 5 & Mitigation: Automated application security testing",
        "3, Misconfiguration and Inadequate Change Control, Mitigation id: 2 & Mitigation: User access review",
        "3, Misconfiguration and Inadequate Change Control, Mitigation id: 6 & Mitigation: Quality Testing",
        "3, Misconfiguration and Inadequate Change Control, Mitigation id: 10 & Mitigation: OS Hardening and Base Controls",
        "3, Misconfiguration and Inadequate Change Control, Mitigation id: 7 & Mitigation: Data Protection by Design and Default",
        "3, Misconfiguration and Inadequate Change Control, Mitigation id: 4 & Mitigation: Data Encryption"
    ],
    "ExternalAPI": [
        "2, Insecure Interfaces and APIs, Mitigation id: 3 & Mitigation: Automated Secure Application Deployment",
        "2, Insecure Interfaces and APIs, Mitigation id: 6 & Mitigation: Quality Testing",
        "2, Insecure Interfaces and APIs, Mitigation id: 10 & Mitigation: OS Hardening and Base Controls",
        "2, Insecure Interfaces and APIs, Mitigation id: 4 & Mitigation: Data Encryption",
        "6, Unsecure Third Party Resources, Mitigation id: 8 & Mitigation: Supply chain risk management",
        "6, Unsecure Third Party Resources, Mitigation id: 1 & Mitigation: Least Privilege"
    ]
}

```

Figure 22 - mitigations for threats in threat model for OP.V2

Attack and defense trees

CSATM generated attack and defense trees (ADTrees) for all resources based on the threat and mitigation model created by CSATM in the step prior.

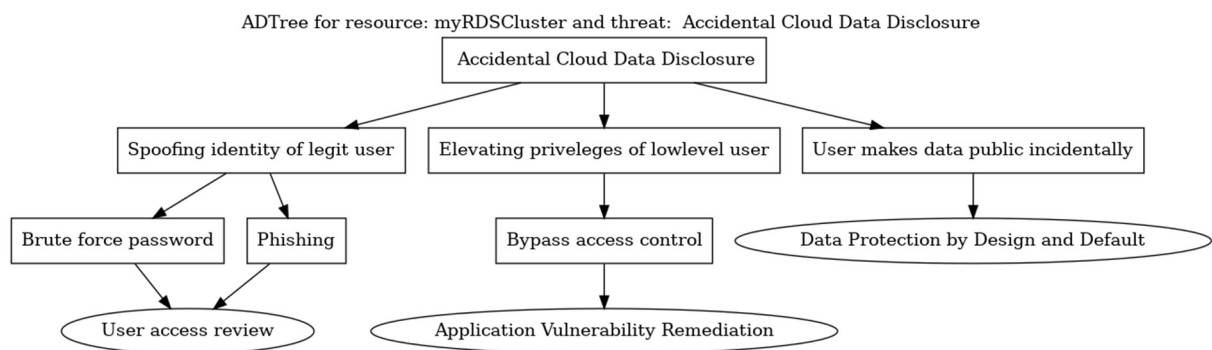


Figure 23 - ADTree for myRDSCluster and threat Accidental Cloud Data Disclosure

The ADTree in Figure 23 displays four attack paths that could lead to data disclosure that was not intended by the system owner. Three mitigations have been suggested by CSATM. Each ADTree for any resource storing or processing data with the same inherent threat looks identical, as the tool has no further context.

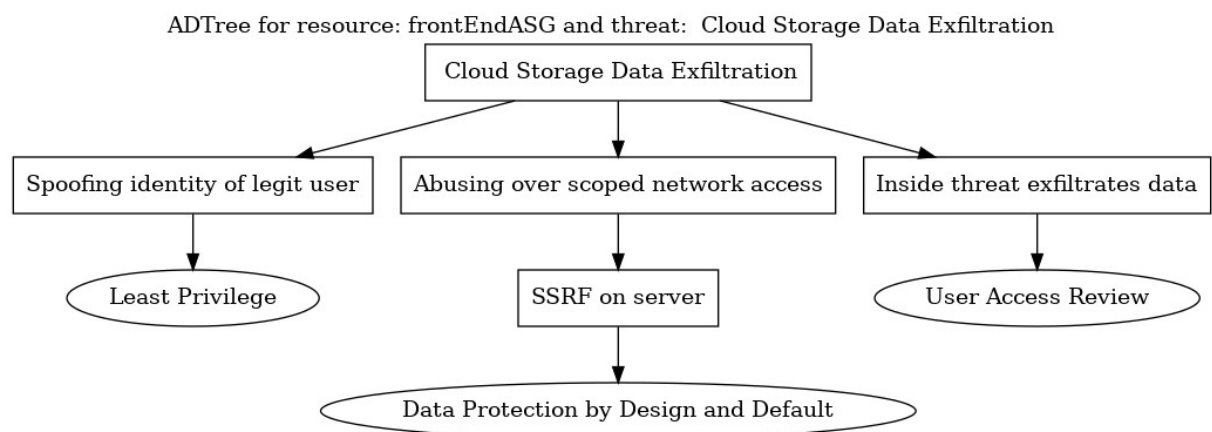


Figure 24 - ADTree for frontEndASG and threat Cloud Storage Data Exfiltration

In Figure 24 the ADTree for Cloud Storage Data Exfiltration is displayed. It contains three attack paths and three mitigations.

