Jere Koivunen

Installing and Testing Open Source Security Appliances

Metropolia University of Applied Sciences

Bachelor of Engineering

Information and Communication Technology

Bachelor's Thesis

3 December 2018

Author Title Number of Pages Date	Jere Koivunen Installing and Testing Open Source Security Appliances 57 pages 3 December 2018
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Professional Major	Communication Networks and Applications
Instructors	Marko Uusitalo, Senior Lecturer

This thesis includes installation and testing of open source security appliances as well as discovering if they are sophisticated enough to serve companies in providing security. IPFire and OPNsense were chosen for this thesis. The testing was aimed to give information of important firewall features, usability and overall reliability. The test results were used to compare IPFire and OPNsense to each other and to compare open source firewalls in general to commercial firewalls. At the end of this thesis the reader will have a clear insight to what open source firewalls are capable of and what crucial things are missing when compared to commercial firewalls.

This thesis contains theory concerning the basic concepts of firewalls and the protocols used in them. The hardware used to test IPFire and OPNsense are presented and the reasons for choosing them are explained. The environment in which the open source firewalls were tested, what was tested and how the tests were performed will be stated in this thesis. The future of open source firewalls is also briefly covered.

Keywords: Firewall, threat prevention, NGFW, open source

Tekijä Otsikko Sivumäärä Aika	Jere Koivunen Avoimen lähdekoodin tietoturvalaitteiden asennus ja testaus 57 sivua 03.12.2018
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikka
Ammatillinen pääaine	Tietoverkot ja sovellukset
Ohjaajat	Marko Uusitalo, Vanhempi opettaja

Tämän opinnäytetyön tavoitteena oli selvittää avoimen lähdekoodin palomuurien nykytilaa ja, arvioida ovatko ne riittävän kehittyneitä yritysten käyttöön. Opinnäytetyöhön valittiin kaksi avoimen lähdekoodin palomuuria: IPFire ja OPNsense. Nämä palomuurijärjestelmät asennettiin dedikoidulle laitteelle, jossa testattiin niiden toimivuutta, käytettävyyttä sekä yleistä toimintavarmuutta. Testien perusteella IPFireä ja OPNsenseä vertailtiin ensin keskenään ja sitten kaupallisiin palomuureihin. Vertailun avulla saadaan hyvä yleiskuva, missä asioissa avoimen lähdekoodin palomuurit ovat jäljessä verrattuna kaupallisiin vaihtoehtoihin. Tuloksia voi käyttää esimerkiksi pienyritykset, jotka haluavat lisätä tietoturvaa pienellä budjetilla.

Opinnäytetyössä käydään läpi palomuureihin liittyvää teoriaa ja avataan protokollia, joita palomuurit käyttävät suojaustoimintojen toteutuksessa. Opinnäytetyössä kerrotaan, millä perusteilla kyseiset avoimen lähdekoodin palomuurit valittiin työhön. Työstä selviää, minkälaisia laitteita ja komponentteja palomuurien testauksessa käytettiin ja minkälaisessa ympäristössä testit suoritettiin. Työssä on myös käyty läpi, mitä asioita kannattaa ottaa huomioon palomuureja testattaessa ja minkälaisilla metodeilla testit toteutettiin. Lopussa käydään myös lyhyesti läpi avoimen lähdekoodin palomuurilaitteiden tulevaisuuden näkymiä.

Työssä tehtyjen testausten ja arviointien jälkeen päädyttiin siihen, että OPNsense on ominaisuuksiltaan ja käytettävyydeltään kokonaisvaltaisempi ratkaisu, jossa on otettu paremmin yrityksen tarpeet huomioon. Testauksissa paljastui myös avoimen lähdekoodin palomuurien heikko kyky torjua uhkia sovellustasolla sekä IPS-ominaisuuden avulla. Kaupallisiin palomuureihin verrattuna myös käytettävyydessä esiintyi suuria eroja. Palomuurien rasitustestien jälkeen todettiin, että Linux-käyttöjärjestelmä on todennäköisesti paremmin optimoitu kuin FreeBSD-käyttöjärjestelmä. Ottaen huomioon kaikki asiat, jotka tulivat esiin työ aikana, avoimen lähdekoodin palomuurit soveltuvat parhaiten pienten yritysten käyttöön.

Avainsanat	Palomuuri, uhkien torjunta, NGFW, avoin lähdekoodi

Contents

List of Abbreviations

ACE	Access Control Entry
ACL	Access Control List
CARP	Common Address Redundancy Protocol
CPU	Central Processing Unit
DoS	Denial of Service
DDoS	Distributed Denial of Service
DHCP	Dynamic Host Control Protocol
DNS	Domain Name System
DMZ	Demilitarized Zone
HA	High Availability
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Prevention System
IT	Information Technology
MAC	Media Access Control

NGFW	Next Generation Firewall
NIC	Network Interface Card
NTP	Network Time Protocol
OSI Model	Open Systems Interconnections Model
OSPF	Open Shortest Path First Protocol
PC	Personal Computer
QoS	Quality of Service
SSH	Secure Shell Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UI	User Interface
USB	Universal Serial Bus
UTM	Unified Threat Management
VoIP	Voice over Internet Protocol
VPN	Virtual Private Network
VRRP	Virtual Router Redundancy Protocol
WAN	Wide Area Network

1 Introduction

In my four years of studying at Metropolia it has become increasingly clear that one of the biggest concerns in the field of IT is how to make services and information accessible in every day of the year and around the clock. The development in IT over the years has enabled us to achieve this and therefore, almost everything in our daily lives is more dependent on networks. However, there are many threats that aim to disrupt the access to services and information or even destroy or steal information. The growth of IT related systems and the high value of information stored in them have also drawn the attention of criminals. This has become one of the biggest threats because it can cause tremendous financial losses. Also, as IT systems have evolved it has become much easier for even novice hackers to cause harm because the attack tools and techniques have become more automated and thus require less technical expertise. So, the use of security appliances has become a necessity in businesses no matter what size. Security appliances provide flexible work environments, access control, high availability of services, auditing, traffic shaping, threat prevention and much more. The high level of visibility provided by modern firewalls also causes issues with privacy. Policies must be made within companies that determine what information is allowed to be gathered from clients.

This thesis discusses the state of open source firewalls and compares them to commercial firewalls. In this thesis two open source firewall distributions are installed on a dedicated device and tested with different methods. The open source firewall distributions chosen for this thesis are IPFire and OPNsense. IPFire is based on the Linux operating system and OPNsense on FreeBSD. Other well-known open source security appliances used world-wide are, for example, pfSense, Smoothwall, ClearOS, Untangle, IPCop and Endian firewall. The purpose of the testing is to find out any differences between open source firewalls and to get an overview of open source firewall capabilities in general. The results received from testing are used to evaluate open source firewalls and compare them to commercial firewalls.

2 Theory

2.1 Basic firewalling concept

When it comes to firewalls, there are certain principles that most modern firewalls follow. The firewall in most cases is the device that is on the network edge between the private and public networks often referred to as inside (private) and outside (public) networks. By default, all traffic originated from the public network is blocked and any traffic leaving the private network is allowed. If it is necessary for a service running inside the private network to be accessible from the public network a network segment called demilitarized zone (DMZ) should be created. A DMZ can be considered as a network that is in between the private and the public network. By running services that need to be accessed from the internet in the DMZ instead of the private inside network, security is greatly enhanced. The default ruleset for a DMZ is that traffic moving from the DMZ network to the private inside network is blocked and traffic from the private inside network is allowed to the DMZ network. To make the services available to the public network the default rule for outgoing and incoming traffic to the public network is that all traffic is allowed. These default firewall rules can be modified as required. Figure 1 illustrates the basic operation of a firewall in more detail.

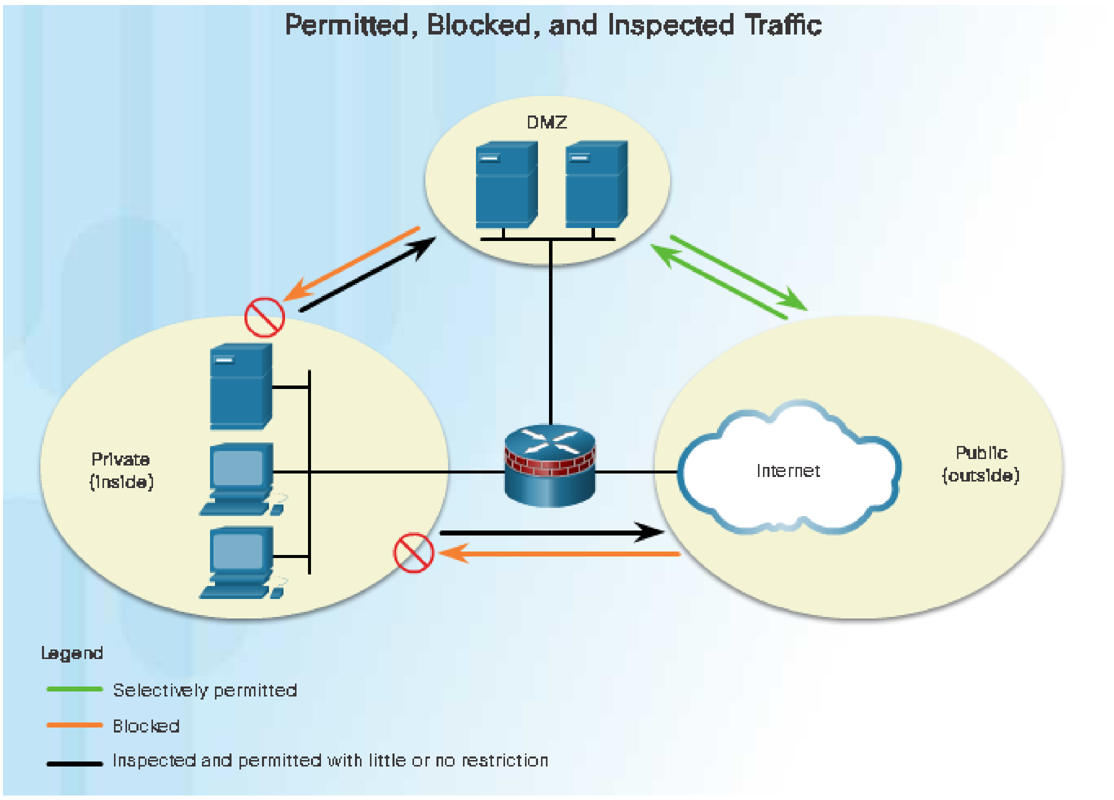

Permitted, Blocked, and Inspected Traffic
DMZ
Private (inside)
Internet
Public (outside)
Legend
Selectively permitted
Blocked
Inspected and permitted with little or no restriction

Figure 1. Common rules for firewall zones

It is possible to create multiple zones on modern firewalls which can be configured with their own ruleset. [1]

2.2 Firewall types

2.2.1 Stateless

A stateless packet filtering firewall is one of the oldest firewall mechanisms. A packet filtering firewall discards or accepts traffic based on the source and destination IP addresses as well as the source and destination ports. In this method the firewall goes through the packets one by one and keeps no track of the ongoing connections. Stateless firewalls are usually implemented using Access Control Lists (ACL). An access control list is simply a list of Access Control Entries (ACE) or more commonly rules that either accept or block traffic based on IP addresses, ports and protocols. ACLs can be configured for both inbound and outbound traffic. Below in figure 2 is an example network that is used to demonstrate how ACLs function.

