

4TH YEAR PROJECT 2011

Honeypot IDS

SNORT Intrusion Detection System

Ruairi MacTiernan - B00029564

An investigation into malicious and dangerous network traffic with the SNORT Intrusion Detection System and Rapid Miner.

Declaration of Original work

I certify, by penalty of perjury, that this is my original work; except where sources are duly acknowledged, and that this project has not been submitted in part or in whole to fulfil the requirements of any other subject or course.

X

Ruairi MacTiernan
B00029564

Abstract

This paper relates to a project that was done for a 4th year project in Blanchardstown IT in 2011 entitled '*Honeypot IDS*' which was designed to setup and monitor an IDS system on a live network.

This paper will look at the different types of IDS systems that are available as well as looking at the differences between an IDS and a Honeypot, as well as what they are designed to protect against.

We will see how the system is designed and setup as well as how it is run. On completion of the operational stages of the system we will then examine how the results are extracted from the database and look at how they are examined.

With the extracted results we will need to mine through them to try to find the malicious data, for the purpose of creating new rules that may stop them in the future.

By the end of the project I will have a good understanding of the network traffic that has been captured and what makes up some of the more interesting packets.

Acknowledgements

I would like to thank my project supervisors Michael O'Donnell and Geraldine Gray for their help and guidance during this project till its completion.

I would also like to thank my family for allowing me to hijack the home computer network and trusting me not to kill it.

Table of Contents

Declaration of Original work.....	2
Abstract	3
Acknowledgements.....	4
Table of Figures:	9
Chapter 1: Introduction.....	12
1.1: Chapter introduction	12
1.2: What is a Honeypot.....	12
1.2.1: Natural world example's.....	12
1.2.2: IT Application's.....	13
1.3: Honeypot History	14
1.4: What is an IDS.....	15
1.5: How it works.....	16
1.5.1: Stage 1	18
1.5.2: Stage 2	18
1.5.3: Stage 3	19
1.5.4: IDS.....	20
1.6: Why do we use them.....	21
Chapter 2: Project Proposal	24
2.1: Chapter Introduction	24
2.2: Proposal	24
2.3: Project Goals.....	25
2.4: Expected Outcome	26
2.4.1: Practical Outcome	26
2.4.2: Learning Outcome.....	26
Chapter 3: Literary review	28
3.1: Chapter Introduction	28
3.2: Reasons for Security detection	28
3.3: Systems.....	29
3.3.1: IDS.....	29

3.3.2: Honeypots	30
3.3.3: Passive 'v' Reactive	31
3.4: Types of Attacks.....	32
3.4.5: Denial of Service (DoS) attacks:	34
3.4.6: Port Scanning	37
3.4.7: Tampering and Spoofing.....	38
3.4.8: ICMP Attack (aka Smurfing).....	39
3.5: Attackers.....	40
3.4.1: How they happen.....	40
3.4.2: What to look for when being attacked	41
3.4.3: What to do when being attacked	41
3.6: What is produced in the records	43
3.6.1: How do we distinguish an attack.....	43
3.6.2: What areas to look for	44
3.7: Processing results	46
3.7.1: What do we use	46
3.7.2: Data mining techniques.....	46
3.7.3: Designs of interest	47
3.7.4: Data mining of network attacks	49
Chapter 4: System Analysis and Design.....	52
4.1: Introduction.....	52
4.2: Evaluation.....	52
4.2.1: Operating systems	52
4.2.2: Attacker	55
4.2.3: IDS / Capture	56
4.2.3.1: Open source.....	57
4.2.3.2: Commercial.....	58
4.3: Project setup.....	60
4.3.1: Setup	60
4.4: Requirements.....	63
Chapter 5: The Project	64
5.1: Chapter Introduction	64

5.2: Phase 1	64
5.2.1: The setup.....	64
5.2.2: Equipment.....	65
5.2.3: System Additional	66
5.3: Phase 2	66
5.3.1: Implementation and setup.....	66
5.3.2: Running IDS.....	67
5.3.3: Relationship between systems	69
5.4: Phase 3	70
5.4.1: System shutdown	70
5.4.2: Checking records	70
5.4.3: Finding interesting information.....	72
5.4.4: Processing of data	72
5.5: Phase 4	73
5.5.1: Results	73
5.5.2: Conclusions / pre-Conclusions	74
Chapter 6: Data Analysis	75
6.1: Chapter Introduction	75
6.2: Design introduction.....	75
6.2.1: Imported data	76
6.2.2: Standard design Features.....	79
6.2.3: Outlier operator	79
6.2.4: Other areas for information.....	85
6.3: Process 1- Detect Outliers (IPHdr)	86
6.3.1: Introduction.....	87
6.3.2: Process design setup.....	89
6.3.3: Examining results.....	92
6.3.4: Graphing the results.....	93
6.3.5: Conclusion.....	97
6.4: Process 2 – Detect outliers (TCPHdr).....	98
6.4.1: Introduction.....	99
6.4.2: Process design setup.....	101
6.4.3: Examine results.....	103

6.4.4: Graph Results	104
6.4.5: Conclusion.....	107
6.5: Process 3 – Detect outliers (UDP)	108
6.5.1: Introduction.....	109
6.5.2: Process design setup.....	111
6.5.3: Examining results.....	113
6.5.4: Graphing results.....	114
6.5.5: Conclusion.....	117
6.6: Process 4 – Detect outliers (ICMPHdr)	121
6.6.1: Introduction.....	121
6.6.2: Process design setup.....	125
6.6.3: Examine results.....	128
6.6.4: Graphing results.....	129
6.6.5: Conclusion.....	132
6.7: Process work conclusions	134
6.8: Writing a new rule	135
Chapter 7: Conclusions and Further work	139
7.1: Chapter Introduction	139
7.2: Outcome of work	139
7.3: Conclusions from Processes.....	140
7.3.1: Timestamps.....	141
7.4: Possible future work.....	141
7.4.1: Outlier detection and Classification	142
Chapter 8: Personal Reflections	143
Bibliography.....	144
Appendix 1: Diagram Key	152

Table of Figures:

Figure 1: Venus Fly Trap (America)	13
Figure 2: Honeypot network (MacTiernan, 2010/2011)	17
Figure 3: Attacker finds entry point to network (MacTiernan, 2010/2011)	18
Figure 4: Attacker diverting to Honeypot (MacTiernan, 2010/2011)	19
Figure 5: IDS Setup (MacTiernan, 2010/2011)	20
Figure 6: SNORT Capture	20
Figure 7: Fake Email about a Blizzard, World of Warcraft account	22
Figure 8: Gantt chart of project time	25
Figure 9: DOS attack (MacTiernan, 2010/2011)	35
Figure 10: Port scanning (MacTiernan, 2010/2011)	38
Figure 11: IRC chat network (Wikimedia)	39
Figure 12: Decision Tree (Time-Management-Guide.com)	48
Figure 13: Outlier example (Bloom)	49
Figure 14: OS percentages (Michael DeAgonia)	54
Figure 15: Recording stages (MacTiernan, 2010/2011)	62
Figure 16: System requirements table	63
Figure 17: Home network setup (MacTiernan, 2010/2011)	65
Figure 18: SNORT command line start	67
Figure 19: SNORT initializing	68
Figure 20: System relations (MacTiernan, 2010/2011)	69
Figure 21: SNORT end of use report	71
Figure 22: Read Database operator	73
Figure 23: Event Meta View	74
Figure 24: ERD Diagram - Tables that will be used	76
Figure 25: ERD Diagram - Tables not used	77
Figure 26: Included tables in rapid miner	78
Figure 27: Densities parameters	80
Figure 28: Densities outliers' classification	80
Figure 29: LOF parameters	82
Figure 30: LOF Outliers	83
Figure 31: Detect Outlier (COF) distance functions (rapid-i)	84
Figure 32: SNORT output files	85

Figure 33: Alert folder.....	85
Figure 34: Alert file contents.....	86
Figure 35: IPHdr process information	86
Figure 36: IP header contents (Wikipedia)	87
Figure 37: IP header Meta data	88
Figure 38: Process 1 - IPHdr design.....	89
Figure 39: Process 1 setup values.....	91
Figure 40: Detect IP outliers	92
Figure 41: Detect IP outliers close view.....	93
Figure 42: Outlier graph (x = outliers / y = destination)	94
Figure 43: Outlier graph (x = timestamp / y = protocol)	95
Figure 44: Outliers filtered (x = sig_class_name / y = sig_name)	96
Figure 45: Segment of data view (sig id - 11307)	97
Figure 46: TCP process information	98
Figure 47: TCP header contents (Wikipedia)	99
Figure 48: TCP header Meta data.....	100
Figure 49: Process 2 – TCPHdr design.....	101
Figure 50: Process 2 setup values.....	102
Figure 51: Detect TCP outliers.....	103
Figure 52: Detect TCP outliers close view	104
Figure 53: Outliers graph (x = outliers / y = source port)	104
Figure 54: Outliers graph (x = sequence value / y = off value)	105
Figure 55: Outliers graph (x = destination port / y = source port)	106
Figure 56: Segment of data view (sport = 49319)	107
Figure 57: UDP process information.....	108
Figure 58: UDP header (Wikimedia).....	109
Figure 59: UDP Meta data	110
Figure 60: Process 3 - UDPHdr design.....	111
Figure 61: Process 3 setup values.....	112
Figure 62: Detect UDP outliers	113
Figure 63: Detect UDP outliers close view	114
Figure 64: Outlier graph (x = outlier / y = udp_dport).....	115
Figure 65: Outliers graph (x = outliers / y = udp_csum)	116
Figure 66: Outliers graph (x = udp_dport / y = sport)	117

Figure 67: Stand out entire	117
Figure 68: Port 53	118
Figure 69: Port 53 access	119
Figure 70: Port 1900.....	119
Figure 71: Port 1900 access.....	120
Figure 72: ICMPHdr process information	121
Figure 73: ICMP ping message, host reached	121
Figure 74: ICMP ping message, host unreachable.....	122
Figure 75: Trace root ICMP	123
Figure 76: ICMP header contents (Wikimedia).....	123
Figure 77: ICMP type/code example, Type no. 3 (Wikimedia).....	124
Figure 78: ICMP header Meta data	125
Figure 79: Process 4 - ICMPHdr design.....	126
Figure 80: Process 4 setup values.....	127
Figure 81: Detect ICMP outliers	128
Figure 82: ICMP attack window	128
Figure 83: Checksum function	129
Figure 84: Outliers graph (x = CID / y = sig_class_name)	130
Figure 85: Outliers Graph (x = CID / y = sig_name).....	131
Figure 86: Outliers graph (x = outliers / y = sig_name).....	131
Figure 87: Outliers graph (x = sig_class_name / y = sig_name).....	132
Figure 88: ICMP Port-scan.....	133
Figure 89: IP Look-up (92.62.43.77) (IP-Lookup)	133
Figure 90: IP Look-up (85.236.110.226) (IP-Lookup)	133
Figure 91: DDOS rule example.....	136
Figure 92: SNORT Rule options (Roesch)	136
Figure 93: Our Simple new rule	137
Figure 94: Logged RTE event	137
Figure 95: Custom rule to register IRC data.....	138
Figure 96: Logged IRC chat server classification.....	138

Chapter 1: Introduction

1.1: Chapter introduction

In this chapter I will be outlining the theory behind the use of a Honeypot and some of its additions; we will also look at how the system takes shape and the reasons behind its use. The explanation of what a Honeypot is will be looked at in respect to the more formal IT standards but also to some less formal to try to explain what it is.

1.2: What is a Honeypot

So what is a honeypot? In essence, a Honeypot is something that is designed to attract and trap something for its own means and use. It will sit idle listening and waiting for something of interest to trigger its sensors and cause a reaction that will produce some information or physical grabbing of its target. The idea of the Honeypot is all around us, in the natural and un-natural world, even though we may not see them at first glance. Honeypot's are not a new thing but where we are today we have had to take the natural examples and adapt them to work for the un-natural application.

1.2.1: Natural world example's

In the Natural world there is one example above all others that people will know to see but not realise that it can relate to a Honeypot. This example is the *Venus Fly Trap* (see Figure 1). The Venus is one of the very, very rare examples of a *carnivore's* plant (i.e. It eats meat), what it does is it will sit idle blowing in the wind, but it attracts fly's and other small insects to it by excreting a sweet smelling and possible tasting liquid from its opening.



Figure 1: Venus Fly Trap (America)

The leaves of Venus' Flytrap open wide and on them are short, stiff hairs called trigger or sensitive hairs. When anything touches these hairs enough to bend them, the two lobes of the leaves snap shut trapping whatever is inside. The trap will shut in less than a second. (America)

Once something enters the mouth it will crawl to the bottom and along the way will brush against hairs triggering the trap and closing the mouth. Its prey is then digested and consumed for the plant to survive and live.

This is not quite the same as the Honeypot in IT but is still very similar, and can almost identical in action depending on the system that is implemented.

1.2.2: IT Application's

In IT a Honeypot is a little different, but as was said already can be almost the same depending on the implementation of it that is used. What the Honeypot does in IT and Computer systems is act as a security device for a network. It will sit on the network, usually in a De-Militarized Zone (DMZ¹) and monitor the network traffic for any signs of dangerous traffic. However there are two main types that are available, the Honeypot or Honeynet and the Intrusion Detection System (IDS).

¹ DMZ = A section of a network that is completely open to everyone both inside and outside that network. Usually used for Web Servers or anything else that access is required by non-company members.

A Honeypot will mimic a vital system service such as a Windows Server and purposely attract any dangerous data that enters the network into itself, keeping the main network safe from harm. The Honeynet is a collection of Honeypots that interlink throughout a network, monitoring different systems, but in essence does the same thing as a Honeypot.

“A honeypot is a decoy system put on a network as bait for attackers. The attackers believe the honeypot is a legitimate system and attack on it, without being known that their activities are being monitored.” (Chinese University of Hong Kong)

At the core of all Honeypots are the IDS which are the monitoring system of network traffic. The Honeypot is not actually one specific type of software or hardware but is in fact a collection of different elements that make it up.

- IDS
- Mimic system (Windows Server)
- Packet Analyser
- Data Storage of analysed packets

A true Honeypot system will also have the ability to run as a virtual (Low Interaction) or real (High Interaction) system which can mean the difference between cost and protection.

1.3: Honeypot History

The honeypot theory (Honeypot Project²) is a relatively new concept having only been in existence for the past few years. On the 25th of February 1999 the Honeypot concept came into existence when it's founder (Lance Spitzner, ex-Military turned Network Security specialist) plugged a home computer into the internet one evening, and had it hacked within 15 minutes of doing so. To solve this issue Lance Spitzner decided to take a military approach and apply what he learned from his military career.

“My commander use to tell me that to defend against the enemy, you have to first know who your enemy is: their methods of attack, tools and tactics, and objective.”
(Honeynet Porject, 2001)

² The Honeynet Project - <http://www.honeynet.org/>

The Honeynet project was setup to help not necessarily stop the attack, but to understand them with hope to stopping them. To better understand and help set everything up he asked himself a few basic questions,

“... what are the attacker’s goals? What are attackers trying to achieve? Why? How do they identify vulnerable systems and then compromise them? What happens once attackers control a system? How do they communicate among themselves? Are we dealing with a single threat or a verity of threats?...” (Honeynet Poroject, 2001)

With these questions in hand he then proceeded to set up the Honeynet project, a community of people looking to help understand the composition of a malicious attack and why it might be doing this, and to prevent the intrusion of attackers into what is meant to be a secure system, and the advancement of future safeguards against malicious attacks.

1.4: What is an IDS

At the core of this project is the setup and analysis of an IDS, so what is it?

An IDS is the core of the Honeypot, but an IDS system will not attract any traffic like a honeypot. While the Honeypot mimics a system, this one will not, and can in fact be implemented as a firewall with the correct equipment, which means that it is more of a pure monitoring system. An IDS will monitor the traffic as it occurs.

There are two main types of IDS, the Network IDS (NIDS) and the Host IDs (HIDS). The NIDS would be the most common one, sitting at the head of the network reading all incoming traffic before it hits the firewall, however the NIDS can also be the firewall preventing certain types of traffic through regardless if it is classed as safe or not. This can then be tied in with other systems to act as a first stage alert system to turn on any preventative measures that can be used.

The HIDS will monitor only one machine on a network or monitor traffic on a network going to and from one specific machine (like a server), usually this will be attached to a Server of some kind or other vital system, to ensure that all traffic is strict and above board. However you will never have a HIDS on its own, and will always be tied in with a NIDS. Combined these can form their

own version of a Honeynet³ called a *Sensor Server*⁴ system, where the Server would be the NIDS and the Sensors are the HIDS that relay information back to the server with their ID number.

While The Honeypot is how these types of systems are classed, there are very few actually Honeypots in commercial use due to their high risk of attack. Rather the Honeypot is used on more educational or experimental networks. The IDS would normally be desired due to their expandability of systems and implementation of external hardware such as the location and area based sensors, which can then tie into backup systems and emergency systems such as UPS's⁵ and emergency shutdowns or network isolation switches.

1.5: How it works

A Honeypot is a fake system, it doesn't actually exist as it claims to, and all it will do is mimic a real system for the roll of being attacked first. Once a Honeypot is attacked, all of the attacker's information is recorded, and stored in a database for use at a later date.

Honeypot = In computer terminology, a honeypot is a trap set to detect, deflect, or in some manner counteract attempts at unauthorized use of information systems.
(Honeypot)

The IDS will monitor the network traffic for data that it deems as *Not Desirable* and will act accordingly

IDS = Software that detects and logs inappropriate, incorrect, or anomalous activity.
IDS are typically characterized based on the source of the data they monitor: host or network. A host-based IDS uses system log files and other electronic audit data to identify suspicious activity. (Federal Financial Institutions Examination Council)

This IDS is what turns a standard computer or server into the Honeypot. The Honeypot's task is to simulate the actual network or networked machine, but more importantly to read the information from the attacking source. The recording of all the information gathered from the IDS is done

³ Multiple Honeypots combined on a network for form a net

⁴ Sensor is the Host or area IDS and the Server is the recording location or Network IDS

⁵ Uninterruptable Power Supply's – a battery backup in case of a mains power shutdown

through Capture application such as 'PCap' or 'WinPCap'. These capture applications are then able to interoperate the information from the IDS and write it to the database for storage.

The IDS itself can be a fairly simple program or with some of the more expensive programs that incorporate various features (become applications) and can become a lot more complex. The choice for IDS is about as vast as the choice of car that someone might want to buy. However all IDS tools all share a common ground, they are all based on the SNORT application, either currently based or originally during their development. When the Honeypot Project was formed the original IDS that was created for this was one called SNORT. Since The Honeypot Project is an open source (a free thing), the IDS that accompanies it is also free and open source (conforms to the GNU General Public License (Open Source Initiative)).

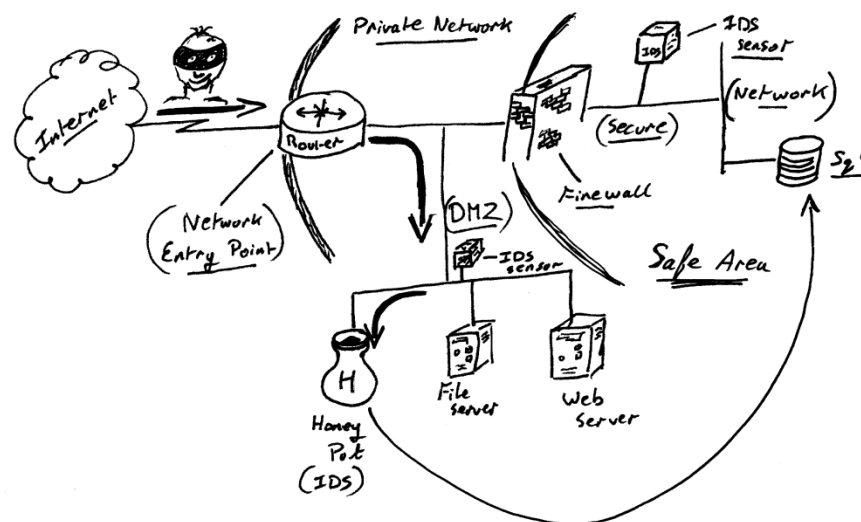


Figure 2: Honeypot network (MacTiernan, 2010/2011)

1.5.1: Stage 1

How the IDS interacts with an attacker:

There is more dangerous data floating around the internet and connected systems than there are people on the earth. So how do these things find us? There are a number of techniques that can be used by these bots and attackers, the most common of which would be “*phishing*”. This is a process of sending out a request and seeing who responds, then when a response is received they go to do further investigation.

When an attacker finds your network they naturally have to enter it through an entry point. Most systems will have only one entry point, and as good practice goes one is all should be available at any given time, it is ok however to have some more as backups (redundancy).

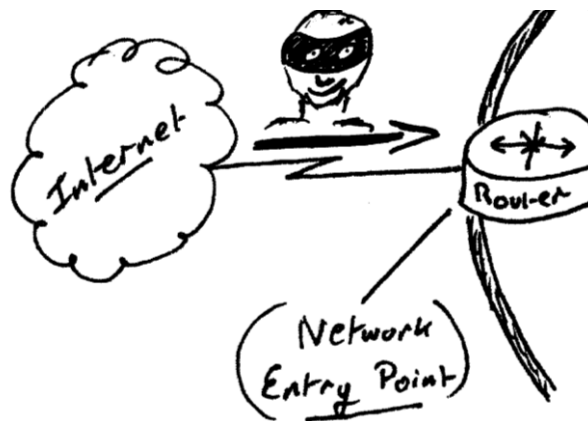


Figure 3: Attacker finds entry point to network (MacTiernan, 2010/2011)

1.5.2: Stage 2

Once the attacker has found the network and has gained entry to it, what happens next can depend on the setup of the overall Honeypot or IDS. After the Router or entry point, there will be a firewall which will prevent pre-defined traffic from passing through. The Honeypot and other public access systems will be placed on the DMZ (see Figure 4). When the attacker enters they should see the

Honeypot which it will mistake for a valid system or some kind, it will then try to penetrate this to see what is available, if anything.

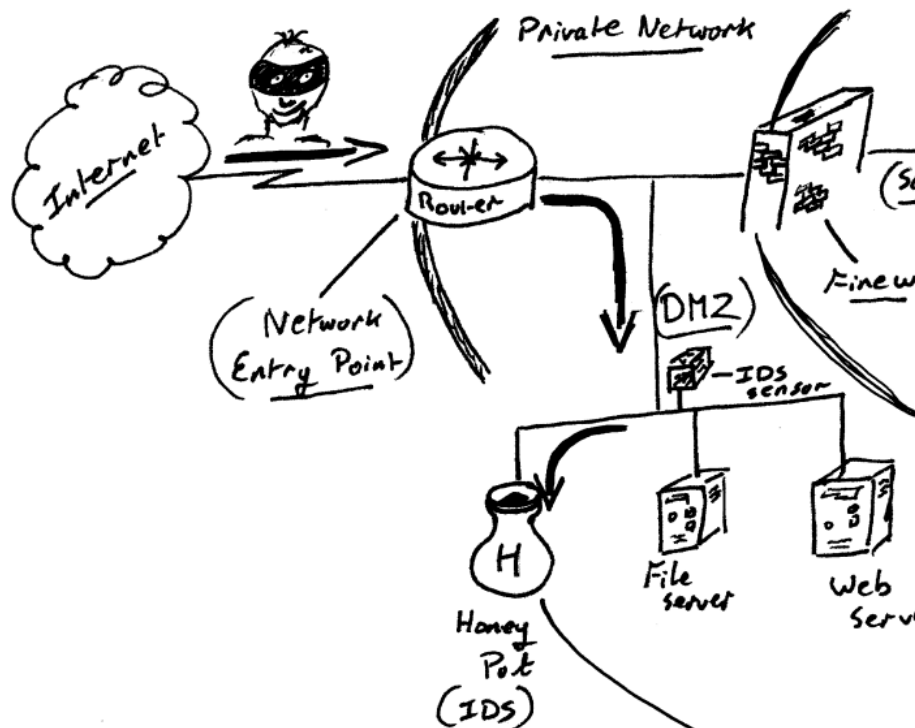


Figure 4: Attacker diverting to Honeypot (MacTiernan, 2010/2011)

1.5.3: Stage 3

Once the Attacker has started to attack its target (in this case the fake system, The Honeypot) all its information will then be recorded and registered for further processing and analysis.

However the danger in allowing the attacker into the DMZ can be that they may disrupt or damage some other systems that are present, such as the Web Server which will also be fairly un-protected due to having to allow non-company connection into it.

It is because of this that the Honeypot method isn't in commercial use as much as the IDS monitoring system. With the Honeypot, everything is held behind the firewall so that any dangerous traffic must get through that first.

1.5.4: IDS

In Figure 5 we see how an IDS is setup behind the Firewall, we may also have an IDS in front of a firewall if we want to ensure that nothing enters that we don't want. These IDS systems can be placed through the network, and even on the actual working machines to ensure that all systems are monitored at all times. There would be one main *Server* IDS and the rest will be classed as *Sensors*.

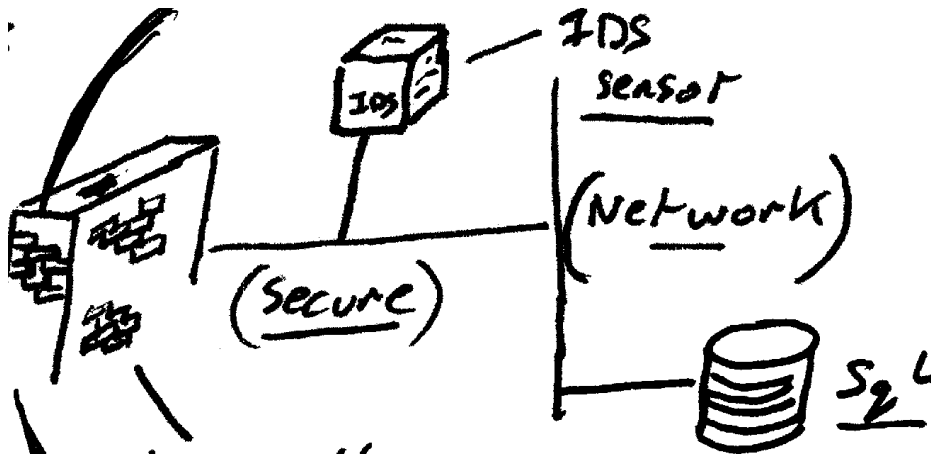


Figure 5: IDS Setup (MacTiernan, 2010/2011)

The advantage with this is that because we have the firewall the firewall can take most of the force of any attacks that happen, and the IDS will monitor the rest. The other advantage of the IDS is that due to their *rule based* monitoring; we can also set up rules to monitor standard 'safe' network traffic to ensure that staff are not accessing sites and resources that they should be accessing on company times.

```
03/13-18:01:03.764010 192.168.1.106:2942 -> 192.168.1.1:5691
TCP TTL:128 TOS:0x0 ID:23470 IpLen:20 DgmLen:40 DF
***A*** Seq: 0xF605369D Ack: 0xAA4EEE61 Win: 0xFD56 TcpLen: 20
=====
```

Figure 6: SNORT Capture

Figure 6 shows a screen output from the SNORT⁶ IDS, which shows some of the available information that can be gathered from a data packet on a network. Some of the available information in this is the Destination and Source IP Address, Time To Live (TTL), Sequence and Acknowledge numbers and length of packet.

With the information available from the recorded packet we should be able to determine if the particular packet is dangerous or not, depending on a number of different attributes, which depends on what type of attack we are looking for.

1.6: Why do we use them

Why do we need or use Honeypot or IDS? Currently one of the most common forms of attack as we have already said is phishing. This entails sending out a request to try to get a response from somewhere. Most people would be familiar with this through fake emails from various banks or other high profile account's, such as online gaming accounts. What they do is try to get your account information to that they can gain access to it to remove all assets that you might have.

Looking at Figure 7 you will notice that there is no mention of 'e-mail your details', rather they ask that you go to a *link* for the purpose of verifying your account. If you were to do this then they would have your details and your account would be history. Phishing with networking, while ultimately is still the same form; the approach is a little different. With the email phishing you send them your details and you never hear from them again, with the network phishing your machine responds to their request and then they start the attack process.

⁶ Open Source IDS that is the foundation for all modern IDS and Honeypot systems

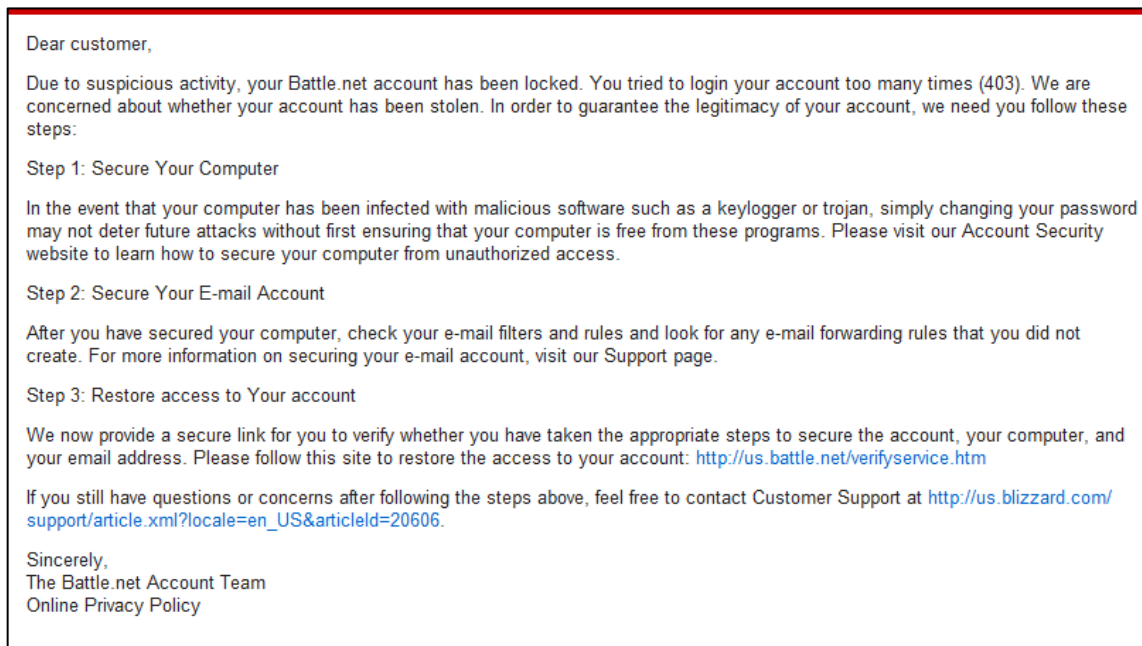


Figure 7: Fake Email about a Blizzard, World of Warcraft account

With internet and network traffic and data packets, the idea and approach is roughly the same. A packet is sent out across the internet to see who is listening; they will know this because a response to their request will be returned looking for an acknowledgment to the request. Once the destination has been found the attack begins.

To prevent these attacks we need some form of preventative measure, but to get these preventative measures we need to understand how the attack occurs, and this is where the IDS comes in. the IDS will allow its users to read all available data relating to the received packet and will allow us to design future preventions against such attacks.

The Honeypot itself would be used more towards the Education side of things rather than the Commercial as was already mentioned, this again is due to the nature of having to actually attack the attacker rather than stop them. Though this does have some advantages in that the attracted attacker will be re-directed towards the Honeypot and away from the network, which in theory will keep it safe. The problem though comes in the setup of the Honeypot. Since there are very few designated Honeypot systems available, and those that are available are highly expensive, Honeypots need to be constructed, if for instance we left the e-mail service online (i.e. connected to

an e-mail server) then that Honeypot has the risk of becoming a sender of the virus, or phishing bot itself, turning your network into a hostile hive for other networks.

The IDS on the other hand has very little risk involved in it. Once setup, it will not stop or prevent dangerous traffic itself but will monitor and classify traffic as it occurs. However the IDS has the ability to indirectly stop the traffic through the addition of other modules and connected systems. The IDS also has the advantage of being able to monitor internal and safe network traffic to determine if staff and network members are accessing things that they shouldn't or are not allowed to.

Chapter 2: Project Proposal

2.1: Chapter Introduction

In this chapter I will discuss the plan for this project, the tools to be used and the format of the project timeline. We will look at what each stage of the project is designed to do and how it will be implemented.

2.2: Proposal

For this project I will set up an IDS System on my own home network, this will be done for the purpose of examining network traffic for signs of dangerous or malicious traffic, and understanding how and why these attacks happen. I will also gain an understanding how to set such utilities up that will monitor and prevent such attacks, including the relevant rules and classification designs.

This project will be based on ‘The Honeynet Project’, a community of Network Safety specialists that came together to form a solid knowledge base of attacker data.

This will be a research project designed to look into the theory and practice of the Honeypot IDS system, and will show how attacks happen into a network and where they come from. The collected data will be analysed using a data mining tool to find the relevant patterns and common threads that make up a malicious attack. Using the collected data from the attacks and mining it to find these patterns, I will then try to design a suitable rule set to prevent any future attacks.

