Improving Security in Software-as-a-Service Solutions

CHRISTOFFER TOFT
VIKTOR EDÉUS
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Improving Security in Software-as-a-Service Solutions

Christoffer Toft and Viktor Edéus
{dic11cto, dic11ved}@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisor: Martin Hell

Examiner: Thomas Johansson

June 7, 2017

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

The essence of cloud computing is about moving workloads from your local IT infrastructure to a data center that scales and provides resources at a moments notice. Using a pay-as-you-go model to rent virtual infrastructure is also known as a *"infrastructure-as-a-service"* (IaaS) offering. This helps consumers provision hardware on-demand without the need for physical infrastructure and the challenges and costs that come with it. When moving to the cloud, however, issues regarding the confidentiality, integrity, and availability of the data and infrastructure arise, and new security challenges compared to traditional on-premises computing appear. It is important for the consumer to know exactly what is their responsibility when it comes to securing software running on IaaS platforms. Axis has one such software solution, henceforth referred to as the 'Axis-hosted cloud service'. There is a need for Axis to improve the client-cloud communication, and in this report, we detail a prototype solution for a new secure communication between client and cloud. Additionally, an evaluation of the prototype is presented. The evaluation is based on a model constructed by studying literature from state-of-the-art cloud service providers and organizations dedicated to defining best practices and critical areas of focus for cloud computing. This was collected and compiled in order to present a summary of the most important aspects to keep in mind when deploying software on an IaaS. It showed that the cloud service fulfills many industry best-practices, such as encrypting data in transit between client and cloud, using virtual private clouds to separate infrastructure credentials from unauthorized access, and following the guidelines from their infrastructure provider. It also showed areas where there was a need for improvement in order to reach a state-of-the-art level. The model proved to be a useful tool to ensure that security best practices are being met by an organization moving to the cloud, and specifically for Axis, the prototype communication solution can be used as a base for further development.

Popular Science Summary

Imagine being the victim of an accident that was recorded by a surveillance camera. You would probably hope that the camera detected the accident and sent an alarm to someone who could help you? Imagine then that the communication had been tampered with, blocking the alarm to the operator and leaving you to your own devices! The result of this thesis in part stops this situation from happening. At the intersection of networking and surveillance, Axis Communications delivers network-connected products that help provide physical safety for every type of business. For this to work, the products must be safe from attacks by malicious entities. Blocking alarm signals may result in trespassing, theft and even physical damage to people and property, and as such is critical to solve. The solution presented in this paper solves this problem by sending these alarms via secure communications channels, increasing the security of the product. This has implications for everyone using Axis' products. The authors also evaluated one of Axis' cloud based solutions for controlling network cameras based on a self-developed model for determining if it adheres to best practices in the field. It showed that the cloud service has some best practices in place, but also some flaws that might be of concern. Fixing these is crucial for the company in order to secure the data of its customers. The model is a tool for companies who deploy or wish to deploy their software to the cloud and wish to heighten their overall security in order to protect their own and their customers data. It is based on a literature study with the purpose of illuminating the responsibilities and issues when leasing computational power in the cloud.

Table of Contents

List of Figures

List of Tables

List of Acronyms

CGI - Common Gateway Interface
CSA - Cloud Security Alliance
CSP - Cloud Service Provider
DMZ - De-Militarized Zone
HIPS - Host Intrusion Prevention System
IaaS - Infrastructure-as-a-Service
PaaS - Platform-as-a-Service
SaaS - Software-as-a-Service
Idm - Identity Management
MFA - Multi Factor Authentication
NIPS - Network Intrusion Prevention System
NIST - National Institute of Standards and Technology
NSG - Network Security Group
SCa - Secure Channel add-on
SLA - Service Level Agreement
SSDLC - Secure Software Development Life Cycle
SSO - Single Sign-On
TTP - Trusted Third Party

Chapter 1
Introduction

What is *the cloud*? According to The National Institute of Standards and Technology's (NIST) Definition of Cloud Computing, 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 [1]. There is a fair chance that you are using a Software-as-a-Service (SaaS) offering today. A SaaS is an application running on a cloud infrastructure, accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings [1].

Because of the complexity of these cloud solutions, many attack vectors may exist, and there is a constant need to improve upon the architecture. And this phenomenon is growing. According to Cisco, in the coming 5 years, growth of data centers will increase by almost 100%, global cloud IP traffic will reach 15.1 ZB annually by the end of 2020, up from 3.9 ZB in 2015, and the SaaS delivery model will stand for 74% of the total cloud workloads [2]. Thus, more and more of the communication end up in data centers and at cloud providers. This means that the governance of one's private data is out of the one's hands, creating a need for secure end-to-end communication.

As of today, cloud services are widely used in several different sectors of society. When storing sensitive data in the cloud however, issues arise concerning topics such as privacy, data integrity, availability, who owns the data, and more [3, 4]. The attack surface of clouds is larger than that of regular IT systems and as such, security requires more effort from tenants building on the infrastructure, the developers of cloud services, and the security professionals that assess and standardize cloud security. The most predominant cloud attacks are the Distributed Denial of Service (DDoS) or DoS attacks, and Man-in-the-Cloud attacks [5, 6]. The Man-in-the-Cloud attacks are detailed in [7]. Some of the attacks on popular SaaS offerings are mentioned in [6]. The paper also argues that there are additional attack surfaces in the cloud compared to regular server-to-client interfaces. Other papers such as [8] show that 90% of the identity management (IdM) solutions of some 22 popular SaaS cloud providers can be bypassed allowing attackers to gain

access to private information. In [9], the authors show that information can be extracted across tenant boundaries using side-channel attacks. These papers highlight the fact that cloud security is a considerable concern and moving to the cloud should be an informed decision. The questions *how do you enable secure communication between cloud and device* and *how secure is it?* are the foundation of this thesis. In order to answer them, we assess the general level of cloud security by looking at some of the top tier cloud providers' best-practices regarding security. We summarize our findings in a model that companies and software providers can use when deploying to the cloud to heighten their awareness of attack surfaces and subsequently act and improve their cloud security posture. We also apply the model onto the Axis -host cloud service, a platform- and software-as-a-service, where we were tasked with analyzing the security of their current cloud-device communication and providing a proof-of-concept solution for a new, more secure communication.

1.1 Goals

This thesis project was carried out at the Research- and Development-department of Axis Communications AB. Axis is a manufacturer of network cameras for the physical security and video surveillance industries [10]. Axis' products contain an embedded computer and a custom version of Linux. As such, some restrictions apply when designing a communication solution. Furthermore, some of the findings in this report are specific to this company's products and solutions, while some are generally applicable for all types of SaaS and platform-as-a-service (PaaS) providers. The goals of this project are the following:

1. Analyze current client/server-communication, determine its capabilities, deficits and future needs.
2. Based on the findings, evaluate the Secure Channel add-on, an internally developed embedded client facilitating secure communications;
    - Is the Secure Channel add-on capable of fulfilling the needs of the Axis-hosted cloud service, such as accommodating different types of traffic to and from the internal services of the cloud service?
    - Can the Secure Channel add-on solution be altered or extended in order to accommodate the current and future needs of the cloud service?
3. Implement a prototype or proof-of-concept where the Secure Channel add-on is used in the cloud service environment.
4. Answer how well the new solution with the included Secure Channel add-on compares to the state-of-the-art-solution. Suggest steps to be taken in order to further enhance the security.
5. An evaluation of how future needs of Axis' cloud service can be included in the Secure Channel solution, and a design proposition of how those can be included.

The results of each of these bullet points are discussed in Chapter 5.2. Furthermore, two questions arose: *how do we enable secure communication between cloud and device*; and *how secure is it?* These two questions are at the heart of the study and will be discussed throughout the report.

1.2 Limitations

Cloud security is a complex area consisting of security on several different levels ranging from physical security and politics to cryptography and user management. This thesis will focus on SaaS solutions that utilize the IaaS from a cloud provider, i.e. aspects such as physical hardware security and facilities will not be addressed.

1.3 Previous Work

"Cloud Security - a Comprehensive Guide to Secure Cloud Computing" by Ronald L. Krutz and Russel Dean Vines published in 2010 [11] aims to provide insight into the capabilities, vulnerabilities, advantages and trade-offs of the cloud while also describing methods of gaining the benefits of the cloud computing with minimal risk. The book clarifies issues and concerns about privacy and security that may arise when being introduced to cloud computing, such as geographical dispersion, size and structure. Guidelines on how to maneuver the field of cloud computing are provided in an extensive way.

"Cloud Security and Privacy - An enterprise perspective on Risk and Compliance" by Tim Mather, Subra Kumaraswamy and Shahed Latif [12] tries through a systematic investigation of what constitutes cloud computing, what it offers in terms of security and answer what is wrong with security in cloud computing. Implications of cloud computing security on privacy, auditing and compliance on both the cloud service provider and the customer are also explored in the book. Different perspectives on security for larger organizations versus small to medium-size business are included in the book as well.

White papers from mature cloud service providers such as Amazon Web Services, Microsoft Azure and IBM, organizations as Cloud Security Alliance (CSA) as well as several scientific research papers [3, 4, 5, 6, 7, 8, 9, 13] and the two books mentioned above form the foundation for the theoretic part of this thesis.

Security in the cloud is a major part of this thesis work, and understanding its ins-and-outs is crucial to making sound suggestions for its improvement. In [13], the authors list different security threats depending on the service model delivered. Axis today serves corporate customers with their hosted cloud service which can be described as a SaaS. For this type of service, the authors claim that security threats such as interception, modification of data at rest and in transit and session hijacking are most likely. These are specific vectors that we have taken into consideration when making the proposal and model in Section 4. Furthermore, they suggest implementing a trusted third party (TTP) that can facilitate secure interactions between two parties. The Axis TTP service takes the role of TTP in the Axis-hosted cloud serivce, whose role is to serve certificates and point devices

to the correct cloud. Thus, parallels can be drawn between the authors' suggested optimal solution and Axis' current solution.

1.4 Outline

The remainder of this report is structured as follows:

- *Background* presents useful theory to understand the areas of knowledge this thesis will cover. In this section, cloud computing and the different service models of cloud computing, encryption, authentication and private key infrastructure are explained.
- *Method* presents the scientific method used and the reasoning behind our decisions.
- *Cloud Security - Best Practices* Features a run-down on how to best implement a secure SaaS solution in the cloud with regards to industry best practices and standards. The chapter concludes with a model to easily identify what areas need overview. This is aimed for organizations and people who wish to move software to the cloud.
- *A Case Study of an Axis-hosted cloud service* details the specific work done at Axis with implementing a secure communication solution for the Axis-hosted cloud service;
- *Discussion* covers our thoughts on the general outcome of the thesis, the work done at Axis, and the general direction to go from here.
- *Conclusion* concludes the report with our final thoughts on our work and in general, the security of the cloud.

Chapter 2
Background

This chapter clarifies vital aspects considered and techniques used throughout the thesis. It aims to provide the reader with an overview in order to ease the understanding of said concepts.

2.1 Confidentiality, Integrity, Availability

A Comprehensive Guide to Secure Cloud Computing (2010) [11] refers to the CIA triad of information system security as a vital part of cloud software assurance [11, p. 63]. Confidentiality, integrity and availability are also mentioned as key concerns for a secure cloud in [12, p. 67-71]. These are not cloud-specific concerns, they apply to on-premise IT operations as well. However, in order for the SaaS to be seen as secure and trustworthy by a potential customer, it is fundamental to provide confidentiality, integrity as well as high availability of the data.

According to the ISO27000 standard, confidentiality is the property, that information is not made available or disclosed to unauthorized individuals, entities, or processes [14]. Confidentiality can be achieved either by the use of encryption or by enforcing access control list restrictions and file permissions on sensitive information. The two are of course not mutually exclusive.

Integrity is the property of accuracy and completeness [14], which in terms of data could be considered to be the process of maintaining and assuring the accuracy and completeness of data during its life-cycle. To ensure the integrity of the data put in the cloud, one should perform data integrity checks utilizing e.g. MACs or digital signatures. To minimize the risk of someone accidentally or deliberately modifying or deleting data the scope of users who has write access to the data should be limited.

Availability is defined by [14] as the property of being accessible and usable upon demand by an authorized entity. Having securely stored data is pointless if no one can access it. High availability is therefore a desirable trait.

2.2 Cloud Computing

Cloud computing is “the delivery of computing services, such as servers, storage, databases, networking, and more - over the Internet“ [15]. According to NIST [1], cloud computing has five essential characteristics;

- *On-demand self-service*

  Meaning a subscriber can provision further computational power without the need for human intervention.

- *Broad network access*

  Specifies that the resources be available over the network and accessed through standard mechanisms, such as mobile phones, tablets, laptops, etc.

- *Resource pooling*

  Means that the provider's resources are pooled to serve multiple consumers with different physical and virtual resources dynamically assigned and reassigned according to demand.

- *Rapid elasticity*

  States that capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward relative to demand. To the consumer, resources often appear to be unlimited and readily available for use.

- *Measured service*

  Resource usage (such as bandwidth, storage, processing or user accounts) in the cloud is metered, monitored and controlled. This provides transparency for both the provider and consumer of the service in question.

2.2.1 Different Service Models in Cloud Computing

Cloud computing can be realized on different levels based on the product being served [1].

- *Software-as-a-Service (SaaS)*

  The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage or even individual application capabilities, with the possible exception of a limited user-specific application configuration settings.

- *Platform-as-a-Service (PaaS)*

  The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying infrastructure but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

- *Infrastructure-as-a-Service (IaaS)*

  The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer

> is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications and possibly limited control of select networking components (e.g. host firewalls).

Cloud Security Alliance (CSA) is the world's leading organization that seeks to help ensure a secure cloud computing environment by defining and raising awareness of best practices [16]. Their reference model for different cloud service models is shown in Figure 2.1. The figure depicts what is included in each of the service models, SaaS delivering a complete software running on a specific virtual machine. IaaS, in contrast, only delivers the bottom 60%. The thesis will mainly consider the SaaS and IaaS model since the Axis-hosted cloud service is a SaaS built on top of an IaaS.

2.3 Trusted Third Party

A TTP within the field of cryptography is considered to be able to form trust between two parties by facilitating secure interactions between them, under the premise that they both trust the trusted third party. This entity provides end-to-end security services based on standards and is e.g. useful across different domains. The establishment of a trust relationship between two parties can be thought of as the result of a series of specific acceptances, techniques and mechanisms reviewed by the trusted third party [13]. The loss of the traditional security boundary that results when moving to the cloud can be addressed by the use of a Trusted Third Party [13]. The trusted third party is crucial for establishing the connection between the device and the cloud in our proof-of-concept solution.

2.4 Encryption

Encryption is a fundamental part of providing confidentiality of data. By means of an encryption algorithm or cipher, plain text or clear text can be enciphered using a cryptographic key. A decryption key deciphers the message back to plain text.

