



Norwegian University of
Science and Technology

Detecting Identity Thefts In Open 802.11e Enabled Wireless Networks

Eirik Holgernes

Master of Science in Communication Technology

Submission date: June 2010

Supervisor: Stig Frode Mjølsnes, ITEM

Norwegian University of Science and Technology
Department of Telematics

Problem Description

MAC sequence counting is a well proven technique in wireless Intrusion Detection Systems (wIDS). During a project done by Eirik Holgernes and Øystein Aas Pedersen fall 2009, discoveries of potential problems with link-layer Quality-of-Service/802.11e enabled networks was uncovered in regards to MAC sequence counting as a detection technique in wIDS.

This master thesis will focus on these issues, and look into the possibilities for enhancing the detection techniques to avoid both false negatives and false positives in regards of detecting attacks. During the project Eirik and Øystein developed a wIDS, named SIDS (Simple Intrusion Detection System) to be used in Wireless Trondheim open wireless network. The work done in this thesis will use SIDS as a framework and testing environment.

An article published by Chandrasekaran, Deymoous, Ganapathy, Gruteser and Trappe: Detecting Identity Spoofs in 802.11e Wireless Networks focus on how to cope with the 802.11e security flaw, and suggests an optimized MAC sequence counting algorithm to deal with the 802.11e issues. This thesis will focus on further testing of the proposed algorithm, and rely on experiments to analyze the problem further. Finally it will focus on revealing potential weaknesses and improvements in the solution proposed by the article.

Assignment given: 15. January 2010
Supervisor: Stig Frode Mjøltnes, ITEM

Abstract

Open wireless networks are commonly deployed as a result of easy access, user-friendliness, as well as easy deployment and maintenance. These networks do not implement strong security features, and clients are prone to a myriad of possible attacks. Identity attacks are considered one of the most severe, and as a result of this Intrusion Detection Systems (IDS) can be deployed. With the introduction of 802.11e/Quality-of-Service on a link-to-link basis in 802.11 networks, most IDS will become obsolete as they often rely on a detection technique known as MAC Sequence Counting Analysis. This specific technique will become useless if 802.11e/QoS is enabled on the network. In this thesis I have analyzed the problem further, and suggest new techniques, both implemented and verified as an IDS, as well as analytic theories in order to enhance MAC Sequence Counting Analysis to cope with the new features of 802.11e. There has been related work on the same issue, but this thesis questions their use of unreliable physical parameters in order to detect attacks. As we will see, my new proposed techniques rely on analysis of the 802.11 standard and the 802.11e amendment, and are not dependent on parameters which could be unreliable in urban and mobile environments. Experiments and analysis will demonstrate the validity of the new suggested techniques, and the outcome of the thesis will be divided into two parts; Development of an optimized Intrusion Detection System and an enhanced algorithm in order to detect attacks which exploits the new features of 802.11e.

Preface

This thesis is my final contribution from my 5 year study for the master's degree in Communication Technology. The work has been accomplished at the Department of Telematics at the Norwegian University of Science and Technology (NTNU). The assignment is given by Wireless Trondheim, and the work is a continuation of the work done by myself and Øystein Aas Pedersen during a project assignment fall 2009.

I would like to express my sincere regards to the staff at Wireless Trondheim for invaluable guidance of the practical work, as well as providing the lab equipment and offices. Without this, the work achieved in this thesis would be impossible. Finally, I will give a very special thanks to my supervisor ph.d. Martin Eian at the Department of Telematics. Your theoretical feedback and interest in my work has been outstanding, and without your support, the analytic and theoretical achievements in this thesis would not have been possible.

Trondheim, June 2010



Contents

Abstract	I
Preface	III
Table of Contents	III
List of Figures	IX
List of Tables	XIII
Abbreviations and acronyms	XV
I Introduction	1
1 Introduction	3
1.1 Background	4
1.1.1 Project Assignment 2009	5
1.1.2 Wireless Trondheim	6
1.2 The Problem	7
1.3 Contrubitions	8
1.4 Report Outline	9
	V

1.5	Related Work	11
1.5.1	RSS and IDS	11
1.5.2	802.11e and IDS	12
1.5.3	Specification-based IEEE 802.11 Attack Detection	13

II Developing an IDS 15

2 Background Theory 17

2.1	IEEE 802.11	18
2.1.1	Ad-Hoc And Infrastructure	18
2.1.2	The 802.11 MAC Frame	19
2.1.3	802.11 States	25
2.2	Attacks In IEEE 802.11 Environments	26
2.2.1	Eavesdropping And Traffic Analysis	27
2.2.2	Masquerading: Session Hijacking	28
2.3	Intrusion Detection Systems	28
2.4	Detection Techniques	29
2.4.1	MAC Sequence Control Analysis	30
2.4.2	RSS Monitoring And Physical Parameter Analysis	30
2.4.3	Traffic Pattern Analysis	31
2.4.4	RTS/CTS Roundtrip Analysis	31
2.4.5	Protocol Specific Analysis	32
2.5	IEEE 802.11e	32
2.5.1	Wireless Multimedia Management (WMM)	33
2.5.2	The QoS Control Field	33
2.5.3	Sequence Numbers and 802.11e	34
2.6	The Physical Layer	35

3	Method	37
3.1	Threat Model	38
3.2	QoS optimized Attack	41
3.3	Proposed Algorithm For QoS Considered Detection	42
3.3.1	Initial step	44
3.3.2	Step 2a	44
3.3.3	Step 2b	45
3.3.4	Step 2c	45
3.3.5	Step 2d	46
3.3.6	Step 2e	46
3.4	Developing SIDS v2.0	48
3.4.1	check.pl	50
3.5	Enhancements in SIDS v2.0	51
3.5.1	Passive Detection - 802.11 State Control	52
3.5.2	Active Detection - Active Probing	55
3.6	Attacking And Generating Offline Capture Files	57
3.6.1	NoAttack.cap	59
3.6.2	FreeloaderAttack.cap	60
3.6.3	FreeloaderPING.cap	60
4	Results	61
4.1	SIDS	62
4.1.1	MAC Sequence Counting Analysis from SIDS v1.0	62
4.2	SIDS v2.0 - Passive Detection	63
4.3	SIDS v2.0 - Active Detection	65
4.3.1	Firewalls	66
4.4	WMM enabled	67

III	Discussion and Analysis	69
5	SIDS v2.0	71
5.1	Implemented Detection Techniques in SIDS v2.0	71
5.1.1	Exploiting 802.11 states	72
5.1.2	Active Probing using ICMP	74
5.2	SIDS - Where Are We At?	75
5.2.1	SIDS as an IDS	76
6	Analytic Detection Techniques	77
6.1	The 802.11e and WMM relationship	78
6.2	Application and TID analysis	79
6.2.1	Application and Protocol Based Detection	79
7	New Proposed Algorithm	81
7.1	The Algorithm	82
7.2	New Features	83
8	Future Work	87
8.1	Active Probing - Beyond Proof of Concept	88
8.2	Completing The IDS	88
9	Conclusion	89
9.1	Simple Intrusion Detection System v2.0	90
9.2	Enhanced Algorithm For QoS Optimized Attacks	91
	References	93
A	Valid Types and Subtypes	95

B SIDS V2.0 code	97
B.1 sidsv2.pl	97
B.2 check.pl	104

List of Figures

2.1	The OSI Reference Model	19
2.2	The 802.11 frame	21
2.3	Overview of the different Control Frames	24
2.4	The different states in 802.11 [1]	26
3.1	The lab setup	39
3.2	Valid (blue) and Invalid (red) Managment Frames in 802.11 states	52
3.3	Active Probing using ICMP packets to verify freeloader attacks	58
4.1	Result of running Passive Detection on FreeloaderAttack.cap - 802.11 state control	65
4.2	Duplicate ICMP replies in FreeloaderAttackPING.cap . . .	66
4.3	SSH packets flagges with background WMM classification .	68

List of Tables

2.1	The different subtype frames of management frames.	23
2.2	The different subtype frames of control frames.	24
3.1	Valid and invalid management frames in 802.11 states	54
6.1	Relationship between TID bits and WMM Classes	78
A.1	Valid type and subtype combinations in 802.11 frames	96

Abbreviations and Acronyms

AC Access Categories

AP Access Point

BSSID Basic Service Set Identifier

CRC Cyclic Redundancy Check

CTS Clear-to-Send

CUWN Cisco Unified Wireless Network

DoS Denial-of-Service

IDS Intrusion Detection System

IEEE Institute of Electrical and Electronics Engineers

IPS Intrusion Prevention System

LAP Lightweight Access Point

LLC Logic Link Control

MAC Media Access Control

MITM Man-in-the-Middle

MSDU MAC Service Data Unit

MPDU MAC Protocol Data Unit

NIC Network Interface Controller

OS Operating System

OSI Open System Interconnection Reference Model

PLCP Physical Layer Convergence Procedure

QoS Quality of Service

RSS Received Signal Strength

RSSI Received Signal Strength Indication

RTS Request-to-Send

RTT Round-Trip-Time

SIDS Simple IDS

STA Station

TID Traffic Identifier

TrT Wireless Trondheim

wIDS Wireless Intrusion Detection System

WLAN Wireless Local Area Network

WLC Wireless LAN Controller

WMM Wi-Fi Multitmedia

Part I

Introduction

1

Introduction

This chapter will give a more in-depth background of the problems provided in the foreword of this thesis. I will first overview the background for wireless Intrusion Detection Systems and their use in open wireless networks. Further I will give a more detailed description of the problems handled in this thesis. Section 1.4 will give an overview of the report outline, and show how the report is built up. Finally, in section 1.5, I will relate to work done by others in this same area of research.

1.1 Background

Wireless networks are deployed widely across the globe today. Open Wireless networks are commonly deployed to cope with user friendliness, easy access as well as easy deployment and scalability. Typical areas of deployments include Airports, Cafes and primarily in public places. Due to the number of increasing users that wants quick access to the internet, there are demands for networks that will give users easy access without too much hassle in the connection and authorization phase.

As a result of this, open 802.11 wireless networks are deployed. These networks offers no security features by the means of the 802.11 standardizations, but they often implement simpler solutions - such as captive portals. Captive portals will pick up every package sent from a proper associated Station (STA) to an Access Point (AP), and return a login page to the client if correct credentials are not given. Once the client enters proper credentials, the captive portal will mark the clients STA Media Access Control (MAC) address on a white list, and will thus forward all traffic to and from the internet to the associated clients STA. The login credentials can be a pre-negotiated username/password combination, or by paying a fee to get temporary access - usually a 24 hour window access.

Unfortunately, as no security or encryption features are implemented, these networks are exposed to several security threats. The goal behind the administrators of these networks are usually not to secure their users or the network from being penetrated, but to make sure that their users get quick and easy access to the internet. Nevertheless, there are demands for systems to detect *if* attacks are carried out. This is the case due to legal matters, were one wants to make sure that if an illegal act has been carried out from a clients credentials, we must determine that it was in fact the

legit client who was unlawful, and not someone else stealing his/her session and credentials. This is known as identity attacks. Systems designed to detect attacks like these are commonly referred to as Intrusion Detection System (IDS). More on these in section 2.3.

1.1.1 Project Assignment 2009

During the mandatory project for our study in Communication Technology fall 2009, myself and Øystein Aas Pedersen developed a simple Intrusion Detection System, named Simple IDS (SIDS). The report is provided in [2]. SIDS is a program written in Perl, and its goal is to be deployed next to open wireless networks to detect if a legit clients connection is, or has been, stolen. SIDS is what is commonly referred to as a Passive Wireless Intrusion Detection System (wIDS). This means that it will not interfere with the current infrastructure of the network, but be deployed in the same geographical area as the network being monitored. SIDS use a technique called MAC Sequence Counting analysis to detect if two STAs using the same MAC address are associated to the same AP at the same time, and is thus a countermeasure to MAC freeloader attacks [2].

During the project we uncovered a potential weakness in regards to Quality of Service (QoS) enabled networks and the use of MAC Sequence Counting as a detection technique. By enabling QoS on the link-layer in these networks, many IDS may become obsolete as the detection techniques are not optimized to cope with new features of the 802.11e amendent, often implemented by Wi-Fi Multimedia (WMM). There has been some work carried out simultaneously with ours, as provided in [3], were the authors suggest using parameters from the physical layer of the Open System Interconnection Reference Model (OSI) as a reliable source for verifying 802.11e optimized attacks described in section 3.2.

Nevertheless, in this master thesis I question how their work will suite a real-life setting, such as for instance deploying the solution in an urban and mobile environment as the one obtained by Wireless Trondheim. As physical parameters are not reliable in a mobile and urban environment, I will in this master thesis suggest new techniques, and hopefully find other ways which are more reliable and suitable for environments were parameters from the physical layer are not reliable.

1.1.2 Wireless Trondheim

Wireless Trondheim (TrT), or Trådløse Trondheim in Norwegian, is an NTNU organization which main goal is to provide easy internet access for the inhabitants and student in the city of Trondheim, Norway. TrT use a distributed architecture for their wireless network, known as Cisco Unified Wireless Network (CUWN). Login credentials and security are provided with the use of Captive Portals. Students can log in with their university login credentials to get free access, and the rest of the inhabitants can pay a small fee to get access for a specific time frame. (Usually 24 hours). Throughout the city there are deployed Lightweight Access Point (LAP)s to cover most of the downtown area as possible, maintained and serviced by Wireless Local Area Network (WLAN) controllers.¹

In this master thesis I will conduct my experiments on the same lab equipment as we used in the project 2009. The lab setup is a direct replica of the CUWN network maintained by Wireless Trondheim. This means that the development of SIDS v2.0 in this master thesis should suit the network topology and deployment of the network maintained by TrT well, and as I will show in the conclusion of this thesis, with further testing and some

¹Wireless Trondheim are well described in the project assignment provided in [2]

rewriting of code, SIDS v2.0 could be a complete IDS deployed in CUWN networks, such as the network maintained by Wireless Trondheim.

1.2 The Problem

MAC sequence counting has proven to be a good and reliable detection technique in 802.11 networks [4]. However, there are complications in optimizing the detection technique to cope with the new features from 802.11e, described in section 2.5. One of the main issues is that most IDS do not distinguish between the different QoS classes, thus concluding that an attack is occurring when a client use link-layer QoS. With the introduction of 802.11e QoS enabled wireless networks, came the feature were multiple transmission queues in sequence counting were allowed. This means that the STA keeps one counter for each of the 8 Traffic Identifier (TID) QoS classes available from 802.11e. The result is that most IDS with MAC Sequence Counting as a reliable technique for detecting will be obsolete. The MAC Sequence Counting algorithms simply are not capable of distinguish between the TID classes. One could think that it should be fair to just keep track of each of the sequence number counters and analyze them independently, but as I will show in section 3.2, this potentially opens up for new attacks.

Chandrasekaran, Deymious, Ganapathy, Gruteser and Trapper [3] suggest using parameters from the physical layer to improve the MAC sequence detection algorithm described in 2.4. Their work focus mainly on measuring Received Signal Strength (RSS) values originating from the same MAC Source address, and calculate the Euclidian distance using received RSS values from a STA between two time intervals. Simultaneously with the writing of this master thesis, Øystein Aas Pedersen is writing his thesis

on the behavior of RSS measurements in a city-like environment. Due to this, this thesis will propose and demonstrate new thinking in the areas of combining MAC Sequence Counting analysis with link-layer QoS enabled wireless networks. I will show that it is not necessary to rely on uncertain physical parameters, such as RSS measurements analysis in order to conclude if attacks are carried out in 802.11e enabled open 802.11 wireless networks.

