Aalto University
School of Science
Master's Programme in Computer, Communication and Information Sciences

Gaurav Bhorkar

Security Analysis of an Operations Support System

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science (Technology)

Espoo, November 25, 2017

Supervisor:	Tuomas Aura, Professor
Advisor:	Henri Laamanen, M.Sc. (Tech.)

Aalto University
School of Science
Master's Programme in
Computer, Communication and Information Sciences

ABSTRACT OF
MASTER'S THESIS

Author:	Gaurav Bhorkar		
Title:	Security Analysis of an Operations Support System		
Date:	November 25, 2017	**Pages:**	81
Major:	Mobile Computing, Services and Security		
Supervisor:	Tuomas Aura, Professor		
Advisor:	Henri Laamanen, M.Sc. (Tech.)		

Operations support systems (OSS) are used by Communications service providers (CSP) to configure and monitor their network infrastructure in order to fulfill, assure and bill services. With the industry moving towards cloud-based deployments, CSPs are apprehensive about their internal OSS applications being deployed on external infrastructure. Today's OSS systems are complex and have a large attack surface. Moreover, a literature review of OSS systems security does not reveal much information about the security analysis of OSS systems. Hence, a security analysis of OSS systems is needed.

In this thesis, we study a common architecture of an OSS system for provisioning and activation (P&A) of telecommunications networks. We create a threat model of the P&A system. We create data flow diagrams to analyse the entry and exit points of the application and list different threats using the STRIDE methodology. We also describe various vulnerabilities based on the common architecture that OSS vendors must address. We describe mitigation for the threats and vulnerabilities found and mention dos and don'ts for OSS developers and deployment personnel.

We also present the results of a survey we conducted to find out the current perception of security in the OSS industry. Finally, we conclude by stressing the importance of a layered security approach and recommend that the threat model and mitigation must be validated periodically. We also observe that it is challenging to create a common threat model for OSS systems because of the lack of an open architecture and the closed nature of OSS software.

Keywords: OSS, BSS, security analysis, telco, provisioning

Language: English

Acknowledgements

I would like to express my sincere gratitude to my supervisor Professor Tuomas Aura and my advisor at Comptel Mr Henri Laamanen for their feedback and guidance on the thesis.

I would also like to especially thank Comptel Corporation for giving me an opportunity to work on the thesis. I appreciate my R&D team for bearing with my absence and allowing me to focus on the thesis.

I want to thank my friend Manish Thapa for his constant encouragement and support. I appreciate and will miss the countless discussions over tea, lunch and dinner.

Finally, I want to thank my family and friends for all their support.

Thank you! Dhanyavad! Shukriya!

Espoo, November 25, 2017

Gaurav Bhorkar

Abbreviations and Acronyms

AES	Advanced Encryption Standard
ADSL	Asynchronous Digital Subscriber Line
API	Application Programming Interface
AuC	Authentication Center
ATIS	Alliance for Telecommunications Industry Solutions
BSS	Business Support System
BRAS	Broadband Remote Access Server
CPE	Customer Premise Equipment
CSP	Communication Service Provider
CSRF	Cross Site Request Forgery
CRM	Customer Relationship Management
DES	Data Encryption Standard
DFD	Data Flow Diagram
DoS	Denial of Service
DDoS	Distributed Denial of Service
DSL	Digital Subscriber Line
DSLAM	DSL Access Multiplexer
EAI	Enterprise Application Integration
ESB	Enterprise Service Bus
FTP	File Transfer Protocol
HLR	Home Location Register
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP Secure
IEC	International Engineering Consortium
IPTV	Internet Protocol Television
JMS	Java Message Service
LDAP	Lightweight Directory Access Protocol
MitM	Man-in-the-Middle Attack
NE	Network Element
NEI	Network Element Interface

NMS	Network Management System
NVD	National Vulnerability Database
OWASP	Open Web Application Security Project
OS	Operating System
OSS	Operations (or Operational) Support System
OM	Order Management
P&A	Provisioning and Activation
PII	Personally Identifiable Information
RADIUS	Remote Authentication Dial-In User Service
REST	REpresentational State Transfer
SDLC	Software Development Life Cycle
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
STRIDE	Spoofing, Tampering, Repudiation, Information disclosure, Denial of Service, Elevation of Privilege. A threat modelling technique by Microsoft
SIM	Subscriber Identification Module
SSL	Secure Sockets Layer
Sync/Async	Synchronous/Asynchronous
TCP	Transmission Control Protocol
Telco	Telecommunications Company
TL1	Transaction Language 1
TLS	Transport Layer Security
TMF	TM (TeleManagement) Forum
WSDL	Web Services Description Language
XML	eXtensible Markup Language
XSD	XML Schema Definition
XSS	Cross Site Scripting

Contents

Chapter 1

Introduction

The telecommunications industry has played a crucial role in the advancement of the information and the communication age. The industry has seen tremendous transitions in the last century and continues to grow rapidly. From offering telephone services traditionally, the industry has grown to offer services including Mobile, Internet, IPTV, Cloud, and even mobile payments. Such a vast array of services and an ever growing number of subscribers has allowed telecommunication service providers to increase their revenues every year. In 2015, Insight Research projected that the worldwide telecom services revenue was \$2.2 trillion and is expected to reach \$2.4 trillion in 2019 [32].

Initially, telecommunications network infrastructure was not as complex as it is today. As the types of services offered by a Communications Service Provider (CSP) increase, so does the complexity of telecommunication networks. There is new infrastructure constantly deployed and old infrastructure maintained to support legacy services. For example, at the time of writing of this thesis, CSPs are gearing up for the upcoming 5th generation (5G) of mobile networks but they still support and maintain the 2nd generation (2G) legacy network infrastructure. CSPs need software systems to monitor, configure and manage such a complex network infrastructure. Furthermore, in order to cut costs and speed up processes, CSPs are reluctant to hire human resources where the work can be automated by software. One such example is the so-called swivel-chair integration, where a person re-keys request details from one system, e.g., a Customer Relationship Management (CRM) system to another system, e.g., a Network Management System (NMS). Therefore, as the number of subscribers and the services increase, there is the need for efficient integration of business software with the network infrastructure and for supporting the end-to-end operations of a CSP seamlessly.

An Operations (or Operational) Support System (abbr. OSS), as the name suggests, is a system of software components working together to support the operations of any enterprise. OSS systems are used in a variety of fields including the telecommunications industry. Communication Service Providers are one of the largest users of OSS systems because of the complex landscape of network infrastructure and the services offered to their customers. Telecommunications operators need OSS systems in order to manage, configure and monitor their networks. In addition, OSS systems also perform service fulfillment, assurance and billing for the CSP's customers. In other words, besides monitoring networks, OSS systems orchestrate services by configuring the necessary elements in a network.

Security is an important area of concern for telecommunication companies. Modern day security expectations require enterprises to protect the privacy of their customers, prevent unauthorized access, provide uninterrupted service, etc. Many companies have annual security audits to improve the security of their networks. OSS systems are no exception to such audits either. Since OSS systems form a middleware between the customer-facing software and the network elements in the infrastructure, a large amount of sensitive information passes through them.

OSS products have been in use for the last 25 years and have undergone numerous changes in architecture and features to support the changing network landscapes. The security expectations that were there two decades ago have changed considerably. Today's OSS solutions are more complex and have a larger attack surface. Many of the OSS systems are transitioning to the cloud. Thus, OSS products are not internal to the enterprise anymore. OSS vendors and customers must focus on strengthening the security of their OSS solutions.

This thesis project has been carried out at Comptel Corporation, which offers several OSS products including Provisioning and Activation, Catalog, and Order Management. Considering responsibly the company's disclosure policies, we have tried to combine OSS expertise from the company and security research to do a security analysis of the most common components of the OSS architecture (Provisioning and Activation).

1.1 Problem Statement

> *A great many of today's security technologies are "secure" only because no-one has ever bothered attacking them* [30].

Gutmann makes the above statement in his book about security engineering. The quote holds true for most computer software, and OSS systems certainly are exposed to cybersecurity threats. Most CSPs see OSS systems as internal to the enterprise, meaning that the software is hosted and runs within the borders of the enterprise intranet. Today, with the industry moving towards cloud-based solutions, it is imperative for OSS vendors to revisit the security assumptions of their products. Furthermore, a literature review on the security of OSS systems does not reveal much information. Perhaps, a well-defined threat model and mitigation are needed.

The goal of this thesis is to do a security analysis of a typical OSS (Provisioning and Activation) system. This includes the following:

1. Explain the importance of securing OSS systems
2. Create a threat model
3. Find out common vulnerabilities
4. Suggest mitigation techniques

1.2 Structure of the Thesis

This thesis is divided into six chapters in all. The reader is advised to read the chapters sequentially.

Chapter 2 gives a background on OSS systems, their typical components and an example of how service provisioning works. The chapter also explains the different security terminologies used in the thesis.

Chapter 3 describes a threat model for a typical OSS (Provisioning and Activation) system. We study the assets and adversaries, create diagrams and analyse threats in this chapter.

Chapter 4 describes common vulnerabilities that could be present in an OSS system.

Chapter 5 suggests mitigation strategies for OSS vendors.

Chapter 6 summarizes the important points in the thesis and draws conclusions. We also discuss a survey we conducted to learn about the perception of security in the OSS industry.

Chapter 2

Background

This chapter explains the background information required for the subsequent chapters. We explain what is an OSS system, its components, and a common architecture in Section 2.1. Section 2.1.3 demonstrates a typical provisioning and activation scenario on an OSS system, which would help the reader understand the provisioning and activation process. Section 2.2 explains the required security terminologies. Finally, we describe the security analysis process in Section 2.3.

2.1 What is an OSS System?

An Operations Support System (OSS) is used by Communications Service Providers (CSP) to manage their networks. An OSS system supports several functions that are required to manage a telecommunication network such as service provisioning and activation, network inventory, fault management, and network configuration. CSPs typically collaborate with OSS vendors to customize their OSS systems depending on their network structure and service portfolios. According to the TM Forum Business Process Framework for Telecomunications Operators, an OSS system offers Fulfillment, Assurance, and Billing of services [82].

An OSS system is usually accompanied by a Business Support System (BSS), which acts as the customer-facing software and accepts service-related requests from the customers or the operator. The requests are forwarded to the OSS system for the actions to be performed on the network. It must be noted that not all requests to OSS systems involve network configuration.

2.1.1 OSS Applications

A CSP has different operational activities depending on the type of business it operates. Accordingly, an OSS system consists of several different application components based on the requirements of the communications service provider. An OSS landscape of one CSP might be quite different from its competitors. However, most of the OSS systems have a common subset of application components. The International Engineering Consortium (IEC) report [33] on OSS systems lists OSS applications for various types of networks (e.g., Digital Subscriber Line (DSL), Global System for Mobile Communications (GSM), and Internet Protocol Television (IPTV)). The following sections list and explain the application components that are used in a typical OSS system.

2.1.1.1 Catalog System

CSPs constantly need to come up with innovative product bundles with regards to changing market scenarios. A Catalog System, also called as a Service Catalog provides services to manage the various technical products and bundles offered by a CSP. The Catalog System increases the system modularity and reusability by using predefined service offerings. The TM Forum's Frameworx Standard for Information Framework (SID) defines different types of catalogs, viz., Product Catalog, Service Catalog, and Resource Catalog, and explains the related information models [83].

An example of a product bundle in a Catalog System is a service plan offered by a typical Telecom operator to its customers. For example, consider a *Home Plan* which consists of several individual services such as a telephone subscription with 200 minutes of inclusive calls, a 10 Mbps broadband connection, and a basic IPTV subscription.

2.1.1.2 Inventory

An Inventory is a database that tracks and manages all the network related resources used by a CSP [9]. Depending on the nature of the product, OSS vendors use different names for this component such as Network Inventory, Service Inventory, or Resource Inventory. The main task of the Inventory is to manage and keep track of the telecommunications infrastructure (network, resources and services). Thus the CSP has end-to-end stock-keeping of the resource utilization in the infrastructure.

For example, the operator can query the Network Inventory to find out the number of DSL Access Multiplexer (DSLAM) ports that are free in a particular telephone exchange. The Inventory provides services to the order management or provisioning and activation components to check resource availability before provisioning or activating a service.

2.1.1.3 Order Management

An Order Management (OM) system is used to manage order processing. As per the TM Forum, an OM system handles the end-to-end lifecycle of a customer's request [81]. The order can be anything ranging from a request for configuring products and services to a trivial enquiry. The OM system captures an order or a request from the northbound BSS system and then decomposes the order into subsequent smaller requests according to the product catalog information provided by the catalog system. The subsequent requests are then sent to the provisioning and activation system. The OM system implements a business workflow to handle the order processing.

The OM system has interfaces to several systems including northbound CRM systems, Product Catalog, Inventory, and the provisioning and activation system. Furthermore, it provides a user interface to the user to graphically track the progress of the order processing.

In many cases, an OM system overlaps with the provisioning and activation system and also falls partially in the BSS area [81]. Some vendors try to include the functionality of order management in their provisioning and activation systems to simplify the system architecture.

2.1.1.4 Provisioning and Activation

A Provisioning and Activation (P&A) system is used to provision resources and activate services on a telecommunications network based on the customer requests. It automates the tasks of manually connecting to the network elements and configuring them. As per the ATIS Telecom Glossary [19], a provisioning process has several definitions as follows:

"The process of configuring hardware and software to activate a telecommunication service"

"The processes that supply a telecommunications service to the customer including all equipment"

The process of provisioning might not always involve delivering hardware equipment to the customer but in some cases it is necessary. Activation, on the other hand is the process of configuring the network infrastructure so that the customer can start using the service. During the activation phase, the P&A system checks if the network is provisioned and then updates the network elements with appropriate configuration to enable the service. After activation is complete, the system updates the necessary components such as the Network Inventory and notifies the northbound BSS system of the request status.

The P&A system has an important sub-component called as the provisioning logic, which runs on every inbound request. The provisioning logic determines important validations such as checking the inventory and verifying if the services are already provisioned. The provisioning logic also determines the splitting of a request into smaller tasks in order to fulfill the request. Furthermore, it manages the workflow for the execution of the request. The provisioning logic is customized according to the network infrastructure and service portfolios of the CSP.

The P&A system is an important component in the OSS landscape of a CSP from the point of view of reliability, performance, and security since it forms a middle layer between the customer-facing BSS applications above and the network infrastructure below. This leads to a high number of incoming and outgoing connections to the system.

2.1.1.5 Network Elements

A Network Element (NE) provides networking facilities. Typically, these devices route the data or provide supporting functionality for the same. Formally, the US Telecommunications act of 1996 defines an NE as, "A facility or equipment used in the provisioning of a telecommunications service" [28]. The components of the GSM network switching subsystem such as Home Location Register (HLR), Authentication Center (AuC), and Voicemail System (VMS). are NEs as are the components of the broadband network such as DSLAM, Broadband Remote Access Server (BRAS), and Remote Authentication Dial-In User Service (RADIUS) server. Devices such as Lightweight Directory Access Protocol (LDAP) servers or generic servers running on the network can also be termed as NEs.

In order to provision and activate a service, several NEs must be configured. This configuration is done by the P&A system by connecting to each NE and applying the configuration according to the service specifications. Therefore, the P&A system maintains a template of operations according to the NE type and the connection interface.

An NE can have multiple interfaces for configuration management. Some of the common interfaces include a Command-line Interface (CLI), Transaction Language 1 (TL1) defined by Telcordia [80], Common Management Information Protocol (CMIP) [34], Simple Network Management Protocol (SNMP) [24], File Transfer Protocol (FTP), Representational State Transfer

(REST), and Simple Object Access Protocol (SOAP). The traditional interfaces such as CLI and TL1 are explained by Mishra [54]. In some cases, NE vendors have their own proprietary interfaces for which the OSS vendors need to develop custom middleware interfaces [54].

2.1.1.6 Field Service Management System

A Field Service Management System is used for on-field services which require an employee of the CSP or a contractor to travel to customer premises and install or fix equipment physically. Consider a DSL provisioning scenario where the P&A system creates a manual task in the Field Service Management System for a technician to install a DSL modem at the customer's premises.

2.1.1.7 Other Components

The components present in a CSP's OSS landscape vary according to the type of communication services offered. The above sections listed the most common components. Examples of other components include Number Management Systems and Subscriber Directory Systems. The OSS system might also internally contain sub-components such as EAI Middleware or Message Queues (e.g. JMS).