2.4.1 Symmetric Encryption

In symmetric encryption, the same key is used to encrypt and decrypt messages. In order to prevent ineligible parties from recovering the key, it has to be distributed in a secure way to the involved parties. Symmetric encryption is commonly used when bulk data needs to be encrypted over an already established connection [17]. This is due to the fact that symmetric encryption is faster than asymmetric encryption.

2.4.2 Asymmetric Encryption

Asymmetric encryption uses a key pair, where one key is secret and one is non-secret, often referred as private and public key, where the private key is used to

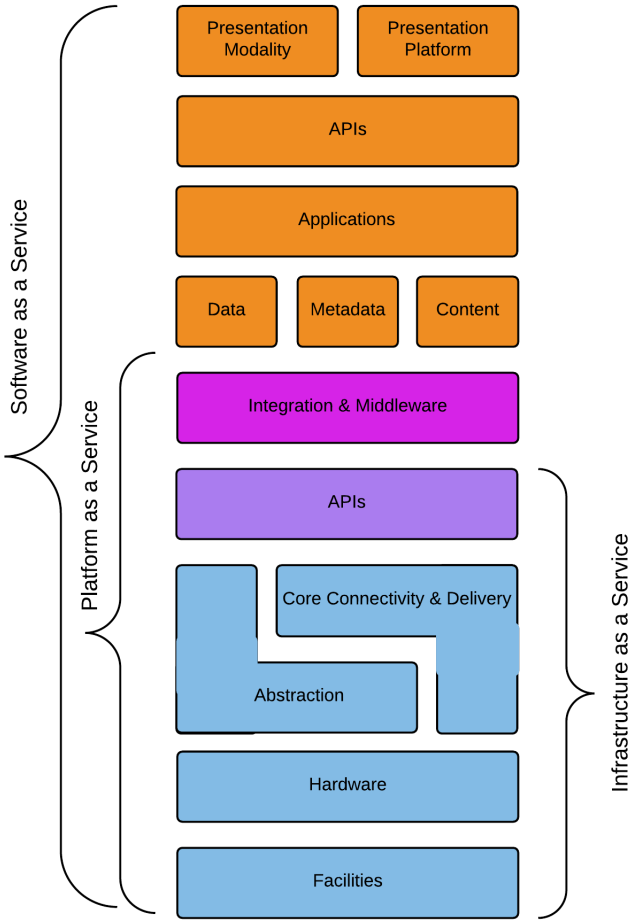

Figure 2.1: CSA Reference Model (Image credit: Cloud Security Alliance (Security Guidance for Critical Areas of Focus in Cloud Computing v.2.1).

encrypt and the public key is used to decrypt. Compared to symmetric encryption the asymmetric encryption is slower, however an advantage of asymmetric encryption is that two parties do not need to share a common key for encryption. Since the encryption key is public, the problem with distributing keys securely is not present in asymmetric encryption, instead there is the problem of securely distributing e.g. certificates. TLS uses asymmetric cryptography to exchange symmetric keys needed for the duration of the communication session [17].

2.5 Public Key Infrastructure

As mentioned in Section 2.4.2 a key pair can be generated by anyone in order to encrypt and decrypt messages and prove their ownership of the private key, however, this is not sufficient to prove someones identity. The client and the cloud needs to consult a trusted third party in order to be able to prove their ownership of their key pair. The trusted third party is more commonly known as a Certificate Authority (CA). A certificate that is signed by a CA can be used to prove ones ownership of a key pair.

Since it is not practical to have one CA signing all certificates for a large system, a tree structure of intermediate CAs, originating from one CA-root can be created. Intermediate CAs are able to sign certificates on behalf of the CA-root and in that way a chain of trust is realized with the CA-root at the top, intermediate CAs in between and the issued certificate at the bottom. This chain makes it possible to trust a certificate at the bottom of the chain as long as you trust the CA-root [18]. This tree structure is known as a part of a public key infrastructure (PKI).

2.6 Authentication

In order for the device and the cloud to be sure of who they are communicating with, they can authenticate themselves and verify each others identities. Usually when a device connects to a server, it is only the device who verifies the identity of the server. TLS does have support for authenticating the client, however it is rarely used due to the fact that it requires provisioning of certificates to clients. In our case, mutual authentication is critical.

Chapter 3
Method

This chapter describes the work process and the reasoning and motivation behind specific actions taken. Initially, we needed to stipulate and clarify what exactly was desired from Axis point of view. From the initial master thesis proposal, a vague description of the expected work was outlined; "the current server solution communicates with Axis devices via a secure command channel. The device can also communicate with the cloud but via another connection. Axis wishes to determine whether the communication from both device and cloud can fit in a two-way, secure connection, escaping the need for the device to initiate a new and possibly insecure connection every time it wants to communicate".

The *initial* proposal stated the following goals:

1. Analyze current client(device)/server-communication, determine if this solution can be adapted to send API-calls to the server software in the cloud.
2. Alternatively, find another solution to securely communicate from camera to cloud service.
3. Implement a prototype or proof-of-concept for the solution if possible.

Startup meetings were had with the developers and our supervisors to clarify and hash out the finer details regarding the goals. We started to develop an understanding of the current client/server communication by reading source code, having meetings with the architects of the client/server solution, and reading the reference manuals. Our understanding of the current communication is shown in Figure 5.3. At a meeting, it was discovered that a separate team who had previously shown interest in researching how secure two-way cloud-device communication could be realized had already developed a prototype. This was named the "Secure Channel add-on (SCa)". The communication principles of the SCa is shown in Figure 5.6.

At this point it was decided, rather than reinventing the wheel, to aim for a new set of goals and tasks to perform. In meeting with our supervisors, it was decided to analyze this newly developed solution in order to find eventual synergies and to study the possibility of using it in the Axis-hosted cloud service. We came up with the improved list of goals shown in Section 1.1.

The first point on the list was a three-part question:

- Analyze current client/server-communication, determine its capabilities, deficits and future needs.

The rationale behind this question was that the answers would gain insight into what would be expected and demanded in a new solution. The answers we came up with were the product of practical hands-on testing of the client-cloud communication using e.g. CURL and other command line tools to get a feel for how the communication worked, as well as consulting the developers and reference manuals present. A list of items that were considered to be defining factors are presented in Section 5.1.

When implementing an improvement, the first step must always be to measure the existing solution and obtain data or measurements. This will be the baseline upon which changes can be made and evaluated. The current capabilities of a system would need to be present in a new solution, and the deficits would need to be solved for it to be a worthy successor. Axis already had an idea what the shortcomings of the current communication solution were, this list is shown in Section 5.1. The future needs would at least have to have been reflected upon and planned for, to ease their eventual implementation. When analyzing the current system, the complexity was quickly noticed, creating a need for a top-down image showing the different parts in the system and how they would need to be modified. This helped visualize where modifications had to be made and what parts needed to communicate with each other in order for the proposed communication solution to work. This is shown in Figure 5.5.

The second item relates to the Secure Channel add-on:

- Based on our findings, evaluate the Secure Channel add-on;
  - Is the add-on capable of doing what the existing solution does, such as accommodate different types of traffic to and from the internal services of the cloud solution?
  - Can the Secure Channel add-on be altered or extended in order to accommodate the current and future needs of the cloud solution?

It was of great interest to Axis to evaluate the Secure Channel add-on, because of the potential benefits it offered. The results of our evaluation and whether the add-on is capable of accommodating the current cloud service traffic as well as its future needs is described in Chapter 5.

We communicated with the authors of the Secure Channel add-on and then started developing the proof-of-concept solution that was based on the Secure Channel. We chose to implement the server in JavaScript (Node.js) because of the strong prototyping properties of the language, allowing us to produce a minimum working example quickly and subsequently improving it with short development iterations. Node.js is also a language that excels when there is a need for clients to push data and keep long duration connections (see e.g. [19, 20]), something that is desired for Axis cloud service. The add-on (the client) was written in C++11 by its original authors, so the choice had already been made in that regard.

There is a strong emphasis on client-server security in this thesis and as such, we dove into the core part of the communication first - to get a persistent channel working using the WebSocket protocol. (Secure) WebSockets is a communication protocol that enables full-duplex, real-time data transfer between client and server. It requires certificates to enable the mutual authentication and as such, SSL/TLS

certificates needed to be in place (see Section 2.5 for details). The fact that the Secure Channel add-on that we used still was under development proved to be a challenge, because integrating changes from two development branches became somewhat complex. This step of the work corresponds to the item

- Produce a proof-of-concept where the Secure Channel solution is used in the cloud service environment.

The work was split up so that one of us was working on the back-end and the other was working on the add-on. The reason for the split was that we could each focus on one area since the roles were quite varying, and the programming languages used differed. The time would be best spent if we were specialized on one thing, minimizing the need for context switching. A great deal of effort was spent in learning how to build add-ons for Axis devices, setting up a build environment, configuring the camera and setting up a cloud service instance. The details of this process are specific for Axis, and as such will not be reported on. Only what we consider general takeaways and important points are presented in this thesis.

The two questions mentioned in Chapter 1: *how do you enable secure communication between cloud and device* and *how secure is it?* can be answered preliminarily by looking at state-of-the-art cloud providers. However, there is no static, cover-all security solution that works for all software. The security must be tailored depending on the type of software, its clients, the usage patterns, the criticality of data and more. This can be a daunting task. SaaS providers are difficult to measure against a baseline since there is no industry standard for how to determine software security [12, p. 55]. We chose to look at three of the four top IaaS providers based on their respective market shares [21].

The ones chosen were Amazon AWS, Microsoft Azure and IBM. We chose not to analyze Google Cloud Platform due to time restraints. We chose to look at primarily IaaS solutions because it is the cloud model Axis uses to deploy software to the cloud. An additional reason is the interesting balance of responsibilities between IaaS provider and customer that arise. We also chose to collect and summarize data from non-profit organizations such as CSA, which aims to define and raise awareness of best practices to help ensure a secure cloud computing environment [16], and NIST who defined cloud computing in its well-cited work [1]. In 2012 they released a 81-page document on recommendations for cloud computing in [22]. These sources are used because of their high trustworthiness and to balance out the views of for-profit organizations.

An analysis of the best-practices of the above companies is provided in Chapter 4. It concludes with a model of critical areas of security for organizations that wish to move their workloads into the cloud. We derived the areas in Chapter 4 from a combination of the best practices of the providers and the suggested best practices of [11, 12]. At first sight, many of the areas overlap, e.g. user permissions are included in several areas other than 4.3. In addition, we felt the need to include an area that most providers are not too keen to promote; Portability. How easy it is to get your application up and running at another provider is of interest for the SaaS vendor (such as Axis in this case), but from a providers point of view, portability is not a priority. Portability is highlighted by CSA as an important part of security assurance [23]. CSA, being a non-profit organization, might be

more inclined to promote portability since, in contrast to cloud providers, they do not seek to "lock in" customers by making them dependent on their services. In fact, they do not have any customers at all.

The results of this thesis are thus split in two; first, the general model for critical areas of security is presented in the forthcoming chapter. This is applied to the case at Axis (in Chapter 4), where we through talks with the architects and the developers responsible for managing Cloud Ops, as well as studying source code and other information sources got an understanding of what the system lacked. We believe the results from applying the model on the Axis-hosted cloud service are very accurate, though we reserve that Axis may want to keep any severe security holes or flaws from us. The second result is the work done at Axis trying to improve the current communication, which is presented as a list of concerns that were remedied, as well as a list of concerns we did not succeed in remedying, either because of time constraints or of the fact that they did not add to the proof-of-concept. This is presented in Chapter 5.

Chapter 4

Cloud Security - Best Practices

This chapter highlights primarily security best practices in top-end cloud providers. The best-practices are collected and summarized from the three vendors mentioned in Chapter 3, as well as CSA [16], and the books [11] and [12]. We also refer to NIST [22]. The chapter will be concluded with a summary and a model that will help SaaS vendors establish a State-of-the-Art security posture.

Generally, moving to the cloud is a good investment if no prior hardware is owned that can compete with the features of cloud computing. According to [11, p. 141], smaller companies often see a larger security return by using public cloud infrastructure than larger, better-funded organizations with larger infrastructure. The reason for this is that large companies often have prior investments into IT infrastructure and its accompanying security architecture, whereas the smaller companies can more readily employ the security benefits offered by a public cloud service provider (CSP). Below, we cover some of the technical details and important areas to assess and secure in order to improve the security posture.

4.1 Service Agreements and Service Level Agreements

Every organization that enters into a deal with a cloud service provider signs a service agreement, a legally binding agreement between two parties. It usually consist of (1) a collection of promises made to consumers, (2) a collection of promises explicitly not made to consumers (limitations), and (3) a set of obligations that consumers must accept [22]. Generally, they cover things such as availability, remedies for failure to perform, data preservation and other topics. The recommendations from NIST [22, p. 61-62] are to:

- Pay attention to terminology - Common terms may be redefined by a cloud provider in ways that are specific to that provider's offerings.
- Remedies - Unless a specific service agreement has been negotiated with a provider, remedies are likely to be extremely limited; consumers may wish to negotiate remedies that are commensurate with the damage that might be sustained.
- Compliance - Consumers should carefully assess whether the service agreement specifies compliance with appropriate laws and regulations governing consumer data.

- Security, Criticality and Backup - Consumers should carefully examine the service agreement for any disclaimers relating to security or critical processing, and should also search for any comment on whether the provider recommends independent backup of data stored in their cloud.
- Negotiated Service Agreement - If the default service agreement does not address all the consumer needs, they should be discussed with the provider prior to use.
- Service Agreement Changes - NIST recommends developing a plan to migrate workloads to alternate cloud providers or back on-premise in the event of a change of service terms that is unacceptable to the customer.

Service Level Agreements (SLAs) are shorter documents stating the technical performance promises made by a provider including remedies for performance failures. The Service Level is the level of availability (uptime) a CSP offers, and any events that cause downtime are subject for recompense on the customers part. These are paid out by the cloud service provider using service credits. The credits are used to pay for their services, which means the compensation is effectively a rebate. We compared the three providers' service level agreements with a focus on recompense when the service is down. They offer slightly varying recompense as shown in Table 4.1(Sources [24, 25, 26]).

Table 4.1: SLA recompense for different cloud providers.

	Uptime	**Service Credits**
Microsoft	<99.95%	10%
	<99%	25%
	<95%	100%
IBM	<99.95%	10%
	<99.90%	25%
Amazon	<99.95%	10%
	<99.0%	30%

In [22], the authors recommend that the organization understands the terms of the service agreements that define the legal relationships between cloud customers and cloud providers, and what responsibilities are the customer's and those of the service provider. CSA does not mention any specific recommendations tied to SAs and SLAs in their report [23].