1.3 Contributions

My contributions in this master thesis are how to implement a trustworthy IDS which does not become unusable when QoS/802.11e is enabled on the network. As most wireless IDS heavily rely on MAC Sequence Counting for detection, these will become obsolete as they will trigger false alarms as a result of a STA using QoS on the network, where in fact there was no attack at all. In a worst case scenario they are not capable of detecting attacks at all, if an attacker knows how to exploit the new features of 802.11e as described in section 3.2. In section 1.5 we will see that related work has been done on this same area of research, but I am questioning their work, and will in propose new thinking and more reliable techniques to cope with the problems QoS/802.11e introduces. The outcome of this thesis will thus be divided into two parts:

(1) SIDS v2.0 *I will improve, implement and test new detection techniques in SIDS v2.0*

(2) Algorithm *I will propose a new algorithm in order to detect QoS optimized attacks without relying on physical parameters.*

1.4 Report Outline

The work done in this master thesis is a continuation of the work carried out by me, Eirik Holgernes, and Øystein Aas Pedersen during the fall 2009. The thesis will keep strong focus the proposed solution provided in [3], as well further development of SIDS in regards to the problem description overviewed in section 1.2. Due to this, the paper will be divided into three parts. **Part I - Introduction** provides the introduction and background needed to understand the problem, as well as given an overview of the report outline and related work. **Part II - Developing an IDS** mainly concentrates on how SIDS v2.0 is developed, and keeps focus on my methods of choice, as well as the results when testing the IDS. **Part III - Discussion and Analysis** will start by discussing the implemented techniques in SIDS, and further concentrate on analytic detection techniques not implemented in SIDS. Finally, as a result of this, I propose an enhanced algorithm for QoS/802.11e optimized attacks.

Chapter 2 provides the *background theory* relevant for the work done in this thesis. The chapter starts with basic knowledge on the 802.11 standard, and further elaborates on attacks in 802.11 networks. Further I will describe the basics behind Intrusion Detection Systems and techniques commonly used by these. The chapter will end by detailing the 802.11e/QoS standard which is important in regards to the work done in this thesis.

Chapter 3 will show my *methods of choice* in the creation of an Intrusion Detection System. The chapter will first define a proper threat model, and continue to show how I programmed SIDS v2.0, as well as propose new techniques, and how these

were implemented in the IDS. This chapter is best read in combination by looking at the code provided in appendix B.

Chapter 4 will show the *results* of the work described in chapter 3. As I have proposed two new techniques for optimizing Intrusion Detection Systems in order to cope with the QoS/802.11e problems described in this introduction, this chapter will show the results when executing SIDS v2.0 in different scenarios.

Chapter 5 will concentrate on the work done on *SIDS v2.0*, and discussions on the implemented techniques in the IDS. The results and source code from SIDS v2.0 will be my first contribution from this master thesis as described in section 1.3.

Chapter 6 will elaborate on *analytic detection techniques* not implemented in the IDS yet. It elaborate on theories and analysis of the background theory chapter 2 in order to propose new analytic detection techniques.

Chapter 7 proposes a *new algorithm* as an epitome of the techniques and methods discussed in chapter 5 and 6. This algorithm is my second contribution in this thesis as shown in section 1.3.

Chapter 8 suggests subjects that are suitable for *future work* as a result of the work done in this thesis. There are several aspects that is not considered in this thesis, and the final outcome will show that two major subjects should be further analyzed; completing the development of SIDS v2.0 and work on Active Probing as a detection technique.

Chapter 9 *concludes* the thesis, and will provide the reader with a summary of the work achieved.

1.5 Related Work

When looking at a particular area of research, it is important to gain knowledge of what others have achieved in the same area. As stated throughout this introduction, I will keep focus on developing an Intrusion Detection System, as well as how to cope with newly introduced problems that came with 802.11e/QoS in regards to IDS in general. In section 1.5.1 I will look at work done on how to use RSS in localization based measurements, and its use in Intrusion Detection Systems monitoring *mobile* environments. Further in section 1.5.2 we will see that work has been done on how to optimize detection techniques to cope with the problems with 802.11e networks and wIDS. Finally, in section 1.5.3 I will look at a specific technique proposed by Jason Smith [5] on how to use protocol and standard specifics to detect abnormalities and attacks in wireless networks.

1.5.1 RSS and IDS

Wilson M. Yeung and Joseph K. Ng paper on "Wireless LAN Positioning based on Received Signal Strength from Mobile device and Access Points" [6] takes a great interest in how to use RSS measurements on mobile environments to measure the geographical location of mobile devices. The paper states that they need to use average RSS values in order to determine a single position, as RSS values are not always conclusive in small time intervals. At an even more interesting level, Øystein Aas Pedersen is writing his thesis on RSS measurements in urban and mobile environments; "Detecting Wireless Identity Spoofs in Urban Settings Based on Received

Signal Strength Measurements" [7]. The results from his work show that RSS is not at all reliable and conclusive when moving away from office landscapes and lab environments, and into urban environments. The rate of failure stated above may be adequate concerning location based services, but Intrusion Detection Systems should rely on reliable and conclusive parameters in order to trigger alarms on attacks. This will be kept in strong focus throughout this thesis, and as we will see, my new proposed algorithm for detecting QoS optimized attacks does not rely on RSS measurements, as opposed to the suggested algorithm from [3].

1.5.2 802.11e and IDS

As described in section 1.1.1, during our semester assignment project fall 2009, myself and Øystein Aas Pedersen discovered the flaw with many Intrusion Detection Systems as overviewed in section 1.2. During the completion of the paper, we discovered a paper which describes and suggests solutions to the 802.11e QoS introduced problems. This paper is given in [3], and this thesis keeps strong focus on their suggested algorithm on how to cope with MAC Sequence Counting in regards to detecting attacks in 802.11e enabled networks. As we will see in section 3.3, the paper suggests using RSS measurements in order to verify if an attack is ongoing. The work done by [6] and [7] from the previous section shows that in mobile, and especially urban environments, RSS values are not always conclusive. As a result of this I will look into how to use techniques which is not dependent on unreliable parameters to detect attacks.

1.5.3 Specification-based IEEE 802.11 Attack Detection

Jason Smith [5], chapter 7, section 7.3 proposes to look at specifications from the 802.11 and 802.1X² standards to detect attacks. The paper keeps strong focus on Denial-of-Service (DoS) attacks, but many of the ideas and methods are interesting in my setting as well. Smith exploits how states in 802.1X, EAP works, by looking at expected management frames to detect abnormalities in the authentication and association phase of clients. This thesis will use some of the basics behind this detection technique, but will instead concentrate on the three 802.11 states specified in section 2.1.3 which is used in open 802.11 wireless networks. The ideas with the use of behavior based detection technique in regards to how IEEE 802.11 operates is interesting, and we will see that I will use some of the same structure and thinking as Jason Smith, but in a less complicated matter.

²802.1X provides authentication features in 802.11 networks

CHAPTER 1. INTRODUCTION

Part II

Developing an IDS

2

Background Theory

In order to understand the methods and ideas in this thesis, one should have a thorough understanding of some basic background theory. This chapter will provide the basic background theory necessary in order to understand the work done in the subsequent chapters. First I will give a short introduction to the IEEE 802.11 standard, and relate it to different attacks in 802.11 networks. Further, in section 2.3 I will provide a quick overview of the ideas behind Intrusion Detection Systems. To understand the detection techniques in regards to these, I will look at different detection utilized by IDS in section 2.4. As 802.11e is an important part of this thesis, I will provide a detailed introduction of the amendment in section 2.5. Finally, a

quick look at the RSS values and their use is provided in section 2.6.

2.1 IEEE 802.11

802.11 is a standard formalized by the Institute of Electrical and Electronics Engineers (IEEE), and is detailed in [1]. In short terms, the idea behind 802.11 is to provide a set of standards to make wireless networks work across different platforms, as well as between different vendors. 802.11 networks, or WLANs, are deployed widely across the globe, both in private homes as well as in public areas. In my thesis I will keep focus on the latter. There are many different standards and variations of the 802.11 standard set, but I will keep my focus on the actual frames and frame types used by 802.11, also known as the MAC frame. This was last defined in the 802.11-2007 [1]. 802.11 operates in the Data link layer and the physical layer of the OSI model provided in figure 2.1. The Data Link layer is a combination of a Logic Link Control (LLC) layer and a MAC layer. This basically means the layer used to link up two stations, and how they should communicate to each other. This is the final layer before the actual physical data is sent through a physical channel, such as Ethernet or by radio in wireless networks.

2.1.1 Ad-Hoc And Infrastructure

Wireless Networks can either be deployed in Ad-Hoc- or infrastructure mode. In Ad-Hoc mode every STA acts as an AP, and every AP is a STA. This setup does not require a centralized entity managing the clients, but every client manages itself and those connected to it. In infrastructure wireless networks, there is a specific set of hardware which provides a set of services to clients, commonly explained as AP serving STAs. In this scheme

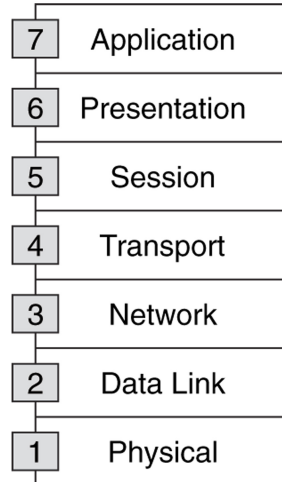


Figure 2.1: The OSI Reference Model

one AP can serve many clients, or STAs, often providing access to a wired network, like a backbone network, or as a bridge to the internet. This is a centralized approach, and due to the common deployment of these, this master thesis will focus on the latter case.

2.1.2 The 802.11 MAC Frame

All communication in 802.11 environments between APs and STAs are exchanged by using three different types of frames. Which frame type to be used depends on what the purpose of the communication is. The different use of frames types and subtypes depends on how to exchange data, as well as establishing and maintaining the wireless link between the two nodes. Every 802.11 frame consists of which 802.11 version is used, the frame type, as well as various indicators and flags set for the specific frame. Also com-

mon for each 802.11 frame is that every frame consists of the source and destination MAC address, sequence number, frame body and a frame Cyclic Redundancy Check (CRC) control field. This is further explained below. An overview of how the frames are built up are given in figure 2.2. The use of fields will vary depending on which frame types are used, but the setup scheme is always the same. Figure 2.2 does not show any other layers than LLC or MAC layer, but before each frame there also exist a preamble frame and a Physical Layer Convergence Procedure (PLCP) Header, but their use are meant for physical layer purposes. For a complete list of the type of frames and their subtypes, see table A.1 in appendix A.

Explanation Of The 802.11 MAC frame The first field is the frame control, and consists of two -2- bytes. This is an important field in regards to monitoring and analyzing 802.11 frames. The field will tell you different parameters about the frame, like which type it is, as well as other important parameters about this specific frame. Duration is the calculated value from a STAs perspective on how long it will take until the frame is transmitted. Address 1, 2, 3 and 4 are MAC addresses. Address 1 is always the address of the originating node, address 2 is the destination address, and address 3 is in most cases the address to the final destination inside the backbone network. The 4th address is an optional one. All these addresses are written as 48 bit hex format, commonly known as MAC addresses. Seq is a field providing sequence number for each frame delivered between two stations. More on sequence numbers in the next paragraph. Finally the actual data or payload is inserted. This field is variable, but can be a maximum of 2312 bytes per frame. The 4-byte checksum field exists for error control, such as CRC check in order to make sure that the bits in the frame were received correctly. Below are given an overview of the different frame types used in

802.11, how they are built up, and when they are used [1] [8].

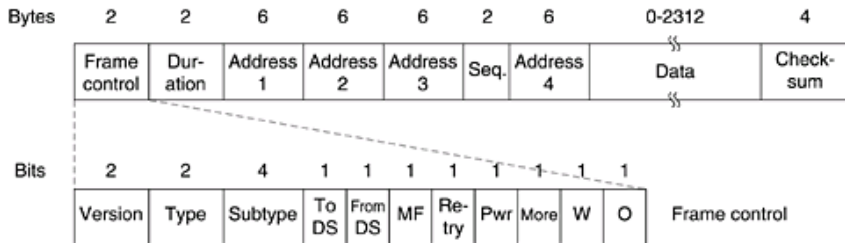


Figure 2.2: The 802.11 frame

Sequence Numbers In 802.11 Frames In 802.11 environments, as defined in the 802.11-2007 standard [1], the transmitting party always keeps track of a unique sequence number for each frame successfully transmitted to the receiving party. This means that a STA transmitting a 802.11 frame to an AP, keeps track of a sequence number counter for each frame it sends to that specific AP. The sequence number field from figure 2.2 shows that there are reserved 2 bytes, or 16 bits, for sequence control. 4 of these are used for fragmentation purposes to ensure that every MAC Service Data Unit (MSDU) or MAC Protocol Data Unit (MPDU) have one specific fragment number until all pieces of the packet are correctly delivered.

The remaining 12 of these are reserved to provide sequence number control of frames independent of which fragment or type of frame it is. As a result of this there can be a maximum of 2^{12} different combination of sequence numbers, which in human terms read 4096. Thus, the station marks the first frame with zero -0-, increase the sequence number field with one -1- for every frame, wrapping around 4096, and begins the procedure all over. Thus, the STA and AP keeping track of fragmentation of data, as

well as the frames being received in correctly order. Also note that sequence control fields is not present in control frames, as they are meant only for simple control communication, and are not sequential important.

Management Frames

The purpose of management frames are to establish and maintain the wireless link between the AP and STAs. It is not just important to establish a link between STA and AP, but also to maintaining the link in terms of making sure that the STA are still there, as well as other link specific parameters to obtain a good connection. Management frames have many subtypes, and in table 2.1 I have given a quick overview of the different subtypes and when they are used.

Control Frames

The control frames are used in assistance in delivery of data frames. Control frames are used in order to avoid frame collision due to the hidden station problem¹, as well as acknowledging when frames are transmitted correctly. The different types of control frames are explained in table2.2 and provided in figure 2.3. As described in section 2.1.2, the sequence control field is not present in control frames. Also note that the network does not have to enable RTS/CTS due to traffic overhead, resulting in the network not answering on RTS control messages.

¹The hidden station problem is when two STAs associated to the same AP don't see each other resulting in frame collisions

Frame Subtype	Description
Authentication Frame	These frames are used in the process of accepting or rejecting the radio interface of a Network Interface Controller (NIC). The procedure for using these frames depends on which security parameters are used. For my concern I consider open 802.11 wireless networks with no encryption from the 802.11 standard set, so two authentication frames are sent in order to authenticate
Deauthentication Frame	These frames are used when a station wishes to terminate potential secure sessions.
Association Request Frame	These frames are used when a STA wishes to associate its NIC with the NIC of the AP. The frame contains parameters of the NIC, what is supported, as well as the SSID of the network it wishes to associate with.
Association Response Frame	Responses to the Association Request Frame. Approval/Disapproval of the NIC.
Reassociation request frame	If an STA roams between APs finding a better AP in terms of signal strength for instance, it will issue a reassociation request frame to the new AP.
Reassociation response frame	The AP accept or rejects the Reassociation Frame from above.
Disassociation frame	This is used to terminate the association.
Beacon Frame	AP periodically can broadcast its existence and relay information. Includes parameters like SSID, timestamp and so forth.
Probe request frame	Used by one NIC to probe other NICs. For instance probe APs in the area to reveal their existence.
Probe response frame	Response to the probe request frame above. Include information like capability information, supported data rates etc.

Table 2.1: The different subtype frames of management frames.

Data Frames

Data frames encapsulate payloads from a higher layer in the OSI model, and sends them between stations. These are the frames which consist of the actual data. The frame consists of a frame control field, the 4 address fields, the frame body and a frame check sequence field. The payload can be up to 2312 bytes long, which means that most information needs fragmentation of many data frames in a row to complete sending all the information. In

CHAPTER 2. BACKGROUND THEORY

Frame Subtype	Description
Request to Send (RTS) frame	To avoid collisions in the delivery of data frames in the hidden station problem, the RTS/CTS combination is optional in the wireless environment.
Clear to Send (CTS) frame	Response to a RTS frame. Together with RTS this makes a two-way handshake before issuing data frames.
Acknowledge (ACK) frame	When receiving a data frame, the receiving station will perform error control on the received frame, and acknowledge the frame if no errors are calculated. If the sending station does not receive an ACK for a period of time, it will retransmit the frame.