2.1.2 OSS Architecture

As mentioned in the earlier section, the architecture of an OSS system depends on the type of the CSP and the services provided by them. Some CSPs go with a multi-OSS solution while others prefer a single OSS to manage their entire service portfolio. In this section, we establish a common architecture for an OSS solution with the focus on service provisioning. This architecture will be used as a reference for further security analysis. Figure 2.1 shows the components and the data connections in a typical P&A focussed OSS system.

2.1.3 Example: Service Provisioning

In this section, we explain an example service provisioning process. In this scenario, the customer creates a DSL service provisioning request in the CRM system. The following points below specify the data flow and the provisioning steps executed by the P&A system. The scenario is visualised in Figure 2.2. Notice the connections and the data exchanged between the components, which are important while considering the security implications.

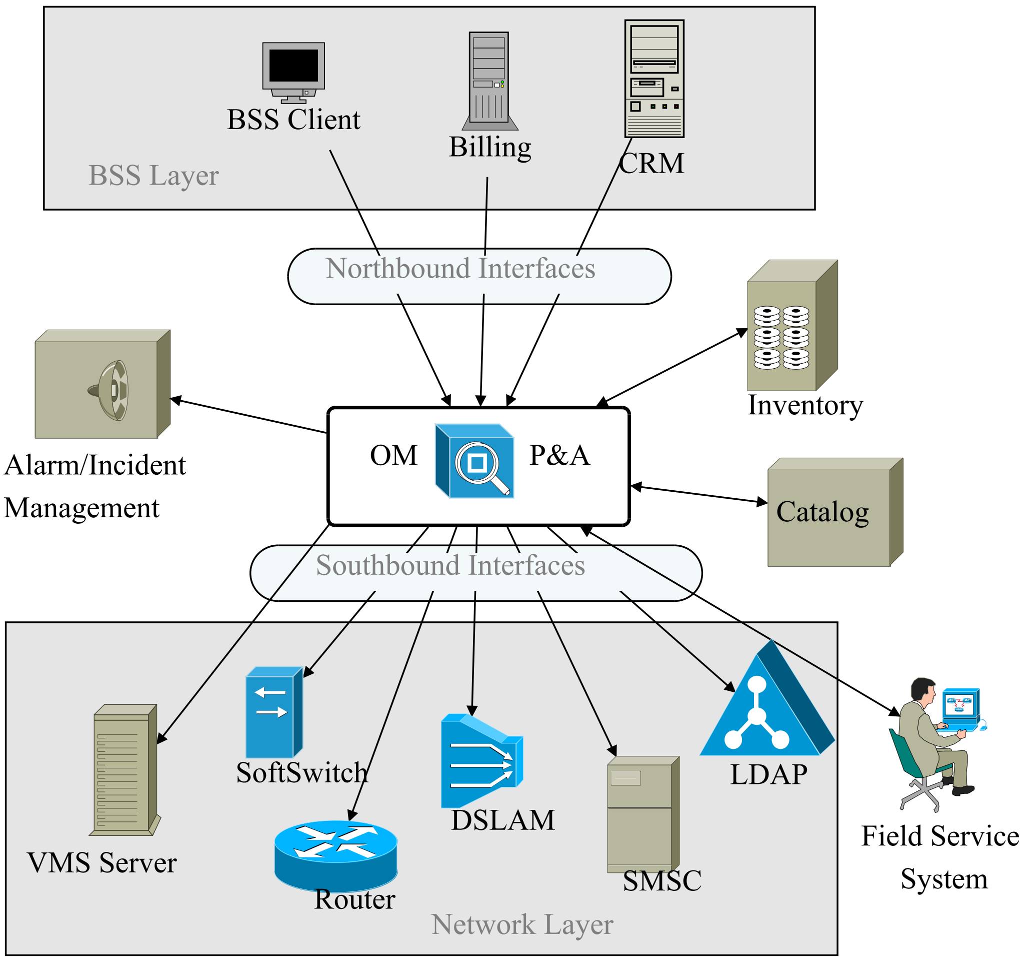


Figure 2.1: P&A Architecture

1. As soon as the operator puts the customer request in the CRM system, it sends a request to the P&A system for further processing.

The P&A system executes the next tasks according to a predefined provisioning logic.

2. The P&A system queries the Network Inventory to check if there is free capacity (free ports on DSLAM, hardware availability, etc.). If there is capacity and equipment available, the Network Inventory is updated.

3. Configure the BRAS NE with appropriate parameters.

4. Create a manual task for the engineering staff to install the Customer Premise Equipment (CPE) and connect the line to DSLAM manually.

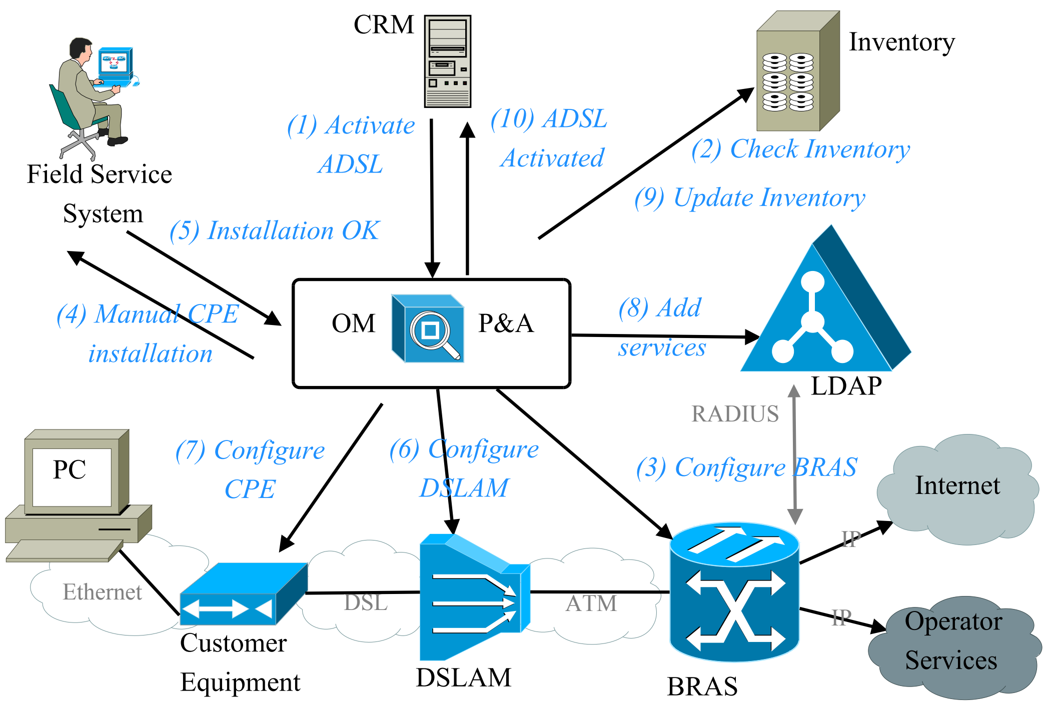


Figure 2.2: DSL Provisioning Example

The P&A then waits for a confirmation of the manual task and keeps the request pending.

5. If the status of the manual task received is complete (that is, all hardware is installed), continue with the next steps.

6. Configure the DSLAM NE to activate the DSL connection for the customer.

7. If required, configure CPE (customer's modem) remotely

8. Create an entry in the LDAP/RADIUS NE if it does not exist. Add the requested subscribed services, authorizations, etc.

9. Update the network inventory with the new capacity.

10. Send back a response to the CRM system the final status of the provisioned service. Consequently, according to the configuration, send a notification to the billing system in order to start billing.

2.2 Security Terminology

In this section, we explain the security terminology that will give background information to the reader for the subsequent chapters. A glossary of key information security terms is presented by Kissel [44] and in RFC 4949 [75]. Following are the most common definitions related to computer security discussed in this thesis:

Asset Any application, data, person, property, etc. that is important for an organization or an individual [44].

Attack An unauthorized attempt to access services or information. An attempt to destroy, disrupt, or deny services or information, etc. [44].

Adversary An entity that attacks a system or is a threat to the system [75]. An entity with a malicious intent.

Vulnerability A flaw or weakness that could be exploited to compromise the system [75].

Threat Anything that can exploit a vulnerability in order to cause damage, theft, denial of services or information, etc. [44].

Threat Model A threat model is a process by which potential threats can be identified and enumerated in software.

Risk The level of impact on an organisation's services, assets, individuals, etc. from an information system operating, given the potential impact of the threats and the likelihood of that threat occuring [44].

Spoofing "Pretending to be someone or something" [76]. Spoofing usually involves stealing an identity in order to pretend to be someone or something.

Tampering Modifying anything that one is not authorized or supposed to modify [76]. This includes modifying an information store or data flow.

Repudiation Claiming that one did not do something [76]. For example, claiming that one did not use a service that was billed.

Information Disclosure Leaking information to someone who is not supposed to see it [76]. For example, stealing financial data and making it available to the public.

Denial of Service A type of attack that leads to a service being unavailable to users [76]. A disruption in services often causes significant financial losses.

Elevation of Privilege Doing something that one is unauthorized to do [76]. For example, a normal user modifying or accessing something that only an administrator should do.

2.3 Security Analysis

The term *Security Analysis* is not well defined and often includes several alternative terms such as *Threat Modelling* and *Risk Assessment* [45]. In general, analysing the security of a software involves identification of threats, vulnerabilities, risks, attacks, and countermeasures [45]. A structured approach for analysing the security is threat modelling [58]. In this thesis, we create a threat model, identify vulnerabilities and countermeasures for an OSS system.

A threat model helps to identify the most likely threats in a systematic way. Myagmar et al. [55] explain why threat modelling should be considered a basic security requirement. A system cannot be secured by merely using the industry standard encryption and complex security jargon. The paper [55] also explains the common high-level steps in threat modelling, viz., identification of system characteristics, asset and entry points, and threats. Examples of threat modelling techniques are the STRIDE methodology [79], PASTA [84], and Trike [72].

The STRIDE methodology was described by Microsoft. It categorises threats based on spoofing, tampering, repudiation, information disclosure, denial of service, and escalation of privilege [79]. Refer to Section 2.2 for the definitions of each STRIDE category.

We use the following four-step framework described by Shostack [76] for security analysis. In addition, we also find out the common vulnerabilities that are possible in the context of OSS systems.

1. **Diagramming**: Understanding the data flow in the system by drawing a Data Flow Diagram (DFD).
2. **Threat Enumeration**: Classifying the threats using STRIDE.
3. **Mitigation**: Improving the system security by providing ways to reduce the threats.
4. **Validation**: Checking if the threats have been mitigated.

Chapter 3

Threat Model

In this chapter, we create a threat model for an OSS system for provisioning and activation. We use the four-step framework described in Section 2.3 which involves diagramming, threat enumeration, mitigation and validation. We also identify the assets in a typical OSS system in Section 3.1 and discuss about adversaries in Section 3.2. The first two steps, diagramming and the resulting enumeration of threats are described in Section 3.3. The diagrams help to visualize the system and find out trust boundaries. As a result, finding out potential threats becomes easier. Once we know the threats and common vulnerabilities, we describe the next steps, mitigation and validation in Chapter 5.

3.1 Assets

An asset is anything that has some value [79]. The STRIDE methodology is not asset-centred but software-centred, which means that it focuses on the processes and working of the software in order to find the threats. Nevertheless, an asset analysis gives a broader perspective of what needs to be protected. Moreover, identifying critical assets helps in realising the importance of securing them. We have identified the following assets in an OSS system:

3.1.1 Information Assets

Subscriber Identity: There is a high amount of personally identifiable information (PII) that goes through an OSS system. Typically, this information includes customer names, street addresses, phone numbers, and email addresses. In some countries, CSPs are required by law to collect PII from their customers [22]. This information is valuable information for both the customer and the CSP. Any breach in this data is considered a serious security incident.

Subscriber Data: The data that is linked to a subscriber includes information such as details of subscriptions, billing, payment, and SIM card. Any tampering to this information may directly affect the revenue from the customer. OSS systems also carry sensitive authentication related information. Examples include, the parameters to configure an authentication server (e.g. RADIUS), and the GSM AuC parameters (e.g. KI). This sensitive information, if leaked, can cause serious damages.

Network Information: A P&A system stores a model of the network. This includes connection information such as usernames, passwords, cryptographic keys, connection types, and the network structure. A breach of this information will expose the network elements.

Provisioning Logic: The provisioning logic is a core component of a P&A system to generate tasks for the network elements. Any tampering of the logic will lead a faulty configuration of the network.

Northbound System Data: The BSS systems connect to the OSS system in order to send requests and create orders. The latter also sends responses to the former. OSS systems store connection information, e.g. credentials, reply-to addresses, and other information related to the northbound systems.

Catalog and Inventory Data: The product catalog and inventory data form a critical information store for P&A. Hence any tampering to this information will lead to disaster.

Archived Information: An OSS system is a system where information flows through it rather than getting stored for a long time. However, for several reasons including regulatory constraints, CSPs often archive the orders, requests and tasks data. This passive data can reveal some important information described in the above paragraphs.

User Data: The user database is used by the OSS system for authentication purposes to control access to monitoring and configuration functions.

This includes usernames, passwords, and phone numbers. This information can either be internal to the OSS system or retrieved from an Identity Provider (IdP).

3.1.2 Software Assets

Each component of the OSS system can itself be considered a software asset. This typically includes the following: P&A system (and subcomponents such as Network Manager, Logic Builder, internal queues, and its user interface), Catalog system, databases, IdP systems, and Inventory system.

3.1.3 Physical Assets

The physical assets that are of concern in an OSS system are the hosts on which the components are running. Furthermore, the network elements can also be considered as physical assets (e.g. the Nokia DX200 HLR system). Customer premise equipments such as modems and wired phones are also physical assets which are handled through the Field Service Management System.

3.1.4 Intangible Assets

The most important intangible asset for any CSP is its reputation. The customer loyalty of any telecommunications company depends on its reputation in the market. A company with a weak security policy and less regard for customer protection cannot last long in such a competitive market. Hence, CSPs focus on maintaining a good reputation in the market [22].

3.2 Adversaries

An Adversary (also interchangeably called an Attacker) is an entity with a malicious intent. Adversaries prevent the system from achieving its goals. Adversaries could steal, tamper, corrupt, spoof identity, deny service, and perform various other malicious activities. It is important to “know your enemies” while doing a security analysis. The following subsections explain different adversaries in the context of an OSS system.

3.2.1 External Adversaries

External adversaries are untrusted entities who lie outside the boundaries of the enterprise. External adversaries may be unknown to the enterprise, e.g. a hacker. A hacker on the Internet can try to attack the system. The intention of such attackers can include anything, such as trying to impress others, for fun, or compromising the system for breaking into other systems (i.e., making the system a part of a botnet). Examples include hacktivists, terrorists, and organized criminals.

One should not neglect the possibility of competitor CSPs (or partner CSPs) trying to attack the system remotely. Perhaps, partner CSPs trying to use the system in ways they are not permitted to use. The intent might not be malicious, but could well be to gain more from the system than permitted. Example, a partner CSP trying to gain more capacity than allotted by the host CSP.

The customers of a CSP can be considered as external adversaries. Customers may try to exploit vulnerabilities in the system to their advantage. For example, a customer may try to attack the system in order to get more data capacity for the price paid. In some cases, customers try to bypass restrictions such as blocked websites or daily call limits.

Government agencies can try to attack OSS systems in order to get customer data. For example, intelligence or spy agencies may try to obtain information by secretly attacking the system when the legal methods fail. Foreign governments may also try to hack a CSP's OSS system as a part of developing cyber-attack capabilities.

3.2.2 Internal Adversaries

Internal adversaries are entities that may be known to the enterprise and may work within the enterprise's boundary. Internal adversaries usually have legal access to the CSP's enterprise network. Insiders can also have permissions to use the OSS systems. Thus, it is easier for an insider to attack the system or just steal information because they can exploit the trust between them and the CSP or its systems. Several issues regarding insider threats have been examined by Colwill [26].

An employee can perform tasks on the system that might be considered a security violation. For example, an employee may copy subscriber information from the request parameters while monitoring requests on the P&A system. Or a curious employee modifying requests which could lead to a faulty configuration of the network. Some employees may be interested in finding out data about their friends and family. Employees stealing subscriber data (such as phone numbers) is a serious problem.

Another case of insiders who could exploit a trust relationship are contractors with malicious intent. CSPs may hire contractors or outsource full or part of OSS system's management to a separate company. In this case, the CSP has to trust an external entity which could turn malicious.

3.3 Diagramming and Threat Enumeration

A DFD (Data Flow Diagram) helps understand the flow of data across the whole system. Thus, we can find out the external and internal *boundaries* of the system. A trust boundary is where principals with different privileges interact [76]. A trust boundary presents an *attack surface* for malicious entities to gain access or interact with an entity [5]. A DFD contains entities, processes, data stores, and data flow as drawn in Figure 3.1.