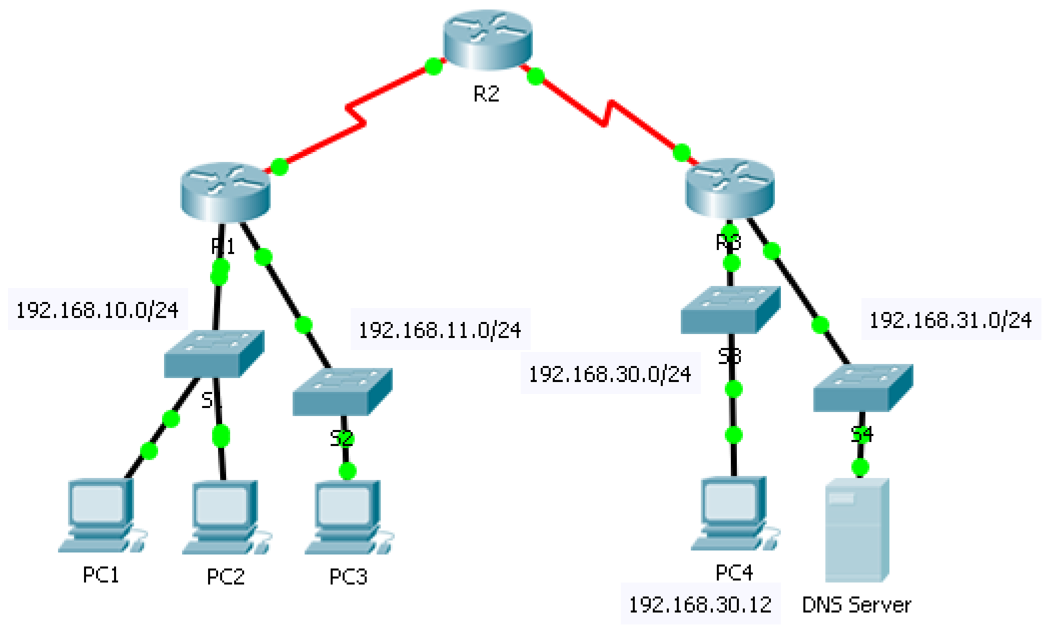
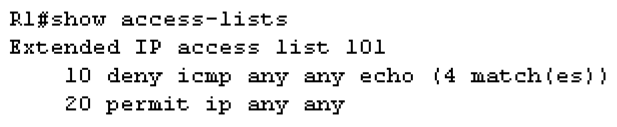


Figure 2. Example network topology

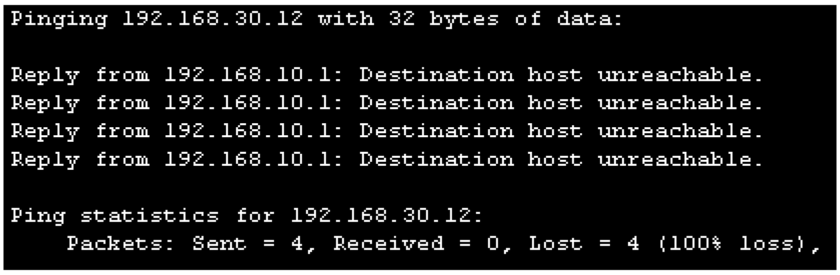
In figure 3 is the actual ACL that is used to block ICMP traffic.



```
R1#show access-lists
Extended IP access list 101
    10 deny icmp any any echo (4 match(es))
    20 permit ip any any
```

Figure 3. Access list blocking ICMP traffic

The access list in figure 3 has two entries: The upper entry will drop ICMP traffic that is originated from any network to any destination network and the lower entry will accept all traffic with IP protocol from any network to any destination. In this case the access list is attached to the outbound interface of R1 which means that both the 192.168.10.0/24 and the 192.168.11.0/24 networks are filtered when they communicate through R1.



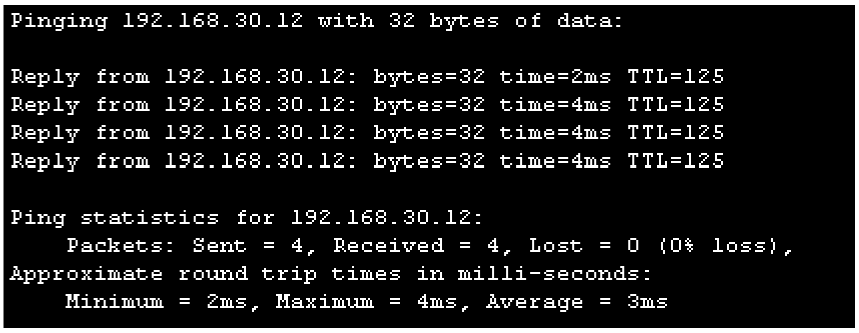
```
Pinging 192.168.30.12 with 32 bytes of data:

Reply from 192.168.10.1: Destination host unreachable.
Reply from 192.168.10.1: Destination host unreachable.
Reply from 192.168.10.1: Destination host unreachable.
Reply from 192.168.10.1: Destination host unreachable.

Ping statistics for 192.168.30.12:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

Figure 4. Ping attempt with access list on

In figure 4 the traffic is blocked by the ACL configured on Router 1.



```
Pinging 192.168.30.12 with 32 bytes of data:

Reply from 192.168.30.12: bytes=32 time=2ms TTL=125
Reply from 192.168.30.12: bytes=32 time=4ms TTL=125
Reply from 192.168.30.12: bytes=32 time=4ms TTL=125
Reply from 192.168.30.12: bytes=32 time=4ms TTL=125