The stages of this project will be broken down into two main areas, 1st Semester and 2nd Semester. 1st Semester will consist mainly of research into what a Honeypot is and definition of the overall project including the goals and predicted results, and which type of honeypot should be chosen. 2nd Semester will be based around the running of the Honeypot and gathering of results, but will still contain a large amount of report work.

From the Gantt chart below (See Figure 8) I show the estimated time line of the project. There is no actual time line placed on this project because it relies nearly entirely on being attacked, this is not really something that can be forced, rather must be waited for. Paper and Report work will remain the

dominant element of the time line but less emphasis will be placed on it in during the majority of the 2nd semester.

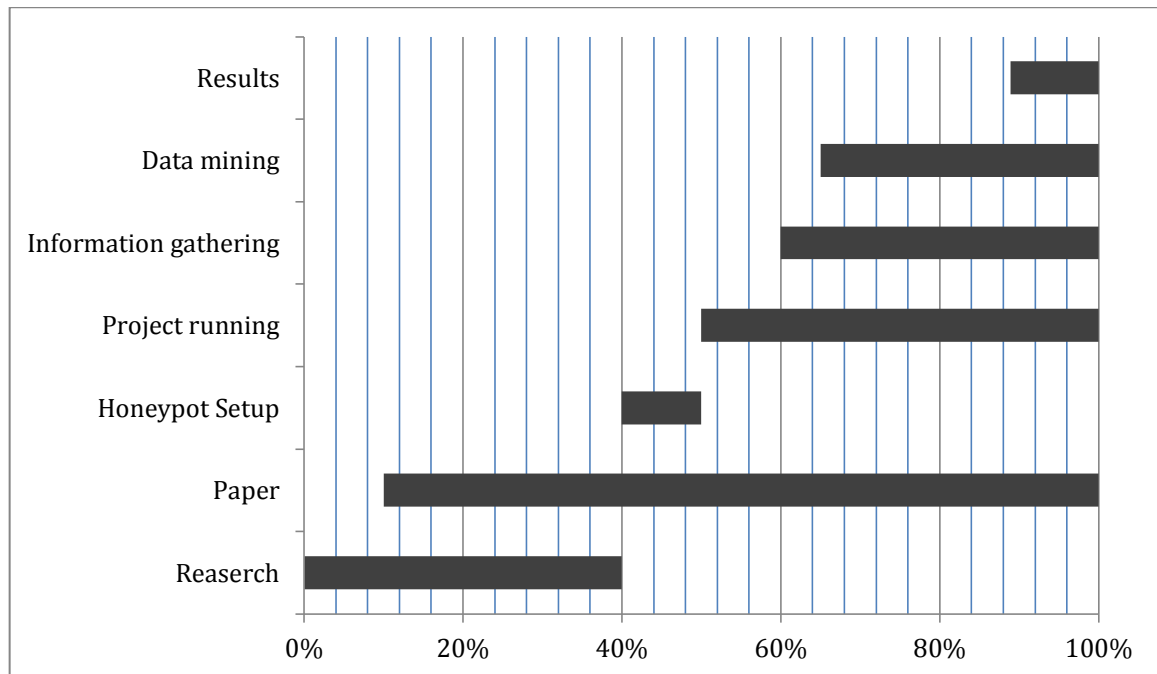


Figure 8: Gantt chart of project time

2.3: Project Goals

The ultimate goal of this project is to understand about malicious data and attacks on a network. Because of the way the world is going with its public and private network structure, things are becoming bigger and bigger and as such more security is needed. However in private areas there is not as much security as there probably should be. With this project I will be trying to understand what the goals of these attacks are, and how to decipher them.

Once I know about how and why this happens we will then be looking into creating some rules that will hopefully try to classify this traffic better. If by the end of the project there is a good enough result, this paper will be looked at being submitted to the Honeynet Project as a white paper for others to reference, and use in the further study of the network attack.

2.4: Expected Outcome

At the end of this project there are two expected outcomes that should be available. The first is the outcome from the actual project and the second is the personal or learning outcome.

The project outcome should have a number of questions answered. Such as, have any attacks happened, what kind of attacks have happened, what caused them and how could they have been prevented if at all.

From a personal point of view I would hope that I will better understand how various packers are made up as well as how they enter a network through the gateway or access point, or even if they are already on the network how do they interact with network users.

2.4.1: Practical Outcome

The expected outcome of this project is that all incoming traffic is not recorded, and that it will be filtered to find only certain types of traffic. That recorded information will then be analysed to see what kind of traffic is coming in and to find out what aspects of it may be of use to generate new rules with the purpose of preventing further attacks, and classifying other kinds of data traffic.

When analysing this data, we will see the common trends and threads that run through an IP packet, where it has been, where it is going, what it wants, etc.

I will also be looking at how security benefits a network and system setup, and how to go about doing this. What are the possible solutions to prevention as well as understanding?

2.4.2: Learning Outcome

While the idea of the SNORT and other IDS applications is to record and prevent attacks from happening, the idea of this project and in fact the based around 'Honeynet Project', it is to understand how and why these attacks happen.

When answering this I hope to understand more about how networks work and how they propagate information as well as how attacks happen and how they work. I also intend on understanding more

about data and how to identify different types of traffic with focus being put on malicious data attacks.

This project is highly based on network and system security which is my intended future career area.

The overall aim from this project is to understand what kind of attacks occur and how those attacks are done, where they come from and what their common aspects are. From this I also expect to understand the make-up of malicious data and understand how they travel.

If the outcome of this project is of a sufficient standard, I will consider it for submission to the Honey Net Project, for future reference by others.

Chapter 3: Literary review

3.1: Chapter Introduction

In this chapter we will look at some of the different types of attacks that can happen as well as some of the different types of systems available. We will look into what is recorded in a data packet and what areas that we could look at to determine the kind of packet that it is. Finally we will look at the processing of the information and what kind of processes might yield some good results.

3.2: Reasons for Security detection

With the ever changing world of telecommunications there is an ever increasing need for secure transport of data over a public infrastructure (the Internet).

In recent news it is coming to light that the number of hackers in the world is not only increasing but also making a shift in global location. *CNN* has recently reported on a mass injection of the hacking community based in and around China, they also speculated and made claims that this was all instigated by the Chinese government to try to begin to gain access to the knowledge and accounts of companies from 'The West' or the Western world states (Europe & America), and for international commercial and government espionage (Canada Television).

Whether this is true or just speculation it illustrates a point of people becoming more and more aware of the safety of their data, both on the public network (Internet) and on a supposed "Private Computer". Because of these sorts of things that are happening around us, it is becoming more and more prevalent that we all have some form of protection to prevent undesired access into our data, from Firewalls on our personal computers to totally isolated systems with little to no outside access, and for people to be able to identify these attacks is becoming of greater value.

To be able to identify these attacks and manipulations companies that supply the protection equipment are starting to not only turn to the hacking community as the house alarm installers turned to ex-thief's and burglars, but also to the standard user. Each time an attempt is made on a modern machine that runs most windows environments or anti-virus software, a report of this attack is returned to the manufacturer for analysis.

Because these companies are running in such a violent environment, they would also run a lot of what this project is about (IDS systems). There are even more and more specialist companies coming to market that specialise in IDS systems only and nothing else. Which like with the anti-virus software will also have reports sent back to the mainframe for analysis, so that they can improve their systems and create new rules and identification algorithms.

3.3: Systems

What systems are on the market or available for protection of data. Initially we need to ask what we want to do. This is because there are as many different types of protection as there are flavours of ice-cream. We have our IDS systems, Honeypots, Antivirus's and Firewalls. Though even these can be broken down into different types of systems, and different levels of security.

The up and coming for of protection is the IDS system or NIDS (Network IDS) which would be more used on applied networks, whereas the Honeypot is still generally used as a research tool, but does still get implemented in some cases.

3.3.1: IDS

An IDS (also known as a *sniffer*⁷) is a system or application that is designed to monitor traffic entering and exiting the network or host location, or even traffic that is flowing around a network (internally). Its soul responsibility is to find or sniff out the signatures and packets that it has listed as interesting or dangerous.

There are two main types of IDS available the Network IDS (NIDS) and the Host IDS (HIDS). They do pretty much as they sound, the NIDS will monitor a network and will usually be placed somewhere near the top of the network.

network intrusion detection systems (NIDS) monitors packets on the network wire and attempts to discover if a hacker/cracker is attempting to break into a system (or cause a denial of service attack). (Graham)

⁷ sniffs out things that it knows need to be found

While the HIDS will be placed around the network somewhere to monitor one specific station⁸ or area, generally something quite high priority such as a Server machine or server subnet.

Host based intrusion detection (HIDS) refers to intrusion detection that takes place on a single host system. Currently, HIDS involves installing an agent on the local host that monitors and reports on the system configuration and application activity. In specific vendor implementations these HIDS agents also allow connectivity to other security systems. (Berge)

Even though the HIDS and NIDS operate differently, they can be made to work together, and generally if you have a HIDS you would have it connected to a NIDS. This would make a *Sensor-Sever* IDS or client-server system. Much like the Client-Server systems that many people use to log into their office computer in the morning, this will work in a similar way.

The sensor (HIDS) will monitor a specific machine on the network, and should anything happen to it then it will relay that information back to the Server (NIDS). Though there can also be multiple NIDS machines acting as sensors on the network monitors different areas (I.e. Accounting vlan sensor).

The system that is being setup in this project is an NIDS system that will sit attached to the entry point router of the network, and monitor all traffic that passes through the router and on the network. This will allow the detection of any traffic to any machine that is connected to the network both wired and wireless.

3.3.2: Honeypots

The Honeypot is a system that is designed to attract an attacker into it and keep it away from the main system. Generally these will not actually be placed in the network they are protecting but will be maintained on what is called a DMZ or *De-militarized Zone*, so that the firewall to the main network can stay intact and can continue to do its job without fear of having an over abundant of attacks hitting it. The Honeypot itself comes in 2 varieties, *Production honeypots (low interaction)* and *Research honeypots (high interaction)*.

⁸ Station – single computer terminal or network item

Production Honeypots: Production honeypots are designed primarily for network security and defence. They have not been designed to collect information on hacking activities. For this reason, they are usually easily deployable and do not interact much. These are installed inside the production network and are usually used by corporations and companies to enhance network security. (Tech-faq)

High Interaction honeypots are specifically designed machines that will attract the attackers to it, while the Low interaction is the same but a cheaper option in that it is a virtual install of the honeypot machine. The main advantage of this is cost, it is cheaper to install and maintain a virtual machine rather than a physical one.

Research Honeypots: Research honeypots, as their name implies, are made specifically for collecting information about attackers and malicious software. They are usually managed by educational institutions or non-profit research organizations and are used to gain more insight on Internet "black hat" operations, strategies and motives. The ultimate purpose is to identify threats and find ways of dealing with them more effectively. These are difficult to manage and deploy but they are able to gather a lot of information. This is why they are used primarily by government organizations, the military and research organizations that have the resources to manage and deploy them. (Tech-faq)

There is also the security advantage to the virtual one. If something was to happen to the honeypot and it went down or was erased, or even if you want to update your hardware on the machine that it is being held on, you simply re-load the image or backup that was made of it when it was running.

3.3.3: Passive ‘v’ Reactive

In both the Honeypot and IDS systems there are again two main types, *Passive* and *Reactive* systems.

A passive IDS simply detects and alerts. When suspicious or malicious traffic is detected an alert is generated and sent to the administrator or user and it is up to them to take action to block the activity or respond in some way. (Tony Bradley C. M.)

The reactive system will, when detection of something dangerous will enact a set of protocols to counter the threat. Currently on the market these would generally be designed as NIDS systems rather than host based, and as such will be made by some of the larger network equipment manufacturers such as *Cisco* with their *IronKey* system.

A reactive IDS will not only detect suspicious or malicious traffic and alert the administrator, but will take pre-defined proactive actions to respond to the threat. Typically this means blocking any further network traffic from the source IP address or user. (Tony Bradley C. M.)

Along with the passive and reactive systems there is also the combined option of the IDPS or *Intrusion Detection and Prevention System*. Which will remain passive to conserve network bandwidth and resources, until it detects something on its watch list. When it does detect something it will then switch to a reactive state to deal with the problem, and once it has been dealt with it will change back to passive again.

The term IDPS is commonly used where this can happen automatically or at the command of an operator; systems that both "detect" (alert) and/or "prevent." (Wikipedia)

3.4: Types of Attacks

The reason we use an IDS or Honeypot in the first place it to detect any attacks of dangerous activity that is entering or trying to enter the network behind the IDS. If an IDS tool is to be in any way useful they not only need to be able to detect the usual things such as virus and hacking attacks (Signature based) but also the ones that might be less obvious (Anomaly based). With the paid systems, such as KFSensor, you would expect this to be a given since they make you pay for the system, and this in fact is the case, KFSensor does have the ability to check all of the current types of attacks and signals that might pass through a network. However due to its licensing and such all the code relating to the application is secure and protected, which mean that you have little flexibility when it comes to generating new rules and classifications, this can generally only be done by the makers. SNORT on the other hand, because it is fully open source (and most of the paid version is based on it) the user has full capability to create and generate their own rules to do what they want then to do.

The basic rule set that comes with SNORT will cover the user against most of the different types of traffic that might occur on a network. Some of the rules that are included are can cover the following. (Google defined)

Backdoor Attack	<i>A method of bypassing normal authentication, securing remote access to a computer, obtaining access to plaintext, and so on, while attempting to remain undetected.</i>
Botnet's	<i>Software agents, or robots, that run autonomously and automatically.</i>
DOS attack	<i>A denial-of-service attack or distributed denial-of-service attack is an attempt to make a computer resource unavailable to its intended users.</i>
DNS attack	<i>Domain Name System attack, also called DNS Spoofing or DNS cache poisoning, is aiming to redirect users to potentially malicious web servers by changing the records used to convert domain names to numerical addresses, which is used as another way for online fraudsters to install aggressive.</i>
NetBIOS	<i>NetBIOS is an acronym for Network Basic Input/Output System. It provides services related to the session layer of the OSI model allowing applications on separate computers to communicate over a local area network. As strictly an API, NetBIOS is not a networking protocol.</i>
MySQL	<i>SQL injection is a code injection technique that exploits a security vulnerability occurring in the database layer of an application.</i>
PoP attack	<i>A simple tackle attack</i>
Port Scan	<i>Series of messages sent by someone attempting to break into a computer to learn which computer network services, each associated with a "well-known" port number, the computer provides.</i>
Spyware / Virus's	<i>Spyware is a type of malware that is installed on computers and collects little bits of information at a time about users without their knowledge. The</i>

presence of spyware is typically hidden from the user, and can be difficult to detect.

Web Attack *Use of the internet to attack a visiting computer or person, fake web pages, request for computer information, automatic downloads.*

ICMP Attack *Attacker sends forged ICMP echo packets to vulnerable networks' broadcast addresses. All the systems on those networks send ICMP echo replies to the victim, consuming the target system's available bandwidth and creating a DoS to legitimate traffic.*

However in this project we will be only concerned with a few of the many that are available, in particular DOS attacks ICMP attacks and Port Scanning, since these would be the most common types of attack currently being done.

One reason why SNORT is a good choice of IDS is because it allows the user to completely customize the rules to detect anything that they might want to, not just the malicious stuff that might happen but also to see if someone or something is trying to access a certain website or service that you are hoping is forbidden by the network or hardware on that network. Because of this the standard rule set is quite extensive and can allow the system to detect nearly any kind of event that the network may encounter or experience, even including a *Test* rule set as well as an *In Development* set that can be used if needed.

3.4.5: Denial of Service (DoS) attacks:

Denial of Service attacks are when a user is denied access or denied full access to a service on a network that they either need or want to access.

A denial of service (DoS) attack is an incident in which a user or organization is deprived of the services of a resource they would normally expect to have. In a distributed denial-of-service, large numbers of compromised systems (sometimes called a botnet) attack a single target. (TechTarget)

A Dos attack isn't one that can be classed as dangerous to systems like a virus or hack would be, but it is still something that can be quite violent to a system. The idea of this is to cause disruption and possible loss of data during genuine transmissions.

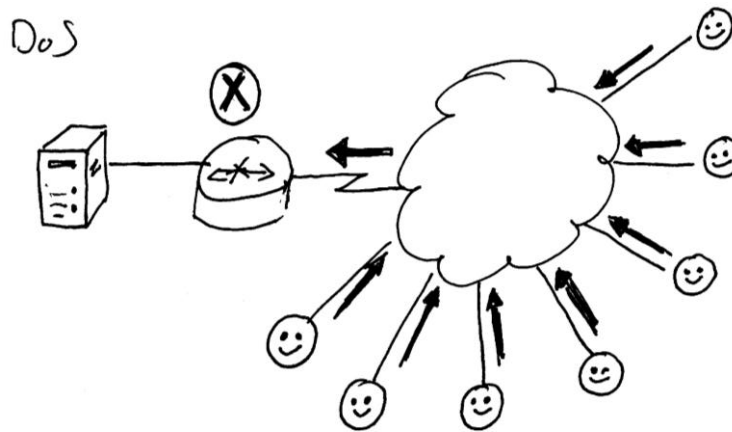


Figure 9: DOS attack (MacTiernan, 2010/2011)

In recent years one such attack was taken on the *MasterCard* website but the supporters of the Wikileaks founder *Julian Assange*. After Julian was arrested for hacking into the United States CIA information databases (and other less than convincing allegations that were placed against him) MasterCard decided that they didn't want anything to do with such a high profile case, and as such cut off all credit card processing for donations and payments to and from the Wikileaks website. The supporters were not happy about this and decided to take the matter into their own hands, they then proceeded to bombard the MasterCard site with information and open connection attacks causing the Denial of Service attack which rendered the entire site useless to anyone trying to do business across the MasterCard network (website, atm⁹, shop credit card machines) and eventually forced it to shut down while work was done to try to prevent this. (Withney)

The DoS attack is accomplished by overloading the connection or system with too much information all at once and causing it to either slow to a crawl or to crash completely. The overall objective is to shut down the use of the service that is being attacked, however to shut down the

⁹ Automatic Transaction Machine (aka, the hole in the wall)

service does not necessarily mean to crash it, it just means to render it unusable to anyone that might want or need it. This form of attack is quite common across the internet and most people will have experienced it at one time or another. Sometimes it can even be done by accident by a router trying to send the same information a number of times and not getting a response from the destination. From an attack point of view, this can be as simple as one person bombarding the destination with packet after packet through the use of ICMP flooding or bombs (NUKE, E-Bomb, ...) which will cause the destination machine to maintain an open connection and receivership of data and information causing either the hardware to shut down or to overload the physical connections causing a major bottleneck in a network drastically reducing the flow of data through the open connection, generally only allowing the flow to go in one direction. (Sirkanth)

Some ways this is done are -

- ICMP Flooding – where massive amounts of ICMP echoes (pings) are sent to the destination overloading the destination machine.
- NUKER (virus attack) – malicious attack designed to disrupt data transition through a software attack from within the actual machine.
- Bomb (e-bomb) – an overload of the open connection with prevention of closing through the use of continuous sending of large amounts of data from the internet selected at random. Usually just garbage and spam from the internet.
- Boomerang attack – in a boomerang attack, the attacker spoofs the IP address of the intended victim. In this kind of attack the attacker sends packets that elicit some kind of reply. If the attacker sends enough packets initially, a large volume of replies is returned to the victim, causing it to hang or crash. (Bidgoli, 2004)
- Reflection attack – When an honest protocol participant sends a message to its authentication party, the intruder eavesdrops or intercepts the message and sends a modified version back to the originator. (Computational science and its applications, May 8-11, 2006)
- Degradation of service – aka Pulsing Zombie attack. Unlike a regular Zombie attack that paralyzes a system with a steady stream of attacks, the pulsing zombie attacks with irregular small bursts of attack traffic from multiple sources on a single target over an extended period of time. These are more difficult to detect and trace since they are slow and gradual and do not appear as malicious. (Atlantic, 2007)

- Blind denial of service – this can be more of an Spoofing attack where the attacker receives the information from the destination and then clones the IP and relevant details into its own IP Packet. This information is then sent off again as a masked packet to a new host or back to the same one again to cause the routing problem and forced shutdown of the system. (Web of Trust)

In one form or another there will rarely be just one of these types of attack being enacted during an attack. Unless it is just a simple system overload due to number of opened connection and data transition, there will usually be a number of them combined together.

Port scanning + Brute force attack to gain entrance, + Zombie machine = e-bomb to other systems.

3.4.6: Port Scanning

This is a fairly unintelligent form of hacking but probably the most popular type of attack in current times because of the use of ever increasingly intelligent firewalls and other preventive measures. What this will do is go to a network (IP Address) and scan all known ports to find an open or low security one, and then try to use this open port to gain access into the system, or use a brute force attack to try to crack any security measures that are in place.

“Port scanning software, in its most basic state, simply sends out a request to connect to the target computer on each port sequentially and makes a note of which ports responded or seem open to more in-depth probing.” (Tony Bradley C.I.)

When a network is found they will start to scan the ports. First they start with the “well known” ports such as FTP, HTTP, and SMTP to try to gain access. Failing that it will move on to some of the more unknown ports and finally to the not used ports in hopes that someone has left one of them open.

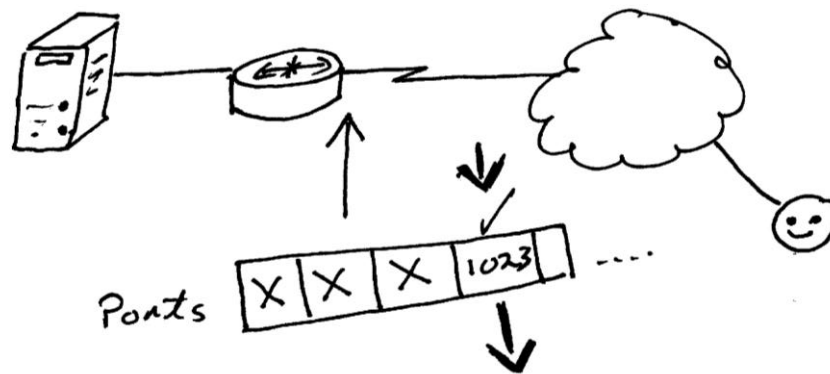


Figure 10: Port scanning (MacTiernan, 2010/2011)

Essentially with the TCP/IP protocol there are a total of 65,536 ports that can be attacked, and since only about 50 of them are actually used for anything that means that there is still over 65,000 ports that may not be monitored. Although the non-used ports would (in theory) be closed and not available for data transition in or out.

3.4.7: Tampering and Spoofing

Tampering is when a packet leaves a network it has a number of fields in its head that are set to tell the receiving router what it is and how to handle it. This information contains such things as Check Sum value, Sequence value, packet size and Destination/Source IP address. If someone were able to intercept the packet they might be able to change this information to either conceal something by changing a check sum value or sequence value, redirecting it to another destination or even sending a new packet and masking it as another one from a different source. However this is not fool proof and though it can be hard to determine if this has happened or not, to do this we need to look at a number of different areas, such as the check sum and sequence value to determine if what we are receiving is what is actually stated in the header.

Spoofing is another common form of attack recently, what it does is pretends to be something that it's not. An attacker will sit listening for some data that it finds interesting, when it does find

something it will intercept that packet and simulate it. It then returns to the source or keeps going to the original destination, but masked as the original packet.

3.4.8: ICMP Attack (aka Smurfing)

The ICMP attack is a cross between *Spoofing* and *DOS* attack, its primary goal is to bring down a machine or network with an overloading of ICMP packets. The ICMP packet is a request and acknowledgment or Ping and Echo reply. The way this works is that the attacker will send out a ping request but will have a faked IP address, usually the address of the machine that it wants to attack.

“The attack is to construct a packet with the source address forged to be that of the victim, and send it to a number of smurf amplifiers. The machines there will each respond (if alive) by sending a packet to the target and this can swamp the target with more packets than it can cope with” (LAMPSON)

The *Smurfing* attack is very much a hackers attack (i.e. the hacking community) hence the comical name, but its acts are not funny. Usually the Smurf attack is done to take over a server, usually something like an IRC¹⁰ so that they can take control of the multiple servers that make an IRC system as we can see in Figure 11.

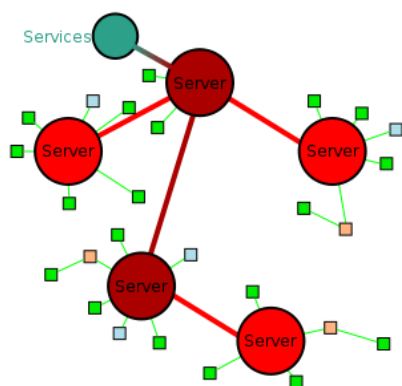


Figure 11: IRC chat network (Wikimedia)

¹⁰ Internet relay chat

3.5: Attackers

When someone does get attacked, majority of the time they will not know what it is that is happening, or if anything is even happening at all. Just because you can't see it, doesn't mean it isn't happening.

One of the biggest problems with the attacker is that people don't know what to do when they do get attacked, like most things people will either ignore it or freeze up themselves.

A few different types of malicious activities performed by network attackers and hackers are summarized here:

- *Illegally using user accounts and privileges.*
- *Stealing hardware designs.*
- *Stealing software.*
- *Running code to damage systems.*
- *Running code to damage and corrupt data.*
- *Modifying stored data.*

(TopBits.com)

3.4.1: How they happen

There are a number of ways that an attack can happen on a system. Generally thought it comes down to 2 forms or attack, Internal and Remote.

An Internal Attack would be one where someone has managed to download some kind of *Bot application*¹¹ that is taking over the system resources and causing general problems for the users and system. The Remote attack on the other hand is a bit more extensive than this.

In a remote attack, the attacker must first find a suitable system that they can attack, and then they need to gain access to the system to plant the code or do what it is that they want to do. This is done through some of the for mentioned attack types. The remote attack is also either human interaction where a person is actually typing on a keyboard to do the actions themselves, or it is all automated with a 'bot' like with the internal attacks.

¹¹ Automated program

3.4.2: What to look for when being attacked

Depending on the type of attack that is being done at that moment, the signs could be as visual as something popping up on your screen or you losing control of your accessories (mouse, keyboard), or more likely your system slowing to a crawl or even nothing at all that you can tell.

The most likely thing that will be noticed is a slowing of the network and maybe system performance. This can usually be seen on any home computer that is connected to the internet, because people would normally download things they shouldn't or download things badly that will have a joyrider on them getting a lift to a new computer system to disrupt.

Other things that may show signs of an attack are system function changing state, like a CD draw opening for no reason or computer monitor turning off. It may even be a case of an application starting and running in the background that you might not necessarily notice. But all these things do have signs if you are looking for them. The thing is to know how the machine should act, then when something happens and the state changes you will notice the difference, and be able to look deeper into the problem.

With networks being attacked, things will again slow down, but network attacks (depending on the attack) can also cause packets and connections to completely drop for no reason. While this is an annoyance it can also be system critical and cause some very dangerous problems on a network.

3.4.3: What to do when being attacked

Taken from an article titled "top 10 things to look for when under a DDOS attack" (orbixhost)

1. *Don't Panic*: This is fairly explanatory, try to keep your head about you and stay calm. If you start to panic you don't know what you might do.
2. *Create a backup of all your stored contacts*: Most companies will hold all their contact information (emails, phone number, IRC chat ...) on one single storage device, usually the email server or main server. If this gets attacked and damaged, there is a very high likelihood that this information will be lost. If it is kept off site on in a different physical location to the usual server then you have a good solid backup and are able to contact people should you need to.

3. *Setup a “war room”*: This is essentially a crisis centre that will have everyone of note contained in the one space ready to do what is needed. But be ready for the long haul.
4. *Get at least one IT member to the colocation¹² ASAP*: This will allow someone to be able to console into the system directly since it will be most likely that network connections will have gone down. The advantage here is that you will be able to have a first-hand view of what is going on.
5. *Old Equipment*: This can be of great advantage using older equipment, which will allow you to re-establish a connection to the network and can allow you to re-connect a machine to it to do some sniffing and try to figure out what has happened.
6. *Understanding the nature of the attack*: Knowing what the attack is, is a key thing. You need to be able to find out not only where they are on your system but what they are looking for. This can help to relax some paranoia when it comes to who is attacking since some people in larger companies might believe that it could be a competitor.
7. *Document, Document, Document*: Document everything that happened and everything that you do. That way should something happen (i.e. Legal) then you have records of everything even after they attacker erases their tracks, if they do.
8. *Call your ISP*: Even though your ISP is another company and will have their own process to follow, they can be of great help in both finding out what the problem is and stopping it. They may also be able to help prevent anything like it in the future with various service that they might provide.
9. *Inform clients*: informing your clients is a vital action. Setting up a ‘We are down’ web service or even just calling them to let them know that something has happened. This will allow you some time with possible client transactions and will make sure that they know something has gone wrong somewhere.
10. *Review the attack*: After any form of down time with a system there should be a full review done to figure out what happened, how it happened and what could be done in the future. This report should then be filed safely for future reference should something like it happen again.

¹² Colocation (or collocation or co-location) is the act of placing multiple (sometimes related) entities within a single location.

3.6: What is produced in the records

Depending on the IDS that is implemented, there will be different types of records that will be recorded. With SNORT there will be 3 main records produced,

- Alert
- Port scan
- Log

The difference is that an Alert will pretty much just record every event, while with the Log and Port scan will only record specific events to a designated file and location. However even though SNORT will record these events automatically there is also an option to add an additional writing location to the rule string using the “log” rule option.

The records themselves will hold a number of crucial bits of information from the port used to the timestamp of the attack. However it is not a simple case of it saying ‘I am a DoS attack’ since this is something that we are meant to be able to find out from our understanding of the different attacks.

3.6.1: How do we distinguish an attack

So what is an attack? An attack is an act of malicious activity on a Private Network or device that is designed to cause harm and/or influence system design and structure.

Operations to disrupt, deny, degrade, or destroy information resident in computers and computer networks, or the computers and networks themselves. (The free Dictionary)

In terms of this project an attack is anything that is out of the normal bounds of daily network traffic. So because of this, what do we look for? What we are essentially looking for are outliers from within the data. Data that is not following the regular trend or out of the ‘Norm’.

In most cases you will never know that you have been attacked, since there are usually very few signs because the attacker wants to remain in the systems as long as possible without being detected. Usually all that will be seen is a slowing of operations, but in the more extreme cases there will be a partial or total loss of data and services.

In the past month (April 2011) there was a very major attack done on the Sony PlayStation online network (PSN), where someone managed to hack into their system and compromise everything from game accounts to user credit card details. There is only speculation as to how this was done but it is thought that the culprit re-programmed his Sony PlayStation 3 (which they allow you to do) and used this to mail a Trojan into the system which opened up a backdoor from him. Or is possible that he managed to use an SQL injection attack to gain access into their database.

How was the PlayStation Network hacked, though? Ironically, for security reasons, and because Sony is historically very tight-lipped on such matters, we will probably never know the exact attack vector — but we can certainly make some well-educated guesses about how the PlayStation Network was hacked. First, given its proximity to Anonymous's recent attacks, it's likely that the database breach is somehow related. It's safe to assume that Anonymous could have learned about a weakness in the PSN's security mechanisms, and then passed that data on to another group of hackers — and from there, if the hole was big enough, the attackers might have been able to simply step right in with an SQL injection attack. (Allinfosec)

There was a lot of data stolen and compromised in this attack and once Sony found about this penetration they voluntarily shut the system down to prevent any further data loss or corruption. This is just one of the most recent attacks that have been made on high profile companies, however because of the way the company has dealt with the problem and its customers it has hit headline news around the world.

3.6.2: What areas to look for

In general, depending what it is that we are looking for (i.e. a specific type of attack) will depend on what to look for in the data packet. However there are standard things that we can look for that will indicate certain things.

Ports are used by every data stream that enters a network, but there are specific ports for specific types of data. Port '80' is TCP/UDP port used for HTTP internet traffic, while '25565' is used by MySQL for network connections and remote use. There are also a large number of un-defined ports available but that are still used by various applications such as '3724' that is used by a company called Blizzard for their MMORPG game called *World of Warcraft* or '17500' which is used by

Dropbox to synchronize file contents between the host and server and host's. This is not something that is of massive concern, however a lot of people who use these un-defined ports will open them up on the router to allow for a better connection to the source, this will then leave the port wide open to any form of attack unless closely monitored, which most wouldn't be.

The **IP Address** is also another place we can look for certain attacks, though with this we need to know the IP address of the attacker. Otherwise all it is used for is future definitions for classification of possible attacks. The reason we say possible attacks even if an attack has come from there in the past is that this will most likely not be the attacker but will unfortunately be someone who has been compromised and as such their location and IP address is being used to propagate an attack further. This is a form of *spoofing* attack.

If we can see into the data we may be able to verify the **Checksum** number against what is actually in the packet. This attack will intercept the data in transit and replace what is in it with something else. However doing this will change what the checksum will be, but since you are unable to alter the checksum all they can do is hope it goes unnoticed. The problem here is that we can rarely see into a packet until it's too late. This form of attack is known as a *Man in the middle* attack.