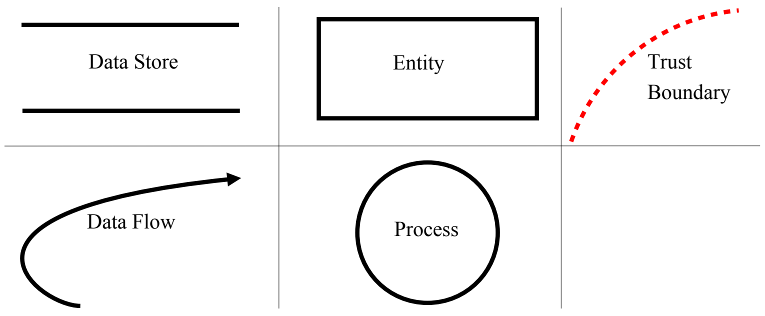


Figure 3.1: DFD Elements [76]

For enumerating threats, each element in the DFD is analysed for STRIDE threats using STRIDE-per-Element as described by Shostack [76]. According to STRIDE-per-Element, some threats apply only to certain elements of a diagram as per Table 3.1.

Table 3.1: STRIDE-per-Element

	S	**T**	**R**	**I**	**D**	**E**
External Entity	X		X			
Process	X	X	X	X	X	X
Data Flow		X		X	X	
Data Store		X	?	X	X	

We use Microsoft's threat modelling tool [16], which provides a simple and efficient way to create a DFD for a system. We use the top-down approach by first studying the high-level system also called the context diagram. Thereafter, we draw the level-1 diagrams which reveal more details of the system. These diagrams, along with associated threats, are described in subsequent Sections 3.3.1-3.3.4. Note that Appendix A contains the full list of threats while the above sections mention only the most important threats to consider.

3.3.1 Context Diagram (High Level)

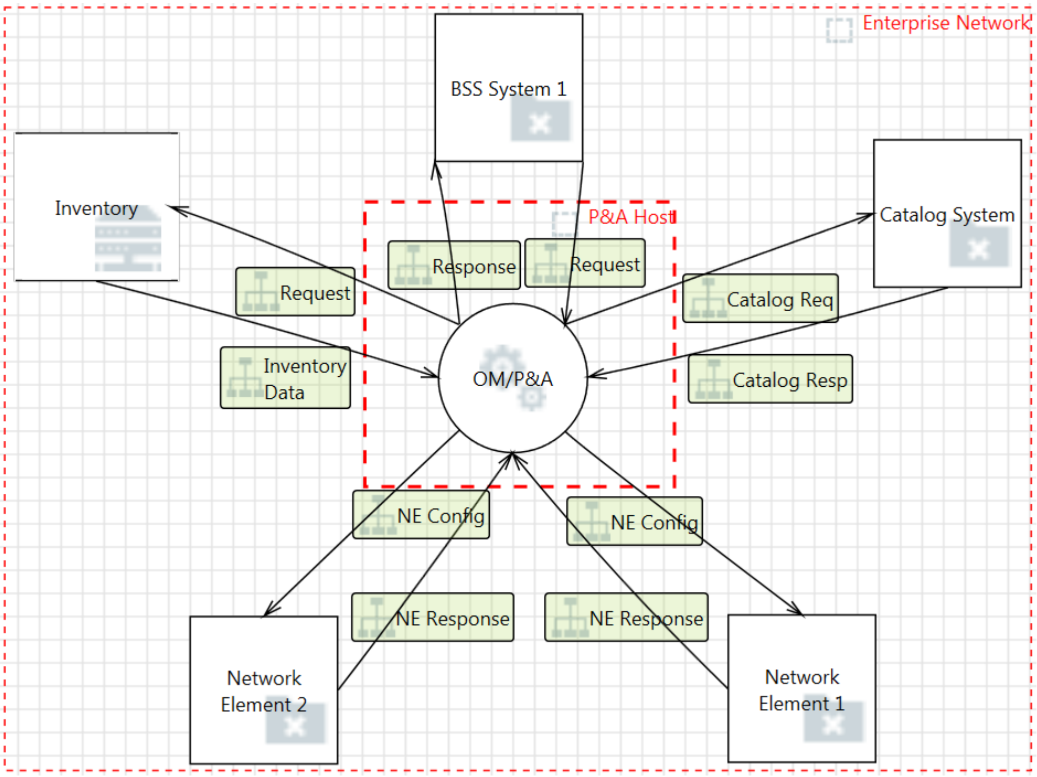


Figure 3.2: Context Diagram of a typical OSS

Figure 3.2 refers to the data flow between the BSS systems, the OSS system, and the NEs. Generally, all the systems are within a single enterprise-wide network, which forms a trust boundary area. The access to the enterprise network is controlled by security gateways and firewalls. A host trust boundary exists between the northbound BSS systems and P&A system since the communication is between two different hosts. Similarly, a host trust boundary is crossed when the P&A system sends configuration tasks to NEs and receives responses. Likewise, when the P&A system connects to other OSS components such as a Network Inventory system and Product Catalog system, which run on different hosts, the communication crosses a trust boundary. In a multi-host environment where OSS systems are replicated across multiple hosts and communicate with each other to distribute load and to synchronize data, the number of trust boundaries is even higher.

Based on the context DFD, Appendix A.1 lists the major threats. Following are the important threats to consider:

1. An attacker may spoof a BSS system or an external system such as Inventory or Catalog. A malicious system posing as a BSS can disrupt the network by sending incorrect configuration requests to the P&A system.

2. If the attacker is able to spoof a system, it can mount a man-in-the-middle (MitM) attack, which might lead to tampering of data or sniffing of important information on the interfaces. Alternately, the attacker might just passively listen to (sniff) the information on an interface.

3. An attacker may deny service to the BSS systems by flooding the P&A system with requests or by compromising the northbound interface.

3.3.2 User Interface

A user interface is used for configuration and monitoring. In general, authentication and authorization services are provided by an external authentication server using Kerberos, LDAP, RADIUS, and other protocols. In some cases, authentication and authorization are done locally. In our example, the user interface is served by a web application, which is a sub-component of the P&A system as shown in Figure 3.3. All the communication between the web application and the user's browser, the external authentication server, and the database server passes through a host trust boundary.

We have analysed the threats based on the level-1 DFD for the user interface (UI) in Appendix A.2. The important threats to consider are as follows:

1. Cross-site scripting (XSS) in the user's browser can lead to a number of security issues including elevation of privilege and information disclosure. The web application might not validate user input properly.

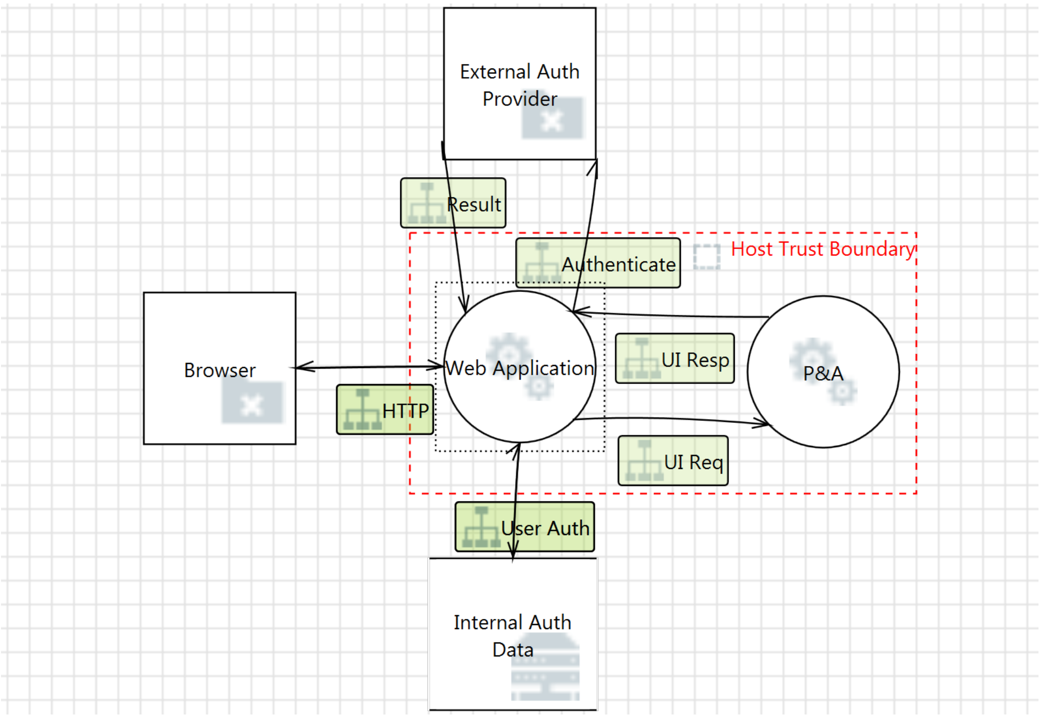


Figure 3.3: User Interface

2. A lack of input validation can also allow the attacker to inject SQL statements in the user input and to unauthorized information disclosure.

3. An attacker may listen to the interface between the browser and the web application and sniff valuable information such as passwords and personal information. MitM attacks can be mounted by attackers.

4. In the case of a weak authentication, the attacker can brute-force combinations of username and passwords. Weak access control can lead to unauthorized users getting access to protected information.

5. The communication between the identity provider and the web application is an important attack surface. An attacker could steal authentication tokens and session information by sniffing or by mounting a MitM attack.

3.3.3 Northbound Interface

A P&A system provides northbound interfaces for receiving provisioning and activation requests as well as for sending responses back to the BSS systems. Figure 3.4 shows different types of northbound interfaces and the data flow to the P&A system. There are several types of northbound interfaces generally supported including SOAP, REST, and CORBA. Some vendors have their own non-standard protocols over TCP. The responses can be synchronous or asynchronous. In our example, we assume that each interface is a separate component. Before a session is established, a BSS system is authenticated based on pre-stored information (key, password, etc.) in the data store. A trust boundary lies between the BSS system and the interface and also between P&A system and the data store.

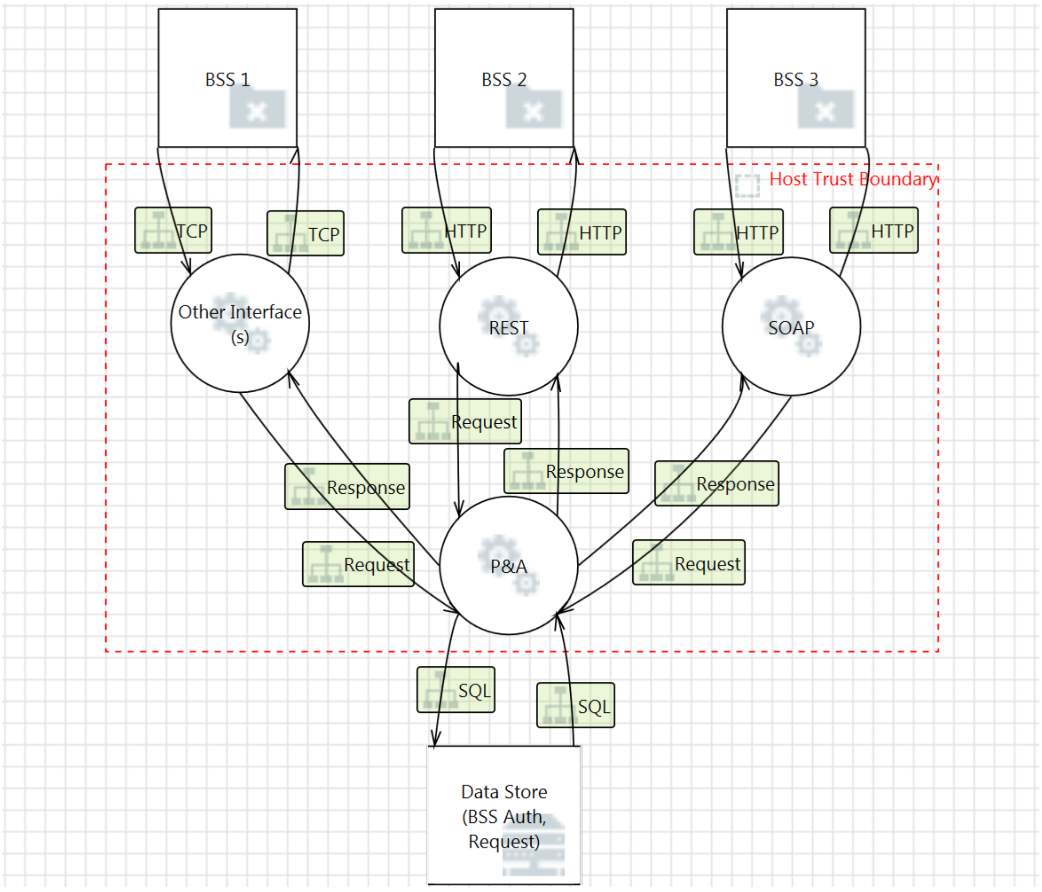


Figure 3.4: Northbound Interfaces

A full list of threats is enumerated in Appendix A.3. The important threats to be considered are as follows:

1. Spoofing of the P&A system by the attacker can lead to the BSS systems sending login credentials and configuration requests to an illegitimate system. On the other hand, spoofing of the BSS system can lead to a malicious entity breaching the trust boundary and sending malicious requests to the P&A system, or the P&A system sending responses to the malicious entity.

2. An attacker may be able to sniff the credentials used by the BSS systems to authenticate with P&A. Data transfer in plaintext is vulnerable to sniffing, which can reveal subscribers' personal information and secret network element parameters.

3. Vulnerabilities in the interface technology can be exploited by attackers. If the interface is SOAP-based, there are threats such as XML Injection, WSDL Scanning, and WS-Addressing spoofing [37]. Threats for REST-based interfaces include XML injection, token thefts, cross-site request forgery (CSRF), and SQL injection [61].

4. A weak authentication or authorization implementation in P&A can lead attackers to determine BSS credentials using brute force and dictionary attacks.

5. An attacker can deny service by using several techniques such as resource exhaustion by sending multiple login requests or a high number of provisioning requests to the interface.

6. An SQL injection attack or sending faulty data may result in data tampering, leading to wrong configuration, denial of service or even corruption of the data store.

3.3.4 Southbound Interface

The southbound interface connects the P&A system to the NEs. Additionally, Field Service Management Systems and ESB systems can also be considered NEs and connected to the southbound interface. Since there are various protocols and communication technologies for NEs, a Network Element Interface (NEI) is used. Different NEIs are used for Secure Shell (SSH), Telnet, TL1, and SOAP based NEs. The NEIs are a part of the P&A system and hence lie within the same host trust boundary.

Figure 3.5 shows two different interfaces and the connected network elements. The network model stores all the details of the network including keys and credentials used to establish a session with the NEs. The network model is queried by the P&A system for details required to connect to an NE.

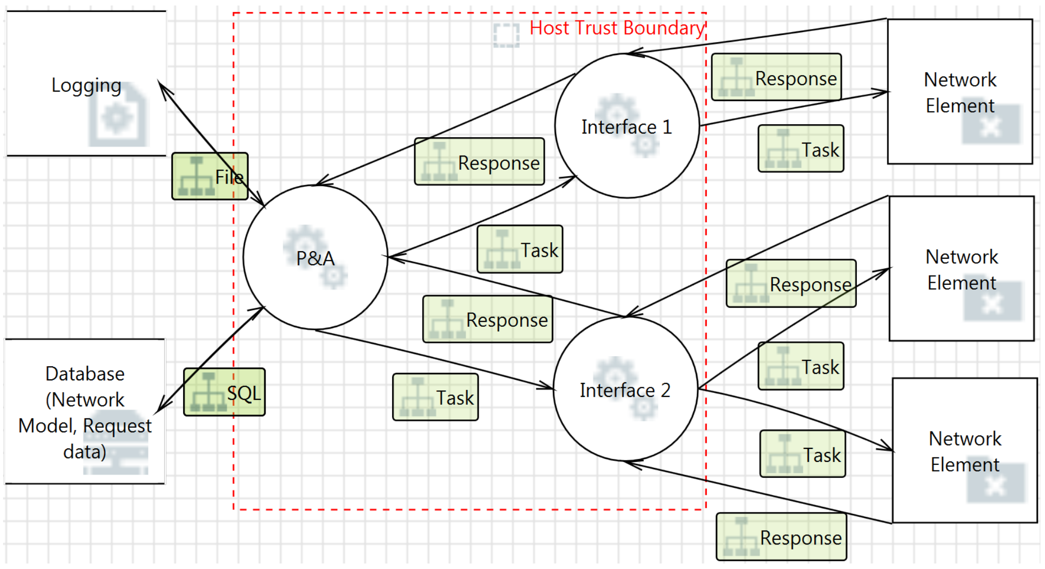


Figure 3.5: Southbound Interfaces

The threats generated upon an analysis of the DFD for southbound interfaces are listed in Appendix A.4. Following are the prominent threats to consider:

1. Similar to threats listed for other diagrams, sniffing of the data flow from the NEI to the NE is an important threat to consider. If the communication is not encrypted, an attacker might be able to deduce sensitive information including passwords and personal information by sniffing the P&A-NE interface.