4.2 Shared Responsibility

Security controls in the cloud are in essence no different from security controls in a regular on-premise IT system, albeit with the added complexity of secure

multi-tenancy. However, depending on the service model, the security *responsibilities* differ greatly, for both the provider and the consumer. Using Google Docs (which is a free, public cloud SaaS) places a very low security responsibility on the consumer since all of the security is handled by the provider (the provider being a SaaS vendor). Amazon's AWS EC2 IaaS offering includes vendor responsibility for security up to the hypervisor, meaning they address security controls such as physical security, environmental security, and virtualization security. The consumer (a SaaS vendor), in turn, is responsible for security controls that relate to the IT system including the operating system, applications, and data [27].

This means that building a SaaS system on top of an IaaS, such as Amazon EC2, comes with a lot of benefits but also responsibilities in the form of shared security concerns. Specifically, in the case of IaaS, Amazon [28] state that they manage security for:

- Facilities.
- Physical security of hardware.
- Network infrastructure.
- Virtualization infrastructure.

The customer (in this case the SaaS vendor) is responsible for the security of the following assets:

- Machine images. The configuration of the underlying machine image, that is used to create the virtual machine within EC2. There are many images to chose from, so making sure the correctly hardened and secure image is chosen is important to making sure no critical security holes are present.
- Operating Systems. It is the customer's responsibility to make sure the operating system is secure. Keeping it up to date, disabling insecure applications, minimizing exposure, protecting credentials and data are recommendations from Amazon.
- Applications. Protecting applications is done in the same way as the operating systems. Keeping critical applications updated, and not using applications with security holes. Only run trusted software. Use antivirus.
- Data in transit. Encrypt data in transit using e.g. IPSec ESP and/or SSL/TLS. Authenticate data integrity using IPSec ESP and/or SSL/TLS. Use X.509 certificates to authenticate the remote end. Use HTTPS when sending internet traffic. Use server certificate authentication.
- Data at rest. Data at rest is generally stored on other Amazon services such as S3 or EBS. Keeping data encrypted ensures severity of data theft is limited. Backing up data ensures there is a possibility to restore. Designating who can use/access/delete what helps mitigate accidental data loss.
- Data stores. This is simply data at rest. Keeping databases and other data stores secure through means of encryption to prevent unauthorized leakage of information.

- Credentials. Create users in AWS IAM (identity and access management) and centrally manage user, security credentials, access keys and permissions policies that control which AWS services and resources users can access. Do not use AWS credentials on a day-to-day basis. Users should have least-privileges for their roles. Limit access to AWS account(s).

Table 4.2 illustrates the shared responsibility model for security in AWS' case.

Managed by AWS Customers	**Managed by AWS**
Customer IAM	AWS IAM
Customer Data	Foundation Services
Platform Management	AWS Global Infrastructure
Application Management	
Client-Side Data Encryption	
Data Integrity Authentication	
Server-Side Encryption	
Network Traffic Protection	

Table 4.2: Table showing areas of responsibility in AWS Infrastructure

Responsibility	**On-Prem**	**IaaS**	**PaaS**	**SaaS**
Data Classification & Accountability	Customer	Customer	Customer	Customer
Client & End-Point Protection	Customer	Customer	Customer	Shared
Identity & Access Management	Customer	Customer	Shared	Shared
Application Level Controls	Customer	Customer	Shared	Cloud P
Network Controls	Customer	Shared	Cloud P	Cloud P
Host Infrastructure	Customer	Shared	Cloud P	Cloud P
Physical Security	Customer	Cloud P	Cloud P	Cloud P

Table 4.3: Microsoft Azure areas of shared responsibility.

A similar picture to the one in Table 4.2 is shown in Table 4.3. The Microsoft Azure shared responsibilities show that network and host infrastructure are shared responsibilities and that the following items are the responsibility of the consumer:

- Data Classification & Accountability. It is important to distinguish sensitive user data from publicly accessible information. Using classification and categorization, a company can take action by securing classified data relative to its eventual business impact and the risk associated with it [29].

- Client & End-Point Protection. The general security concerns regarding e.g. data in transit and at rest, protecting the clients' data, and hardening end-points to resist attacks.
- Identity and Access Management. Handling identity and access management using different techniques for different areas. Multi-factor authentication for access to sensitive and/or disruptive information, role-based access and letting users access only what they need to lower the risk of unintentional misuse of privileges, such as deleting data or accessing classified information. It also lowers the extent of the damage if a user account becomes compromised. User handling is very important in traditional computing as well. Additional considerations include logging and auditing user actions in order to trace and improve if a user error or attack has been detected.
- Application Level Controls. In an IaaS, the customer is responsible for protecting and securing the OS and the application layers of the VMs. This is similar to an on-premise system that needs regular updates, hardening and configuration to withstand attacks and keep data secure.
- Network Controls. According to Azure, the consumer shares responsibility with the CSP to deploy, manage, secure and configure the networking solutions. Practically, this entails the security of network elements such as virtual networking, load balancing, DNS and gateways. These controls are necessary for services to communicate and interoperate safely.
- Host infrastructure. A shared responsibility to ensure the service is optimally configured and secured. This includes configuration of permissions and network access control to enable network communication correctly.

Area	**SaaS**	**PaaS**	**IaaS**
Application & Data	X		
Runtime, Middleware, OS	X	X	
Compute, Storage, Network	X	X	X

Table 4.4: Depiction of areas that IBM are responsible for in IBM cloud services.

For IBM infrastructure, we could not find extensive information, however the different levels are shown in Table 4.4 [30]. The table shows that for IaaS, the customer responsibility is that of the Application and Data, but also the Runtime, Middleware and OS (X marks an IBM managed area). We assume that "shared responsibility" for the areas that are not marked with an X also extend to responsibility for their security. Securing the application data is touched upon later in this chapter, and data security, again, relates to securing data in transit and at rest. Just like the other models, the responsibility is on the customer to secure

OS, Middleware and Runtime, i.e the VM+OS layers. This has similarities with information found in Tables 4.3 and 4.2. At this point, it is easy to argue that these companies claim similar responsibilities for the IaaS model. NIST has the same idea and this is depicted in Figure 4.1. The CSA reference model is shown in Figure 2.1.

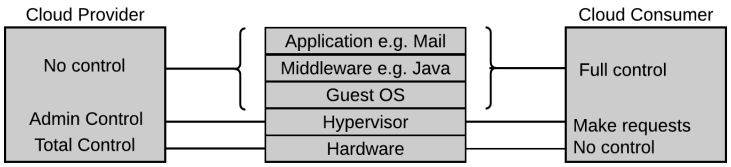

Figure 4.1: NIST IaaS Component Stack and Scope of Control.

Summary of Customer Responsibilities

Summarizing the major points from these three top tier cloud service providers, NIST and CSA yield the following list of critical areas of security concern:

- Identity and Access Management. Users should be granted access based on least privilege. Separate admin and user accounts, use rigid authentication, limit and constrain access. Relates to the authentication part of the CIA triad. Often relates to authenticating users to the compute instance being paid for, but in large also corresponds to the users of the system being served on top of the IaaS.
- Data at rest and in transit. This encompasses encryption of communication between end-point and client, secure storage of data using e.g encryption of disks, secure deletion of data. Relates to confidentiality in the CIA triad.

These are the two items that generally are very deeply connected to the end-user of the SaaS being built (remember the Google Docs user should not be concerned with encrypted traffic to and from the server where the user's private data is being stored). Thus, there is the added responsibility, as a customer of the IaaS provider, to protect the clients' privacy and data through the means listed above. Additionally, the following two items are the customer's responsibility:

- Application security. The security of the provided application. If it is accessed via web browsers, additional (traditional) web security measures apply, such as JavaScript weaknesses, database injections, cross-site scripting and similar attacks.
- Network security. Firewalls, Intrusion detection/prevention Systems (IDS/IPS), Network Access Control Lists (NACLs), DNS, use of De-Militarized Zones (DMZs) and other techniques. Keeping specific ports closed and more.

The points that have not been covered are: *Machine Images* and *OS* (AWS), *Host Infrastructure* and *Network Controls* (Azure) and *Middleware*, *OS* and *Runtime* (IBM). These are the customer's responsibilities. We call these 'OS and VM Level Application Security'. The best practices regarding this is presented in Section 4.5.1.

The responsibilities listed above provide the basis for the rest of the analysis in this chapter. The situation is that of deploying a SaaS on an IaaS, meaning the customer is a SaaS vendor. Below we list, for each of the customer responsibilities, a short summary of the best-practices regarding the responsibility area. Additionally 'Portability' will be discussed. Finally, a summary section will list the general best practices for each of these areas.

4.3 Identity and Access Management

It is important to distinguish sensitive user data from publicly accessible information. Using classification and categorization, a company can take action by securing classified data relative to its possible business impact and the risk associated with it [29]. In order to be able to hold an individual accountable for e.g. carrying out an unauthorized action, the system needs to be able to determine actions and behaviors of individuals in the system. This capability is often referred to as accountability [11].

Ensuring that users have the correct permissions to perform their tasks and access the resources they need, but not anything else (called least privilege), is an important part of the security management system. You can also set permissions for groups of users, then add users to that group. This is easier than giving each user of the system individual permissions. The best practices of the three providers when it comes to identity & access management are listed below.

The best practices of AWS are [31]:

- Lock away root access keys
- Create individual accounts
- Remove unnecessary credentials
- Rotate credentials regularly
- Delegate by using roles instead of by sharing credentials
- Configure a strong password policy for your users
- Enable multi-factor authentication (MFA) for privileged users
- Grant least privilege
- Use groups to assign permissions to users

The Azure best practices [32]:

- Centralize your identity management

- Enable Single Sign-On
- Deploy password management
- Enforce multi-factor authentication (MFA) for users
- Use role-based access control
- Control locations where resources are created using resource manager
- Guide developers to leverage identity capabilities for SaaS apps
- Actively monitor for suspicious activities

IBM recommends the following best-practices regarding users and credentials [33, 34]:

- Do not share master-login credentials
- Use role-based or user-specific accounts
- Manage your passwords responsibly
- Single sign-on
- Multi-factor authentication
- Reporting
- De-provisioning access

To summarize, the common items are; (1) use single sign-on, (2) multi-factor authentication, (3) use groups to assign permissions to users, alternatively use role-based accounts, (4) de-provision access (remove unnecessary credentials, deploy password management), (5) reporting or monitoring for suspicious activities (not AWS). Additionally, locking away root access keys, rotating credentials regularly, not sharing master-login credentials are items of highest importance. We refer to these as 'create a secure credential management routine'. Only AWS states to use individual accounts, which is a strong method to detect breaches if coupled with logging.

Single Sign-On refers to a single point of authorization (such as an OAuth or SAML service) which generates a token for the user. The token describes the access level of the user. The main benefit of this is a centralized user authorization service, removing the need for the user to remember several accounts and passwords. AWS did not list single sign-on as a best-practice. According to [27], one of the security challenges for SaaS vendors is supporting identity management and sign-on services. There are a few models to implement this, such as;

- using an Independent IdM stack, where the SaaS provider handles all user accounts, passwords, etc;
- Credential Synchronization, where the SaaS vendor replicates user account information and credentials between enterprise and SaaS application;
- Federated IdM, where the user account information including credentials is managed and stored independently by each tenant. (An example of this is using a Google Account or Microsoft Account to authenticate oneself).

For each of these models, different advantages, disadvantages and challenges appear.

Multi-Factor Authentication usually combines something *you know* with something *you have*, making it harder for an attacker to gain access if they retrieve one of the two factors. This is recommended by all the top service providers, and as such is considered extremely important. This is commonly used by banks to provide access to accounts and elevating access from user to privileged user. Thus, it can be combined with pure password authentication to keep adhering to least-privilege rules.

Group or Role-Based Access help in limiting users' access levels and removes the need for sharing credentials. This also makes sure the right people have access to only the things they need, which compartmentalizes any damage that can be done. Group or role-based access also facilitates granting and revoking user access.

De-Provisioning Access or Removing Unnecessary Credentials is a best-practice that helps tighten the security of the system. If unused credentials are removed, one less possible security hole exists.

Reporting and Monitoring User Access can help with tracking both malicious and erroneous behavior, such as mistakingly removing or changing resources. It is closely tied to accountability, which is the property of proving responsibility for an action. Reviewing logs can be a great tool during audits and analysis of security breaches or cases of misuse of the system.

Creating a Secure Credential Management Routine refers to handling credentials sanely. Keeping them locked away using specific security mechanisms to ensure their safety. This item also encompasses a password policy for the users, such as requiring a specific length of the password, and a change of password after a specific duration of time.

If administrator accounts are compromised with unlimited access to the infrastructure, massive amounts of damage can be inflicted. This also ties back to using least-privilege accounts for all actions taken. NIST recommends when renting computing resources from an IaaS cloud provider in the form of virtual machines or physical servers that a limited set of trained/trusted users (from the consumer organization) alone are provided administrative access to those resources [22]. Apart from that, not many insights regarding identity and access management can be found in NISTs report.

4.4 Handling of Data

In a traditional on-premise application deployment model, the sensitive data of each enterprise continues to reside within the enterprise boundary and is subject to its physical, logical and personnel security and access control policies. However, in the SaaS model, the enterprise data is stored outside the enterprise boundary, at the SaaS vendor end. Consequently, the SaaS vendor must adopt additional security checks to ensure data security and prevent breaches due to security vulnerabilities in the application or through malicious employees. This involves the use of strong encryption techniques for data security and fine-grained authorization to

control access to data [27].

With IaaS, you can use similar storage and data transport systems that you use in the traditional enterprise model, but in a virtual form. As the customer has control of everything on the virtual machine they can, and indeed should, implement data storage encryption and data transport encryption [35].

4.4.1 Multi-Tenancy in the Underlying Architecture

Different technologies are used to ensure multi-tenancy, and different types of hypervisor techniques exist. According to [35] it is part of the customers due diligence to "investigate what systems the cloud vendor has in place to isolate different customers' data and systems. An important aspect of this assessment is an evaluation of how the network traffic of each tenants systems are isolated from the other tenants in the system." NIST [22] recommends to ensure that the provider has mechanisms in place to protect VMs from attacks

- From other VMs on the same physical host.
- From the physical host.
- From network originated attacks.

Typical attack detection and prevention mechanisms include Virtual Firewalls, Virtual IDS/IPS, and Virtual Private Networks.

You should note that there is almost nothing you can do to improve the hypervisor and host system security in the IaaS offering as it is out of your control. However, choosing the infrastructure offering with the best multi-tenant architecture and separation of host and guest traffic is paramount to improving the security posture. When the decision is made, the focus of the organization should be to prevent *network originated attacks* as this is within their control.

4.4.2 Data in Transit and Data at Rest

The line between what is the customers and what is the cloud service providers responsibility is often blurred when it comes to data protection. According to the top cloud service providers, it is highly important to protect data in transit.