Ping statistics for 192.168.30.12:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 4ms, Average = 3ms
```

Figure 5. Ping attempt with access list off

After removing the ACL, the ICMP traffic is allowed to traverse and the host with the IP address 192.168.30.12 can respond to the ping requests which is shown in figure 5.

The networking company Cisco has separated ACLs into two categories: Standard and extended access lists. The standard ACL's filter traffic only based on where the traffic originates from. This could be a specific source IP address or range of source IP addresses. Standard ACLs operate in layer 3 (Network layer) of the OSI model which is responsible for the routing of packets between hosts that are in separate networks. Standard ACLs are usually placed as close to the destination as possible so that it only filters traffic that is going to a specific network. Extended ACLs are superior to standard ACLs because they can filter packets based on source and destination IP addresses, source and destination TCP and UDP ports and protocol types for example IP, ICMP, OSPF and other network layer protocols. This brings OSI model layer 4 (Transport layer) functionality to access lists. Because of the ability to specify what traffic is blocked more accurately it is recommended that extended ACLs are placed near the source of the traffic. Stateless packet filtering firewalls usually filter traffic based on source and destination IP addresses but sometimes also uses source and destination ports in filtering. [2] [3] [4]

2.2.2 Stateful

A stateful packet inspection firewall is considered as the second generation of firewalls. It has higher performance compared to the stateless firewall because it keeps track of connections between two hosts and once the connection is allowed the individual packets are not inspected. A stateful firewall uses state tables to store information about the ongoing connections. To do this stateful packet inspection firewalls utilize OSI layer 4 (Transport layer) and OSI layer 5 (Session layer) functionalities. The session layer functions enable a stateful firewall to record in which stage each connection is.

In general, the transport layer in the OSI model makes sure that data is transferred across the network in a way that a specific application requires. For example, TCP is used if one wants to send a file reliably to another host in a different network because it establishes a connection between the communicating parties. Once the connection is established the data is divided into segments each containing sequence numbers to indicate in which order the data is to be received. If errors are detected or the data is received out of order it is retransmitted to ensure the validity of the data. On the other hand, some applications like VoIP require transmission of data with low latency. In this case the transport layer protocol UDP is used because it does not establish a connection like TCP does nor does it use sequencing or error checking which makes it considerably faster. The port numbers used by transport layer protocols TCP and UDP represent a specific application. Some well-known ports are for example TCP port 80 and 443 which are used for web-browsing. Port numbers can be used in extended ACLs to block specific traffic in this case web-browsing. [5]

At the time when stateful firewalls were developed network applications could be controlled using port numbers. Today it is much more difficult to draw the line if an application on a specific port is used for good or malicious purpose. Also, new applications are developed constantly, and it is impossible for a stateful firewall to identify and block all the required application traffic based on port numbers only. Another known issue with stateful packet inspection is stateless protocols like UDP and ICMP. Unlike the TCP protocol UDP does not send acknowledgement messages between hosts to let them know if the data has been transmitted. Most stateful firewalls handle UDP sessions with timeout values. UDP sessions are tracked by measuring the time since the last packet in a session has been sent. If this idle time in the session exceeds the configured timeout value, the UDP session is closed. All modern firewalls still use a stateful inspection firewall but have other built-in features to support its shortcomings. [2] [3] [4] [6]

2.2.3 Next generation firewall

Describing next generation firewalls (NGFW) is quite challenging because it is a combination of state of the art firewall features and qualities. The ICT researching, and consulting company called Gartner has defined NGFWs as: “deep-packet inspection firewalls that move beyond port/protocol inspection and blocking to add application-level inspection, intrusion prevention, and bringing intelligence from outside the firewall [7].”

On top of the traditional firewall features a next generation firewall has many key attributes that help protect against modern threats. The traditional firewalls were only able to inspect packets based on IP addresses, ports and protocols which left them vulnerable to threats that were residing on the application level for example malware. Next generation firewalls can inspect the IP packets throughout and determine if it contains anything malicious. Also, because applications do not rely on specific port numbers anymore, the application-level inspection of next generation firewalls is even more important. Next generation firewalls are also able to separate different functions of an application. Let’s say that a specific application has many functions such as email, file sharing and instant messaging. If there is a situation where only the email function of the application must be allowed, and others blocked a NGFW is able to do just that. It can also apply quality of service (QoS) or traffic shaping for applications. QoS or traffic shaping are used to prioritize certain applications or protocols so that if the network is under heavy load the prioritized traffic still works normally and is not slowed down.

Another important feature in the NGFWs is the intrusion prevention system (IPS). The IPS is explained in more detail in the section 2.3. It has been possible to implement IPS in earlier stateful firewall solutions, but the next generation firewalls have the IPS engine integrated in to the system which improves the performance and throughput. This leads to the fact that NGFWs have hardware that is specifically designed to process traffic with low latency. Also, NGFWs take advantage of cloud-based databases that have the latest information about malicious attack patterns.

Nowadays a large amount of traffic is being encrypted with the purpose of delivering data securely. Encryption provides confidentiality which means that only the predetermined people that have access to the data can view it. This also opens a window for malicious purposes since the traditional firewalls are not able to inspect the encrypted traffic. NGFWs however can decrypt and inspect encrypted traffic and block the traffic if required but the decryption and encryption also require a lot of processing which means that performance is likely to degrade. There are also some evasive applications that aim to avoid the firewall defenses. An example of these is the TeamViewer application which lets a person take over a remote computers graphical user interface (GUI) to do administrative tasks. This is very useful in most cases but also poses a risk to security. NGFWs must be able to identify and control these evasive techniques. [8] [9] [10]

2.2.4 Unified Threat Management

UTM is short for Unified Threat Management. The aim of unified threat management is to bring all security features into one machine which simplifies the setup and ultimately enhances security. Usually a UTM includes at least a stateful packet inspection firewall, Intrusion detection and prevention system, anti-virus, anti-spam, VPN functionality and web filtering. The drawback in most UTM implementation is that each component is separate from each other and every time a new function is enabled it reserves more CPU usage. This means that the more functions are enabled the more the system performance degrades. UTM solutions are more appropriate for small and medium size businesses because the cost of maintaining the security is lower due to central type of management and the throughput does not suffer as much because there are less users. [11]

2.3 Intrusion detection and prevention

An intrusion detection system (IDS) is a mechanism for monitoring traffic that flows through a networking device. An IDS is responsible for logging and reporting if the traffic contains malicious patterns. There are two approaches to how an IDS determines whether traffic is malicious or not: knowledge-based and behavior-based detection. Knowledge-based intrusion detection uses a database of signatures that match to known malicious data patterns. This method requires constant upkeep since every time a new malicious pattern is created the database must be updated for the traffic to be detected reported by the IDS. The traffic not matching the patterns is considered safe. The other approach to intrusion detection is the behavior-based or anomaly-based detection. In this approach a baseline is configured which is considered normal traffic and anything that deviates from this is suspected of being a threat. The anomaly-based detection produces more false-positives but on the other hand it can detect threats that are previously unknown. A false-positive is when an IDS starts alerting for traffic that is legitimate.

IDS's operate passively. This means that all traffic that passes through the network is mirrored and the IDS analyzes traffic that is a copy of the original traffic. This way the IDS does not affect the network performance because the traffic is not inspected before but after it is forwarded to its destination. This also means that an IDS lets malicious traffic pass through. The intrusion prevention system (IPS) operates much like an IDS but is designed to prevent malicious traffic from entering the target network. To be able to do that an IPS functions in active state rather than passive which means that it inspects all traffic before it is forwarded. This can have a negative effect on the network flow since the inspection of every single packet entering or leaving the network takes quite a lot of processing. Like IDS, IPS can operate in either knowledge-based or behavioral-based manner. The intrusion detection and prevention can be either host-based or network-based. Host-based intrusion detection and prevention aims to protect a single device from threats that exploit vulnerabilities in the operating system or other components whereas the network-based protects the entire network. It is often a good idea to implement both. [10]

2.4 Virtual private network

A virtual private network (VPN) is an encrypted tunnel that carries traffic securely across public networks. VPNs can be configured using many different protocols, but the most common protocols today are IPsec and SSL or TLS. VPNs can be used to connect individual clients to a private network (remote-access VPN) or to connect two private networks (site-to-site VPN) over the public internet. VPNs bring enhanced security by authenticating users connecting to a specific private network over the internet and by encrypting the traffic flowing through the tunnel. VPNs are easily scaled if more users are required. VPNs are compatible to almost every operating system and enabling remote access of clients helps reduce costs and increase productivity.

2.4.1 IPsec VPN

The Internet Protocol Security (IPsec) is an OSI Model network layer standard that defines what measures must be taken to ensure that a VPN is secure. IPsec itself has two protocols available: Authentication header (AH) and Encapsulating Security Payload (ESP). AH is only used for authentication and integrity and lacks the ability to encrypt traffic. ESP also encrypts traffic and therefore it is superior to AH. The main things that must be considered when securing a VPN are confidentiality, integrity and authentication.

Confidentiality is ensured with encryption. Common protocols that provide encryption in VPN communications are DES, 3DES and AES. These encryption algorithms are symmetric which means that both parties that are communicating with one another are using the same secret key to encrypt and decrypt the data. AES has superseded the older DES and 3DES algorithms and it is recommended that DES and 3DES are not used anymore.

Integrity ensures that data sent in VPN communications is not altered during transit. This is done by creating a hash from the data before sending it to its destination. When the data is received a hash is created from the same piece of data using the same hashing algorithm and this way if the two hashes match the data has not been changed during transmission.

Authentication must also be implemented before an IPsec VPN connection can be established. Authentication is a way of making sure that only the legitimate parties have access to VPN services configured for them. There are many different options for authentication in IPsec. Some of the more popular ones are digital certificates, pre-shared keys, one-time passwords, biometrics and basic username and password.

In addition, IPsec contains a method for securely exchanging the keys used to encrypt and decrypt data. The algorithm used for this key exchange is Diffie-Hellman (DH). The DH algorithm has groups to indicate how strong the generated DH key is which will be used to encrypt the actual pre-shared key that will be used to encrypt the real data sent through the IPsec VPN tunnel. The recommended DH groups to be used are DH14-16, DH19-21 and DH24. The more bits the key has the more secure it is, but it also takes longer to generate it. DH uses public key exchange to deliver the secret keys. In public key exchange both the sender and the receiver have a private and a public key. Say that data is sent from A to B. To securely send information A will use B’s public key to encrypt the data to be sent. When the encrypted data is received by B it can decrypt the data using its own private key. Likewise, if B sends data to A, B will encrypt the traffic using A’s public key and A will decrypt it using its private key.

Before any traffic starts traversing the IPsec VPN tunnel all the above parameters must match between the sending and the receiving end. All the information that is required to establish an IPsec VPN connection are stored in databases called Security Associations. The SAs negotiate the parameters between the two endpoints and generate and exchange the keys to be able to pass this information to each other securely. SAs also handle the authentication between the endpoints. To establish the connection SA’s use protocols called Internet Security Association and Key Management Protocol (ISAKMP) and Internet Key Exchange (IKE). IKE works in conjunction with ISAKMP to authenticate the sender and receiver and create a secure tunnel which is used to exchange information that will be used for the actual user data communications. [12]

2.4.2 SSL VPN

The SSL VPN uses the Secure Sockets Layer (SSL) or Transport Layer Security (TLS) that are supported by all modern web browser applications. The SSL VPN connection is initiated using a web browser and it does not require a separate client software to be downloaded to the client making it a “clientless” VPN. The clientlessness allows for good compatibility because virtually any machine can initiate the connection if it has a web browsing application installed. The SSL protocol provides private data communications, authenticity and data integrity. When the client prepares to initiate the connection to the SSL VPN host an SSL handshake will take place. During the SSL handshake the SSL version number, encryption algorithms and the session-specific data are negotiated. The SSL handshake also includes the authentication process where certificates are used to verify the identities of the client and the host. The handshake also includes the generation and exchange of shared private keys which will be used to encrypt the traffic moving in the SSL VPN tunnel. Figure 6 demonstrates the steps of a SSL handshake in the chronological order. [12] [13] [14]

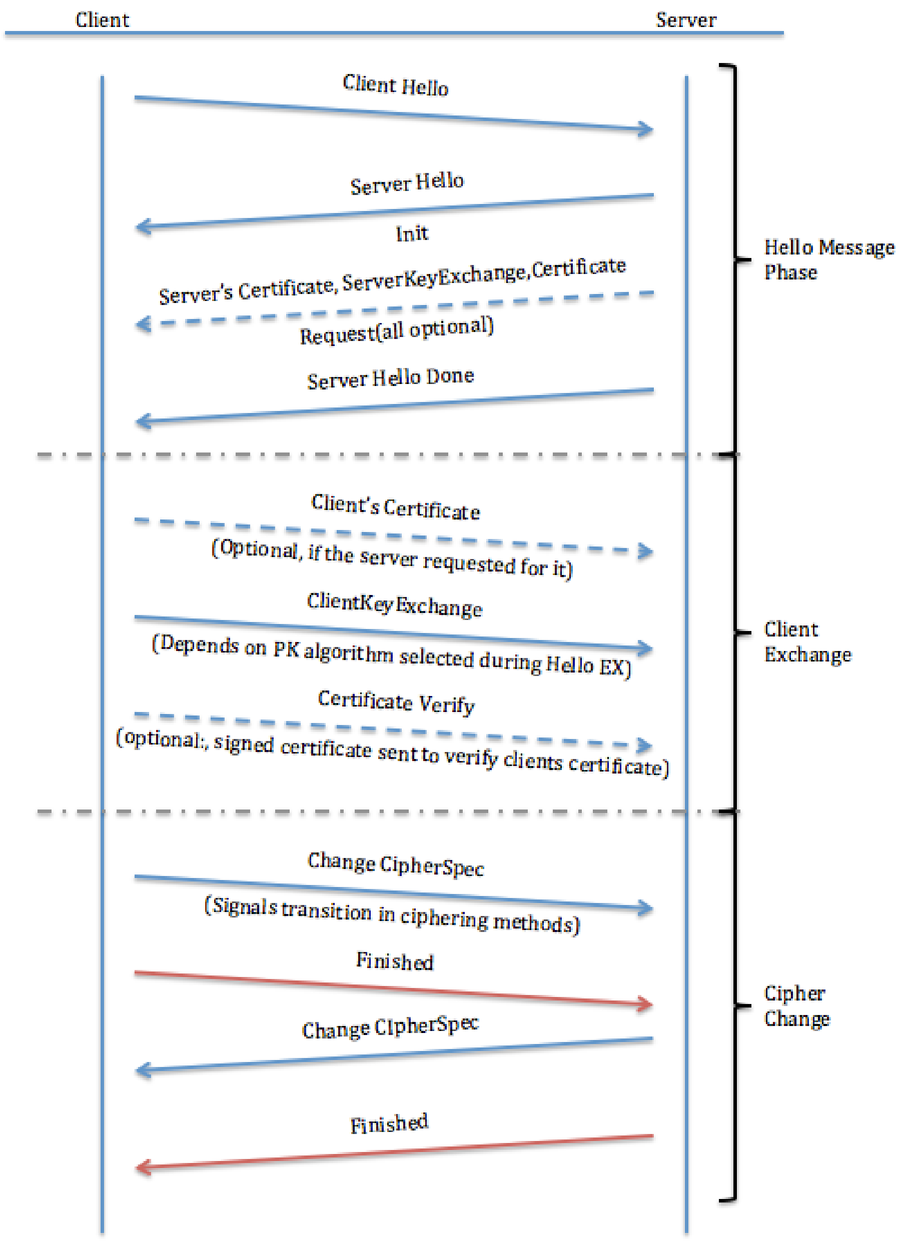

Client
Server
Client Hello
Server Hello
Init
Server's Certificate, ServerKeyExchange,Certificate
Request(all optional)
Server Hello Done
Hello Message Phase
Client's Certificate
(Optional, if the server requested for it)
ClientKeyExchange
(Depends on PK algorithm selected during Hello EX)
Certificate Verify
(optional:, signed certificate sent to verify clients certificate)
Client Exchange
Change CipherSpec
(Signals transition in ciphering methods)
Finished
Change CIpherSpec
Finished
Cipher Change

Figure 6. SSL handshake

2.5 Dynamic Host Control Protocol

The Dynamic Host Control Protocol (DHCP) is used for automated IP address assignment for network clients. The DHCP sends networking information to clients to enable them to communicate with other clients on the internal or external networks. DHCP has several options for assigning IP addresses to clients. Most clients get their IP addresses through dynamic allocation which means that the client gets an IP address from a pre-configured pool of addresses. The dynamic allocation gives or “leases” IP addresses for a specific amount of time and when the time runs out or the client does not need the address anymore it is freed and can be used by another device on the network. The amount of time an IP address is reserved for a client after leasing it is known as lease time.

Another method for delivering IP addresses to clients is the automatic allocation. This is where IP addresses are also automatically assigned from a pool of addresses but instead of leasing them for a specific amount of time they are assigned statically which means that they do not change. There is also a possibility to manually allocate addresses for clients. Manual allocation might be used if a specific device requires a static IP address. Static IP addresses might be needed if there is a service running on it that must be reachable from the outside networks for example a web server.

The DHCP IP address assignment happens in four steps. First the client sends a DHCP discover message to the network as a broadcast. The broadcast message is sent to all possible destinations in the local area network. When the message is received by the DHCP server, it reserves an IP address from the pool to the client requesting it. The DHCP server then sends the a DHCP offer message to the client as a unicast to the clients MAC address. As the client receives the DHCP offer it then decides if it accepts the offered networking information. When the client accepts the information, it sends a DHCP request message back as a broadcast. If there is more than one DHCP server in the network the DHCP request message will inform about declining the other DHCP offers to the servers. This ensures that only one IP address is assigned to the client. The DHCP request message is also used to renew an existing IP address lease. In the last step the DHCP server sends a DHCP acknowledgement message and pings the IP address assigned to the client to make sure that the IP address is not already reserved. When the client receives the acknowledgement, it checks if there is an IP assigned to its own MAC address and starts using it. The subnet mask and default gateway information are also provided by the DHCP server.

If a DHCP server that is outside the LAN must be used, a DHCP relay or IP helper address must be configured. Normally the DHCP messages are sent within a specific broadcast domain but by using the DHCP relay function a DHCP server can be used that is located outside of the local network. The DHCP relay is the IP address of the external DHCP server and it is usually configured on a router or a firewall. [15]

2.6 Domain Name System

The Domain Name Service (DNS) is a system that is used to resolve computer hostnames that are in a more human-friendly format to numeric IP addresses. DNS servers are responsible for the name resolution and when a client for example visits www.google.com a DNS query is sent asking for the IP address of this hostname. The DNS query will advance in specific steps to resolve the fully qualified domain name. First the computer operating system and the web browser sending the query checks its own cache memory to resolve the IP address. The DNS cache is a database that is used to save recently queried DNS records for a specific amount of time so that if those should be queried again the process would be faster. If the IP address cannot be resolved this way the DNS query is forwarded to a DNS resolver. The DNS resolver might also have a DNS cache configured which will also be checked before the DNS query is forwarded. If the IP address is not resolved with the information stored in the cache the DNS resolver will initiate a query to DNS root servers. There are 13 root servers located around the world. The root servers are responsible for directing the DNS resolver to the correct authoritative servers and they start by querying the top-level domain (TLD) for example .com, .org or .au. The hostname is dissected from right to left and step-by-step the DNS resolver is directed closer to the fully qualified domain name (FQDN). When the FQDN has been resolved it is forwarded to the client that initially started the DNS query. FQDN objects can be created in firewalls to deal with changing IP addresses. When an address object is created using an FQDN the object remains valid even if the IP address of the FQDN would change. [16] [17]