Table 2.2: The different subtype frames of control frames.

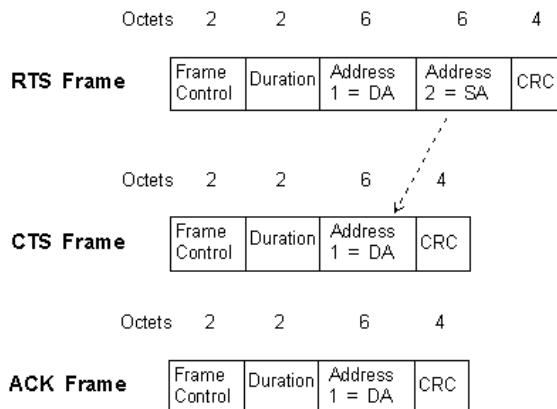


Figure 2.3: Overview of the different Control Frames

streaming of content, i.e. UDP, a continuation of fragmentation and sending is required. Therefore Data Frames rely strictly on the order in which frames are delivered, where fragmentation ID, control frames and sequence number play a crucial role in order to successfully transmit information between the STA and AP. Data frames follow the exact frame setup as provided in figure 2.2.

2.1.3 802.11 States

The 802.11-2007 standard [1] chapter 11.3 shows that in 802.11 environments each STA keeps track of states for each AP (or another STA in non infrastructure mode) it is connected to and vice versa. There are several messages that can be exchanged between a STA and AP in the authentication and association phase. Each AP keeps a state of every STA that has been used or is being used against the AP. The different states and the relationship between them are shown in figure 2.4. The initial state is where a STA is not authenticated or associated with the AP. When a STA wants to associate to an AP, it first sends out an authentication frame as described in 2.1.2. This is to inform the AP that it wants to associate, and that every parameter is in order. This is marked as State two -2- in the figure below. If the authentication response is successful, the STA can go ahead and send an association request to the AP, and if the Association request is successful, the AP is proper authenticated and associated with the AP; State 3. If the STA sends a deauthenticate frame to the AP while in state 3, the AP will also disassociate the STA, bringing the state back to state zero -0-.

- State 1 - Initial state, unauthenticated, unassociated
- State 2 - Authenticated, not associated
- State 3 - Authenticated and Associated

In figure 2.4 not all messages are shown. There are a total of 7 management frames that can issue a state change. In addition to the Authenticate, Associate, Deauthenticate, disassociate and deauthenticate frames shown in the figure, a STA can also issue re-associate and re-authenticate frames if

the STA is roaming between APs, or for some other reason the association or authentication is lost.

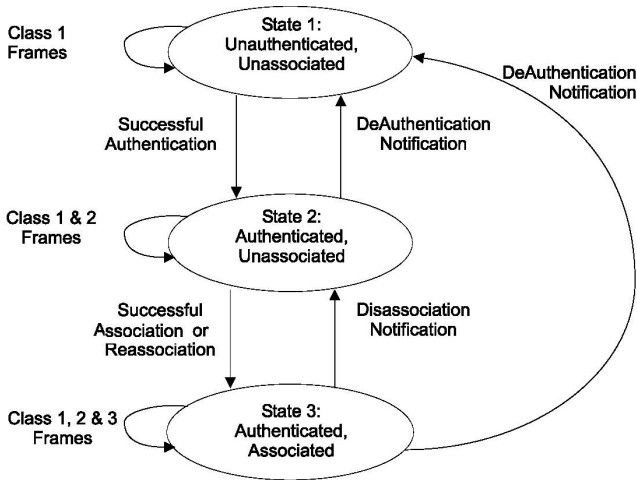


Figure 2.4: The different states in 802.11 [1]

2.2 Attacks In IEEE 802.11 Environments

There are numerous different attacks that can be carried out in wireless networks. A potential flaw with wireless networks, contrary to wired networks, is that potentially everyone can hear everything at all times. Thus, the access of data is more accessible than on closed wired networks. [9] has divided the different attacks into eight categories, namely:

1. Network Discovery;
2. Passive Eavesdropping and Traffic Analysis;
3. Message Injection and Active Eavesdropping;

4. Message Deletion and Interception;
5. Masquerading: Malicious AP and Session Hijacking;
6. Man-in-The-Middle;
7. Denial of Service (DoS) attacks;
8. IEEE 802.11i specific attacks

According to my threat model as described in section 3.1, I will in this thesis focus on two of these. Passive Eavesdropping and Traffic Analysis and Masquerading: Malicious AP and Session Hijacking as defined in my threat model chapter 3, section 3.1. It is important to distinguish between active and passive attacks. Passive attacks means that the attacker acts only as an observer to the network, and does not interact with the system at all. Thus, passive attacks are almost impossible to detect in wireless environments. In active attacks, the attacker's goal is to gain access to network resources, issue Denial-of-service attacks or change some of the topology or security mechanisms of the network. These attacks can be detected.

2.2.1 Eavesdropping And Traffic Analysis

This is a passive attack, and they are usually easy to perform. Here, the attacker use some sniffing tool² on his hardware, picking up the same information as the AP or STA if he is within the same geographical area as the network information he wants to analyze. Thus, eavesdropping is usually a simple matter to perform, but the actual traffic analysis can be more complex depending on if the information is encrypted, as well as skills

²Wireshark, airodump etc

in terms of know-how of how the 802.11 protocols operate. In most cases, the attacker use eavesdropping and traffic analysis attack in combination with other types of attacks to achieve his/her goal. This attack is often the starting point from an attackers perspective in order to gather information about clients, as well as other network parameters [9].

2.2.2 Masquerading: Session Hijacking

This attack is an active one, here the attacker's goal is to imitate a legitimate AP or steal and imitate the characteristics of a legit STA. Again, according to the threat model described in 3.1, I will keep focus on the latter case. After performing the attack described above, the attacker gathers information about STAs and network topology, and can for instance change his MAC address to that of a valid user. This is a MAC Spoofing attack, and as discussed earlier I will keep strong focus on detection of these. The attack described in the previous section is also important in regards to gather information about Quality-of-Service parameters as focused on in this thesis. In open 802.11 wireless networks, no traffic is encrypted in the MAC/LLC-layer, thus resulting in the attacker reading all frame information in the same fashion as the AP or STA.

2.3 Intrusion Detection Systems

Intrusion Detection Systems are systems designed primarily to detect abnormalities in penetration of the network, or any of its clients. These systems are in contrary to Intrusion Prevention Systems, or IPS, which goal is to detect *and* prevent attacks. IDS commonly include various detection techniques and combinations of these in order to conclude that an attack has occurred - or is occurring. Usually one categorizes IDS in four types [10]:

- Network Based;
- Network Behavior Analysis (NBA);
- Host Based;
- Wireless

In this thesis I will look explicitly on the last one, Wireless Intrusion Detection Systems. Wireless IDS can be further divided into two categories; Passive and Active. Passive IDS are deployed next to the wireless medium in which it wants to monitor, and rely on eavesdropping and traffic analysis to detect attacks. In active IDS, the system may take an active part of the network; usually in terms of probing STAs or other network nodes either upon suspicion of attacks or on a random or regular basis. A good overview of wireless intrusion detection systems can be found in [11] and in our project 2009 [2].

2.4 Detection Techniques

Many techniques for detecting attacks have been proposed earlier. Below I have categorized detection techniques that have received attention in the security community, as well as others in which I am considering in this thesis. For a detection technique to be good, it is of high importance to keep the false positives as well as false negatives to a minimum. One has to make sure that the system in fact triggers alarms upon attacks, as well as trust the system to raise alarms with high credibility. In the latter case, the system may become unusable, cracking under the weight of number of false alarms.

There is a common understanding that one detection technique usually is not good at its own. The key is a combination of different techniques to increase the level of credibility of the detection, and many Wireless IDS rely on strong statistical and probability calculations together with combinations of techniques to trigger alarms of attacks [10].

2.4.1 MAC Sequence Control Analysis

As described in section 2.1.2 and 2.5.3 there exists a wrapping sequence counter for each successful frame transmitted between a STA and AP. If only one unique STA is connected to the AP the sequence number for each frame should be increased by one -1- for every frame delivered. Nevertheless, if a masquerading attack is performed as described in section 2.2.2, where the MAC address has been spoofed, one should see two simultaneously counters originating from the same source MAC address. This could be an indication of an attack, but as we will see further in this thesis, there are complications in this detection technique in QoS enabled 802.11 networks.

2.4.2 RSS Monitoring And Physical Parameter Analysis

This method includes parameters from the physical layer of the OSI model provided in figure 2.1. As vendors include ways for their NIC to measure the signal strength between STA and AP, the IDS could analyze these measurements to see if RSS measurements are oscillating between two or more values within a short time frame. This could be an indication of an attack, where the same source STA is geographically located at two different places - which of course is impossible. The great advantage with this technique is that it is not protocol dependent, meaning that it does not have to consider protocol specifics from higher layers of the OSI model, and is thus easy to

implement and does not care about encryption, security features, QoS or other protocol specific parameters. The disadvantage is that the measurements are not always conclusive, as well as manufactures and vendors may use different ways for measuring this value.

2.4.3 Traffic Pattern Analysis

This detection technique can be quite advanced. This is also an example of a passive detection technique, where the IDS analysis traffic and compare this with what the IDS thinks is normal traffic behavior. This technique usually requires advanced coding skills, as well as challenges in logistic and statistical analysis to determine what goes under the term *normal behavior*. Usually this is implemented to cope with Denial-of-Service attacks, or other attacks that requires the attacker to be generating abnormal traffic flows in terms of type and amount of traffic.

2.4.4 RTS/CTS Roundtrip Analysis

As shown in table 2.2, two of the control frames are called Clear-to-Send and Request-to-Send. If this is enabled in the network, a STA will issue a RTS frame, and wait for a CTS from the AP before transmitting data. A detection technique called CTS/RTS Round-Trip-Time (RTT) analysis can use this protocol specific feature in order to determine if an attack has occurred. An IDS can calculate the time spent before it sees a RTS frame until the CTS is received on the IDS, and see if this differs a lot from the last time this was analyzed. In the same fashion as RSS monitoring analysis, it can determine if a STA source address is located at two different geographical places at the same time. Nevertheless, not all networks enable

CTS/RTS, as a result of keeping network overhead traffic to a minimum - including Wireless Trondheim.

2.4.5 Protocol Specific Analysis

This category of detection technique is quite vague. In short terms meaning the general concept of looking at protocol or standard specifics to see if the network traffic corresponds with how the protocol or standard is expected to behave. This is best described with an example, and as we will see in section 3.5.1, I have exploited how the 802.11 states operate by looking at management frames originating from STAs to detect abnormalities in the association and authentication phase of 802.11 networks.

2.5 IEEE 802.11e

802.11e is an amendment to the IEEE 802.11-2007 [1] standard. The main principle behind 802.11e is to provide Quality of Service on 802.11 frames between APs and STAs. This is achieved by letting the applications on the STA inject the 802.11 frames to set a specific quality class on the frame before it is sent to the AP. If the AP, or the wireless environment, has a 802.11e/QoS scheme enabled, it will prioritize the frames based on which QoS class the clients STA has given his/her packets. Areas of use includes for instance that a client can prioritize a conference call, such as Skype, and pre-define these frames on a higher QoS class then for instance sending a simple e-mail. This will ensure that all frames that are used for Skype purposes will be prioritized in the wireless network, providing the user with a smoother conference call not disturbed by less time-critical operations such as shipping an email or downloading a file.

2.5.1 Wireless Multimedia Management (WMM)

WMM is a Wi-Fi alliance interoperability certification, and is based on the IEEE 802.11e amendment [12] [1]. Instead of using all eight TID classes available from the 802.11e amendment, WMM provide four different classes, or Access Categories (AC), namely; voice, video, best effort and background. This is reflected to 4 different combinations of TID bits, and the remaining 4 is not used. The STA application sets the TID bits accordingly to reflect the payload of the frame, if it's a voice, video, Best Effort or background frame. Voice, such as Skype, has the highest priority, and next comes video, then standard best-effort for applications that does not support or use WMM, and finally background applications like file downloads and print jobs [12]. *It is important to understand that it is the application or protocol used that sets these priority policies and not the WMM in itself.* Below are three requirements in order for WMM to be in place³:

1. the access point is Wi-Fi CERTIFIED for WMM and has WMM enabled;
2. the client (device) that the application is running on must be Wi-Fi CERTIFIED for WMM;
3. the source application supports WMM

2.5.2 The QoS Control Field

As detailed in section 2.1, each 802.11 frame where the QoS subfield field is set to 1, has a QoS field (See table A.1). This field consists of 2 bytes, or

³List taken from <http://www.smallnetbuilder.com/wireless/wireless-features/30833-does-wi-fi-multimedia-wmm-really-do-anything-part-1>

16 bits. The first four bits are reserved for the TID subfield, and is always used to set priority classes, or TID classes on 802.11 frames. The next fields are used for QoS related matters. For further reading of this please review section 7.1.3.5 in the IEEE 802.11-2007 [1]. If no QoS are used these bits are always set to zero -0-. As a result of this there can be a total of 8 different QoS, or TID classes per 802.11 frame. The use of QoS and 802.11e depends on which Quality-of-service scheme offered by the STA and the network environment. 802.11e simply provides a framework for Quality-of-service on the link between STA and AP.

2.5.3 Sequence Numbers and 802.11e

As explained in 2.2, for management and data frames there exists a sequence control field in order for the transmitting and receiving party in 802.11 environments to ensure correct delivery of MSDU or MMPDU in sequential order. If the QoS subfield is set to zero -0- (see table A.1), and no QoS is used, the STA keeps track of one sequence number counter for all Data and management frames sent to the associated AP. *However, if the QoS subfield is set to one -1-, and the QoS field exists, the STA keeps track of one sequence number counter pr TID class pr unique receiver.* This is of high importance in this master thesis, and the problems described in section 1.2 and 3.2 is a direct result of this. There also exist another wrapping counter for management frames and QoS data frames with a broadcast/multicast address in the Address 1 field, as well as all non-QoS data frames sent by QoS STAs.

2.6 The Physical Layer

Even though this thesis primarily focus on the data link-layer and MAC layer - layer 2 in the OSI Reference Model provided in figure 2.1, I will give a brief introduction to measurements of RSS values on the physical layer. RSSI stands for Received Signal Strength Indication, and is a value measuring the power in a received signal. The calculations of this value are vendor specific, and are unit less in the range from 0 to 255. The max value is also vendor specific, where i.e. Cisco cards range the RSSI value from 0-100, and for instance Atheros⁴ range from 0 - 127. The value is acquired on the receiving STA from the preamble frame of 802.11 frames, which is attached on the frame before the actual 802.11 MAC frame from figure 2.2 is sent. For more in-depth theory and analysis of RSS, please refer to the master thesis by Øystein Aas Pedersen provided in [7].

⁴Atheros is a developer of semiconductor system solutions for wireless and other networking products.

CHAPTER 2. BACKGROUND THEORY

3

Method

With all the background theory in place, I will in this chapter show my methods of choice in developing an Intrusion Detection System. The chapter will start by defining a proper threat model. This is important in order to understand to which threats we are trying to protect against. Further, in section 3.2, I will go in to detail on how QoS optimized attack work in theory, as well as describing them with a simple scenario. As stated in section 1.5.2, this thesis will keep strong focus on the proposed solutions and algorithm from [3], and I will in section 3.3 go in to detail on these. How SIDS v2.0 was developed, and which enhancements the system provides in contrast to the proposed algorithm are provided in sections 3.4 and 3.5. Finally, in

section 3.6, I will describe how I created different scenarios to test SIDS v2.0 on.