Another area we can look at is the **time stamp** or time interval between packets being received and then received again. If we receive too many too fast this could be an indication of a *DOS* or *stack overflow* attack, where the attacked is trying to force a system response by sending too much too fast, or with the stack overflow or buffer overflow they might be trying to fill the buffers up fast enough that the machine will grant entry regardless of user verification. The password buffer overflow use to be a problem with the Cisco 7xx series routeing equipment. (Cisco systems)

Looking at the packets and traffic that enters the network may show various different parts that could be of use, however these will change each time so looking at them with the eye may be difficult. One thing to do is test with the IDS tool with various settings, and see what happens, but this may not be the most productive option.

3.7: Processing results

Once we have the results from our IDS we need to try to find out what we have, do to this we can do a process of what is called Data Mining. This will show us the information that we need as well as be able to graph is to make it that little bit easier to understand the results.

3.7.1: What do we use

In data mining there are a number of ways to do it. Traditionally it would be to read through the data and apply a number of different mathematical equations to it to produce a series of graphs and figures that will help us to find out what we have and locate any hidden trends and patterns within the data. Thankfully though with computers becoming more powerful these processes can now be done by various applications making the process much faster and much more accurate. Some of the available applications are Rapid Miner, ODM (Oracle Data Miner), SPSS (IBM Data Miner) and SAS Analytical which has recently received the top data mining tool award.

Straight away though we can say that SAS Analytical and SPSS will not be used due to the high cost of each and the form of implementation and use that they have, being a high training curve with specialist certifications available for each. That leaves ODM and Rapid miner which are both free to use under GNU-GLP¹³, of which Rapid miner would be the best option because with ODM you would really have to run the Oracle Database with it since they are designed to work optimally together, whereas rapid miner can work with nearly any database and we will be using MySQL.

3.7.2: Data mining techniques

Depending on the kind of data that you have will depend on what you can do. The steps involved in a process are *Pre-processing*, *Data Mining* and *Result verification*.

Pre-processing is when we have to apply certain things to the dataset to change it to a more usable format, or to filter out or add in certain elements. For example we might want to bin all the dates into weekly dates rather than daily or by the second, we might also want to add an additional field

¹³ GNU-GLP = GNU General Public License

to use as a prediction field. We can also read in specific data too which we will most likely have to since we are dealing with databases and not static text files or ‘.xls’¹⁴ files which will need to be queried as well as just read in.

With the **Data mining** we then need to take our pre-processed data and apply a variety of models and classification algorithms, if done right should display some interesting (or not interesting) results as to the nature of the data and what it contains both visual and hidden.

After mining the data we need to **Verify** it, this can be as easy as letting the mining process do it for you (which should be done anyway) which will produce a confidence vector as to how confident it is that it is correct. Or we can use the infinite universe of information called the internet, to find information about specific things that we might find in the data such as the use for specific ports and what does actually use them, or DNS/IP look-up tables to find out who has been contacting the network.

3.7.3: Designs of interest

With some Honeypots, IDS tools and IPS¹⁵ systems there is a lot of data that can enter, and not all of it is classified. So to classify unknown data we would most likely need to implement a form of clustering algorithm, one of which would be easiest to read and understand by most people would be the Decision Tree classification algorithm or a bit more complex but still as easy to understand the K-NN algorithm. If however we are dealing with totally unknown data then we can apply a Clustering algorithm to it, which will cluster the data into groups of data that have the same information or similar information that will expose some trending pattern that would allow us to discover what it is, whether it is hostile or not and what kind of hostile data it may be.

But for this project we will mostly be concerned with finding the outliers in our data sets, since we are expecting the data to have already been classified by the IDS. However the classified data will not really tell us what it is and that is the tricky part, trying to identify what type of data we are receiving whether it is hostile or friendly and trying to find out what kind of hostile traffic it is (ICMP attack, DOS attack ...), but the advantage is that we know the “odd-one-out” so now we just need to examine the packets to try to find certain key areas such as ports or IP addresses. Once

¹⁴ Excel spread sheet file format

¹⁵ IPS = Intrusion Prevention System (an advanced form of IDS)

we have these we could then go online (Internet) and search for these key areas to see what comes back.

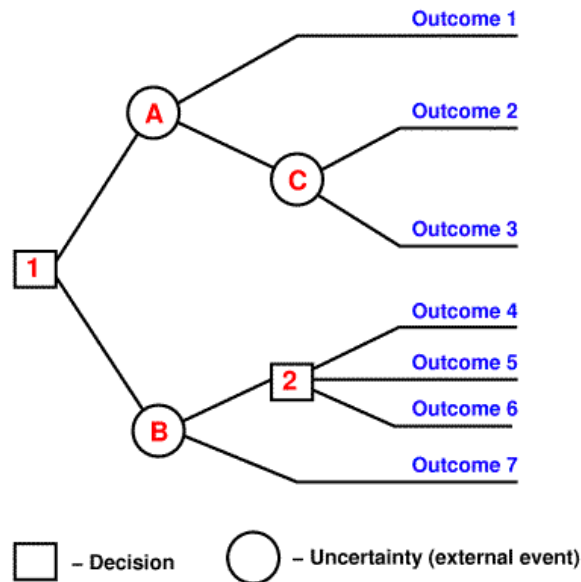


Figure 12: Decision Tree (Time-Management-Guide.com)

A decision making tree is essentially a diagram that represents, in a specially organized way, the decisions, the main external or other events that introduce uncertainty, as well as possible outcomes of all those decisions and events. Figure 12 (Time-Management-Guide.com)

In particular in this project we will mostly be concerned with finding the *outliers* since the data that is collected should have been filtered already.

"Outlier" is a scientific term to describe things or phenomena that lie outside normal experience. In the summer, in Paris, we expect most days to be somewhere between warm and very hot. But imagine if you had a day in the middle of August where the temperature fell below freezing. That day would be outlier. (Gladwell)

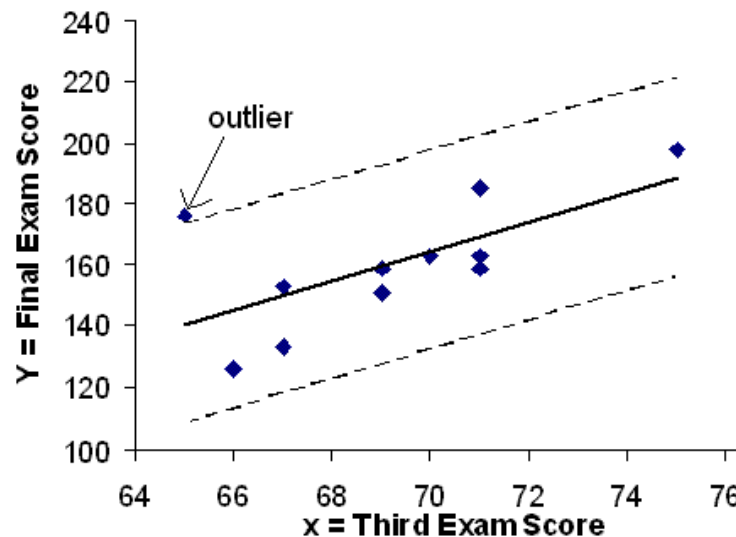


Figure 13: Outlier example (Bloom)

The reason we are doing this is because there will be a lot of traffic on the network because of so many machines on it, and we don't want to sift through over a million entries only to find out that it's all rubbish or safe traffic. Having SNORT filter and identify it beforehand will save a lot of processing time and computer space. Once we apply the process to find the outliers we should be given back a number of entries that are of interest because they stand out from the crowd. We can then look at this data to try to understand what it is and what it was trying to do.

3.7.4: Data mining of network attacks

It is said that there is a new virus of attack method developed every day, though most of these will be adapted from a previous version. However there is a significant development at least once a year since 1998 (Information Please® Database), and because of this there is a now a critical need for real-time counter measure development.

One of the most common forms of detection is the use of *State-Transition Analysis* which describes goals of patterns that best describe an attack, when one of these attacks is triggered then the state (packet) will be considered as an attack or intruder, which will initiate the preventative measures. However this is still a form of predefined IDS or IPS rather than a real-time system.

NetStat (developed at USCB) performs real-time network-based intrusion detection by extending the state transition analysis technique (first introduced in STAT) to the network environment. The system works on the complex networks composed of several sub-networks. Using state transition diagrams to represent network attacks entails a number of advantages, including the ability to automatically determine the data to be collected to support intrusion analysis. This enables a lightweight and scalable implementation of the network probe. (Daniel Barbará, 2002)

One of the problems with trying to do real time data mining is trying to have a stable data set to work from. In this case we would have to probably develop a new data set every X period of time to use. However this may or may not include the latest developments in attack forms and as such may prove to be less than accurate with its models.

Jam (developed at Columbia University) uses data mining techniques to discover patterns of intrusion. It then applies a meta-learning class-sniffer to learn the signature of attack. The association rules algorithm data mines relationships between fields in the audit trail records, and the frequent episodes algorithm models sequential patterns of audit events. Features are then extracted from both algorithms and then used to compute models of intrusion behaviour. The classifiers build the signature of attacks. So essentially, data mining in JAM builds a misuse detection model. (Daniel Barbará, 2002)

Another option is try to read the data as it enters the network and keep adding it to the already existing model information. This however is a lot harder to implement and as such is not something that would generally be used since it would be a case that some complex algorithms as well as a super-fast computer would need to be used.

Another form of detection is through the use of *Statistical Analysis Methods*. Similar to the previous there is a group of information that is known about attackers in general and about specific attacks. This information is then used to generate a series of statistics about attacks so that anything that enters the network can either be proven to be an attack or assumed to be an attack (or not).

Another form of system would be what is known as an *expert system*, where the data is known about, but not about the specific attack at that time. What happens is it will monitor the traffic on a network and compare it against its statistical knowledge base.

IDES (Intrusion Detection Expert System) employ statistical anomaly detection. In particular, it used audit data to categorize user activity and detect deviations from normal user behaviour. Information extracted from audit data includes user login, logout, program execution, directory modification, file access, system calls, and session location change and network activity. The NIDES system extends the IDES by integrating response logic with the results produced by the anomaly detection system.

(Daniel Barbará, 2002)

These forms of IDS and IPs systems are among the more advanced and as such are also among the few that are in operation within companies, be it high profile companies though.

Chapter 4: System Analysis and Design

4.1: Introduction

In this chapter we will look at the different techniques and tools used in Honeypot systems. We will also analyse these different areas and compare them against each other to try to find the best possible option for the project.

4.2: Evaluation

This project will be to analyse inbound data from the public Internet, which has entered into a private network, but should only reach as far as the Honeypot that has been setup. With the use of a Honeypot system, incoming traffic is gathered and the accompanying capture utility will record the information to the database. This data will then be analysed with the use of a data mining tool which will produce a series of trends and patterns from the attacker's information.

4.2.1: Operating systems

With the hundreds of millions of computers out there in the world each one needs an Operating System (OS), for this there are 3 main ones that people can choose from, Windows; Linux based and Mac OS.

Windows: with various versions from 3.1 to the latest versions, Windows 7. Through all the different versions there have generally been the Home and Professional versions. For a Honeypot, most people would recommend one of the Professional or Business versions since they have the most versatile and secure options from the range, giving the users the ability to install certain packages, tools and utilities, or not to install. There are also the Server versions which would be the ideal choice although they usually cost a lot more than most people can afford for a project such as this, unless they have a versions already. Server advantages are that you can create much more secure areas for the different members and applications to go to, and also make the system more enticing for an attacker since it's a server system that can run additional services (mail server, file server ...).

Linux: With Linux we need to look at what version is available and what we need it for, since there are hundreds of distributions available. A distribution is like Windows being made by Microsoft, it is an ownership of a release, but each distribution can have various versions. Linux offers a great modularity approach to system design by having each application apply to a different module of the OS. Linux will also prevent every user from being the Root or super user (su); each user will have a certain level of privileges available to them depending on their category. Because of this level format we are able to section off areas of the systems to prevent attackers from accessing. One Great advantage with Linux is that there is a version available called Honeywall which is in itself a Honeypot system. A problem with using Linux however is, that the majority of the interface will have to be done through the bash command line. This make it's more time consuming to use but will also mean that any additional systems and hardware that need to be installed will also pose a problem doing so.

Mac OS: it is based on the UNIX core which is also the basis for the Linux and to a certain extent Windows, although the Mac OS is a lot more developed. Mac OS like Linux has a modular approach to the OS and restricts users to the areas they are allowed, with no super user. This approach to OS programming makes it a lot more secure and less prone to being attacked since gaining access to the core of the system is a completely restricted. One of the main downfalls with OS-X is that it can only be installed on Mac machines due to a core chip located on the motherboard of the machine, this prevents a lot of machines from being able to use the OS and as such only Mac machines will have the Mac OS system.

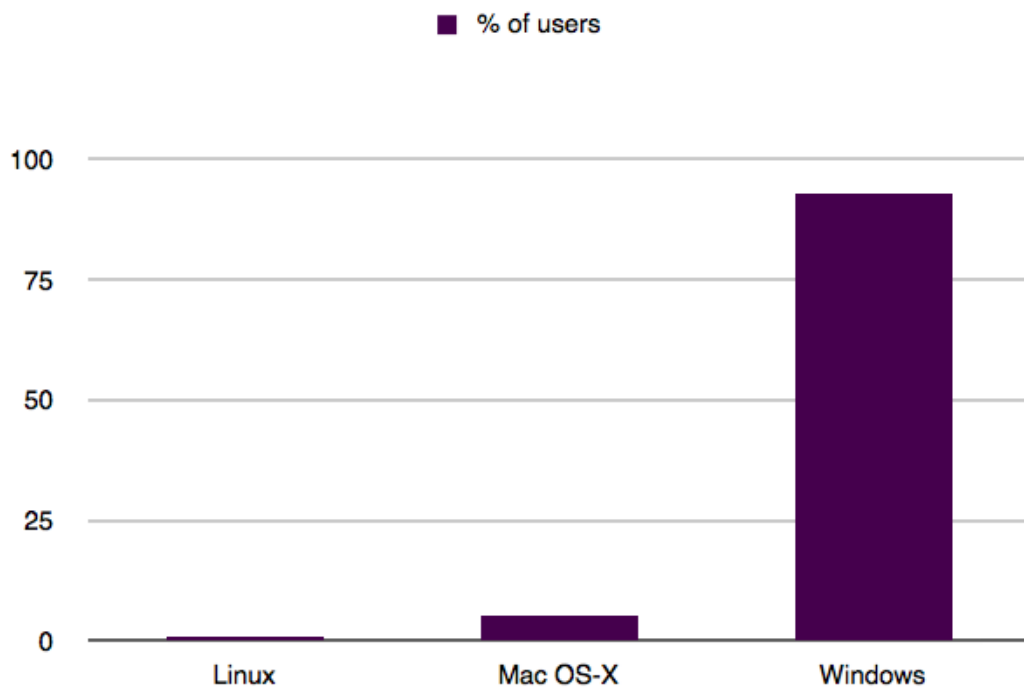


Figure 14: OS percentages (Michael DeAgonia)

What does all this mean for the Honeypot system then? Well what we have to first do is choose what kind system we will be running the Honeypot on, this will then help us in choosing the actual IDS that we are going to use for the Honeypot. So our choices so far are Windows, Linux and Mac OS. In relation to being attacked, and not which is the more accessible, the most likely to be attacked would be any one of the Windows systems. This is purely down to the fact of numbers of available systems with windows taking up over 90% of all systems worldwide. (See Figure 14) taking this into account we look at the available Windows systems that are available, with the current technology in computers this will immediately rule out the use of anything older than Windows 95 or even Windows 2000. Ideally we will want to use a Professional or Business version, and ideally something that is more recent since the core of the system has been changed. From personal usage I will be going with one of the Windows Servers versions, either 2003 or 2008 depending which one I can get. Both of them are relatively the same with a few extra options available in the newer version.

4.2.2: Attacker

The attacker is a can be anything from an alien source that tries to illegally access a network or system that does not have the authorisation to legitimately do so. These attackers are generally automated bots¹⁶ that search the internet for an open connection that connects to a private network, but can on occasion (less now than in computer history) be a human hacker¹⁷. These bots try to access networks with hopes to gain access to something that could be of interest, or of use to the bots owner. However some bots are there purely for the purpose of causing destruction of data and hardware and disturbance to systems.

Examples of types of malicious and intrusion software:

1. Rabbits - A Rabbit virus (aka, Fork Bomb) will enter into a system and cause a replication or duplication fault. What this does is opens a file, usually something small like the cmd.exe or similar small system file, and will keep doing so, so that it consumes all the system resources. When doing this the system will reach a point that will cause it to shut down, this can be either memory overload or more sever can be over heating the CPU, motherboard or hard drive. A memory overload will generally cause a shutdown or restart but an overheat, can cause the CPU to melt, Blue Screen of Death terminal system fault or even critical system failure. Using a Rabbit can also (depending on the system being attacked) cause a buffer overload, which will reset system access and allow the attacker into the core or just access past the password's (Cisco routing equipment suffer from this problem)
2. Trojan - Trojan comes from the Historical reference 'Wooden horse of Troy'. This is an application that is masquerading as something else. Usually something like a spam-bot (low risk) will use this technique, or more sever you will get criminals running them to try to get your information for their own means (e.g. Fake antivirus utilities) which they can then either use or sell.
3. Spyware – this can be placed on your computer to spy on what you are doing (Inc. key loggers) this information can then be sent back to its owner for them to use or more serious use against you.

¹⁶ Bot – Automatic program that can run a series of commands (e.g., batch file)

¹⁷ Hacker – someone who invades a system without authorisation

4. Macro – a Macro is a program that is designed to do automated tasks. You pre-program it with simple actions that will then all be done together once the macro is used (e.g. Batch file)
5. Destructive Virus – this can be any form of what would be called a ‘normal virus’. It will enter into the system and simply attack it, causing nothing more than destruction, of any or all systems files and files located on the HDD¹⁸ or connected computers.

4.2.3: IDS / Capture

The IDS is used on many high profile networks, but an IDS on its own does not make up a Honeypot, a Honeypot is made up of a number of different tools, applications and utilities such as the IDS and Capture utility.

Once the attacker has found the open connection they will then proceed to enter into the system. The IDS itself will (depending on the application) open up a port on a firewalled network which will allow access into the Honeypot while keeping the rest of the network safe from harm (remember nothing is ever air-tight). The Use of the IDS is simply to track the attacker’s data and interoperate it. It is left to the Capture utility to do the actual print out and recording of all the attackers information, although some IDS applications have a Capture utility built into them so a separate one is not always needed.

With IDS tools there are two options that we can go for. Either Commercial or Open Source, and within those two options come different options again depending what kind of Honeypot we are looking to setup.

- Standard IDS: This will sit at the front of a network on an OS acting as bait for attackers.
- IDS/Honeypot OS: This will be a Linux based OS that contains all the tools, utilities and bait programs that are needed in a fully functioning Honeypot.
- The Honeynet: option will be a specially designed IDS that will work with multiple installations through a network and link together.
- Search-bot: Search utilities like Google have recently developed systems to try to stop the increasing amount of attacks and malicious activity that happens through the various search

¹⁸ HDD – Hard Disk Drive / Hard drive

engines, where the attacker will re-direct a Google link to another site of their own creating.

For this project I will be sticking with the regular IDS, however depending on some testing I may decide to run the IDS OS alongside the Standard IDS in a virtual environment as a test. I will also run the KF-Sensor Honeypot beside it to act as a backup in case something goes wrong.

4.2.3.1: Open source

These Honeypot's have the great advantage that they are FREE, however like with all free open source applications available they do require a little more work and knowledge to get them working. If you manage to get them working and have the patients to stick with it though, the rewards can be a lot greater, since the open source community love to tweak the software and develop additional packages to suite a broad range of needs or one specific need within that tool to a greater efficiency.

1. SNORT – is the original and definitive IDS application for the Honeypot Project. This is the foundation that all current IDS systems are based on, and as such all IDS applications produce SNORT standard results, and SNORT compatible results. The biggest advantage with SNORT is that it is FREE, but also because it runs on the Open Source GNU License it is highly customizable with all source code being publically available. This customization come in the form of Packages, since SNORT is an open source application anyone is allowed to create a package that will work with it, as long as it conforms to the GNU Public License agreement. (SNORT)
 - a. Advantages – FREE, Highly Customizable, Industry Standard, can work on any OS, very detailed packet information.
 - b. Disadvantage – more difficult to get this working since most are at the command line level. Things may go wrong in setup with the number of available packages, requires more utilities to create the honeypot.
2. HoneyD – HoneyD is more typical of the Open source community in that it is a small application that has mostly a text based interface. This is done for simplicity but also to reduce overhead in the application. This IDS is among some of the more versatile, since it has the ability to be used as a virtual Honeypot or even having the capability to combine

multiple copies together to form a Honeynet. The reason for allowing for a virtual IDS is to allow for internal network security and monitoring. (HoneyD)

- a. Advantages – open source, free, versatile, can be made into a Honeynet easily.
 - b. Disadvantages – knowledge of bash and Linux command line structure is required.
3. Honeywall – this is another Honeynet standard. At its core again is the SNORT application, but with Honeywall you get an entire Operating System. Once installed, as with KF Sensor all required utilities, tools and applications are automatically included. Like with SNORT the Honeywall is an open source OS based on the Fedora Core with a custom shell and added utilities. (The Honeynet Project). This could be a possible option to run as a Honeypot for this project, investigation into running and using this will be looking into during the 2nd stage of the project, it may be run as a secondary Honeypot.
 - a. Advantages – No OS required,
 - b. Disadvantages – some knowledge of Linux needed,
4. Google Hack (GHH) – With Google's growing market share's, they are also growing in what they offer. Gone are the days when Goggle was just a search engine provider, they now offer everything from e-mail to the latest Google TV. To accompany their search engine capability they have also created a 'Search engine Honeypot'. This works the same as any other honeypot but is specifically designed to work with search bots and piggyback bots (piggyback on genuine sites but with fail links). (Google)
 - a. Advantages – One of the only Search Engine Honeypot's on the market, developed specifically for the Google search engine.
 - b. Disadvantages – you need to run a search engine to get any real benefits from this.

4.2.3.2: Commercial

Commercial Honeypot's are ones that are designed and developed by companies for resale. These are not open source and as such the source code of the applications are restricted and protected by a vast array of legal contracts and stipulations, this restricts and limits the number of optional packages and tools available for them. Even though the tools option is reduced, the commercial option is by no means a lesser option, if someone was to go with the commercial option they have the full backing from the company that they pay.

1. Decoy Server – Formerly ‘Man Trap’ / Formerly ‘Mind Trap’. Decoy Server has recently been taken over by Symantec and renamed from Man trap; this is one of the leading commercial IDS systems that available, and is in on-going development by Symantec. (Arabtrust)
 - a. Advantages – large backing of a successful company, large array of tools and options available, full GUI included with the system, easy to set up
 - b. Disadvantages – very costly to buy, limited OS installs available
5. KF Sensor – at the core level this is based on the SNORT application, however it differs in that, because it is a paid for application, the majority of the required utilities are included with it. This makes things a lot easier to set up so concentration can be aimed towards the actual captured data packets. KF Sensor is designed to be used by corporate customers, as such it has a low overhead, simple to use and requires less training, and allows to the ability for a remote access removing any remote access from the intrusion detection tables. (KF Sensor)
 - a. Advantages – easier to use since all required packages are included (Inc. GUI), can provide a lot more information easier about the data packets and external connected users
 - b. Disadvantages – very costly to buy, limited to Windows OS for running
2. Specter – this is currently the market leader in IDS systems. One of the options that Specter boasts about is the ability to allow for the option of remote access, but not counting it as an intrusion into the network. (Specter) Designed to simulate up to 13 different OS’s, Specter will lure attackers away from the main network keeping it secure. The software simulates numerous typical services including SMTP mail, FTP, POP3, IMAP4, HTTP, Secure Shell (SSH), DNS, Sun remote procedure call (SUN-RPC), NetBus, Subseven Trojan, BackOrifice2K, Telnet, and finger. SPECTER immediately alerts administrators of hostile activity, offers detailed activity logs and a log analyser, and provides information about the sophistication level of any attack. (WindowsITPro)
 - a. Advantages – currently industry leading in IDS paid for software, easy install with gull GUI
 - b. Disadvantages – Highly Costly to buy with no free trials, only runs on windows
3. NetBait –This can be installed on and adapted to any segment of your network. It gives you the flexibility to configure NetBait hosts based on your own setup of actual OS and service configurations. In addition, it provides you with more ways to capture intruder data, while keeping all generated traffic private and under your complete control. (NetBait)

- a. Advantages – an open source IDS, install on any current OS (Windows – Linux)
- b. Disadvantages – not free, pay for IDS through support fees, requires a large network to set this up.

4.3: Project setup

There will be a number of elements to this project that will need installing and set up, however there are only a few mandatory installs the rest will be option and only done to help improve the results and understanding of.

Mandatory	Optional
IDS	Graphing tool
Capture tool	Secondary capture tool
Database	TCP/IP viewer
Data miner	Database GUI

4.3.1: Setup

To set this project up there are a number of things that are going to be needed, aside from the obvious of an Internet, network connected computer. We will need the IDS tool that we have talked about in the above section to detect the intrusions into the network and track their movements, and a database to store the information in. Depending on the IDS option however, there will be a need for other various running applications. The majority of the open source tools (OS option excluded) will also require the addition of a capture utility to turn the information from the IDS into a readable format (dependant on IDS version) so that it can be written to the database.

IDS: for the IDS tool I am mostly looking at the open source tools. This is because the open source option is not just free, which is a bonus, but because open source has a massive backing community, although if you have a paid for commercial one, there is also a lot of backing through the supplying company. With the commercial ones you're still limited in what they can provide. The open source IDS tools have no real limit, the code behind them is, as it sounds, OPEN, and as

such people have the ability to create any additional package that they can think of, for anything they might need.

The reason for using an open source IDS is that if I don't have something available to me on the original download, and I need something else in addition to the IDS, then I will most likely be able to get it. However I am fairly confident that I will not need this, as I will be using a straight forward IDS. The only addition that will be needed will be a Capture tool to turn the information from the IDS into a readable format to be written to the database.

At the moment the option I have decided on in to run with SNORT, but I may decide to run an additional Honeypot for testing and comparison in a virtual machine using the Honeywall OS.

SNORT: Which is in fact the IDS leader, especially since nearly every other IDS is based in some form or another around this tool, and because it was also the original IDS. SNOT has a simple installation process and is able to get up and running with the minimalist amount of fuss and hassle. However SNORT does need a capture tool installed as well as a few other applications depending on the OS it is being run on to make it work (Barnyard, WinPCap) but again these install and run with the minimal fuss and hassle. SNORT also has the great advantage that because it is open source it has been designed in such a way that you can install the features and packages that you require and nothing else, in this case I will not run a GUI. This will allow for a low strain on the CPU and memory. SNORT will be installed on Windows server 2003 which will have various services running on it to act as an enticement for anything that might be looking into the network to see if there is anything of interest.

Honeywall is my other option. The advantage with this is that it is a fully installable OS that runs SNORT as its IDS core, the distribution of Linux is based on Cent-OS. The advantage with this is I don't need to worry about installing other additional packages or application, with the exception for a database. Even though this will have a fully installed OS all work will still be done through the command line interface (cli). One of the advantages with Honeywall is that you can run it as a live CD, the use of this is that you can set up a machine without having to actually install anything, so it can be run as a test in a new location or can be run as a diagnosis tool for a network.

Other than the IDS tool there may be a need for other applications such as the database and capture tool if needed.

The **database** is going to be used with the data mining section, but its main responsibility is to store the information that is produced from the IDS/Capture tool. Depending on which database is used (SQL or ODB). The data mining tool will either be 'Rapid miner' or 'Oracle data miner' (ODM), at this moment though I will be sticking with an SQL database and Rapid miner, though I will be testing the Oracle option once the final setup of the Honeypot has been established and is running.

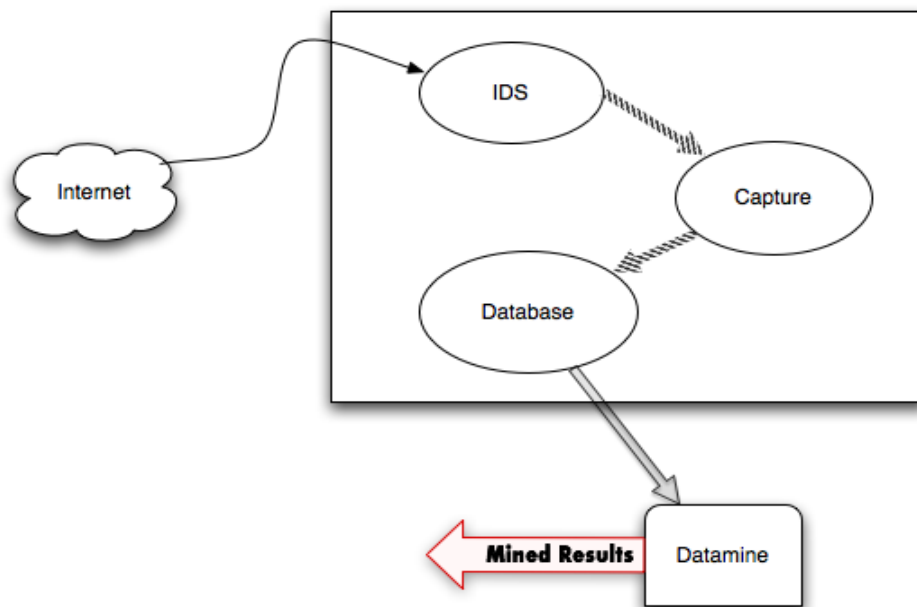


Figure 15: Recording stages (MacTiernan, 2010/2011)

How does this all work together?

- We have the IDS tool that reads the information from the attacker's data packers.
- To interoperate the information from the IDS we need a Capture tool, this can be pre-included with the IDS or can be an external option.
- Once the Capture tool has converted the information into a readable format, it can then write to a database (if required).

Once we have the data in the database we can do what we want with it, in this case it will be taken back out at regular intervals (records will be kept of this) to be mined, to find the common information, and find the trends that indicate an attacker. This data can then be used to create new rules that will, if done correctly, block that attack in the future or attacks that share similar traits.

4.4: Requirements

Required to run SNORT IDS	Additional Installations and Equipment (optional)
Dedicated PC	Barnyard
Open Internet connection	Winshark (backup to WinPCap and SNORT)
SNORT IDS application	TCP View (shows active TCP connections)
Data Mining application (rapid miner)	MySQL Dashboard (admin tool)
Database (MySQL)	Network Switch
WinPCap	Graphing software (Adodb, BASE)

Figure 16: System requirements table

Chapter 5: The Project

5.1: Chapter Introduction

In this chapter I will be guiding you through the different steps taken to setup, run and dismantle the entire project. This will be filled out like a diary on a phase by phase basis and will detail relevant and sometimes irrelevant information about the system.

5.2: Phase 1

Phase 1 involves an understanding as to what the entire project is about, how it works and what it is used for.

5.2.1: The setup

The setup of the system was talked about being at the college on the research network, however due to accessibility and possible security risks it was decided that it would be better at my home network. This however proved its own risks and problems, since the home network has about 22 different connected machines through the house owned by four different people.

With it all being on a home network, security is the ultimate boundary, how do I allow enough traffic into the network to monitor it while restricting everything else to protect the other machines on it. In Figure 17 above we see how the network is setup, with wired and wireless equipment. The router its self is a *Linksys wireless N* home router that has a built in firewall, traffic filter and DMZ. Because this router filters so much traffic and has its own database that it keeps up to date, allowing the traffic I want in is not that easy. Running the setup as standard will not allow anything that would be of any use into the network, so the only option is to remove the firewall and turn off the routers preventative measures. But this will cause its own problems.

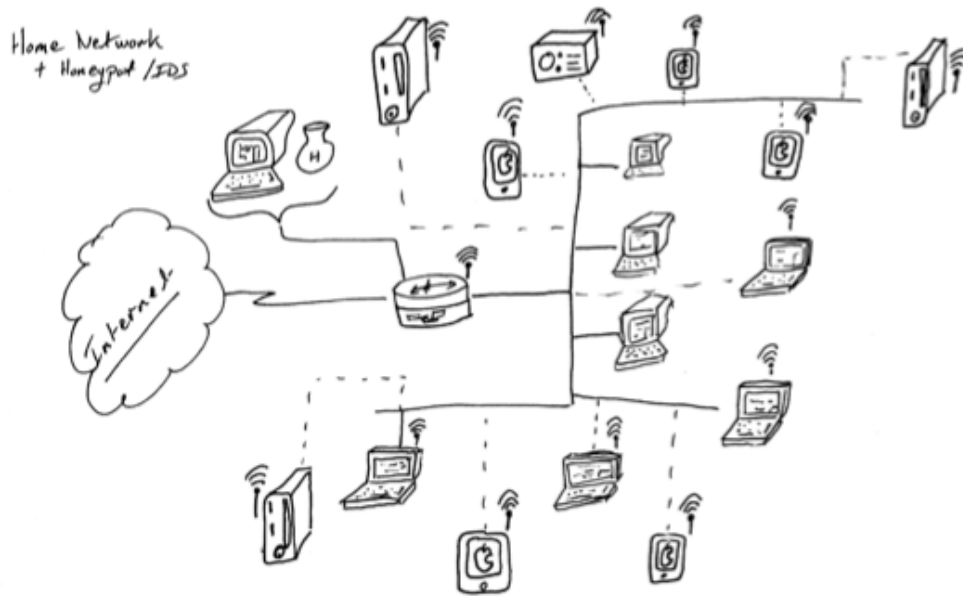


Figure 17: Home network setup (MacTiernan, 2010/2011)

To reduce any danger to the other machines on the network all other computers and wireless devices will be monitored separately and will have their own full security enabled (firewalls and filters).