AWS recommends encrypting data in transit by using IPSec and/or SSL/TLS in order to avoid accidental information disclosure as well as preventing data integrity compromise. To prevent MITM attacks, ID spoofing and peer identity compromise they recommend using server certificate authentication based on the server common name [28, p. 35]. Azure encourages you to always use SSL/TLS when exchanging data across different locations and if needed, isolate the entire communication channel by using a virtual private network [36]. Azure claims that failure to do so results in being more vulnerable to MITM attacks, eavesdropping and session hijacking [36]. The best practices of IBM worryingly does not mention SSL/TLS at all, however, their cloud developer site which is slightly more technically oriented urges the importance of HTTPS, SSL and TLS when moving data from one place to another [37].

According to AWS, permissions and volume or application-level encryption are viable strategies to mitigate accidental information disclosure of data at rest.

Permissions also help prevent accidental deletion and data integrity compromise according to AWS. To further mitigate data integrity compromise they recommend data integrity checks such as MAC/HMAC and regular backups of data. Backing up your data also reduces the impact of an accidental deletion as well as giving you the ability to restore lost data in case of a system failure or a natural disaster [28]. IBM highly recommends establishing role-based permissions to reduce the risk of accidental deletion of data. Performing backups of your data is also encouraged [33]. In their developer guide, they stress the importance of protecting your data from outside tampering by using encryption and hashing for data integrity [37]. Azure states that encrypting the database and using specific roles for reading and changing data in the database should be established [36]. Furthermore, using data encryption at rest is a mandatory step towards data sovereignty i.e. owning the data, compliance and data privacy. This can massively help organizations and users to gain trust in that their data is "theirs".

Secure deletion of data is also a hot topic when the physical storage space is out of one's reach. A possible risk is that data left behind by previous tenants can be recovered by new tenants. NIST states that a consumer should "require that a cloud provider offers a mechanism for reliably deleting data on a consumer's request" [22]. CSA [23, p. 159] states that zeroing or encrypting disks/memory are solutions to this problem, and AWS [28, p. 34] also supports this stating "[to] securely decommission data, you can implement data encryption at rest using customer-managed keys, which are not stored in the cloud. Then in addition to following the previous process, you would delete the key used to protect the decommissioned data, making it irrecoverable."

4.5 Application Security

The Open Web Application Security Project Top 10 (OWASP Top 10) Project summarizes, based on input from security experts from various information systems, a list of the top ten web application security vulnerabilities. The most recent list is from 2013, a new one was initially expected in 2016 but will likely be released in 2017 [38]. The list from 2013 comprises of the following vulnerabilities [39]:

- Injection. Injection flaws, such as SQL, OS and LDAP injection when untrusted data is sent to an interpreter as part of a command or query.
- Broken Authentication and Session Management. Incorrectly implemented application functions related to authentication and session management.
- Cross-Site Scripting(XSS). Untrusted data is sent from the application to a web browser without proper validation or escaping.
- Insecure Direct Object References. References to internal implementation objects such as directories, files or database keys are exposed by a developer.
- Security Misconfiguration. Outdated software, not having defined and deployed secure configurations for the application, framework, application server, web server, database server and platform.

- Sensitive Data Exposure. Web applications do not properly protect sensitive data such as credit cards and authentication credentials e.g. by using encryption.

- Missing Function Level Access Control. Web applications only verify function level access rights prior to enabling functionality in the UI. The same access control should be done on the server when functions are accessed.

- Cross-Site Request Forgery. An attack that causes a user's browser to perform an unwanted action on a trusted site for which the user is authenticated to. This is often achieved through an email, malicious website or a program.

- Using Components with Known Vulnerabilities. Most libraries, frameworks and software modules usually run with full privileges. If a vulnerable component is exploited an attack can facilitate serious data loss or server take over. Applications using components with known vulnerabilities undermine application defenses and enable a range of possible attacks and impacts.

Azure recommends using the OWASP Top 10 as a starting point for guidance on how to secure your application. They also suggest including security aspects early on in the development and penetration testing are good practices [40].

IBM are restrictive when it comes to best practices for application security. They do however strongly promote their own service "IBM Security AppScan", claiming that it is "a leading application testing suite". According to their official developer program, "AppScan provides full coverage of the OWASP Top 10 for 2013". They also claim that OWASP Top 10 is regarded as industry best practice in preventing web application vulnerabilities [41]. In addition to the OWASP Top 10, they suggest that using web application scanning tools, like their own, is fundamental for maintaining a secure web application [41].

AWS recommends being aware of the OWASP Top 10 and to build applications accordingly [42]. Their own tool, Amazon Inspector is said to identify security vulnerabilities and deviations from security best practices in applications. It does not explicitly say it covers OWASP Top 10, but it would be a reasonable assumption that it does.

The OWASP Top 10 is referred to as a minimum standard for web application security by [11]. The OWASP Top 10 is also referred to in [12] and they put emphasis on knowledge of well-known vulnerabilities as a key to develop a secure application as well as embedding security into the software development life cycle [12]. CSA also strongly recommends using a secure software development life cycle (SSDLC) when developing or migrating applications to the cloud [23, p. 103].

Generally, all service providers offer extra security controls that are marketed as premium services. This is a decision for the customer to make, demanding an analysis of possible security risks and how much they are willing to pay to have them fixed. Keep in mind that any abuse of the IaaS instance may lead to the account being terminated. This means that not only can a potential security breach cause damage to the tenant and the customers but may result in a termination of contract.

4.5.1 OS and VM Level Application Security

Additional security concerns appear when moving to the cloud, and in the case of IaaS offerings, keeping the underlying virtual infrastructure secure (i.e. machine images, OS, host infrastructure), configured correctly and updated is crucial to the security of the whole platform. The Azure [43] best-practices are:

- Harden the VM.
- Install and manage Anti-Malware.
- Install the latest security updates.
- Deploy and test a backup solution.

Hardening the VM is done by limiting access and only exposing endpoints that are necessary for the function of the service, as well as keeping it configured and updated. Additionally, there are pre-hardened images for most popular CSPs provided by e.g. CIS [44]. Backing up data seems to be a best-practice regardless of the topic, and security can be furthered by using a backup solution separate from the cloud you are hosted on.

AWS [28, p. 50-64] has the following list of best-practices:

- Use security zoning and network segmentation.
- Strengthen network security.
- Test security.
- Protect against DDoS.

By means of security zoning, you impose sets of security controls onto segments of the infrastructure. This can help separate high-risk assets, access, logging and other functionality between zones. Strengthening network security is handled in Section 4.6. Testing security is done using white- or black-box testing, issuing penetration testing by in-house or 3rd party testers, and vulnerability assessments. Protecting against DDoS attacks is partly done by AWS, they claim. However, end-customer traffic and data are the responsibility of the tenant, and this data is not DDoS protected by AWS.

Unfortunately, no specific information about OS and VM level security was found for IBMs offerings. CSA states [23, p. 157-160] a quite extensive list of recommendations for the virtualization topic, a select few are:

- Separate production environments from test/development and highly sensitive data.
- Secure each virtualized OS by using hardening software in each guest instance.
- Ensure that secure by default configurations follow or exceed industry baselines.
- Make sure that the security vulnerability assessment tools or services cover the virtualization technologies used.

- Consider performance when testing and installing virtual machine security tools, as performance varies widely.
- Patching virtual machine images at rest or protect them until they can be patched.

Additionally, one of the requirements are:

- Virtualized operating systems must include a firewall (inbound/outbound), Host Intrusion Prevention System (HIPS), Network Intrusion Prevention System (NIPS), web application protection, antivirus, file integrity monitoring, and log monitoring, etc. Security countermeasures can be delivered via software in each guest virtual instance or by using an inline virtual machine combined with hypervisor-based API's.

Comparing these recommendations, we can see that security zoning or separating production environments are common for both AWS and CSA. Azure states that hardening the system is a best-practice, which almost correlates with security zoning in that it limits access to disparate functions (i.e production vs test code/setups, internal and external traffic). Installing and managing anti-malware software (Azure) is a practical example of strengthening the network security that AWS recommends, likewise, CSA states that a requirement is that the virtualized OS has antivirus software. Overall, Azure's recommendations are practical examples, with "harden the VM" being the exemption to the statement. However, using the other sources we get a feel for what is demanded when hardening; Firewalls, HIPS and NIPS, web application protection, logging, as well as installing the latest security updates, i.e. keeping all systems patched.

Some examples of strengthening the network security (AWS) include using NACLs that allow management of IP traffic, security groups to manage access, firewalls and applying access control at other layers. This is what CSA touches on with their requirement above. It would seem then, that AWS is compliant with CSA recommendations. They also state to "harden software". We assume that for software that is not being developed by you, there is no way to harden it except to update it regularly. If the software is being developed by you, however, security should be a point of consideration throughout the development life cycle.

As a general recommendation we advise the customer (SaaS vendor) to discover what exactly is being promised by the IaaS vendor when moving to the cloud, as well as the technology the IaaS is built on. Different vendors promote different OS and VM level security best practices which lead us to believe there are differences in infrastructure and subsequently the security they provide (we reserve, however, that using premium services may or may not remedy these concerns). Things such as the possibility of side-channel attacks in multi-tenancy systems are highly tied to the type of tenant isolation [9].

4.6 Network Security

According to CSA, Network security consists of security services that restrict or allocate access and that distribute, monitor, log, and protect the underlying resource services [23, p. 168].

Azure states that the following network security best practices should adhere to [45];

- Logically segment subnets.

  Segmenting the subnets using a Network Security Group (NSG). Allowing you to put resources that belong to the same security zone or role in their own subnets. We generalize this as 'keep separate connection rules for each interface and limit them to only accept connections from required interfaces'.

- Deploy DMZs for security zoning.

  Azure recommends deploying a DMZ to further enhance the level of network security. This enables you to place your network access control management, monitoring, etc at the edge of your virtual network where you can enable e.g. intrusion detection/intrusion prevention systems or firewall rules.

- Optimize uptime and performance.

  They suggest employing load balancing whenever possible in order to increase availability by redirecting traffic. Performance is also improved since the processor, network and memory overhead for serving requests are distributed across the load balanced servers.

- Disable RDP/SSH access to Azure virtual machines.

  This is recommended by Azure because of the potential security problems these protocols pose. Attackers can gain access to virtual machines by using various brute force techniques and once they have access they can use the machine to compromise other machines on the virtual network.

IBM Best Practices states [33];

- Use the private network.

  IBM advice network spanning to enable system communication over a private network. Therefore, when possible, use a VPN connection to interact with your devices to ensure that the interaction is done in the most secure environment possible.

- Make sure to configure your firewall.

  Leaving your firewall in Bypass Mode can be seen as having a security system that is never turned on. It is highly recommended to create rules and activate your firewall in order to block unwanted activity.

- Do not leave known ports open on the public network.

  Recommended ports to disable or restricting access to on the public network is RDP and SSH. Failure to disable these ports can leave the system vulnerable. Consider moving RDP or SSH to a custom port if these services must be available.

The following concludes the best practices for network security in AWS [28];

- Always use security groups.

  They work as firewalls at hypervisor level for your instance/instances. Managing access to instances that have similar functions and security requirements by using security groups is highly recommended.

- Augment security groups with NACLs.

  By combining security groups with NACLs another level of control is provided since NACLs are not instance specific. NACLs can allow or deny traffic before reaching a security group when working in conjunction with security groups.

- Protect data in transit.

  Ensure confidentiality and integrity of data as well as the identities of the communicating parties by encrypting the data with IPSec which extends the IP protocol stack and allows applications on upper layers to communicate without modification.

- Network security layers.

  Apply network security at different levels of your network, e.g. external, internal and DMZ layers. This will further enhance the network security by enabling intrusion detection/prevention systems to be put at the edge of the virtual network.

AWS and Azure agree on that network security layers such as a DMZ or VPN as well as the use of NACL/NSG to be crucial parts of ensuring the network security of the SaaS. The NACL and NSG are roughly the same, Azure's NSG originates from an NACL but differs when it comes to the number of security groups and number of rules per security groups that is supported. The AWS NACL is limited to 20 security rules per subnet while the NSG can have up to 100 security groups and up to 200 security rules per security group. Consulting the IBM developer page there is no mention of NACLs or NSGs. There are however recommendations to implement DMZs. The best practice common for Azure and IBM is the disabling of known ports and even though this is not mentioned by AWS in their best practice, it is unlikely that they would suggest otherwise. Protecting data in transit is only listed by AWS, however it would be reasonable to assume that Azure and IBM considers this to be rather important as well. In the same way, although only mentioned by Azure, load balancing should be seen as best practice. AWS claims security groups acts as a firewall for your instance/instances and IBM list firewalls in their best practice. It would be surprising if Azure argued against the use of firewalls as a best practice. Furthermore, IBM suggests using VPN to increase the security of interactions with your devices. This might not be crucial but is definitely a feature.

One should note that at the network level, many of the security challenges are not cloud specific. Security challenges associated with clouds are exacerbated, but not specifically caused by the fact of it being in the cloud [12, p. 60].

4.7 Portability

Portability can be seen as how effortlessly applications as well as their associated data can be moved to another cloud provider [11]. Most cloud providers offer increased security and performance services at an additional cost. These services are often well tested and implemented and probably worth their cost in terms of money. However, by relying on too many services provided by a certain cloud provider to ensure the security of the SaaS renders it susceptible to changes made by the provider. The cloud provider might decide to discontinue a service or dramatically raise the price of it. If open-source or in-house alternatives are used instead the SaaS is less dependent on the cloud provider and thus has a higher portability. A higher level of portability can therefore increase availability.

CSA argues that portability is a key aspect to consider when selecting cloud providers since it can aid in preventing vendor lock in and deliver business value by allowing the deployment of an identical solution on different clouds [23]. NIST recommends that a cloud platform consumer should "formulate a strategy for future migration of Virtual Machines and their associated storage among alternate cloud providers" [22].

4.8 Recommendations and Model

In this section, we present our findings as a model for SaaS with the goal of highlighting the different aspects of security and performance needed to create a secure cloud application. Under each section, a selection of must-haves and nice-to-haves are listed, based on a synthesis of the information above.

Service Agreements

Service agreements touch lightly on the aspect of shared responsibility. They specifically touch on service availability, which you as a customer is highly dependent on. If the CSP's IaaS service goes down, your business is the one being affected. It is recommended that the SA and SLA are read and understood, specifically with regards to what areas of security you as a customer are responsible for. A misunderstanding of the responsibilities may result in financial loss for the customer, loss of data and/or other assets. NIST also recommends negotiating the service agreement and the remedies if the terms are not acceptable to you as a customer. This might work out fine for cloud providers who are in dire need of customers, but for the giants, you are likely to get what they offer and nothing more. Offering individual service agreements is not scalable for a world-class CSP. It is likely however that the large CSPs have other pay-as-you-go services that remedy these concerns.

Notably, Microsoft offers 100% recompense in the form of service credits (for use in their service) if the server is up less than 95% of the time (see Table 4.1). This comes out to about 1.5 days per month. If the uptime was that low, the question is; how many customers would like to stay with that specific cloud provider? This can seem like a generous offer, but in reality, anything less than nominal uptime is a bad sign. Another notable item is that Microsoft for its Single-Instance

Virtual Machines offers 99.90% connectivity, increasing to 99.95% when using an availability set (two or more instances). IBM has the most solid offer; 25% recompense for anything less than 99.90% availability.