3.1 Threat Model

In order to understand what we are trying to protect us against, it is important to define a proper threat model. In this section I will provide a threat framework to highlight the threats which I will consider in this thesis, and briefly discuss which threats I am not considering. Recapping back to section 2.2, the attacks on 802.11 environments can be extensive, and some quite advanced. We must define an attacker in terms of motive, skills, know-how and equipment, as well as look into the actual network which needs protection. The lab setup from the project assignment from 2009 are provided in figure 3.1. As stated in the introduction of this thesis, this network is a replica of the CUWN network Wireless Trondheim has deployed in the city of Trondheim. The wired part of the network should be secured by locks and physical access, and the amount of work needed to gain access from the internet to the administrating network should be highly extensive. Thus, I will only consider the following concerning network access:

(1) Network *The wired network is considered secure, where the wireless network is not.*

This means, that in figure 3.1, I will consider threats on LAP1 and LAP2, as well as wireless STAs. Further we need to define the attacker in matters of motive, skills, know-how and equipment. As described in the introduction, an attackers motive to carry out an attack in these types of network can be for self-pleasure, or in a worse case perform illegal actions

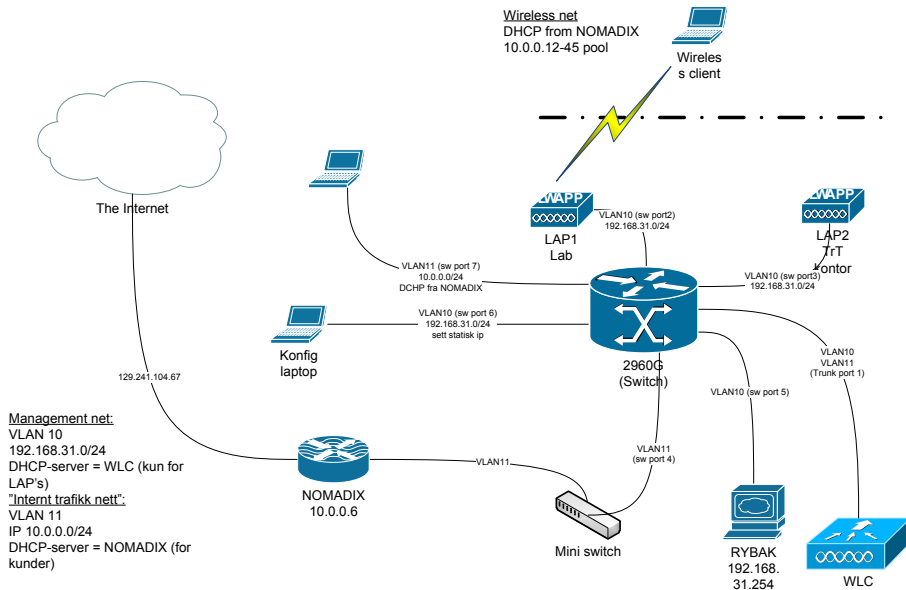


Figure 3.1: The lab setup

masquerading a legitimate and innocent clients credentials. (I.e. downloading of illegal content, such as child-pornography). The motive could also be for own satisfaction in regards to be able to fool an advanced intrusion detection system.

(2) Motive *The attacker's motive is to spoof a legitimate identity, and use his or her credentials to access the network and internet.*

In terms of skills, we must define how much work and skill-set the attacker possesses accordingly to the motive behind the attack. It is reason-

able to conclude, that the personal selfish gain from just penetrating an open 802.11 wireless network should be low for an attacker with high level of skills. Nevertheless, as a more sophisticated intrusion detection systems are deployed, the credit gained by fooling the IDS should be also considered. As we will see in section 3.6.2, MAC freeloader attacks are in its simplest form, and easy to carry out. So this should be our first type of attacker. The second one could be of an attacker that wants to bypass the system as a proof-of-concept, or to frame a client or perform illegal actions on the net.

(3) Skills *The common attacker has the skills to carry out a MAC freeloader attack, and understand the basics behind the 802.11 protocol. I Will also consider a more advanced attacker, whose goal is to bypass the system, with thorough understanding of 802.11 wireless networks and Intrusion Detection Systems.*

As described in section 2.2, I will consider two types of attacks in this thesis, namely eavesdropping(2.2.1) and masquerading(2.2.2) attacks. Eavesdropping is a simple gesture, but important in an attackers perspective to gain knowledge of clients, network topology and parameters. Masquerading the clients STA in open 802.11 networks is a matter of simply changing the MAC address to that of the legitimate client. A combination of these two attacks can result in a MAC freeloader attack, and the equipment needed for this is a simple laptop with a wireless network card.

(4) Equipment *The attacker has limited access to advanced equipment. A laptop with a wireless NIC is considered.*

3.2 QoS optimized Attack

As briefly explained in the introduction section 1.2, an advanced attacker can fool the detection system by exploiting how the Intrusion Detection System cope with the QoS problems described earlier. As described in the theory chapter 2 section 2.5, 802.11e use 8- TID QoS classes, and the STA keeps one sequence counter for each of these classes. The first problem with this approach is that IDS based on standard MAC Sequence Counting can't distinguish between the different counters for each class, thus not being able to detect which sequence number originates from the legitimate STA, and a potential attacker. It is reasonable to believe that the IDS can counter this by looking at the TID bits in each frame, and keep one counter for each QoS class read, and as a result of this trigger alarms when one or more of the independent 8- sequence counters jumps between values.

The Attack However, if an attacker analyses which TID bits that are in use by the legitimate STA (Usually none or only a few combinations as we will see in section 6.1), it can inject his/her frames to simply use one or more of the other classes not used by the legit clients STA. This will result in the detection system again not being able to distinguish which frame belongs to the client and a potential attacker. This is a result of the IDS observing a standard MAC sequence counter increase in a specific TID class, and there is no way to understand that it was in fact an attacker masquerading as the proper clients STA using this TID class.

Simple Scenario The explanation above may be difficult to understand; so I will further try to explain this with a simple scenario. We can think of Alice as a legitimate client surfing the web with an iPhone, on a bus,

through TrT's network. In this case she only uses Safari on her iPhone surfing the web. On a café nearby sits Bob. He has installed Wireshark, and is analyzing 802.11 frames that are sent in the clear on the network. Bob sees Alice's MAC address, and sees that all frames from Alice are marked with TID bits 000 - Best Effort. Bob spoofs Alice MAC address, and follows a standard MAC Freeloader attack as explained in Ørjans master thesis [13]. As Bob has quite the skills, he uses iptables¹ on his Linux distribution to make sure that every frame sent from his laptop is marked as TID class 010 (Video). Further, Bob, uses his newly achieved connection on TrT's network to issue an attack on a highly important mainframe somewhere. The IDS did not trigger an attack, as it believed that Alice simply was streaming video content. Alice now has to take blame for Bob's actions.

3.3 Proposed Algorithm For QoS Considered Detection

The algorithm proposed by [3] is given below, and suggests new techniques as a counter to the attacks described above. The work done by developing SIDS v2.0 uses the ideas and structure of this algorithm. As SIDS grew and became more advanced during the writing of this master, I found it necessary to structure the IDS in a more programmer friendly fashion. As a result of this I used the algorithm from [3] as a starting point for developing SIDS v2.0, and divided the algorithm into several steps, or blocks, to easy compare the intrusion detection system to the algorithm itself and explanations given in this thesis. These steps are given as functions, or subroutines,

¹iptables is a user space application program that allows a system administrator to configure the tables provided by the Linux kernel firewall and the chains and rules it stores

3.3. PROPOSED ALGORITHM FOR QoS CONSIDERED DETECTION

in SIDS, as I will show in section 3.4.

```
Algorithm: Find-Identity-Spoofs
Input    : S: Sequence of wireless packets
Output   : Attack/No_Attack
MACs = list of MAC sequence numbers in S;
if MACs in linear progression then
    return No_Attack;
else if MAC variation in valid range then
    return No Attack;

/* MAC sequences not in linear
progression; check frame types */

FTypes = frame types extracted from S;
if FTypes 2 {Management, Regular Data} then
    return Attack;

/* Frame type must be QoS-Data; examine
priorities */

QoS-Priorities = QoS priorities extracted from S;
if QoS-Priorities are all the same then
    return Attack;

/* QoS priorities are either mixed, or
mixed QoS-data and regular data */

Perform differential localization for packets in S;
if Euclidean distance between successive packets exceeds
threshold then
    return Attack;
return No_Attack;
```

Listing 3.1: Proposed algorithm for QoS considered identity spoof detection

To understand the algorithm itself, as well as relate the work done on SIDS v2.0 to the algorithm, I have sub-divided the algorithm into several steps. In this section I will give an in-depth analysis of the algorithm as pro-

posed by [3], and show how I implemented it in SIDS v2.0. I will also show my new thinking and methods to optimize the IDS. Below I have given an overview of the different steps, and from step 2a to 2d, SIDS does not differ from the proposed algorithm from [3]. We will see that instead of relying on parameters from the physical layer, I will propose techniques which exploit features in how the 802.11 protocol operates. This will be further described in section 3.5.2. I will also propose another passive technique, which I have called 802.11 state control, which is not directly related to the work done by [3], but should be a good extra detection technique as I will further discuss in section 5.1.1.

3.3.1 Initial step

```
MACs = list of MAC sequence numbers in S
```

First we want to keep a sequence of wireless packets; this should be the input to the IDS. Second we keep a list of different parameters for each STA connected to our wireless environment, sorted uniquely by MAC addresses.

3.3.2 Step 2a

```
if MACs in linear progression then  
return No_attack;
```

This, together with Step 2b, is what most IDS based on MAC Sequence Counting are capable of doing. If the MAC sequence numbers are not linear, it will return an attack alarm for the specific MAC address. This step just verifies if all the sequence numbers originating from the same MAC source

3.3. PROPOSED ALGORITHM FOR QoS CONSIDERED DETECTION

address are increasing by one -1-, and concludes no attack if this is the case. To cope with the QoS problems discussed in chapter 1, section 1.2, this algorithm does not conclude an attack yet, and eventually jumps to step 2c. First a quick look at Step 2b.

3.3.3 Step 2b

```
else if MAC variation in valid range then  
return No_Attack;
```

As known, wireless packets can be lost, and as many wIDS are deployed to operate as passive detection systems, there can be frames that made it to their intended destination, but the IDS itself was not capable of picking up the frame. This will result in gaps in the sequence number analysis, and the algorithm must take this into consideration. Therefore it also concludes that no attack is occurring if the sequence numbers gaps are within an approved threshold. If this is not the case, it will further jump to Step 2c. Recapping to our work in the project assignment 2009, I will set this threshold offset to five -5- [2].

3.3.4 Step 2c

```
if FTypes is Management or Regular Data then  
return Attack;
```

If we are on step 2c, we know with certainty that the frames from this MAC address are not in linear progression. If the frames are not QoS Data, and they are in fact normal management or data frames, we may assume

that an attack has occurred. Recapping back to section 2.5 it is only if QoS are used on the frames were we could potentially see different TID classes, and variations in the sequence number higher than of the normal threshold set in step 2b. Therefore we can conclude that an attack has in fact taken place if the sequence numbers are more offset than the approved threshold, as well as the frames are not of type QoS or control frames as explained in 2.1.2.

3.3.5 Step 2d

```
if QoS-Priorities are all the same then  
return Attack;
```

At this stage we know that the frames are not in linear progression, as well as type QoS. If QoS are enabled on the network, the frames could still be marked as QoS frames as explained in 2.1.2, and therefore the algorithm must verify if this is in fact the case. It will check to see if QoS is enabled by the network, but not used. In this specific case, all TID bits in the 802.11 frame are set to zero -0-, and one will conclude an attack in the same fashion as in Step 2c.

3.3.6 Step 2e

```
if Euclidean distance between  
successive packets exceeds threshold then  
return Attack;
```

3.3. PROPOSED ALGORITHM FOR QoS CONSIDERED DETECTION

At this stage, we know that the QoS priorities, or TID classes, are either mixed, or we have mixed QoS-data and regular data within the same sequence. At this stage many IDS based on MAC Sequence Counting are confused, overwhelming the administrator or log file with false positives of an ongoing attack - including SIDS v1.0! The reason for this is that, as mentioned in 2.5, the STA keep track of one sequence number per TID channel, and if the IDS does not keep track of the TID channels together with the sequence numbers it will become useless as described in section 3.2.

This is where everything gets more interesting. As the algorithm proposes, they will use Euclidean distance to verify if an attack is indeed ongoing. These calculations heavily rely on utilizing physical parameters. As these parameters are unreliable in mobile environments, and has yet to be proven as a good detection technique, I will propose new thinking in the areas of how to increase the reliability of the algorithm from [3], by not relying on physical parameters, but exploiting features of the 802.11 protocol. In section 3.5.2 I will show a real-time dependent technique known as active probing, using ICMP packets to verify if a freeloader attack indeed is ongoing. I will also in section 3.5.1 exploit how the 802.11 protocol operates, by constantly analyzing and listening to specific management frames to check if two STAs are communicating with an AP at the same time. Before I go into detail, I will first show how SIDS v2.0 is programmed to suit the algorithm from [3], and then show how these new techniques are implemented in SIDS v2.0. In chapter 6 "Analytic Detection Techniques" I will elaborate on other types of detection which could be an enhancement to the techniques implemented below.

3.4 Developing SIDS v2.0

SIDS v2.0 is developed for experimental purposes at this stage. During the writing of this master thesis, I have used SIDS as a testing framework for new techniques to analyze the behavior when the script was executed on the same offline cap files every time. As explained in 3.3, SIDS v2.0 was programmed according to the algorithm from section 3.3. The main program, known as `sidsv2.pl` provided in B.1, has primarily three functions; *1.* It does the actual frame sniffing. *2.* it calls an external perl script if it calculates that a frame sequence check is necessary, and *3.* keeps track of management frames as I will elaborate on in section 3.5.1. SIDS is able to perform package sniffing in both offline and online mode. To test new techniques and methods and get comparable results, I have performed all of my testing in an offline mode. This means that I have used hardware equipment meant for SIDS to do the actual sniffing of packets, creating offline capture files. ². To keep good track of things, I have used the same offline capture files every time as shown in the Results chapter 4.

The sequence length of MAC frames to keep before performing a check is really up to the administrator of the network. As I performed all my measurements in offline mode, I just set the sequence to 55 000 packets, approximately the length of a 20 minutes CAP file. (Of course filtered on only one specific BSSID, as well as on only QoS, Management and Data frames). In a real-time running of the program I recon this number to be set somewhere between 100 - 500, though not tested and verified. For each frame intended for the specific AP to be monitored, a few parameters about the frame are written to a file on the disk stored as "`<sourcemac>.check`",

²CAP files are files that are formatted to keep 802.11 frames, i.e. readable by Wireshark. www.wireshark.org

namely:

- The source MAC address;
- The MAC sequence number;
- Frametype;
- Subtype;
- TID bits - three bits;
- Timestamp in milliseconds;
- Timestamp in seconds

When 55 000 frames are written to the file with these parameters, `sidsv2.pl` calls another script called `check.pl`, asking for the script to check to see if the parameters are good according to the algorithm from section 3.3. This means that `sidsv2.pl` can keep on sniffing frames, and not keep track of all the different checks at all times, but call his "child script" `check.pl` if it finds it suitable. In simple terms, there is always one instance of `sidsv2.pl` running, but there can be many `check.pl` running at the same time.