5.2.2: Equipment

For the equipment for this project I will be using a normal (but high quality) home router (Linksys wireless N – wrt350N gateway router) with the firewall turned off to allow maximum network traffic for the IDS filtering and classify.

I will also have the firewall turned off on the computer that is going to be used for the IDS, but to prevent total corruption of any data on the hard drive should something happen, there will be a second partition on the hard drive to record all the data with the MySQL database, this information will also be backed-up to an *Archive* database as it is recorded.

5.2.3: System Additional

What is needed for the setup to work? Aside from the equipment that was already mentioned, there will be some additional software that will need to be setup and run at different stages.

1. Initial installation and setup of SNORT and ancillaries
2. Database: MySQL to hold all the information that is collected
3. Rapid Miner to extract the outliers and other information that might be needed
4. WinPCap is a windowing capture utility that will take each piece of information that enters the network and put it together to form a readable packet by the IDS

Also run will be a few backup systems to either catch any problems or act as a backup in information, and graphical illustration tools.

1. KF Sensor will watch all traffic and will also act as a decoy system on the network in case something should happen (this is the free trial version so has limited configuration options)
2. BASE and ACID will allow the formation of graphs using PHP

5.3: Phase 2

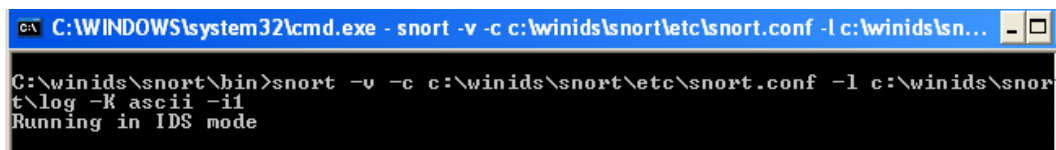
Phase 2 involves implementation of the IDS and system. This is where we can setup it up and run it.

5.3.1: Implementation and setup

- WinPCap should auto run when the machine is started; this is the design of the actual utility itself.
- MySQL should also auto start and run when the machine is started, though a manual start may be required (*mysld* at the command prompt to start)
- SNORT can be setup as a windows service is desired, so that it also runs at the machine start-up using the command “*snort /SERVICE /INSTALL -c [install location] -l [log location] -K ascii -ix*”. This will then run SNORT when you log into windows. However for this project we will not bother with this and simply run the IDS with the command in Figure 18 to allow us more flexibility as to when we want to run it.

5.3.2: Running IDS

Because SNORT is a Linux based application, it also runs like a Linux application. To start it you need to enter a command on the command line or terminal with some extra information (see) which will tell the computer where to run the application from, and tell SNORT what to do with the running configuration, as well as other switches that can be used.



```
C:\WINDOWS\system32\cmd.exe - snort -v -c c:\winids\snort\etc\snort.conf -l c:\winids\sn...
C:\winids\snort\bin>snort -v -c c:\winids\snort\etc\snort.conf -l c:\winids\snor
t\log -K ascii -i1
Running in IDS mode
```

Figure 18: SNORT command line start

- Snort = tells the computer to run the application
- -v = verbose mode, so that I can see what is going on with the system. This however can cause a slowing of the computer since it must now process the graphics as well as the network traffic, this is an option and does not need to be run.
- -c = tells the application to run using the *configuration file* which we must tell it its location after the switch.
- -l = tells snort to log all alerts and other traffic to a .txt or .ids file. This log location can be defined in the configuration file. The location of the log files placement will follow the switch like with the configuration file location
- -K = tells snort that all information will be recorded in a format which will also follow this switch (also available is HEX, Binary and a few others)
- -ix = the 'I' refers to the interface, while the number that follows it is the interface number (-i1 = interface number 1) this must be set to the interface that you want to monitor. We can use '-W' to find out what ports are connected to the network that we can monitor.

Once SNORT loads up the configuration file it will display all the other files and rules that it was told to implement. This screen can take a while to load since there may be a lot of information on it.

***NB.** Majority of the rules that SNORT uses are commented out from the rule files to reduce the number that will be loaded, this is to allow for users to customize the system for what they need. This project however will be run with all rules active for maximum classification of traffic.*

```
==== Initialization Complete ====

  o''~>~
  ' ' ' '
eam

-*> Snort! <*-
Version 2.9.0.3-ODBC-MySQL-FlexRESP-WIN32 IPv6 GRE <Build 98>
By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-t

Copyright (C) 1998-2010 Sourcefire, Inc., et al.
Using PCRE version: 8.10 2010-06-25
Using ZLIB version: 1.2.3

Rules Engine: SF_SNORT_DETECTION_ENGINE Version 1.13 <Build 18>
Preprocessor Object: SF_SSLPP <IPv6> Version 1.1 <Build 4>
Preprocessor Object: SF_SSH <IPv6> Version 1.1 <Build 3>
Preprocessor Object: SF_SMTP <IPv6> Version 1.1 <Build 9>
Preprocessor Object: SF_SDF <IPv6> Version 1.1 <Build 1>
Preprocessor Object: SF_FTPTELNET <IPv6> Version 1.2 <Build 13>
Preprocessor Object: SF_DNS <IPv6> Version 1.1 <Build 4>
Preprocessor Object: SF_DCERPC2 <IPv6> Version 1.0 <Build 3>
Commencing packet processing (pid=3504)
```

Figure 19: SNORT initializing

The MySQL should already be running but should be checked anyway, this can be started through the command line with “*mysqld*” from any point in the system.

Once SNORT starts we will see the logo seen in Figure 19 with the SNORT pig. This will list through everything that is being installed and implemented in this running, and will show the opening and installation of all the rules. This can take some time (took my system over 3 minutes to begin)

5.3.3: Relationship between systems

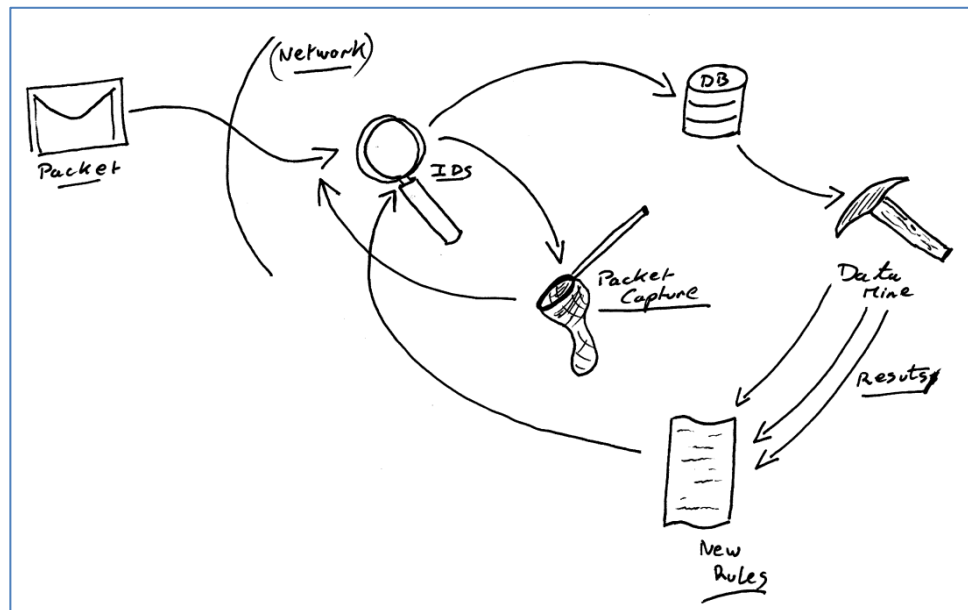


Figure 20: System relations (MacTiernan, 2010/2011)

- When the packet enters the network it is examined by the IDS
- If it is deemed as safe then nothing is done, otherwise the information in the packet is logged to the database with the packet capture utility (WinPCap). The packets should only be placed in the database if they are identified as being possible or actually dangerous, this means that the IDS has classified the packets and has a good idea what they are, though some may be able to be misclassified or slip through the net.
- Once in the Database, the information can be extracted by Rapid Miner to perform some various information retrieval processes on it.
- Should anything of interest be available or gained from the mining process, then we can generate our new rules. However there is only a need for the generation of rules should the information not already be classified, which it should be in the first place since it has been placed in the database.
- The new rules are then added to the existing ones in the SNORT rule library. Since the SNORT rule library is so extensive there may already be duplicates in there, or they may be commented out, this should be checked for.

5.4: Phase 3

This is the general running of the phase, it involves very little interaction once the system is running, but it would be advised to monitor and create backups of the data at various intervals and stages.

In the case of this project there will be 4 backup stages at the end of each of the 4 weeks that the IDS will be running. Each backup will consist of the data that was collected, screen shots of each process of working and fresh start for the next stage (not including the SQL database) to allow for the system to start fresh each time.

5.4.1: System shutdown

This application will be run over 4 weeks, and at the end of each week the system will be shut down to ensure that all records can be safely backed up and events can be checked.

Ending SNORTS services is very simple, by just pressing '*ctrl c*' will cause it to finish what it is doing. Once it does close SNORT will display some statistical information about what it has experienced (see Figure 21).

5.4.2: Checking records

Assuming that SNORT has done its job, and has filtered the information that we need, and it has been recorded it, SNORT will give you a report of all the network activity while it was running. This will include how many of each type of protocol packet (tcp, ip, udp ...) was recorded, as well as how many packets were scanned 'v' how many recorded. As we can see in Figure 21 SNORT scanned a total of over 16 million packets but only filtered or recorded about 10% of that (logged 1.7 million)

5.4.3: Finding interesting information

Once we have the system shutdown we can now check the database for information about what has happened. SNORT has the ability to create its own database tables at the start of the installation. To check these we need to access our database and view what is there, depending on how the setup was done there will be from 20 tables up in the database with a possible backup or archive database to work alongside it.

In the database there are a number of tables that can be classed as no use, one of which is the *Data* tables which holds the encrypted data that was received on the interface that we are monitoring.

Tables of interest will be the Header tables (TCPHdr, IPHdr ...) and the event table which will record all the traffic regardless of what it is, but will have little information about it.

5.4.4: Processing of data

To process the data that we have now collected we will need a mining tool or at least some form of graphing tool. Recommended would be Rapid Miner or ODM, having not worked with ODM we will stick with Rapid Miner, but be aware that with Rapid miner (being a java based program) it can take up a lot of the computer resources (i.e. ram / memory) and because of this we will most likely need to run with a sample of the data rather than the entire data set.

5.5: Phase 4

Phase 4 is the crux of the project, this is where I will take the data that I have and turn it into a decent result that will show what data has been coming into the network and what it was trying to do.

5.5.1: Results

Once we have all the results saved to the database we can now start to use them in our mining tool (rapid miner). To do we will need to import them into the process window. Rapid miner allows us to directly access the database to use the results but not to actually save them in the process itself.

To do this we need to use the *Read Database* operator, this will allow us to access the tables located in our database and bring all the information into our mining operation. Be sure that the SQL is running or rapid miner won't be able to read anything.

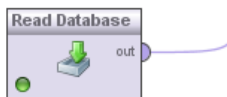


Figure 22: Read Database operator

Once we read in the tables that we need (this operator uses queries to read in specific information) we can then output it (see Figure 23) and see what is available, this will now show details about the data that we are using. We will return to this aspect at a later stage.

ExampleSet (3923 examples, 0 special attributes, 4 regular attributes)					
Role	Name	Type	Statistics	Range	Missings
regular	sid	integer	avg = 1 +/- 0	[1.000 ; 1.000]	0
regular	cid	integer	avg = 19899.497 +/- 11401	[6.000 ; 39458.000]	0
regular	signature	integer	avg = 5.026 +/- 0.298	[1.000 ; 9.000]	0
regular	timestamp	date_time	length = 19 days	[15-Feb-2011 19:58:45 GM	0

Figure 23: Event Meta View

5.5.2: Conclusions / pre-Conclusions

The expectation for the data after mining is that we will find a number of specific outliers from the norm of the data that will display unusual activity on the network. This may either be an in or outbound action, but may be difficult to discover in which direction due to the NAT on the external of the network. This will be looked at when it happens.

Once all the data has been gone through, we can then look to see if there is any particular pattern that can emerge from what has been collected (dates and times, types of data, line activity ...) which we can decide the types of steps that could be taken to prevent this in future. It is expected that there will be some activity but due to time and location there may not be an exceptional amount.

Looking at the results that have been obtained from the 4 different protocol tables in the data set we can now come to some conclusions as to the kinds of that have and are able to penetrate at specific ports and levels of the network.

Chapter 6: Data Analysis

6.1: Chapter Introduction

In this chapter we will look at some of the processes that were run and the results that were obtained from them. These results will be examined to find the components that make up the hazardous and dangerous data.

6.2: Design introduction

Because the IDS system is not primarily designed in the same format as the Honeypot, the process of finding dangerous traffic or signatures is also different. Whereas with the honeypot we look for the patterns in data traffic to find the malicious and dangerous signatures or something that will indicate such, with the IDS system we are primarily looking for traffic that is out of the ordinary, or 'not the norm' for the majority of traffic that has been recorded or that enters the network.

With the data that has been gathered we already have filtered traffic, but due to the number of machines on the network it may not be entirely possible to filter all traffic that comes through the access point¹⁹. With this we will be running the data through a series of 'outlier' operators in the rapid miner application to find the data packets that are different enough as to make them stand out from the rest.

These outliers will then be examined to find out what they are, some of the key parts that may be of interest would be port source or destination and protocol, but because the network is uses NAT²⁰ the IP addresses will almost always belong to the attached network and as such the source and destination IP will be of little to no use.

¹⁹ Entry point into a network (usually a router).

²⁰ NAT changes the IP address from a private to a public and vice-versa, as it passes through the Entry point.

Another part of this data that might be of interest could be the time that the data was collected. Since this data is all being collected on a private network, the time of general network use is quite well known, and as such it will be known what kind of internet traffic will be used (gaming²¹, email, general web surfing), other areas of the data will be looked at as we encounter it through the following processes.

6.2.1: Imported data

Looking at the available data Tables in the SQL database (see Figure 24) we see a lot of unused tables. This is due to not having them implemented through the different optional packages that are available to a Linux based program such as SNORT.

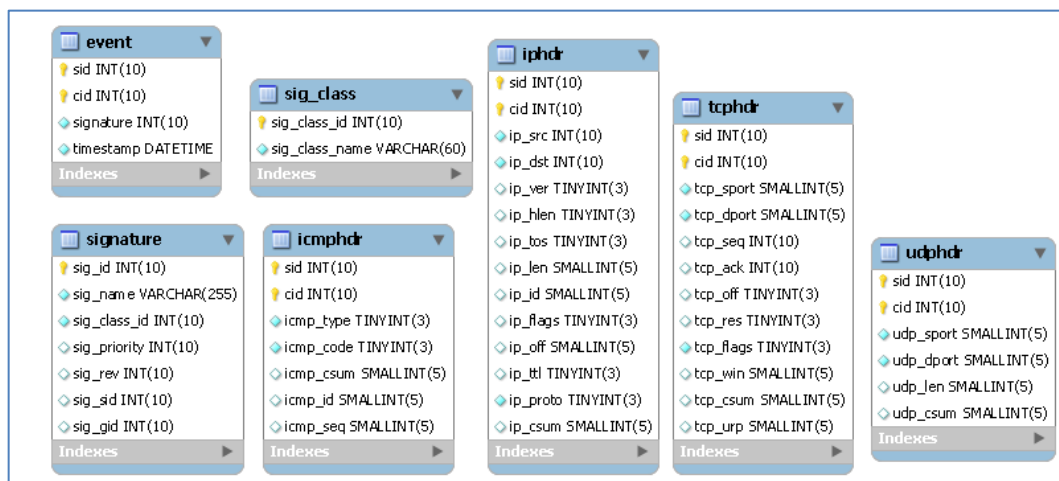


Figure 24: ERD Diagram - Tables that will be used

The *BASE* and *ACID* tables relate to optional extras that will produce a set of graphs and act as a backup to the original system. *BASE* (Basic Security Analysis Engine) is a web page front end to the *ACID* graphing tool. Both run on the PHP environment, but are not needed to run or utilize *SNORT*.

²¹ Internet Gaming, MMO's (massively multiplayer games) or console games all done over the internet.

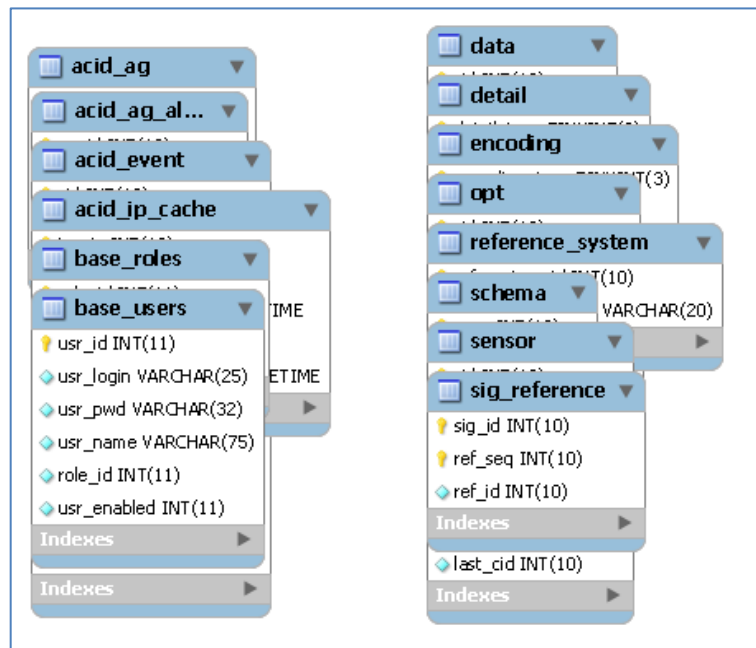


Figure 25: ERD Diagram - Tables not used

In Figure 25 we see the other tables that are in the database. However these tables will not be used for a few reasons. The ACID and BASE tables are empty due to the fact there were never implemented, and the tables to the ‘right’ of them have little to on useful information held in them. *Data* holds the actual data from the recorded packets in it, but due to that all being HASHED it is all encrypted. Some of the other tables such as *encoding* and *sensor* will only have one value for every entry since there is only one IDs machine and one database being used.

For example:-

- Opt has no information contained in the table.
- Data is the actual data itself and because it is internet traffic data, it is hashed²² and encrypted.
- Sensor, we are not running any additional sensors, and as such there is only one listed with an ID of ‘1’.
- Encoding is the system encoding that is doing the recording, which is linked to the Sensor table.

²² Encrypted using the *Hashing Algorithm*

Looking at the tables that are of interest there are a number of different ones available. The '*Hdr*' tables (IPHdr, ICMPHdr, UDPHdr, and TCPhdr) contain header information from each collected packet, which will have such things as the source and destination IP address, source and destination port, time to live and other areas that will be looked at, at a later date.

In particular to these '*Hdr*' tables the other one that will be accompanying every time will be the Event table, which contains a record of ever recorded piece of information (39472 recorded events) during the time of running, though this has actually been filtered from original which was in the region of approximately 30million packets scanned. This will provide an ID to the data as well as time record of when it was collected and a signature id ranging from 1 – 3 depending on the level of threat.

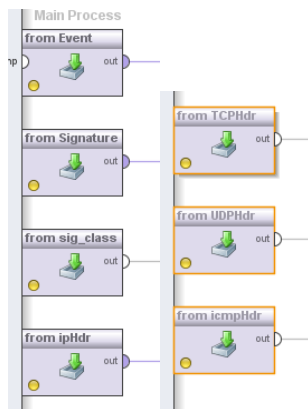


Figure 26: Included tables in rapid miner

With the header tables (*Hdr*) the signature tables (*sig_class_name*, *signature*) will also be included which should link to the header tables to tell us what kind of packet was recorded, though this will not link across in some cases. We will also see the *Event* table included as a base line reference for the ID since this holds all the packets (if a limited amount of information) that have been recorded.

6.2.2: Standard design Features

Through the different process that will be run there will be a number of what can be classed as ‘Standard implementations’, these will be operators and blocks that will be imported into the design process that will provide a function to finding the information that we need.

In the processes that will be done here though, the basic design of the process will generally be the same for each one run.

- Read database – will read in the tables that we need from our SQL database
- Set role – lets us choose the ID for the join
- Join – will join different tables together with the ID to complete the Data field information
- Detect outliers – finds the outliers that there are or are not
- Filter Examples – will allow us to filter out the outliers to make them easier to read

With the imported data base’s from the different protocol types (IP, TCP, UDP) there will also be a need for the *Signature* tables to be included, however some of the other tables will not have a link available between them. In this case though they are still included and still counted, the information provided from them will be ignored.

6.2.3: Outlier operator

In the following processes we will be using the ‘*detect outliers*’ operator. This will allow us to find the data that is not of the norm or beyond the normal trend (trend line). Of the four that are available (Distance, LOF, COF, Densities). while each of these do the same thing, they each do it in a slightly different manner and produce varying results. Each of these operators uses a slightly different format in finding the various outliers, some of which are better than others.

Densities: The densities outliers function works by taking a group of points and testing them against the rest of the data-set. Should these points or part of them be further away than the defined distance then it is classed as an outlier, otherwise it is re-entered back into the dataset for re-grouping.

This operator is a DB outlier detection algorithm which calculates the $DB(p,D)$ -outliers for an ExampleSet passed to the operator. $DB(p,D)$ -outliers are Distance based outliers according to Knorr and Ng. A $DB(p,D)$ -outlier is an object to which at least a proportion of p of all objects are farer away than distance D . It implements a global homogenous outlier search. (rapid-i)

This form of outlier detection carries a distance value as well as a percentage value relating to the number of points that relate to the distance.

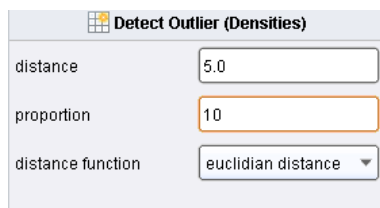


Figure 27: Densities parameters

After testing a few different values and variations in the outlier parameters, the outcome is that it doesn't work for this data set, since all the returned values are 100% outliers (see Figure 28). This will not work for use since we are trying to find outliers and this operator simply classifies everything as an outlier (1585 true / 0 false) which is no use to us in finding specific outliers within the dataset.

id	signature	integer	avg = 4.992 +/- 0.432	[1.000 ; 9.000]
outlier	outlier	binominal	mode = true (1585), least = false (0)	false (0), true (1585)
regular	sid	integer	avg = 1 +/- 0	[1.000 ; 1.000]

Figure 28: Densities outliers' classification

Class Outlier Factor (COF): Detection of outliers is done by ranking each point in relation to its neighbour and then normalizing the deviation distance. The distance between the point and its nearest neighbour is then summed and is distanced to it class, within a given range to decide if it is an outlier.

(COF = PCL(T,K) - norm(deviation(T)) (rapid-i)

Using this formula the function will find the outliers by class rather than its nearest neighbour. But is more likely to find the rarer cases of outliers.

It detects rare / exceptional / suspicious cases with respect to a group of similar cases (rapid-i)

While in theory this form of outlier detection would produce some good results from finding hidden or specific outliers from the already classified data, it will not apply to this dataset. Each time it is run it says that something is of a different or wrong data type (i.e. Nominal) and when corrections are made, a new error is thrown up.

Unlike with others depending on the distance function that is chosen will actually change the values that you are able to enter into the parameters. However this still doesn't change that it will not work with this data set.

Distance: This one seems to be the best option in finding the various outliers that we have in our dataset. Whereas the others are concerned with X Y and X, this one looks for a given number of outliers with a given number of neighbours (user defined) however the problem with this one is that because the parameters for the outlier location are user defined they may not be entirely accurate and as such if we ask for 10 outliers we will find 10 regardless of how many there in reality is. With this outlier operator user discretion would still be needed, and a keen eye to find the distinguishing outliers from the defined information.

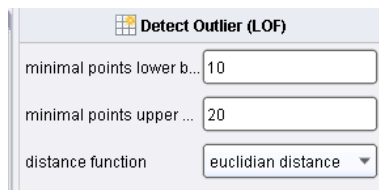
This operator performs a D^k_n Outlier Search according to the outlier detection approach recommended by Ramaswamy, Rastogi and Shim in "Efficient Algorithms for Mining Outliers from Large Data Sets". It is primarily a statistical outlier search based on a distance measure similar to the DB(p,D)-Outlier Search from Knorr and Ng. But it utilizes a distance search through the k-th nearest neighbourhood, so it implements some sort of locality as well. (rapid-i)

This process works by taking the points that are the furthest distance from their k-neighbour and saying 'that is an outlier', it can say this because it takes that because that point is more out on its own it has less or no similarity to any other and as such makes it an outlier to the rest of the data.

Despite the problems with this form of outlier detection, the Distance outlier detection is the simplest and most efficient for largest data sets, and given that we have an issue in this project with

the computer memory constraints this will prove to be the best option to run, though may not prove the best results given a better processing environment.

Local Outlier Factor (LOF): With the LOF function we have the ability to find our outliers with a distance rating applied to each. This distance is the distance from the other objects within that container. Each point is given the distance based on the minimal upper and lower bound points of the outlier function.



Detect Outlier (LOF)	
minimal points lower b...	10
minimal points upper ...	20
distance function	euclidian distance ▼

Figure 29: LOF parameters

The approach to find those outliers is based on measuring the density of objects and its relation to each other (referred to as local reachability density). Based on the average ratio of the local reachability density of an object and its k-nearest neighbours (e.g. the objects in its k-distance neighbourhood), a local outlier factor (LOF) is computed. The approach takes a parameter MinPts (actually specifying the "k") and it uses the maximum LOFs for objects in a MinPts range (lower bound and upper bound to MinPts). (rapid-i)

On the test set, the results were fairly good in that they produced good values relation to how far the point is away from the rest of the set, and as such you can evaluate the relevance of that point. However unlike with the distance function this will only produce the outliers that are outlying rather than produce 10 because we asked for 10.

signature	outlier	sid	ip_src	ip_dst	ip_ver	ip_hlen	ip_tos	ip_len	ip_id	ip_flags	ip_off	ip_ttl	ip_proto	ip_csum	tim
1	0.400	1	323223577e	323223577e	4	5	0	84	61214	0	0	64	1	1896	15-F
1	0.310	1	323223577e	323223577e	4	5	0	84	52746	0	0	64	1	10364	15-F
1	0.266	1	323223577e	323223577e	4	5	0	84	46958	0	0	64	1	16152	15-F
1	0.139	1	323223577e	323223577e	4	5	0	84	1356	0	0	64	1	61754	15-F
1	0.139	1	323223577e	323223577e	4	5	0	84	1356	0	0	64	1	61754	15-F
2	0.132	1	323223577e	323223577e	4	5	0	84	34393	0	0	64	1	28717	15-F
1	0.130	1	323223577e	323223577e	4	5	0	84	11307	0	0	64	1	51803	15-F
3	0.126	1	323223577e	323223577e	4	5	0	84	3692	0	0	64	1	59418	15-F
2	0.124	1	323223577e	323223577e	4	5	0	84	3692	0	0	64	1	59418	15-F
1	0.124	1	323223577e	323223577e	4	5	0	84	3692	0	0	64	1	59418	15-F
2	0.100	1	323223577e	323223577e	4	5	0	84	28425	0	0	64	1	34685	15-F
1	0.091	1	323223577e	323223577e	4	5	0	84	25980	0	0	64	1	37130	15-F
3	0.080	1	323223577e	323223577e	4	5	0	84	24168	0	0	64	1	38942	15-F
1	0.071	1	323223577e	323223577e	4	5	0	84	22366	0	0	64	1	40744	15-F
1	0.071	1	323223577e	323223577e	4	5	0	84	22366	0	0	64	1	40744	15-F
2	0.071	1	323223577e	323223577e	4	5	0	84	21617	0	0	64	1	41493	15-F
1	0.071	1	323223577e	323223577e	4	5	0	84	21617	0	0	64	1	41493	15-F
4	0	1	323223577e	323223577e	4	5	0	84	29529	0	0	128	1	17197	15-F

Figure 30: LOF Outliers

As we can see in Figure 30 the further the distance from the K point the high the number. And as we will see later the values that have been given the High number in this test (IPHdr table) is also the same as the operator that we have chosen to use.

This would indicate that even though the Distance and LOF are different in the way they perform, they both return relatively the same operators in this dataset. As such because we will be dealing with a number of different size data set (from 36000 to 58 data entries) I will stay with the predefined number of outliers in the *Distance* outlier operator.

Distance function: In all of the available outlier operators there is an option called *Distance Function* which has a number of different distance functions available which will calculate the location of the outlier in a different way. However if the outlier is a 100% true it should be found in all of them regardless of how it is found. But the advantage comes in that each of these functions will be able to find other possible outliers that may or may not be entirely visible at first.

Each of the 4 available outlier operators carries the same distance functions, squared distance, cosine distance, inverted cosine distance, angle, and euclidian distance.

Euclidian distance: is the straight-line distance between two points (as the crow flies).

Square distance: is the root-mean-average between 2 points. The outliers are decided on based on the square of each point.

Cosine distance: this is a similarity measure, done on the cosine of each of the points from its neighbour.

Inverted Cosine distance: this is the reverse of the Cosine distance.

Angle: working along the same lines as the Euclidian distance, this takes the straight line between two points and the angle between them from a third point.

The COF outlier detector doesn't use these however, this has its own functions that are specific to itself, depending which measure type you want to do. There are 4 groups of function, Mixed Measure, Nominal Measure, Numerical Measure and Divergence. Each of which has its own choice of functions.

- ***mixed measure:***
Range: MixedEuclideanDistance; default: MixedEuclideanDistance
- ***nominal measure:***
Range: NominalDistance, DiceSimilarity, JaccardSimilarity, KulczynskiSimilarity, RogersTanimotoSimilarity, RussellRaoSimilarity, SimpleMatchingSimilarity; default: NominalDistance
- ***numerical measure:***
Range: EuclideanDistance, CamberraDistance, ChebychevDistance, CorrelationSimilarity, CosineSimilarity, DiceSimilarity, DynamicTimeWarpingDistance, InnerProductSimilarity, JaccardSimilarity, KernelEuclideanDistance, ManhattanDistance, MaxProductSimilarity, OverlapSimilarity; default: EuclideanDistance
- ***divergence:***
Range: GeneralizedIDivergence, ItakuraSaitoDistance, KLDivergence, LogarithmicLoss, LogisticLoss, MahalanobisDistance, SquaredEuclideanDistance, SquaredLoss; default: GeneralizedIDivergence

Figure 31: Detect Outlier (COF) distance functions (rapid-i)

6.2.4: Other areas for information

Along with the information that is held in the databases, SNORT also has the ability to write out certain information to a file, or files.

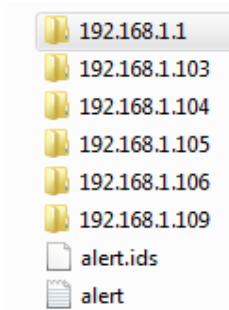


Figure 32: SNORT output files

In the SNORT configuration there is an allowance to write out to what is called an *Alert* file, while will record all alerts to one file as well as write to a folder for each network IP that in involved with that alert.

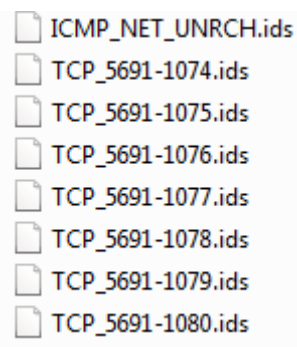


Figure 33: Alert folder

Contained in these files are some information pertaining to the actually data and packet itself. It is not as extensive as the information held in the database but does offer some easier to understand information and is more immediate and to-hand than that in the database.