One of the key limitations in the Service Agreements is security; providers generally assert that they are not responsible for the impacts of security breaches or for security in general, i.e. unauthorized modification or disclosure of consumer data, or service interruptions caused by malicious activity. Generally, service agreements are explicit about placing security risks on consumers. In some cases, providers promise to use best efforts to protect consumer data [22]. We also recommend comparing recompense levels and the possibility of tailored service agreements to best support the SaaS being built.
Must Haves:

- Compare providers for the best service agreement and service level agreement to make sure recompense is commensurate with risk and cost of downtime.
- Make sure you know the extent of uptime the CSP offers.
- Make sure you provision enough hardware to keep availability high.
- Make sure you are prepared for downtime.

Nice to Have:

- Build redundancy from the beginning in order to keep up with off-site downtime.
- Provision additional modifications to fit the organizational needs via support offerings.

Identity and Access Management

The providers are more or less unanimous when it comes to their best practices of identity and access management. Enforcing multi-factor authentication, having role or group based permissions for users, enabling single sign-on, securely managing credentials and administrator/root accounts and establishing a strong password management routine are the key factors shared between them. Multi-factor authentication and strong password requirements reduce the risk of a successful brute force attack, such as dictionary attacks. Single sign-on can encourage users to comply with strong password requirements since they do not need to remember several different passwords. However, if an identity using SSO is compromised, a higher risk of data loss occurs because the perpetrator gains access to more systems than if they required separate passwords. However, revoking access to a single sign-on entity is easier than revoking many separate ones. Multi-factor authentication is very secure but can result in increased effort depending on the amount of factors and their relative time-cost. We recommend that MFA is used to elevate privileges from user to administrative roles. De-provisioning should be done in order to minimize the number of accounts vulnerable to compromise.
Must Haves:

- Use single sign-on.

- Enforce multi-factor authentication for more privileged accounts.
- Role or group based permissions for users.
- De-provision accounts that are not used.
- Report and monitor user activities.
- Create a secure credential management routine.

Nice to Have:

- Use key management solutions, such as hardware security modules (HSMs) to store keys.
- Leverage IaaS solutions such as Key Vault, Key Protect, etc.

Handling of Data

Confidentiality and integrity of data at rest are of high importance. AWS provides several options for the customer to implement this, e.g. AWS can encrypt your stored data. They also offer the ability for customers to manage their own encryption keys, which can be considered to further enhance the confidentiality of sensitive information than if AWS would manage your encryption keys [46]. Microsoft Azure provides a similar service with their Azure Key Vault and IBM has the IBM Key Protect service. In the case of having the provider managing the encryption keys, questions arise such as who has access to the keys and where are the keys stored. Another way of dealing with these questions is to encrypt the data prior to uploading it to the cloud, but this might not always be a feasible approach.

Data in transit should always be encrypted, especially between cloud and client. Depending on the type of traffic, we see several methods of keeping these connections secured. On the application level, using HTTPS is mandatory, additionally, IPSec and VPNs are recommended by AWS and Azure.

For data at rest, encryption should be used in order to make sure that leaks do not expose plain text data. Keeping decryption keys separately stored is of great importance and again stresses the importance of having a secure credential management routine. Regarding who owns the decryption keys, the answer is one of how much you trust the organization that owns them. Owning the decryption keys enables data sovereignty. Encryption of data at rest additionally includes databases, VMs as well as SQL data input and output using e.g. middlewares designed for this. Hashing should be used to be able to provide integrity of data.
Must Haves:

- Use SSL/TLS or IPSec when transferring data, even internally.
- Use cryptographically secure algorithms for encryption of data at rest and in transit.
- Back-up your data regularly. Encrypt backups.
- Make sure that data decommissioning is irreversible.
- Restrict access to data using roles and permissions.

Nice to Have:

- Encrypt database I/O by means of middlewares.
- Establish a virtual private network to isolate the client-server communication.

Application Security

Even though it is highly recommended to be aware of the OWASP Top 10 when developing a SaaS application, it should be noted that it is considered to be a starting point for developing a secure application. It is not an exhaustive checklist that will ensure the security of your application if followed, it is rather a minimum requirement. Other attack vectors than the ones mentioned by OWASP Top 10 exist and these have to be considered as well. Incorporating security aspects in the software development life cycle and performing third-party penetration tests is advised to ensure that the security of the application is established and maintained. Premium services could be considered as an extension to prior security measures, they do however come at a price, both financially and as a loss of portability. As a part of the development life cycle, testing the software for security flaws can be done in-house or by using 3rd parties, but as with all software development and testing, should be done by people that are separated from the source code. The actions we recommend taking when testing security is to assess vulnerabilities by means of external parties, pen-test both internally and externally, white and black-box testing software and platforms. Keep the CSP informed of any of these actions, as they may be seen as illicit and may warrant a termination of contract unless notified.

Regarding VM and OS level security, a general hardening recommendation is in place. Do not assume that machine images and VM OSes that are used by default are secure. Implement a routine to secure them and improve them. CIS [44] have prehardened images for use that have gone through testing phases in order to withstand cyber attacks. However, these should be inspected and understood and not only adopted as a catch-all. Depending on the IaaS vendor, make sure you know the underlying technology and the risks that come with it. Treat VM OSes and other virtualized software as regular software, i.e. keep it updated and use firewalls, intrusion detection systems and access control to keep attackers at bay. Do not assume that security automatically is better when you use virtualized software.
Must Haves:

- Make sure you consider OWASP Top 10 to secure your system.
- Adopt a secure software development life cycle.
- Issue security assessments.
- Know the infrastructure you are paying for.
- Test security.
- Use network segmentation (VM).

- Treat virtualized software and infrastructure as on-premises solutions (VM).
- Continually patch and update both own and 3rd party software.

Nice to Have:

- Use Hardened Images from e.g. CIS or other organizations.
- Paying for cloud-specific scanning services such as ”Amazon Inspector” or ”IBM Security AppScan”. Keep in mind the possible performance hit.
- Use certified third-parties to conduct security audits and assessments.

Network Security

The importance of assessments that test and validate the network security of a SaaS vendor are highlighted in [27]. Vulnerabilities detected during network penetration and packet analysis tests, session management weaknesses and insecure SSL trust configuration can be exploited to gain access to user credentials and sensitive data as well as hijack active sessions.

Microsoft Azure suggests in their best practice white paper that availability can be thought of as being uptime and performance [45]. The reasoning behind this is that the data can be considered inaccessible if the performance is too poor for the data to be usable. Therefore uptime and performance are key factors from a security perspective, Microsoft Azure recommends utilizing load balancing as a popular and effective method to increase availability [45]. The use of load balancing also helps to mitigate one of the largest threats to availability, DoS/DDoS attacks. Both Microsoft and AWS suggest using load balancing to enhance resilience against DoS/DDoS attacks, AWS also recommends using additional services such as CloudFront which enables geoblocking, which might be useful for isolating attacks originating from a particular geographic location, and web application firewalls that target specific DDoS request to minimize the effect of a DDoS attack [47]. AWS, Microsoft and IBM [48, 49, 50] can provide bandwidth and processing to combat almost any network traffic spike, be it regular traffic or an actual attack. A downside is that it can incur high costs and opens up the door for "economic DDoS" attacks. However, if the cost to endure is less than the cost to mount a DDoS, provisioning can be used as a mitigation tool for these attacks. Disabling of known ports, firewalls and layering security, were all best-practices the providers recommended, as such, they will be considered as must have entities.
Must Haves:

- Implement security layers.
- Disable access through known ports and disable services not used.
- Utilize load balancing.
- Configure and activate your firewall.
- Use SSL/TLS or IPSec to protect data in transit.
- Perform network security assessments.

Nice to Have:

- Geo-blocking.
- Web application firewalls that target specific DDoS requests.
- Use VPN.

Portability

Many vendors that claim to be "open" and "standard based" provide services and extensions that can impede portability [23]. If portability is not appropriately addressed it can result in costly problems since provider lock-in may restrict the ability to move to another cloud provider that might be offering a more beneficial deal. Disruption of service due to the processing of incompatibility and conflicts between platform, provider or application differences is also undesirable [23]. CSA suggests the following considerations which will act as must haves when it comes to portability.
Must Have:

- Use open standards for identity such as SAML and develop an internal IAM system.
- Escrow encryption keys locally.

Nice to Have:

- Maintain encryption keys locally.
- Use open source components to avoid being locked-in by the provider

In Figure 4.2 the above stated best practices are illustrated as a table. In the following chapter, the model will be applied to the Axis-hosted cloud service, in order to determine the level of security it exhibits.

Area	Must haves	Nice to haves
Service Agreements	• Compare providers for the best SA and SLA • Make sure you know the extent of uptime the CSP offers • Make sure you provision enough hardware or services to keep availability high • Make sure you are prepared for downtime.	• Build redundancy from the beginning. • Provision additional modifications to fit the organizational needs via support offerings
Identity & Access	• Use SSO • Enforce Multi-factor Authentication • Role- or group-based permissions for users • De-provision accounts • Report and monitor user activities • Create a secure credential management routine	• Use key management solutions such as HSMs to store keys • Leverage IaaS solutions such as Key Vault, Key Protect, etc.
Handling of Data	• Use SSL/TLS or IPSec for data in transit, even internally. • Use cryptographically secure algorithms to encrypt data at rest and in transit. • Backup data regularly. Encrypt backups. • Make sure that data decomissioning is irreversible. • Restrict access to data using roles and permissions.	• Make sure database I/O is encrypted • Establish a virtual private network to isolate the communication.
Application Security	• Consider OWASP Top 10 when securing the system. • Adopt a secure software development lifecycle • Issue security assessments • Know the infrastructure you are paying for • Use network segmentation (VM) • Test security • Treat virtualized software as on-premises solutions. (VM) • Continually patch and update both own and 3rd-party software	• Use hardened machine images. • Use scanning tools from cloud service providers. • Use certified 3rd-parties to conduct security audits and assessments
Network Security	• Implement a layered security model • Disable access through known ports and disable services not used • Utilize load balancing • Configure and activate your firewall • Use SSL/TLS or IPSec to protect data in transit • Perform network security assessments	• Geo Blocking • Web application firewalls that target specific DDOS requests • Establish a vpn to isolate the communication
Portability	• Use open standards for identity such as SAML, and develop an internal IAM system. • Escrow encryption keys locally.	• Maintain encryption keys locally • Use open source components to avoid being locked-in by the provider.

Figure 4.2: A model of critical areas of security.

Chapter 5

A Case Study of an Axis-hosted cloud service

The Axis-hosted cloud service is a product of Axis Communications AB, a software that offers remote monitoring, increases alarm operation efficiency with remote verification and alarm reset, provides easy management of customers' security concerns, and more. The cloud service is aimed at surveillance companies, or 'video service providers'. A hosting provider develops the service and portal according to the video service providers requirements. In some instances, the hosting provider is the same as the video service provider, and in some instances, the hosting provider is Axis. When Axis is the hosting provider, the service is delivered as a SaaS, with the underlying infrastructure being Amazon EC2 instances. The cloud service user interface or 'portal' is accessed via a web browser, and provides the user with access to video streams for connected devices, as well as possibilities for configuring them, recording and upgrading the firmware of cameras. The portal receives alarms from cameras that are configured to send them based on specific triggers that the user specifies. The main module making the communication between camera and the cloud service possible is the 'cloud service proxy'. On the camera, the Camera Client has a persistent connection ('command channel') to the cloud service proxy.

5.0.1 Communication Between Cloud and Device

Immediately after a product is unboxed, the operator/user enters its credentials on the Axis-hosted cloud service web portal. These credentials along with other configuration parameters are sent to the *Axis Trusted Third Party(TTP)* (see Figure 5.1) for verification and to notify the system that the specific product is 'active'.

At this point, the TTP is ready for the "one-click-connection" step. If a user presses a dedicated hardware button on the device, it will try to connect to a known TTP address and a certificate exchange will be made, establishing trust not between the TTP and the device, but between the cloud service (cloud service proxy part) and the Camera Client. The TTP knows who owns the camera because of the registration step, and tells the camera which back-end proxy to connect to, see Figure 5.2. After this is done, the command channel is established between the proxy and the device.

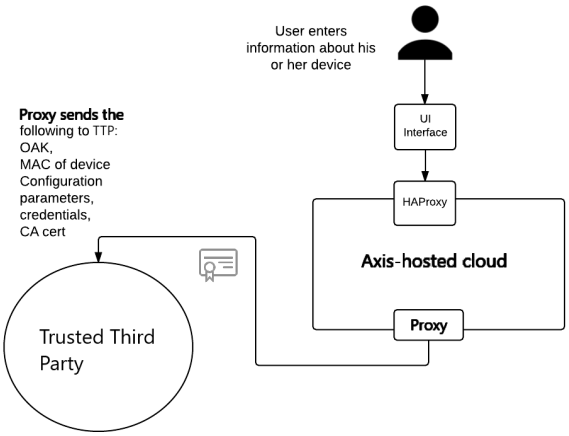

Figure 5.1: User entering credentials into web GUI

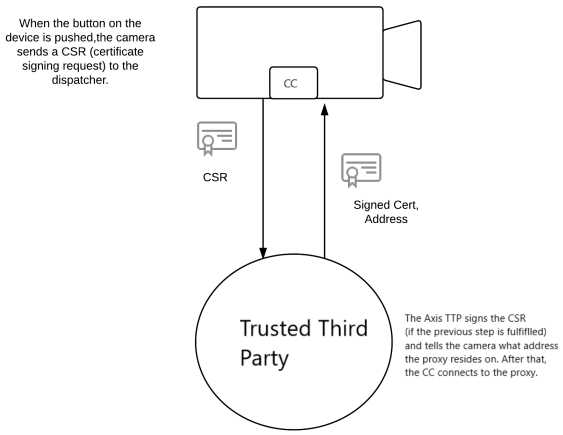

Figure 5.2: Dispatching the device using certificates.

When the dispatching is complete, the camera is susceptible to connection requests from the cloud. When a connection needs to be established for some reason, e.g. to stream images or to retrieve information from the device, a GET request is made over the command channel. The device then answers with a CONNECT to upgrade the connection to a TLS tunnel over TCP/IP. See [51]. After this, the requestor sends or retrieves information from the device as expected. When communication is complete, the tunnel is torn down and closed. A new connection is established every time a user requests data from the device, and the command channel is the only permanent connection between device and cloud.