6.3.4 Threat modeling of OnlinePharmacy.V2 using CSATM

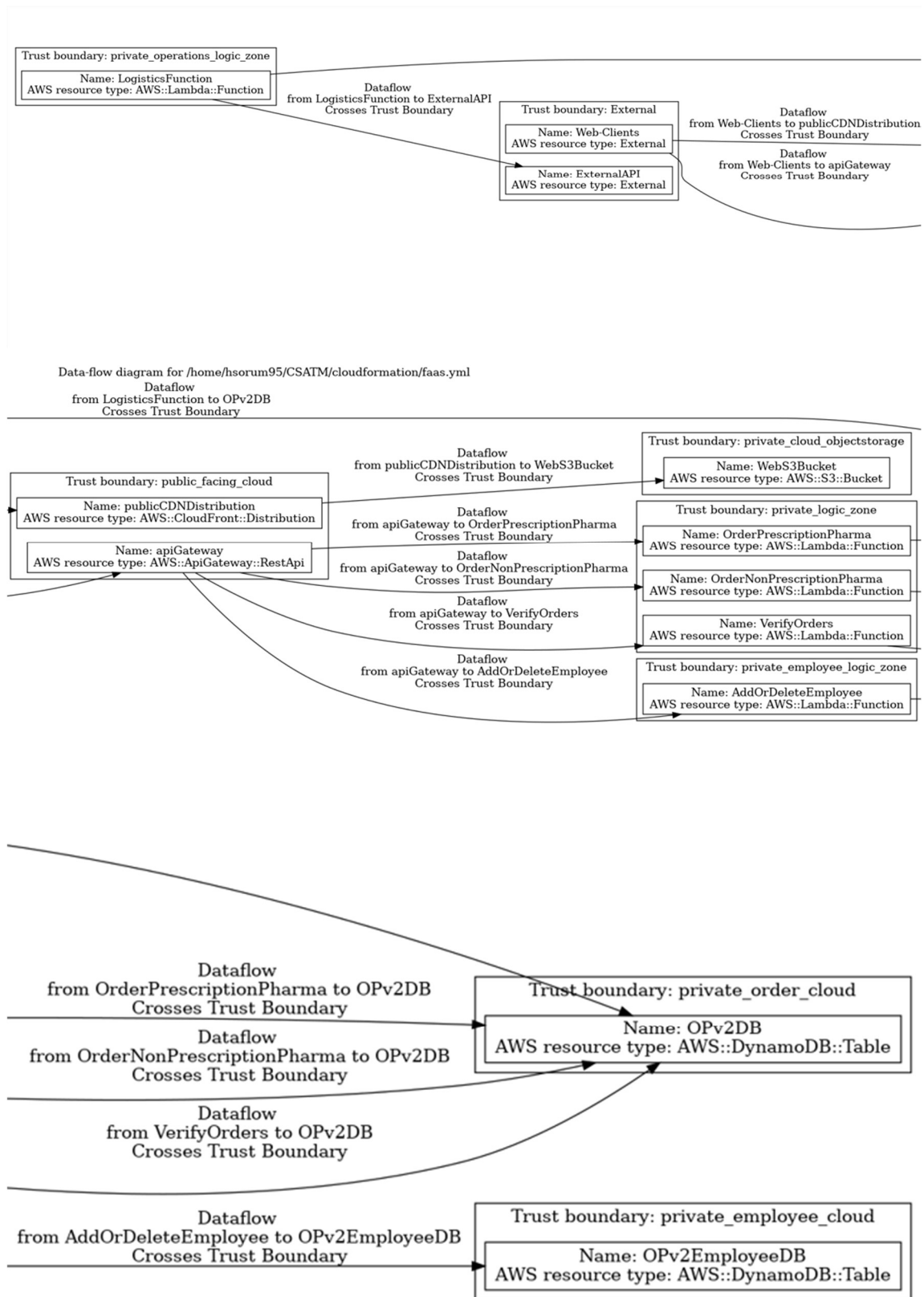


Figure 25 - DataFlowDiagram of OP.V2

Using CSATM, with the CloudFormation-template faas.yml as the input file, we get this data flow diagram for the OP.V2 system. Figure 25 displays the DFD. The DFD has been split into three outside of CSATM to better fit the format of this document. The original DFD can be found with the code and all other artifacts belonging to this thesis on <https://github.com/hsorum95/CSATM>. CSATM has identified and displayed 13 dataflows.

Resources

CSATM has also created a list of all resources within a trust boundary.

```
Name: publicCDNNDistribution
  Resource-type: AWS::CloudFront::Distribution
  Resource-TrustBoundary: public_facing_cloud
  DataFlows:
    - Source: publicCDNNDistribution
      Destination: WebS3Bucket
      DataSensitivity: Crosses Trust Boundary
    - Source: Web-Clients
      Destination: publicCDNNDistribution
      DataSensitivity: Crosses Trust Boundary
Name: apiGateway
  Resource-type: AWS::ApiGateway::RestApi
  Resource-TrustBoundary: public_facing_cloud
  DataFlows:
    - Source: apiGateway
      Destination: OrderPrescriptionPharma
      DataSensitivity: Crosses Trust Boundary
    - Source: apiGateway
      Destination: OrderNonPrescriptionPharma
      DataSensitivity: Crosses Trust Boundary
    - Source: apiGateway
      Destination: VerifyOrders
      DataSensitivity: Crosses Trust Boundary
    - Source: apiGateway
      Destination: AddOrDeleteEmployee
      DataSensitivity: Crosses Trust Boundary
    - Source: Web-Clients
      Destination: apiGateway
      DataSensitivity: Crosses Trust Boundary
```

Figure 26 - Resource list from CSATM

Figure 26 is an excerpt of resources in OP.V2. The excerpt displays two resources where each has multiple dataflows.

Threats

Using CSATM a threat model for OP.V2 has been created.