Figure 34: Alert file contents

Figure 35: IPHdr process information

6.3.1: Introduction

In this process we will be looking at the *IPHdr* table and we will try to identify some of the outliers that it contains. This is the biggest of all the tables in the recorded data, and as such holds all the same information as the events table (total count is 39,472), because of this we are not able to run with the entire data set because of the computer's memory²³ restraints, and as such we will be required to run with a sample of the data (having tested, the used data size is 0.06 of the entire data set or 6% of total)

bit offset	0-3	4-7	8-13	14-15	16-18	19-31
0	Version	Header Length	Differentiated Services Code Point	Explicit Congestion Notification	Total Length	
32			Identification		Flags	Fragment Offset
64	Time to Live		Protocol		Header Checksum	
96	Source IP Address					
128	Destination IP Address					
160	Options (if Header Length > 5)					
160 or 192+	Data					

Figure 36: IP header contents (Wikipedia)

In the IP header field there is a lot of information that is held. Of what is available we will not need the *Destination* or *Source IP Address* since it will only point to the internal IP address range of the connected network due to the NAT of the external-internal addresses, and the *IP Version* since the network is only running on IP ver4²⁴ and hasn't had IP ver6²⁵ implemented on it yet.

²³ Memory = RAM (Random Access Memory)

²⁴ IP Version 4 = older format of IP ranges that have run out.

²⁵ IP Version 6 = new ip range that will replace version 4. Contains a lot more number variations.

sid	integer	avg = 1 +/- 0	[1.000 ; 1.000]
ip_src	integer	avg = 3232235776 +/- 0	[3232235776.000 ; 3232235776.000]
ip_dst	integer	avg = 3228474786.938 +/- 87112787.200	[1249725824.000 ; 3512045312.000]
ip_ver	integer	avg = 4 +/- 0	[4.000 ; 4.000]
ip_hlen	integer	avg = 5 +/- 0	[5.000 ; 5.000]
ip_tos	integer	avg = 0 +/- 0	[0.000 ; 0.000]
ip_len	integer	avg = 43.145 +/- 16.916	[40.000 ; 162.000]
ip_id	integer	avg = 32558.973 +/- 18946.926	[18.000 ; 65518.000]
ip_flags	integer	avg = 0 +/- 0	[0.000 ; 0.000]
ip_off	integer	avg = 0 +/- 0	[0.000 ; 0.000]
ip_ttl	integer	avg = 125.514 +/- 12.370	[64.000 ; 128.000]
ip_proto	integer	avg = 5.967 +/- 0.404	[1.000 ; 6.000]
ip_csum	integer	avg = 33166.160 +/- 18941.345	[173.000 ; 65491.000]
timestamp	date_time	length = 19 days	[15-Feb-2011 19:59:30 GMT ; 06-Mar-2011 1
cid	integer	avg = 19240.179 +/- 11518.747	[13.000 ; 39463.000]
sig_class_name	nominal	mode = misc-activity (8), least = network-scan (1)	bad-unknown (4), misc-activity (8), network-s
sig_id	integer	avg = 4.231 +/- 2.455	[1.000 ; 9.000]
sig_name	nominal	mode = Consecutive TCP small segments exceeding thre	ICMP PING *NIX (2), ICMP PING (2), ICMP PII
sig_priority	integer	avg = 2.692 +/- 0.480	[2.000 ; 3.000]
sig_rev	integer	avg = 4.462 +/- 2.665	[1.000 ; 9.000]
sig_sid	integer	avg = 388.462 +/- 492.549	[5.000 ; 1917.000]
sig_gid	integer	avg = 40.385 +/- 61.489	[1.000 ; 129.000]

Figure 37: IP header Meta data

If we run the process as shown in Figure 22, and look at the *Meta data* output screen (see Figure 37) we see the information that is available in the data set, and its ranges. This can help us decide what fields of the table may of use in finding and deciphering the outliers. Looking at this we see the IP Destination (ip_dst) and IP Source (ip_src), these are not shown in the data as an actual IP range, but as a series of numbers that is recognized by the IDS application. We also see the IP Version (ip_ver) has a variation of +/-0 which means that they are all version 4, which was expected.

Looking at more of the fields that we have available, we also see that there is a lot of information that cannot be gained from the header, such as header length (ip_hlen), Type of Service (ip_tos) and a number of others that can be seen above. Anything with a variation of +/- 0 is of little use in finding outliers since this offers no alternative which means that they will all be the same value.

6.3.2: Process design setup

Setting up the Process is fairly straight forward and will be similar for each process run. Importing all of the tables into the process window, we are required to select an attribute from each to use as an ID for the 'join' function. The 3 tables that will be imported each time are the Signature tables and the event table with the header table changing for each process. When joining the header/event tables to the signature tables we will need to reset the ID values and change it to something that matches the signature tables. This means removing the original ID and setting a new one.

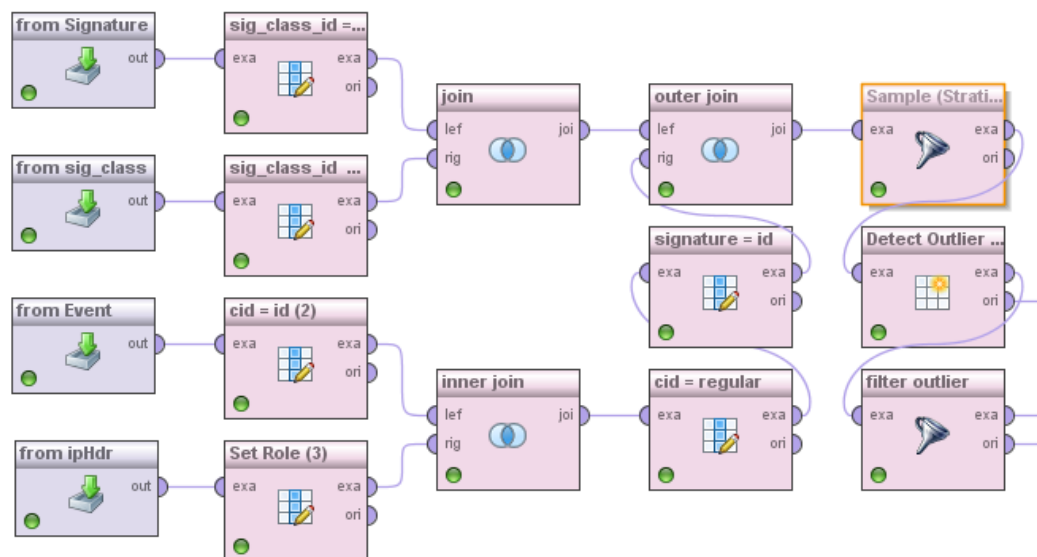


Figure 38: Process 1 - IPHdr design

The reason we use the above design is so that we can use the information from our database in the rapid miner process. We are also including a select attribute operator so that we can exclude any columns that might be deemed as not necessary (i.e. data). These two imported tables must then be joined together using their ID that we define using the *Set Role* operator.

This is the basis for most of the processes that will be run in this way, because joining the tables together will allow us to identify what row of the database they are entered in as well as identify what time they were entered into it, and should tell us what kind of signature those entries are.

Setup Information		
Operator	Attribute	Value
Sample	Sample – Relative	0.06 (6%)
Join	Join Type	Inner / Inner / Outer
Set Role	ID – Event	CID
	ID – IPHdr	CID
	ID – Signature	Sig_class_id
	ID – Sig_class_name	Sig_class_id
	ID – event/ip_hdr	signature
	Regular – event/ip_hdr	CID
Detect Outliers	Distance Function	Euclidian Distance
	Number of outliers	100
	Number of neighbours	5
Filter Example	Condition class	Outlier = true

Figure 39: Process 1 setup values

In further processes, the *Sample* operator may or may not be used depending on the number of records that will be utilized by that process. The Outliers operator will be used here to find the elements that are particularly different to the others.

6.3.3: Examining results

On running the process we see the results for the entire data set (sample window) with the outliers detected in Figure 40, what this shows is a mix of where the signature tables have and have not been applied to the different data entries, this is because there is not be a linking attribute between them, and as such means that they have little relation to each other, however this does not mean that they are irrelevant.

ExampleSet (1982 examples, 2 special attributes, 22 regular attributes)														View Filter (1982 / 1982): all						
Row No	signature	outlier	sid	ip_src	ip_dst	ip_ver	ip_hlen	ip_len	ip_id	ip_ttl	ip_proto	ip_csum	timestamp	cid	sig_class_name	sig_id	sig_name			
1	1	true	1	3232235776	3232235776	4	5	0	84	17023	0	0	64	1	46087	15-Feb-2011 19:59:30 GMT	13	misc-activity	2	ICMP PING
2	3	true	1	3232235776	3232235776	4	5	0	84	17023	0	0	64	1	46087	15-Feb-2011 19:59:30 GMT	15	network-scan	9	SCAN UPnP service disc
3	1	true	1	3232235776	3232235776	4	5	0	84	11307	0	0	64	1	51803	15-Feb-2011 19:59:30 GMT	21	misc-activity	4	ICMP Echo Reply
4	2	true	1	3232235776	3232235776	4	5	0	84	11307	0	0	64	1	51803	15-Feb-2011 19:59:30 GMT	22	bad-unknown	6	Consecutive TCP small
5	1	true	1	3232235776	3232235776	4	5	0	84	61214	0	0	64	1	1896	15-Feb-2011 19:59:30 GMT	25	misc-activity	2	ICMP PING
6	1	true	1	3232235776	3232235776	4	5	0	84	17787	0	0	64	1	45323	15-Feb-2011 19:59:30 GMT	37	misc-activity	3	ICMP PING BSDtype
7	2	true	1	3232235776	3232235776	4	5	0	84	15943	0	0	64	1	47167	15-Feb-2011 19:59:30 GMT	42	bad-unknown	7	Bad segment, adjusted
8	1	true	1	3232235776	3232235776	4	5	0	84	24168	0	0	64	1	38942	15-Feb-2011 19:59:30 GMT	49	misc-activity	4	ICMP Echo Reply
9	2	true	1	3232235776	3232235776	4	5	0	84	24168	0	0	64	1	38942	15-Feb-2011 19:59:30 GMT	50	bad-unknown	6	Consecutive TCP small
10	1	true	1	3232235776	3232235776	4	5	0	84	24651	0	0	64	1	38459	15-Feb-2011 19:59:30 GMT	53	misc-activity	1	ICMP PING *NIX
11	2	true	1	3232235776	3232235776	4	5	0	84	24651	0	0	64	1	38459	15-Feb-2011 19:59:30 GMT	54	bad-unknown	6	Consecutive TCP small
12	1	true	1	3232235776	3232235776	4	5	0	84	37420	0	0	64	1	25690	15-Feb-2011 19:59:30 GMT	69	misc-activity	1	ICMP PING *NIX
13	1	true	1	3232235776	3232235776	4	5	0	84	21281	0	0	64	1	41829	15-Feb-2011 19:59:30 GMT	89	misc-activity	4	ICMP Echo Reply
14	6	true	1	3232235776	3232235776	4	5	0	91	32800	0	0	64	6	13909	18-Feb-2011 16:26:26 GMT	153	?	?	?
15	6	true	1	3232235776	3232235776	4	5	0	156	47438	0	0	64	6	64741	18-Feb-2011 17:14:29 GMT	200	?	?	?
16	6	true	1	3232235776	3232235776	4	5	0	156	31788	0	0	64	6	14856	18-Feb-2011 17:45:04 GMT	231	?	?	?
17	6	true	1	3232235776	3232235776	4	5	0	156	52300	0	0	64	6	59879	18-Feb-2011 18:15:39 GMT	262	?	?	?
18	6	true	1	3232235776	3232235776	4	5	0	91	53893	0	0	64	6	58351	18-Feb-2011 18:30:57 GMT	277	?	?	?
19	6	true	1	3232235776	3232235776	4	5	0	156	33266	0	0	64	6	13378	18-Feb-2011 21:25:43 GMT	393	?	?	?
20	6	true	1	3232235776	3232235776	4	5	0	91	58263	0	0	64	6	53981	18-Feb-2011 21:49:44 GMT	419	?	?	?
21	6	true	1	3232235776	3232235776	4	5	0	91	6047	0	0	64	6	40662	18-Feb-2011 21:58:18 GMT	425	?	?	?
22	6	true	1	3232235776	3232235776	4	5	0	156	64906	0	0	64	6	47273	18-Feb-2011 22:00:40 GMT	430	?	?	?
23	6	true	1	3232235776	3232235776	4	5	0	91	40364	0	0	64	6	6345	18-Feb-2011 22:39:59 GMT	468	?	?	?
24	6	true	1	3232235776	3232235776	4	5	0	91	12562	0	0	64	6	34147	18-Feb-2011 22:55:16 GMT	483	?	?	?
25	6	true	1	3232235776	3232235776	4	5	0	156	45238	0	0	64	6	1406	18-Feb-2011 22:59:39 GMT	484	?	?	?
26	6	true	1	3232235776	3232235776	4	5	0	91	62563	0	0	64	6	49681	18-Feb-2011 23:56:27 GMT	545	?	?	?
27	6	true	1	3232235776	3232235776	4	5	0	156	59758	0	0	64	6	52421	18-Feb-2011 00:20:28 GMT	567	?	?	?

Figure 40: Detect IP outliers

What we asked the process to do was to detect 100 outliers from the sample of just under 2000 entries. Because SNORT has the ability to classify so many different types of packets there may not always be a link between the packet and the signature table, this is shown in Figure 40 and Figure 41, which shows the top 13 entries as having a signature entire, and the rest of the entries have no signature name or type attached. This would lead us to believe that the top 13 are known

bad data types and as such have been classified, whereas the rest have no known data types or are not of a sufficient threat level to be classified in this manner.

2	true	1	3232235776	3232235776	4	5	0	84	24168	0	0	64	1	38942	15-Feb-2011 19:59:30 GMT	50	bad-unknown	6	Consecutive TCP small
1	true	1	3232235776	3232235776	4	5	0	84	24651	0	0	64	1	38459	15-Feb-2011 19:59:30 GMT	53	misc-activity	1	ICMP PING *NDK
2	true	1	3232235776	3232235776	4	5	0	84	24651	0	0	64	1	38459	15-Feb-2011 19:59:30 GMT	54	bad-unknown	6	Consecutive TCP small
1	true	1	3232235776	3232235776	4	5	0	84	37420	0	0	64	1	25690	15-Feb-2011 19:59:30 GMT	69	misc-activity	1	ICMP PING *NDK
1	true	1	3232235776	3232235776	4	5	0	84	21281	0	0	64	1	41829	15-Feb-2011 19:59:30 GMT	89	misc-activity	4	ICMP Echo Reply
6	true	1	3232235776	3232235776	4	5	0	91	32600	0	0	64	6	13909	18-Feb-2011 16:26:26 GMT	153	?	?	?
6	true	1	3232235776	3232235776	4	5	0	156	47438	0	0	64	6	64741	18-Feb-2011 17:14:29 GMT	200	?	?	?
6	true	1	3232235776	3232235776	4	5	0	156	31788	0	0	64	6	14856	18-Feb-2011 17:45:04 GMT	231	?	?	?
6	true	1	3232235776	3232235776	4	5	0	156	52300	0	0	64	6	59879	18-Feb-2011 18:15:39 GMT	262	?	?	?
6	true	1	3232235776	3232235776	4	5	0	91	53893	0	0	64	6	58351	18-Feb-2011 18:30:57 GMT	277	?	?	?
6	true	1	3232235776	3232235776	4	5	0	156	33266	0	0	64	6	13378	18-Feb-2011 21:25:43 GMT	393	?	?	?

Figure 41: Detect IP outliers close view

Since I am the one who set up the system, I also know that the first 100 or so entries in the database are test ping's (ICMP + ECHO Reply) that were used to test that the system was working, this is shown in the top 13, some of which have been classified as ECHO's and Replies. We can also see here that there are some that have been classed as 'Bad or Unknown'; these are packets that have exceeded the threshold limit that was set in the configuration files.

6.3.4: Graphing the results

If we look at the data and outliers in the following graph's we can see the extent of the possible exceptional packets that have been collected. From looking at the graphed results the key fields of interest appear to be *protocol*, *destination* and *timestamp*'s. This is further confirmed by referring back to the Meta data tables in Figure 37 which shows that these fields have a limited but varied range of possibilities within them.

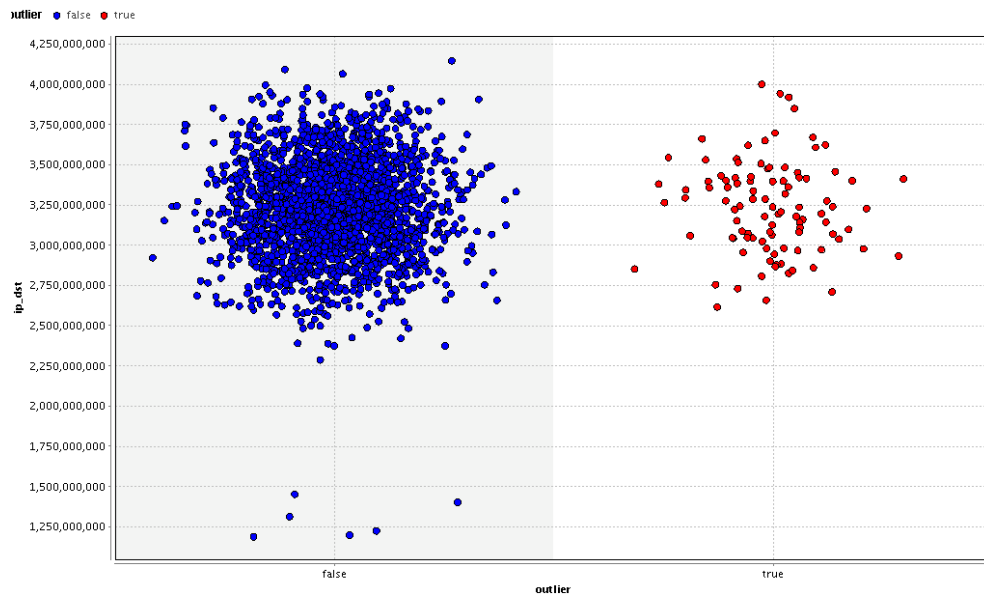


Figure 42: Outlier graph (x = outliers / y = destination)

Looking at the graph in Figure 42 see can clearly see the outlier's (true) from the rest of the data set. These outliers are shown to have a mixed range among the main cluster. From what we know about the data the majority of these being ICMP pings it is possible that they are bad replies to a ping requests.

Looking on further to Figure 43 we see a graph of *protocol* against *timestamp*. This shows that the majority of the outlying points are at the start of the collection between the dates of the '10th February 2011' and the last visible point reaching the '26th of February 2011', with the majority of the points being at the start of this time line.

This again illustrates that the original test Pings have been detected by the IP signature rules and have been categorized.

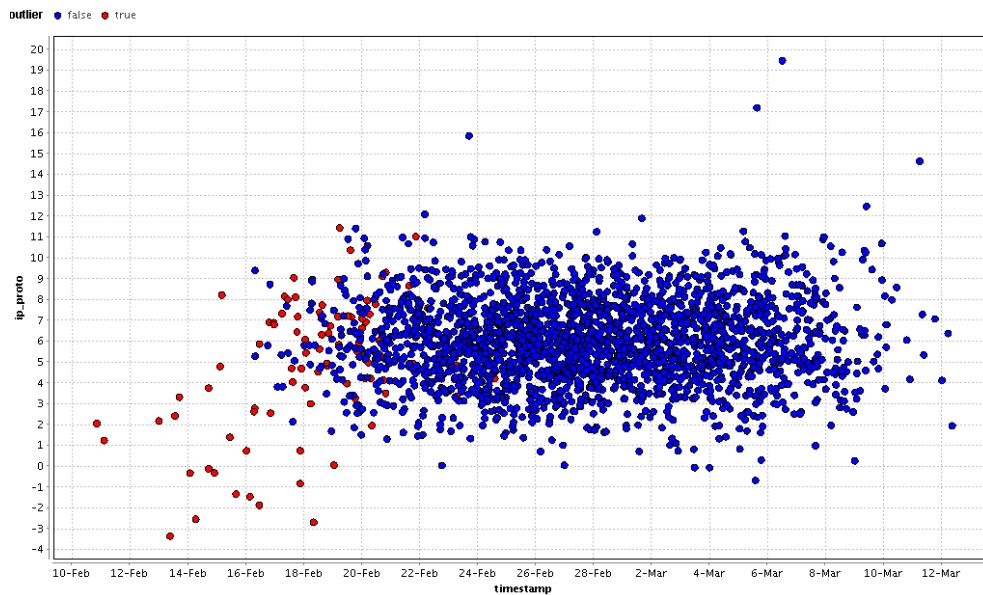


Figure 43: Outlier graph (x = timestamp / y = protocol)

After looking at all the data as a whole, we can now look at the top 13 hits that we got from the original outliers data view that were not only extracted as being outliers but also as being of type ‘X’²⁶ which was classified by the signature tables.

If we filter out the classified outliers with the *filter* block in rapid miner, and set the graph fields to the signature type and name, we can look at just the outliers and examine them a little closer. In Figure 44 we can see the makeup of the signatures in relation to each other, with the majority of outliers belonging to the “misc_activity” class. This tells us that they are a general activity and could possibly not be dangerous at all, but due to the initial testing they are shown to be out of the norm. They are listing as ICMP pings and replies.

²⁶ X = the type of data that it was assigned to.

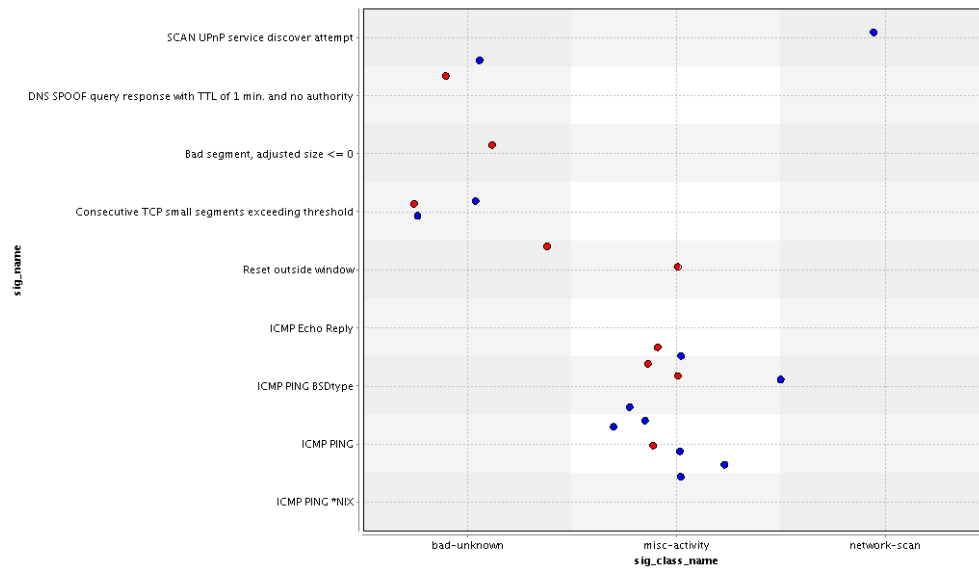


Figure 44: Outliers filtered (x = sig_class_name / y = sig_name)

However if we look at some of the other outliers in the graph we see that there are some shown to be “bad sectors” which may mean that the packet has become corrupt along the transmission due to high volumes of traffic on the line at that time. Although looking again we see that there is an entry for a “DNS Spoof” which can mean that something is trying to gain access to the network.

The ‘DNS spoof’ here is actually a spoofing attack, where someone or something is disguised as something else to try to gain access into a network. This was possible a phishing attack that was attempted on one of the machines on the network. We also see that the ‘DNS Spoof’ points are crossing over with the ‘SCAN UPnP’ points, which is a scan to try to find UPnP devices, or a device discovery. This could be an outsider trying to find a machine on the network to see if there is anything of interest on it.

Also seen on the graph is a section for ‘Reset outside window’ and ‘TCP Segments exceeding threshold’ these may or may not be anything of significance, but are the result of packets being either too big or too small for the threshold that is user defined in the ‘*Threshold.conf*’ file in SNORT

6.3.5: Conclusion

After looking at the graphed results and tabled results we can draw some conclusions. After looking the data and comparing it to the data logs, most of the data is classified as ‘Priority: 2’ or ‘Potentially Bad Traffic’, this can be anything that is sent through the network that has exceeded the used defined (or pre-defined) packet window size.

*02/18-19:08:36.255256 [**] [129:12:1] Consecutive TCP large segments exceeding threshold [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.1.109:53307 -> 192.168.1.106:139*

*02/18-19:08:36.262520 [**] [129:12:1] Consecutive TCP small segments exceeding threshold [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 192.168.1.109:53307 -> 192.168.1.106:139*

Since there are a number of computers and other machines located on the network (see Figure 17) it is more than possible that this is mostly internet game traffic or general internet surfing traffic, and as such is not entirely possible to tell which is which within the confines of the information held here.

SNORT defines three types of *thresholding* - Limit, Threshold, and Combined.

1. *Limit: Alert on the 1st X events during the time interval, then ignore events for the rest of the time interval.*
2. *Threshold: Alert every X times we see this event during the time interval.*
3. *Combined: Alert once per time interval after seeing X occurrences of the event then ignore any additional events during the time interval.*

17023	0	0	64	1	46087	15-Feb-2011 19:59:30 GMT	15	network-scan	9
11307	0	0	64	1	51803	15-Feb-2011 19:59:30 GMT	21	misc-activity	4

Figure 45: Segment of data view (sig id - 11307)

If we look at the segment of data in Figure 45 we can track the signature ID's (11307) to the help identify the different signatures and what they mean. In the case of ID number 11307 (above) *this is generated when activity relating to the spyware application "computer monitor keylogger" is detected*, this tells us that one of the machines on the network may have some form of key logger. This however is not an attack in the sense of being attacked but rather someone has been downloaded that contained this program, and it is relaying the information back to where it sends it.

Because the information in this section is so vast but yet not very well defined, in that it covers all network traffic and after looking at all the aspects of the IPHdr data there are few conclusions that can be grounded, however the IPHdr covers all traffic that is on the network and as such has some limited detail even though this table has the most number of fields. We will see more from the different "bad segments" that we have seen here later when we look at the ICMP, UDP and TCP data entries since there are more specific and can give more detail.

6.4: Process 2 – Detect outliers (TCPHdr)

TCPHdr process information	
Number of data entries	39,472
Filter rate	0.06 (6%)
Number when sampled	1982
Tables used	TCPHdr, Event, Signature, Sig_class
Outlier neighbours	20
Number of Outlier	100

Figure 46: TCP process information

6.4.1: Introduction

In this process we will look at the *TCPHdr* table and we will try to identify some of the outliers it contains. This table is of the same size as *IPHdr* table (39,472 entries) but has fewer fields in each entry (smaller by 3 fields). Again like with the *IPHdr* table we will not be able to run with the entire data set due to the computer hardware constraints, and as such will again run with a sample ratio of 6% (0.06).

Bit offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Source port																Destination port															
32	Sequence number																															
64	Acknowledgment number																															
96	Data offset				Reserved				C	E	U	A	P	R	S	F	Window Size															
									W	R	R	C	C	H	T	N	I															
									R	E	G	K	S	H	I	N																
128	Checksum																Urgent pointer															
160	Options (if Data Offset > 5)																										padding					
...	...																															

Figure 47: TCP header contents (Wikipedia)

In the TCP header field (see Figure 47) we can see the overall make-up of the header. Unlike with the *IPHdr* field there are no IP fields, since we are not dealing with the same implementation. The TCP header deals with the *Destination* and *Source* Ports of the data packet. This is an internal network protocol and as such will not need direct access to the outside network (Internet), but will need direction to a port that has access (or doesn't have access) to. Some other areas that may be of relevance when looking at TCP packets could be again *Timestamp* and also *Data Offset* or *Checksum*. *Window Size* may also be important since some attackers can alter the size of the window to try to disguise what it is trying to do.

Name	Type	Statistics	Range	Missings
signature	integer	avg = 5.035 +/- 0.183	[5.000 ; 6.000]	0
outlier	binominal	mode = false (2348), least = true (10)	false (2348), true (10)	0
sid	integer	avg = 1 +/- 0	[1.000 ; 1.000]	0
tcp_sport	integer	avg = 4727.329 +/- 9456.616	[80.000 ; 65219.000]	0
tcp_dport	integer	avg = 5484.605 +/- 1049.508	[80.000 ; 5691.000]	0
tcp_seq	integer	avg = 2132257958.782 +/- 1245441658.786	[1728029.000 ; 4293606689.000]	0
tcp_ack	integer	avg = 2178282351.300 +/- 1238220726.183	[3404731.000 ; 4294827107.000]	0
tcp_off	integer	avg = 5.102 +/- 0.543	[5.000 ; 8.000]	0
tcp_res	integer	avg = 0 +/- 0	[0.000 ; 0.000]	0
tcp_flags	integer	avg = 20.132 +/- 0.805	[4.000 ; 24.000]	0
tcp_win	integer	avg = 1745.255 +/- 9722.165	[0.000 ; 65535.000]	0
tcp_csum	integer	avg = 33164.760 +/- 18610.014	[21.000 ; 65521.000]	0
tcp_urp	integer	avg = 0 +/- 0	[0.000 ; 0.000]	0
timestamp	date_time	length = 16 days	[18-Feb-2011 16:08:57 GMT ; 06-M	0
cid	integer	avg = 19451.358 +/- 11386.964	[133.000 ; 39461.000]	0
sig_class_name	nominal	mode = unknown	bad-unknown (0), misc-activity (0),	2358
sig_id	integer	avg = ? +/- 0	[∞ ; -∞]	2358
sig_name	nominal	mode = unknown	ICMP PING *NIX (0), ICMP PING (0)	2358
sig_priority	integer	avg = ? +/- 0	[∞ ; -∞]	2358
sig_rev	integer	avg = ? +/- 0	[∞ ; -∞]	2358
sig_sid	integer	avg = ? +/- 0	[∞ ; -∞]	2358
sig_gid	integer	avg = ? +/- 0	[∞ ; -∞]	2358

Figure 48: TCP header Meta data

After running the basic process from Figure 22 we can see the *Meta Data* output in Figure 48. This shows us the available information ranges in the TCPHdr tables as well as the *Signature* tables. We can also see that the Signature tables have little impact on the TCP information, which means that using these to plot a graph will provide no additional useful information to the TCP data.

Looking at the output we can also see that the *Destination Port* has a range of over 65,000 possibilities and *Source port* has a range of nearly 5600 possibilities, which could make this a very valuable field when looking for the outliers. We can also see that the *tcp_urp* and *tcp_res* field have no values in them, which means they were never used and as such will also be of no use to us.

We can also see a sequence number field (*tcp_seq*) and an acknowledgment field (*tcp_ack*) which would be of little use even though they have a very large range, because we don't know the request number and the sequence number will be out because we are only using a sample of the data not entire continues strings (i.e. 1-2-3-4).

As an example, even though both the IPHdr and TCPHdr tables come from the same data collection event, run at the same as each other, the IPHdr table was populated for 19 days, while

the TCPHdr table was only populated for 16 days. This shows the input of the data being 3 days out from each other even though they were started at the same time. This is not that significant but it does show the difference in traffic usage over a network.

6.4.2: Process design setup

Again we can see in this process like with the previous one that we are using the same basic process and the same sample option, we have also again added the *Filter outliers* operator to filter out the outliers for us to view on their own.

The new table that we are importing into this process is the *TCPHdr* table which will replace the IP once from before. The ID's for each will remain the same (CID) and the joins will also remain the same (inner / outer)

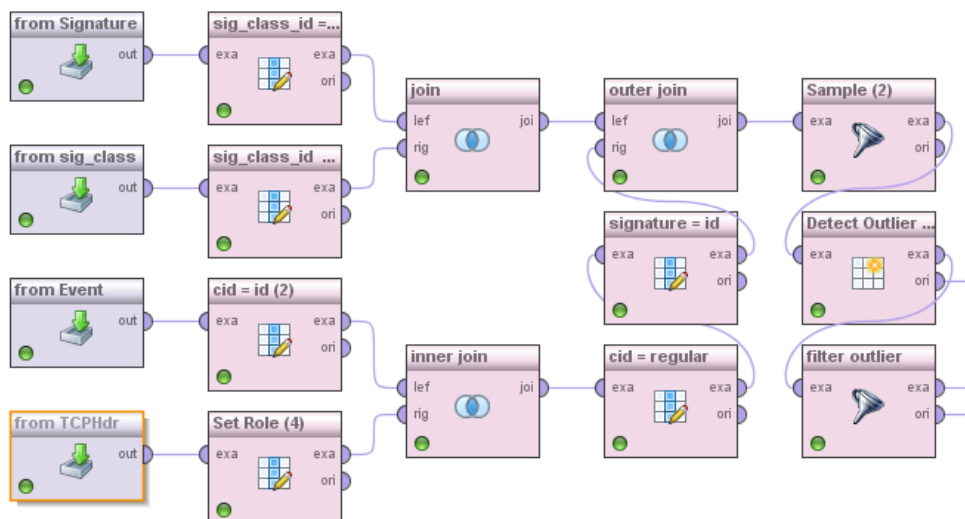


Figure 49: Process 2 – TCPHdr design

After testing with various configurations of the values that are available to the user, it has been settled on the final values for each of the operators used in this process are in Figure 50, though some of these might be altered to test for different graph outputs, the general values will be the same (only quantity would change or join type).