2.7 Proxy server

Proxy servers have many features that make them useful in a variety of different situations. Because a big part of traffic today is web-based in home and office environments there is a need to filter it to block malicious content. Web proxy servers are used for this purpose and to be able to filter web traffic all the client web traffic initiated by the clients must be forwarded through a web proxy server. For example, if a client sends a request to open a web page it is first forwarded to the web proxy which then makes the request for the client. Once the web page is received from a web server the proxy forwards it to the client.

A web proxy can filter traffic in many ways. On most proxies there are preconfigured categories for websites. The categories can be used to block specific type of websites for instance gambling, violence or advertisements. Websites can also be manually blacklisted (blocked) or whitelisted (allowed) or custom blacklists can be imported on some proxies which include well known malicious websites. Clients also tend to download content from the internet. Web proxies can be used to block users from downloading certain risky file types for example executable files that might be used by hackers to compromise computer systems. Another method for restricting web access is to filter IP addresses or computer hardware addresses also known as MAC addresses. Web access can be blocked for specific hosts or complete networks. Time restrictions can also be applied so web access can be allowed for example from 8 am to 9 pm every day or just specific weekdays.

Web proxies can be configured to operate in two different modes. The normal approach to configuring a web proxy for network clients is to configure the web proxy parameters manually to the client’s web browser in the settings. For a savvy user this is easy to bypass unless the browser settings are only configurable by administrator accounts. The other mode of operation is the transparent mode. When a web proxy operates transparently the proxy settings do not have to be manually configured for the client web browser. The transparent mode is invisible from client’s perspective.

Other common proxy features are caching, monitoring of web traffic and providing anonymity for clients accessing the web. A caching proxy temporarily saves information the clients have requested from the web recently. This action improves the network performance because if the clients ask for the same web resources repeatedly the proxy does not have to request the content again and again instead it is saved in its cache. Proxies are also often used to monitor web traffic. In many countries it is mandatory to inform the clients about their web sessions being monitored. Useful information gathered through monitoring could be the IP addresses that have been blocked for trying to access restricted websites. On some occasions users want to ensure their anonymity when accessing the web. A web proxy can do this by altering the client requests so that they cannot be tracked back to the original source. [18]

2.8 Network Time Protocol

In networking there are many services that require the synchronization of clocks for them to function correctly. The network time protocol (NTP) is used to synchronize clocks in computers and servers over the public internet. The common implementation includes a NTP server which clients use to find the most accurate time available. The NTP servers themselves use a hierarchical system where the source of the initial time values is measured by atomic clocks, GPS or other radio clocks that are known as reference clocks. Reference clocks are stratum 0 devices which is considered as the first tier of clocks. The stratum scale goes up to 15 and stratum 16 is considered as unsynchronized. NTP servers that are in a specific stratum use other same tier stratum servers and one tier higher stratum servers to keep their own clock synchronized. For example, a server that is in stratum 4 uses stratum 4 and 3 servers for synchronization of its own clock. NTP has algorithms that are designed to take network errors and latency into account. NTP traffic uses UDP protocol for delivery of time information in port 123. UDP protocol is optimal for NTP because it provides lower latency. [19]

2.9 High Availability

High availability (HA) is a common method for providing redundancy with firewalls. It is implemented as a high availability cluster with two or more firewalls connected to each other. The most common implementation of high availability includes two firewalls where one firewall is the active device handling all the traffic that flows in and out and the other firewall is in passive mode which means that it is constantly polling the active device for the event of failure. If the active device stops responding to the heartbeat messages sent from the passive device, it is considered to be down and the operation switches to the passive device. To make a seamless switch from the active firewall to the passive firewall the passive firewall must hold information about all the connections that are ongoing in the active device. The passive device also must share the same configuration with the active device for the switchover to be successful. The synchronization of data is often done on dedicated links between the HA-members. Another possible implementation is an active-active configuration where both firewalls are processing traffic. Both devices in the HA-cluster maintain session tables and routing information by synchronizing with each other. In active-active mode the communication sessions are divided between the HA-members which makes recovering faster. It also works as a method for balancing traffic between two devices making active-active implementation better at handling high traffic loads.

Protocols called VRRP and CARP were used in this thesis to implement high availability. VRRP stands for virtual router redundancy protocol and common implementations include two devices with and active-passive setup. In VRRP a virtual router is configured to act as the gateway for a network and traffic that leaves the LAN is forwarded through it. The active device also known as the master is decided by configuring priority values and the device with a higher priority will become the master. All traffic is forwarded to the master device and if it fails the backup device takes the role of the master while the gateway of the network remains the same. The master device sends multicast packets to other devices configured with VRRP and if the backup devices stop receiving these messages they know the master device is down and a new election for a master device begins. The common address redundancy protocol (CARP) uses the same idea and the same methods to provide redundancy. [20] [21] [22]

3 Background and planning

3.1 Hardware

The hardware used in this thesis to test the open source security appliances was mostly provided by Metropolia. The same hardware was used to test both IPFire and OPNsense to provide realistic test results. The firewall hardware in this case was an HP EliteBook 8470p notebook computer. The system specifications for this laptop are displayed in table 1.

Table 1. HP EliteBook 8470p hardware specifications

Processor	Intel(R) Core(TM) i5-3320M CPU 2.6GHz
Memory	8192MB RAM (2 x Samsung 4096MB RAM @ 1600MHz)
Network interface card	Intel® 82579LM Gigabit Ethernet Controller
Power adapter	Input AC 120/230 V (50/60 Hz) Output 65-Watt, 18.5 V

In addition to the firewall hardware itself two NICs were required to be able to install the firewall to the network edge between the internet and the internal private network and to provide a DMZ zone to the test environment. For this purpose, A-Link NA1GU USB network adapters were used which have 1Gb throughput and support for USB 2.0. The first network adapter was the interface for the inside and the second was for the DMZ network. For the testing of the firewall, two hosts were also needed that would act as a server and a client. To accomplish this, two home PCs were used. The PC hardware is also capable of providing 1Gb throughput, so they suited this thesis well and were not causing a bottleneck to the testing environment.

Overall the hardware used in this thesis is not optimal for firewalling purposes. The laptop that does the firewalling is designed to be an office workstation rather than processing the traffic flows leaving or entering a network. Since a firewall device must be able to process tens of thousands of packets per second it needs a custom designed architecture to serve its purpose. The components that are most heavily used in a firewall are usually the processor, memory and the NICs. The components must be designed to provide minimal latency to ensure that all traffic can pass through without causing networks to congest. Firewall vendors have specifically designed processors that differ from consumer processors in many ways. For example, some firewall vendors use a processor architecture in their products where the data processing and the firewall management are separated so if the firewall becomes flooded with traffic the management will still work as usual. The laptop serving as a firewall in this thesis has decent processor and memory resources, but the biggest downside are the NIC's that provide only 1 Gb throughput. In a small environment this would be acceptable but if the environment would grow larger this would most likely be the first bottleneck this laptop would face.

3.2 Choosing firewall distributions for testing

When searching for the firewall distributions for this thesis multiple things were considered that would affect the final decision. To make the thesis as comprehensive as possible it was desirable to find two firewall distributions that would differ from each other as much as possible while still being feature-rich and reliable. The popular firewall distributions were mostly based on FreeBSD, OpenBSD or Linux operating systems. Out of curiosity and the desire to see different results in testing the firewall distributions were chosen to represent different operating systems.

Originally the idea was that one could choose any old or new excess computers lying around and turn them into functioning firewalls with modern security features. This means that the distributions must be compatible with a variety of different hardware. Practically this means that the firewall distributions should support 32 and 64-bit computer architectures from different manufacturers.

Another important factor that was considered was how easy it is to install, configure and maintain the firewall. There is always a chance that someone could make a humanly error and the chance of that happening reduces if the firewall has good documentation and instructions as well as simple configuration. The importance of a good documentation is often underestimated, and it can help recover from many problematic situations. It also helps security administrators to understand what they are doing and what effects their actions have on the firewall system. Simple configuration and maintenance with good documentation ultimately makes a firewall system less vulnerable to security threats.

The features provided by the firewall distribution also played a role in the decision making. The firewall distributions had to fit for many different purposes, and that is why distributions that had a lot of features were favored. Although having more features enabled on a firewall increases its attack surface and makes it more vulnerable it also has more use cases and fits into different environments.

Based on these requirements OPNsense and IPFire were chosen for this thesis. Before the decision was made the internet was researched for previous evaluations of open source firewalls. All the reviews that were found about these distributions were very positive and they were in many cases rated among the top five best firewall distributions. [23]

4 Testing

4.1 Testing environment

The testing environment in this thesis included a client, a server and a firewall. The server was placed in the LAN which is also considered as the inside network. Depending on the test the client was either in the DMZ zone or the outside zone.

4.2 Properties to be evaluated

Before testing there had to be careful thought to what properties should be evaluated in a firewall. From a business point of view one of the most important things is the availability of services. The cost of downtime becomes higher as a company grows and large companies would most likely suffer huge financial losses for even minutes of network services downtime. To counter this, it is recommended for companies to set up high availability clusters so that if a device failure occurs another device acting as a backup would take its place. This function should be tested in regular intervals for the event of failure.

The critical firewall security mechanisms should also be tested thoroughly before putting a firewall into production. Nowadays in addition to the traditional packet filter other security features are becoming more important as the nature of network traffic and attack vectors are changing. Features like IDS/IPS, web filter, antivirus, application detection and SSL decryption were developed to support the traditional network security. If these features are enabled to enhance security, there is a good reason to test if they function as they should and how these features affect the system operation. Many features have different kinds of drawbacks that must be considered before enabling them.

As the network environments grow larger the administrating tasks become more time consuming. Automating some tasks that must be carried out regularly makes the life of an administrator a lot easier. Automating tasks helps keep the system up to date. Automation is also useful for creating backups that can be used to recover from device failures. Making sure the firewall and its features are up to date and that the backups are working is something an administrator must keep a close eye on.

Another subject for testing is the usability of a firewall device. When the amount of networks, addresses and security policies on the firewall is getting big it is important for an administrator to find exactly what he needs with little effort. Address objects and address groups are crucial to keep a firewall organized and optimized. These make management easier because an administrator can always refer to the same address object and if changes are made it affects the whole system so every security policy that contains the address object would be updated to correspond to the new IP address or network. Address objects can be grouped to create address groups. The same way one can create address groups, services can be grouped together to make service groups. For example, a service group for web traffic would be created by adding ports 80 and 443 to make a group. The main thing that creates a good user experience is the UI of the firewall. Everything that an administrator needs to perform daily tasks like viewing logs or making changes to the security policies must be easily accessed from the main page. The structure of the UI should be clear and different functions should be grouped logically. To find specific information fast the UI has to include elaborate search engines. For example, when looking for connection attempt in the logs is it possible to filter the logs by port/protocol, source/destination IP addresses, time window etc. A firewall provides tons of information about the system itself. When viewing the health of the firewall hardware, entities like CPU usage, memory usage, hardware temperatures and traffic flows are measured. How these numbers are displayed on the UI and how accurate and customizable these graphs are affect the usability and define how useful the data really is.

Support and documentation are relevant in every phase of a firewall device lifecycle. Whether it is testing and piloting before deployment, installing into a production environment, maintaining the system or running down old hardware the documentation is there to support every process. Sometimes it happens that everything does not go according to plan, and therefore support and documentation should be evaluated when choosing firewalls.

Since firewalls eventually protect the wealth of businesses it is a good thing to look at the power consumption of a firewall device. In bigger companies this might not be a high priority but for smaller firms it’s a thing worth noting.

4.3 Firewall testing methods

Iperf was used in this thesis to measure the throughput of the firewall. Iperf requires two hosts for testing where one host acts as a server and the other as client. To measure the throughput the client was placed in the internet side and the server located in the LAN. In addition to measuring throughput Iperf can be used to measure the link quality as well. Values like jitter, latency and datagram loss can be measured to get an overview of the link quality. It is also possible to send both TCP and UDP traffic with this tool and it supports common protocols like IPv4 and IPv6. Iperf works on multiple platforms as well. [23]

To see if the firewall can detect applications passing through, traffic had to be sent through a port that it normally doesn’t use. This test also required a client and a server. The application detection was tested by having an SSH server listen to port 5001 and using the client to connect to the server on that port using the SSH protocol. SSH traffic uses port 22 by default and the firewall rules were configured to drop SSH traffic. The

SSH traffic should not be able to pass through the firewall no matter what port it uses if it is able to detect application level traffic.

The testing of the antivirus functionality in the firewall was fairly simple. After setting up the proxy server and integrating the antivirus with the proxy the antivirus was tested by downloading the Eicar test files. The Eicar test files contain different file types and files that are encrypted. If the antivirus is functioning correctly it will display a page indicating that a virus has been detected when there has been an attempt to download the Eicar test files. [24]

The web content filter test method was similar to the antivirus test. Once the web filter was correctly set up and the categories were configured to block for example advertisements the access to the blocked sites could be tested. Obviously, the web filter should display a page that informs a user that the site is being blocked by an administrator if it works.

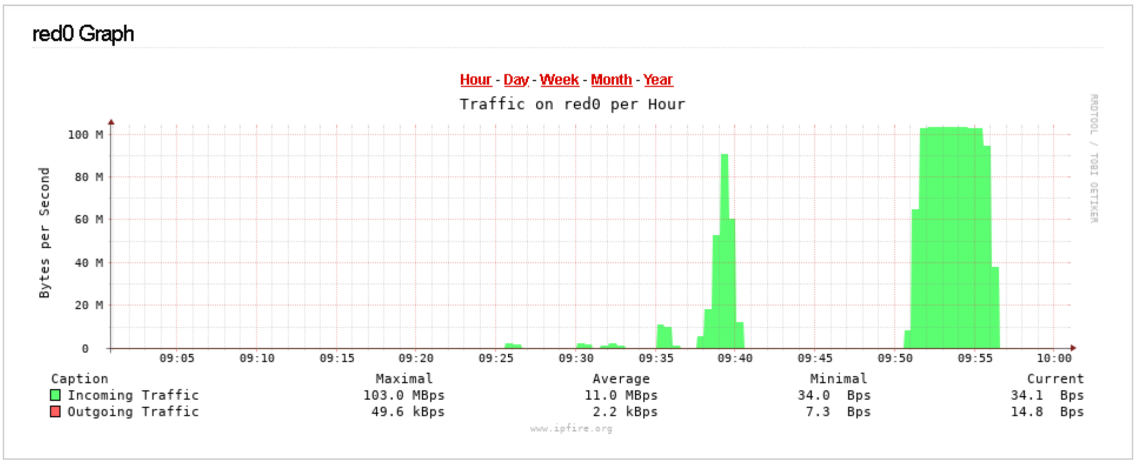
The IDS and IPS functions were tested by using the Nmap software which is used for scanning networks and detecting vulnerabilities. Nmap was used to perform two types of port scans to see if the firewall distributions can detect the scans. The first type of port scan was a TCP SYN scan where Nmap sends TCP SYN messages to the ports being scanned and seeing if they respond. This way the connection between the Nmap and the hosts being scanned is never established and Nmap receives the information if ports are open on the system. The second type of scan uses a technique where the TCP segment is modified to use alternative flags that will inform the scanner if the scanned port is closed. Otherwise if the port is open no response is sent back to the scanner. This scan is more commonly known as the Xmas scan. [25]

As an additional test to see how the firewall distributions would react to high amounts of traffic a DoS attack was executed to the WAN interface of the firewall. A tool called Hping3 was used for this purpose. The Hping3 was installed on a host that located in the outside network. The tool was used to generate packets at a very high rate to flood the WAN interface of the firewall with traffic. To increase the amount of bandwidth usage Hping3 was used to add 1000-byte payload data to the packets sent in flooding purposes. To see the effects of the DoS attack, a ping test was set to run on the background to give information about the latency and to see if traffic is dropping. Also, a file download was attempted during the DoS and a video stream was open to see if the video feed would have errors. [26]

4.4 Test results

4.4.1 IPFire

The throughput test results on IPFire were surprisingly good. The components used in this thesis allowed for a maximum of 1 Gb throughput and IPFire was able to come very close to it. Figure 6 shows the actual test result.

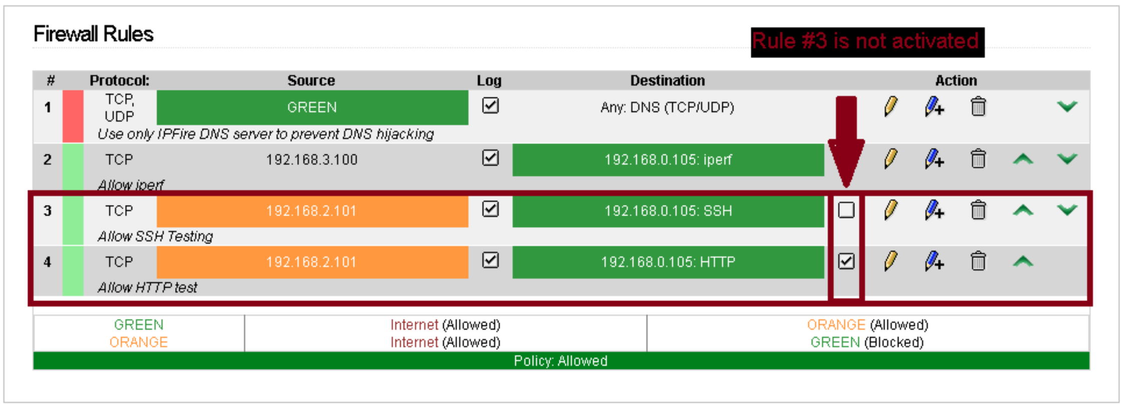