```
{
  "WebS3Bucket": [
    "8, Accidental Cloud Data Disclosure/ Disclosure",
    "9, Misconfiguration & Exploitation of Serverless & Container Workloads",
    "3, Misconfiguration and Inadequate Change Control",
  ],
  "OrderPrescriptionPharma": [
    "8, Accidental Cloud Data Disclosure/ Disclosure",
    "9, Misconfiguration & Exploitation of Serverless & Container Workloads",
    "3, Misconfiguration and Inadequate Change Control",
    "7, System Vulnerabilities",
    "5, Insecure Software Development"
  ],
  "OPv2DB": [
    "8, Accidental Cloud Data Disclosure/ Disclosure",
    "9, Misconfiguration & Exploitation of Serverless & Container Workloads",
    "3, Misconfiguration and Inadequate Change Control"
  ],
  "ExternalAPI": [
    "2, Insecure Interfaces and APIs",
    "6, Unsecure Third Party Resources"
  ]
}
```

Figure 27 - Excerpt of threats per resource from OP.V2 threat model created by CSATM

In figure 27 we see an excerpt of the threat model. The excerpt contains one of all resource types in the system which had a threat related to it.

Mitigations

Using CSTAM, mitigations for the threats found have been identified. Figure 28 contains two resources from the resource, all threats found for that resource, and all mitigations found for that threat. The list of mitigations found is sizable. No good solution to determine what is the most appropriate mitigation for each threat, while keeping the tool extensible was found.

```

{
  "WebS3Bucket": [
    "9, Misconfiguration & Exploitation of Serverless & Container Workloads, Mitigation id: 9 & Mitigation: Vulnerability Identification",
    "9, Misconfiguration & Exploitation of Serverless & Container Workloads, Mitigation id: 3 & Mitigation: Automated Secure Application Deployment",
    "9, Misconfiguration & Exploitation of Serverless & Container Workloads, Mitigation id: 5 & Mitigation: Automated application security testing",
    "9, Misconfiguration & Exploitation of Serverless & Container Workloads, Mitigation id: 6 & Mitigation: Quality Testing",
    "9, Misconfiguration & Exploitation of Serverless & Container Workloads, Mitigation id: 10 & Mitigation: OS Hardening and Base Controls",
    "9, Misconfiguration & Exploitation of Serverless & Container Workloads, Mitigation id: 1 & Mitigation: Least Privilege",
    "9, Misconfiguration & Exploitation of Serverless & Container Workloads, Mitigation id: 7 & Mitigation: Data Protection by Design and Default",
    "9, Misconfiguration & Exploitation of Serverless & Container Workloads, Mitigation id: 11 & Mitigation: Application Vulnerability Remediation",
    "9, Misconfiguration & Exploitation of Serverless & Container Workloads, Mitigation id: 4 & Mitigation: Data Encryption",
    "3, Misconfiguration and Inadequate Change Control, Mitigation id: 9 & Mitigation: Vulnerability Identification",
    "3, Misconfiguration and Inadequate Change Control, Mitigation id: 5 & Mitigation: Automated application security testing",
    "3, Misconfiguration and Inadequate Change Control, Mitigation id: 2 & Mitigation: User access review",
    "3, Misconfiguration and Inadequate Change Control, Mitigation id: 6 & Mitigation: Quality Testing",
    "3, Misconfiguration and Inadequate Change Control, Mitigation id: 10 & Mitigation: OS Hardening and Base Controls",
    "3, Misconfiguration and Inadequate Change Control, Mitigation id: 7 & Mitigation: Data Protection by Design and Default",
    "3, Misconfiguration and Inadequate Change Control, Mitigation id: 4 & Mitigation: Data Encryption"
  ],
  "Web-Clients": [
    "2, Insecure Interfaces and APIs, Mitigation id: 3 & Mitigation: Automated Secure Application Deployment",
    "2, Insecure Interfaces and APIs, Mitigation id: 6 & Mitigation: Quality Testing",
    "2, Insecure Interfaces and APIs, Mitigation id: 10 & Mitigation: OS Hardening and Base Controls",
    "2, Insecure Interfaces and APIs, Mitigation id: 4 & Mitigation: Data Encryption",
    "6, Unsecure Third Party Resources, Mitigation id: 8 & Mitigation: Supply chain risk management",
    "6, Unsecure Third Party Resources, Mitigation id: 1 & Mitigation: Least Privilege"
  ]
}

```

Figure 28 - mitigations for threats in threat model for OP.V2

Attack and defense trees

CSATM generated attack and defense trees (ADTrees) for all resources based on the threat and mitigation model created by CSATM in the step prior.

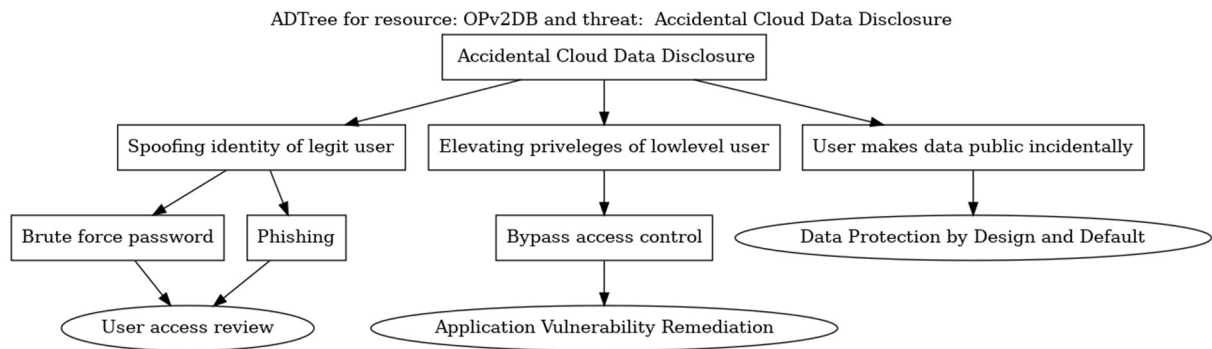


Figure 29 - ADTree for OPv2EmployeeDB and threat Accidental Cloud Data Disclosure

Figure 29 is one of many ADTrees generated for OP.V2. The ADTree contains four attack paths and three mitigations. As all resources of the same service-type in the system are defined the same in the code the tree is identical to any other ADTree for the same threat and resource-type.