2. Spoofing of an NE or the P&A system is a major threat. If the attacker is able to spoof an NE, then P&A system will send sensitive information to the illegitimate NE. If an attacker manages to get access to the P&A host and replace (or spoof) an NEI component, then it is possible to mount MitM attacks from the spoofed NEI.

3. If there is improper input validation, the attacker can use SQL injections in the request data to try to retrieve the credential information for NEs from the network model. If the authentication scheme is weak for the data store, malicious users can get hands on the network model data.

4. If the network model data is disclosed, the attackers can try to decrypt the NE credentials by several methods including brute force, rainbow tables and dictionary attacks.

5. A weak key management scheme can be exploited by attackers. If the encryption keys for encrypting the network model are stored on the same P&A host, malicious users might be able to get access to it.

6. A disclosure of NE logs can reveal a lot of NE related secret information as well as task information.

3.3.5 Provisioning and Activation

Figure 3.6 shows different components within a P&A system. The Request Module stores incoming requests to be processed in the R-Queue, while the Task Module stores the tasks to be sent to NEs in the T-Queue. The queues act as a buffer for the incoming requests. Apache ActiveMQ [1] is an example practical implementation of a message queue.

The Provisioning Logic Component is used to split the request into multiple tasks for several NEs depending on a stored logic. A data store is used to keep track of requests, tasks, and monitoring information. All the components except the BSS systems, database, and the NEs are within the trust boundary of the host on which the P&A system is installed.

Similar to other diagrams, most of the common STRIDE threats apply to this DFD as mentioned in Appendix A.5. Following are some of the important threats to note:

1. If an attacker is able to break into the P&A host, they might spoof the queues in order to get access to the request and task messages.

2. Components may be inadvertently exposed to outside the host because of misconfiguration of firewalls. For example, the ports of R-Queue and T-Queue exposed to the network. Attackers can mount denial of service attacks on the ports.

3. An attacker could tamper provisioning logics in order to disrupt the proper creation of task, eventually leading to improper network configuration.

4. If the request and NE logs are not maintained properly, it is difficult to address repudiation claims.

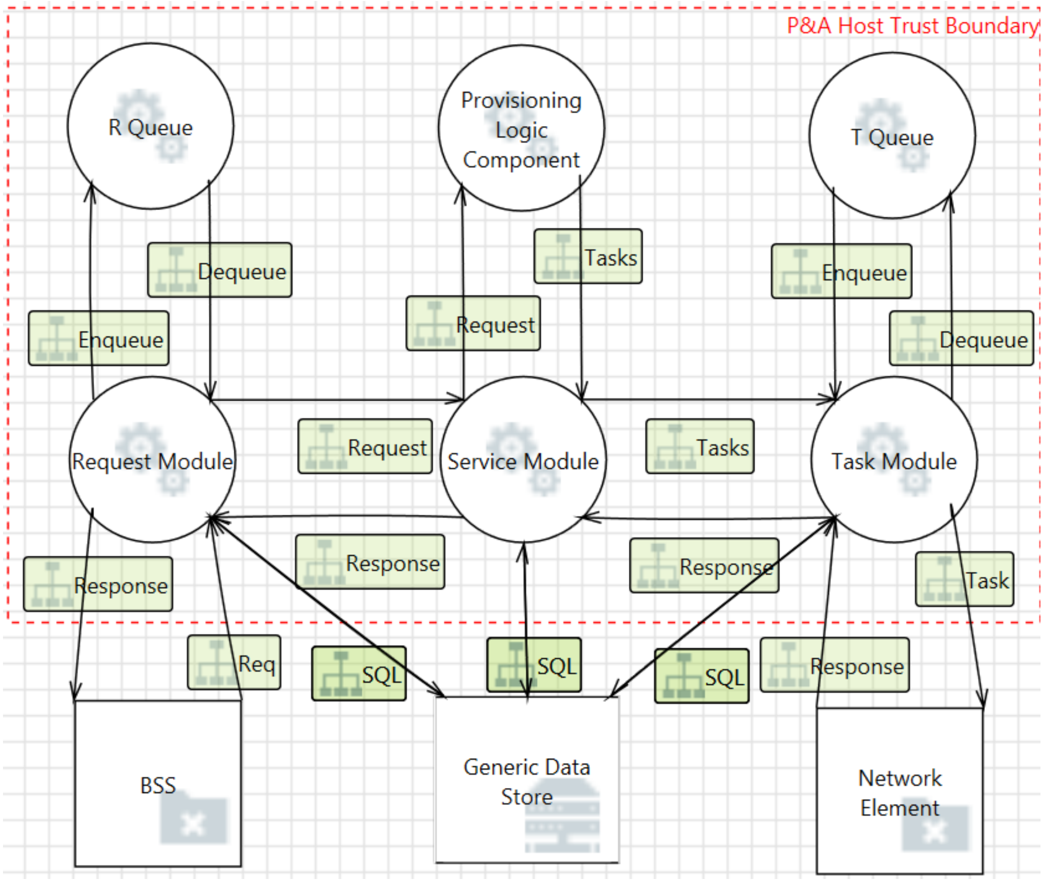


Figure 3.6: Provisioning and Activation

5. Insecure access to logging data can disclose classified information. The request log might contain important metadata for NEs as well as private subscriber information.

Chapter 4

Vulnerabilities and Security Risks

In this chapter, we describe vulnerabilities that are common to P&A and OSS systems in general. Most of the OSS systems serve their UI using a web application and use components such as ActiveMQ [1] for queuing messages. The application also opens several ports for accepting requests, communication between components, sending responses, providing a UI, etc. Taking into account these common components, the subsequent sections explain the various vulnerabilities that vendors must address in their products. Note that the examples and figures mentioned in this chapter may have some vendor-sensitive information removed considering responsible disclosure.

4.1 Default Username and Password

OSS products are usually shipped with a default administrator user with credentials, which are easy to guess such as, admin/admin or administrator/admin. This makes it easy for attackers to use dictionary attacks on the administrator account. Often, the product documentation itself mentions the default administrator credentials. Thus, in this case, it is trivial for an attacker to obtain the default password. A default username and password is convenient to have during the initial setup of the application. However, products often do not automatically prompt the user to change the password after the first login. Thus, it is possible for the administrator account to remain active with a default password. If the administrator account is compromised, it could lead to situations such as the attacker modifying provisioning logics secretly, changing the network configuration, stealing information, creating new users, or even denying service.

4.2 Poor Key Management

4.2.1 Using pre-generated Keys

During the development phases, the developers might end up pushing the keys generated for testing to the installation package for convenience. This includes keys that are used for data encryption or transport layer security (TLS). As a result, many OSS products do not generate new keys for each installation and the same keys from the code repository are shipped in every installation package. The product documentation usually mentions generating new keys manually after the installation is completed.

Except for the convenience that this practice provides, this is a serious vulnerability. If an attacker gets access to one of the product installations, then every installation is compromised. As a result, the attacker can perform malicious activities on the processes where the keys are used including the decryption of subscriber information and network element passwords. Addressing such a scenario can require a lot of work and can generate multiple customer cases resulting in a chaotic situation. A similar vulnerability is mentioned by Cisco regarding default SSH keys [29].

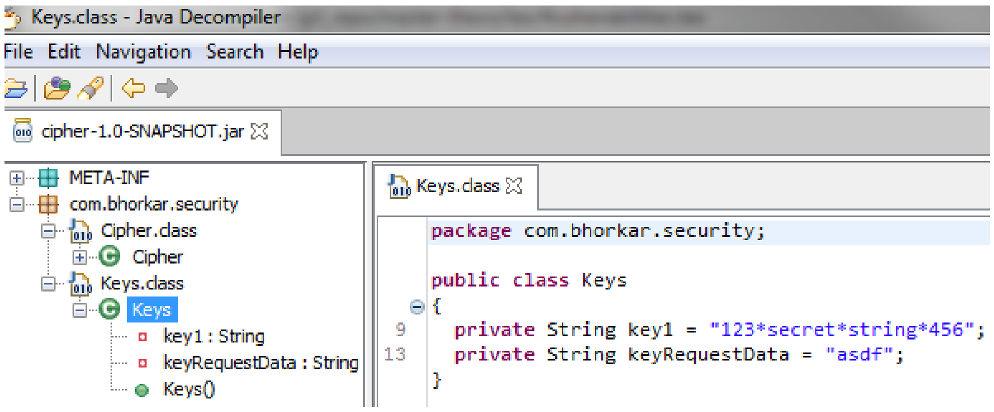
4.2.2 Keys Hard-coded in the Code

For data encryption (of passwords, subscriber keys, etc.), developers often use symmetric encryption such as AES or Triple-DES. The keys are often stored in the code as `String` variables.

Storing keys as `String` variables is a serious vulnerability since an attacker can decompile the executable code to reveal the values of the variables. Even if the executable code is obfuscated, it is possible for an attacker to retrieve the value of the required `String`, given the time and resources. There are several tools available to decompile the executable code.

Figure 4.1 shows an example of decompiled Java code to reveal the key string using the JD tool [6] by simply opening the `.jar` file in the decompiler. It is also possible to reveal the key strings by running the executable in a debugger such as `jdb`.

Since the keys are hard-coded in the code, every installation of the application has the same keys. Thus, the attacker needs to compromise only one installation in order to retrieve the key data to compromise all the installations.



```
package com.bhorkar.security;

public class Keys
{
  private String key1 = "123*secret*string*456";
  private String keyRequestData = "asdf";
}
```

Figure 4.1: `String` values revealed using Java Decompiler

4.3 Passwords Stored in Plaintext

Many components within the OSS system require authentication. For example, ActiveMQ [1] authenticates clients before brokering the messages to other clients. We found that organizations tend to use simple configurations available for external components. In case of ActiveMQ, when the simple authentication configuration is used, the client usernames and passwords are stored in the settings file (activemq.xml) in plaintext. This is a security flaw.

Another case of a bad security design is the storage of NE credentials in plaintext. NE passwords stored in the database or the file system in plaintext is a high security risk.

4.4 Insecure Authentication Practices

Encrypting user passwords for authenticating users was a common practice during the early years of the World Wide Web. However, storing passwords in an encrypted format is not recommended as it would lead to massive information disclosure if an attacker gets access to the encryption key. Instead of storing passwords in an encrypted format, the recommended way is to store the hash value of the passwords. However, passwords not stored in a salted hash format are vulnerable to dictionary attacks [43]. This applies to passwords used for authenticating users as well as the northbound client systems in an OSS system. Since OSS systems have been around for a long time, legacy authentication practices might still be in use.

4.5 Use of Weak Cryptographic Algorithms

We found that OSS applications may use weak cryptography in their components. According to a NIST report [20], several encryption and hashing algorithms are considered insecure. We found that the DES and MD5 algorithms are still used for encryption and hashing purposes respectively. These algorithms are considered inadequate for modern security requirements. A weak encryption algorithm is a critical security problem in an OSS system since the encrypted data includes sensitive information.

4.6 Open Ports on the OSS Host

Ports that are open to the network can attract unnecessary attention from malicious users. The more ports are open on a host, the wider the attack surface. A deep port scan using Nmap [11] shows a list of open ports. Attackers can exploit vulnerabilities in the services running under the ports or may mount attacks to deny services to legitimate users.

4.6.1 SSH Port

An open SSH port accessible from a bigger network or the Internet leads to a variety of different attacks. Attackers can try to brute force login credentials in order to guess the root or some other user's password. A study by Owens et al. [67] determines that SSH servers are the target of a variety of brute force attacks.

Another risk is to allow login for any user including the root user. This can lead to disastrous consequences since a compromised root user will allow administrator access to the attacker. Owens et al. [67] mention that the root user was targeted in 25% of all the malicious login attempts in their experiments because *root* is always a valid username on Linux hosts.

4.6.2 Other Ports

An OSS system has multiple components. A typical P&A system opens ports for external systems as well as internal components. Based on the P&A components described in Section 3.3.5, a few use cases for opening ports are:

1. The request module may open two ports for external systems, one for accepting requests and the other for sending responses.

2. A management port for each component (Network, Service modules)
3. Ports for communicating with R-Queue and T-Queue

An NMap port scan of a host will show many open ports as shown in Figure 4.6.2. From the output, we can determine that along with the ssh port, there are several open TCP ports which are used for various purposes. The attack surface is increased if the ports for internal communication are made visible on the network. An attacker may try to connect to each of the ports and exploit vulnerabilities associated with the respective service.

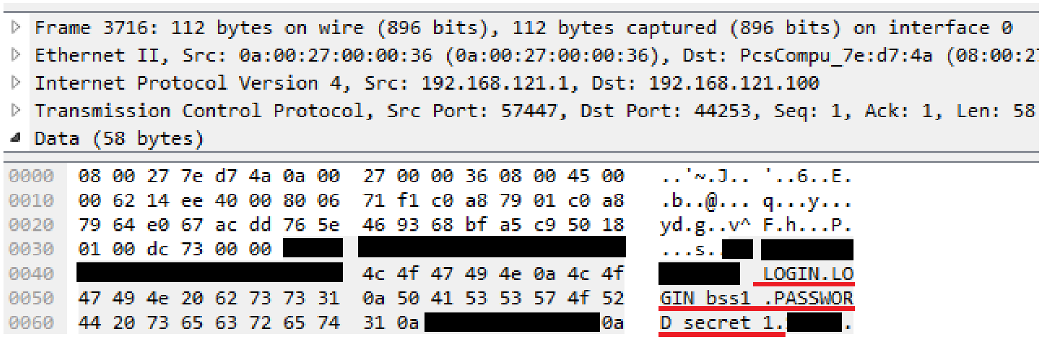
```
[user@host ~]$ sudo nmap -p- 192.168.122.1
Starting Nmap 6.40 ( http://nmap.org ) at 2017-08-15 15:48 +02
Nmap scan report for 192.168.122.1
Host is up (0.0000060s latency).
Not shown: 65480 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
53/tcp    open  domain
111/tcp   open  rpcbind
44250/tcp open  unknown
44253/tcp open  unknown
44254/tcp open  unknown
44255/tcp open  unknown
44256/tcp open  unknown
44257/tcp open  unknown
...
Nmap done: 1 IP address (1 host up) scanned in 1.16 seconds
```

Figure 4.2: NMap Port Scan for a P&A host

4.7 Insecure BSS-OSS Interface

As described in the threat model (Section 3.3.3), the northbound interface faces threats such as sniffing, spoofing, and denial of service (DoS). If the link between the BSS and OSS systems is not encrypted, an attacker can sniff data. An attacker can deliver a malicious packet sniffer to the target BSS or the OSS host by several means such as via an email attachment, using existing remote code execution vulnerabilities, or insider knowledge.

We used a Wireshark [15] instance running on a BSS system to sniff data packets sent to the P&A system in order to replicate the same scenario. As shown in Figure 4.3, we successfully retrieved the login credentials used by the BSS system. Consequently, the credentials can be used to impersonate the BSS system.



```
▷ Frame 3716: 112 bytes on wire (896 bits), 112 bytes captured (896 bits) on interface 0
▷ Ethernet II, Src: 0a:00:27:00:00:36 (0a:00:27:00:00:36), Dst: PcsCompu_7e:d7:4a (08:00:2
▷ Internet Protocol Version 4, Src: 192.168.121.1, Dst: 192.168.121.100
▷ Transmission Control Protocol, Src Port: 57447, Dst Port: 44253, Seq: 1, Ack: 1, Len: 58
◢ Data (58 bytes)

0000  08 00 27 7e d7 4a 0a 00  27 00 00 36 08 00 45 00   ..'~.J.. '..6..E.
0010  00 62 14 ee 40 00 80 06  71 f1 c0 a8 79 01 c0 a8   .b..@... q...y...
0020  79 64 e0 67 ac dd 76 5e  46 93 68 bf a5 c9 50 18   yd.g..v^ F.h...P.
0030  01 00 dc 73 00 00 [redacted] [redacted]            ...s.[redacted] [redacted]
0040  [redacted]               4c 4f 47 49 4e 0a 4c 4f   [redacted] LOGIN.LO
0050  47 49 4e 20 62 73 73 31  0a 50 41 53 53 57 4f 52   GIN bss1 .PASSWOR
0060  44 20 73 65 63 72 65 74  31 0a [redacted]      0a   D secret 1.[redacted].
```

Figure 4.3: Sniffing BSS credentials using Wireshark

Attackers have several techniques at their disposal for a MitM attack including ARP poisoning, and DNS spoofing. Ornaghi et. al. [57] describe MitM attacks such as key manipulation, downgrading attack, and command injection. The example described in Section 4.8 can also be used to remotely sniff the BSS-OSS interface.