In this setup the communication is one-way, meaning the device cannot spontaneously send requests or data to a remote address. A situation in which this becomes a problem is when the camera needs to notify users about user configured events that trigger in the camera, such as motion detection, alarms and other ac-

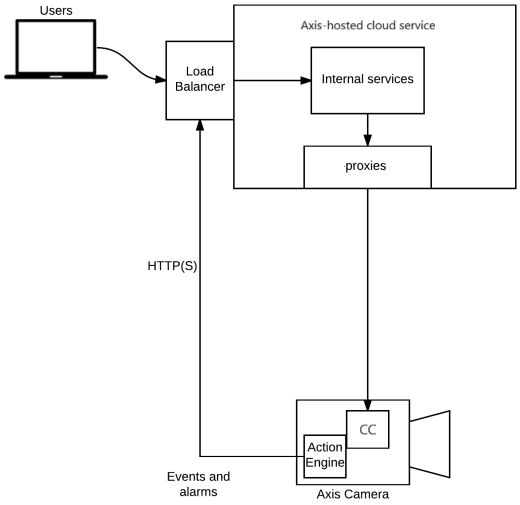

Figure 5.3: Axis-hosted cloud service to device communication.

tions. When this happens, ActionEngine, the module responsible for taking action based on events occurring in the camera, will establish an HTTP(S) connection to the cloud service outside of the command channel (see Figure 5.3). This is the mechanism that allows notifications from cameras in the cloud service. Depending on the type of notification, different modules inside the cloud will be informed. The cloud service has no way of knowing when these messages are scheduled to arrive, and thus can not decide if a transmission has been blocked.

5.1 Procedure

This section describes the process of developing the proof-of-concept solution. The goal was to include all communication in a single, two-way connection that could handle all data to and from the device. The principle is shown in Figure 5.4. With our current understanding of the systems involved in Axis cloud service, we developed an overview picture, shown in Figure 5.5. This image shows the pain points and the areas that needed change in order for our prototype to work. With this image as a starting point, we set about establishing the new communications channel first.

The starting point was the Secure Channel add-on (SCa) that was under development by a separate team, shown in Figure 5.6, and a simple back-end implemented in Node.js called the WebSocket Hub (WSH).

The communication initially consisted of nothing more than sending GET requests over a WebSocket connection and waiting for a response. This is shown next to points '1' and '2' in Figure 5.5. It should be noted that in both Figure 5.4 and Figure 5.5, the proxy is illustrated as relaying requests to the WSH, but this is not how the final design was made. This was simply a misunderstanding when

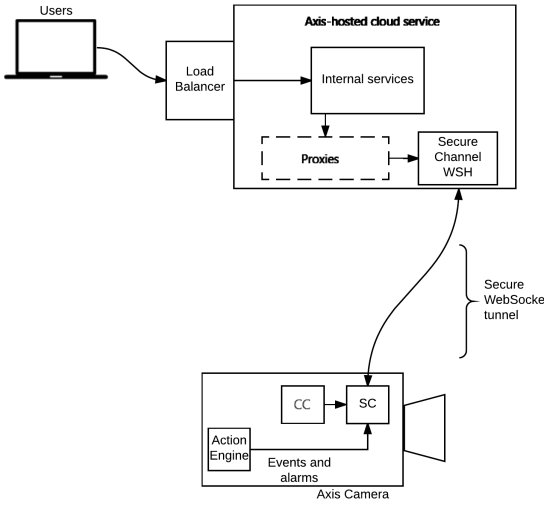

Figure 5.4: Desired structure.

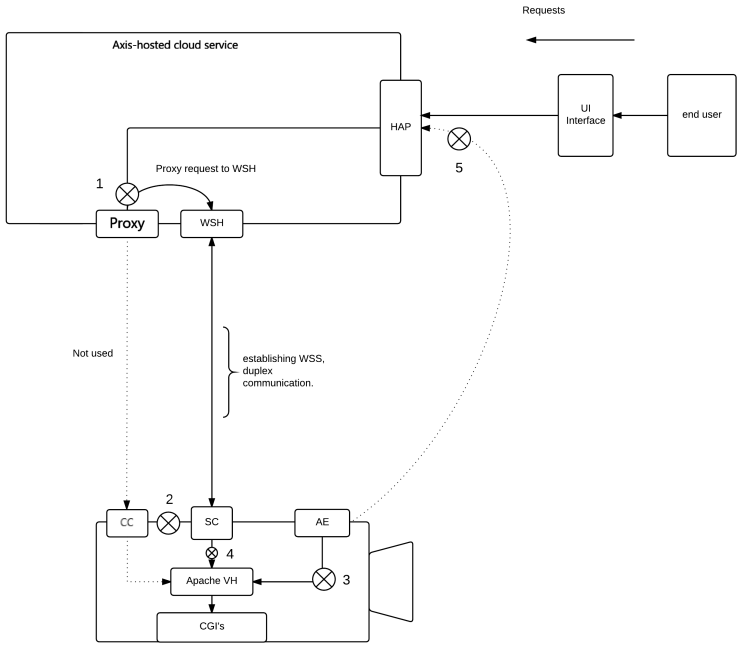

Figure 5.5: A top-down view of the Axis-hosted cloud service and related modules.

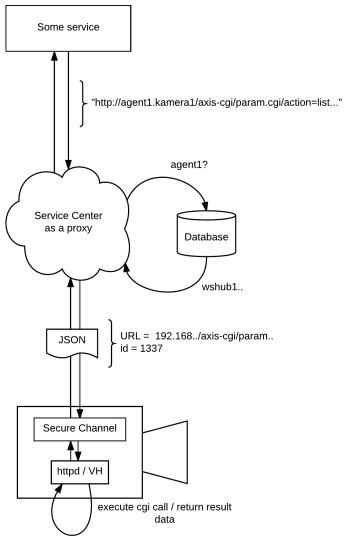

Figure 5.6: The internally developed solution - "Secure Channel".

starting the work. Instead, the WSH should handle all of the communication that the original cloud service proxy handles.

Some of the questions we needed to answer were how to establish trust between SC and WSH, and if we should use certificates that already existed or to create some kind of mechanism to distribute this, as an appropriate method for distributing certificates and dispatching devices had not yet been designed. The Secure WebSocket connection needs certificates on both ends for mutual authentication and without them, a connection is never established. Self-signed certificates were used in the meantime to allow for the TLS tunnel. Later, we developed our own dispatcher service to tackle this issue (See Figures 5.8 and 5.9). With certificates in place, we could start using the channel. Simple requests were made, yielding responses around 50KB of plain text. We quickly realized that our needs differed from the ones the SCa development team had. Mainly because they focused on sending large chunks of data *to the device* while we were more interested in the opposite, i.e sending chunks of data *from the device* to the cloud. Further development and modification to the add-on had to be made in order to fit our needs.

For the Node.js back-end to be able to replace the cloud service proxy component, the back-end would have to mimic the behavior of it. The component is a proxy that has an open connection established to each device that is connected to the cloud (the command channel). The cloud service proxy also features MAC addressing in addition to using IP addresses.

Figure 5.7 shows the role of the WSH. In this figure, requestors are typically modules inside the cloud (either triggered by client requests or by specific code), that need to communicate with a device. The WSH then does a lookup in a registry class that stores a mapping of IP and MAC. This is needed since MACs

are a requirement for addressing but the clients identify themselves using IPs when connecting (since WebSockets use TCP/IP). When a client connects, it sends a "hello" message along with its MAC address so the WSH can make this mapping. After that, the WSH handles the communication between the client and the device. When a request aimed for a device is received on the "internal" side (i.e., the cloud-facing side) a unique id is generated for the requestor. This is then sent to the device, and responses to this requestor have the same id appended to it.

The WSH does not simply pipe everything it receives but parses the JSON-formatted data coming in and reads the requestor id in order to send the response back on the connection requesting the information. This is also shown in Figure 5.7.

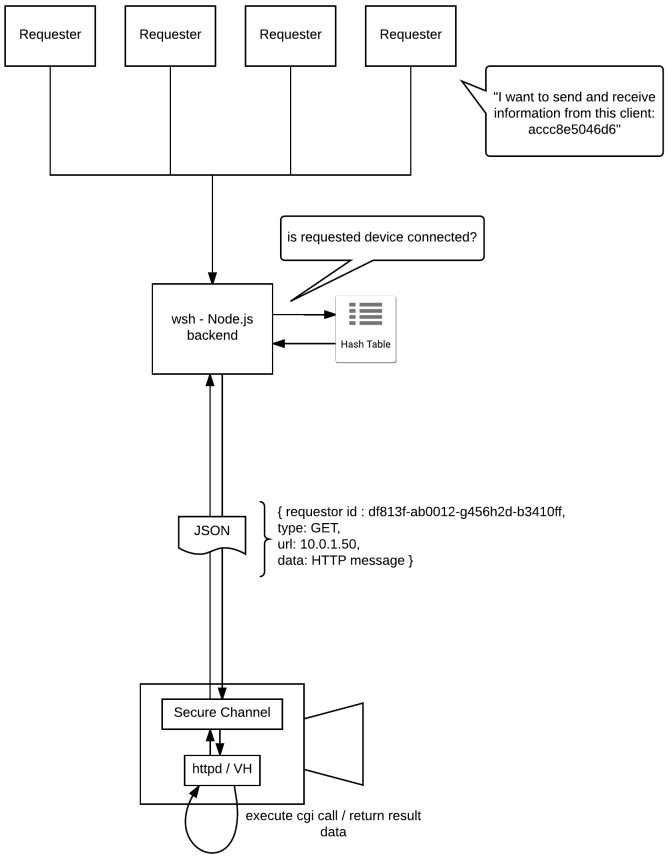

Figure 5.7: Our implementation of WebSocket Hub (WSH) and SCa on camera.

In order to facilitate the desired functionality where notifications from the ActionEngine are sent through the command channel rather than a side channel, it was necessary for us to develop a firmware plug-in for the device. This plug-in

would take on the same responsibilities as the previous notification plug-in, but instead, redirect the notifications to the SCa on the device. This issue is also present in the overview image, shown in Figure 5.5, point 3.

Since the WebSocket connection requires certificates on both ends to set up the connection, we had to distribute a certificate to the add-on in a secure way. We decided to utilize the already established command channel to dispatch the add-on and the backend. To enable this, the add-on had to expose a CGI that could be reached from the localhost on the device. A dispatcher server was developed, also in Node.js, that has the role of authenticating client and server. The dispatcher service asks the add-on through the established command channel for a CSR. The add-on generates a key pair and responds with a CSR. Upon receiving the CSR, the dispatcher sends its signed certificate and the trust chain. The dispatcher also informs the add-on of the address of the WSH to connect to. Once the add-on receives this information it tries to establish a WebSocket connection to the back-end. This way communication is established from inside the device's net, allowing for the same NAT/Firewall "hole-punching" properties that the cloud service proxy and CC connection use. This is illustrated in Figures 5.8 and 5.9.

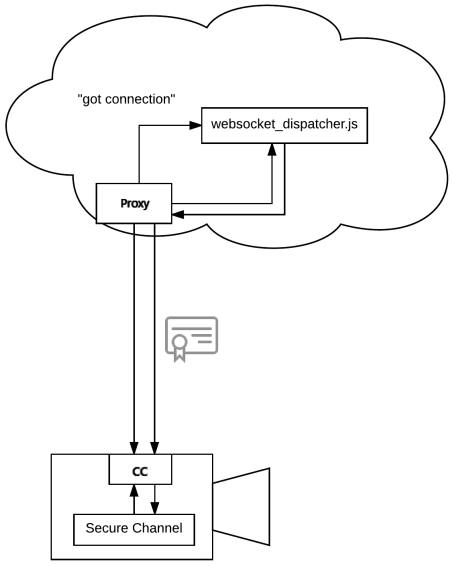

Figure 5.8: The new dispatching.

When the WebSocket connection is up, it is possible to leave the previously established cloud service proxy - CC command channel idle. All requests go through the newly established connection, as well as notifications that originate from the device. More demanding requests such as streaming video required more sophisticated modifications to the add-on. One of the major issues was the internal structure of the SCa, shown in Figure 5.10.

As the figure shows, the responses of the requests pass through a message queue prior to being sent to the WebSocket handler. The WebSocket handler, in

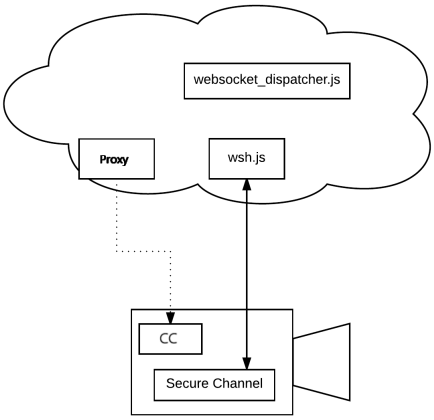

Figure 5.9: After dispatching using the new dispatcher.

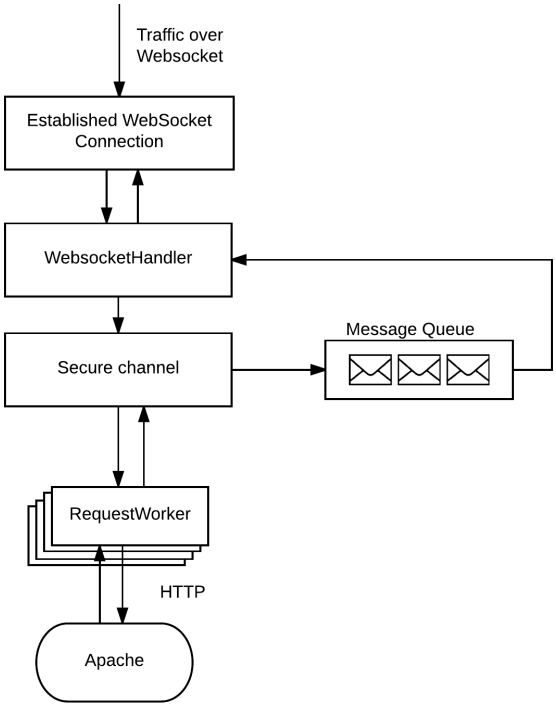

Figure 5.10: Message handling in Secure Channel add-on.

turn, sends a message from the message queue in a FIFO manner each time the

WebSocket connection is writable. Since the main usage of the Secure Channel from the original authors' point of view is to send large files, e.g. firmware upgrades and subsequently receive a message back with the result, the use of a single message queue does not affect performance. However, when requesting large amounts of data, e.g. a video stream, the message queue is prone to be filled up quickly if the WebSocket connection cannot keep up with sending the messages. This is a highly undesirable behavior for the cloud service.

As WebSockets have no notion of HTTP (A WebSocket connection exists on OSI layer 4), a message format for data has to be established. A common format is the Javascript Object Notation (JSON) format, (see [52]) which is a text format that uses key/value pairs and lists to represent data. Since JSON is realized using strings, binary data is not supported by the format. This means that escaping the binary data is a requirement for this protocol. The most common way of escaping binary is to Base64-encode it (see [53]). This puts some overhead on the data being sent, compared to e.g. sending using plain HTTP(S), which is how the cloud service proxy and CC communicate. However, Secure Channel was designed to handle strings primarily, and as such it has problems handling binary data. This means that adapting to the service center JSON protocol also called for a couple of modifications of the add-on, e.g. Base64 encoding of the binary data to be sent.