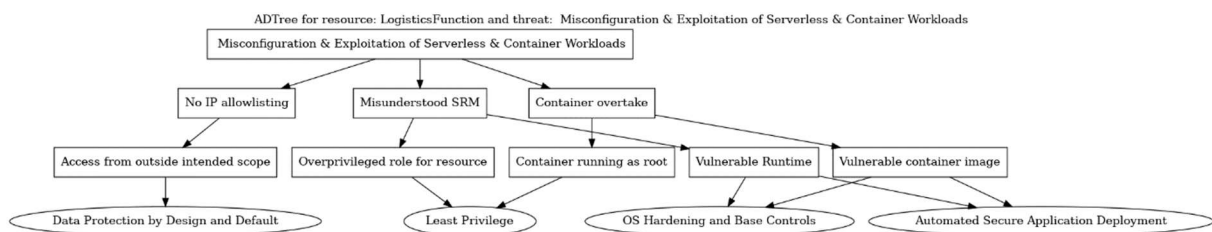


Figure 30 - ADTree for LogisticsFunction and threat Exploitation of serverless and container workload.

The ADTree in figure 30 contains five attack paths and four mitigations. The ADTree contains some mitigations remove multiple attack paths. These can be viewed as more effective mitigations than those that only remove one attack path. This is of course dependent on the cost of implementing the control and the likelihood of the paths in question.

6.3.5 Comparison of manual and automatic threat models

The manual threat modeling provided more concrete threats. These are very applicable to the system, but require a lot of context and information that isn't easily accessible by the input the tool takes. The IaC doesn't contain enough information to deduce concrete threats. The output

from CSATM is at times only an enumeration of mappings between threats and mitigations already done. The DFDs generated by CSATM are of high value and could be used as augmentation in a subsequent manual process. The threat model and mitigations list could also be used as a starting point for a subsequent manual process.

One approach which could have provided more detailed and specific threat models were if the tool had more information about the defaults of the cloud services. With this information the tool could know what settings and configurations the resources would have if nothing else was explicitly programmed in the IaC. This could be regarding security related settings such as encryption, access, and authentication.

The security posture of a system is not improved by threat modeling alone. The quality of the remediations, mitigations or risk awareness is what can increase the cyber resiliency of a system or an organization. An incomplete or bad threat model can be a security risk if we treat the threat model as the complete representation of technical risk associated with the entity. The resulting security effort can be lacking if based on such a threat model. This is also underpinned by the fact that no model is a complete representation of the reality, but an abstraction to create understanding of the entity being modelled. Key threats or risks can be left out when abstracting. In this aspect the threat model created by CSATM is better than the manual. While very high level, nothing is left out if it is a part of the threat catalog.

7 Discussion

In this chapter the results of the case study will be evaluated, and the usefulness of partially automating threat modeling will be discussed.

7.1 Manual threat modeling of Online Pharmacy.V1 and Online Pharmacy.V2

Manually threat modeling a system is cumbersome and complicated. The modeler or modelers must first get a good grasp of the intended function of the system. Then they must start breaking down the system to understand how they interact. Following this the creative process of analyzing what threats can be relevant for the system and how might those threats compromise the security goals. When the threats are systematically documented modelers must think of ways to either mitigate or remediate the threats with new design patterns.

When manually modeling the OP.V1 system as a part of this case study, many of the threats are caused by the same design flaw, lacking encryption on the network traffic internally. This could be due to this flaw being especially bad in terms of security posture, or that the modeler was biased in a way that affected the modeler to focus on this error. There is a possibility that this wouldn't have been a problem if the modeling was done by a group instead of an individual threat modeler.

For OP.V2 the threats modeled are more diverse, but very broad. They are also related to implementation-time configuration and settings. This is most probably due to the usage of PaaS-services where the CSP assumes more responsibility of the security posture of the system. OP.V2 is more prone to vulnerabilities caused by poor implementation of the software or security misconfigurations. Those security problems can't be modeled prior to implementation.

It is very hard to determine what is and what is not a good threat model. Adam Shostack is paraphrased "a good threat model is one that produces valid findings" [77]. This thesis does therefor not try to conclude on whether the manual threat model created for the systems is a high-quality threat model.

7.2 Automated threat modeling of Online Pharmacy.V1 and Online Pharmacy.V2

The threat catalog used by CSATM in the case study mixed between threats that were symptoms of an incident rather than the cause. This affected the quality of the threat model. Focusing on symptoms rather causes can also cause ineffective use of resources to improve the security of the system. One could rank the mitigations based on occurrences in the output and prioritize based on that ranking. If the threats were more focused on causes, then this would be even more valuable.

Another challenge with the lacking focus in the threat catalog is that it made creating ADTrees more complex. The root node of an ADTree is what the adversary wants to achieve, such as was displayed in Figure 5 “Open Safe”. Some of the threats in the catalog that do not fit well with this. They were: “Misconfiguration and Inadequate Change Control”, “Insufficient ID, Credential, Access and Key Mgt, Privileged Accounts” and “System Vulnerabilities”. These are causes, while other threats such as “Accidental Cloud Data Disclosure/ Disclosure” and “Cloud Storage Data Exfiltration” are symptoms. Lastly there was one threat that is a combination “Misconfiguration & Exploitation of Serverless & Container Workloads”. A misconfiguration can be the cause of an incident, while the exploitation is a symptom of a breach. For this thesis this was solved by only creating ADTrees for the three threats that are symptoms.