Setup Information		
Operator	Attribute	Value
Sample	Sample : Relative	0.06 (6%)
Join	Join Type	Inner / Inner / Outer
Set Role	ID – Event	CID
	ID – IPHdr	CID
	ID – Signature	Sig_class_id
	ID – Sig_class_name	Sig_class_id
	ID – event/tcp_hdr	signature
	Regular – event/tcp_hdr	CID
Detect Outliers	Distance Function	Euclidian Distance
	Number of outliers	10
	Number of neighbours	20
Filter Example	Condition class	Outlier = true

Figure 50: Process 2 setup values

6.4.3: Examine results

Looking at the results below, we can see that like with the IP table there are a lot of ‘?’ in the signature fields. This indicates that again there is no link between the tables, and as such these were not implemented in the finding of the outliers, however unlike with the IP table there are no signs of the signature fields having been used (see Figure 41 for reference to the signatures being read)

Row No.	signature	outlier	sid	tcp_sport	tcp_dport	tcp_seq	tcp_ack	tcp_off	tcp_res	tcp_flags	tcp_win	tcp_csum	tcp_urp	timestamp	cid	sig_class_name	sig_id	sig_name	sig
1	6	true	1	49296	139	4075368804	1502680604	8	0	24	65535	42162	0	18-Feb-2011 16:08:57 GMT	133	?	?	?	?
2	6	true	1	49319	139	1738510242	3914358016	8	0	24	65535	55964	0	18-Feb-2011 16:11:08 GMT	136	?	?	?	?
3	6	true	1	49873	139	1652870161	2213457798	8	0	24	65535	3038	0	18-Feb-2011 16:39:32 GMT	164	?	?	?	?
4	6	true	1	51583	139	2841548943	879648398	8	0	24	65535	50952	0	18-Feb-2011 17:38:31 GMT	225	?	?	?	?
5	6	true	1	51753	139	1078350426	3399557480	8	0	24	33304	785	0	18-Feb-2011 17:47:15 GMT	234	?	?	?	?
6	6	true	1	51757	139	198210395	2208667102	8	0	24	33304	58859	0	18-Feb-2011 17:47:15 GMT	236	?	?	?	?
7	6	true	1	51872	139	3892255141	2345772486	8	0	24	65535	29767	0	18-Feb-2011 17:56:00 GMT	243	?	?	?	?
8	6	true	1	52346	139	1991485704	3639104812	8	0	24	65535	13009	0	18-Feb-2011 18:26:35 GMT	275	?	?	?	?
9	6	true	1	52429	139	2343473154	299644385	8	0	24	65535	53887	0	18-Feb-2011 18:35:19 GMT	282	?	?	?	?
10	6	true	1	53907	139	3881775494	515815358	8	0	24	33304	26660	0	18-Feb-2011 19:36:29 GMT	341	?	?	?	?
11	6	false	1	54360	139	299703365	468479609	8	0	24	65535	14295	0	18-Feb-2011 20:09:15 GMT	361	?	?	?	?
12	6	false	1	54360	139	299703365	468479609	8	0	24	65535	13117	0	18-Feb-2011 20:09:15 GMT	362	?	?	?	?
13	6	false	1	54364	139	2232845989	2703403254	8	0	24	65535	38514	0	18-Feb-2011 20:09:15 GMT	363	?	?	?	?
14	6	false	1	54428	139	1221574449	3306872478	8	0	24	65535	5606	0	18-Feb-2011 21:03:52 GMT	369	?	?	?	?
15	6	false	1	54452	139	1057800756	4205361076	8	0	24	65535	33687	0	18-Feb-2011 21:06:03 GMT	373	?	?	?	?
16	6	false	1	54500	139	1184379863	2355728331	8	0	24	65535	30271	0	18-Feb-2011 21:10:25 GMT	376	?	?	?	?
17	6	false	1	54504	139	2011920519	2583971804	8	0	24	65535	49421	0	18-Feb-2011 21:10:25 GMT	378	?	?	?	?
18	6	false	1	54618	139	318384187	4123253147	8	0	24	65535	32825	0	18-Feb-2011 21:23:31 GMT	390	?	?	?	?
19	6	false	1	54868	139	3432437690	2255399785	8	0	24	65535	48962	0	18-Feb-2011 21:47:33 GMT	417	?	?	?	?
20	6	false	1	54923	139	2758609950	1280772796	8	0	24	65535	21197	0	18-Feb-2011 21:51:55 GMT	422	?	?	?	?
21	6	false	1	54958	139	2404589782	2127572785	8	0	24	65535	60436	0	18-Feb-2011 21:56:18 GMT	424	?	?	?	?
22	6	false	1	54985	139	4246041931	4100072206	8	0	24	65535	40392	0	18-Feb-2011 21:58:29 GMT	429	?	?	?	?
23	6	false	1	55047	139	1660022235	2663963181	8	0	24	65535	8838	0	18-Feb-2011 22:05:02 GMT	433	?	?	?	?
24	6	false	1	55102	139	646557882	2238062927	8	0	24	65535	12896	0	18-Feb-2011 22:11:35 GMT	438	?	?	?	?
25	6	false	1	55503	139	392210614	2030328403	8	0	24	65535	47145	0	18-Feb-2011 22:46:32 GMT	474	?	?	?	?
26	6	false	1	55554	139	174259769	2026379341	8	0	24	65535	32022	0	18-Feb-2011 22:53:05 GMT	478	?	?	?	?
27	6	false	1	55581	139	1735387967	461926066	8	0	24	65535	48419	0	18-Feb-2011 23:04:01 GMT	490	?	?	?	?

Figure 51: Detect TCP outliers

As with the IP and all future processes we can see the outliers having been detected, and are not shown by the new field called ‘Outliers’ as being TRUE. In Figure 52 we can see a closer view of the 10 outliers and the information about them. As was said before which we can see a bit more easily here, the *tcp_res* and *tcp_res* have no values and no bearing on the outcome of the process, but we keep them in anyway so that we have full header’s available for viewing.

signature	outlier	sid	tcp_sport	tcp_dport	tcp_seq	tcp_ack	tcp_off	tcp_res	tcp_flags	tcp_win	tcp_csum	tcp_urp	timestamp	cid
6	true	1	49296	139	4075368804	1502680604	8	0	24	65535	42162	0	18-Feb-2011 16:08:57 GMT	133
6	true	1	49319	139	1738510242	3914358016	8	0	24	65535	55964	0	18-Feb-2011 16:11:08 GMT	136
6	true	1	49873	139	1652870161	2213457788	8	0	24	65535	3038	0	18-Feb-2011 16:39:32 GMT	164
6	true	1	51583	139	2841548943	879648398	8	0	24	65535	50952	0	18-Feb-2011 17:38:31 GMT	225
6	true	1	51753	139	1078350426	3399557460	8	0	24	33304	785	0	18-Feb-2011 17:47:15 GMT	234
6	true	1	51757	139	198210395	2208667102	8	0	24	33304	58859	0	18-Feb-2011 17:47:15 GMT	236
6	true	1	51872	139	3892255141	2345772486	8	0	24	65535	29767	0	18-Feb-2011 17:56:00 GMT	243
6	true	1	52346	139	1991485704	3639104812	8	0	24	65535	13009	0	18-Feb-2011 18:26:35 GMT	275
6	true	1	52429	139	2343473154	299644385	8	0	24	65535	53887	0	18-Feb-2011 18:35:19 GMT	282
6	true	1	53907	139	3881775494	515815358	8	0	24	33304	26660	0	18-Feb-2011 19:36:29 GMT	341

Figure 52: Detect TCP outliers close view

We can also see that the *tcp_off* (data offset) is also the same not only in the outliers but also similar to the majority in the rest of the data set, this can also be of little importance, not because it is the same as the rest of the outliers but because there is little change in the rest of the data in this field.

6.4.4: Graph Results

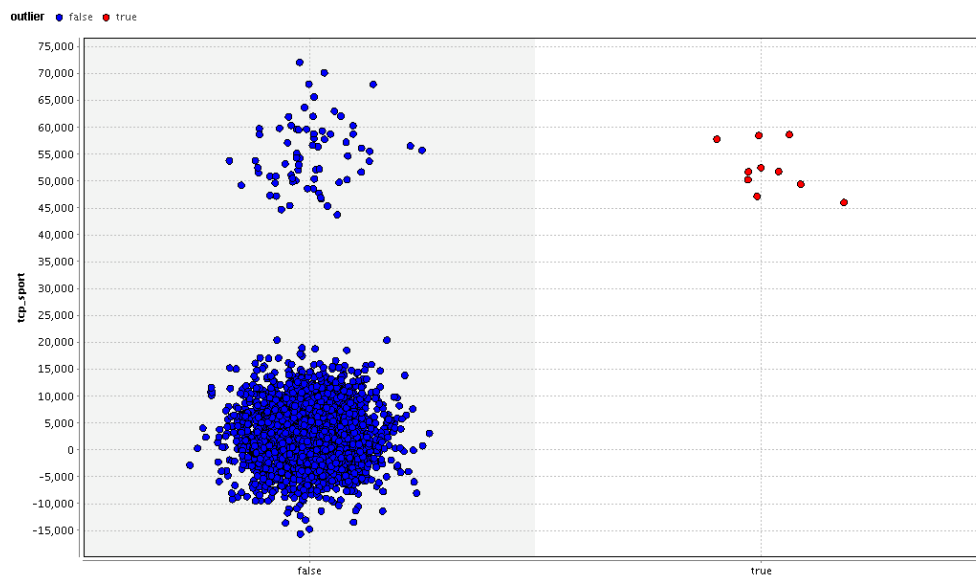


Figure 53: Outliers graph (x = outliers / y = source port)

Now that we have our data and we have the outliers found from that data, we can now see what they look like. Sticking with the *Scatter* diagram to illustrate the different clusters that are in the data, we can see that if we graph the *outliers* against the *source port* in Figure 53. We can see that there are a group of 2 clusters, even though the outliers are separate that is only because of the X axis that we have chosen.

The outliers that can be seen can also be seen to be in the port range of 49,000 to 53,000 which are in the range of the TCP ports that are used either friendly or hostile use. Ports 49,152 to 65,535 are classed as Dynamic/Private ports that are used by both TCP and UDP protocols.

We can also see how the clustering of these outliers are arranged by looking at Figure 54, where the graph is the data Offset (tcp_off) that we mentioned earlier against the packet sequence number (tcp_seq). Even though the sequence number may not be entirely of use in this case since we only have a sample of data over a short period of time and no way of telling what the previous or sequential sequence number packet might be, it does show that the data is in two clusters and that the outliers are in the smaller of the 2 as was mentioned in the previous image.

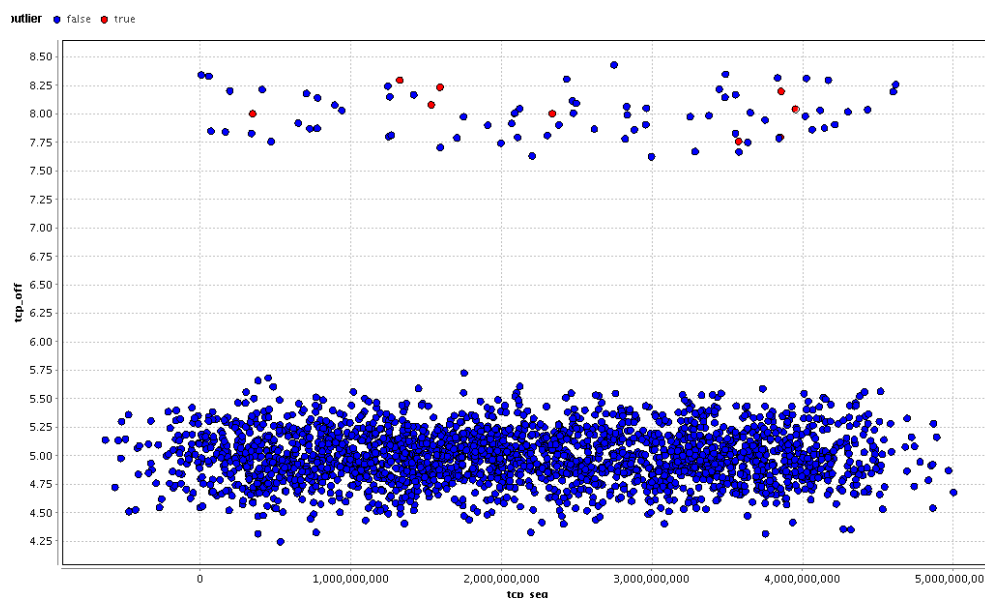


Figure 54: Outliers graph (x = sequence value / y = off value)

TCP 'off' = Many TCP/IP software stack implementations provide options to use hardware assistance to automatically compute the checksum in the network adapter prior to transmission onto the network or upon reception from the network for validation. (Wikipedia)

Now that we have looked at some of the fields there is one other that we should look at that might tell us something more about what is happening, that is the source 'v' destination ports (*tcp_sport* 'v' *tcp_dport*). This will show us when a packet comes into the network, where it is going to or if the packet is from the network where it wants to exit from. In Figure 55 we can see the produced results of graphing source against destination port. Again the graph isn't entirely dissimilar to previous ones showing what could be classed as three clusters, but with defiantly two distinct clusters.

What this is showing us, is that even though they are coming in through the higher ports which are in fact the TCP port range that are dynamically assigned, they are leaving through the lower end of the port range.

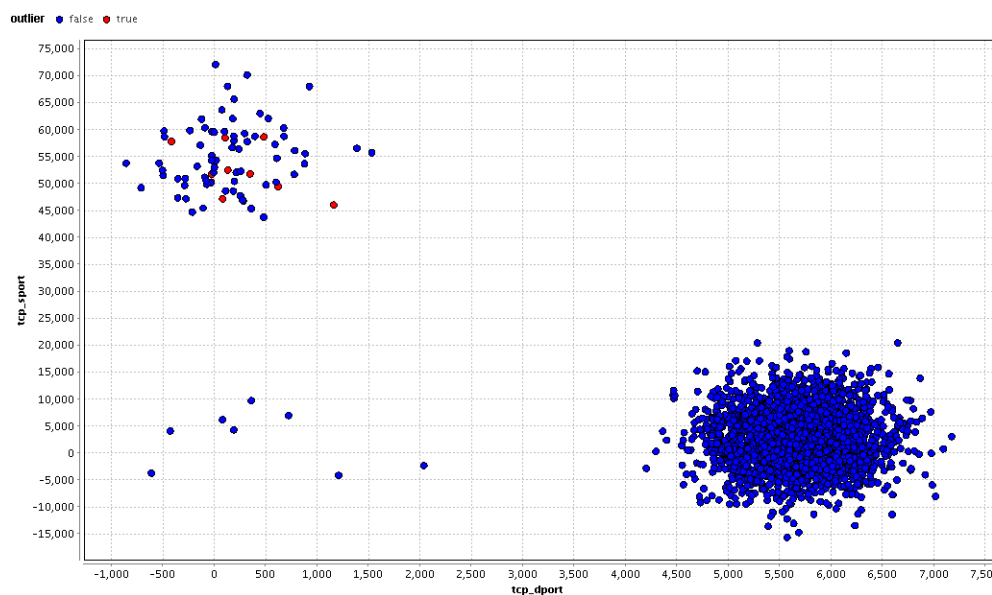


Figure 55: Outliers graph (x = destination port / y = source port)

The range of outbound ports that appear to be used are -500 to +2000, but if we look at the information in the data view above (see Figure 52) we can see that the destination port is actually port number 139, it is due to graphing that the X axis values have been weighted by the process, but we can see the true values from the data views.

6.4.5: Conclusion

After looking at the results and the graphs above we can see quite clearly that the range of outbound ports is from '49295' to '53907'. This port range is a standard TCP port range, but is not statically assigned but is instead dynamically assigned depending on what needs to use it. This means that it will remain closed until something tries to pass through it, which is when it will open up and remain open until transmission has finished.

6	true	1	49296	139	4075368804	1502680604	8	0	24	65535	42162	0	18-Feb-2011 16:08:57 GMT	133
6	true	1	49319	139	1738510242	3914358016	8	0	24	65535	55964	0	18-Feb-2011 16:11:08 GMT	136

Figure 56: Segment of data view (sport = 49319)

Looking at Figure 56 we can see a segment of the packet transmission with source port number '49319' and destination port number '139'. After doing some investigation, port '49319' throws up no specific dangers. However the port range is highly susceptible to Trojan use and since a Trojan is already on your system that means that it is an internal attack so we would need to remove this before any attack preventions becomes effective.

On the other side, destination port number '139' is used by a TCP/UDP NetBIOS service session.

"NetBIOS (Network Basic Input/Output System) is a program that allows applications on different computers to communicate within a local area network (LAN). It was created by IBM for its early PC Network, was adopted by Microsoft, and has since become a de facto industry standard." (Beal)

The NetBIOS sessions are generally safe, in that it is an application or action that the user has done to open it (i.e. internet chat, internet game) which requires the open port and transmission to safely

send and receive data. However there is also a number of ‘Worms’ that also use this port to gain access to your network.

Outbound scans if occurring in volume should be considered an indication of a possible worm infection on the source computer and should be investigated. (Binary Vision)

The main responsibility is the TCP protocol is to handle traffic packets, and as such will generally be used as an network internal protocol, though the protocol will be used on each network (my network, internet network, destination network), as such the TCPHdr table will not tell us where the attack originated from or where it is headed, but it will tell us how it plans to get there (i.e. what port).

And using this port information we can then open or close ports or monitor them based on what we can learn from here. Is the IDS detects someone coming in on a port, it can then close it early or enact some preventative measures.

6.5: Process 3 – Detect outliers (UDP)

IPHdr process information	
Number of data entries	65
Tables used	UDPHdr, Event, Signature, Sig_class
Outlier neighbours	10
Number of Outlier	10

Figure 57: UDP process information

6.5.1: Introduction

In this process we deal with the *UDPHdr* table (UDP Header) trying to identify its outliers. This is one of the smallest tables in the data set in both number of entries at a total count of 65, and the number of fields being only 5. However this is still quite an important since it is the primary protocol of application data transmission.

UDP (User Datagram Protocol) is a simple OSI transport layer protocol for client/server network applications based on Internet Protocol (IP). UDP is the main alternative to TCP and one of the oldest network protocols in existence. UDP network traffic is organized in the form of datagrams. A datagram comprises one message unit. The first eight (8) bytes of a datagram contain header information and the remaining bytes contain message data. (Mitchel)

Since there are so few entries in this dataset we are able to use the entire set, rather than samples as we have had to do in previous processes. This will allow us to view the data in its entirety and should help us with a better understanding as to what has been collected.

bits	0 – 15	16 – 31
0	Source Port Number	Destination Port Number
32	Length	Checksum
64	Data	

Figure 58: UDP header (Wikimedia)

Looking at each of the fields in the DUP header in Figure 59 we can see again, like with the TCP that we are dealing with *Port's* rather than IP addresses. As well as the length of the “datagram” and its checksum value. However the checksum value is not very relevant since it is not verifiable due to that we cannot see into the *Data* field (it is encrypted), though we can still test with it to see what results may arise and if they differ to other tests.

Looking at Figure 59 we can see the fields that are of primary interest here are the *Source* and *Destination* port numbers or ranges as well as the *length*. This is because they have a high range value and because there are not really any others in the header that are available (only 5 fields available from the header).

Name	Type	Statistics	Range
signature	integer	avg = 8.095 +/- 2.422	[1.000 ; 9.000]
outlier	binominal	mode = false (64), least = true (10)	false (64), true (10)
sid	integer	avg = 1 +/- 0	[1.000 ; 1.000]
udp_sport	integer	avg = 55447.631 +/- 8477.927	[53.000 ; 65373.000]
udp_dport	integer	avg = 2663.569 +/- 6156.092	[1900.000 ; 51532.000]
udp_len	integer	avg = 104.231 +/- 6.202	[55.000 ; 105.000]
udp_csum	integer	avg = 23389.846 +/- 5489.023	[2463.000 ; 30469.000]
timestamp	date_time	length = 11 days	[22-Feb-2011 19:06:31 GMT ; 05-Mar-
cid	integer	avg = 27489.538 +/- 11476.768	[4831.000 ; 36178.000]
sig_class_name	nominal	mode = bad-unknown (4), least = net	bad-unknown (4), misc-activity (4), ne
sig_id	integer	avg = 5 +/- 2.739	[1.000 ; 9.000]
sig_name	nominal	mode = Reset outside window (1), le	ICMP PING *NIX (1), ICMP PING (1), I
sig_priority	integer	avg = 2.556 +/- 0.527	[2.000 ; 3.000]
sig_rev	integer	avg = 4.778 +/- 3.114	[1.000 ; 9.000]
sig_sid	integer	avg = 414.333 +/- 589.285	[5.000 ; 1917.000]
sig_gid	integer	avg = 43.667 +/- 64	[1.000 ; 129.000]

Figure 59: UDP Meta data

Looking at the Meta data in Figure 59 we also see that SNORT has provided us with a *timestamp* field of the individual packets and datagrams as they pass though the network. This is another field that could be of interest in deciphering what is of interest to us in the next stage.

The source port (udp_sport) has a range of over 65,000 possibilities while the destination port (udp_dport) has a range of nearly 49,000. So indicates that the range of packet issuing ports is higher than that of the receiving port, though could indicate that port scanning has taken place with the sending trying to find an accessible port and having to scan the ranges of what is available.

The datagram length may or may not be useful, with a range of 50, since we cannot see how long it should be only what we are being told it is. However this may be of sum use in that the smaller the packet the less likely it will have useful information held in it and the more likely it will be that it is a scanning packet or something that is testing the water to see what happens if it enters. We can still test for this during the processes.

6.5.2: Process design setup

Again the process setup is very similar to the other processes that we will run and have run with only one change in the operator's having the *sample* operator not in use. Looking at the process diagram in Figure 60 we can see that the sample operator has been greyed out, indicating that it is not in use this time, this is simply because we have such few entries in the UDP table that we do not need to filter them and filtering would produce fewer entries than any of the other sampled tables.

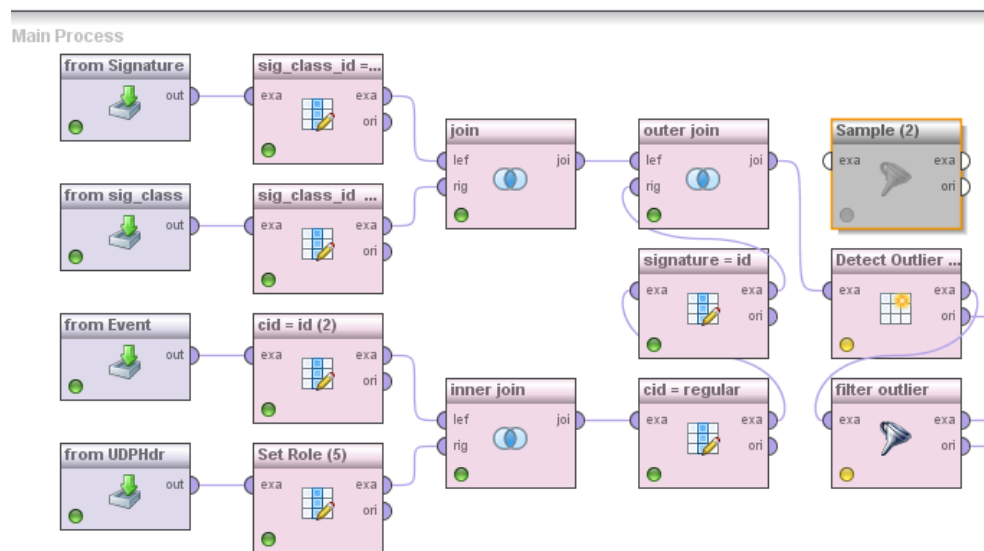


Figure 60: Process 3 - UDPHdr design

Other changes in the process design are the number of outliers has been reduced to 10 and the number of neighbours has also been reduced to 10, again this is because of the lack of entries compared to other tables. However the ID's will remain the same as previous processed as will the filter remain the same, and the join operators.

Setup Information		
Operator	Attribute	Value
Join	Join Type	Inner / Inner / Outer
Set Role	ID – Event	CID
	ID – IPHdr	CID
	ID – Signature	Sig_class_id
	ID – Sig_class_name	Sig_class_id
	ID – event/ip_hdr	signature
	Regular – event/ip_hdr	CID
Detect Outliers	Distance Function	Euclidian Distance
	Number of outliers	10
	Number of neighbours	10
Filter Example	Condition class	Outlier = true

Figure 61: Process 3 setup values

As was mentioned before the *Sample* operator is not used in this process because the data set is simply too small to need it. Because of this we are able to sue the entire data set which may help to better understand what is happening in it.

6.5.3: Examining results

Once we run the process we can see the results of the process in the *data view* window in Figure 62. What this shows us is that the top 10 have been classified as outliers. It also shows that again like with previous processes, the *Signature* has not been applied to the data, and as such is depicted as a series of '?' implying that there is no value in that location. What we can do in this case is ignore the signature tables entirely from the data and concentrate on the UDP field only.

outlier	sid	udp_sport	udp_dport	udp_len	udp_csum	timestamp	cid	sig_class_...	sig_id	sig_n
true	1	53	51532	55	2463	22-Feb-2011	4831	?	?	?
true	1	63203	1900	105	16827	25-Feb-2011	12294	?	?	?
true	1	57417	1900	105	22613	25-Feb-2011	12296	?	?	?
true	1	57425	1900	105	22605	25-Feb-2011	12298	?	?	?
true	1	54109	1900	105	25921	25-Feb-2011	12300	?	?	?
true	1	61787	1900	105	18243	25-Feb-2011	12302	?	?	?
true	1	56923	1900	105	23107	25-Feb-2011	12304	?	?	?
true	1	61148	1900	105	18882	25-Feb-2011	12306	?	?	?
true	1	61156	1900	105	18874	25-Feb-2011	12308	?	?	?
true	1	56702	1900	105	23328	25-Feb-2011	12310	?	?	?
false	1	60536	1900	105	19494	25-Feb-2011	12312	?	?	?
false	1	58352	1900	105	21678	25-Feb-2011	12314	?	?	?
false	1	58360	1900	105	21670	25-Feb-2011	12316	?	?	?
false	1	50310	1900	105	29720	25-Feb-2011	12318	?	?	?
false	1	50318	1900	105	29712	25-Feb-2011	12320	?	?	?
false	1	50326	1900	105	29704	25-Feb-2011	12322	?	?	?
false	1	50334	1900	105	29696	25-Feb-2011	12324	?	?	?
false	1	50342	1900	105	29688	25-Feb-2011	12326	?	?	?
false	1	50350	1900	105	29680	25-Feb-2011	12328	?	?	?
false	1	50358	1900	105	29672	25-Feb-2011	12330	?	?	?
false	1	64333	1900	105	15697	26-Feb-2011	16606	?	?	?
false	1	61321	1900	105	18709	26-Feb-2011	16608	?	?	?
false	1	63833	1900	105	16197	26-Feb-2011	16610	?	?	?
false	1	51784	1900	105	28246	26-Feb-2011	16612	?	?	?
false	1	63001	1900	105	17029	05-Mar-2011	36096	?	?	?
false	1	61793	1900	105	18237	05-Mar-2011	36098	?	?	?
false	1	56385	1900	105	23645	05-Mar-2011	36100	?	?	?
false	1	55176	1900	105	14864	05-Mar-2011	36102	?	?	?

Figure 62: Detect UDP outliers

Glancing over the outliers we see that there is little difference in a lot of the data that is available, the UDP Length (*udp_len*) is 105 across the board, and the destination port (*udp_dport*) is equally straight at 1900 across all records. However the checksum (*udp_csum*) changes by a large variant across the table as does the source port (*udp_sport*). There is however one entire that stands out at a glance, the very first one which we can see more closely in Figure 63, we will look at again later.

8	true	1	53	51532	55	2463	22-Feb-2011	4831
9	true	1	63203	1900	105	16827	25-Feb-2011	12294
9	true	1	57417	1900	105	22613	25-Feb-2011	12296
9	true	1	57425	1900	105	22605	25-Feb-2011	12298
9	true	1	54109	1900	105	25921	25-Feb-2011	12300
9	true	1	61787	1900	105	18243	25-Feb-2011	12302
9	true	1	56923	1900	105	23107	25-Feb-2011	12304
9	true	1	61148	1900	105	18882	25-Feb-2011	12306
9	true	1	61156	1900	105	18874	25-Feb-2011	12308
9	true	1	56702	1900	105	23328	25-Feb-2011	12310
9	false	1	60536	1900	105	19494	25-Feb-2011	12312

Figure 63: Detect UDP outliers close view

Looking at the data in Figure 63, we can pretty much guess that for 9 of the 10 entries will be of or about the same thing; this can indicate that though it is away from the normal flow of things, it is not out of the overall ordinary on that particular network.

6.5.4: Graphing results

Looking at the outliers in the following graphs should help us identify what the data is doing and where it is going.

From Figure 64 we can clearly see the outliers apart from the rest of the standard data, however because this is an outliers graph we can also tell that the outliers fall nested within the rest of the data. This however does not mean that they are the same as the rest of it just that they are within the same ranges. Looking at it a little more closely we can also see that there is one particular entire that is located quite a distance from the rest (top right hand corner). This is in fact the exceptionally different entire that we saw in the previous figure (see Figure 34) indicating a drastic change in the flow of data. This graph is done on outliers 'v' destination port.

Looking at a different view of the data in Figure 65, we can again see that single entire that is drastically different from the rest. Not only does this show that that entire has a different destination port out of the range boundaries of the others but it also has a much smaller checksum to the others. Looking back at the data view in Figure 63, we can see that the check sum vale of that entire is 55, nearly half that of the others in the view. This could indicate some form of scanning being done on the network at that time.

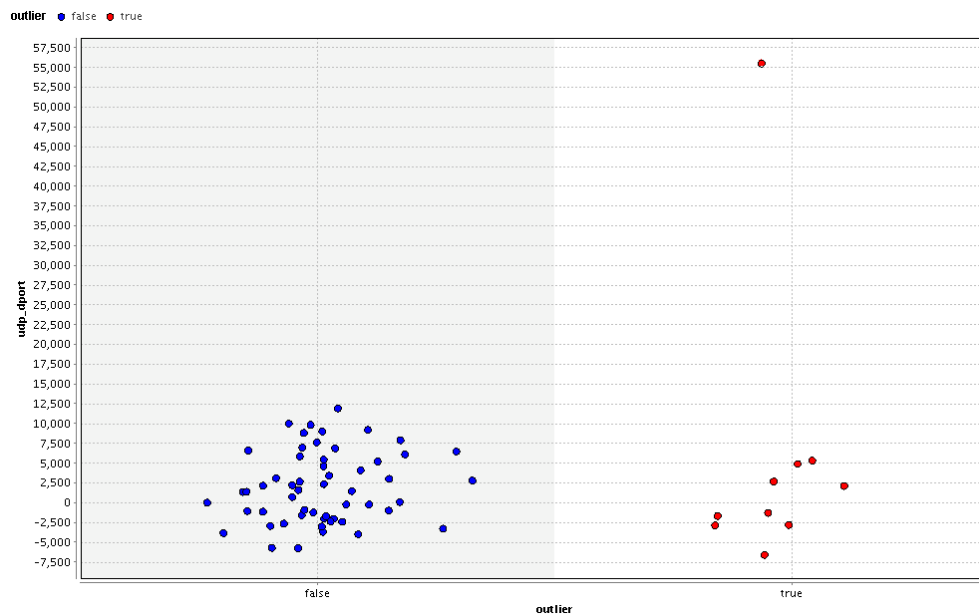


Figure 64: Outlier graph (x = outlier / y = udp_dport)

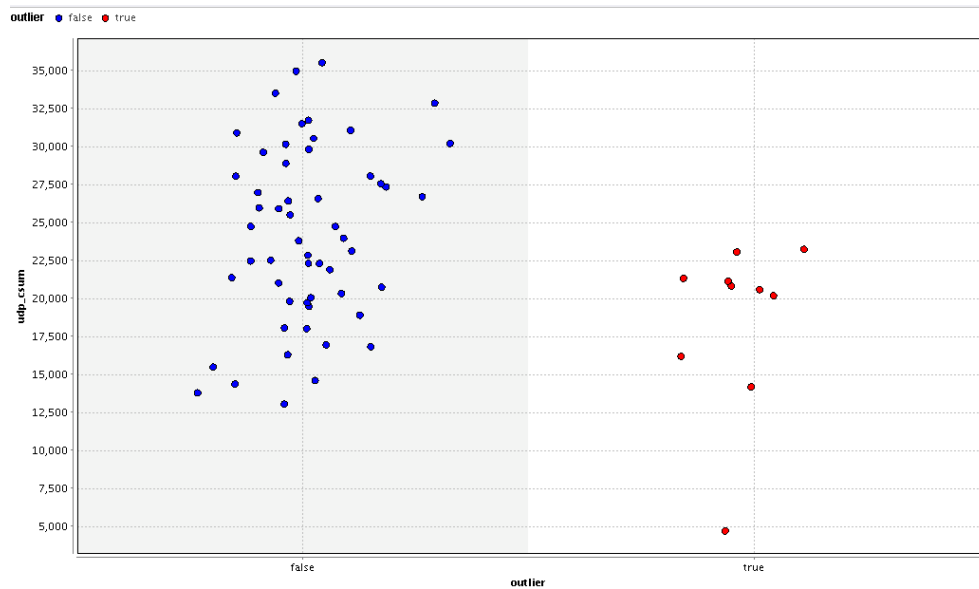


Figure 65: Outliers graph (x = outliers / y = udp_csum)

Looking at the outliers on their own (see Figure 66) we see only 9 points that have been graphed on the source 'v' destination port ranges. This is because there is actually 10 there, but due to two of them being so similar they are nearly on top of each other. And because that single outlier is so different to the rest we are unable to expand the graph area in case of losing some visibility on other points.