Moreover, the BSS-OSS interfaces are implemented in a variety of different technologies such as REST and SOAP. Vulnerabilities in REST and SOAP implementations may allow remote code execution, SQL injection, and other attacks. For SOAP services, there are several attacks such as WSDL spoofing, Access control bypass, and XPath injection [36, 87]. REST-based web services are prone to attacks such as Token theft, DDoS, and Buffer Overflows. References [37] and [61] describe attacks and mitigation for SOAP and REST interfaces, respectively. Demonstration of each of these attacks is out of the scope of this thesis.

4.8 Insecure Browser-OSS Interface

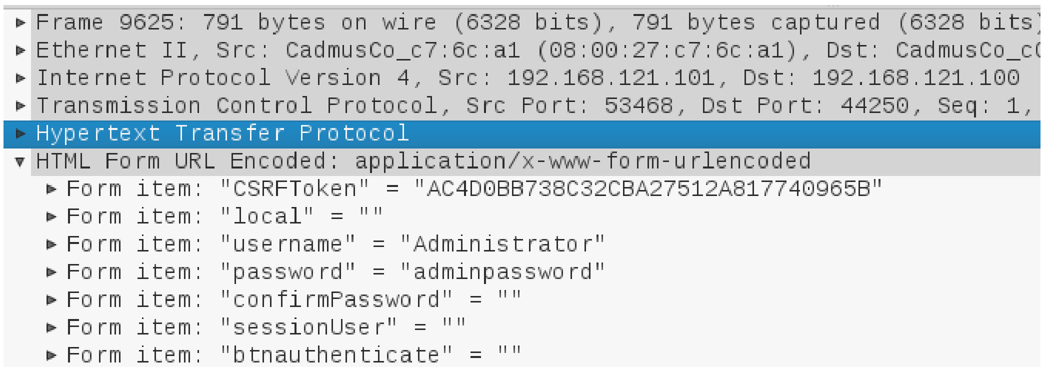
The Browser-OSS interface is vulnerable to the same attacks as mentioned in Section 4.7 with respect to the BSS-OSS interface. In this section, we demonstrate sniffing the login credentials of a user from a remote host in the network as opposed to sniffing on one of the communicating hosts. If the browser does not use HTTPS to communicate with the server, it gives the attacker a chance to sniff the connection. We used ARP spoofing [86] to poison the ARP cache with the MAC address of a malicious host (for sniffing).

Thus all the traffic from the web browser intended for the target P&A host passes through the malicious host. We followed the following process:

1. Set up a malicious host in the same network as the target host (web browser).

2. Start ARP spoofing using the Ettercap tool [4] to poison the ARP cache both ways, that is, for the target host and the P&A system.

3. Start Wireshark [15] to sniff packets that flow through the malicious host.



```
▸ Frame 9625: 791 bytes on wire (6328 bits), 791 bytes captured (6328 bits)
▸ Ethernet II, Src: CadmusCo_c7:6c:a1 (08:00:27:c7:6c:a1), Dst: CadmusCo_c
▸ Internet Protocol Version 4, Src: 192.168.121.101, Dst: 192.168.121.100
▸ Transmission Control Protocol, Src Port: 53468, Dst Port: 44250, Seq: 1,
▸ Hypertext Transfer Protocol
▾ HTML Form URL Encoded: application/x-www-form-urlencoded
  ▸ Form item: "CSRFToken" = "AC4D0BB738C32CBA27512A817740965B"
  ▸ Form item: "local" = ""
  ▸ Form item: "username" = "Administrator"
  ▸ Form item: "password" = "adminpassword"
  ▸ Form item: "confirmPassword" = ""
  ▸ Form item: "sessionUser" = ""
  ▸ Form item: "btnauthenticate" = ""
```

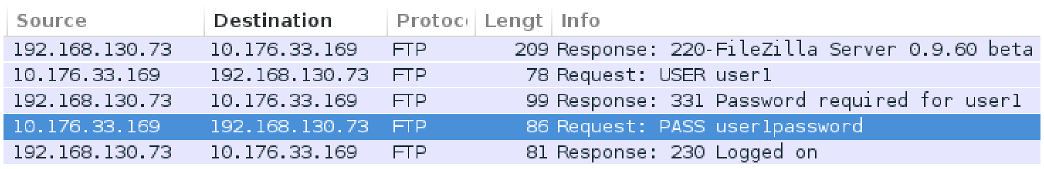
Figure 4.4: Sniffing User credentials with Wireshark

Figure 4.4 shows the sniffed credentials for an administrator user captured from a malicious host on the same network as the user's host.

If the communication is HTTPS, attackers can use protocol downgrade attacks to downgrade the encryption protocol to a weaker version or to clear text communication. One such example is SSLStrip [49] where the man-in-the-middle replaces HTTPS links with HTTP so that sniffing and tampering is possible. Marlinspike [50] describes several tricks to defeat SSL using SSLStrip.

4.9 Insecure OSS-NE Interface

There are myriad of southbound interfaces used by a P&A system. The security of each interface depends on the type of interface, manufacturer, etc. Newer interfaces such as SOAP, REST over TLS, and SSH are more secure than older interfaces. One of the older interfaces still in use is TL1 over Telnet, which uses plaintext communication. The same technique as described in Section 4.8 can be used to sniff the interactions between the P&A system and the network element. Figure 4.5 shows a snapshot of login credentials sniffed from an FTP-based interface (for NEs which accept file uploads). The sniffing was possible since the NEI did not use transport layer security and sent the credentials in plaintext. In some cases, the login credentials are encrypted but the rest of the communication is plaintext, which might contain sensitive information. Southbound interfaces that use SNMP to configure network elements can be compromised by exploiting vulnerabilities in the SNMP protocol such as insecure settings, spoofing, and other vulnerabilities in SNMP's trap and request handling processes [38].



Source	Destination	Protoc	Lengt	Info
192.168.130.73	10.176.33.169	FTP	209	Response: 220-FileZilla Server 0.9.60 beta
10.176.33.169	192.168.130.73	FTP	78	Request: USER user1
192.168.130.73	10.176.33.169	FTP	99	Response: 331 Password required for user1
10.176.33.169	192.168.130.73	FTP	86	Request: PASS user1password
192.168.130.73	10.176.33.169	FTP	81	Response: 230 Logged on

Figure 4.5: Wireshark capture of a P&A and an FTP based NE session

4.10 Sensitive Data Exposure on the UI

As described in Section 3.1.1, there are multiple information assets in an OSS system. Some information assets are considered very sensitive. An example of critical information is the encryption key which is configured in the AuC system in the GSM core network infrastructure as well as in the SIM card. The encryption key must be kept secret in order to prevent SIM cloning. P&A requests for the configuration of the AuC system often contain a KI parameter which carries the encryption key. OSS developers might not hide the KI parameter while displaying the request data on the UI, thereby leading to disclosure of sensitive data to unintended users.

Another case of sensitive data exposure is the display of NE credentials on the UI. Once an NE's connection details are created on the UI, the credentials are typically stored in the database in an encrypted form. NE passwords are stored because they are used later while establishing communication with NEs. However, the credentials might be shown on the UI in plaintext while viewing or modifying the connection details. This vulnerability may be prevalent in many OSS systems because of insecure design or development practices.

4.11 Logging of Sensitive Data

OSS systems log high amounts of data, which includes request logs, provisioning logic execution logs, user audit trails, NE logs, and component logs. Apart from helping in finding faults in the application, logging also helps in finding security breaches. Nevertheless, the high amount of data that goes into OSS logs makes logging an important asset to secure. Figure 4.6 shows sensitive data such as credentials and personal information found in logs of a P&A system. Logs are often archived in insecure network storage. Attackers can use a combination of attacks to get access to logs. An attack scenario could include the use of an XSS or a CSRF attack to steal the log data displayed on the UI. If there is a directory traversal vulnerability in the UI, attackers can exploit it view log files.

```
$BEGIN$ 1_1
Task started at 20170829-160904
Parameters:
task_type = create
product_name = DX200
ne_type = HLR
ne_id = hlr1
MSISDN1 = 358991234567890
...
KI = 94764B49AF37AC9D21B0EEEFEFD65C6E
...
$END$ 1_1
```

Figure 4.6: Network component logs revealing a sensitive parameter (KI)

4.12 Web Application Security Risks

As mentioned in Section 3.3.2, the UI component of many OSS systems is a web application which communicates with other P&A components. If the UI component is compromised, an attacker can compromise other components by misconfiguring the system or creating backdoors. The following list describes the top 10 security risks that plague web applications as per OWASP [65]. We have appended the list with OSS related examples.

1. **Injection**. An SQL injection sent from an input field or as a URL parameter can reveal, corrupt or modify sensitive data.

2. **Broken authentication and session management**. A session hijack is possible if session IDs are managed improperly such as putting session IDs in the URL or non-HTTP-only cookies, using same session IDs, or not timing out sessions. The authentication may be broken if passwords are easily guessable and default passwords are not changed (see Section 4.1).

3. **Cross-Site scripting (XSS)**. If the application does improper input validation, an attacker can insert malicious JavaScript code in the input field. An XSS attack could be reflected or persistent depending on where the input data is used. In a persistent attack, we were able to send a malicious JavaScript code the login page which got stored in the login audit trail. When an administrator viewed the audit trail, the malicious JavaScript to steal the session ID was executed.

4. **Insecure direct object references**. In this case, a user with malicious intent is able to view unauthorised objects by directly referencing them. For example, in an OSS system, a user with regular privileges might be able to view request data that only an administrator should be able to view because of lack of proper control.

5. **Security misconfiguration**. Wrongly configuring security can lead to the system being compromised even without the maintainers noticing it. Allowing weak ciphers in the server's SSL/TLS configuration is one example [69].

6. **Sensitive data exposure**. For example, the sensitive data exposure in UI and logs described in Sections 4.10 and 4.11 respectively. Data backups are also quite vulnerable to exposure since databases and files including logs are routinely backed up. Also consider the interfaces which are vulnerable to sniffing (see Section 4.8).

7. **Missing function level access control**. This risk is a result of incorrectly implementing access control. For example, the privileged functionality is hidden on the UI but the server doesn't check for privileges. Thus, an attacker can gain access by calling the server API, or the URL, and thus bypassing the UI.

8. **Cross-Site request forgery (CSRF)**. A CSRF attack exploits the trust that the server has on the browser. Consider the scenario where an attacker sends a carefully crafted phishing email message to the P&A administrator with a link that creates a new BSS user. If the

administrator clicks the link when already logged in to the P&A system, a new BSS user will be created with the administrator's session.

9. **Using components with known vulnerabilities**. OSS developers often do not use the latest versions of third-party components to maintain compatibility. For example, using older versions of ActiveMQ vulnerable to many JMS-related attacks [40].

10. **Unvalidated redirects and forwards**. Attackers can redirect innocent users to malicious websites if redirects and forwards are not validated. For example, an attacker crafts a URL to forward the user to a malicious URL which looks similar to the OSS system's UI and asks for passwords, such as,
`http://osshost.company.com/redirect.jsp?next=evilhost.com`

4.13 Denial of Service

An attacker can mount denial of service attacks on an OSS system via the northbound interface, the UI or by exploiting the host's open ports (Section 4.6). A common way of denying service is to overload the application causing buffer overflows and crashing a component [85]. Examples of DoS attacks include the Syn flooding attack [27], and HTTP request flooding [88]. DoS attacks rose by 28% in Q2 2017 [17].

We were successfully able to block the propriety northbound interface of a P&A system by flooding the request port with a high number of connection requests using the `tcpflood` tool [14]. The output shows a high number of ESTABLISHED TCP connections on the P&A host.

```
[pna@host ~]$ sudo netstat -tpn | grep 192.168.121.100
tcp  192.168.121.100:44253 192.168.121.102:45734 SYN_RECV
tcp6 192.168.121.100:44253 192.168.121.102:45576 ESTABLISHED
tcp6 192.168.121.100:44253 192.168.121.102:45490 ESTABLISHED
tcp6 192.168.121.100:44253 192.168.121.102:45538 ESTABLISHED
tcp6 192.168.121.100:44253 192.168.121.102:45624 ESTABLISHED
tcp6 192.168.121.100:44253 192.168.121.102:45650 ESTABLISHED
tcp6 192.168.121.100:44253 192.168.121.102:45822 ESTABLISHED
tcp6 192.168.121.100:44253 192.168.121.102:45762 ESTABLISHED
tcp6 192.168.121.100:44253 192.168.121.102:45542 ESTABLISHED
...
```

DoS attacks can also be targeted at the application layer instead of the transport layer of the TCP/IP protocol. For example, in the Slow Read DoS attack [68], the attacker sends an unusually small receiving window size (usually zero) to receive data slowly from the server and holds up the connection and resources. We were successfully able to deny service on the Apache Tomcat Web Server [2] with the SlowHTTPTest tool [13] which uses techniques similar to Slow Read DoS. Attackers can exploit these techniques to deny service to the UI instance as well as HTTP based northbound interfaces.

4.14 Vulnerabilities in Message Queues

Message queues form an essential part of P&A and other OSS systems. In a P&A system, a message queue is used to buffer incoming requests from the northbound API as well as outgoing tasks for the NE. As described in Section 3.3.5, message queues are separate components, and the P&A system communicates with them via ports for sending and receiving messages. Kaiser [40] explained vulnerabilities related to Java deserialization in JMS implementations such as ActiveMQ [1]. In this section, we will describe some vulnerabilities in ActiveMQ [1] which is one of the most popular JMS implementations.

A misconfiguration can enable the ActiveMQ Web Console in production environments. The Web Console is used for monitoring ActiveMQ. We were able to log in to a P&A system's ActiveMQ component through the Web Console using the unchanged default administrator credentials and get access to queue data and administrative controls. Moreover, ActiveMQ versions before version 5.8.0 did not require any authentication (refer CVE-2013-3060 in NVD [10]).

Kalra [41, 42] analysed multiple vulnerabilities related to ActiveMQ. In its basic authentication scheme, ActiveMQ requires the username and password to be stored in the configuration file in plaintext (see Section 4.3) and provides no account lockouts for failed logins, thus being vulnerable to password guessing attacks. Several other vulnerabilities related to ActiveMQ include XSS in the Web Console (refer CVE-2016-0782 in NVD [10]), and directory traversal in file upload for blob messages (refer CVE-2015-1830 in NVD [10]). Many of the vulnerabilities have been fixed in the latest versions (>5.15.0). However, OSS implementations may continue to use older versions of ActiveMQ to maintain compatibility.

4.15 Tampering of Provisioning Logic

In a P&A system, the provisioning logic is a crucial component which determines the proper processing of requests based on pre-defined conditions (see Section 3.3.5). There are several provisioning logics in a P&A system. The system refers to the logic rules and the request parameters to run the required logic on the request. The logic then dictates the workflow of the request and the creation of sub-tasks for network configuration. P&A vendors implement the provisioning logic component differently. Some are template-based, while others are flow-based systems [54]. The impact of a compromised provisioning logic can be adverse and the detection is difficult. Usually, the provisioning logic is stored in the file system and the logic rules are stored in the database.

Vulnerabilities in other components can be exploited to tamper the provisioning logic. If there is a directory traversal vulnerability in the UI application, attackers can use it to reveal the logic. In case of improper validation for file uploads, an attacker can create a malicious path to point to a tampered logic or replace a logic with a tampered one. Logics which are stored in the database are vulnerable to SQL injection in the fields which are not validated properly. In other cases, tampering of the provisioning logic could be possible because of improper implementation of authorization policies. A non-administrative user may bypass authorization conditions and modify the provisioning logic.

Chapter 5

Mitigation and Validation

In this chapter, we propose and discuss mitigations for our threat model and the common vulnerabilities discussed in the previous chapters. This chapter can also be considered as a checklist for auditing an OSS system's security configuration. When determining mitigation methods, we consider the host trust boundary since all the components of an OSS application run within a host. In addition, we consider the entry and exit points, that is, the northbound interfaces, user interface, interfaces with other hosts, and the interfaces to network elements. The OSS system and its entry and exit points form the attack surface [47].