`sidsv2.pl` third job is to keep track of specific management frames, and perform analysis on these in a state machine fashion, as I will show in section 3.5.1. This is a new feature not discussed by [3], but as we will see in the results chapter give good results as an extra freeloader detection technique in Intrusion Detection Systems. Combining different techniques will improve the quality of the detection mechanism, and add reliability to the system in regards to both false negatives and false positives. More on this in chapter5-

3.4.1 check.pl

As explained above, `sidsv2.pl` will call instances of `check.pl` when enough frames are picked up from the same source MAC address and a check is required. `check.pl` is implemented to suit the proposed algorithm described in 3.3, and is thus divided into subroutines for each step. Step a and b are performed in the same subroutine named `check_linprog`. We can see by the code below that it will increase an alarm counter with one -1- if the offset between the sequence number from the testing frame is more than -5- times greater than the previous frame. It also verifies that the MAC sequence counter is not looping around 4095 to avoid false alarm triggers as explained in 2.1.2.

```
if (($SEQ - $OLD > 5) && ($OLD != 4095) && ($SEQ != 4095)) {  
    $counter = $counter+1;  
}
```

It will proceed with Step 2c, called `check_frtypes`, if the counter has more than 5 hits³. If not, it will delete the file "`<sourcemac>.check`", and drop further testing. The instance of `check.pl` will then terminate.

The `check_frtypes` subroutine simply checks to see if the subtype bit of QoS are set, as well as verify that the frame is not a control frame. Control frames do not carry MAC sequence numbers, and should therefore not be considered here. If the frames are not of type QoS, then we can conclude that an attack has been carried out. If not, we must proceed to check if we have equal QoS priority on all frames, hence call the subroutine `check_priority` (step 2D).

³These parameters are due to change, and should be carefully considered and tested if SIDS is to be deployed


```
if (($FRT == ("11" || "01") && QOSBIT == "0") {  
    my $logmsg = "Step 2c - Attack on Source MAC: $MAC";  
    write_log($logmsg);  
} else {  
    equal_priority($file);  
}
```

The subroutine function `equal_priority` simply checks if all TID bits are the same for every package. If this is the case, QoS are enabled on the network, but not used by the STA as explained in section 3.3.5. If the TID bits are different, `check.pl` knows that the sequence of frames from this MAC is either mixed TID bits, or we have mixed TID bits -and- regular Data Frames. Now there is a possibility that a freeloader attack has occurred, but this should be verified. In the next section I will propose enhancements and new methods to verify a potential attack if QoS are enabled and used by the client. (Or the attacker)

3.5 Enhancements in SIDS v2.0

To cope with the problems using unreliable physical parameters, I have implemented and tested two new techniques in SIDS v2.0. The first, 802.11 State Control, is not directly related to the proposed algorithm from above, but should be a good extra detection technique in Intrusion Detection Systems. The second, Active Probing, is implemented in Step 2e, and has replaced the recipe from [3] where they suggest calculating Euclidian distance to verify the attack.

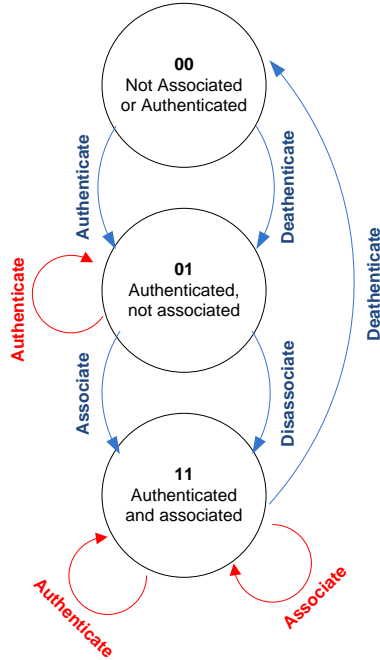


Figure 3.2: Valid (blue) and Invalid (red) Management Frames in 802.11 states

3.5.1 Passive Detection - 802.11 State Control

As described in section 2.1.3, 802.11 operates with different states when an AP and STA is to be connected. 802.11-2007 [1] defines three different states. In this section I will show how I implemented a state approach in SIDS to enhance detection of freeloader attacks. This method use the ideas provided by Jason Smith [5] as described in chapter I section 1.5.

In section 2.1.3 I explained how the 802.11 protocol defines different

states when a STA wants to associate and authenticate with a corresponding AP. If we define the management frames in figure 2.4 as valid frames, the state transitions from the figure is what I will from now define as normal behavior. This means that the transitions in the figure are expected Management Frames messages one should see when a legit STA is communicating with an AP. Nevertheless, in a freeloader attack, the behavior of the state machine is radically changed.

In figure 3.2, the transitions marked with red are frames originating from a MAC address which should not be sent due to the way the 802.11 states operates. For instance, if a legit STA is in state 3; authenticated and associated, one should not see an authentication or association frame originating from the STA. It is important to understand that if for some reason the STA gets disassociated, it will send a re-association message in state 2, and not a standard association message. Also note that if a STA gets de-authenticated, it must also send an authentication message before it can send an associate message.

As a result of this, one should not see an authenticate message in state 2, nor should one not see authentication or association messages when in state 3. These transitions are marked with red in figure 3.2. Table 3.1 will give an overview over which frames are allowed and which is not as described above.

As described in section 3.4, `sidsv2.pl` analyze management frames, and keep track of the different states for each STA communicating with the AP. There exists a hash map, which keeps the states of each MAC address that has been or is communicating with the AP. The states can be set to 00, 01 or 11, accordingly to figure 2.4. For each packet processed, the script will

State	Valid frames	Invalid frames
State 1	Authentication Deauthentication	Association Request Reassociation Request Disassociation
State 2	Association Request Reassociation request Deauthenticate	Disassociation Authentication Request
State 3	Deauthenticate Disassociation	Association Request Authentication Request

Table 3.1: Valid and invalid management frames in 802.11 states

filter on Management Frames, and if one of the 7 following subtypes is seen, it will trigger an analysis of the event:

- Association Request;
- Association Response;
- Reassociation Request;
- Reassociation Response;
- Disassociation;
- Authentication;
- Deauthentication

If one of these messages are seen, SIDS will verify if the message is of a valid frame according to that state as detailed in table 3.1. It will first read that it is in fact a Management Frame, check the subtype bits to see if it is any of the above, and finally verify if the specific frame is expected according

to the state the STA is in. If the frame is a valid one, it will update the STAs state to the new correct one, i.e. receiving a Deauthentication frame in state 11, would correspond to setting the new state to 00 (see figure 3.2). If the transition is not valid, it will write to a log file `"/var/log/sids"` with the time of the event, and what happened. An example of the code analysis for Authentication frames are given below:

```
# Authentication check
if ($subtype == "1011") {
    if ($mac_state{ $src_mac } == ("11" or "01")) {
        my $logmsg = "$src_id already authenticated";
        write_log($logmsg);
    } elsif ($mac_state{ $src_mac } == "00") {
        $mac_state{ $src_mac } = "01";
    }
}
```

Here you can see that upon receiving a frame type 00 and subtype 1011, which according to table A.1 is an Authenticate Management Frame, SIDS will first verify if the state of this specific MAC address is set to be in either "11" or "01" (State 1 and 2 accordingly). If this is not the case it will call `write_log` telling the log file that the MAC address was already authenticated when receiving an Authenticate message. Otherwise it will set the correct state for the STA.

3.5.2 Active Detection - Active Probing

This enhancement relate directly to how the algorithm from section 3.3 is build up. As RSS measurements are not always reliable, I have chosen a new technique to verify if an QoS optimized attack really is ongoing. This

technique cannot be performed offline, but is real-time dependent. Until now, we have only seen examples of passive detection techniques. Passive Detection techniques does not actively participate in the network. In all of our cases above, SIDS has been standing explicitly as a passive node, monitoring and analyzing 802.11 frames. The advantage of this is that one does not need to interfere with the current infrastructure of the network, but we can place monitoring nodes next to the serving network. As everything in our example are using wireless technologies, one could conclude that we do not need more configuring then a simple wireless NIC and a power outlet to set up SIDS on a node somewhere.

Nevertheless, passive monitoring has its limits. If we want to make sure that an attack has occurred, it can be of good fashion to actively participate in the network by trigger and challenge different nodes in order to analyze the response of the event. This is what is called active probing. The IDS can be a part of the backbone network, and has access to trigger events in order to challenge STAs associated with the AP. The idea behind the probing is to verify that the challenge response from the STAs is what should be expected if no attack is ongoing. If in fact a freeloader attack is being performed, active probing can help the IDS reveal and verify the attack, where both the legit STA and the attacking STA may answer to challenges.

PING - Using ICMP Echoes And Replies In figure 3.3 I have showed the principle behind a simple active probing detection event. Here, I use ICMP echoes and ICMP replies to check whether two responses are seen when issuing a PING request from inside the network. If we recap back to section 3.3, we can implement active probing techniques in Step 2E, where a potential attack should be verified. For instance, if SIDS has one wireless NIC analyzing frames according to the description in section 3.4,

3.6. ATTACKING AND GENERATING OFFLINE CAPTURE FILES

and one NIC facing towards the backbone of the monitoring network, it can trigger PING packets to the specified IP address if it wants to verify if a QoS-enhanced attack is occurring. It will send a PING packet ⁴ from the wired interface, into the backbone network with the STA's IP address as destination. This packet will, due to the logic of network topology, find its way through the Wireless LAN Controller, and to the AP serving the STA's, and finally be sent as an ICMP echo frame to the wireless STA. STAs NICs usually reply to ICMP echoes in form of ICMP REPLY packets. If an attacker has spoofed the IP address of the client, it is likely that both the legit STA and the attacking STA will send ICMP reply packet back to the AP, and to the IDS cabled interface again. The IDS can at this point both analyze the receiving packets from the cabled network, as well as analyze ICMP frames from the same MAC address. If it sees duplicate ICMP reply packet, it can log the event.

3.6 Attacking And Generating Offline Capture Files

As I wanted to test different techniques and get good comparable results, I decided that it was a good manner to create offline capture files capturing different scenarios. These recordings were of course done on the hardware meant to be running the IDS. First I needed to set the wireless NIC in monitor mode, in order for the NIC to act as a passive monitoring node. This is done by issuing:

```
sudo iwconfig wlan0 mode monitor channel 6
```

⁴Read about PING at <http://en.wikipedia.org/wiki/Ping>

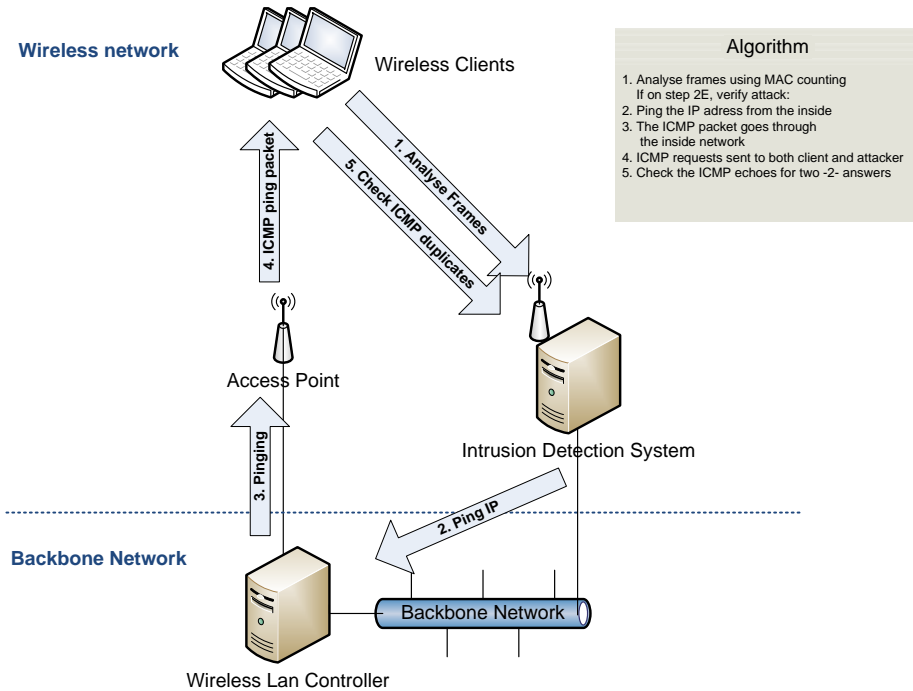


Figure 3.3: Active Probing using ICMP packets to verify freeloader attacks

This will set the wireless interface `wlan0` into monitor mode, and it will thus not participate in any network, but act as a passive monitoring device. It will listen to all WLAN traffic on channel 6. `Airodump-ng`⁵ was used to capture frames, and was issued in the following:

```
airodump-ng -w <name>.cap
--channel 6 --bssid 00:23:5D:0D:FB:F0 wlan0
```

⁵Airodump-ng is used for packet capturing of raw 802.11 frames

3.6. ATTACKING AND GENERATING OFFLINE CAPTURE FILES

The Basic Service Set Identifier (BSSID) is the MAC address of the AP to be monitored, and is a good filter when acting as a passive node. The AP is operating on channel 6, and the interface which should be used for sniffing is wlan0. All files below were created according to this recipe.

3.6.1 NoAttack.cap

This file was created for verification purposes in order to test different techniques when no attack or abnormalities were occurring in the network. This capture is a 20 minute record, and has captured a scenario where different STAs log on and off the wireless network, as well as generating traffic and browsing the internet, as most open 802.11 wireless operates. The following STA's were authenticated, associated and used sporadically during the record :

- Apple MacBook Pro;
- Apple iPhone;
- HTC Magic;
- Windows 7 - Hewlet Packard;
- Ubuntu 9.10 - IBM;

The purpose behind this record is to have a test file to check for false positives when testing new techniques. None of the detection techniques described in section 3.4 and 3.5 should trigger alarms on this capture file, as I made sure that no attack or abnormalities were issued during the record.

3.6.2 FreeloaderAttack.cap

In the same matter as in section 3.6.1, this capture file was generated throughout a 20 minute timeframe. The scenario, however, was somewhat different. All the devices from the list above were connected from the start, except the IBM laptop running Ubuntu 9.10. After 10 minutes, a freeloader attack was generated to freeload on the Apple Macbook Pro's legit connection to the network.

The attack was carried out in the same fashion as explained in Ørjan Bækkelund's master thesis from 2008 [13]. The goal behind this task was not to give another proof of concept for a freeloader attack, but to create a capture file with a MAC freeloader attack to test on. The attack is detailed in the master from Ørjan Bækkelund [13], chapter 3, section 3.3.

3.6.3 FreeloaderPING.cap

In order to test and verify the passive detection technique concept from 3.5.2, I also generated a third capture file. This is also a 20 minute record and is quite similar to the one above. The difference is, that during the attack, I issued a

```
ping 10.0.0.22 -c 3
```

from the IDS. Three PING packets, or ICMP request packets, was sent out on the cable interface eth0, and I got successful pinging as an output in the console from the IDS whenever the ICMP packet went through. More on this in the subsequent chapter, section 4.3.

In the next chapter we will see the results when testing SIDS on these capture files, and further discuss my findings in chapter 5.

4

Results

This chapter will show the results when testing new detection techniques, as well as SIDS v2.0 in various scenarios and settings. The chapter will first verify the problem with SIDS v1.0, and how the IDS was not optimized to deal with the 802.11e problems from section 1.5.2. Further I will present the results when SIDS v2.0 was optimized with the new detection techniques as proposed in section 3.5. Finally, in section 4.4 I will briefly elaborate on other results in which I found was interesting according to the outline of this thesis.