```
Caption                         Maximal        Average        Minimal        Current
🟩 Incoming Traffic            103.0 MBps      11.0 MBps      34.0  Bps      34.1  Bps
🟥 Outgoing Traffic             49.6 kBps       2.2 kBps       7.3  Bps      14.8  Bps
```

Figure 7. Throughput on IPFire

Figure 7 shows that the throughput peeked at 103 MBps which is 824 Mbps since 1 byte is 8 bits and 103 MBps multiplied by 8 is 824 Mbps. As the theoretical maximum being 1000 Mbps (1Gb) with IPFire the results didn't fall short by much. Enabling the IPS did not affect the throughput on IPFire but the CPU usage increased by approximately 20%.

Application detection is one of the features that is associated with NGFWs. Therefore, it is no wonder that IPFire eventually failed the application detection test. While testing the application detection the client with IP address 192.168.2.101 was used to connect to the SSH server with IP address 192.168.0.105 over port 80. By default, all traffic from the client is blocked to the SSH server in this case. In figure 8 there is a rule that allows HTTP traffic to pass to the destination and by default all else traffic is blocked.



Firewall Rules

Rule #3 is not activated

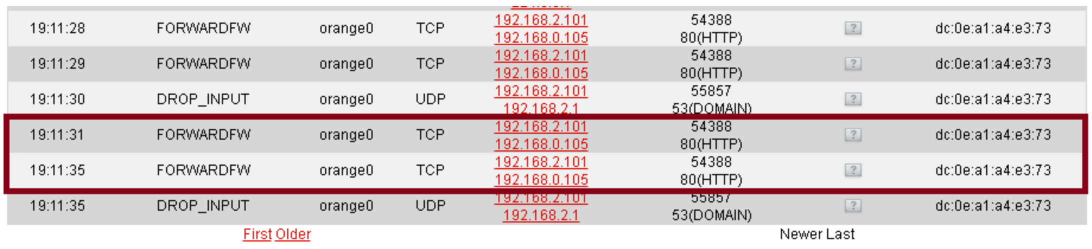
#	Protocol:	Source	Log	Destination		Action
1	TCP, UDP	GREEN	☑	Any: DNS (TCP/UDP)		
	Use only IPFire DNS server to prevent DNS hijacking					
2	TCP	192.168.3.100	☑	192.168.0.105: iperf		
	Allow iperf					
3	TCP	192.168.2.101	☑	192.168.0.105: SSH	☐	
	Allow SSH Testing					
4	TCP	192.168.2.101	☑	192.168.0.105: HTTP	☑	
	Allow HTTP test					

GREEN ORANGE	Internet (Allowed) Internet (Allowed)	ORANGE (Allowed) GREEN (Blocked)

Policy: Allowed

Figure 8. Rule 4 allows HTTP traffic from the DMZ zone to the inside zone

To see if the firewall will block SSH traffic, an SSH connection was initiated from the client using the allowed port 80. The result is demonstrated in figure 9.



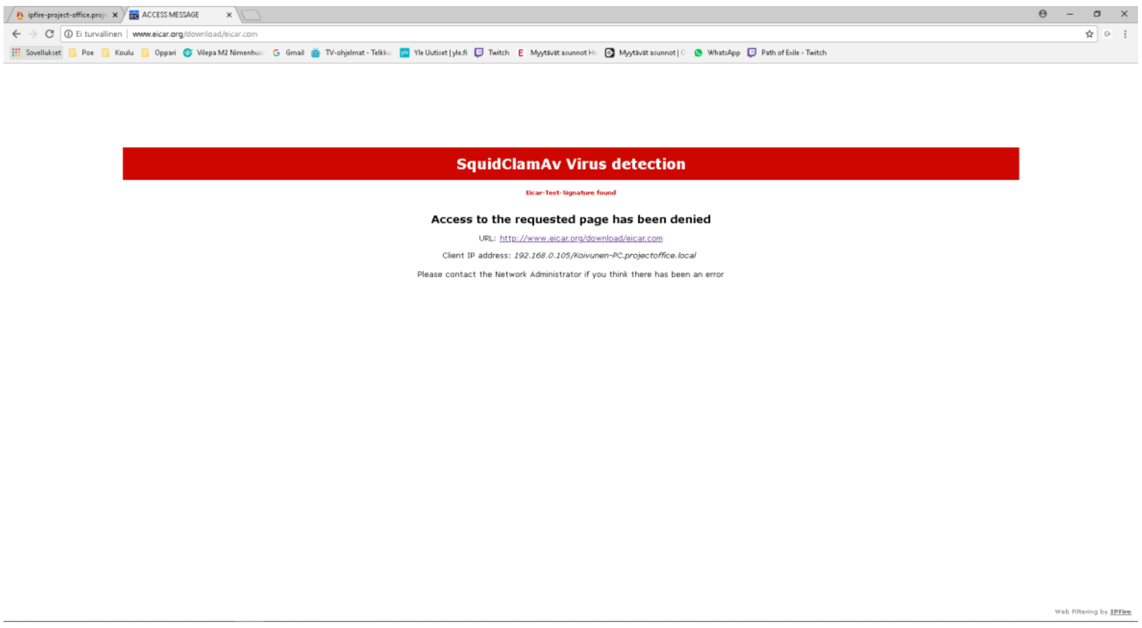
19:11:28	FORWARDFW	orange0	TCP	192.168.2.101 192.168.0.105	54388 80(HTTP)	?	dc:0e:a1:a4:e3:73
19:11:29	FORWARDFW	orange0	TCP	192.168.2.101 192.168.0.105	54388 80(HTTP)	?	dc:0e:a1:a4:e3:73
19:11:30	DROP_INPUT	orange0	UDP	192.168.2.101 192.168.2.1	55857 53(DOMAIN)	?	dc:0e:a1:a4:e3:73
19:11:31	FORWARDFW	orange0	TCP	192.168.2.101 192.168.0.105	54388 80(HTTP)	?	dc:0e:a1:a4:e3:73
19:11:35	FORWARDFW	orange0	TCP	192.168.2.101 192.168.0.105	54388 80(HTTP)	?	dc:0e:a1:a4:e3:73
19:11:35	DROP_INPUT	orange0	UDP	192.168.2.101 192.168.2.1	55857 53(DOMAIN)	?	dc:0e:a1:a4:e3:73

First Older Newer Last

Figure 9. SSH connection allowed over port 80 (HTTP)

The firewall log indicates that the SSH traffic that is forwarded through port 80 is allowed by the firewall which means that the firewall cannot detect application level traffic.

The antivirus on IPFire was implemented by installing an add-on called ClamAV. This is a free antivirus functionality which is installed and configured to work hand in hand with the proxy server on IPFire. IPFire partially passed the test by only being able to detect viruses that were not encrypted. Figure 10 demonstrates an example of a block page displayed when a virus is found.



SquidClamAv Virus detection

Eicar-Test-Signature found

Access to the requested page has been denied

URL: http://www.eicar.org/download/eicar.com

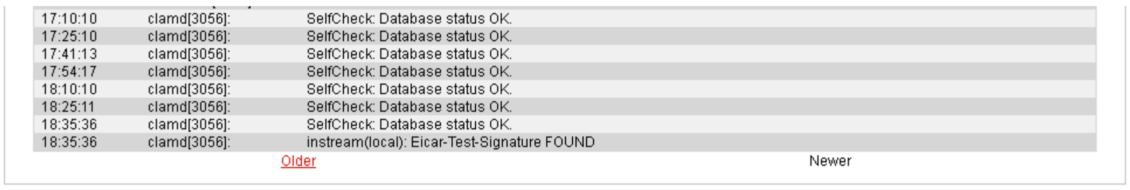
Client IP address: *192.168.0.105/Koivunen-PC.projectoffice.local*

Please contact the Network Administrator if you think there has been an error

Web Filtering by IPFire

Figure 10. Block page after downloading an unencrypted Eicar test file

Upon downloading the unencrypted Eicar test file IPFire also saves a log entry of the event as shown in figure 11.