We describe the concept of layered defence and the standard STRIDE mitigation in Section 5.1. Thereafter, we analyse the entry and exit points in Section 5.2. We also discuss logging security in Section 5.3 and host security in Section 5.4. Section 5.5 discusses validation. Finally, we mention dos and don'ts for OSS professionals in Section 5.6.

5.1 Mitigation Techniques

5.1.1 Onion Model

Defense in Depth has been a military strategy since the age of the Romans [23]. The basic concept of Defense in Depth or the Onion Model is to have multiple layers of security instead of a single layer. The attacker has to break several layers of security to reach the target. Applying the same approach to OSS systems, security mechanisms are implemented at each of the network, host, application, and data levels as demonstrated in Figure 5.1.

Telecommunications companies often put more focus on perimeter security, that is, securing the borders of the enterprise network. Security of the

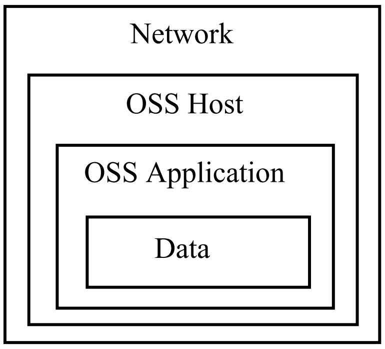


Figure 5.1: Onion Model

host and the applications is often a secondary concern. In case of a single-layered approach, an OSS application is exposed to the attacker as soon as the security boundary is breached. A layered security approach prevents such attacks. Therefore, the following layers of security are important for an OSS system.

1. **Data Security**: Data storage must be protected and any access must be authenticated and authorized.

2. **Application Security**: Any OSS product must have proper authentication and authorization. Mitigations for STRIDE threats must be in place for entry and exit points.

3. **Host Security**: The host on which OSS applications are installed must be security hardened. This means having a properly setup firewall and access control.

4. **Network Security**: Perimeter security should still be present. Only legitimate users and devices should get access to the network.

5.1.2 STRIDE Mitigation

Table 5.1 describes the standard mitigations applicable to STRIDE threats suggested by Meier et al. [52]. The following paragraphs mention the techniques that must be necessarily implemented while building an OSS application. These techniques will be elaborated in subsequent sections depending on the context.

Controlling access to the OSS system is a critical security requirement. Access should be controlled for both UI users and the northbound BSS systems. Authentication can be based on username-password, certificates, or

Table 5.1: Standard Stride Mitigation [52]

Threat	Countermeasures
Spoofing user identity	Strong authentication. Do not store secrets in plaintext. Do not transmit credentials in plaintext. Protect authentication and session cookies.
Tampering with data	Data hashing and signing. Digital signatures. Strong authorization. Use of tamper-resistant protocols. Use of communication protocols that provide message integrity.
Repudiation	Secure audit trails. Digital signatures. Timestamps.
Information disclosure	Strong authorization. Strong encryption. Use of communication protocols that provide message confidentiality. Do not store secrets in plaintext.
Denial of service	Resource and bandwidth throttling. Input filtering and validation.
Elevation of privilege	The principle of least privilege.

API keys depending on the context. The delivery personnel must verify that the authentication is set up properly. Authentication for UI users and BSS systems is described in Section 5.2 in detail.

Authorization is important for access control. Both UI users and BSS systems must be subject to access control by categorising them. Management of UI users and BSS systems should be separate. The principle of least privilege should always be followed.

Data flow that crosses a trust boundary should always go through a secure channel protected with latest security standards. For example, use SSL/TLS ($\geq$ TLS 1.0) for transmitting data over TCP.

OSS systems must use adequate logging for addressing repudiation claims while preventing sensitive data exposure. Section 5.3 explains considerations for logging data.

The use of firewalls to filter incoming traffic helps to harden the OSS host's security and reduces the attack surface. Host security is discussed in Section 5.4.

5.2 Interfaces

In this section, we inspect the different security techniques used in various interfaces of the OSS landscape and provide recommendations. As explained in Chapter 3, the main interfaces are northbound, southbound, user, and inter-component interfaces.

5.2.1 Northbound Interfaces

BSS systems send requests to the northbound interface for network configuration (see Section 3.3.3). Responses are sent back to the BSS systems synchronously or asynchronously. We have the following general recommendations for northbound interfaces:

1. Authenticate and authorize BSS systems. An access control for BSS systems is recommended and can be based on the type of requests. For example, a category to allow only 'read' requests for a BSS system that is used for monitoring purposes. Thus the concept of least privilege [73] can be implemented with an extra layer of control.

2. A username-password based authentication must be avoided. If it is used then it is important to encrypt the wire with a sufficiently secure algorithm (e.g. $\geq$ TLS 1.0). Passwords should be sufficiently long and randomly generated and should have an expiry time. Instead, token based or a certificate-based authentication is recommended.

3. A certificate-based mutual authentication is recommended to prevent spoofing. Both the BSS and OSS system must authenticate each other.

4. The communication channel of the interface must always be encrypted using protocols that support message integrity to prevent sniffing and tampering.

5. For asynchronous responses, the reply-to address can be suspicious and must be validated properly. The asynchronous response channel must be encrypted. Furthermore, the OSS system must enforce that the

reply-to address specified in a request by the BSS system must be already configured in the OSS system. This can ensure that responses only get sent to trusted response handler systems.

6. The BSS system should authenticate asynchronous responses coming from the OSS system in order to prevent unauthorized responses being submitted by malicious systems.

7. The freshness of both requests and responses must be checked using nonces or sequence numbers.

The subsequent subsections describe security features that are recommended based on the technology used by a northbound interface.

5.2.1.1 SOAP Web Service

SOAP web services use the XML based SOAP protocol for messaging and HTTP for transporting messages. The service description is published using WSDL in a `.wsdl` file on the server. Most SOAP web service implementations follow the WS-Security [56] standard for providing secure functionality. The most common web service frameworks include Apache Axis2, Metro, and Apache CXF which provide support for the WS-Security standard [78].

WS-Security provides end-to-end security which protects the data at the message level instead of the transport level. Transport level security can be achieved by using SSL/TLS. End-to-end security in WS-Security has a higher performance cost than transport level security [77]. If there are no intermediary nodes involved (which forward SOAP messages), then we suggest SSL/TLS encryption for transport security. Message-based encryption in WS-Security is relatively difficult to implement and may lead to security bugs during development.

We recommend the following actions, based on, and in addition to the best practices mentioned by OWASP [63] and WS-Attacks [87].

1. Use the latest version of the web service framework in the OSS product.

2. Do not publish WSDL unless required, in order to prevent WSDL disclosure. Some enterprises also recommend authentication for WSDL retrieval.

3. Use strong rules for schema validation. The XSD should define maximum length, character set for all the input and output parameters for the web service.

4. XML Entities should not be allowed in SOAP messages in order to prevent DoS by an XML Bomb attack, external entity attack and other attacks [35].

5. Disable determination of the operation from the SOAPAction HTTP header in order to mitigate SOAPAction spoofing attack [36].

6. The web service should be WS-I Basic Profile [51] compliant at least.

5.2.1.2 REST Web Service

RESTful APIs offer manipulation of resources over HTTP with operations that correspond to HTTP verbs (`GET`, `POST`, `PUT`, `DELETE`, etc.). The textual data is returned in formats such as JSON, XML, or Text. HTTP is not a secure protocol since the data is transported in plaintext format. Hence, REST APIs must use HTTPS (RFC 2818) [71] for transport security. Similar to SOAP, there are many frameworks available to deploy RESTful web services. Few examples for Java include Jersey and Apache CXF frameworks. Unlike SOAP, REST is not a standard but an architectural style. Unlike WS-Security for SOAP, there is no complete security framework available for REST but many components can be used to secure REST-based interfaces.

We recommend the following actions, based on, and in addition to the best practices mentioned by OWASP [61].

1. Use the latest version of REST API library or framework, if used.

2. Use HTTPS (RFC 2818) [71] for securing the transport layer. Restrict the usage of HTTP.

3. Basic Access Authentication (RFC 7617) [70] must be avoided. If a username-password based authentication is required, we recommend using Digest Access Authentication (RFC 7616) [74] with a strong hashing algorithm such as SHA-256.

4. Use API keys or token-based authentication (for example, JSON Web Tokens (RFC 7519) [39]). Use different profiles for BSS clients for authorization. Provide mechanisms to revoke, refresh, and change authorizations for API tokens/keys. Since REST APIs are stateless, authenticate and authorize every API request.

5. For authorization, OSS vendors can implement OAuth 2.0 (RFC 6749) [31]. OAuth 2.0 is generally used when a user of Application-1 wants to give access to Application-2 to access their data in Application-1.

In the BSS-OSS use case, we have two different layers and the users do not *share* data. OSS users only administer and monitor while BSS users do not have access to the OSS. Hence, a *two-legged* OAuth 2.0 is relevant in which user interaction is not required for server to server authorization [46].

6. Do not pass sensitive data in the URL. An example of an insecure HTTP request with sensitive parameters in the URL is as follows:
   ```
   POST /request/?key=abc&subscriber_id=23 HTTP/1.1
   Host: pa-host
   ```

7. Restrict or strictly authorize methods that alter the state such as `POST`, `PUT`, and `DELETE`. Preferably, BSS systems should not be able to delete requests but rather cancel requests.

8. Validate the request for malformed data and the `Content-Type` header. Check the `Accept` header and reject the request if the format is not supported. This will reduce unnecessary load on the application.

5.2.1.3 Custom Connections

Many OSS vendors design their own protocols for BSS to OSS communication. Some are simple connections that resemble a Telnet session while others are complex protocols. We do not recommend implementing own security protocols for connections from the BSS system to the OSS system. Well known and already established protocols must be used.

However, we suggest the use of a secure communication channel (SSL/TLS), mutual authentication, and nonces/timestamps for freshness, as a bare minimum.

5.2.2 Southbound Interfaces

We mentioned in Section 3.3.4 that OSS systems (e.g. P&A) connect to southbound systems or NEs in order to execute the tasks generated for the requests received. Furthermore, Section 3.3.5 described that a P&A system uses a network model to keep track of NEs and connection details. The network model contains connection information, usernames and passwords, and other details of NEs. Hence it is important to allow access to this information only to privileged users.

The network element interface (NEI) component of a P&A system is responsible for implementing the task on the NE. Hence, proper file permissions must be set to make sure that NEI files cannot be tampered in order to prevent scenarios where the NEI is replaced by a malicious NEI to perform MitM attacks. Installation of new NEIs must be verified for authenticity of the publisher. Addition and configuration of NEs in the network model must be allowed only to users with administrative privileges.

Using a username and a password is one of the most common methods for a P&A system to authenticate itself to an NE. The username and password must be stored in an encrypted format using a strong encryption algorithm (such as AES-128) [20] and keys must be stored in secure locations. These practices will improve security with regards to vulnerabilities mentioned in Sections 4.2 and 4.5. Once configured, the password shall never be displayed on the user interface. The communication must always be carried over an encrypted channel.

Avoid configuration of an NE by automating the user interface. For example, many vendors develop NEIs that automate manual configurations over a Telnet shell which leads to an insecure OSS-NE interface (see vulnerability in Section 4.9). Shell access to an NE must always be protected with SSH. The use of proper APIs is recommended, if available. Modern NEs provide APIs using techniques such as REST, SOAP, or CORBA, which provide better security provisions. If the former technique is used, the user interface should show a warning in order for the administrators to take notice. For the latter techniques, follow the best practices mentioned in Section 5.2.1.

The network security engineers must make sure that steps are taken to prevent attackers from bypassing the OSS or P&A systems altogether to reach the NEs. Configure the NEs to allow connections only from trusted networks, that is, the network where the OSS system is running. Access to an NE must be prohibited unless it is from a trusted network. If supported, the login accounts on the NEs must be configured with different authorization levels.

Sharing or leasing of NEs to other operators is a common practice. Other operators should not be allowed to configure leased NEs directly, and all requests for configuration must go through the main operator's OSS systems. This reduces misuse and protects from threats arising from the other operator's network.

5.2.3 User Interface

Many OSS systems serve the UI as a web application and must address the security risks described in Section 4.12. It is recommended to follow the best practice guidelines from reputed sources such as the Mozilla Foundation [12]. Enforce HTTPS by default for the UI by using HTTP Strict Transport Security (HSTS) and disable HTTP. If not, a warning must be shown on the

UI that HTTPS must be configured for secure access. The UI should have a proper access control to show the content according to the authorization level of the users. Access control should be implemented on the server side to prevent attackers bypassing the UI.

Exposure of sensitive data on the UI must be prevented. It is recommended to mask sensitive parameters of requests and tasks on the UI and allow only the administrator level users to view them. The system must allow on a system-wide level, the addition and deletion of parameter names that are considered sensitive. There must also be provisions to specify a sensitive parameter that must be masked in the incoming request itself (e.g., by specifying an extra attribute for the parameter).

All form input and URL parameters must be properly validated for malicious input. Autocomplete must be disabled for forms that accept sensitive data. Use parameterized queries and escape SQL statements in inputs to prevent SQL injections. All input must be checked for maximum length in order to prevent buffer overflows and code injections.

Any data added to the HTML must be escaped prior to outputting. This includes escaping HTML, JavaScript, and CSS. There are several libraries available for escaping data. Use `HttpOnly` attribute for sensitive cookies that must be accessed only at the server. Use a strong content security policy to prevent XSS. Special care must be taken to return appropriate `Access-Control-*` headers to only entertain legitimate-origin requests.

Use CSRF tokens to mitigate CSRF. There are various frameworks that provide CSRF mitigation. Apache Tomcat comes with a built-in CSRF filter that must be configured. Note that any framework or external components used must be updated to the latest versions periodically.

Furthermore, user management should be only allowed to an administrator. User passwords must expire periodically and strong password policies must be enforced.

5.2.4 Other Interfaces

Inter-component interfaces connect the sub-components of an OSS application. Often such interfaces are provided by message queues such as ActiveMQ, for example, to pass messages from the service module to the task module in a P&A system. Another example of inter-component interfaces is Java RMI (remote method invocation). Communication across these interfaces should always be encrypted, especially if the components reside within different trust boundaries. If an external component is used for interfacing such as ActiveMQ, it should be configured with the recommended security guidelines to prevent vulnerabilities described in Sections 4.3 and 4.14. For example, instead of using the ActiveMQ simple authentication, certificate-based authentication must be used.

Another important interface is the connection to the database component. The communication must use a secure channel. The connection strings containing the credentials and connection parameters must be stored in an encrypted format.

There are interfaces to other OSS applications as well. The communication typically uses the northbound API of the target application. For example, when a P&A system connects to a Catalog system, it uses the Catalog system's northbound API. The same guidelines from the northbound interfaces apply when communicating from one OSS system to another.

5.3 Logging

OSS developers must use adequate logging for accounting purposes. Important transactions should be logged in an audit trail with proper timestamps. Audit trails are useful while addressing repudiation claims. For example, P&A systems should have a request log, NE tasks log, BSS transaction log, and a user audit trail. An audit trail for changes to provisioning logic is useful while investigating the tampering of provisioning logics (see Section 4.15). In addition to addressing repudiation, logs help in finding faults and determining security breaches.

However, due care must be taken not to log sensitive request or task parameters. Such data must be masked before logging. Chuvakin et al. describe several aspects of what not to write into logs [25]. The same considerations regarding sensitive data exposure as mentioned in Section 5.2.3 for the UI should apply while logging data. Moreover, the request and task parameters that are considered sensitive must be configurable in the system. The configuration can be shared by the UI and the logger components.

5.4 Host Security

According to the Onion Model of security described in Section 5.1.1, a secure OSS host strengthens the defence in depth. An out-of-the-box operating system may be insecure by default [21] and must be configured correctly before deploying the OSS solution. Security hardening reduces the host's attack surface. Essentially, a security hardening process generally follows the principle of "if it is not permitted, it is forbidden" [48].