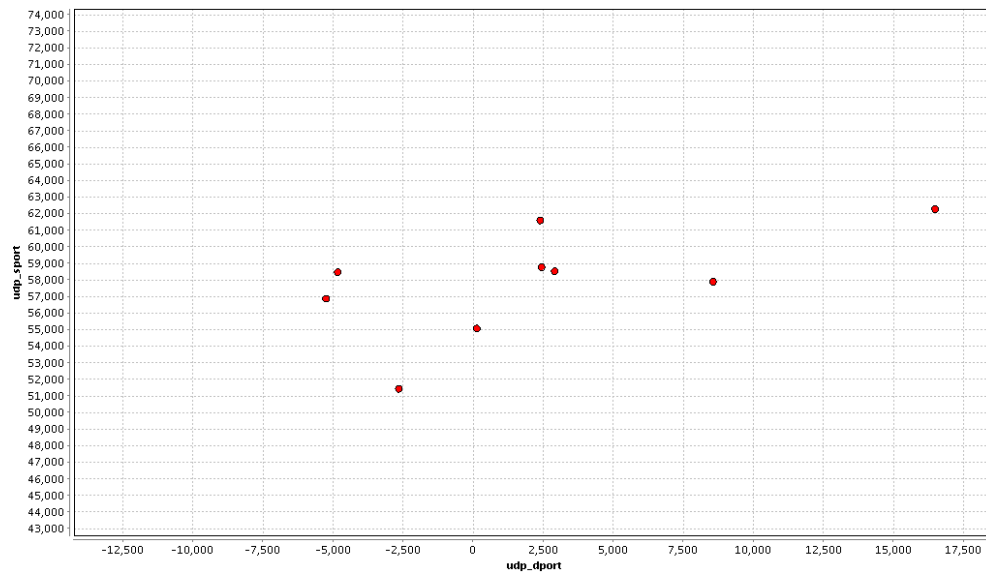


Figure 66: Outliers graph (x = udp_dport / y = sport)

Looking at the graphs we can clearly see the layout of the different outliers that we have available, this can illustrate to us that because something is out of the norm doesn't necessarily mean that it will be out of the overall group. It simply means that it is not following the current trend of flow.

6.5.5: Conclusion

Looking at the two distinct entries that are in our outliers we can see that the source port of each are '53' and '63203' respectively.

outlier	sid	udp_sport	udp_dport	udp_len	udp_csum	timestamp	cid
true	1	53	51532	55	2463	22-Feb-2011	4831
true	1	63203	1900	105	16827	25-Feb-2011	12294

Figure 67: Stand out entire

Thought it is source port '53' and destination port '1900' that are of interest, this is because they deal with scanning and DNS services, while the other ports are dynamic and will just deal with the return of the information.

Port 53 details:	
Protocol:	TCP & UDP
IAMA status:	Official
Range:	Well-known
Traffic:	inbound, outbound, both
Notification:	N/A
Related Ports:	21211, 1025

Figure 68: Port 53

Port 53 is DNS service port that deals with both UDP and TCP protocols, but depending on the protocol that is being used with it will change the way that the port behaves.

The UDP is primarily responsible for individual lookups while the TCP covers the zone transfers involving record databases of full names. (Uniblue)

This port also has a problem in that it is highly susceptible to attacks Worms and Trojans. Once the Worm or Trojan is in the network is can then contact its originating location and request or send information back again, this can also include *Keyloggers* which can monitor your key presses and record usernames and passwords or even credit card numbers.

DNS reveals a security gap as zone transfers may expose the whole map of networks making it prone to attackers. (Uniblue)

The DNS permits transmissions of Datagrams through network computers, and is primarily utilized involving ISP's²⁷ and unmanaged network such as home network which will allow for 'AnyCast' or Broadcast of messages to cause infection on the network.

²⁷ ISP = Internet Service Provider

Services or applications using this port: Domain name server
MALICIOUS SERVICES / APPLICATIONS: ADMworm

Figure 69: Port 53 access

Port 1900 details:	
Protocol:	TCP & UDP
IAMA status:	Conflict
Range:	Registered
Traffic:	inbound, outbound, both
Notification:	True
Related Ports:	09335

Figure 70: Port 1900

Port 1900 is used by a computer service or application that is sending information from a request made through the port 53. This is a SSDP (Simple Service Discovery Protocol) that was developed by Hewlett-Packard (HP) and Microsoft during the draft process of the Internet. This protocol is designed to allow network clients the ability to discover different network services.

The Microsoft SSDP service is officially registered with IANA as the protocol running on the network port 1900. This service is essentially associated with the automatic enabling of the discovery feature related with plug and play devices. This computer port is used to transmit data to identify the connection of UPnP capable devices to the system or network. (Uniblue)

Because this is a return port and not a receiving port like with '53' this will unicast²⁸ rather than broadcast or multicast, which means that it will have an associated address assigned to it. This port can be a high risk one since any attacker that enters the network through the receiver port will more than likely be able to return through this one.

²⁸ Send messages to 1 single address.

Services or applications using this port: SSDP for UPnP (Universal Plug & Play) , Windows Alerter MALICIOUS SERVICES / APPLICATIONS: Remote Access Trojan

Figure 71: Port 1900 access

```
02/25-00:39:02.591428  [**] [1:1917:9] SCAN UPnP service discover attempt [**]
[Classification:  Detection of a Network Scan] [Priority:  3] {UDP}
192.168.1.112:51532 -> 192.168.1.106:1900

02/25-00:39:32.900240  [**] [1:1917:9] SCAN UPnP service discover attempt [**]
[Classification:  Detection of a Network Scan] [Priority:  3] {UDP}
192.168.1.112:61148 -> 192.168.1.106:1900
```

If we look through our Alert files and records we can find the Scan that was done on the network. A *UPnP service discovery* scan has been done on the network from internal, the source host was at *192.168.1.112* and was scanning the IDS machine at *192.168.1.106*. Since this machine (IDS machine) doesn't actually run anything and everything that might need access to the internet or network is turned off, this scan was not originally sent from the IDs machine. This tells us that there is either someone on the network that could be classed as hostile or someone on the network that is trying to build a view of the network design.

6.6: Process 4 – Detect outliers (ICMPHdr)

IPHdr process information	
Number of data entries	280
Tables used	ICMPHdr, Event, Signature, Sig_class
Outlier neighbours	10
Number of Outlier	10

Figure 72: ICMPHdr process information

6.6.1: Introduction

This is the final header table that we can look at. The ICMPHdr table deals with ICMP messages or Ping messages. The ICMP its self may not be an actually attack, but it will be the lead up to an attack. This is because the “ping” can tell someone if that location is available or unreachable.

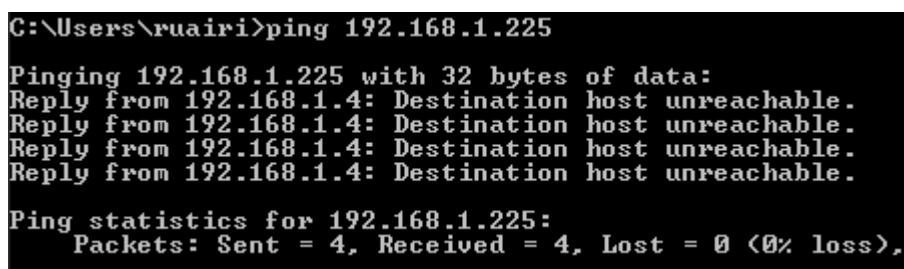
```
C:\Users\ruairi>ping www.google.com

Pinging www.l.google.com [209.85.143.104] with 32 bytes of data:
Reply from 209.85.143.104: bytes=32 time=27ms TTL=53
Reply from 209.85.143.104: bytes=32 time=29ms TTL=53
Reply from 209.85.143.104: bytes=32 time=29ms TTL=53
Reply from 209.85.143.104: bytes=32 time=28ms TTL=53

Ping statistics for 209.85.143.104:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 27ms, Maximum = 29ms, Average = 28ms
```

Figure 73: ICMP ping message, host reached

The ping is one of the simplest actions that can be done a computer, and it one that everyone can do. The action of a ping or the more details trace-root (*tracert*) is in fact a tool that is commonly used by IT and Network technicians to check the connectivity of a network connection, however this ‘tool’ is also used by other less desirable people as the beginning of an possible attack on an as of yet unknown destination.



```
C:\Users\ruairi>ping 192.168.1.225

Pinging 192.168.1.225 with 32 bytes of data:
Reply from 192.168.1.4: Destination host unreachable.
Reply from 192.168.1.4: Destination host unreachable.
Reply from 192.168.1.4: Destination host unreachable.
Reply from 192.168.1.4: Destination host unreachable.

Ping statistics for 192.168.1.225:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
```

Figure 74: ICMP ping message, host unreachable

In itself the ping is not a malicious act, but because it will tell us if the destination is reachable or not. Looking at Figure 73 we can see the ping being used successfully by ping of the address *www.google.com* which is then translated into an IP address to find the location. However if we look at Figure 74 we can see an invalid IP address being pinged and the message returns as ‘*host unreachable*’, this can either mean that the host has their *ICMP ECHO*²⁹ service turned off which means that they location will be unattainable or that host address is not in existence at this time in which case they will still be unattainable.

²⁹ ICMP ECHO = A response to an ICMP request

```

C:\Users\ruairi>tracert www.google.com

Tracing route to www.l.google.com [209.85.143.99]
over a maximum of 30 hops:

  1      1 ms      1 ms      2 ms  192.168.1.254
  2     28 ms     26 ms     26 ms  b-ras2.hlp.dublin.eircom.net [159.134.155.101]
  3     27 ms     28 ms      *    ge-6-1-6.pe1.hlp.hlp-srl.eircom.net [86.43.246.3]
 4     30 ms     26 ms     26 ms  tenge-6-1-1.core2.srl.core.eircom.net [86.43.253.137]
 5     27 ms     31 ms     27 ms  tenge-3-2-1.core2.dbn.core.eircom.net [86.43.252.41]
 6     28 ms     27 ms     27 ms  tenge-1-2-1.pe2.6cr.6cr-6cr.eircom.net [86.43.255.98]
 7     62 ms     27 ms     26 ms  ge4-0-0.peering1.tcy.dublin.eircom.net [86.43.244.133]
 8     30 ms     29 ms     27 ms  74.125.51.185
 9     27 ms     43 ms     27 ms  209.85.252.162
10     29 ms     53 ms     28 ms  209.85.253.201
11     31 ms     39 ms     49 ms  216.239.47.26
12     29 ms     28 ms     29 ms  dy-in-f99.1e100.net [209.85.143.99]

Trace complete.

```

Figure 75: Trace root ICMP

But the standard ping is not the only tool that can be used. As was mentioned already is another one called the trace route. What this will do is show all the ‘hops’³⁰ that are encountered along the route to the destination. At each point the address is shown, and using this address we could then try to penetrate a different location.

Bits	0-7	8-15	16-23	24-31
0	Type	Code	Checksum	
32	Rest of Header			

Figure 76: ICMP header contents (Wikimedia)

Looking at the make-up of the header (see Figure 76) we can see that again like with the UDP header there are very few fields of information available. Of particular interest in the header would be the *Type* and the *Code* fields.

³⁰ A HOP is 1 step along the route

The *Type* field has 255 possibilities, though only about 17 of these are actually used, the other 238 are either reserved or not in use. After this each of the *Type* fields will also have an associated *code* field, which will tell us more information about it. If we look at an example of this in Figure 77 we can see a Type of '3' with the associated codes that can go with it along with the meaning for each of these codes.

3 - Destination Unreachable ^[4]	0	Destination network unreachable
	1	Destination host unreachable
	2	Destination protocol unreachable
	3	Destination port unreachable
	4	Fragmentation required, and DF flag set
	5	Source route failed
	6	Destination network unknown
	7	Destination host unknown
	8	Source host isolated
	9	Network administratively prohibited
	10	Host administratively prohibited
	11	Network unreachable for TOS
	12	Host unreachable for TOS
	13	Communication administratively prohibited

Figure 77: ICMP type/code example, Type no. 3 (Wikimedia)

After we run the basic process mentioned back in Figure 22 we can see the Meta data output screen in Figure 78. What this shows us the ranges that our data has in relation to the *Type / Code* tables. What we have is an *icmp_type* range of '0-8' and an *icmp_code* range of '0', which tells us that in each of the types the code will be the first one in the list of code '0'. In this case the range of '0' does not mean that there is no information available, but rather the opposite that there is information there.

Name	Type	Statistics	Range	Missings
signature	integer	avg = 1.900 +/- 0.945	[1.000 ; 4.000]	0
outlier	binominal	mode = false (270), least = true (10)	false (270), true (10)	0
sid	integer	avg = 1 +/- 0	[1.000 ; 1.000]	0
icmp_type	integer	avg = 7.200 +/- 2.404	[0.000 ; 8.000]	0
icmp_code	integer	avg = 0 +/- 0	[0.000 ; 0.000]	0
icmp_csum	integer	avg = 13930.479 +/- 25052.263	[26.000 ; 65415.000]	0
icmp_id	integer	avg = 34309 +/- 0	[34309.000 ; 34309.000]	0
icmp_seq	integer	avg = 13.500 +/- 8.092	[0.000 ; 27.000]	0
timestamp	date_time	length = 0 days	[15-Feb-2011 19:59:30 GMT ; 15-Feb-2011 19:59:30 GMT]	0
cid	integer	avg = 55.900 +/- 32.383	[1.000 ; 112.000]	0
sig_class_name	nominal	mode = misc-activity (112), least = n	bad-unknown (112), misc-activity (112)	28
sig_id	integer	avg = 5 +/- 2.587	[1.000 ; 9.000]	28
sig_name	nominal	mode = ICMP PING *NIX (28), least =	ICMP PING *NIX (28), ICMP PING (28)	28
sig_priority	integer	avg = 2.556 +/- 0.498	[2.000 ; 3.000]	28
sig_rev	integer	avg = 4.778 +/- 2.941	[1.000 ; 9.000]	28
sig_sid	integer	avg = 414.333 +/- 556.888	[5.000 ; 1917.000]	28
sig_gid	integer	avg = 43.667 +/- 60.460	[1.000 ; 129.000]	28

Figure 78: ICMP header Meta data

Looking again at the Meta data in Figure 78 we can also see that the *signature* data has also been applied to the ICMP data. However in this case rather than previous cases, the classification is a lot more prevalent and will in fact tell us what each outlier is when it is found. Other fields that may be of interest are again the *Timestamp* and possible the *icmp_seq* value.

6.6.2: Process design setup

As with the previous design (UDP) we again have no need for the *Sample* (see Figure 79) operator since there are only 280 data entries. Because of the lower number of entries we are also limited in the number of outliers that we can request; in this case we will be using the same attributes as the UDP (see Figure 80) process and asking for 100 outliers with 10 neighbours.

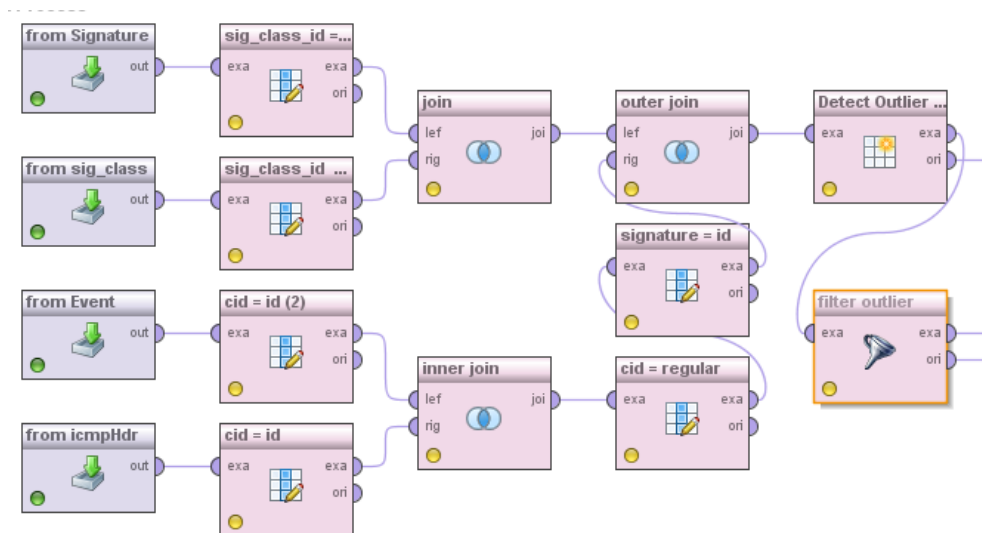


Figure 79: Process 4 - ICMPHdr design

Once the outliers have been found we can then make things a little easier by filtering them out and displaying only the outliers. This will help us to identify what they are, while ignoring the clustered more similar data.

Setup Information		
Operator	Attribute	Value
Join	Join Type	Inner / Inner / Outer
Set Role	ID – Event	CID
	ID – IPHdr	CID
	ID – Signature	Sig_class_id
	ID – Sig_class_name	Sig_class_id
	ID – event/ip_hdr	signature
	Regular – event/ip_hdr	CID
Detect Outliers	Distance Function	Euclidian Distance
	Number of outliers	10
	Number of neighbours	10
Filter Example	Condition class	Outlier = true

Figure 80: Process 4 setup values

6.6.3: Examine results

Once we have our outliers found and extracted from the rest of the data set, we can look more closely at them. Looking at the output we can see that the ICMP entries had been classified through the *sig_id* field in the signature data.

signature	outlier	sid	icmp_type	icmp_code	icmp_csum	icmp_id	icmp_seq	timestamp	cid	sig_class_	sig_id	sig_name	sig_priority	sig_rev	sig_sid
1	true	1	8	0	2981	34309	0	15-Feb-2011 1	misc-activity	1	ICMP PING *NIX		3	7	366
1	true	1	8	0	2981	34309	0	15-Feb-2011 1	misc-activity	2	ICMP PING		3	5	384
1	true	1	8	0	2981	34309	0	15-Feb-2011 1	misc-activity	3	ICMP PING BSDtype		3	6	368
1	true	1	8	0	2981	34309	0	15-Feb-2011 1	misc-activity	4	ICMP Echo Reply		3	5	408
2	true	1	8	0	2981	34309	0	15-Feb-2011 2	bad-unknown	5	Reset outside window		2	1	15
2	true	1	8	0	2981	34309	0	15-Feb-2011 2	bad-unknown	6	Consecutive TCP small segments exceeding threshold		2	1	12
2	true	1	8	0	2981	34309	0	15-Feb-2011 2	bad-unknown	7	Bad segment, adjusted size <= 0		2	1	5
2	true	1	8	0	2981	34309	0	15-Feb-2011 2	bad-unknown	8	DNS SPOOF query response with TTL of 1 min. and no authority		2	8	254
3	true	1	8	0	2981	34309	0	15-Feb-2011 3	network-sca	9	SCAN UPnP service discover attempt		3	9	1917
4	true	1	0	0	5029	34309	0	15-Feb-2011 4	?	?	?		?	?	?

Figure 81: Detect ICMP outliers

Thought this might seems a bit linier what it actually shows is an entire ping cycle (ICMP request, ICMP ECHO ...) and what this actually is, is an ICMP attack with DNS spoof. What defines these as outliers and possible threats is the abnormal sized checksum value (*icmp_csum*). This number means little to us as people but if the machine it can mean the difference between good or bad data.

sig_class_...	sig_id	sig_name
misc-activity	1	ICMP PING *NIX
misc-activity	2	ICMP PING
misc-activity	3	ICMP PING BSDtype
misc-activity	4	ICMP Echo Reply
bad-unknown	5	Reset outside window
bad-unknown	6	Consecutive TCP small segments exceeding threshold
bad-unknown	7	Bad segment, adjusted size <= 0
bad-unknown	8	DNS SPOOF query response with TTL of 1 min. and no authority
network-sca	9	SCAN UPnP service discover attempt
?	?	?

Figure 82: ICMP attack window

If we look at Figure 83 we can see where the checksum comes from. Initially we have our data, which is then passed through the *checksum function*, and a value is produced. The idea behind the checksum value is that if the data is tampered with, the value that we have and the value of the data checked again on receipt will be different. This will tell the recipient that the data is not original and as such may be harmful.

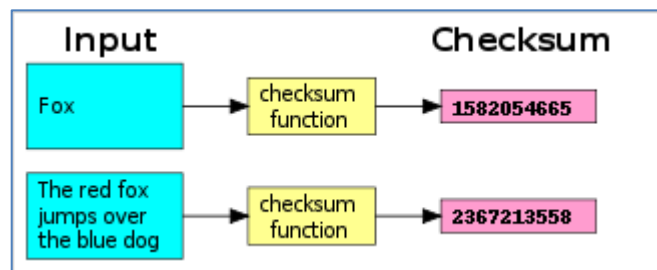


Figure 83: Checksum function

This can be a vital action for things like Personal details or passwords, but is equally important to make sure that you have not become the focus of a ‘*man-in-the-middle*³¹’ attack.

6.6.4: Graphing results

Now that we have looked at our data we can look at the graphs that represent what we may or may not know. In Figure 84 we can clearly see the 3 categories (*sig_class_name*) of data that are in the ICMP table. What this tells us is that the majority of the data that has been classified is a mix of ‘Miscellaneous activity’ (*misc-activity*) and bad or unknown sectors (*bad-unknown*). This can tell us that there may be dangerous activity in this but it may also be normal activity that has been abnormal this time.

Of the three classifications that are available, the network scan classification (*network-scan*) is of most significant. However this may also be a normal activity since other computers on the network may need to find other devices, or the machine in question may have sent out a request and the response needs to find an entry point into it to make a valid response.

³¹ Man-in-the-middle = where your data is intercepted and replaced with something else more harmful

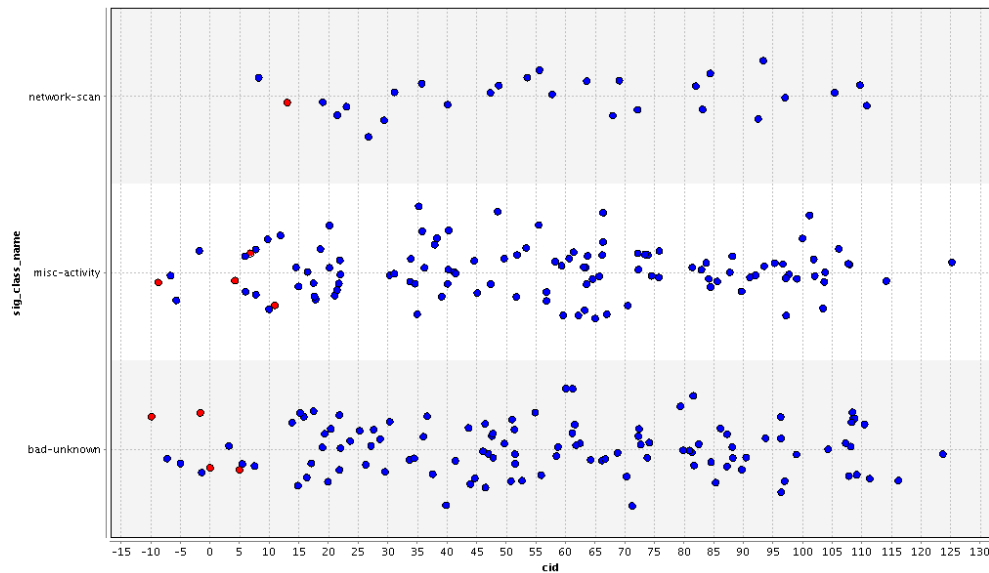


Figure 84: Outliers graph (x = CID / y = sig_class_name)

Looking at this graph however does reveal that all the outliers have happened in the first few events of the scanning and recording process of the IDS. Again with the *misc-activity* and *bad-unknown* this may have caused some irregular activity with the machine connecting to the network and building its routing tables, but the network-scan should not have happened since this usually deals with port activity.

If we take a slightly different stance on the graph and look at a slightly different one (see Figure 85) we can see a more detailed classification using the signature name (*sig_name*) rather than the class, this gives a better understanding of what is going on. Again we can see that the outlying events all seemed to happen during the initial stages of the IDs setup, but now we can see that the events are more accurately classified.

Scanning over the graph we can see that the most significant classifications would be the 'Bad segment size', 'UPnP Scan' and more worrying is the 'DNS Spoof' which would be associated with a port-scan attack.

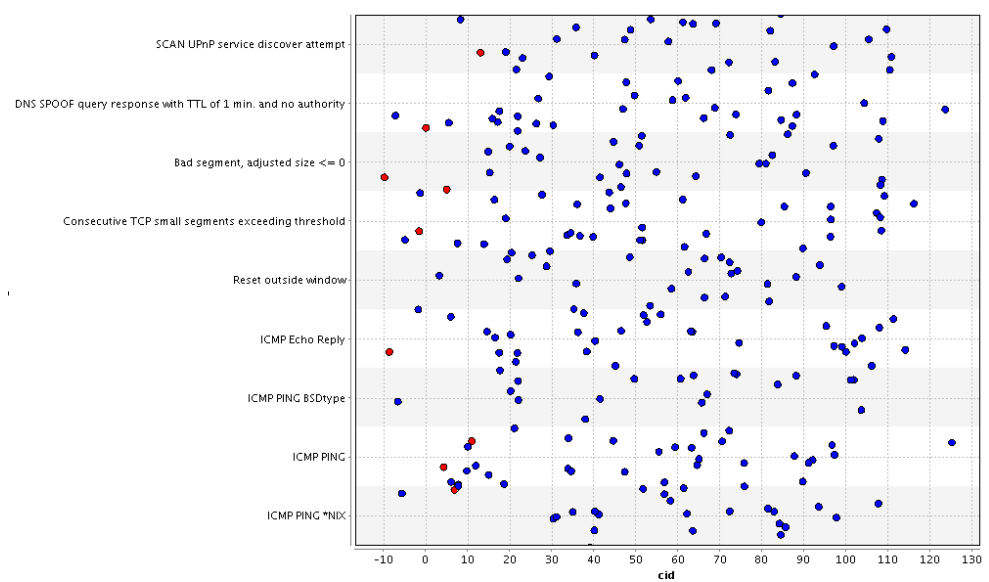


Figure 85: Outliers Graph (x = CID / y = sig_name)

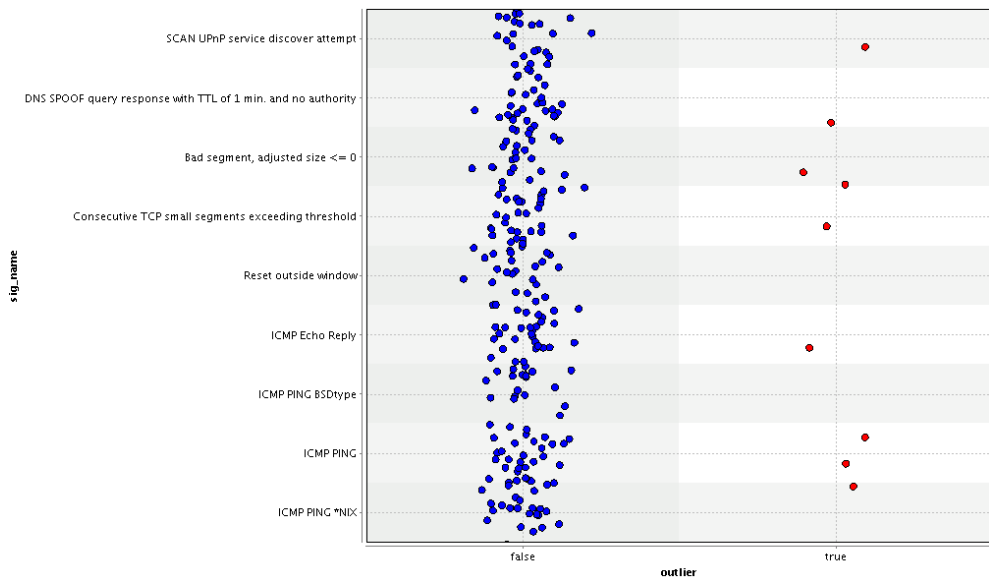


Figure 86: Outliers graph (x = outliers / y = sig_name)

We know that the *sig_name* and *sig_class_name* show some interesting information as to what is being recorded, so if we graph them together we can see a little more about this. If we look at

Figure 87 we can see that they are graphed against each other and this also shows that there are three distinct clusters viewable.

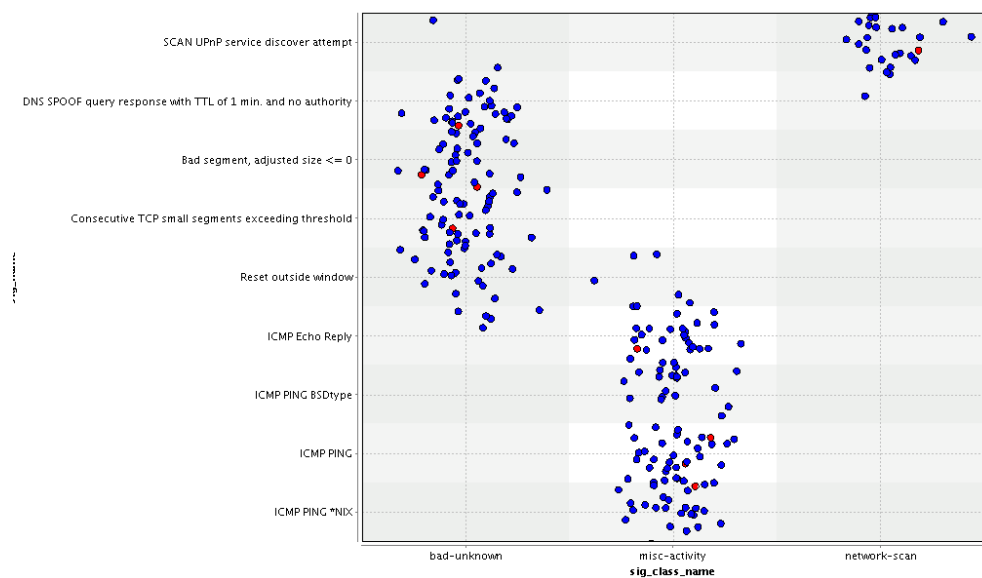


Figure 87: Outliers graph (x = sig_class_name / y = sig_name)

What we can see in Figure 87 is a distinct representation of where each of the outliers falls. Particularly we can see that in the top right hand corner there is the *network-scan* cluster which has only one outlier in it, this is the single outlier that we mentioned earlier that could be a scan with its port number of '53', this is confirming what we initially guessed and has also confirmed that this will most likely be a hostile signal.

6.6.5: Conclusion

After looking at the graphs we can come to some conclusions. In the ICMP header, even though we know that during the initial setup of the system there were pings done to establish connectivity, there has also been some, what could be classed as malicious or dangerous activity. This can be seen in the activity files that SNORT writes for us. In particular if we look at the "*Portscan*" file we can see that there has been a scan done (see Figure 88).

```
Time: 02/16-13:16:58.266210
event_ref: 0
92.62.43.77 -> 192.168.1.106 (portscan) TCP Portscan
Priority Count: 6
Connection Count: 6
IP Count: 2
Scanner IP Range: 85.236.110.226:92.62.43.77
Port/Proto Count: 6
Port/Proto Range: 113:8118
```

Figure 88: ICMP Port-scan

As we can see in this output from the port-scan file this is one of the few rare occasions on this network that the recorded IP address has not been NATed. What we have are a number of network external addresses that have been recorded, the scanner IP range is *92.62.43.77* and *85.236.110.226* respectively.

What we can do with these addresses is do what is called an *IP Lookup*, to do this we need to use the internet and go to an IP Lookup website; in this case I used ‘ip-lookup.net’. What came back from this look-up was that they belong to the IRC chat IP address range. In particular these ones seem to belong to a company that is based in Norway (see Figure 89) and the United Kingdom (see Figure 90).






IP : **92.62.43.77** Neighborhood 
Host : proxyscanner.quakenet.org  
Country : Norway  

Figure 89: IP Look-up (92.62.43.77) (IP-Lookup)

IP : **85.236.110.226** Neighborhood 
Host : irc.multiplay.co.uk  
Country : United Kingdom  

Figure 90: IP Look-up (85.236.110.226) (IP-Lookup)

Since this particular network that is being monitored runs so many machines on it, and some of those machines run games, it is likely that at some point one of them has connected to an IRC chat client, and as such that machine (192.168.1.106) has also been used by the IRC to act as a broadcast point for other people in the IRC. However this is not conducive to a stable or secure network, we want to stop these sorts of things from taking place, since to turn a user's computer into an IRC broadcaster they will need to do a port scan, which may reveal some weak points for anyone looking to attack.