4.1 SIDS

In chapter 3, section 3.3 and 3.4 I showed how SIDS was implemented accordingly to the structure from the proposed algorithm in [3]. The new features in SIDS was tested in the same matter as in our project of 2009 - using the Wireless Trondheim testing lab, based on a typical CUWN provided in figure 3.1¹. In my testing setup, I had one LAP connected to the network, and used this for analysis of STAs associated to the access point. As the lab already was set up to replicate the wireless network of Wireless Trondheim from before, I had more time experimenting and testing different techniques. The IDS and the lab was set up directly according to the project of 2009, given in figure 3.1. SIDS was tested on the IDS server "Rybak" from the figure 3.1, and it was also upgraded with a wireless NIC to act as the Wireless monitoring node listening on frames with destination address for the AP as explained in 3.4. As the physical location of the IDS/Rybak was approximately 50cm away from LAP1, I did not have problems with losing frames in the IDS that was picked up by the LAP. Even with 55 000 frames processed by the IDS, it was only 3 MAC Sequence Number offsets as a result of losing frames. This, of course, may have affected my results in a positive matter in contrary to a real-world testing of the techniques, but as I will explain here and in chapter 5, the new techniques should be suitable in an urban environments as well.

4.1.1 MAC Sequence Counting Analysis from SIDS v1.0

First of all, I wanted to test the rate of false positives when no attack was carried out. I used SIDS v1.0 from our project assignment 2009 to

¹See the project in [2] for further details on this

test standard MAC Sequence Counting Analysis without optimizing it to deal with the new features of 802.11e/WMM. First, I disabled WMM on the Wireless LAN Controller, and verified that SIDS v1.0 did not trigger alarms when executing the script live on several devices in my lab. No alarms were set off. After this I carried out a standard MAC freeloader attack on the MacBook pro, and the IDS wrote to the log file almost at an instant that a freeloader attack had occurred. At this stage nothing new in regards to our project assignment from 2009.

WMM enabled As WMM was enabled when issuing the record files from 3.6, I did not have to test the IDS in a live system. I executed SIDS v1.0 on NoAttack.cap, and the log file was overwhelmed with alarms that freeloader attacks had occurred. I also executed it on FreeloaderAttack.cap, but there was no way for me to see from the log file which alarms that were correct, and which was a false. This verifies that SIDS v1.0 becomes useless when enabling Quality-of-Service on the link-layer in the network.

4.2 SIDS v2.0 - Passive Detection

In this section I will show the results of implementing a passive detection mechanism, which I called 802.11 state control. In my Theory chapter 2 section 2.1.3 I explained how 802.11 STA and AP's operate with 3 different states for each STA communicating with an AP. After implementation of code as detailed in 3.5.1, I tested this technique on the two of the cap files from section 3.6.

No Attack To verify the rate of false positives, I executed SIDS v2.0 on NoAttack.cap from section 3.6.1. During this 20 minute record, SIDS v2.0

did not state that a single attack was executed, in contrary to v1.0. As SIDS v2.0 distinguish between TID classes, the log file did not get overwhelmed with false positives.

Freeloader Attack Second, I executed SIDS v2.0 on FreeloaderAttack.cap from section 3.6.2. As you can see from figure 4.1, the IDS successfully triggered on the freeloader attack. As explained in section 2.1.3, it is standard for a STA to first issue an authenticate message, and then an associate message when connecting to an AP. From the figure we can see that SIDS triggered two alarms during the freeloader attack. As the legit STA was already authenticated and associated, SIDS triggered an alarm when the attacking STA issued an authenticate request, as well as on the association request straight afterwards. As the STA was already proper associated with the AP before the freeloader attack was carried out, I had to manually set the state machine in SIDS to set the proper STA as already in state two before the record began. Otherwise I would have experienced false negatives, as the authentication and association from the attacking STA would simply just update the state machine in SIDS from state 0- to 2-, implying that a legit STA was associating - not triggering attacks.

This means that we can see from figure 4.1, that SIDS is able to detect when new STA's connect to an AP, masquerading the legit STA's MAC and IP address. As we will see from my discussion section 5.1.1 on page 72, it is very difficult for an attacker to bypass this detection and it would require an advanced set of skills in order for the wireless NIC to not send authenticate and association messages when connecting to an AP.

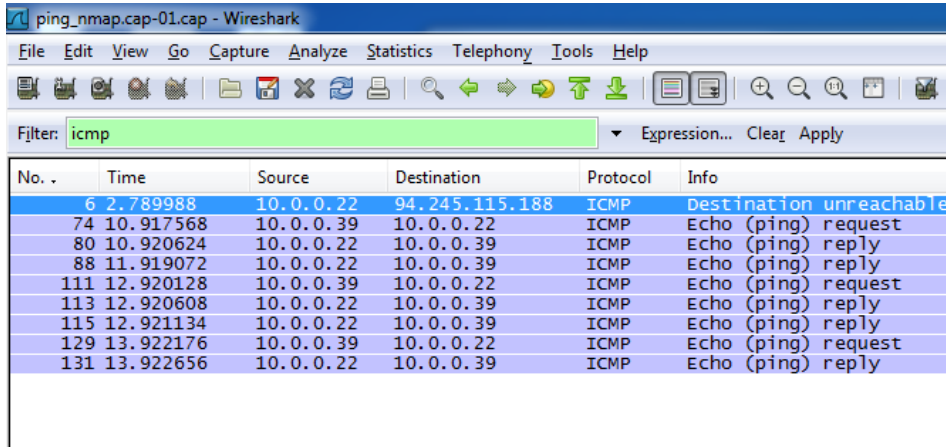
```
root@rybak:/home/rybak/Master# ./sidsv2.pl
Capturing traffic ...
Attack! 002608e8f8a3 already authenticated
Reassociation Request
Attack! 002608e8f8a3 already authenticated
Attack! 002608e8f8a3 already associated
```

Figure 4.1: Result of running Passive Detection on FreeloaderAttack.cap - 802.11 state control

4.3 SIDS v2.0 - Active Detection

As an enhancement of detection of freeloader attacks, I suggested using Active Probing as a detection technique as described in section 3.5.2. As a proof of concept, I implemented an example of active probing, namely ICMP echo/ICMP reply. The basics behind this detection technique is to probe the STA with an ICMP Request packet (Or PING as more commonly referred to), and check if the STA responds two identical ICMP reply packets. This is a fairly easy technique, and as we will see below, it can easily be avoided by an attacker. However, as I will discuss in section 5.1.2, other types of probing techniques can be used, and experiments on TCP SYN probing and other probing techniques should be looked into. Nevertheless, as a proof of concept, this technique had good results when experimenting with different STA's, and as one can see from figure 4.2, the IDS received two duplicate ICMP replies when issuing a ping request during a freeloader attack.

This is also included in SIDS, where `sidsv2.pl` constantly is checking whether there is written an file named `<ip-address>.check` on the disk. If this is the case, `sidsv2.pl` knows that `check.pl` has issued a probe request, and will check if duplicate ICMP frames are received from that IP address. As this has to be performed live, i.e. not on capture files, I just verified in



The image shows a Wireshark capture window titled 'ping_nmap.cap-01.cap - Wireshark'. The filter is set to 'icmp'. The packet list pane shows the following data:

No. .	Time	Source	Destination	Protocol	Info
6	2.789988	10.0.0.22	94.245.115.188	ICMP	Destination unreachable
74	10.917568	10.0.0.39	10.0.0.22	ICMP	Echo (ping) request
80	10.920624	10.0.0.22	10.0.0.39	ICMP	Echo (ping) reply
88	11.919072	10.0.0.22	10.0.0.39	ICMP	Echo (ping) reply
111	12.920128	10.0.0.39	10.0.0.22	ICMP	Echo (ping) request
113	12.920608	10.0.0.22	10.0.0.39	ICMP	Echo (ping) reply
115	12.921134	10.0.0.22	10.0.0.39	ICMP	Echo (ping) reply
129	13.922176	10.0.0.39	10.0.0.22	ICMP	Echo (ping) request
131	13.922656	10.0.0.22	10.0.0.39	ICMP	Echo (ping) reply

Figure 4.2: Duplicate ICMP replies in FreeloaderAttackPING.cap

Wireshark that the capture file indeed captured two identical ICMP echo replies as shown in figure 4.2. This figure shows a part of the FreeloaderPING.cap from section 3.6.3 filtered on the ICMP protocol.

After this, I tested several devices, and as we will see in the following subsection, there were inconsistent results due to firewall issues.

4.3.1 Firewalls

The FreeloaderPING.cap is a record when issued a PING request on a Macintosh running OS X Snow Leopard² as the legit STA, and Ubuntu 9.10 as the attacking STA. In this scenario, there were no firewalls enabled on either of the devices. As a result of this, I obtained successful detection, as both computers answered the ICMP request.

Nevertheless, as I wanted to test the technique using different Operating

²Apple's newest Operating System

System (OS)s, I also tried a freeloader attack on a STA running Windows 7. As windows 7 has a default firewall rule not to answer on ping (see chapter5, section 5.1.2 for details), only the attacking STA running Ubuntu 9.10 in this case answered the request, and I did not have a successful detection of the attack.

The result of this, as you might understand, that for an attacker to avoid this detection technique, it is as simple as setting up a simple firewall not responding to ICMP, or PING, requests. Nevertheless, as we will see in section 5.1.2, this is just a proof-of-concept, and other techniques might be used as described in section 8.1.

I also verified a few other devices to find out which one has firewall enabled by default, and from the list below, only Windows 7 dropped the PING request.

- Apple MacBook Pro;
- Apple iPhone;
- HTC Magic;
- Windows 7 - Hewlet Packard;
- Ubuntu 9.10 - IBM;

4.4 WMM enabled

When experimenting with SIDS, I also came across several results which I found interesting. In this section I will look at a spesific result that could be of interest in order to detect if a QoS attack is performed. Figure 4.3 is a screenshot of Wireshark, where the packet is marked with the application SSH. This was captured when WMM was enabled on the network,

CHAPTER 4. RESULTS

and as you can clearly see from the figure, SSH sessions are marked as "Background" classification as explained in section 2.5.1. This matter will be further analyzed in chapter 6 on page 77.

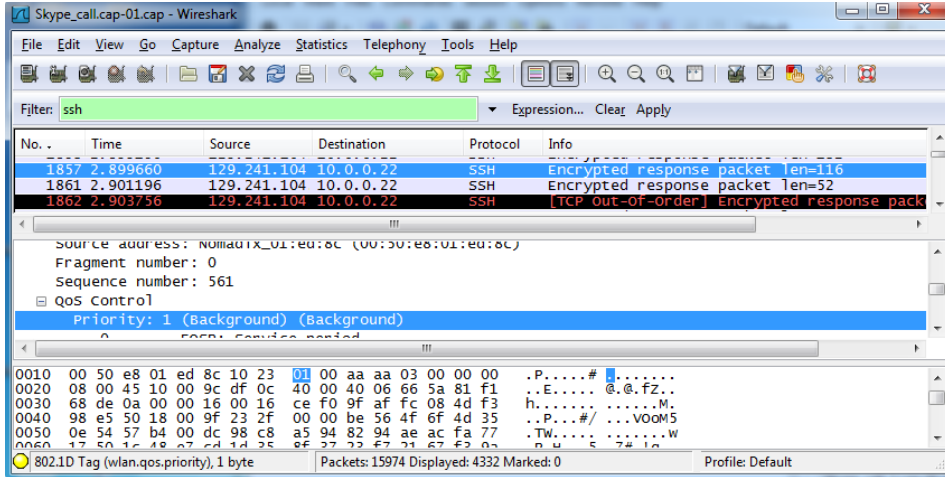


Figure 4.3: SSH packets flagged with background WMM classification

Part III

Discussion and Analysis

5

SIDS v2.0

In this chapter I will look at how my methods of choice provided in chapter 3 suits the results gained from chapter 4. I will start by looking at SIDS as an Intrusion Detection System, and analyze the results provided in the previous chapter. The chapter will also in section 5.2 discuss where SIDS stands to act as a complete detection system.

5.1 Implemented Detection Techniques in SIDS v2.0

In chapter 3 section 3.5 I implemented two new type of techniques in SIDS. The first, Passive Detection, use 802.11 protocol specifics in order to deter-

mine that a freeloader attack is ongoing. It is of importance to understand, as I described in section 3.1, that this detection technique only detects attacks of type Freeloader as described in section 3.2. If an attacker decides to wait for the STA to drop off from the network before the attack is carried out, the attack is known as a session hijacking attack, and will become much more advanced to discover. Nevertheless, in order to detect freeloader attack, this detection technique can be quite tricky to avoid from an attackers perspective as we will see in a moment.

The second technique is an active one. This basically means that it will actively probe associated STA's in order to analyze the outcome of the request to verify if an attack is occurring. This is described in section 3.5.2, and results in the IDS having to be a part of the backbone network of some sort.¹ This specific implementation is barely meant as a proof-of-concept, and as we will see in 5.1.2, that from an attackers perspective, this specific detection technique using ICMP can easily be avoided by simply enabling a firewall rule in the OS. As I will discuss soon, there should be other techniques that is suitable for this type of scenario, and as stated before, the more detection techniques the IDS is capable of performing, the higher the possibility is for discovering attacks.

5.1.1 Exploiting 802.11 states

The theory behind 802.11 states are given in 2.1.3. As the 802.11 states are defined and given in the 802.11-2007 standard provided in [1], manufactures and vendors strongly embed this into the Wireless NIC's or the software drivers of their cards. It is a part of the way that 802.11 operate, with no exceptions. One might think that the attacker simply could re-program the

¹This may not be true if the active probing is done through the wireless NIC from the IDS

drivers not to participate in 802.11 states by excluding management frames meant to associate and authenticate to the network. The fact of the matter is, that these drivers are often quite advanced in terms of programming, and due to the fact that the 802.11 protocol works this way, there should be an extensive amount of work on re-programming the drivers in order to optimize the freeloader attack to bypass the detection system. This is in fact possible, but the amount of work programming each driver could potentially be enormous, and perhaps not worth it. Nevertheless, as we have seen before in the security community, if one person does all the hard work, it is simple to distribute the drivers through the internet. I have not looked into the detail of work needed to re-program the drivers, but it is reasonable to believe that it should be a lot of logical challenges as well as strong programming skills in order to bypass the way 802.11 states operates, and embed this into the drivers.

The Results As we have seen in the results from section 4.2, SIDS was perfectly able to detect if another STA were using the legit STA's credentials (hence, IP-address and MAC address), and was authenticating and associating with the same AP. It is important for the IDS to keep track of all 7 Management Frames that can affect the STAs state. These are provided in the list from section 3.5.1. By now SIDS are not optimized to deal with re-association and re-authentication frames, but this should be an easy implementation if the IDS is to be deployed. Another aspect which should be looked at is what happens if the IDS is not able to detect a management frame of importance to the 802.11 state for the STA. If the frame is not picked up by the IDS, the state control diagram in SIDS will become different from that of the AP and STA. This will probably result in the IDS triggering false alarms, or get confused and will not trigger alarms.

Nevertheless, during my experiments I did not lose a single management frame, and one should make sure before deploying the system that the IDS is as close to the monitored AP as possible, as well as verify that it will not lose these frames in a highly trafficked environment.

5.1.2 Active Probing using ICMP

In section 3.5.2 I showed how I implemented an active detection technique, issuing a PING request, and see if we get two answers originating from the same source MAC address. This is implemented in SIDS v2.0 at step 2e as described in section 3.3.6, instead of performing an Euclidian distance algorithm as proposed by [3]. Instead of relying on physical parameters, I have implemented a technique which is based on how 802.11 networks operate. The implementation is just a proof-of-concept, and due to the firewall problems discussed below, one should look into how to use other probing techniques as well. Nevertheless, the results from section 4.3 shows that it in fact two replies were recorded on the IDS, meaning that the detection technique works under the right circumstances. I have also implemented this in SIDS, by issuing a ping packet to the desired IP address if a QoS optimized attack should be verified. As we will see in the next chapter, this technique alone is not necessary good enough. If the attacker has gained knowledge of how the IDS works, and drops every probe request from the IDS, the system will not trigger alarms. As a result of this, my final proposed and optimized algorithm in chapter 7 suggest that this technique should be embedded, but techniques from chapter 6 should also be considered and implemented.