To solve this the threat catalog should consist of only causes, whilst being mapped to symptoms of such cause. This would allow the algorithm to build out more valuable attack trees across our threats, mitigations, remediations, symptoms and causes.

7.3 Considerations, discussion and learnings based on CSATM

In chapter 4.2 using CWEs, CVEs and the MITRE ATT&CK framework as threat input into CSATM was discussed. This can be a good improvement of the current tool, but implementing this ended up not being prioritized. The implementation itself proved not to be trivial, and there were other factors as well. First off none of these are cloud specific or

targeted at cloud security. Uncertainty whether we would enrich the field of cloud security by implementing these lead the focus elsewhere.

CVEs are known vulnerabilities in software. To detect CVEs in our design we would need insight into the application code. We only had the IaC. As proposed in chapter 4.2 using SAST and SCA-tools we could have gathered this information and used in as proposed in that chapter in situations where application code is accessible.

The MITRE ATT&CK framework was not implemented as an input as there was not enough data about the system contained in the IaC to do this successfully.

CWEs are descriptions of known common weaknesses. These relate much better to traditional threat modeling than the previous two frameworks. In this thesis the threat catalog could contain mappings to CWEs, but we chose to rather focus the time elsewhere as the threats were found to be descriptive enough.

The tacit knowledge of the threat modeling group can be lost when parts of the process are handled by an automatic or semi-automatic tool. Quantifying the knowledge of the group is challenging, but must not be forgotten when reviewing the results.

7.4 State of current tooling and comparison

The opensource tools discussed in chapter 2.4 and chapter 3 are mostly tools built to support individuals or teams conducting threat modeling. They reduce the overhead and simplify the process without compromising the effort put into thinking about how the system can be abused. For this purpose, the tools can support many different types of workflows. This is the same purpose the tool presented in this thesis tries to achieve. An assumption that is made with the CSATM is that the modelers already have adequate understanding of their own infrastructure and how it's supposed to function. Therefore the time spent creating diagrams and other artifacts used in traditional threat modeling is superfluous. This is of course the same assumption seemingly made by other tools as for example Threatspec. The main differentiator between Threatspec and CSATM is the perspective that is modeled. Threatspec has an application code focus, and doesn't rely on how the application is hosted, while CSATM has a more infrastructure, network and systems-based perspective.

The commercially available tools mentioned in this thesis have not been investigated thoroughly, but their messaging suggest they want to reduce the need for manual labor to a minimum. What is written both in the academic literature, and by trusted industry leaders suggest that this is the wrong approach to the problem. The threat model itself gives little or no value when the organization doesn't have intimate knowledge of how it was created or gained familiarity with the threats themselves and how they might be mitigated or remediated. Whilst having a systematic record of threats to the system is beneficial, enabling security actions based on this isn't trivial as the threats are not an agreement within the group that builds and maintains the system, but rather what can be seen as an outsider's perspective on the effort of the group.

7.5 Advantages of manual threat modeling

As discussed previously threat modeling contributes value to the security goals of a system in more ways than just the threat model. Mitigation and remediations designed based on identified threats, and the improved knowledge and understanding of the system is valuable for the organization. Manual threat modeling with cross-functional teams also educates and upskill the individuals involved in the process. This is beneficial to both the individuals and the organization.

Manual threat modeling is very adaptive. Any individual or organization can create their own framework, processes, methodologies or approaches. This can lead to better implementations of threat modeling for that individual or organization. Building out threat libraries and attack taxonomies for the threat landscape relevant to the system can be done “on the fly” while modeling. For automated threat modeling the up-front investment in creating good threat libraries and attack taxonomies is larger as there has to be a baseline of input to the automation engine to provide any valuable output.

7.6 Advantages of automated or partially automated threat modeling

Automated threat modelings main advantage is the alleged time savings and scalability. The impact of this advantage is determined by the approach and framework chosen. No studies have been found that quantifies the reduction in time spent and scalability when automating threat modeling compared to manual threat modeling. It is therefore challenging to conclude how large this upside is, if there is an upside at all. Security teams in organizations are outnumbered by multiples as large as 10x or 100x compared to development teams. When comparing application security experts or even threat modeling experts to developers these factors increase even more. This indicates that security teams must enable, educate and where possible automate part of the threat modeling process to ensure high quality threat models in their organization. Automation can be specific to the domain of the threat model. It proved hard to engineer automation for generic cloud environments.

The bias introduced in the threat model by the modelers is still relevant even for automated threat modeling. The threat libraries used has been created by an individual or a group who might have biases themselves. When using automated threat modeling the bias is introduced through the library used. An example is western citizens might only include foreign and threat actors from non-allied countries in their threat landscape assessment, even if cyber operations can be used against allied countries. Another would be the tendency to introduce threats one has experienced often in real life. These might not be more prevalent in reality, but due to the personal experience of the creator they will appear more often in threat models. This insufficient library or taxonomy would leave the threat model lacking. By using multiple and diverse threat libraries this can be more easily removed from the perspective of an automated modeling process than from a manual one. In this thesis where the modeler was allowed to refine the model and re-run the tool the modeler can introduce personal biases based on their own experiences and background.

The use of CSATM is most successful and beneficial when used in a normal threat modeling setting where a diverse group of representatives partake. The tool aids and adds additional context to base informed discussions about the system and the corresponding threat model, and might reduce the time and resources required from security experts.

8 Conclusion

In this chapter I will conclude my thesis and answer the research question for this thesis.

The aim of this thesis was to study the feasibility of automating threat modeling of cloud deployed applications. The thesis also studied whether fully automating it or if a semi-automated approach is more appropriate. Furthermore, the impact of IaC on threat modeling and integration of a threat modeling tool into a SDLC has been discussed.