6.7: Process work conclusions

After examining all of the collected data we can see that there may not have been any actual attacks, but there is defiantly some anomalous activity happening on the network. Because of this activity happening there is also a possibility that there would be an allowance for something malicious to happen through some of the less active ports. Areas of specific interest in the processes would be the not just the outliers but the anomalies within those outliers.

Specifically the “*segments exceeding threshold*” in the IP header tells us that some of the transmissions are going beyond the acceptable defined sizes and as such are causing problems with the stability of the rest of the data transmission. This could also indicate that something is trying to gain access to the network by forcing the router to reset itself or to trick the router into thinking that the data is something that it's not.

We would also be interested in the TCP port usage. Particularly the use of the port ‘139’, which is used by the *TCP/UDP NetBIOS service session*. Since this is a nearly defunct port, there is little reason for anything to be using it, and as such could be a cause for investigation in case there is a Trojan or worm on the network that is transmitting data through it.

In the UDP table there is another port security problem with port number ‘53’, which is associated with the SSDP protocol. Again like with port ‘139’ this is also not a used port much anymore, since I was used in the development of the internet and network technology. This port is also highly susceptible to attacks from worms and Trojans, but also from external network port scans trying to find an access point.

All of the port problems so far seem to have a similar thread running through them, in that they are all in the same or a close range to each other, and are all involving the same form of dead or dying technology. With the ICMP table we can see that these almost all tie in together, since the ICMP table showed a port scan and penetration from two IP addresses, 92.62.43.77 and 85.236.110.226. These two IP addresses were then found out to belong to the range that is shared and used by the IRC chat community, and as such there is a possibility there the ports were used by the IRC chat client to broadcast the signal on to other users.

Whether this is a hostile action or not, it does open up a large hole in the network security. To allow something that is not dangerous into a network unmonitored is a risky action since it could allow the network to remain open to possible attacks while the transmissions are going on by the IRC chat user's and clients. If however IRC is being used then it would need to be monitored on the network and only allow it in through an opened designated port rather than have it scanning for a port of its own. Either way this could be potentially dangerous even if it is allowed access and as such would still need to have the port's monitored.

6.8: Writing a new rule

Because the SNORT IDS is a rule based detection system and not anomaly based, we can essentially create a rule that will do whatever we want, and not just detect "hostile" traffic. One advantage with the SNORT IDS is that we can also use it to monitor user traffic to websites or torrent use. This can allow us to keep track of the kinds of things that people are doing on our network.

Now that we have a good idea of what is happening on the network and what kinds of traffic we are getting we can now create a new rule that will define specific events for tracking. To do this we need to have an understanding of not only what the traffic information is (IP addresses, ports, protocols ...) but also what the make-up of the rule is, since it can change for each given event type that we need or want it for.

If we look an existing rule (see Figure 91) we can see the general make-up of a SNORT rule. Starting with the kind of event that we are guarding against ("*alert*") and then the protocol that we are looking at or that is related to the particular event ("*udp*").


```
alert udp $EXTERNAL_NET any -> $HOME_NET 20433 (msg:"DDOS shaft
agent to handler"; flow:to_server; content:"alive"; reference:arachnids,256;
reference:cve,2000-0138; classtype:attempted-dos; sid:240; rev:7;)
```

Figure 91: DDOS rule example

After this is the source address followed by the destination address (“*\$HOME_NET*”) which can either be a particular host address on a network or just the entire network itself. Using the “Dollar” symbol (\$) in front of the name (i.e. *HOME_NET*) tells SNORT that this is a variable and that the condition of this variable will be set somewhere in the configuration file. We can then set that variable to any address that we need there. The direction arrow between the two address variables can point either way showing which way the traffic that is being classified is flowing, or can point in both directions (“<>”).

Rule Options

Rule options form the heart of Snort's intrusion detection engine, combining ease of use with power and flexibility. All Snort rule options are separated from each other using the semicolon ";" character. Rule option keywords are separated from their arguments with a colon ":" character. As of this writing, there are fifteen rule option keywords available for Snort:

- **msg** - prints a message in alerts and packet logs
- **logto** - log the packet to a user specified filename instead of the standard output file
- **minfrag** - set a threshold value for the smallest acceptable IP fragment size
- **tth** - test the IP header's TTL field value
- **id** - test the IP header's fragment ID field for a specific value
- **dsize** - test the packet's payload size against a value
- **content** - search for a pattern in the packet's payload
- **offset** - modifier for the content option, sets the offset to begin attempting a pattern match
- **depth** - modifier for the content option, sets the maximum search depth for a pattern match attempt
- **flags** - test the TCP flags for certain values
- **seq** - test the TCP sequence number field for a specific value
- **ack** - test the TCP acknowledgement field for a specific value
- **itype** - test the ICMP type field against a specific value
- **icode** - test the ICMP code field against a specific value
- **session** - dumps the application layer information for a given session

Figure 92: SNORT Rule options (Roesch)

When we write our rule we need to know what it is that we want to classify. Taking something simple like,

“Classify and log all internet/network traffic to and from the web address ‘*www.rte.ie*’ ”.

With this we can then begin to write out our new rule that will do that we want it to (see Figure 93). Using the rule we have now created we can *Log* all traffic to and from the address that we want, but as of yet the addresses have not been defined; only the variables have been. To define the variables we need to make some changes to the *snort.conf* (snort configuration) file.

```
Log tcp $WEB_ADDRESS any <> $HOME_NET any (msg:"An attempt to log  
into the RTE website has been made";)
```

Figure 93: Our Simple new rule

The changes that have to be made are to set the variable (*WEB_ADDRESS*) to an actual location. To do this we use a variable definition (“*var: <name> <value>*”), the *var* indication that it is a variable followed by the name of it, and the condition attached to it.

What we would end up with for our RTE web address is either, “*var WEB_ADDRESS www.rte.ie*” or if we do DNS or IP Loop-up we can obtain the actual IP address of the web site. (89.207.56.140) which will change it to “*var WEB_ADDRESS 89.207.56.140*”, either should work no problem.

The outcome of this will then produce a log of the event in the log file (or alert if we want it there) that would look something like what is in

```
02/18-17:06:01.585330  [**] [129:12:1] An attempt to log into the RTE website has  
been made {TCP} 192.168.1.109:50540 -> 89.207.56.140:139
```

Figure 94: Logged RTE event

Taking some more complexes we can look at the port scans that were made by the IRC chat servers. We know that the IP addresses are 92.62.43.77 and 85.236.110.226, and we know the ports that were used were port '53', we also know that the web addresses of each are "*multiplay.co.uk*" and "*quakenet.org*". We also know that from Figure 88 the protocol used was "6" with a priority also of "6". With this information we should be able to come up with a rule that will read the packets as they come in and classify specifically to that application.

Let's say we want to make it an *alert*, and have it log to the alert files. And that we want to ensure that the ICMP packets are registered.

```
Alert ICMP $IRC_ADDRESS any -> $HOME_NET :53 (msg:"a port scan by an
IRC chat client has been made"; logto:"c:\logs\alerts\IRC_Logs.log";
content:"|IRC|"; )
```

Figure 95: Custom rule to register IRC data

If we run this and we do get another attack or attempted penetration from the IRC chat server we will get a new message in the location specified in the rule. What the print out will be, will be something like in Figure 96.

```
02/15-19:58:55.179280  [**] [1:384:5] a port scan by an IRC chat client has been
made [**] [Priority: 6] {ICMP} 92.62.43.77 -> 192.168.1.106
```

Figure 96: Logged IRC chat server classification

Because SNORT is open source and anyone can create their own rules to do what they want, everyone can use them, even most of the paid for IDS applications such as KF-Sensor and NetBait use the SNORT rules standard and even the SNORT rules themselves. This makes SNORT not only highly adaptable but also shows it to be the industry standard for most (if not all) IDS systems.

Chapter 7: Conclusions and Further work

7.1: Chapter Introduction

After looking at the different stages and steps that have been taken in setting up and monitors an IDS, followed by creating a new, we can now look at the overall project.

Where we will suggest possible actions that could be taken after the life of this project as well as look at the conclusions of the work done.

7.2: Outcome of work

First off I would like to say that while there is quite a bit of data in this project, to make a fully justified objective conclusion is a bit hard to do, since we have only got a limited amount of data from a very short period of time. In reality this kind of system would be continually updated and examined over its life span. The data would be continually examined by machines that are more than capable of handling the larger volumes of data, and as such would have a much better understanding of the kinds of attacks that will impact on that particular system.

That being said though, this project has run to fairly expected levels. There was little that was out of the ordinary, although during the final week of operation there was an extortionate amount of data collected (1.7 million filtered) which was well beyond the handling capabilities of the computer that did the processing, as such this final weeks data had to be ignored during the processing stages.

After running for about 4 weeks, the volume of usable data collected was over 35000 filtered entries while the amount of scanned data was closer to 1million. This sounds like a lot though when you look at the network that it was set up on and the use of that network you can understand that it might be fairly normal.

From the Hand held devices these would be used for e-mail and not much else, while the Computers (PC) would have been used for email, web content and internet/online games. It is the PC's and the Games consoles that are being used for online games as well as downloading 'torrents' that will have generated the higher volume of traffic. This in its self is not dangerous but

the maintaining of an open connection to download and transmit the continuous streams of data and traffic could lead to possible attacks and penetrations of the network by an outside source. Because of this traffic coming from known devices it is also partly known what kind of traffic would have been scanned and also some of what was recorded.

7.3: Conclusions from Processes

With the amount of data that was collected over the time period that was available. The diversity was not really enough to come up with a first class conclusion; because of this some of the conclusions might be lacking in clarity or definition and may not be entirely accurate. That being said though, the understanding of the data has been greatly increased and given a larger amount of data over a longer period of time (and a better research computer) the outcome would be a lot better and could produce some high level results.

One of the key things that stands out from the data was the massive amount of ‘outside window reset’ packets, which mean that the number of packets being recorded were there because they were either too large or too small for them to be properly classified and as such were classified as ‘*possible dangerous*’, to get around this we would have had to increase the *max_tcp* value to allow for the larger packet size, however doing this would allow for a larger quantity of recorded data and may have skewed the data more than it may have already been.

Another thing that stands out was the *port scan* that was done by an IRC server. As far as I am aware no one on the network uses IRC, so this would lead me to believe that they have gotten my data stream from somewhere and piggybacked on it to my home network, possibly from one of the machines that is used for gaming. This IRC itself is not a dangerous thing, although hackers do like to use IRC and they have been known to hack into it at times (smurfing) which could lead to my network being attacked. Thankfully at this time it doesn’t appear to have been anything more than either a scan or a broadcast point.

Also looking at some of the ports that were used (port 53) it would also appear that one of the machines on the network (most likely a PC) has some form of virus or Trojan on it. This is not an uncommon thing, but the opening of the ports indicates that it is broadcasting or transmitting data out of the network. Once this project has been totally finished this is something else that will be investigated and all machines checked on the network.

7.3.1: Timestamps

Throughout this document I have referred to timestamps. In relation to them being of possible benefit when finding the outliers, this will come in knowing your network. What hour's is it used? How is it used? What is it used for? In answering some simple questions about your network or the network that is being monitored we can see that if a piece of traffic at 4:00pm is not only usual for that time, but if it should be at that time. Should anyone be on the network doing that activity at that time of the day/night?

With relation to the network that has been monitored in this project the general network usage would be between 10am and anything up until 2am the following day. This would be outside of the normal (working) hours of a company but since this is a private network, the 'normal' hours could be anything. However within those hours there will be certain trends that would be known. General internet surfing would happen though almost the entire time, but one different machines at different times. So we can look at the attacks and see where they were either going to or originated from, from within the network. This will allow us to either pay particular attention to that piece of data or to ignore it entirely.

The time of day would also show as to what type of device is being used, since the games consoles (Nintendo Wii, Xbox 360) would not regularly be used after 11pm, and while these don't generally in history get attacked, they can provide a means for an attacker to gain access to the network.

7.4: Possible future work

Aside from a bit of tinkering with the SNORT IDS configuration file to better streamline it, there would be a lot more knowledge to be had before starting it. Although learning on a systems such as this is not something that people do in a weekend, but rather spend lifetime's learning and perfecting. It is a system and records information such that you can only find out about something once it has happened.

One area that could improve the results would be to tinker around with the standard *rule* set. Since I only used the standard rules, and all of the rules by default are turned off (commented out) there is no real knowledge as to what rules would be needed on such a network. Because of this all the rules were included, even those that were deemed as experimental or out of date.

If I could run this project again I would like to have set it up on the college network somewhere, probably on the research network, to make use of their high traffic flow and their static IP address which would make it easier to identify the source of the data that is being sent and received. I would have also used a more powerful machine to try to make it a virtual IDS so I would run a virtual server alongside it to try to increase the chances of being attacked.

To anyone that is thinking of doing this project for themselves, I would highly suggest doing it on an open network with a static IP address. If at all possible it would be well worth going to the IT department of the college and seeing if they could allow you to hook it up next to the main firewall, or in front of the firewall. It is not something that would probably be allowed but it could be argued that this IDS system is a monitor and not a Honeypot so attacks will not be attracted but only found. It could also be said that this is an information system not a preventative or penetrative one.

7.4.1: Outlier detection and Classification

In this project we concentrated on the *Detect Outlier (Distance)*, this was quite heavily influenced due to the constraints of the machine that was doing the work. Given a better machine we would be able to handle the more complex versions such as *COF* which takes more memory to work due to its involvement with not only the point we are looking at and the neighbour but also the class that they both belong to.

Another area that would have been good to use would be classification algorithms. In particular the Tree or Naïve Bayes, which would allow us not only a more visual depiction of the structure of the outliers and traffic but also what may constitute a hostile package. If the data that we have no could generate a good enough model, we could then use that model to test against future data to classify it.

Chapter 8: Personal Reflections

The reason I decide on doing this project in the first place was because I hopefully will get into Networking and Network security once I finish with this degree. This has been an area that I have wanted to get into for a long time now, and I felt that doing the investigation into the Honeypot and IDS on a live network would aid me in future career choices. While this has been confusing at times and frustrating trying to deal with the implementation of the SNORT IDS system, the choice of IDs I felt has been the right one, since it would appear that every other IDS and Honeypot tool is based around the ground work of SNORT, and they all seems to implement the SNORT rule set that can be download from their site.

Initially this project seems like something that I would be highly interested in, but after a while of running it I did come to realise that it was a lot more complicated that was originally though. But I did still enjoy the work, and finding out what about what happening on the network.

After running this project, I feel that I have gained a bit more of an understanding as to how the data flows on a network, or at least on a private network with no real routing on it, and how it is built of its different components.

I feel that while this project may have not gone totally to plan with the amount and types of data that were collected, I have managed to uncover a number of problems on the home network, and will be trying to resolve these in the near future.

As for personal I have managed to gain a greater insight into the process of doing a project of this scale and have a better understanding as to what sort of work needs to go into such a thing. And I would be fully prepared in the future to be able to do large scale documentation and consultation work that might have this magnitude of information involved.

All being said, if I could go back and do this again, I feel that I would choose the same path that I took originally, but I would be more inclined to try to do it on the college network rather than the home network in an attempt to get a better data set. I would also have run it on a virtual machine with a lot more available memory rather than the one I ended up using. The idea of the virtual machine would be in hopes that I can have a second virtual machine running alongside it running some form of server to try to entice more malicious data into it and try to get closer to the Honeypot rather than the IDS. Not that the IDS is a bad design.

Bibliography

Computational science and its applications. (May 8-11, 2006). *ICCSA 2006 : international conference* (p. 862). Glasgow, UK: Springer.

Allen, M. (n.d.). *Security attacks*. Retrieved from The Computer Technology Documentation Project:
<http://www.comptechdoc.org/independent/security/recommendations/secattacks.html>

Allinfosec. (n.d.). *How Playstation Network was HACKED*. Retrieved from Information Security news and exploits: <http://www.allinfosec.com/2011/04/28/how-the-playstation-network-was-hacked/>

America, B. S. (n.d.). *The Mysterious Venus Flytrap*. Retrieved from botany.org:
<http://www.botany.org/bsa/misc/carn.html>

Anoop Singhal, S. J. (n.d.). Data warehousing and data mining techniques for intrusion detection systems. In S. J. Anoop Singhal, *Distrabuted Parallel databases* (pp. 149 - 166). Springer.

Arabtrust. (n.d.). *Symantec Decoy server*. From Whitepaper:
<https://www.arabtrust.com/partner/symantec/decoyserver.html>

Atlantic. (2007). Pulsing Zombie. In Atlantic, *Encyclopedia Of Information Technology* (p. 397). Atlantic Publishers & Dist.

AuditMyPC. (n.d.). *TCP 49000*. Retrieved from AuditMyPC.com: <http://www.auditmypc.com/tcp-port-49000.asp>

BASE project. (n.d.). *Basic Analysis Security Engine*. Retrieved from BASE project:
<http://base.secureideas.net/>

Beal, M. (n.d.). *NetBIOS (Network Basic Input/Output System)*. Retrieved from SearchNetworking.com: <http://searchnetworking.techtarget.com/definition/NetBIOS>

Berge, M. (n.d.). *Intrusion Detection FAQ: What is a Host Intrusion Detection System?* Retrieved from SANS.org (Ernst & Young): http://www.sans.org/security-resources/idfaq/what_is_hips.php

- Bidgoli, H. (2004). "Boomerang" attacks. In H. Bidgoli, *The Internet Encyclopedia in a 3-volume reference work on the internet as a business tool, IT platform, and communications and commerce medium*. (p. 429). John Wiley & Sons.
- Binary Vision. (n.d.). *TCP port 139*. Retrieved from LinkLogger:
<http://www.linklogger.com/TCP139.htm>
- Bloom, R. (n.d.). *Linear Regression and Correlation: Outliers*. Retrieved from Connexions:
<http://cnx.org/content/m33271/latest/>
- Canada Television. (n.d.). *Chinese hackers try to access Canadian gov't data*. Retrieved from CTV News: <http://www.ctv.ca/CTVNews/TopStories/20110216/china-hackers-canada-finance-department-110216/>
- Charlse Scot, P. W. (n.d.). *SNORT for Dummies*. WILEY.
- Chinese University of Hong Kong. (n.d.). *Glossary*. From Chinese University of Hong Kong:
<http://www.cuhk.edu.hk/itsc/security/isglosry/index.html#H>
- Cisco systems. (n.d.). *Cisco Security Advisory: 7xx Router Password Buffer Overflow*. Retrieved from Cisco Products and services:
http://www.cisco.com/en/US/products/products_security_advisory09186a00800b13a6.shtml
- da_pathfinder. (n.d.). *What is a port scan?* Retrieved from Yahoo answers:
<http://answers.yahoo.com/question/index?qid=20061105020422AAtre1p>
- Daniel Barbará, S. J. (2002). Intrusion Detection. In S. J. Daniel Barbará, *Applications of data mining in computer security* (pp. 7-9). Kluwer Academic Publishers.
- Donovan, A. (2010). *Honeypots and Datamining*.
- Efaw, K. (n.d.). *Installing Snort 2.8.5.2 on Windows 7*. From SNORT:
http://www.snort.org/assets/135/Installing_Snort_2.8.5.2_on_Windows_7.pdf
- FAQ: Intrusion Detection Systems. (n.d.). *FAQ: Intrusion Detection Systems*. Retrieved from linuxsecurity: http://www.linuxsecurity.com/resource_files/intrusion_detection/network-intrusion-detection.html

- Federal Financial Institutions Examination Council. (n.d.). *Glossary*. From Federal Financial Institutions Examination Council:
http://www.ffiec.gov/ffiecinfobase/booklets/e_banking/ebanking_04_appx_b_glossary.htm
1
- Gehringer, D. E. (n.d.). *Honeypots*. From Ethics in Computing:
<http://ethics.csc.ncsu.edu/abuse/hacking/honeypots/study.php>
- Gladwell, M. (n.d.). What is Outliers about? In M. Gladwell, *Outliers*.
- Google. (n.d.). *ghh, The 'Google Hack' Honeypot*. From Google Hack Homeypot:
<http://ghh.sourceforge.net/>
- Graham, R. (n.d.). *Network Intrusion detection System*. Retrieved from Linuxsecurity:
http://www.linuxsecurity.com/resource_files/intrusion_detection/network-intrusion-detection.html#1.1
- HoneyD. (n.d.). *General Information*. From HoneyD: <http://www.honeyd.org/general.php>
- Honeynet Porject. (2001). *Know your enemy, Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Adisson-Wesley.
- Honeypot, W. (n.d.). *Honeypot (computing)*. From Wikipedia:
[http://en.wikipedia.org/wiki/Honeypot_\(computing\)](http://en.wikipedia.org/wiki/Honeypot_(computing))
- Information Please® Database. (n.d.). *Computer virus timeline*. Retrieved from Information please:
<http://www.infoplease.com/ipa/A0872842.html>
- IP-Lookup. (n.d.). *IP Look-up*. Retrieved from IP Look-up: <http://ip-lookup.net>
- Javvin network management and security. (n.d.). *Port Scann attacks*. Retrieved from Javvin.com:
<http://www.javvin.com/networksecurity/PortScanAttack.html>
- Kevin Fenzi, D. W. (n.d.). *Linux Security HOWTO*. Retrieved from The Linux Documentation Project: <http://tldp.org/HOWTO/Security-HOWTO/x82.html>
- KF Sensor. (n.d.). *About, KF Sensor*. From KF Sensor:
<http://www.keyfocus.net/kfsensor/index.php>

- LAMPSON, R. N. (n.d.). *Network Attack and Defence*. Retrieved from University of cambridge:
<http://www.cl.cam.ac.uk/~rja14/Papers/SE-18.pdf>
- Leslie Cherian, T. D. (n.d.). *Honeynet/Honeypot Project*. Retrieved from Clarkson University:
<http://web2.clarkson.edu/projects/itl/projects/honey/>
- MacTiernan, R. (2010/2011). Original Work. *Honeypot Intrusion Detection*. IT Blanchardstown.
- Masaki Ishiguro, S. G. (n.d.). *An Analysis on Distribution of Malicious Packets and Threats over the Internet*. Retrieved from Asian-Pacific Advanced Network:
http://master.apan.net/meetings/xian2007/publication/009_Ishiguro.pdf
- Michael "Mullins CCNA, M. (n.d.). *Prevent hacker probing: Block bad ICMP messages*. Retrieved from TechRepublic: <http://www.techrepublic.com/article/prevent-hacker-probing-block-bad-icmp-messages/5087087>
- Michael DeAgonia, P. G. (n.d.). *Mac OS-X Smackdown: Linux vs. Mac OS-X vs. Windows Vista vs. Windows XP*. From ComouterWorld:
http://www.computerworld.com/s/article/print/9075000/OS_Smackdown_Linux_vs._Mac_OS_X_vs._Windows_Vista_vs._Windows_XP
- Michael Mullins CCNA, M. (n.d.). *Which Honeypot Should I use*. From TechRepublic:
http://articles.techrepublic.com.com/5100-10878_11-1042527.html
- Microsoft. (n.d.). *Common types of network attacks*. Retrieved from Microsoft TechNet:
<http://technet.microsoft.com/en-us/library/cc959354.aspx>
- Mitchel, B. (n.d.). *UDP*. Retrieved from About.com:
<http://compnetworking.about.com/od/networkprotocolsip/g/udp-user-datagram-protocol.htm>
- M-Vincent. (n.d.). *Linux Vs. Mac OS-X Vs. Windows*. From In My Life: <http://in-my-worldz.blogspot.com/2009/06/linux-vs-mac-os-x-vs-windows.html>
- NetBait. (n.d.). *About NetBait Enterprise*. From NetBait:
<http://netbaitinc.com/products/nbenter.shtml>
- Oded Z. Maimon, L. R. (n.d.). Classifying Tumors. In L. R. Oded Z. Maimon, *Data mining and knowledge discovery handbook*. Springer.

Open Source Initiative. (n.d.). *GNU General Public License, version 3 (GPLv3)*. From Open Source Initiative: <http://www.opensource.org/licenses/gpl-3.0.html>

orbixhost. (n.d.). *Top 10 things to look for when under a DDOS attack*. Retrieved from Serchen interactive: <http://forums.serchen.com/showthread.php?41924-Top-10-Things-to-Look-For-when-Under-a-DDOS-Attack!>

O'Shaughnessy, S. (2010). *Attack Data Generator*.

Pakhare, J. (n.d.). *Different Type of Computer Viruses*. From Buzzle: <http://www.buzzle.com/articles/different-types-of-computer-viruses.html>

rapid-i. (n.d.). *Detect Outliers (COF)*. Retrieved from rapid-i wiki: [http://rapid-i.com/wiki/index.php?title=Detect_Outlier_\(COF\)](http://rapid-i.com/wiki/index.php?title=Detect_Outlier_(COF))

rapid-i. (n.d.). *Detect outliers (Densities)*. Retrieved from rapid-i wiki: [http://rapid-i.com/wiki/index.php?title=Detect_Outlier_\(Densities\)](http://rapid-i.com/wiki/index.php?title=Detect_Outlier_(Densities))

rapid-i. (n.d.). *Detect Outliers (Distance)*. Retrieved from rapid-i wiki: [http://rapid-i.com/wiki/index.php?title=Detect_Outlier_\(Distances\)](http://rapid-i.com/wiki/index.php?title=Detect_Outlier_(Distances))

rapid-i. (n.d.). *Detect Outliers (LOF)*. Retrieved from rapid-i wiki: [http://rapid-i.com/wiki/index.php?title=Detect_Outlier_\(LOF\)](http://rapid-i.com/wiki/index.php?title=Detect_Outlier_(LOF))

Roesch, M. (n.d.). *Writing Snort Rules: How To write Snort rules and keep your sanity*. Retrieved from ftp://interface.es.dvo.ru/pub/Vyatta/build-iso/pkg/vyatta-snort/debian/my/snort_rules.html

SANS Institute. (n.d.). *ICMP Attacks illustrated*. Retrieved from SANS reading room: http://www.sans.org/reading_room/whitepapers/threats/icmp-attacks-illustrated_477

Sirkanth. (n.d.). *Denial of service attack*. Retrieved from goHacking: <http://www.gohacking.com/2008/11/denial-of-service-attack.html>

SNORT. (n.d.). *About Snort*. From SNORT: <http://www.snort.org/snort>

SNORT. (n.d.). *SNORt user manual*. Retrieved from SNORT.org: https://www.snort.org/assets/166/snort_manual.pdf

Specter. (n.d.). *Details*. From Specter: <http://www.specter.ch/details50.htm>

Tech-faq. (n.d.). *Honeypot*. Retrieved from Tech-faq: <http://www.tech-faq.com/honeypot.html>

TechSupport Forums. (n.d.). *i am being attacked*. Retrieved from TechSupport Forums:
<http://www.techsupportforum.com/forums/f139/i-am-being-attacked-523649.html>

TechTarget. (n.d.). *DoS Definition*. Retrieved from SearchSoftwareQuality:
<http://searchsoftwarequality.techtarget.com/definition/denial-of-service>

The free Dictionary. (n.d.). *computer network attack*. Retrieved from thefreedictionary.com:
<http://www.thefreedictionary.com/computer+network+attack>

The Honeynet Project. (n.d.). *Welcome to Honeywell Projectsite*. From The Honeynet Project:
<https://projects.honeynet.org/honeywall/>

Time-Management-Guide.com. (n.d.). *Use a decision tree analysis to systematically arrive at your smartest choice*. Retrieved from Time-Management-Guide.com: <http://www.time-management-guide.com/decision-tree.html>

Tony Bradley, C. M. (n.d.). *Introduction to Intrusion Detection Systems (IDS)*. Retrieved from About.com: <http://netsecurity.about.com/cs/hackertools/a/aa030504.htm>

Tony Bradley, C.-I. (n.d.). *Introduction to Port Scanning*. Retrieved from About.com:
<http://netsecurity.about.com/cs/hackertools/a/aa121303.htm>

TopBits.com. (n.d.). *Understanding network attacks*. Retrieved from tech-faq.com:
<http://www.tech-faq.com/network-attacks.html>

Uniblue Systems. (n.d.). *port ranges*. Retrieved from pc-Library.com: <http://www.pc-library.com/ports/ranges/5073-5404/>

Uniblue. (n.d.). *TCP & UGP port information 1900*. Retrieved from PC-Library.com:
<http://www.pc-library.com/ports/tcp-udp-port/1900/>

Uniblue. (n.d.). *UDP port 53*. Retrieved from PC-Library.com: <http://www.pc-library.com/ports/tcp-udp-port/53/>

University of Delaware. (n.d.). *Join Operator*. Retrieved from University of Delaware:
http://www.udel.edu/evelyn/SQL-Class2/SQLclass2_Join.html

Web of Trust. (n.d.). *Denial of service*. Retrieved from WOT:

http://www.mywot.com/wiki/Denial_of_Service_Attack#Blind_denial_of_service

Wikimedia. (n.d.). Retrieved from

http://en.wikipedia.org/wiki/Internet_Control_Message_Protocol#List_of_permitted_control_messages_.28incomplete_list.29

Wikimedia. (n.d.). *Decision Tree learning*. Retrieved from Wikipedia:

http://en.wikipedia.org/wiki/Decision_tree_learning

Wikimedia. (n.d.). *File:Ircnetz-Schema.svg*. Retrieved from Wikipedia:

<http://en.wikipedia.org/wiki/File:Ircnetz-Schema.svg>

Wikimedia. (n.d.). *Internet Control Message Protocol*. Retrieved from Wikipedia:

http://en.wikipedia.org/wiki/Internet_Control_Message_Protocol

Wikimedia. (n.d.). *NetBIOS service session*. Retrieved from Wikipedia:

http://en.wikipedia.org/wiki/NetBIOS#Session_service

Wikimedia. (n.d.). *Outlier*. Retrieved from Wikipedia: <http://en.wikipedia.org/wiki/Outlier>

Wikimedia. (n.d.). *User Datagram Protocol*. Retrieved from Wikipedia:

http://en.wikipedia.org/wiki/User_Datagram_Protocol

WikiPedia. (n.d.). *Fragmented Distrabution attack*. Retrieved from Wikipedia:

http://en.wikipedia.org/wiki/Fragmented_distribution_attack

Wikipedia. (n.d.). *Honeypot(computing)*. Retrieved from Wikipedia:

[http://en.wikipedia.org/wiki/Honeypot_\(computing\)](http://en.wikipedia.org/wiki/Honeypot_(computing))

Wikipedia. (n.d.). *Host based intrusion detection system*. Retrieved from Wikipedia:

http://en.wikipedia.org/wiki/Host-based_intrusion_detection_system

Wikipedia. (n.d.). *Intrusion Detection Systems*. Retrieved from Wikipedia:

http://en.wikipedia.org/wiki/Intrusion_detection_system#Passive_and.2For_reactive_systems

Wikipedia. (n.d.). *IP Header*. Retrieved from Wikipedia:

http://en.wikipedia.org/wiki/IPv4_header#Header

Wikipedia. (n.d.). *Network intrusion detection system*. Retrieved from Wikipedia:

http://en.wikipedia.org/wiki/Network_intrusion_detection_system

Wikipedia. (n.d.). *Personal Computer*. From Wikipedia:

http://en.wikipedia.org/wiki/Personal_computer#Operating_system

Wikipedia. (n.d.). *Reflection Attack*. Retrieved from Wikipedia:

http://en.wikipedia.org/wiki/Reflection_attack

Wikipedia. (n.d.). *Transmission Control Protocol (TCP segment)*. Retrieved from Wikipedia:

http://en.wikipedia.org/wiki/Transmission_Control_Protocol#TCP_segment_structure

Wikipedia. (n.d.). *Venus Flytrap*. From Wikipedia: http://en.wikipedia.org/wiki/Venus_Flytrap

Wikipedia. (n.d.). *Zombie Computer*. Retrieved from Wikipedia:

http://en.wikipedia.org/wiki/Zombie_computer










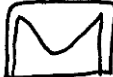
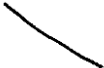










WindowsITPro. (n.d.). *New Version of SPECTER IDS Honeypot Available for Windows XP*. From

WindowsITPro: <http://www.windowsitpro.com/article/security/new-version-of-specter-ids-honeypot-available-for-windows-xp.aspx>

Withney, L. (n.d.). *WikiLeaks supporters attack MasterCard site*. Retrieved from CNet news:

http://news.cnet.com/8301-13578_3-20024966-38.html

Appendix 1: Diagram Key

	Wi-Fi		Wi-Fi radio
	Laptop		Games Console (wireless)
	PC (Computer)		Database
	Router		Firewall
	'The Cloud' (Internet)		Data Packet
	Wired cable		Flow Direction
	Wireless connection		IDS Sensor
	Hand held device (iPod, phone)		Honeypot
	Server machine		Watching packet process
	Mining data process		Capture packet process
	Attacker		