An effective way to reduce the attack surface is to use a firewall. OSS applications open multiple ports for communication between its components. These ports increase the attack surface and must be blocked from the external network. Linux servers can make use of the `iptables` program to block incoming requests from external networks. Use a "deny all, allow some" policy by allowing only the required traffic into the host. For example, Figure 5.2 lists the `iptables` rules to only allow incoming packets for HTTP and SSH while dropping all other packets. In case of a P&A system, the rules must take into consideration ports related to northbound interfaces and other ports for incoming traffic.

```
Chain INPUT (policy DROP)
target prot opt src dst
ACCEPT all  --   *   *  ctstate RELATED,ESTABLISHED
ACCEPT tcp  --   *   *  tcp dpt:ssh ctstate NEW,ESTABLISHED
ACCEPT tcp  --   *   *  tcp dpt:http ctstate NEW,ESTABLISHED

Chain FORWARD (policy DROP)
target prot opt src dst

Chain OUTPUT (policy ACCEPT)
target prot opt src dst
ACCEPT all  --   *   *  ctstate ESTABLISHED
```

Figure 5.2: `iptables` Example

Running an auditing tool to assess the security of a host is recommended. A tool such as Lynis [7] lists the shortcomings in the security parameters of a Unix-like host and suggests proper configuration. There are tools such as Bastille-Linux [3] that help to automate security configuration based on best practices. Furthermore, while setting up a server, it is recommended to follow security guidelines by the Operating System (OS) vendor such as the Red Hat Linux security guide [18] for Red Hat Linux.

Security auditing must be a continuous process since new vulnerabilities are found periodically. Hence, system administrators must make sure to install latest patches for the OS and keep a note of security advisories. Firewall and system logs must be monitored periodically in order to find out any suspicious activity. It is recommended for vendors to provide automated scripts which check if the host's security configuration is correct, before or after the OSS application is deployed.

The OSS application must be installed with an OS user with only the required authorizations. File permissions must be given considering the principle of least privilege. Standard security guidelines must be followed in case of networked storage.

5.5 Validation

Threat modelling is a continuous and iterative process. The validation process involves validating the threat model. Any design changes to the OSS solution must be incorporated in the threat model. Thus, during validation, such left-out tasks are identified.

Along with validating the threat model, the mitigations must also be validated. Testing for security vulnerabilities should occur along with the usual testing activities or more often. Code reviews also help to identify if mitigations are correctly applied.

Finally, use of testing guides such as OWASP Testing Guide [53] and automated penetration testing tools such as Metasploit [8] or OWASP ZAP [64] is recommended.

5.6 Dos and Don'ts

Following are a few guidelines for OSS developers and deployment personnel to follow:

Don't use the same keys for every installation. Every installation should randomly generate new cryptographic keys. This prevents the possibility of all installations being at risk if one of the installations gets compromised.

Do validate all input. Developers must make sure to validate the length of all inputs to the OSS system to prevent buffer overflows. Validate at the server since any validation scripts at the client such as JavaScript can be disabled.

Don't store passwords unless required. User passwords must never be stored, rather hashed using a randomly generated salt and a slow one-way function [60]. This prevents brute force, dictionary, rainbow table, etc. attacks. However, if password storage cannot be avoided, such as in case of NE passwords, they must be encrypted (see Section 5.2.2).

Don't keep sensitive information in plaintext. Sensitive information should not be stored in plaintext, showed on the UI, or logged. Configuration files used only for installation must be removed after installation as the files may contain critical information such as database credentials.

Do configure TLS correctly. We have recommended the use of TLS as a secure channel for interfaces in this thesis. However, TLS must be configured correctly. This includes, enforcing use of versions $\geq$TLS 1.0, disabling weak cipher suites, disallowing fallback to plaintext communication, use of strong keys, and setting up correct certificates. The OWASP Testing Guide Chapter 4.10 [53] describes guidelines to test for an improper TLS configuration. We recommend following the OWASP Transport Layer Cheat Sheet [62], which mentions different rules that must be followed for a secure transport layer configuration.

Do verify installation packages for integrity and authenticity. The release process should be able to create installation packages that are signed. Before installing, packages must be checked for authenticity and integrity. For example, new NEIs packages should be verified before installing.

Do check the dependencies for disclosed vulnerabilities. Developers must check the dependencies (e.g., frameworks and libraries) that have disclosed vulnerabilities. Preferably, this must be automated and the release pipeline must be configured so that a scan for vulnerable dependencies is run by the continuous integration system. An example of such a utility is OWASP Dependency-Check [59].

Do have a proactive security policy. This includes incorporating security in the SDLC. Use code-reviews and static code analysis utilities for finding potentially insecure code. Use automated security testing tools in the continuous integration system. Follow well-known security guides such as OWASP Proactive Controls [66].

Moreover, it is important to give priority to security fixes. Vendors often focus more on building functionality while product security becomes a secondary concern.

Do publish a security guide as part of product documentation. It is easy to misconfigure security parameters during deployment. Hence, it is recommended for OSS vendors to publish a security guide for configuring the security parameters of their OSS products correctly. The guide should at the least mention the basic OS security requirements, user permissions, steps for security configuration of the OSS product and the dependencies, and basic testing guidelines. Besides, a security guideline document increases the customer's trust in the product.

Chapter 6

Discussion and Conclusion

In this chapter, we discuss our observations and draw conclusions. First, we discuss a survey on security we conducted amongst OSS professionals in Section 6.1. Section 6.2 discusses our observations and the challenges faced over the course of this thesis. Finally, the concluding remarks are described in Section 6.3.

6.1 Security Survey

One of the goals of this thesis is to realise the importance of security for OSS systems. In order to find out the current situation and the expectations regarding security, we conducted a survey among professionals who have worked in the deployment and sales of OSS systems in the telecommunications industry. The survey questionnaire is listed in Appendix B and had six questions. The survey did not ask any customer or vendor information and the responses were anonymous. We received 11 responses to the questionnaire. The following paragraphs analyse the responses.

18.2% of the respondents feel that OSS systems are vulnerable to attacks only by malicious insiders. According to another 18.2%, external adversaries are more likely to attack. However, a majority of the respondents answered that an OSS deployment is threatened by all of the attacks, viz., malicious insiders, customers, and external adversaries.

Over 50% of the respondents think that the security of OSS systems is not prioritized by telecommunication companies in general. 63.3% of the respondents answered that the security policies of telecommunication companies concerning their information assets are prioritized based on threat perception and risk management. However, 27.3% respondents still consider that the budget is a factor that solely influences the security policy.

All the respondents think that there should be more security audits of OSS solutions. One of the respondents commented that auditing efforts should be continuous.

According to 63.6% of the respondents, the "Perimeter Security Model" is prevalent in most of the telecommunication companies in the current OSS landscape. Moreover, all the respondents answered that the "Onion Model" should be given more emphasis in OSS solutions.

We also received the following additional comments by some respondents:

1. *"Usually vendors do a security check based on available tools for which they have paid the license of and that software might not have been updated with the new threat which might got missed during application security checks."*

2. *"The biggest challenges are a) ageing systems and b) insiders being sloppy/malicious."*

6.2 Observations and Challenges

The results from the survey questionnaire in Section 6.1 show that there is a consensus amongst OSS professionals that the current security priorities by both OSS vendors and customers are below expectations. We recommend that there should be periodic security audits by the OSS vendors with updated tools and information while following the threat mitigations diligently during the development phase. The customers should focus on the security of the network where the OSS solution is deployed and collaborate with vendors. The security requirements should not come solely from the customers. In such a scenario, vendors tend to simply fulfil the minimum requirements just enough to sell the OSS solution. Vendors should pro-actively work on improving the security of their products and make it a selling point.

The use of cloud services in the OSS landscape has been a point of discussion for several years now. The Software as a Service (SaaS) model is still not preferred by telecommunication companies for their OSS requirements mainly because of lack of trust in keeping their data with an external entity. Specifically, systems which interface directly with the NEs such as P&A systems are usually deployed *in-premise*, i.e., within the enterprise's private network, while the peripheral OSS applications such as Order Management are deployed in the cloud. Thus, in such cases, the configuration of virtual networks and hosts on the cloud must follow the layered security model we described in Chapter 5.

The use of STRIDE methodology and the four-step framework described in Section 2.3 for our threat modelling purposes is open to debate. The main reason why we selected STRIDE is because it is easier to understand from a non-security professional's point of view. OSS customers usually do not employ security experts, and therefore, it is important to use a method for classifying threats that is easy to understand. Moreover, the use of DFDs allowed us to visualize the data flow and identify the trust boundaries efficiently.

Security is not a one-time fix for threats. There could be unknown vulnerabilities that may be found in the future. The list of vulnerabilities described in Chapter 4 should not be considered an exhaustive list. Brainstorming for threats helps, but keeping track of security advisories and following guidelines from well-known sources is important. The vulnerabilities we explained are some of the most commonly found vulnerabilities in computer software and OSS systems.

In the mitigation phase, our goal was to secure the interfaces, the host, and the network in accordance with the principle of a layered defence. Furthermore, the mitigations mentioned are subject to re-evaluation as new threats may be discovered. Hence it is recommended to continually assess, correct and develop new mitigations if needed.

We faced some challenges during the course of this thesis. OSS products are proprietary and closed-source software. As a result, it is difficult to find an open architecture of such systems. The architecture that we described is a generally accepted architecture of a P&A system derived by personally communicating with software developers who have worked for several decades in the industry. However, the reader must note that vendors develop their own architectures and the threats and mitigations may vary accordingly.

6.3 Conclusion

In this thesis, we achieved the goals that were listed in Section 1.1. The summary of goals in the same order as Section 1.1 is as follows:

1. The results from our survey, asset analysis, and the subsequent security analysis done in this thesis, point towards the importance of securing OSS systems.

2. We built a threat model for an OSS system for provisioning and activation in Chapter 3. The threat model helped us determine the different threats using the STRIDE methodology. The DFDs helped us under-

stand the data flow through trust boundaries and to find out the entry and exit points.

3. We described common vulnerabilities in Chapter 4 for OSS vendors to address in their products. These vulnerabilities refer to the ones found in typical web applications as well as the vulnerabilities from the point of view of OSS systems.

4. We described mitigation in Chapter 5. The mitigation puts emphasis on layered defence and focusses on the northbound and southbound interfaces of the OSS systems. We also described general guidelines in the form of dos and don'ts that can help OSS vendors to mitigate threats.

Over the coming years, the OSS market is expected to grow in the traditional telecommunications field as well as other fields such IoT, cloud orchestration, and the energy industry. With new security and privacy regulations, and the increasing customer's concerns over the security of their software, it is necessary for OSS vendors to address the security problems in their products proactively.

Bibliography

[1] Apache ActiveMQ. http://activemq.apache.org/.

[2] Apache Tomcat. http://tomcat.apache.org/.

[3] Bastille Linux. http://bastille-linux.sourceforge.net/.

[4] Ettercap. http://ettercap.github.io/ettercap/.

[5] Introduction to Microsoft Security Development Lifecycle (SDL) - Threat Modeling. https://download.microsoft.com/download/9/3/5/935520EC-D9E2-413E-BEA7-0B865A79B18C/Introduction_to_Threat_Modeling.ppsx. Accessed 14 June 2017.

[6] JD (Java Decompiler). http://jd.benow.ca/.

[7] Lynis. https://cisofy.com/lynis/.

[8] Metasploit Framework. https://www.metasploit.com/.

[9] Network Inventory - Gartner IT Glossary. http://www.gartner.com/it-glossary/network-inventory/. Accessed 26 April 2017.

[10] NIST National Vulnerability Database. https://nvd.nist.gov/vuln/search.

[11] Nmap (Network Mapper). https://nmap.org/.

[12] Security/Guidelines/Web Security - Mozilla Wiki. https://wiki.mozilla.org/Security/Guidelines/Web_Security. Accessed 20 September 2017.

[13] SlowHTTPTest. https://github.com/shekyan/slowhttptest. Accessed 28 August 2017.

[14] tcpflood - TCP Established Flood Tool. https://github.com/zupper/tcpcflood. Accessed 25 August 2017.

[15] Wireshark (Network Protocol Analyzer). `https://www.wireshark.org/`.

[16] Microsoft Threat Modeling Tool. `https://www.microsoft.com/en-us/download/details.aspx?id=49168`, 2016. Accessed 14 June 2017.

[17] Q2 2017 Akamai State Of The Internet. `https://www.akamai.com/us/en/about/news/press/2017-press/akamai-releases-second-quarter-2017-state-of-the-internet-security-report.jsp`, 2017. Accessed 25 August 2017.

[18] Red Hat Enterprise Linux 7 Security Guide. `https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/`, 2017. Accessed 26 September 2017.

[19] ATIS. Glossary - Alliance for Telecommunications Industry Solutions, 2011.

[20] Barker, E., and Roginsky, A. Transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths. *NIST Special Publication 800* (2011), 131A.

[21] Bauer, M. *Linux server security.* O'Reilly Media, Inc., 2005.

[22] Blackman, C., and Srivastava, L. *Telecommunications regulation handbook.* World Bank and the International Telecommunication Union, Washington, DC, 2011.

[23] Byres, E. Defense in depth. *Control Engineering Asia June 2008* (2008).

[24] Case, J. D., Fedor, M., Schoffstall, M. L., and Davin, J. Simple network management protocol (SNMP). RFC 1157, May 1990. URL: `http://www.rfc-editor.org/rfc/rfc1157.txt`.

[25] Chuvakin, A., and Peterson, G. How to do application logging right. *IEEE Security & Privacy 8*, 4 (2010), 82–85.

[26] Colwill, C. Human factors in information security: The insider threat - Who can you trust these days? *Information security technical report 14*, 4 (2009), 186–196.

[27] Eddy, W. M. TCP SYN flooding attacks and common mitigations. RFC 4987, August 2007. URL: `http://www.rfc-editor.org/rfc/rfc4987.txt`.

[28] Federal Communications Commission. Telecommunications act of 1996. *Public law 104*, 104 (1996), 1–5.

[29] Gallagher, S. Two keys to rule them all: Cisco warns of default SSH keys on appliances. Ars Technica. `https://arstechnica.com/information-technology/2015/06/two-keys-to-rule-them-all-cisco-warns-of-default-ssh-keys-on-appliances/`, 2015. Accessed 04 August 2017.

[30] Gutmann, P. *Engineering Security.* 2014. URL: `https://www.cs.auckland.ac.nz/~pgut001/pubs/book.pdf`.

[31] Hardt, D. The OAuth 2.0 Authorization Framework. RFC 6749, October 2012. URL: `http://www.rfc-editor.org/rfc/rfc6749.txt`.

[32] Insight Research. The 2015 telecommunications industry review: An anthology of market facts and forecasts. `http://www.insight-corp.com/reports/review15.asp`, 2015.

[33] International Engineering Consortium. *Operations Support Systems 2002: Enabling the Next Generation Network.* Comprehensive Report. International Engineering Consortium, 2002.

[34] ITU-T. Recommendation X.710 (10/97). *Common Management Information Service* (1998).

[35] Jan, S., Nguyen, C. D., and Briand, L. Known XML vulnerabilities are still a threat to popular parsers and open source systems. In *Software Quality, Reliability and Security (QRS), 2015 IEEE International Conference on* (2015), IEEE, pp. 233–241.

[36] Jensen, M., Gruschka, N., and Herkenhöner, R. A survey of attacks on web services. *Computer Science-Research and Development 24*, 4 (2009), 185–197.

[37] Jensen, M., Gruschka, N., Herkenhoner, R., and Luttenberger, N. SOA and web services: New technologies, new standards - new attacks. In *Web Services, 2007. ECOWS '07. Fifth European Conference on* (Nov 2007), pp. 35–44.

[38] Jiang, G. Multiple vulnerabilities in SNMP. *Computer 35*, 4 (2002), supl2–supl4.

[39] Jones, M., Bradley, J., and Sakimura, N. JSON Web Token (JWT). RFC 7519, May 2015. URL: `http://www.rfc-editor.org/rfc/rfc7519.txt`.

[40] Kaiser, M. Pwning your Java messaging with deserialization vulnerabilities. White paper, 2016. Blackhat USA 2016.

[41] Kalra, G. S. A pentesters guide to hacking ActiveMQ-based JMS applications. White paper, 2014. McAfee Inc.

[42] Kalra, G. S. Threat analysis of an enterprise messaging system. *Network security 2014*, 12 (2014), 7–13.

[43] Kamp, P.-H., Godefroid, P., Levin, M., Molnar, D., McKenzie, P., Stapleton-Gray, R., Woodcock, B., and Neville-Neil, G. Linkedin password leak: Salt their hide. *ACM Queue 10*, 6 (2012), 20.

[44] Kissel, R. Glossary of key information security terms. *NIST Interagency Reports NIST IR 7298*, 3 (2013).

[45] Le Métayer, D. IT security analysis best practices and formal approaches. *Lecture Notes in Computer Science 4677* (2007), 75.

[46] Liu, K., and Xu, K. OAuth based authentication and authorization in open telco API. In *Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on* (2012), vol. 1, IEEE, pp. 176–179.

[47] Manadhata, P. K., Tan, K. M., Maxion, R. A., and Wing, J. M. An approach to measuring a system's attack surface. Tech. rep., Carnegie Mellon University School of Computer Science, 2007.