Starting with research question I, the results from the case study suggests that automating threat modeling is feasible, but requires a lot of engineering to be a sustainable solution compared to a more manual process. A concern to keep in mind is that threat modeling is ideally an activity that starts before a single line of code is written. When creating the threat model based on the IaC that defines the system we cannot uphold this goal.

Secondly, research question II, proved difficult to answer in a satisfying way with the case study approach. The relevance and quality of a threat model is as previously discussed challenging to determine. This fact was not taken into consideration when designing the approach. What can be said is that manual refinement did remove threats that were not relevant due to factors outside of the IaC. This shows that a semi-automated approach has signs of promise.

The third research question III and the fourth research question IV are tightly connected. IaC makes automated threat modeling of public cloud environments during and after the writing of the code feasible. Another approach would be to create a tool that does threat modeling on other inputs like diagrams or drawings. This would create more work compared to what was show in this thesis as the automated tool creates these diagrams based on the code. Integrating the tool into a SDLC pipeline is a trivial task and the tool was used in this fashion when creating the IaC.

8.1 Future work

The tool showed promise, but need more engineering. Extending it to other applicable IaC-languages and other CSPs could be interesting. Creating a more structured and more comprehensive threat catalog and mitigation catalog for cloud security would improve the output of the tool. This could make automating threat modeling for cloud security more plausible. To further test the research questions applying the study in a real-life scenario with an organization or business already doing threat modeling could be a good follow up.

Focusing on the refinement and augmentation of the ADTrees further in academia would be interesting. This would require a better and more structured threat and mitigation catalog. The trees were not sufficiently generated due to the lack of structured data.

Bibliography

- [1] S. Planning, "The economic impacts of inadequate infrastructure for software testing," *National Institute of Standards and Technology*, 2002.
- [2] C. Security. "The 2021 State of Threat Modeling: An interactive e-book publication " <https://resources.securitycompass.com/reports/2021-state-of-threatmodeling#main-content> (accessed).
- [3] M. Howard. <https://threatsmanager.com/testimonial/> (accessed 07.09, 2022).
- [4] K. Rindell, J. Ruohonen, J. Holvitie, S. Hyrynsalmi, and V. Leppänen, "Security in agile software development: A practitioner survey," *Information and Software Technology*, vol. 131, p. 106488, 2021/03/01/ 2021, doi: <https://doi.org/10.1016/j.infsof.2020.106488>.
- [5] (2021). *Executive Order 14028, Improving the Nation's Cybersecurity - Recommended Minimum Standard for Vendor or Developer Verification of Code*. [Online] Available: <https://www.nist.gov/itl/executive-order-improving-nations-cybersecurity/recommended-minimum-standard-vendor-or-developer>
- [6] NIST. "Cyber Security." <https://csrc.nist.gov/glossary/term/cybersecurity> (accessed).
- [7] NIST. "Computer Security Resource Center Glossary." <https://csrc.nist.gov/glossary/> (accessed 22.05, 2021).
- [8] ENISA. "Threat and Risk Management - Glossary." <https://www.enisa.europa.eu/topics/threat-risk-management/risk-management/current-risk/risk-management-inventory/glossary> (accessed 22.05, 2021).
- [9] S. Harris and F. Maymi, *CISSP All-in-One Exam Guide, Seventh Edition*. McGraw-Hill Education, 2016.
- [10] Mandiant. "Advanced Persistent Threats (APTs)." <https://www.mandiant.com/resources/insights/apt-groups> (accessed 14.09, 2022).
- [11] B. O. England, "CBEST Intelligence-Led Testing Understanding Cyber Threat Intelligence Operations," ed, 2016.
- [12] MITRE. "MITRE ATT&CK." <https://attack.mitre.org/> (accessed 25.04, 2022).
- [13] NIST. "tactics, techniques, and procedures (TTP)." cti. https://csrc.nist.gov/glossary/term/Tactics_Techniques_and_Procedures (accessed 25.04, 2022).
- [14] P. Mell and T. Grance, "The NIST definition of cloud computing," *NIST Special Publication* 2011.
- [15] H. Saini, A. Upadhyaya, and M. K. Khandelwal, "Benefits of cloud computing for business enterprises: A review," in *Proceedings of International Conference on Advancements in Computing & Management (ICACM)*, 2019.
- [16] K. Z. Thwe and P. P. Khaing, "Cloud Computing: Benefits and Limits on SPI model," MERAL Portal.
- [17] (2011). *Special Publication 800-145 - The NIST Definition of Cloud Computing*. [Online] Available: <https://doi.org/10.6028/NIST.SP.800-145>
- [18] AWS. "Shared Responsibility Model." AWS. <https://aws.amazon.com/compliance/shared-responsibility-model/> (accessed 22.05, 2021).
- [19] NCSC, "Cloud security shared responsibility model," ed, 2021.
- [20] CloudPassage. "Shared Responsibility Model Explained." Cloud Security Alliance. <https://cloudsecurityalliance.org/blog/2020/08/26/shared-responsibility-model-explained/> (accessed 22.05, 2021).

- [21] CIS, "Cloud Security and the Shared Responsibility Model with CIS," 2020. [Online]. Available: <https://www.cisecurity.org/blog/shared-responsibility-cloud-security-what-you-need-to-know/>
- [22] NSA, "Mitigating Cloud Vulnerabilities," 2020. [Online]. Available: https://media.defense.gov/2020/Jan/22/2002237484/-1/-1/0/CSI-MITIGATING-CLOUD-VULNERABILITIES_20200121.PDF
- [23] C. S. Alliance, "Top Threats to Cloud Computing - Pandemic Eleven," in "Top Threats to Cloud Computing ", RSA Conference, 2022, vol. 6. [Online]. Available: <https://cloudsecurityalliance.org/artifacts/top-threats-to-cloud-computing-pandemic-eleven/>
- [24] A. Johnson, K. Dempsey, R. Ross, S. Gupta, and D. Bailey, "Guide for security-focused configuration management of information systems," *NIST special publication*, vol. 800, no. 128, pp. 16-16, 2011.