When testing the implementation, requests for smaller amounts of data (around 50 kB) did not differ much from the existing solution in terms of performance. However, when requesting a video stream the Secure Channel implementation had a severe drop in performance compared to that of the existing solution.

5.2 Results

This section presents the results to the questions mentioned in Section 1.1

1. *Analyze current client/server communication, determine its capabilities, deficits and future needs.*

   The capabilities were determined to be:

   (a) Be able to send binary data.
   Binary data is used when transmitting streams over HTTP, such as images and video. Because of the nature of the cloud service as a software for video surveillance, this is crucial. Uploading data to the cloud consists of recorded video that is represented in binary (over HTTP(S)).

   (b) Be able to handle streaming data, such as continuous video.
   Like stated above, streaming data in real-time to users who interact with the devices through the user-portal requires the solution to be able to handle chunked-encoding,

   (c) Be able to work through e.g company firewalls and NAT routers.
   By initiating a command channel from inside corporate firewalls (and subsequent connections), it enables the cloud service to communicate

with connected devices without having to configure the firewall and router. (With the exception of needing a port open for establishing the command channel).

(d) Be able to address clients using MAC addresses.
The reason for this is that the cloud service stores a database of the registered/connected devices. Since IP addresses are ephemeral, addressing via MAC is a way to communicate with devices without having to care about cameras having a specific network address. The cloud service proxy component enables MAC addressing by mapping MACs to IPs.

(e) Have support for the RTSP protocol.
In order for the cloud service to serve a video stream to users, RTSP or RTMP enables control of the streaming module of the Axis device.

The deficits were determined to be:

i. The use of cryptographically broken algorithms for some communications from older camera models.

ii. No two-way communication.

iii. No way to tell if alarm signals have been blocked.

iv. New TCP connections established every time data has to be sent.

v. No support for Server Name Indication (SNI).

Results:

The following results show how well our developed proof-of-concept solution met the demands and remedied the deficits of the Axis-hosted cloud service.

a. Our solution can send binary data, however via Base64 encoded JSON structures.

b. It handles streaming video at about 20% of the efficiency the current solution does. It scales inversely with the number of requestors.

c. It works through company firewalls because it uses the existing proxy-CC connection to establish trust, and establishes WebSocket connection from inside the camera's net.

d. It has support for MAC addressing.

e. It does not have support for the RTSP protocol. This was not implemented due to time constraints.

The deficits:

i. The use of cryptographically broken algorithms on older models was not solved within the allotted time frame. We will further discuss this in Chapter 6.

ii. Two-way communication enabled by means of a persistent WebSocket.

iii. Alarms signals now go through the WebSocket channel, as such, blocking attacks are infeasible or at least would notify the cloud service that the WebSocket is down.

iv. Since the WebSocket communication is two-way, no new TCP connections put unnecessary load on the devices.

v. SNI was not implemented. We will further discuss this in Chapter 6.

2. Based on the findings, evaluate the Secure Channel add-on;

   (a) *Is the Secure Channel capable of fulfilling the needs of the Axis-hosted cloud service, such as accommodating different types of traffic to and from the internal services of the cloud service?*

   The deficits listed above would need to be corrected in an improved secure communication channel, as well as the following items;

   i. Be able to send requests to the device's different common gateway interfaces (CGI's), and receive the same HTTP responses as if they had been made through the proxy-CC connection.
   If a replacement is to be made, there should be a consistency between the old and the new.

   ii. Support strong encryption using e.g. OpenSSL ciphers.
   There is an inherent need to secure the data in transit, especially considering its criticality. Actions being recorded on video may consist of theft, destruction, threats and other illegal actions. If this data could be modified, altered or blocked in transit, it could result in a loss of physical safety for people and property involved. Furthermore, blocking alarm signals and other notifications could render surveillance-as-a-service worthless, as alarm operators have nothing to act on. Being future-proof includes supporting interchangeable ciphers in case security flaws surface.

   iii. Prevent MITM attacks using e.g. PKI.
   For similar reasons as the above, ensuring that the device you are talking to really is the one it claims to be is crucial when sending possibly incriminating data.

   Results:

   i. The new solution is in many ways slightly more complex than the proxycloud service proxy-CC connection and does not simply proxy requests. As such, we cannot answer with 100% certainty that our solution handles all CGI calls and requests exactly like the current, however, for the ones we tried (receiving video, reading/updating camera parameters, receiving still images), the behavior was 1:1, meaning the solutions were indistinguishable from one another.

   ii. It does have support for strong ciphers via OpenSSL. Libwebsockets in C++ and the Node.js WebSocket has support for OpenSSL.

iii. We use secure WebSockets which enables client and server to mutually authenticate. Although we used self-signed certificates for our proof-of-concept, real certificate chains can and should be used. Thus, MITM attacks can be said to be mitigated.

(b) *Can the Secure Channel solution be altered or extended in order to accommodate the current and future needs of the Axis-hosted cloud service?*

Results: We are certain that the current prototype can be altered to support the cloud service future needs. Support for SNI can be introduced at an earlier stage such as at a load balancer that keeps track of the different WS hubs in the cloud. However, scaling for several concurrent requests needs to be implemented and this calls for a re-work of the Secure Channel design philosophy. It should be rebuilt from the ground with support for binary data and not have a singleton design to be able to scale for several requestors. Alternatively, Secure Channel could be designed to create an additional data channel that is permanently open.

3. *How does our solution with the included Secure Channel compare to a state-of-the-art-solution. Suggest steps to be taken in order to further enhance the security.*

Applying our model for a state-of-the-art solution on the Axis-hosted cloud service with our proof-of-concept solution yields the results shown in Figure 5.11. The lines that are checked/marked with green are practices that the organization fulfills, the red/marked with a cross are unfulfilled, and the yellow/checked with a footnote are those that are partially fulfilled.

Area	Must haves	Nice to haves
Service Agreements	• ✓Compare providers for the best SA and SLA • ✓Make sure you know the extent of uptime the CSP offers • ✓Make sure you provision enough hardware or services to keep availability high • ✓Make sure you are prepared for downtime.	• ✓ Build redundancy from the beginning.[1] • ✓ Provision additional modifications to fit the organizational needs via support offerings
Identity & Access	• ✓ Use SSO[2] • ✓ Enforce Multi-factor Authentication • ✓ Role- or group-based permissions for users • ✓ De-provision accounts • ✓ Report and monitor user activities[3] • ✓ Create a secure credential management routine	• ✗ Use key management solutions such as HSMs to store keys • ✗ Leverage IaaS solutions such as Key Vault, Key Protect, etc.
Handling of Data	• ✓Use SSL/TLS or IPSec for data in transit, even internally.[4] • ✓ Use cryptographically secure algorithms to encrypt data at rest and in transit.[5] • ✓ Backup data regularly. • ✗ Encrypt backups. • ✗ Make sure that data decomissioning is irreversible. • ✓ Restrict access to data using roles and permissions.	• ✗ Make sure database I/O is encrypted • ✓ Establish a virtual private network to isolate the communication.[6]
Application Security	• ✓ Consider OWASP Top 10 when securing the system.[7] • ✓ Adopt a secure software development lifecycle[8] • ✓ Issue security assessments • ✓ Know the infrastructure you are paying for • ✓ Use network segmentation (VM) • ✓ Test security • ✓ Treat virtualized software as on-premises solutions. (VM) • ✓ Continually patch and update both own and 3rd-party software	• ✗ Use hardened machine images. • ✗ Use scanning tools from cloud service providers. • ✓ Use certified 3rd-parties to conduct security audits and assessments
Network Security	• ✓ Implement a layered security model[9] • ✓ Disable access through known ports and disable services not used • ✓ Utilize load balancing[10] • ✓ Configure and activate your firewall • ✓ Use SSL/TLS or IPSec to protect data in transit • ✓ Perform network security assessments	• ✗ Geo Blocking • ✗ Web application firewalls that target specific DDOS requests • ✓ Establish a vpn to isolate the communication[11]
Portability	• ✗ Use open standards for identity such as SAML • ✓ Develop an internal IAM system.[12] • ✗ Escrow encryption keys locally.	• ✗ Maintain encryption keys locally • ✓ Use open source components to avoid being locked-in by the provider.

Figure 5.11: The model applied to the Axis-hosted cloud service case.

The numbers below correspond to the numbers of the footnotes in Figure 5.11

1. It is unclear whether redundancy was existent from the beginning, however, the design is redundant today.
2. SSO is used for CSP services, but not for the customer facing application.
3. Logging is insufficient, it is hard to prove WHO did WHAT.
4. Internal traffic is not encrypted. Reason: Layered security.
5. No encryption is done for data at rest. Encryption is done for public facing interfaces.
6. The cloud service uses VPC to separate application instances.
7. All of the OWASP Top 10 vulnerabilities are not covered. They have however been considered and it is an active choice from Axis not to cover some of the vulnerabilities.
8. The adoption of a SSDLC is in progress, but not yet fully established.

9. In the application layer, security is layered by means of security groups for specific services. Least-privilege and user accounts to make sure there are few points of entry. Cameras can still circumvent this "top-down" privilege structure by connecting to the 2nd layer.
10. Load balancing is not used for camera connections.
11. See 6.
12. An IAM system is developed, however, there is a lack of traceability in the system.

Generally, the suggestions marked with red in the model should be considered as steps to further increase the security of the SaaS, especially the ones declared as must haves. The areas marked with orange would also need to be asserted. Steps to be taken to further enhance security will be discussed in detail in Chapter 6.

4. *A design proposition of how the future needs can be included in the Secure Channel solution.*

SNI needs to be introduced at a load balancer, making more than one public-facing IP redundant. This is shown in Figure 5.12. This would result in a singular point of entry for the system that would be easier to secure. Additionally, using the Axis TTP solution would be preferable over the one we developed.

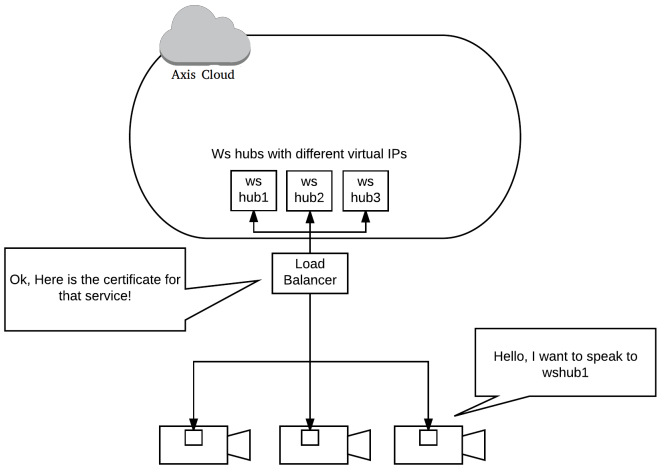

Figure 5.12: The use of Server Name Indication to direct traffic.

Chapter 6
Discussion

This chapter features a reflection on the results reported in Chapter 5 and the best practice model from Chapter 4. The discussion is divided into sections touching each of the topics in Section 5.2 and the best practice model in Section 4.8.

6.1 Best Practice Model

The decision to include the best practices of AWS, Azure, and IBM was mainly due to the fact that they are the most established providers on the market (market shares). One could argue that we should have included Google Cloud Platform (GCP) as well seeing as they have are often grouped up with Azure and IBM when comparing market shares. However, due to time restraints as well as the notion that the combined best practices of the three chosen providers would likely have covered the best practices of GCP, we chose not to include it.

Our motivation for not only including the top three providers best practices to create our model was to add different perspectives. By including non-profit organizations' perspectives on cloud security, as well as books by authors not related to any cloud providers, the model would be more versatile. It would also make it more ideal for our case where the SaaS provider is both a customer of an IaaS and a vendor of a SaaS.

Even though several perspectives are included in the model, it is not to be seen as an exhaustive checklist that guarantees success if followed. It is however derived from the best practices of top cloud providers and complemented with the knowledge of experts from different areas within the information security field, and as such it should be considered a viable tool when deploying a SaaS application in the cloud.

6.2 Capabilities

6.2.1 Binary data

Our new solution uses Base64 encoding over JSON structures to send all types of data. Using Base64 puts a 33% overhead on the actual traffic, with 3 input bytes encoded to 4 output bytes. Additionally, JSON takes up an additional couple of bytes per message. For most of the devices, this is OK, being able to

encode and decode Base64 fast enough to enable streaming. However, for older devices, this may result in too high CPU usage to be acceptable. Unfortunately, no tests with older cameras were made. Partly due to the fact that add-ons only are supported for newer models. The traffic overhead is acceptable for Service Center whose use cases are less dependent on low latency. In the Axis-hosted cloud service situation, however, where users may be on a mobile connection when requesting video, latency may become unacceptable. Keeping in mind the type of traffic the cloud service uses, which ranges from simple HTTP requests for small text messages to bandwidth-demanding video streaming using RTSP and other techniques, and the need for several clients to request this type of data from the same device, the overhead becomes too much. This motivates the use of a separate channel for all the 'heavy lifting', one which does not require Base64 or JSON. This would re-introduce some TLS-handshakes, but in the WebSocket specification, they are not as computationally expensive as SSL/TLS over HTTP.

6.2.2 Streaming Data

Our solution handles streaming data, which is mainly an issue of sending and receiving the right headers. We experienced approximately 1/5th of the speed the current solution performs at. However, we are certain that the structure of the Secure Channel Message Queue is the culprit, as it fills up very fast and can not deal with the raw traffic coming from the underlying modules that serve video. See Figure 5.10. Additionally, the WebSocketHandler class is configured to send strings and not binary data, which puts restrictions on the add-on as a whole.

6.2.3 Hole-Punching Corporate Firewalls

Our solution initiates communication through already established connections, which is not the preferred solution. Instead, the actual Axis TTP solution service should be reconfigured to handle the requests sent by WSH and Secure Channel modules respectively, alternatively, the Secure Channel should send the exact same requests as the original Camera Client, mimicking it. Apart from that, the hole-punching qualities of the solution remain intact, since connections are issued from the device's net.

6.2.4 MAC

MAC addressing works like the cloud service proxy-CC solution. The cloud service does not address clients using IPs, since they may have several cameras connected to a cloud service proxy on several local networks identifying themselves with the same IP. Instead, they know the (unique) MACs, and use them to send on a specific socket. We opted to store IPs and MACs, meaning you can address clients using both. This introduces problems when more than one device has the same IP. In the cloud service proxy, no IP addressing is possible. This should be remedied in our solution as it is not meant to be supported.

6.2.5 RTSP

RTSP requires a TCP connection to send commands such as play, pause, etc., And uses UDP to transmit the actual data. This data can be sent in a WebSocket, which as stated before works on the transport layer, and as such has no knowledge of application layer protocols. Two methods for enabling RTSP is implementing it in the add-on and in the WSH or using pre-programmed libraries for this. This, however, requires more time than what was available for this project and did not add specific value to our proof-of-concept.