[48] Mann, S., and Mitchell, E. L. *Linux system security: an administrator's guide to open source security tools.* Prentice Hall Professional, 2000.

[49] Marlinspike, M. SSLStrip. `https://moxie.org/software/sslstrip/`. Accessed 18 August 2017.

[50] Marlinspike, M. More tricks for defeating SSL in practice. *Black Hat USA* (2009).

[51] McIntosh, M., Gudgin, M., Morrison, K. S., and Barbir, A. Basic security profile version 1.0. *WS-I Standard 30* (2007).

[52] Meier, J., Mackman, A., Dunner, M., Vasireddy, S., Escamilla, R., and Murukan, A. Improving web application security: Threats and countermeasures. *Microsoft Corporation 3* (2003).

[53] Meucci, M., and Muller, A. The OWASP testing guide 4.0. *Open Web Application Security Project* (2014), 30.

[54] Misra, K. *OSS for Telecom Networks: An Introduction to Networks Management*. Springer Science & Business Media, 2004.

[55] Myagmar, S., Lee, A. J., and Yurcik, W. Threat modeling as a basis for security requirements. In *Symposium on requirements engineering for information security (SREIS)* (2005), vol. 2005, pp. 1–8.

[56] Nadalin, A., AmberPoint, G. T., BEA, P. D., BEA, H. L., CommerceOne, S. C., ContentGuard, T. D., ContentGuard, G. L., ContentGuard, T. P., Commerce, S. S. C., Documentum, G. V., et al. Web services security. *SOAP Message Security. Version 1* (2002).

[57] Ornaghi, A., and Valleri, M. Man in the middle attacks. In *Blackhat Conference Europe* (2003).

[58] OWASP. Application Threat Modeling. `https://www.owasp.org/index.php/Application_Threat_Modeling`. Accessed 09 June 2017.

[59] OWASP. Dependency Check. `https://www.owasp.org/index.php/OWASP_Dependency_Check`. Accessed 26 September 2017.

[60] OWASP. Password Storage Cheat Sheet. `https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet`. Accessed 26 September 2017.

[61] OWASP. REST Security Cheat Sheet. `https://www.owasp.org/index.php/REST_Security_Cheat_Sheet`. Accessed 24 July 2017.

[62] OWASP. Transport Layer Protection Cheat Sheet. `https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet`. Accessed 26 September 2017.

[63] OWASP. Web Service Security Cheat Sheet. `https://www.owasp.org/index.php/Web_Service_Security_Cheat_Sheet`. Accessed 12 September 2017.

[64] OWASP. Zed Attack Proxy Project. `https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project`. Accessed 26 September 2017.

[65] OWASP. Top 10-2013. *The Ten Most Critical Web Application Security Risks* (2013).

[66] OWASP. Proactive Controls. `https://www.owasp.org/index.php/OWASP_Proactive_Controls`, 2016. Accessed 26 September 2017.

[67] Owens, J., and Matthews, J. A study of passwords and methods used in brute-force SSH attacks. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)* (2008).

[68] Park, J., Iwai, K., Tanaka, H., and Kurokawa, T. Analysis of slow read DoS attack. In *Information Theory and its Applications (ISITA), 2014 International Symposium on* (2014), IEEE, pp. 60–64.

[69] Parkinson, S. Secure Crypto: Weak ciphers be gone! - RSA Blog. `https://blogs.rsa.com/secure-crypto-weak-ciphers-gone/`. Accessed 23 August 2017.

[70] Reschke, J. The 'Basic' HTTP Authentication Scheme. RFC 7617, September 2015. URL: `http://www.rfc-editor.org/rfc/rfc7617.txt`.

[71] Rescorla, E. HTTP over TLS. RFC 2818, May 2000. URL: `http://www.rfc-editor.org/rfc/rfc2818.txt`.

[72] Saitta, P., Larcom, B., and Eddington, M. Trike v1 methodology document. *Draft, work in progress* (2005).

[73] Saltzer, J. H., and Schroeder, M. D. The protection of information in computer systems. *Proceedings of the IEEE 63*, 9 (1975), 1278–1308.

[74] Shekh-Yusef, R., Ahrens, D., and Bremer, S. HTTP Digest Access Authentication. RFC 7616, September 2015. URL: `http://www.rfc-editor.org/rfc/rfc7616.txt`.

[75] Shirey, R. W. Internet Security Glossary, Version 2. RFC 4949, August 2007. URL: `http://www.rfc-editor.org/rfc/rfc4949.txt`.

[76] Shostack, A. *Threat modeling: Designing for security.* John Wiley & Sons, 2014.

[77] Sosnoski, D. The high cost of (WS-)Security. https://www.ibm.com/developerworks/library/j-jws6/index.html. Accessed 11 September 2017.

[78] Sosnoski, D. The state of web service security. https://www.ibm.com/developerworks/java/library/j-jws19/j-jws19-pdf.pdf. Accessed 12 September 2017.

[79] Swiderski, F., and Snyder, W. *Threat modeling.* Microsoft Press, 2004.

[80] Telcordia. Operations Application Messages - Language For Operations Application Messages, 1996.

[81] TM Forum. Frameworx, Applications Framework, The Digital Services Systems Landscape, 2017.

[82] TM Forum. Frameworx, Business Process Framework (eTOM), 2017.

[83] TM Forum. Frameworx, Information Framework (SID), Catalog Business Entities, 2017.

[84] UcedaVelez, T., and Morana, M. M. *Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis.* John Wiley & Sons, 2015.

[85] US-CERT. Understanding Denial-of-Service Attacks. Security Tip (ST04-015). https://www.us-cert.gov/ncas/tips/ST04-015. Accessed 25 August 2017.

[86] Whalen, S. An introduction to ARP spoofing. https://packetstormsecurity.com/papers/protocols/intro_to_arp_spoofing.pdf, 2001. Accessed 16 August 2017.

[87] WS-Attacks.org. Security Best Practices: Web Services. http://www.ws-attacks.org/Security_Best_Practices:_Web_Services. Accessed 18 August 2017.

[88] Zargar, S. T., Joshi, J., and Tipper, D. A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. *IEEE communications surveys & tutorials 15*, 4 (2013), 2046–2069.

Appendix A

Threat Enumeration

This appendix lists the threats obtained by applying Stride-per-Element on each of the DFDs. We have combined the common threats in order to make the list shorter.

A.1 Context (High level)

ID	Title/Description	Type
C1	*Spoofing of external entity* An external system (Inventory, Catalog, or a BSS) may be spoofed by an attacker, thus leaking data to the attackers system instead of the intended system.	*Spoofing*
C2	*Spoofing of P&A* The P&A may be spoofed which might leads the external entities sending information to an illegitimate system	*Spoofing*
C3	*Lack of input validation* Data flowing across the interfaces between OM/PA and external entities may be tampered leading to processing wrong information	*Tampering*
C4	*External System denies receiving or sending data* A NE, BSS, Inventory, or Catalog might deny receiving or sending data to/from the OM/PA	*Repudiation*
C5	*Information sniffing on the interfaces* Data flowing across the interfaces between OM/PA and external entities may be sniffed leading to information getting leaked to the attacker	*Information disclosure*

ID	Title/Description	Type
C6	*OM/PA crashes or does not respond* An adversary makes OM/PA or an internal component crash or respond slowly leading to a service outage	*Denial of Service*
C7	*Interruption of data flow to and from OM/PA* An adversary may interrupt the data flow between Inventory, Catalog, BSS, NE and OM/PA	*Denial of Service*
C8	*Cross site request forgery* An attacker may exploit the trust between a BSS and OM/PA utilizing an existing session to send unauthorized commands	*Elevation of privilege*
C9	*Elevation using impersonation* An attacker may used sniffed data to impersonate a legitimate BSS in order to run privileged commands	*Elevation of privilege*

A.2 User Authentication

ID	Title/Description	Type
U1	*Spoofing of the external identity provider* The external identity provider may be spoofed by an attacker which may lead to the authentity and authorization requests being sent to the malicious target	*Spoofing*
U2	*Spoofing of the web-application* The web application on the P&A host could be spoofed by an attacker which may lead the browser to send requests to malicious target	*Spoofing*
U3	*Spoofing of the browser* The browser may be spoofed by the attacker which may lead to unauthorized access to the web application	*Spoofing*
U4	*Lack of input validation by the web app* Requests coming from the browser to the web application may be tampered by a malicious user leading to multiple security issues and data discrepencies	*Tampering*
U5	*Repudiation by external entity* External entities such as the identity provider or the browser might deny sending or receiving data	*Repudiation*
U6	*Sniffing of data on interfaces* Data on interfaces to the web application to and from the browser, identity provider and the data store could be possibly sniffed by an attacker leading to a potential leak of valuable data	*Information Disclosure*
U7	*SQL Injection* An attacker may inject sql statements in the browser UI leading to tampering or accessing unauthorized data	*Information Disclosure*
U8	*Weak Authentication and Access control* The web application might provide a weak authentication scheme allowing attacker easily guess user passwords. If the authorization scheme is flawed then attacker can view and change unauthorized data.	*Information Disclosure*
U9	*Weak Credential Storage* If in case of a data breach the stored credentials could be disclosed and subject to a guessing or a brute force attack.	*Information Disclosure*
U10	*Authorizaton Bypass*	*Information Disclosure*

ID	Title/Description	Type
	If it is possible to reach the data store or the configuration files without passing throught the web application. In this case the attacker might be able to bypass all the authorization.	
U11	*Interruption of data flow* An attacker may stop communication to and from any of the components by either flooding the component with requests or interrupting data flow on the interfaces	*Denial of Service*
U12	*Denial of service by crashing component* An attacker may cause the identity provider or the web application to crash thereby causing a denial of service	*Denial of Service*
U13	*Cross Site Request Forgery in the browser* An attacker my utilize existing session of the an authorized user to send unauthorized commands to the web application. The attacker might also be able to steal the session and possibly change provisioning logics.	*Elevation of Privilege*
U14	*Cross Site Scripting* XSS is a common vulnerability in web applications enabling attackers to inject client script scripts into web pages leading to a variety of security implications	*Elevation of Privilege*

A.3 Northbound Interface

ID	Title/Description	Type
N1	*Spoofing of P&A or BSS* Attacker may spoof P&A or the BSS entity. If the P&A is spoofed, then BSS requests may be sent to attackers malicious P&A. If BSS is spoofed, P&A will receive requests from malicious BSS.	*Spoofing*
N2	*Lack of input validation at interface* Lack of input validation at SOAP, REST, etc. interface may lead the attacker to send tampered data. For example, malformed SOAP messages may crash the SOAP interface and reveal sensitive information	*Tampering*
N3	*Corruption of data store* An attacker may use SQL injection in the requests or might tamper data flowing across P&A and the data storage in order to corrupt the BSS related data thereby breaking the integrity of BSS and request Data	*Tampering*
N4	*BSS denies sending/receiving data* External BSS system may claim not sending or receiving data to or from the P&A interface	*Repudiation*
N5	*Weak authentication/authorization* The interfaces might implement a weak authentication scheme such as weak guessable passwords and weak credential change management. A weak authorizaton leads to attackers getting access to unauthorized services.	*Information Disclosure*
N6	*SQL Injection* An attacker may inject SQL statements in request messages on the nortbound interfaces. If data is not escaped properly, it might lead to disclosure of critical information	*Information Disclosure*
N7	*Weak credential storage* If an attacker gets access to credential data, then attacker can try brute force or rainbow table attacks to find valid credentials. A weak credential storage can make this possible.	*Information Disclosure*
N8	*Sniffing of interfaces* An attacker may sniff data flowing across the BSS-P&A interface. If not encrypted, this may result in the attacker getting access to critical data.	*Information Disclosure*
N9	*Interruption in data flow*	*Denial of service*

ID	Title/Description	Type
	Attacker may try to interrupt data flow between the BSS entities and the interfaces by flooding the interface with requests (e.g. Login requests).	
N10	*Elevation using impersonation* A lower priviledge BSS may pretend to be a higher priviledge BSS in order to gain access to unauthorized features. A weak authorization may lead to such a scenario.	*Elevation of privilege*

A.4 Southbound Interface

ID	Title/Description	Type
S1	*Spoofing of Network Element or P&A* If an attacker is able to spoof a network element then the P&A system will send the tasks to the attacker's system thereby revealing important information. If the P&A is spoofed the attacker can impersonate it to gain the trust of NE	*Spoofing*
S2	*Lack of input validation of tasks* Attackers or malicious BSS systems may send provisioning requests with tampered data to misconfigure the NEs. A lack of input validation at P&A or the Network Interfaces will lead to misconfiguration	*Tampering*
S3	*Data repudiation by NEs* A Network Element may claim that it did not receive a task from the Network Element Interface or the P&A	*Repudiation*
S4	*Disclosure of log data* NE logs contain information related to tasks sent to Nes and the responses received. This information is sensitive and can be useful for malicious users.	*Information Disclosure*
S5	*Weak authentication scheme for NEs* A weak authenciation scheme for Network Elements such as encrypting passwords using weak encryption may affect security of the system	*Information Disclosure*
S6	*Data flow sniffing* Data flowing from the NE interface to the Network Element may be sniffed by an attacker leading to revelation of sensitive information. The same applies to sniffing of data flow between P&A and the database	*Information Disclosure*
S7	*Bypassing authentication/authorization* A weak authentication/authorization in P&A may allow attackers to bypass P&A and directly send tasks to Network Elements. Or if the attacker can tamper the Data Store, malicious tasks can be sent to Nes	*Information Disclosure*
S8	*Weak Credential Storage* Network Element access credentials are stored in the data store in encrypted form. If the credential storage is weakly encrypted, attackers might be able to decrypt the credentials using a variety of methods such as brute-force, rainbow tables, dictionary attacks.	*Information Disclosure*
S9	*Component crash or data flow interrupted*	*Denial of Service*

ID	Title/Description	Type
	A high number of requests or tasks may lead to crash of P&A or the NE interface leading to denial of service. Tampered request data could also lead to a crash because of lack of input validation	
S10	*Elevation using impersonation*	*Elevation of Privilege*
	An attacker may impersonate the context of the P&A to gain access or additional privilege to a network element.	
S11	*Remote Code Execution*	*Elevation of Privilege*
	An attacker may try to send code in tampered request data and try to execute it remotely. For example a Code Injection.	

A.5 Provisioning and Activation

ID	Title/Description	Type
P1	*Spoofing the request module* If the attacker is able to spoof the request module it will lead to the BSS sending requests to an illegitimate process	*Spoofing*
P2	*Spoofing the queues* An attacker may tamper with request and task modules to use a malicious queue or spoof existing queues	*Spoofing*
P3	*Lack of input validation* An attacker can tamper data flowing across BSS and request module or NE and task module	*Tampering*
P4	*Tampering of provisioning logics* If an attacker tampers provisioning logic it will lead to wrong processing of requests and potential create network configuration tasks of malicious nature	*Tampering*
P5	*Insufficient auditing* If audit logs are not maintained it is difficult to handle repudiation claims	*Repudiation*
P6	*Sniffing request data* Attackers can sniff important request data on the interfaces connecting components to the data store	*Information Disclosure*
P7	*Risks from logging* Request logs can reveal critical information	*Information Disclosure*
P8	*Denial of service by crashing component* An attacker may try to crash any of the component to deny service. Components such as queues and the request module have open ports which serve as an entry point for attacks.	*Denial of Service*

Appendix B

Survey: Security of OSS Systems

This survey is conducted in order to find out the perception of security in the telecommunications industry with respect to operations support system (OSS) solutions. Please answer the following questions according to your experience and observation of the OSS implementations, customers and vendors.

1. According to you, what is the most critical security threat to an OSS system?

    - ○ Malicious Insider
    - ○ Malicious Customer
    - ○ Competitors
    - ○ Remote Attackers (e.g. Hacker groups)

    Comment: ________________________

2. Do you think that the security of OSS systems is prioritized by telecommunication companies in general?

    - ○ Yes
    - ○ No
    - ○ Not sure

    Comment: ________________________

3. How do telecommunication companies prioritize security policies for their information assets?

    - ○ Based on threat perception/risk assessment

- ○ Budget
- ○ Not sure

Comment: ________________________

4. Do you think that there should be more security audits of OSS solutions?

   - ○ Yes
   - ○ No

   Comment: ________________________

5. According to you, which security model is prevalent in most telecommunication companies (w.r.t. OSS) currently?

   - ○ Perimeter Security (Securing the enterprise-wide network's borders)
   - ○ Onion Model (Securing every application and component)
   - ○ Other (please comment briefly)

   Comment: ________________________

6. In your opinion, should there be more emphasis on application security (Onion Model) by OSS vendors?

   - ○ Yes
   - ○ No

   Comment: ________________________