```
17:10:10    clamd[3056]:    SelfCheck: Database status OK.
17:25:10    clamd[3056]:    SelfCheck: Database status OK.
17:41:13    clamd[3056]:    SelfCheck: Database status OK.
17:54:17    clamd[3056]:    SelfCheck: Database status OK.
18:10:10    clamd[3056]:    SelfCheck: Database status OK.
18:25:11    clamd[3056]:    SelfCheck: Database status OK.
18:35:36    clamd[3056]:    SelfCheck: Database status OK.
18:35:36    clamd[3056]:    instream(local): Eicar-Test-Signature FOUND
```

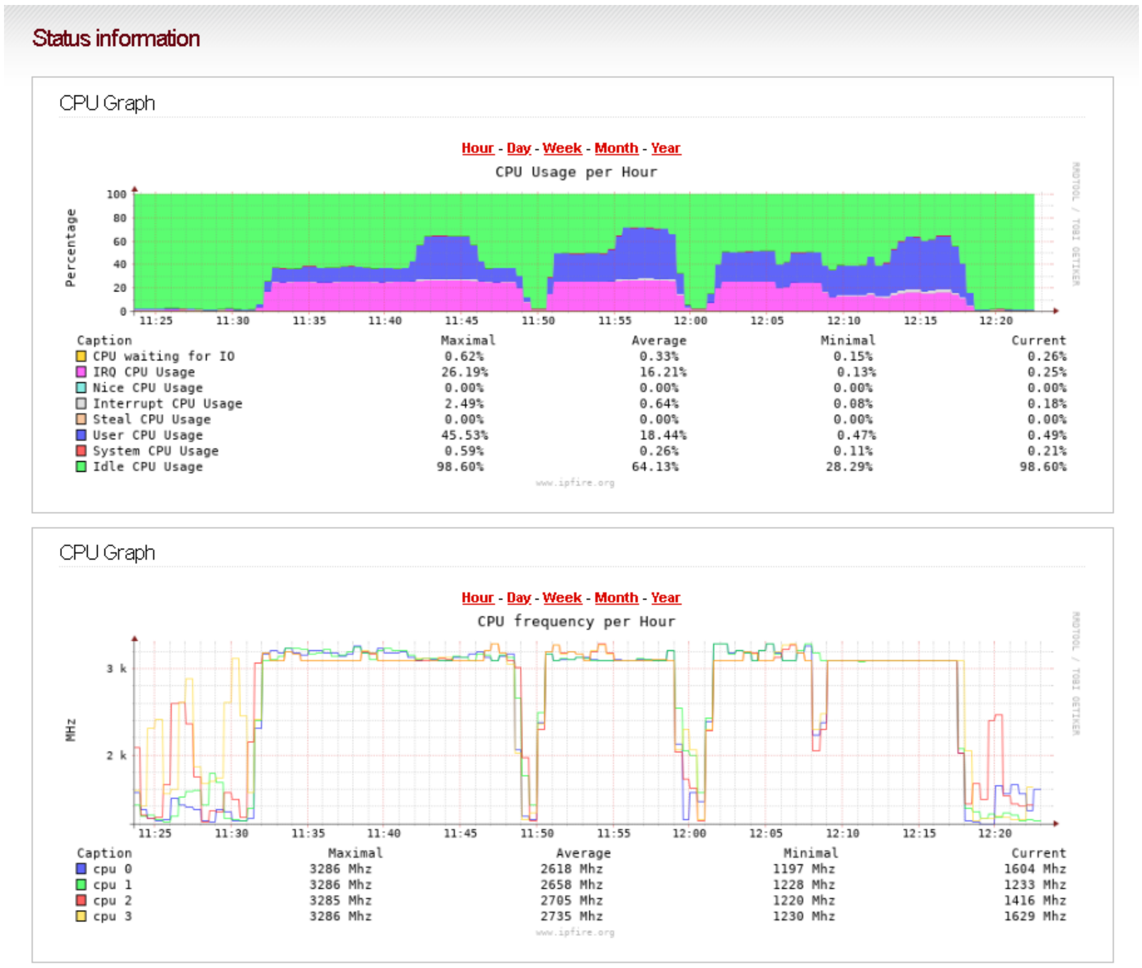
Figure 11. ClamAV log file entry

However, the encrypted Eicar test files were not detected by the IPFire antivirus.

The next step was to find out if the IPFire was able to filter web content and allow access to only the categories that are required. The web content filter on IPFire can operate in transparent mode and by manually configuring the settings from the web browser to enable filtering. The transparent web filtering basically means that the user surfing the internet has no knowledge if the web traffic is being filtered or monitored. The other option is to manually configure the browser to use the web proxy. The test result was identical to the antivirus test result as the IPFire web filter could only filter web content that was not encrypted. This means that the IPFire is not able to filter out most of the web traffic because majority of web traffic today is in an encrypted format for example HTTPS.

The IDS and IPS testing proved that IPFire is not capable of detecting common attack patterns. The IPS on IPFire had signatures that should block SYN scans, DDoS and SSH servers using non-standard ports. None of these showed up on the IPS log files after testing. Also signatures for blocking DoS attacks and detecting SSH on non-standard port were enabled on IPFire. Neither of these attack patterns were detected when they were tested.

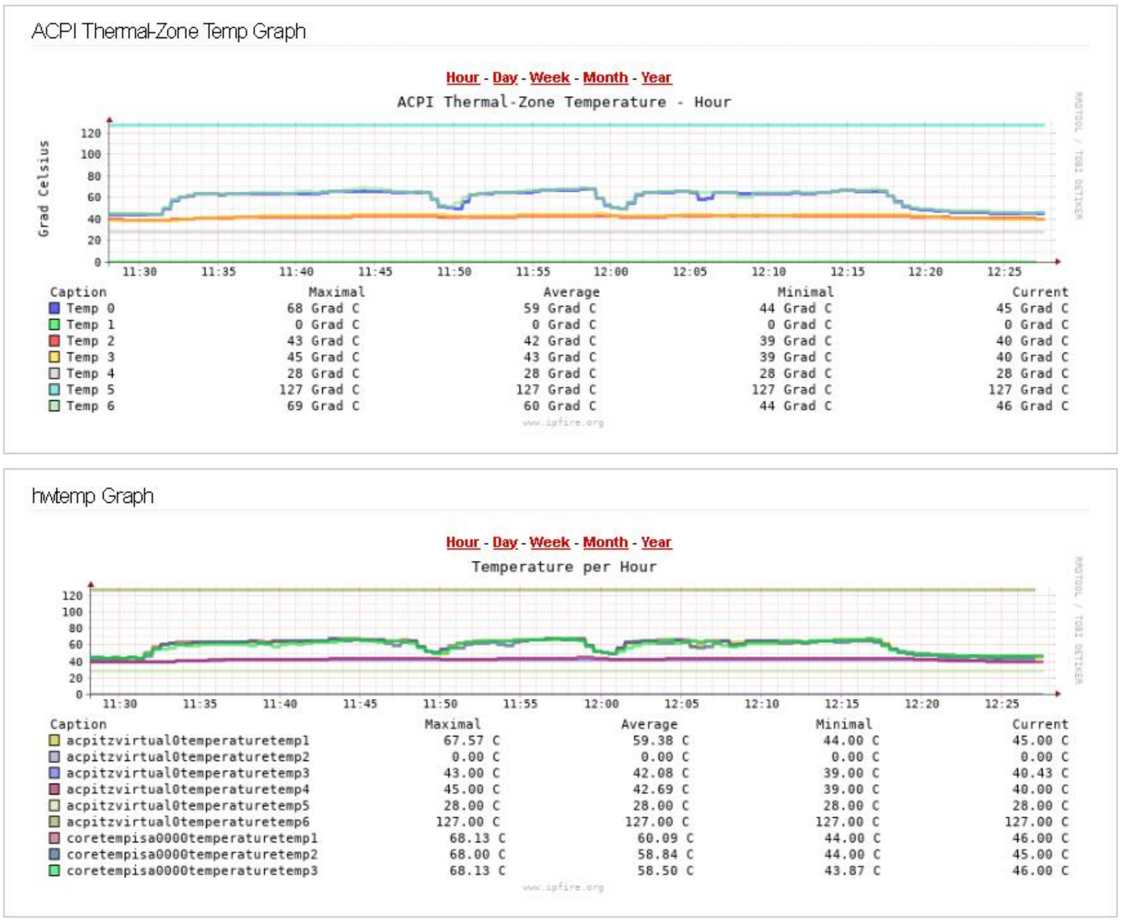
The stress test carried out on the IPFire came out as expected. The DoS attack focused on IPFire drained the overall system resources and started affecting the network negatively. The CPU was under heavy load during the test and it started to affect the hardware operating temperatures. Only the memory usage seemed normal throughout the entire test. The below graphs illustrate the effects of the DoS attack.



Caption	Maximal	Average	Minimal	Current
🟨 CPU waiting for IO	0.62%	0.33%	0.15%	0.26%
🟪 IRQ CPU Usage	26.19%	16.21%	0.13%	0.25%
🟦 Nice CPU Usage	0.00%	0.00%	0.00%	0.00%
⬜ Interrupt CPU Usage	2.49%	0.64%	0.08%	0.18%
🟧 Steal CPU Usage	0.00%	0.00%	0.00%	0.00%
🟦 User CPU Usage	45.53%	18.44%	0.47%	0.49%
🟥 System CPU Usage	0.59%	0.26%	0.11%	0.21%
🟩 Idle CPU Usage	98.60%	64.13%	28.29%	98.60%

Caption	Maximal	Average	Minimal	Current
🟦 cpu 0	3286 Mhz	2618 Mhz	1197 Mhz	1604 Mhz
🟩 cpu 1	3286 Mhz	2658 Mhz	1228 Mhz	1233 Mhz
🟥 cpu 2	3285 Mhz	2705 Mhz	1220 Mhz	1416 Mhz
🟨 cpu 3	3286 Mhz	2735 Mhz	1230 Mhz	1629 Mhz

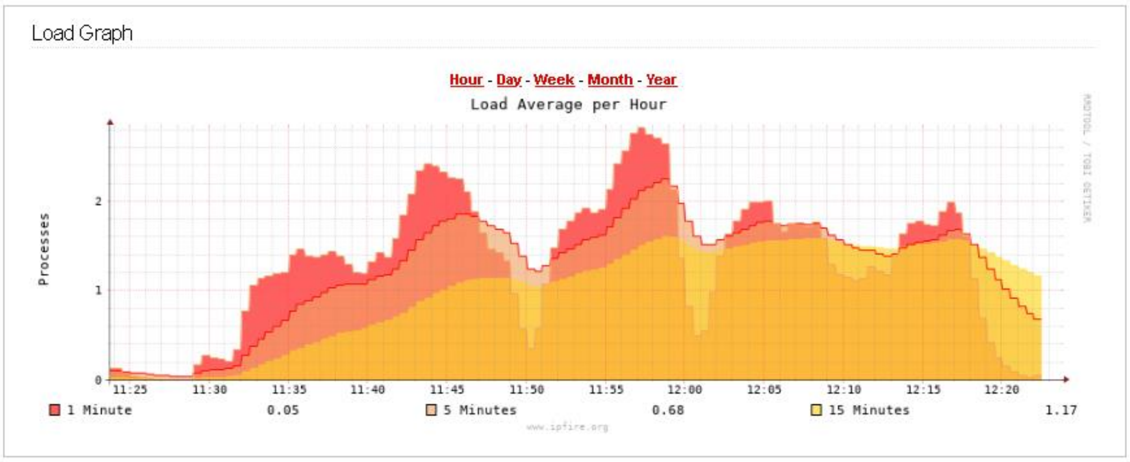
Figure 12. CPU usage and CPU frequency during DoS



Caption	Maximal	Average	Minimal	Current
🟦 Temp 0	68 Grad C	59 Grad C	44 Grad C	45 Grad C
🟩 Temp 1	0 Grad C	0 Grad C	0 Grad C	0 Grad C
🟥 Temp 2	43 Grad C	42 Grad C	39 Grad C	40 Grad C
🟨 Temp 3	45 Grad C	43 Grad C	39 Grad C	40 Grad C
⬜ Temp 4	28 Grad C	28 Grad C	28 Grad C	28 Grad C
🟦 Temp 5	127 Grad C	127 Grad C	127 Grad C	127 Grad C
🟩 Temp 6	69 Grad C	60 Grad C	44 Grad C	46 Grad C

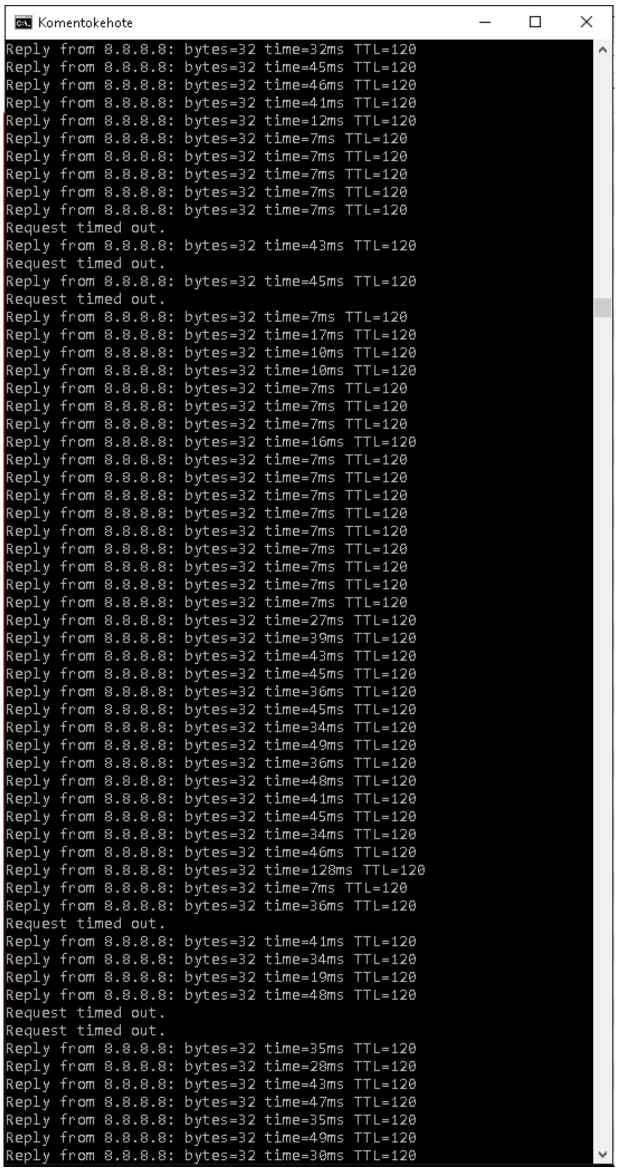
Caption	Maximal	Average	Minimal	Current
🟨 acpitzvirtual0temperaturetemp1	67.57 C	59.38 C	44.00 C	45.00 C
⬜ acpitzvirtual0temperaturetemp2	0.00 C	0.00 C	0.00 C	0.00 C
🟦 acpitzvirtual0temperaturetemp3	43.00 C	42.08 C	39.00 C	40.43 C
🟥 acpitzvirtual0temperaturetemp4	45.00 C	42.69 C	39.00 C	40.00 C
⬜ acpitzvirtual0temperaturetemp5	28.00 C	28.00 C	28.00 C	28.00 C
🟩 acpitzvirtual0temperaturetemp6	127.00 C	127.00 C	127.00 C	127.00 C
🟥 coretempisa0000temperaturetemp1	68.13 C	60.09 C	44.00 C	46.00 C
🟦 coretempisa0000temperaturetemp2	68.00 C	58.84 C	44.00 C	45.00 C
🟩 coretempisa0000temperaturetemp3	68.13 C	58.50 C	43.87 C	46.00 C

Figure 13. System temperatures during DoS



🟥 1 Minute	0.05	🟧 5 Minutes	0.68	🟨 15 Minutes	1.17

Figure 14. Load graph during DoS



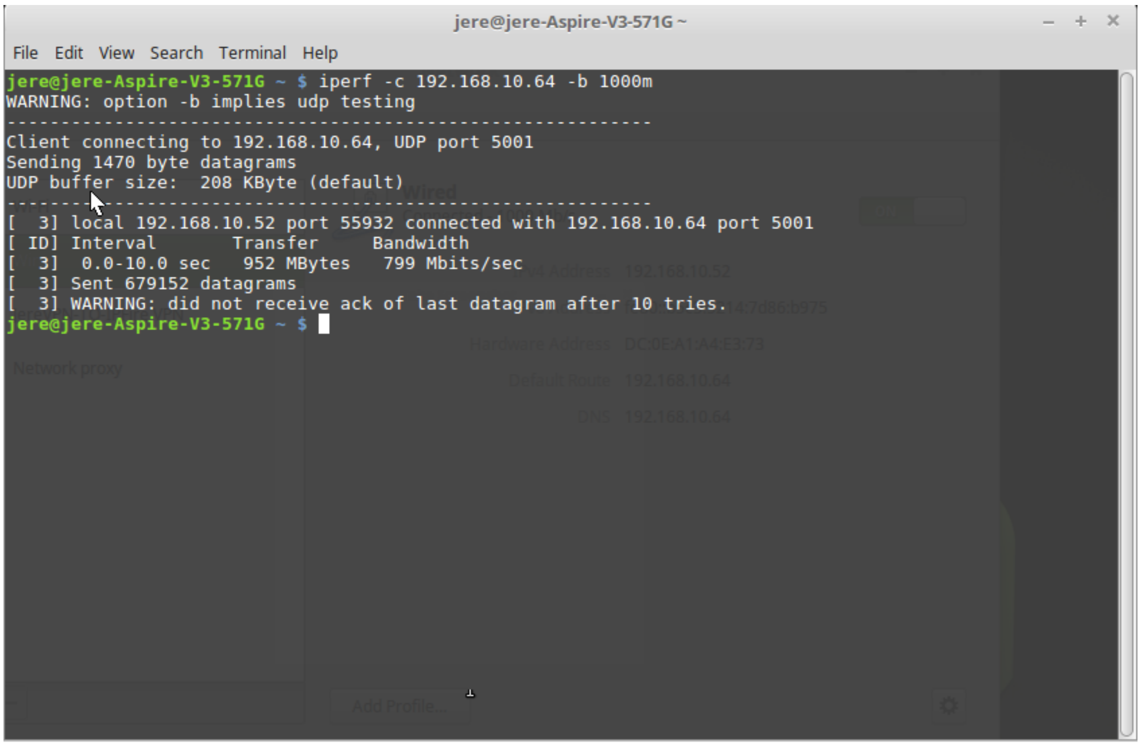
```
Reply from 8.8.8.8: bytes=32 time=32ms TTL=120
Reply from 8.8.8.8: bytes=32 time=45ms TTL=120
Reply from 8.8.8.8: bytes=32 time=46ms TTL=120
Reply from 8.8.8.8: bytes=32 time=41ms TTL=120
Reply from 8.8.8.8: bytes=32 time=12ms TTL=120
Reply from 8.8.8.8: bytes=32 time=7ms TTL=120
Reply from 8.8.8.8: bytes=32 time=7ms TTL=120
Reply from 8.8.8.8: bytes=32 time=7ms TTL=120
Reply from 8.8.8.8: bytes=32 time=7ms TTL=120
Reply from 8.8.8.8: bytes=32 time=7ms TTL=120
Request timed out.
Reply from 8.8.8.8: bytes=32 time=43ms TTL=120
Request timed out.
Reply from 8.8.8.8: bytes=32 time=45ms TTL=120
Request timed out.
Reply from 8.8.8.8: bytes=32 time=7ms TTL=120
Reply from 8.8.8.8: bytes=32 time=17ms TTL=120
Reply from 8.8.8.8: bytes=32 time=10ms TTL=120
Reply from 8.8.8.8: bytes=32 time=10ms TTL=120
Reply from 8.8.8.8: bytes=32 time=7ms TTL=120
Reply from 8.8.8.8: bytes=32 time=7ms TTL=120
Reply from 8.8.8.8: bytes=32 time=7ms TTL=120
Reply from 8.8.8.8: bytes=32 time=16ms TTL=120
Reply from 8.8.8.8: bytes=32 time=7ms TTL=120
Reply from 8.8.8.8: bytes=32 time=7ms TTL=120
Reply from 8.8.8.8: bytes=32 time=7ms TTL=120
Reply from 8.8.8.8: bytes=32 time=7ms TTL=120
Reply from 8.8.8.8: bytes=32 time=7ms TTL=120
Reply from 8.8.8.8: bytes=32 time=7ms TTL=120
Reply from 8.8.8.8: bytes=32 time=7ms TTL=120
Reply from 8.8.8.8: bytes=32 time=7ms TTL=120
Reply from 8.8.8.8: bytes=32 time=7ms TTL=120
Reply from 8.8.8.8: bytes=32 time=27ms TTL=120
Reply from 8.8.8.8: bytes=32 time=39ms TTL=120
Reply from 8.8.8.8: bytes=32 time=43ms TTL=120
Reply from 8.8.8.8: bytes=32 time=45ms TTL=120
Reply from 8.8.8.8: bytes=32 time=36ms TTL=120
Reply from 8.8.8.8: bytes=32 time=45ms TTL=120
Reply from 8.8.8.8: bytes=32 time=34ms TTL=120
Reply from 8.8.8.8: bytes=32 time=49ms TTL=120
Reply from 8.8.8.8: bytes=32 time=36ms TTL=120
Reply from 8.8.8.8: bytes=32 time=48ms TTL=120
Reply from 8.8.8.8: bytes=32 time=41ms TTL=120
Reply from 8.8.8.8: bytes=32 time=45ms TTL=120
Reply from 8.8.8.8: bytes=32 time=34ms TTL=120
Reply from 8.8.8.8: bytes=32 time=46ms TTL=120
Reply from 8.8.8.8: bytes=32 time=128ms TTL=120
Reply from 8.8.8.8: bytes=32 time=7ms TTL=120
Reply from 8.8.8.8: bytes=32 time=36ms TTL=120
Request timed out.
Reply from 8.8.8.8: bytes=32 time=41ms TTL=120
Reply from 8.8.8.8: bytes=32 time=34ms TTL=120
Reply from 8.8.8.8: bytes=32 time=19ms TTL=120
Reply from 8.8.8.8: bytes=32 time=48ms TTL=120
Request timed out.
Request timed out.
Reply from 8.8.8.8: bytes=32 time=35ms TTL=120
Reply from 8.8.8.8: bytes=32 time=28ms TTL=120
Reply from 8.8.8.8: bytes=32 time=43ms TTL=120
Reply from 8.8.8.8: bytes=32 time=47ms TTL=120
Reply from 8.8.8.8: bytes=32 time=35ms TTL=120
Reply from 8.8.8.8: bytes=32 time=49ms TTL=120
Reply from 8.8.8.8: bytes=32 time=30ms TTL=120
```

Figure 15. Ping test during DoS

The ping test in figure 15 shows that during the test some packets are dropping during transmit and the latency fluctuates.

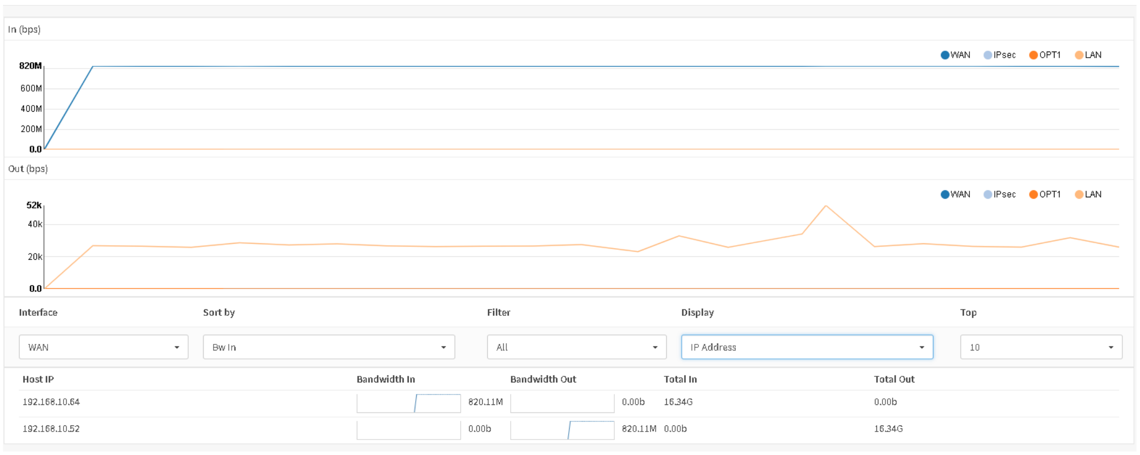
4.4.2 OPNsense

The throughput test on the OPNsense brought good results. The Iperf tests revealed that OPNsense was also able come close to the maximum throughput of 1 Gbps. The results are shown in figure 16 and 17.


jere@jere-Aspire-V3-571G ~
File Edit View Search Terminal Help