- [25] K. Panetta. "Is the Cloud Secure?" <https://www.gartner.com/smarterwithgartner/is-the-cloud-secure/> (accessed 22.05, 2021).
- [26] B. Schneier. "A Plea for Simplicity." https://www.schneier.com/essays/archives/1999/11/a_plea_for_simplicit.html (accessed 22.05, 2021).
- [27] U. 42, "Cloud Threat Report - IDENTItY AND ACCESS MANAGEMENT (IAM) The First Line of Defense ", 2022, vol. Volume 6. Accessed: 29.10.2022. [Online]. Available: <https://www.paloaltonetworks.com/prisma/unit42-cloud-threat-research-volume-six>
- [28] C. S. Alliance. "Cloud Controls Matrix and CAIQ v4." <https://cloudsecurityalliance.org/artifacts/cloud-controls-matrix-v4/> (accessed 2022).
- [29] I. S. O. ISO. "Risk Managment: ISO 31000." <https://www.iso.org/files/live/sites/isoorg/files/store/en/PUB100426.pdf> (accessed 22.05, 2021).
- [30] I. S. O. ISO, "Information security risk management: ISO 27005," 2018. [Online]. Available: <https://www.iso.org/obp/ui/#iso:std:iso-iec:27005:ed-3:v1:en>.
- [31] I. S. O. ISO, "ISO Guide 73:2009 - Risk management — Vocabulary," ISO, 2009. [Online]. Available: <https://www.iso.org/obp/ui/#iso:std:iso:guide:73:ed-1:v1:en>
- [32] A. Jøsang, "Del 7: Risikostyring for informasjonssikkerhet," UiO, 2020.
- [33] "Definition of Risk Level." Stanford University. <https://ocro.stanford.edu/erm/key-definitions/definition-risk-level> (accessed 22.05, 2021).
- [34] R. Ross *et al.*, "Guide for Conducting Risk Assessments," *NIST Special Publication* 2012, doi: 10.6028/NIST.SP.800-30r1.
- [35] I. S. O. ISO. "risk appetite." <https://www.iso.org/obp/ui/#iso:std:iso:guide:73:ed-1:v1:en> (accessed 06.22, 2022).
- [36] "risk treatment." <https://www.iso.org/obp/ui/#iso:std:iso:guide:73:ed-1:v1:en:term:3.8.1> (accessed.
- [37] I. S. O. ISO. "risk tolerance." <https://www.iso.org/obp/ui/#iso:std:iso:guide:73:ed-1:v1:en:term:3.7.1.3> (accessed 06.02, 2022).
- [38] M. Souppaya and K. Scarfone, "Guide to data-centric system threat modeling," National Institute of Standards and Technology, 2016.
- [39] A. V. Uzunov and E. B. Fernandez, "An extensible pattern-based library and taxonomy of security threats for distributed systems," *Computer Standards & Interfaces*, vol. 36, no. 4, pp. 734-747, 2014/06/01/ 2014, doi: <https://doi.org/10.1016/j.csi.2013.12.008>.

- [40] G. E. P. Box, "Science and Statistics," *Journal of the American Statistical Association*, vol. 71, no. 356, pp. 791-799, 1976/12/01 1976, doi: 10.1080/01621459.1976.10480949.
- [41] M. Ekstedt. "Where Is The Trust Boundary?" foreseeti. <https://foreseeti.com/trust-boundary/> (accessed 11.10, 2022).
- [42] A. Shostack, *Threat modeling: Designing for security*. John Wiley & Sons, 2014.
- [43] M. Tatam, B. Shanmugam, S. Azam, and K. Kannoorpatti, "A review of threat modelling approaches for APT-style attacks," *Heliyon*, vol. 7, no. 1, p. e05969, 2021/01/01/ 2021, doi: <https://doi.org/10.1016/j.heliyon.2021.e05969>.
- [44] "The CORAS Method." <https://coras.sourceforge.net/> (accessed 2022).
- [45] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *Leading Issues in Information Warfare & Security Research*, vol. 1, no. 1, p. 80, 2011.
- [46] Microsoft. <https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling> (accessed 06.09, 2022).
- [47] N. Shevchenko. "Threat Modeling: 12 Available Methods " Carnegie Mellon University Software Engineering Institute. <https://insights.sei.cmu.edu/blog/threat-modeling-12-available-methods/> (accessed 22.05, 2021).
- [48] K. T. Vaibhav Garg, Rik Farrow, "An Analysis of Open-source Automated Threat Modeling Tools and Their Extensibility from Security into Privacy," ed.;login:: USENIX - The Advanced Computing Systems Association, 2022.
- [49] "Threatspec," ed, 2019.
- [50] B. Schneier. "Attack Trees." Dr. Dobb's Journal. https://www.schneier.com/academic/archives/1999/12/attack_trees.html (accessed 22.05, 2021).
- [51] A. Buldas, O. Gadyatskaya, A. Lenin, S. Mauw, and R. Trujillo-Rasua, "Attribute evaluation on attack trees with incomplete information," *Computers & Security*, vol. 88, p. 101630, 2020.
- [52] B. Kordy, S. Mauw, S. Radomirović, and P. Schweitzer, "Foundations of attack–defense trees," in *International Workshop on Formal Aspects in Security and Trust*, 2010: Springer, pp. 80-95.
- [53] J. Davis and R. Daniels, *Effective DevOps: building a culture of collaboration, affinity, and tooling at scale*. " O'Reilly Media, Inc.", 2016.
- [54] " Manifesto for Agile Software Development " <https://agilemanifesto.org/> (accessed 2022).