TCP Example This is another example of an active probing technique. If TCP packets are not dropped by the attacker, the IDS could for instance

use nmap² to issue a specific TCP packet named TCP+SYN (TCP synchronize), and see if two identical SYN+ACK (TCP acknowledgement) packets are picked up. The idea behind this detection use exactly the same basics as the one implemented in SIDS, but instead use specifics from the TCP protocol, not ICMP, in order to detect attacks. As you might see, there are many forms for active probing, and one could potentially look at each of the layers in the OSI model from figure 2.1, in order to find suitable protocols and applications that issues active probing. The obvious down sight to this approach, as stated above, is that potentially every probing request could be dropped by an attacker if he has knowledge on how the IDS operates. To cope with this, new techniques will be proposed in chapter 7.

5.2 SIDS - Where Are We At?

As SIDS v2.0 developed throughout the writing of this thesis, it has become quite advanced. SIDS v1.0, as developed during the project assignment, was a fairly simple IDS. The obvious gain from the project report was the discovery of the problems with 802.11e enabled wireless networks as handled in this thesis. The re-writing of SIDS from section 3.3, made the IDS more programmable understandable, as well as optimized to cope with QoS enabled wireless networks. With the new detection techniques from above implemented, the IDS are able to keep false positives to a minimum, as well as trigger alarms when the newly QoS optimized attack is performed. Nevertheless, as described in the previous section, the implemented detection techniques should be easy for an advanced attacker to counter. As a result of this, I need to look into other techniques. In the next chapter

²nmap is a security scanner, used to discover hosts and services on a computer network, thus creating a "map" of the network

I will analyze and elaborate on further enhancements outside techniques implemented in SIDS.

5.2.1 SIDS as an IDS

SIDS is not far from ready to be deployed as a complete IDS system. As described in section 3.4, there are a few parameters that should be tested and analyzed in regards to sequence counting. The most important is how far sequence number offsets can be, and how SIDS should deal with lost frames - especially management frames in regards to passive detection. I will also state that other detection techniques should be implemented in SIDS, not only relying on one single measuring technique. Even though RSS values can be unreliable in urban and mobile environments, one could implement this, together with RTS/CTS RTT as described in section 2.4.4, as an extra detection technique. In the latter case it is important to understand the level of false positives this potentially could provide, and one should carefully analyze how the detection techniques should relate to each other. From [2], we described that triggering of an alarm should base its roots on strong statistical analysis of events triggered from several different detection techniques.

6

Analytic Detection Techniques

In chapter 5 I have implemented and tested two types of detection techniques in SIDSv2.0. During the development of the IDS, I also came across interesting thoughts and ideas to new methods in order to detect attacks in 80.211e enabled open wireless networks. In this chapter I will present my theories, and even though these are not implemented in my IDS at this stage, their implementation and testing should be a less challenging task for future work as described in section 8.

6.1 The 802.11e and WMM relationship

As detailed in my theory chapter 2, section 2.5.1 and 2.5, WMM use only 4 of the 8 available TID bits in 802.11e. Parameters in the beacon frames of AP tells the STAs which QoS scheme that are supported. In my lab setup, WMM is used. This means that the STA can use the 4 classes, namely Best-effort, Background, Video and Voice. Table 6.1 shows how the WMM classes are reflected as 802.11e TID bits.

TID bits	WMM Class
000	Best-Effort
001	Background
010	Video
011	Voice
100	<i>Not used</i>
101	<i>Not used</i>
110	<i>Not used</i>
111	<i>Not used</i>

Table 6.1: Relationship between TID bits and WMM Classes

If an attacker inject his frames to that of a different QoS class that isn't in use in the specific QoS scheme, one could trigger alarms if TID bits are used by a STA which isn't supported by the network. It is unreasonable to believe that a legitimate STA will use TID class 1xx in a WMM environment, as the parameter exchange during the association phase from figure 2.4 tells the STA that only WMM is in used, and there is no point in using other TID classes than the ones defined above.

As a result of this, one should be able to detect this type of attack, simply by verifying that the most significant TID bits in each QoS Data and management frame from a STA is not -1-.

In general, one could look at which QoS parameters are exchanged during the association phase when a STA associates with an AP, and see if the behavior of TID bits in the subsequent frames are according to what is expected when using the specific QoS management scheme.

6.2 Application and TID analysis

The techniques proposed in section 6.1 above are indeed interesting. Nevertheless, as described in section 3.2, a QoS optimized attack does not have to use the -4- last TID classes in order to be successful. A simple scenario will be when a STA is in fact not using WMM Video or Voice Access Categories, or simply is not using WMM at all. In this case, the attacker can flag every frame with TID bits 010 or 011, masquerading as the STA using Voice or Video, again fooling the detection system. In this section I will propose another enhancement to cope with this type attack, by moving away from the MAC layer of the OSI model, and on to higher layers.

6.2.1 Application and Protocol Based Detection

As the network considered in this thesis are of type open 802.11 wireless network, there are no encryption of frames, and the IDS can read information on the application level in the OSI model from figure 2.1. As a result of this, I will here propose another technique which will see if the application or protocol fragmented into a specific frame is logically according to which

TID bits which are set. As described above an attacker could fool TID analysis by using TID bits that are not in use by the STA, but is a part of the QoS scheme. If this is the case, one could look at the TID bits and see if they are reasonable in regards to the application or protocol used. An example is provided in the next chapter, and in more general terms we could describe this technique as

Look at the relationship between the TID bits and which application or protocol used on the frame. If this relationship is not reasonable, it should be an indication that a QoS optimized attack has taken place.

7

New Proposed Algorithm

In section 3.3 I looked at the proposed algorithm for QoS considered detection, and throughout this thesis I have elaborated on alternative detection techniques instead of relying on uncertain physical parameters. I have concentrated primarily on looking at the second layer in the OSI model. In this chapter I will propose a new algorithm based on the results and discussion from chapters II to 6.

7.1 The Algorithm

Even though not every part of the algorithm below is implemented in SIDS v2.0, the discussions from chapter 6 should be fairly easy to implement in Intrusion Detection Systems, such as SIDS v2.0. By analyzing which Quality-of-Service scheme is used, as well as look at the other layers of the OSI model, the algorithm should be a good guide in how to cope with the QoS problems described throughout this thesis. Nevertheless, it is important to understand that this algorithm requires the network to be an open 802.11 wireless network. This is due to the fact that it relies on reading frame information from higher levels in the OSI model, which is usually encrypted when using features from 802.11i¹. The algorithm is provided below, and in section 7.2 I will give my explanations in regards to how the algorithm operates.

```
Algorithm: Find-Identity-Spoofs 2
Input    : S: Sequence of wireless packets
Output   : Attack/No_Attack
MACs = list of MAC sequence numbers in S;
if MACs in linear progression then
    return No_Attack;
else if MAC variation in valid range then
    return No_Attack;

/* MAC sequences not in linear
progression; check frame types */

FTypes = frame types extracted from S;
if FTypes 2 {Management, Regular Data} then
    return Attack;

/* Frame type must be QoS-Data; examine
```

¹802.11i is a part of the 802.11-2007 standard, and provides security features for 802.11 wireless networks


```
priorities */

QoS-Priorities = QoS priorities extracted from S;
if QoS-Priorities are all the same then
    return Attack;

/* QoS priorities are either mixed, or
mixed QoS-data and regular data */

Active probe the clients listed in MACs;
if Two identical responses received then
    return Attack;

/* The attacker could be using firewall or
countermeasures for active probing */

Verify QoS Scheme, analyse TID bits extracted from S;
if TID bits outside QoS scheme then
    return Attack;

/* TID bits are inside the specific QoS scheme used */

Analyse application or protocol used in packets from S;
if Application type differs from expected TID class then
    return Attack,
return No_Attack;
```

Listing 7.1: Optimized algorithm for QoS considered identity spoof detection

7.2 New Features

When writing about new features in an algorithm, it is also important to describe which ones that are removed. As described in this paper, my initial goal was to move away from the physical layer in order to create a more reliable and analytic detection algorithm. As RSS values can be unreliable

in urban mobile environments, the algorithm does not take RSS values into consideration, but relies on analysis of the 802.11 standard, as well as how 802.11e is implemented. As a result of this the following has been removed:

```
Perform differential localization for packets in S;  
if Euclidean distance between successive packets exceeds  
threshold then  
    return Attack;
```

Instead, the algorithm suggest using active probing techniques as described in section 3.5.2 and 5.1.2. Due to the risk of an attacker potentially fooling the Active Probing technique (refr: i.e. Firewalls issues from section 5.1.2), the algorithm continues verification by analyzing the QoS scheme negotiated between the legitimate STA and the wireless environment. It will see if the TID classes are within the same range as the QoS scheme used. If this is the case, it will further look at other layers, and see if the TID classes that are set, are logically in regards to which application or protocol used. For instance, if frames are marked with TID class 010 (video from WMM) and WMM is used, and the packet shows that it is the TCP protocol which is used, it is unreasonable that this is in fact a legit STA - as video streaming uses UDP. I will not in this thesis propose an overview over which applications or protocols are expected to use which TID classes, and is subject for future work as described in chapter 8.

802.11 State Control In section3.5.1 I described my theory and implementation of a passive detection technique which I called *802.11 State Control*. This is not part of the algorithm above, but should definitely be considered in regards to deploying a complete IDS system. This method will

work fine on its own, and is not directly dependent on other techniques like the ones described in the algorithm. Nevertheless, as IDS should provide the administrator of a network with a statistical analysis of what triggered the alarm, one should relate the different techniques to each other, and verify the level of credibility to the triggered alarm by combining different techniques. However, this is not considered here, and is as noted earlier, subject for future work as described in section .

8

Future Work

Even though the work done in this thesis is quite comprehensive, there are several subjects which should be suitable for future work. There are probably many things that could potentially be taken into consideration, but the most important ones are provided in section 8.1 and 8.2. Unfortunately I did not have time to test and verify all of my new techniques, and I find the analytic detection techniques from chapter 6 really interesting. It is reasonable to believe that the work needed to implement and test these techniques in SIDS should be less challenging, and I think that definitely this is a subject for future work and should be further looked into.

8.1 Active Probing - Beyond Proof of Concept

As I have discussed throughout section 5.1.2, the Active Probing technique implemented in SIDS v2.0 is not good enough. The technique is too easy for an attacker to avoid. Nevertheless, the idea behind active probing should be analyzed further, and I believe that there exist other probing techniques that should be suitable for this use. As stated earlier, one could look at each of the layers in the OSI model, looking at protocols and applications that can be used as probing. This is actually an advantage when working with open 802.11 wireless networks, where one has access to everything from all layers.

8.2 Completing The IDS

The new proposed algorithm from chapter 7 is indeed interesting. Nevertheless, there are several issues that should be elaborated on. First of all, the algorithm should be implemented and tested in an IDS, where I of course highly recommend using SIDS v2.0. One of the challenges here is to analyze which applications and protocols use a QoS Scheme, and provide SIDS with the proper information so it can distinguish between these and determine attacks if the application or protocol used is not in logical relationship with the TID bits set. As a result of this implementation, SIDS should be close to testing and deployment, but the centralizations issues discussed in our project assignment from 2009 [2] should also be taken into consideration.

9

Conclusion

In this thesis I have seen that with the introduction of link-layer Quality-of-Service in wireless networks, there are problems with using MAC Sequence Counting Analysis as a detection technique in wireless Intrusion Detection Systems. As many IDS heavily rely on this detection technique, this thesis proposes new techniques in order to cope with the new features. As we have seen, this is a complicated matter, and others [3] have proposed techniques which uses physical parameters to verify if QoS optimized attacks has taken place as described in section 3.2. However, the thesis questions the use of physical parameters in an urban and mobile environment, and throughout this report I propose new and more reliable techniques which use protocol

specific analysis to detect abnormalities and attacks. The outcome of this thesis was divided into two parts. The first part was developing an IDS, named SIDS v2.0 which implements new detection techniques to cope with the 802.11e/QoS problems described in the introduction. The second contribution is a new proposed algorithm which is a more general approach to the same problem. This algorithm is provided in chapter 7 and is based on analytic and theoretical analysis of the 802.11 standard and the 802.11e amendment, as well as some of the techniques implemented in SIDS v2.0.

9.1 Simple Intrusion Detection System v2.0

SIDS v2.0 has really grown during the writing of this thesis. It has been optimized to deal with how to handle several MAC Sequence counter simultaneously, as well as keep track of many several STAs and detection checks at the same time. The IDS also implements a detection technique in order to verify if a QoS optimized attack is taken place as shown in section 3.2 and 5.1.2. I find that by developing SIDS according to the scheme provided in sections 3.3 and 3.4, the IDS has become more programmable understandable, and should be a proper framework in further development of the IDS. Nevertheless, as stated in chapter 8, it also opens up for several subjects that are classified as future work before we can consider SIDS to be a complete Intrusion Detection System. Perhaps a name change of the IDS should be considered.

9.2 Enhanced Algorithm For QoS Optimized Attacks

In chapter 6 I elaborated on new detection techniques which should be a counter measure to the problems from section 3.2. These techniques are based on analytic discussion of the 802.11e amendment, and should be suitable for implementation in wireless Intrusion Detection Systems such as SIDS v2.0. It is reasonable to believe that from an attackers perspective, it would be difficult to avoid the system, as it use protocol and behavior analysis in order to detect attacks. As a summary of this discussion, an enhanced algorithm for detection of QoS optimized attacks are provided in chapter 7. The techniques proposed here does not rely in uncertain physical parameters like RSS measurements, but instead concentrates on how 802.11e operates and is expected to behave. As described in chapter 8 section 8.2 I am proposing that the algorithm should be implemented and tested in SIDS v2.0, and I believe that the ideas presented here should be a great enhancement to Intrusion Detection Systems in general.

References

- [1] IEEE. IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Technical report, IEEE, 2007.
- [2] Eirik Holgernes and Øystein Aas Pedersen. How to uncover session hijacking in open 802.11 networks. Technical report, NTNU, 2009.
- [3] Gayathri Chandrashekar and John Austen Francisco and Vinod Ganapathy and Marco Gruteser and Wade Trappe. Detecting identity spoofs in 802.11e wireless networks. In *GC'09: Proceedings of the IEEE Globecom 2009 Communications and Information Security Symposium*, Honolulu, Hawaii, November/December 2009. IEEE Computer Society Press, Los Alamitos, California, USA.
- [4] R. Gill, J. Smith, M. Looi, and A. Clark. Passive Techniques for Detecting Session Hijacking Attacks in IEEE 802.11 Wireless Networks. In *AusCERT Asia Pacific Information Technology Security Conference Refereed R&D Stream*, page 26, 2002.

- [5] Jason Smith. Denial of service : prevention, modelling and detection. Master's thesis, Queensland University of Technology, 2008.
- [6] Wilson M. Yeung and Joseph K. Ng. Wireless lan positioning based on received signal strength from mobile device and access points. *Real-Time Computing Systems and Applications, International Workshop on*, 0:131–137, 2007.
- [7] Øystein Aas Pedersen. Detecting Wireless Identity Spoofs in Urban Settings Based on Received Signal Strength Measurements. Master's thesis, NTNU, 2010.
- [8] J. Edney and W.A. Arbaugh. *Real 802.11 security: Wi-Fi protected access and 802.11i*. Addison Wesley Publishing Company, 2004.
- [9] Davide Papini. An Anomaly based Wireless Intrusion Detection System. Master's thesis, Kongens Lyngby, 2008.
- [10] K. Scarfone and P. Mell. Guide to intrusion detection and prevention systems (idps). *NIST Special Publication*, 800:94, 2007.
- [11] I. Response, W. Policy, and J. Dixon. Wireless Intrusion Detection Systems.
- [12] Wi-Fi Alliance. Wi-fi certified for wmm - support for multimedia applications with quality of service in wi-fi networks. Technical report, Wi-Fi Alliance, 2004.
- [13] Ørjan Bækkelund. Session hijacking in WLAN based public networks. Master's thesis, NTNU, 2009.