```
jere@jere-Aspire-V3-571G ~ $ iperf -c 192.168.10.64 -b 1000m
WARNING: option -b implies udp testing
------------------------------------------------------------
Client connecting to 192.168.10.64, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 192.168.10.52 port 55932 connected with 192.168.10.64 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   952 MBytes   799 Mbits/sec
[  3] Sent 679152 datagrams
[  3] WARNING: did not receive ack of last datagram after 10 tries.
jere@jere-Aspire-V3-571G ~ $
```

Figure 16. Throughput measured with Iperf



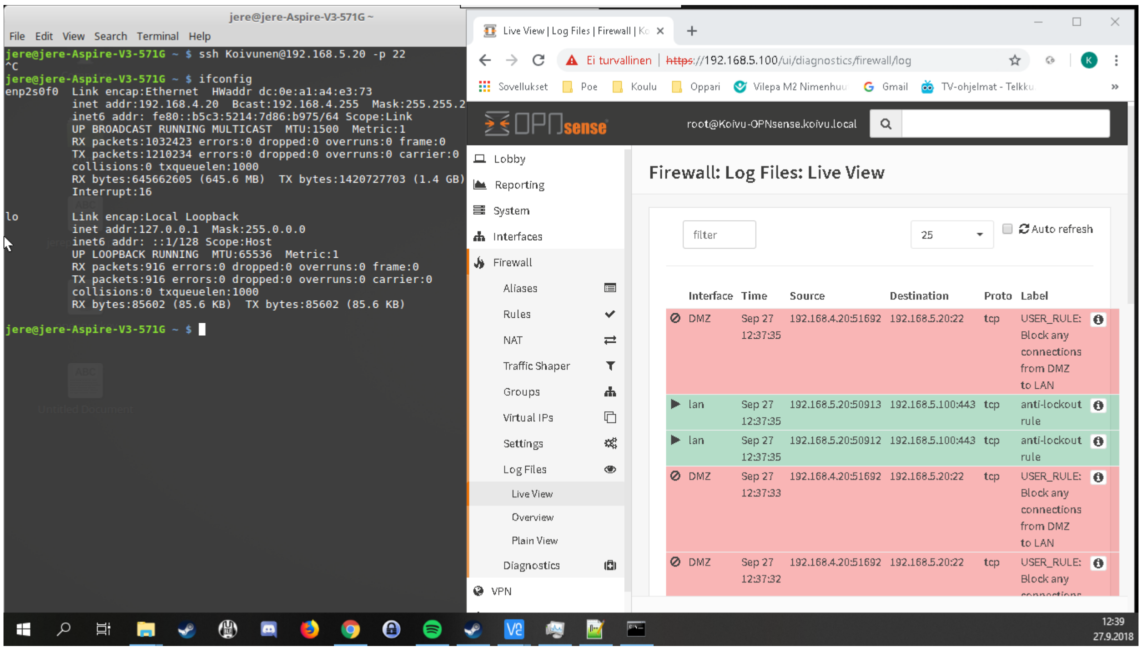
Interface: WAN
Sort by: Bw In
Filter: All
Display: IP Address
Top: 10

Host IP	Bandwidth In	Bandwidth Out	Total In	Total Out
192.168.10.64	820.11M	0.00b	16.34G	0.00b
192.168.10.52	0.00b	820.11M	0.00b	16.34G

Figure 17. Incoming traffic on OPNsense during Iperf test

The maximum throughput achieved during the tests were around 800 Mbps. The throughput tests were repeated after enabling the IPS and this had a big influence on the throughput. When the scan and DoS signatures were turned on in OPNsense IPS, the throughput dropped by half to around 400 Mbps.

The application detection test also failed on OPNsense. The following picture proves that SSH which uses a default port of 22 is being blocked by the firewall.