- [55] Gartner. "DevSecOps." <https://www.gartner.com/en/information-technology/glossary/devsecops> (accessed 30.04, 2022).
- [56] J. Bird, *DevOpsSec*. O'Reilly Media, Inc, 2016.
- [57] W. Xiong and R. Lagerström, "Threat modeling—A systematic literature review," *Computers & security*, vol. 84, pp. 53-69, 2019.
- [58] P. Kamongi and K. K. Mahadevan Gomathisankaran, "Nemesis: Automated Architecture for Threat Modeling and Risk Assessment for Cloud Computing," 2014.
- [59] P. Kamongi, S. Kotikela, K. Kavi, M. Gomathisankaran, and A. Singhal, "Vulcan: Vulnerability assessment framework for cloud computing," in *2013 IEEE 7th International Conference on Software Security and Reliability*, 2013: IEEE, pp. 218-226.
- [60] R. Jhawar, K. Lounis, S. Mauw, and Y. Ramírez-Cruz, "Semi-automatically augmenting attack trees using an annotated attack tree library," in *International Workshop on Security and Trust Management*, 2018: Springer, pp. 85-101.

- [61] S. Paul, "Towards Automating the Construction & Maintenance of Attack Trees: a Feasibility Study," 2014.
- [62] P. Johnson, A. Vernotte, M. Ekstedt, and R. Lagerström, "pwnpr3d: an attack-graph-driven probabilistic threat-modeling approach," in *2016 11th International Conference on Availability, Reliability and Security (ARES)*, 2016: IEEE, pp. 278-283.
- [63] M. Välja, F. Heiding, U. Franke, and R. Lagerström, "Automating threat modeling using an ontology framework," *Cybersecurity*, vol. 3, no. 1, p. 19, 2020/10/01 2020, doi: 10.1186/s42400-020-00060-8.
- [64] I. Barankova, U. Mikhailova, and M. Afanaseva, "Minimizing information security risks based on security threat modeling," *Journal of Physics: Conference Series*, vol. 1441, p. 012031, 01/01 2020, doi: 10.1088/1742-6596/1441/1/012031.
- [65] Z. Aslanyan, F. Nielson, and D. Parker, "Quantitative verification and synthesis of attack-defence scenarios," in *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, 2016: IEEE, pp. 105-119.
- [66] B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," *Future Generation Computer Systems*, vol. 79, pp. 849-861, 2018.
- [67] W. Xiong, E. Legrand, O. Åberg, and R. Lagerström, "Cyber security threat modeling based on the MITRE Enterprise ATT&CK Matrix," *Software and Systems Modeling*, vol. 21, no. 1, pp. 157-177, 2022/02/01 2022, doi: 10.1007/s10270-021-00898-7.
- [68] P. Johnson, R. Lagerström, and M. Ekstedt, "A Meta Language for Threat Modeling and Attack Simulations," presented at the Proceedings of the 13th International Conference on Availability, Reliability and Security, Hamburg, Germany, 2018. [Online]. Available: <https://doi-org.ezproxy.uio.no/10.1145/3230833.3232799>.
- [69] "Threat Modeling Manifest." <https://www.threatmodelingmanifesto.org/> (accessed 01.05, 2022).
- [70] A. Brazhuk, "Threat modeling of cloud systems with ontological security pattern catalog," 05/04 2021.
- [71] OWASP. "OWASP Ontology Driven Threat Modeling Framework." <https://owasp.org/www-project-ontology-driven-threat-modeling-framework/> (accessed 01.05, 2022).
- [72] OWASP. "Threat Modeling Cheat Sheet." https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html (accessed 01.05, 2022).
- [73] OWASP. "OWASP Threat Modeling Playbook (OTMP)." <https://owasp.org/www-project-threat-modeling-playbook/> (accessed 01.05, 2022).
- [74] OWASP. "OWASP pytm." <https://owasp.org/www-project-pytm/> (accessed 01.05, 2022).
- [75] OWASP. "OWASP Threat Dragon." <https://owasp.org/www-project-threat-dragon/> (accessed 01.05, 2022).
- [76] K. Bernsmed, D. S. Cruzes, M. G. Jaatun, and M. Iovan, "Adopting threat modelling in agile software development projects," *Journal of Systems and Software*, vol. 183, p. 111090, 2022.
- [77] Izar Tarandach and M. J. Coles, *Threat Modeling - A Practical Guide for Development Teams*. O'Reilly Media, 2020.
- [78] AWS. "Testing IAM policies with the IAM policy simulator." https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_testing-policies.html (accessed 30.04, 2022).

- [79] M. Fraile, M. Ford, O. Gadyatskaya, R. Kumar, M. Stoelinga, and R. Trujillo-Rasua, "Using attack-defense trees to analyze threats and countermeasures in an ATM: a case study," in *IFIP Working Conference on The Practice of Enterprise Modeling*, 2016: Springer, pp. 326-334.
- [80] R. R. Hansen, P. G. Jensen, K. G. Larsen, A. Legay, and D. B. Poulsen, "Quantitative evaluation of attack defense trees using stochastic timed automata," in *International Workshop on Graphical Models for Security*, 2017: Springer, pp. 75-90.
- [81] R. Jhawar, K. Lounis, and S. Mauw, "A stochastic framework for quantitative analysis of attack-defense trees," in *International Workshop on Security and Trust Management*, 2016: Springer, pp. 138-153.
- [82] Debian. "Debian " <https://www.debian.org/intro/philosophy.en.html> (accessed 2022).
- [83] graphviz. "graphviz." <https://graphviz.org/about/> (accessed 2022).
- [84] Kubernetes. "Kubernetes Readme." Kubernetes <https://github.com/kubernetes/kubernetes#readme> (accessed 22.09, 2022).
- [85] Solutio. "Kamus." Solutio. <https://github.com/Solutio/kamus#readme> (accessed 22.09, 2022).