Valid Types and Subtypes

Type value	Type Description	Subtype Value	Subtype description
00	Management	0000	Association Request
00	Management	0001	Association Response
00	Management	0010	Reassociation request
00	Management	0011	Reassociation response
00	Management	0100	Probe request
00	Management	0101	Probe response
00	Management	0110-0111	Reserved
00	Management	1000	Beacon
00	Management	1001	ATIM
00	Management	1010	Disassociation
00	Management	1011	Authentication
00	Management	1100	Deauthentication
00	Management	1101	Action
00	Management	1110-1111	Reserved
01	Control	0000-0111	Reserved
01	Control	1000	Block Ack Request
01	Control	1001	Block Ack
01	Control	1010	PS-Poll
01	Control	1011	RTS
01	Control	1100	CTS
01	Control	1101	ACK
01	Control	1111	CF-END
10	Data	0000	Data
10	Data	0001	Data + CF-Ack
10	Data	0010	Data + CF-Ack + CF-Poll
10	Data	0100	Null (no data)
10	Data	0101	CF-Ack (no data)
10	Data	0110	CF-Poll (no data)
10	Data	0111	CF-Ack + CF-Poll (no data)
10	Data	1000	QoS Data
10	Data	1001	QoS Data + CF-Ack
10	Data	1010	QoS Data + CF-Poll
10	Data	1011	QoS Data + CF-Poll + CF-Ack
10	Data	1100	QoS Null (no data)
10	Data	1101	Reserved
10	Data	1110	QoS CF-Poll (no data)
10	Data	1111	QoS CF-Ack + CF-Poll (no data)
11	Reserved	0000-1111	Reserved

Table A.1: Valid type and subtype combinations in 802.11 frames

B

SIDS V2.0 code

B.1 sidsv2.pl

```
#!/usr/bin/perl
2 #
# SIMPLE INTRUSION DETECTION SYSTEM v2.0
4 #
#           by
6 #
#           Eirik Holgernes
8 #
10 use Net::Pcap;
    use NetPacket::Ethernet;
```

```

12 use NetPacket::IP;
   use NetPacket::TCP;
14 use Bit::Vector;

16 use IO::Handle;

18 my (
   $pcap_err ,
20   $pcap_descr ,
   $headeroffset ,
22   $READTIMEOUT,
   $SNAPLEN,
24   $PROMISC,
   $TIMEOUT,
26   $INTERFACE,
   $srcmac ,
28   $filecount ,
   @bssid_list ,
30   $file
   );
32
my $logfile = "/var/log/sids";
34
my %SrcSeq = (); # Source MAC/Sequence Number combination
36
my %SrcAlm = (); # Source MAC/Alarm Counter combination
38
my $dumpfile = "/home/rybak/Master/no_qos.cap"; # Execute this script
   on an offline cap file
40
my %qos_mac = (); #QoS/MAC combination
42
my %cnt_mac = (); #Counter and MAC combo
44
my %mac_state = (); # Mac adress and 802.11 state
46
# Due to testing purposes we will set one MAC address as allready in
   state 3 (11) —> PAPER!
48 my $test_mac = "002608e8f8a3";

```



```

50 $mac_state{ $test_mac } = "11";
52 $READTIMEOUT = 1000; # Read timeout on packages
54 @bssid_list = ("00235d0dfbf0", "future"); # Static added for now (LAP
    at office)
56 $PROMISC = 1; # Listen to all packets, Promiscuous mode
58 $SNAPLEN=200; # How much of each frame to be picked up
60 $INTERFACE = "wlan0"; # Defines interface for listening
62 $filecount = 0;
64 # Open capture interface, and loop on each captured packet
    print "Capturing traffic ...\n";
66
68
70 # live packet capture
    # my $capture = Net::Pcap::open_live($INTERFACE, $SNAPLEN, $PROMISC,
        $READTIMEOUT, \ $pcap_err);
72
    # offline packet capture
74
76 my $capture = Net::Pcap::open_offline($dumpfile, \ $err);
78 my $w802 = Net::Pcap::datalink_name_to_val('DLT_IEEE802_11');
80 Net::Pcap::set_datalink($capture, $w802);
82 Net::Pcap::loop($capture, -1, \&process_pkt, '');
84

```

```

86 unless ( defined $capture ) {
      die 'Unable to create packet capture on device ', "wlan0", ' - ',
          $pcap_err;
88 }
die 'Unable to perform packet capture'
90 unless Net::Pcap::loop( $capture, -1, \&process_pkt, '' );

92 # close pcap
if ( $capture ) {
94     Net::Pcap::close( $capture );
}
96
sub process_pkt {
98     my ( $data, $hdr, $packet ) = @_;

100     my $framectl = unpack( 'B*', substr( $packet, 24, 1 ) ); # The 802.11
                       header starts from the 24th byte
102     my $dst_id = unpack( 'H*', substr( $packet, 28, 6 ) );
103     my $src_id = unpack( 'H*', substr( $packet, 34, 6 ) );
104     my $bss_id = unpack( 'H*', substr( $packet, 40, 6 ) ); # Not used
                       by now
105     my $seq_ctl = unpack( 'H*', substr( $packet, 46, 2 ) ); # Keeps the
                       sequence number
106     my $qos = unpack( 'B*', substr( $packet, 53, 2 ) ); # These are
                       the QoS bits
107     my $timestamp_m = ${ $hdr }{ 'tv_usec' }; # Timestamp
                       in milliseconds for this packet
108     my $timestamp_s = ${ $hdr }{ 'tv_sec' }; # Timestamp
                       in seconds for this packet

# Only the 12 last bits on the Seq CTL field used for seq number.
# Counting to 4096 (2^12 = 4096) and loops
110 my $seqnr2 = substr( $seq_ctl, 2, 2 ).substr( $seq_ctl, 0, 1 );

112 my $seqnr = hex( $seqnr2 );

114 # Type of frame
my $frametype = substr( $framectl, 4, 2 );
116 my $subtype = substr( $framectl, 0, 4 );

```

```

118 my ( $subtype_raw, $type_raw ) = split( //, unpack( 'H*', substr(
    $packet, 32, 1 ) ) );
120 # Extract the TID bits from QoS field
122 my $tid = substr($qos, 0, 3);
124 # Check if an ICMP echo check is required:
126 if (-e $ipadress) {
128     print "I will now check for duplicate ICMP reply's";
130 }
132 # Check if the frame is of our conserne, else drop
134 foreach (@bssid_list) {
136     if ($_ eq $dst_id) {
138         # packet is interesting, and intended to be use with our
140         LAP
142         # Filter on management frames, verify 802.11 states
144         if ($frametype == "00") {
146             # Association Request
148             if ($subtype == "0000") {
150                 if (!($mac_state{ $src_mac } == ("11" or "10"))) {
152                     print "Attack! $src_id already associated\n";
154                 } elsif ($mac_state{ $src_mac } == "01") {
156                     $mac_state{ $src_mac } = "11";
158                 } elsif ($mac_state{ $src_mac } == "00") {
160                     # Should NOT happen!
162                     $mac_state{ $src_mac } = "10";
164                 }
166             }
168             # Association Response
170             if ($subtype == "0001") {
172                 print "Association Response \n";
174             }
176             # Reassociation Request
178             if ($subtype == "0010") {

```

```

154     print "Reassociation Request \n";
155 }
156
157 # Reassociation Response
158 if ($subtype == "0011") {
159     print "Reassociation Response \n";
160 }
161
162 # Disassociation
163 if ($subtype == "1010") {
164     if ($mac_state{ $src_mac } == "10") {
165         $mac_state{ $src_mac } = "00";
166     } elseif ($mac_state{ $src_mac } == "11") {
167         $mac_state{ $src_mac } = "01";
168     }
169     print "Disassociation \n";
170 }
171
172 # Authentication
173 if ($subtype == "1011") {
174     if (!($mac_state{ $src_mac } == ("11" or "01"))) {
175         print "Attack! $src_id already authenticated\n";
176     } elseif ($mac_state{ $src_mac } == "00") {
177         $mac_state{ $src_mac } = "01";
178     } elseif ($mac_state{ $src_mac } == "10") {
179         $mac_state{ $src_mac } = "11";
180     }
181 }
182
183 # Deauthentication
184 if ($subtype == "1100") {
185     $mac_state{ $src_mac } = "00";
186     print "Deauthentication \n";
187 }
188
189 }
190
191 # Proceed with Sequence Number Analysis

```

```

194         if ($src_id =~ m/^./) {
                keepsequence($src_id, $seqnr, $frametype, $subtype,
                            $tid, $timestamp_m, $timestamp_s);
                }
196     }
198 }
200 }
202 # Step 1a
sub keepsequence {
204     my ($MAC, $SEQ, $FRT, $SUBTYPE, $TID, $TIMESTAMP_M, $TIMESTAMP_S) =
            @_;

206     # Drop processing packets if a sequence check is ongoing (sample
            testing) DISCUSS IN PAPER!
            # Koble timestamps fra offline mode til en real-time setting. (se
            pÑ lokaltiden her mot timestamps)

208

210     # 1a) Keep one sequence for each MAC adress (on disk), and
            keep the parameters.

212

214     # my $lines = 'wc -l < $MAC' if (-e $MAC);

216     # Keep a sequence of 10 frames for this specific MAC adress
            if ($cnt_mac{ $MAC } < 55000) {
218         open(MACFILE, '>>', $MAC);
                print MACFILE "$MAC:$SEQ:$FRT:$SUBTYPE:$TID:$TIMESTAMP_M:
                    $TIMESTAMP_S\n";
220         close(MACFILE);
            } else {
222

                # rename($MAC, $MAC.".check") or die "Could not rename
                    file: $MAC";
224

```

```

226     # $file = $MAC.".check";

228     # Call a background process check
    print "Kaller pñ bakgrunnsprosess med parametre: $MAC
        count is $cnt_mac{ $MAC } \n\n";
    exec("/usr/bin/perl", "/home/rybak/Master/check.pl", $MAC
        ) or die "exec failed: $?";
230 }

232 my $count = $cnt_mac { $MAC };
    $count = $count + 1;
234
    $cnt_mac{ $MAC } = $count;
236 }

```

B.2 check.pl

```

#!/usr/bin/perl
2 #
# SIMPLE INTRUSION DETECTION SYSTEM v2.0
4 #
#
6 #           by
#
8 #       Eirik Holgernes
#
10 # ———— CHECK SEQUENCE SCRIPT ————
#
12 #

14 my $logfile = "/var/log/sids";

16 my $MAC = $ARGV[0];
    print "background process called with argument: $MAC";
18
    check_linprog($MAC);

```

```

20 |
21 | # Step 2a and 2b
22 | sub check_linprog {
23 |     print "Starting up check_linprog \n\n";
24 |     my $file = $_[0];
25 |
26 |     #Change in the future ... for now
27 |     my $counter = 0;
28 |     my $OLD = 0;
29 |
30 |     # Extract each sequence number, compare and test for linear
31 |     progression
32 |     open (MACFILE, $file);
33 |     while (<MACFILE>) {
34 |         chomp;
35 |         my ($MAC, $SEQ, $FRT, $FRTSUB, $TID, $TIME_M, $TIME_S) =
36 |             split (/:/, $_);
37 |
38 |         # Check if linear progression or within threshold
39 |         if (($SEQ - $OLD > 10) && ($OLD != 4095) && ($SEQ != 4095)) {
40 |             $counter = $counter+1;
41 |         }
42 |         $OLD = $SEQ;
43 |     }
44 |     # Allow 5 errors for now :)
45 |     if ($counter > 5) {
46 |         # Frames are not in linnear progression, go to step 2c)
47 |         check_frtypes($file);
48 |     } else {
49 |         # Frames are in linnear progression, conclude no attack and
50 |         drop sequence
51 |         unlink($file);
52 |         print "\nFrames were in linear progression, do not contiinue
53 |             check \n";
54 |     }
55 |
56 |     close (MACFILE);

```

```

56 }
58 # Step 2c
sub check_frtypes {
60     print "Starting up check_frtypes \n\n";
        # At this point the frames are in the sequence are NOT in linear
            progression, lets check to see if the frames are non-QoS
62     my $file = $_[0];

64     my $counter = 0;
        my $OLD_TID;

66     open (MACFILE, $file);
68     while (<MACFILE>) {
        chomp;
70     my ($MAC, $SEQ, $FRT, $FRTSUB, $TID, $TIME_M, $TIME_S) =
            split (/:/, $_);
        # TID class set to "000" if QoS are not used?
72     if ($TID != "000") {
            $counter = $counter + 1;
74     }

76     }

78     if ($counter > 0) {
        # Assuming the frametypes are QoS, lets check if all priority
            classes are equal
80     equal_priority($file);

82     } else {
        # Frames are either of type MGMT or Regular Data. Conclude
            attack, and drop sequence.
84     my $logmsg = "Step 2c - Attack on Source MAC: $MAC";
        write_log($logmsg);
86     unlink(MACFILE);
        }
88     close (MACFILE);

```



```

90 | }
92 |
94 | # Step 2d – Assuming that the frames are QoS type, lets check if the
   | priority classes are equal everywhere
   | sub equal_priority {
96 |     print "Starting up equal_priority \n\n";
   |     my $file = $_[0];
98 |
   |     my $OLD_TID = 0;
100 |
   |     my $equal = 0;
102 |
   |     open (MACFILE, $file);
104 |     while (<MACFILE>) {
   |         chomp;
106 |         my ($MAC, $SEQ, $FRT, $FRSUB, $TID, $TIME_M, $TIME_S) =
   |             split (/:/, $_);
   |         if (!$TID == $OLD_TID) {
108 |             my @parts = gmtime($TIME_S);
   |             print "The equal counter increased at SEQ: $SEQ at time "
   |                 ;
110 |             printf ("%4d:%4d:%4d\n", @parts[2,1,0]);
   |             print "\n";
112 |             $equal = $equal+ 1;
   |         }
114 |     }
   |
116 |     if ($equal > 1) {
   |         # Priorities are either mixed or mixed QoS-data and regular
   |         data, initiate sequence 2e
118 |         print "\n Equal counter is now on: $equal \n";
   |         verify_attack($file);
120 |     } else {
   |         # Priority classes are equal all the way, conclude attack,
   |         and drop sequence
122 |         my $logmsg = "Step 2d – Attack on Source MAC: $MAC";
   |         write_log($logmsg);

```

```

124     unlink(MACFILE);
125     }
126     close(MACFILE);
127 }
128 }
129
130 # Step 2e
131 sub verify_attack {
132     # Active probing from ICMP echoes in ping
133
134     # By now I just set a static IP to a client (this could be read
135     # from any TCP packet)
136     my $ipaddress = "10.0.0.22";
137
138     # Create a file on disk with the IP. This will tell the main
139     # program, that an ICMP check is required
140     system("touch", $ipaddress);
141
142     # Initiate the ICMP echo request, will wait untill finished
143     system("ping", "-c 1", $ipaddress);
144
145     #
146 }
147
148 # Simple LOG writing function
149 sub write_log {
150     my ($logmsg) = @_ ;
151
152     my ($sec, $min, $hour, $mday, $mon, $year) = localtime(time);
153
154     open LOGFILE, ">>$logfile" or die "cannot open logfile $logfile
155     for appending: $!";
156
157     print "\n ##### LOGGED!! ##### \n";
158 }

```

```
160     printf LOGFILE "%4d-%02d-%02d %02d:%02d:%02d --> ", $year+1900,  
162         $mon+1,$mday, $hour, $min, $sec;  
    print LOGFILE $logmsg, "\n";  
    close LOGFILE;  
}
```