6.3 Deficits in the Current Solution

6.3.1 The Use of Cryptographically Broken Algorithms on Older Models

The issue with cryptographically broken algorithms used for older firmware versions is hard to solve. Creating a backward compatible solution involves *not* changing the cloud service proxy-CC connection, and add-ons such as Secure Channel are only supported in newer firmware. The easiest way to ensure no cameras use cryptographically insecure algorithms such as RC4, is to keep them upgraded. This is also one of the main points in Chapter 4 - application or network security is not stronger than its weakest link, and keeping old software connected to the internet is an inherent security risk. Particularly because of the need for supporting older cryptos across the systems.

6.3.2 SNI

This is a feature that is best supported at an SSL termination layer or some other load balancing server in front of the actual WSH. The issue is partly one of economy, since buying additional IPv4 addresses is costly, especially when considering that the solution should be able to scale. Using virtual addresses behind a load balancer and using TLS with SNI, the devices are able to connect to a predefined load balancer that keeps track of what underlying WSH it should connect with. The principle is illustrated in Figure 5.12.

6.3.3 Alarm Signals Being Blockable

We present a solution to the problem of alarm signals being blockable. By sending them to the internal Apache web server instead of via a separate connection to a pre-defined load balancer IP, we can receive the requests on our CGI and send it via the WebSocket to a recipient in the cloud.

6.3.4 Two-Way Communication

Two-way communication was also a request from Axis to have implemented. The solution shows that this indeed works. This means that the camera can initiate connections to push data into the cloud. This has security concerns which merits investigation, since before, only solicited communication was possible. Now, devices may spontaneously send all types of data. Security measures should be

enacted to make sure connected devices only send legitimate data. Additionally, in the old solution, new TCP connections would be established and torn down for every request made through the could service proxy, but in our situation, this is not the case.

6.4 Evaluation of Secure Channel

SCa, which we used as a starting point, has been scrapped in favor of a new client written in another programming language. The new application does everything SCa did, but with support for older cameras as well. Thus, it was decided to only continue development on the new add-on. As such, the development of our module is likely to cease in favor of the new one. However, the new solution looks very promising, and although ours is likely to be scrapped, maybe it can be seen as an example of how not to proceed. The main positive outcome is that of eliminating ActionEngine not communicating via the established cloud-client communication. We showed that this is easily remedied by rerouting the traffic to a local CGI that can send it via the link to the cloud service. The data traffic throughput was lacking, however it provides very good security using encrypted communication and mutual authentication.

6.4.1 Consistency

Our Secure Channel implementation had the goal of replacing the existing communication solution. Thus, the WebSocket Hub had to parse the JSON data, decode the Base64 strings and apply HTTP headers to the package, as well as determining who the original requestor was. As the WSH sits between the two, it had to handle stripping HTTP headers as the receiver and add HTTP headers as the sender. This put unnecessary work on the WSH, and our opinion is that the best approach would be to pipe the raw data coming in from the device to the requestor without having to interpret it. Several factors hindered this, such as sending data unsolicited being a novel situation, the design of SCa, and more.

6.5 Discussion of Model Application and Further Enhancing Security

As seen in Figure 5.11, the cloud service is lacking security in a couple of areas. While the service agreements, networks security and identity & access management criterion are more or less met at must have level, the handling of data, application security and portability areas are lacking. Data at rest is not encrypted, insecure cryptographic algorithms are still used in order to support older cameras, data in transit inside the system is unencrypted as well as backups. However, if Axis considers that there is no need for them to encrypt their data at rest, and their customers are aware of this, the model might be slightly misleading. The same goes for the escrowing and maintaining of encryption keys for data at rest, if there is no need to encrypt it, why escrow and maintain keys at all? Making sure that

the decommissioning of data is irreversible seems unnecessary when the data is not encrypted in the first place.

By having a layered security model the need for internal encryption is not as crucial, especially since the data does not need to be encrypted. The lack of encryption on backups is however worrying and should be remedied. As for identity and access management, SSO is partially implemented in the system, i.e. only for the CSP services, and the traceability of which user has done what is insufficient. The IAM system used internally would need to be modified and/or extended in order to ensure proper traceability in the system. The adoption of a Secure SDLC (SSDLC) is in progress, it is however not fully incorporated, and as such should be considered an area for improvement.

To further enhance the security of the cloud service, the most essential areas to focus on would be to fully adopt an SSDLC and implementing load balancing for the camera connections. Phasing out older cameras is also important in order to strengthen the security of the system even more, especially since without the need to support older cameras, cryptographically insecure algorithms would not be used.

Chapter 7
Conclusion

Our proof-of-concept solution did not manage to provide a way to solve all the identified deficits of the current system. It did, however, give insight into how a more sophisticated solution could be designed. It highlights critical areas that need to be considered and will hopefully aid developers in designing and implementing the new solution.

When we applied our model onto the Axis-hosted cloud service, it gave us the valuable insight that despite being an established SaaS vendor, some of the considerations in the model might not always apply. E.g. the need for encrypting your data at rest can be considered optional if it is not required by the SaaS vendor's customers. Furthermore, we believe that our best practice model derived in Chapter 4 is a viable tool for not only evaluating the security of a solution but also serve as guidance for applications moving to/being built in the cloud. We would also like to point out that the model is not intended to guarantee all best-practices apply for every type of SaaS. Depending on the needs and requirements of customers of the SaaS, some of the considerations in the model could be optional.

References

[1] P. Mell, T. Grance, *The NIST Definition of Cloud Computing (2011)*, http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf, accessed on 2017-01-10.

[2] Cisco, *Cisco Global Cloud Index: Forecast and Methodology, 2015–2020 (2016)*, http://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf, accessed on 2016-12-08.

[3] G. Kulkarni, N. Chavan, R. Chandorkar, R. Waghmare R. Palwe, *Cloud Security Challenges (2012)*, https://www.researchgate.net/profile/Gurudatt_Kulkarni/publication/239732061_Cloud_Security_Challenges/links/0046351c283daf1730000000.pdf, accessed on 2016-11-25.

[4] P. Samarati, S.d.C di Vimercati, *Cloud Security: Issues and Concerns (2016)*, https://pdfs.semanticscholar.org/b8e7/966baaa39c12e2fc5e6463321921dbfc31fe.pdf, accessed on 2016-12-10

[5] R. M. Jabir, S.I.R Khanji, L. A. Ahmad, O. Alfandi, H. Said., *Analysis of Cloud Computing Attacks and Countermeasures (2016)*, 18th International Conference on Advanced Communication Technology (ICACT), accessed on 2017-03-22.

[6] N. Gruschka, M. Jensen, *Attack Surfaces: A Taxonomy for Attacks on Cloud Services (2010)*, IEEE 3rd International Conference on Cloud Computing, pp. 276-279. Retrieved from IEEE Xplore on 2017-03-22.

[7] Hacker Intelligence Initiative, *Man in the Cloud (MITC) Attacks*, https://www.imperva.com/docs/HII_Man_In_The_Cloud_Attacks.pdf, accessed on 2017-03-21.

[8] C. Mainka, V. Mladenov, F. Feldmann, J. Krautwald, J. Schwenk. *Your Software At My Service (2014)*, CCSW '14 Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security, pp 93-104.

[9] Y.Zhang, A. Juels, M.K. Reiter, T. Ristenpart, *Side Channel attack on PaaS clouds (2014)*, Clo, accessed on 2017-03-22.

[10] axis.com *About Axis*, https://www.axis.com/global/en/about-axis, accessed on 2017-02-28.

[11] R. L. Krutz, R. D. Vines, *Cloud security: A Comprehensive Guide to Secure Cloud Computing (2010)*

[12] T.Mather, S.Kumaraswamy, S.Latif, *Cloud Security and Privacy*, http://www.di.fc.ul.pt/~nuno/PAPERS/security3.pdf, accessed on 2017-03-09.

[13] D. Zissis, D. Lekkas *Addressing cloud computing security issues (2010)*, http://www.cs.joensuu.fi/~parkkine/LuK2015/CloudCompSecurity-FutureGenerCompSyst2012.pdf, accessed on 2016-12-13.

[14] ISO 27000, *ISO 27000, 4th Ed.*, http://www.27000.org/

[15] Microsoft Azure, *What is cloud computing*, https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/, accessed on 2017-03-01.

[16] Cloud Security Alliance, *About Cloud Security Alliance*, https://cloudsecurityalliance.org/about/, accessed on 2017-03-17.

[17] IETF, *IETF TLS* https://www.ietf.org/rfc/rfc5246.txt, accessed on 2016-11-23.

[18] IETF, *RFC4158* https://tools.ietf.org/html/rfc4158#section-1.5.1, accessed on 2017-03-06.

[19] T. Capan, *Why use Node.js?*https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js, accessed on 2017-03-23.

[20] Stack Overflow, *How to decide when to use Node.js*, http://stackoverflow.com/questions/5062614/how-to-decide-when-to-use-node-js, accessed on 2017-03-23.

[21] Synergy Research Group, *Cloud Market Leadership Q4 2016*, https://www.srgresearch.com/articles/amazon-dominates-public-iaas-paas-ibm-leads-managed-private-cloud, accessed on 2017-03-23.

[22] Badger, L., Grance, T., Patt-Corner, R., Voas, J. *Cloud Computing Synopsis and Recommendations (NIST), (2012)* http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-146.pdf, accessed on 2017-03-14.

[23] Cloud Security Alliance, *Security Guidelines for Critical Areas of Focus in Cloud Computing (2011)* , https://downloads.cloudsecurityalliance.org/assets/research/security-guidance/csaguide.v3.0.pdf, accessed on 2017-03-08

[24] Amazon AWS, *Service Level Agreement*, https://aws.amazon.com/ec2/sla/, accessed on 2017-03-23.

[25] Microsoft Azure, *SLA for Virtual Machines*, https://azure.microsoft.com/en-us/support/legal/sla/virtual-machines/v1_5/, accessed on 2017-03-23.

[26] IBM Bluemix, *Service Description*, http://www-03.ibm.com/software/sla/sladb.nsf/pdf/6605-06/$file/i126-6605-06_12-2015_en_US.pdf, accessed on 2017-03-23.

[27] R. Pradnesh, *Securing SaaS applications (2010)*, http://www.infosectoday.com/Articles/Securing_SaaS_Applications.htm, accessed on 2017-03-09.

[28] Amazon Web Services, *AWS Security Best Practices (2016)* https://d0.awsstatic.com/whitepapers/Security/AWS_Security_Best_Practices.pdf

[29] Microsoft Azure, *Data Classification for Cloud Readiness*, http://download.microsoft.com/download/0/A/3/0A3BE969-85C5-4DD2-83B6-366AA71D1FE3/Data-Classification-for-Cloud-Readiness.pdf, accessed on 2017-03-17.

[30] IBM developerWorks, *Gain confidence about data security in the cloud*, https://www.ibm.com/developerworks/data/library/techarticle/dm-1408datasecuritycloud/index.html, accessed on 2017-03-17.

[31] AWS, *Identity and Access Management*, http://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html, accessed on 2017-03-17.

[32] Microsoft Azure, *Azure Identity Management and access control security best practices*, https://docs.microsoft.com/en-us/azure/security/azure-security-identity-management-best-practices, accessed on 2017-03-23.

[33] IBM Bluemix, *Best Practices*, https://knowledgelayer.softlayer.com/articles/best-practices, accessed on 2017-03-13.

[34] IBM developerWorks, *Securing workloads on IBM Cloud: Identity and access management*, https://developer.ibm.com/cloudarchitecture/docs/security/securing-workloads-ibm-cloud/identity-access-management/, accessed on 2017-03-23.

[35] Shinder, W. T., Lussier, C., *Security Considerations For Infrastructure as a Service (IaaS) (2014)*, https://social.technet.microsoft.com/wiki/contents/articles/3808.security-considerations-for-infrastructure-as-a-service-iaas.aspx, accessed on 2017-03-14.

[36] Microsoft Azure, *Azure Data Security and Encryption Best Practices*, https://docs.microsoft.com/en-us/azure/security/azure-security-data-encryption-best-practices, accessed on 2017-03-16.

[37] IBM developerWorks, *Securing workloads on IBM Cloud: Data security*, https://developer.ibm.com/cloudarchitecture/docs/security/securing-workloads-ibm-cloud/data/, accessed on 2017-03-17.

[38] Open Web Application Security Project, *Open Web Application Security Project*, https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project, accessed on 2017-03-15.

[39] Open Web Application Security Project, *OWASP Project Top 10*, https://www.owasp.org/images/f/f8/OWASP_Top_10_-_2013.pdf, accessed on 2017-03-10.

[40] Microsoft Azure, *Secure an App in Azure App Service*, https://docs.microsoft.com/en-us/azure/app-service-web/web-sites-security, accessed on 2017-03-16.

[41] IBM developerWorks, *IBM AppScan*, https://www.ibm.com/developerworks/library/se-owasp-top10/, accessed on 2017-03-17.

[42] AWS, *Security Guidance for AMI Developers*, https://aws.amazon.com/marketplace/help/200897460, accessed on 2017-03-17.

[43] Microsoft Azure, *Security Best Practices for IaaS workloads in Azure*, https://docs.microsoft.com/en-us/azure/security/azure-security-iaas, accessed on 2017-03-27.

[44] Center for Internet Security, *Hardened Virtual Images*, https://benchmarks.cisecurity.org/hardened-virtual-images/, accessed on 2017-03-27.

[45] Microsoft Azure, *Azure Network Security Best Practices*, https://docs.microsoft.com/en-us/azure/security/azure-security-network-security-best-practices, accessed on 2017-03-14.

[46] Amazon Web Services, *AWS KMS cryptography*, https://d0.awsstatic.com/whitepapers/KMS-Cryptographic-Details.pdf, accessed on 2017-03-13.

[47] Amazon Web Services, *Denial of Service Attack Mitigation on AWS*, https://aws.amazon.com/answers/networking/aws-ddos-attack-mitigation/, accessed on 2017-03-14.

[48] Amazon Web Services, *Auto Scaling* https://aws.amazon.com/autoscaling, accessed on 2017-03-13.

[49] IBM Bluemix, *Bare Metal Server* https://console.ng.bluemix.net/catalog/infrastructure/monthly_bare_metal/, accessed on 2017-03-13.

[50] Microsoft Azure, *Autoscale*, https://docs.microsoft.com/en-gb/azure/monitoring-and-diagnostics/monitoring-overview-autoscale, accessed on 2017-03-13.

[51] IETF, *Upgrading to TLS Within HTTP/1.1*, (2000), https://tools.ietf.org/html/rfc2817, accessed on 2016-11-23.

[52] JSON, *JSON*,http://www.json.org/, accessed on 2017-03-28

[53] Base64, *Base64*, https://tools.ietf.org/html/rfc4648, accessed on 2017-03-28

Printed by Tryckeriet i E-huset, Lund 2017

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2017-572

http://www.eit.lth.se