```
jere@jere-Aspire-V3-571G ~ $ ssh Koivunen@192.168.5.20 -p 22
^C
jere@jere-Aspire-V3-571G ~ $ ifconfig
enp2s0f0  Link encap:Ethernet  HWaddr dc:0e:a1:a4:e3:73
          inet addr:192.168.4.20  Bcast:192.168.4.255  Mask:255.255.2
          inet6 addr: fe80::b5c3:5214:7d86:b975/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1032423 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1210234 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:645662605 (645.6 MB)  TX bytes:1420727703 (1.4 GB)
          Interrupt:16

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:916 errors:0 dropped:0 overruns:0 frame:0
          TX packets:916 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:85602 (85.6 KB)  TX bytes:85602 (85.6 KB)

jere@jere-Aspire-V3-571G ~ $
```

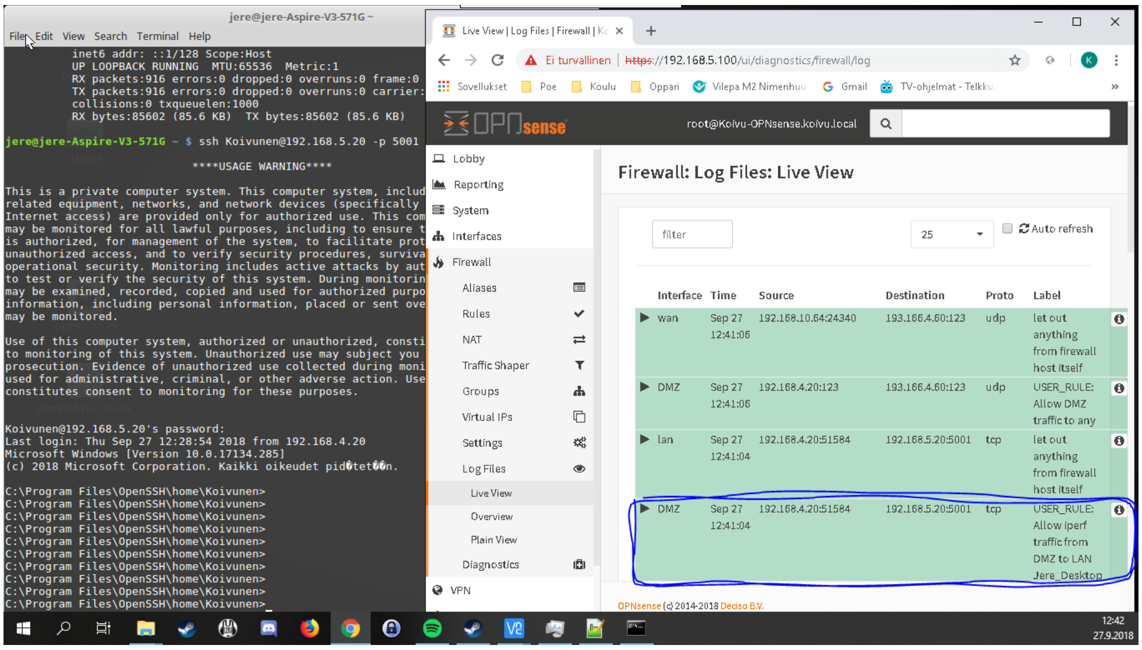
https://192.168.5.100/ui/diagnostics/firewall/log

Firewall: Log Files: Live View

Interface	Time	Source	Destination	Proto	Label
DMZ	Sep 27 12:37:35	192.168.4.20:51692	192.168.5.20:22	tcp	USER_RULE: Block any connections from DMZ to LAN
lan	Sep 27 12:37:35	192.168.5.20:50913	192.168.5.100:443	tcp	anti-lockout rule
lan	Sep 27 12:37:35	192.168.5.20:50912	192.168.5.100:443	tcp	anti-lockout rule
DMZ	Sep 27 12:37:33	192.168.4.20:51692	192.168.5.20:22	tcp	USER_RULE: Block any connections from DMZ to LAN
DMZ	Sep 27 12:37:32	192.168.4.20:51692	192.168.5.20:22	tcp	USER_RULE: Block any

Figure 18. Firewall log entry denying the SSH connection from DMZ zone to inside zone

The next SSH connection attempt was implemented over port 5001 which was allowed on the firewall. This connection was allowed by the firewall which indicates that the OPNsense can only block traffic based on the port used by an application, so no application detection takes place. The successful connection can be seen in figure 19.



```
jere@jere-Aspire-V3-571G ~
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:916 errors:0 dropped:0 overruns:0 frame:0
          TX packets:916 errors:0 dropped:0 overruns:0 carrier:
          collisions:0 txqueuelen:1000
          RX bytes:85602 (85.6 KB)  TX bytes:85602 (85.6 KB)

jere@jere-Aspire-V3-571G ~ $ ssh Koivunen@192.168.5.20 -p 5001

                          ****USAGE WARNING****

This is a private computer system. This computer system, includ
related equipment, networks, and network devices (specifically
Internet access) are provided only for authorized use. This com
may be monitored for all lawful purposes, including to ensure t
is authorized, for management of the system, to facilitate prot
unauthorized access, and to verify security procedures, surviva
operational security. Monitoring includes active attacks by aut
to test or verify the security of this system. During monitorin
may be examined, recorded, copied and used for authorized purpo
information, including personal information, placed or sent ove
may be monitored.

Use of this computer system, authorized or unauthorized, consti
to monitoring of this system. Unauthorized use may subject you
prosecution. Evidence of unauthorized use collected during moni
used for administrative, criminal, or other adverse action. Use
constitutes consent to monitoring for these purposes.


Koivunen@192.168.5.20's password:
Last login: Thu Sep 27 12:28:54 2018 from 192.168.4.20
Microsoft Windows [Version 10.0.17134.285]
(c) 2018 Microsoft Corporation. Kaikki oikeudet pid�tet��n.

C:\Program Files\OpenSSH\home\Koivunen>
C:\Program Files\OpenSSH\home\Koivunen>
C:\Program Files\OpenSSH\home\Koivunen>
C:\Program Files\OpenSSH\home\Koivunen>
C:\Program Files\OpenSSH\home\Koivunen>
C:\Program Files\OpenSSH\home\Koivunen>
C:\Program Files\OpenSSH\home\Koivunen>
C:\Program Files\OpenSSH\home\Koivunen>
C:\Program Files\OpenSSH\home\Koivunen>
C:\Program Files\OpenSSH\home\Koivunen>_
```

Live View | Log Files | Firewall | Ko

Ei turvallinen | https://192.168.5.100/ui/diagnostics/firewall/log

root@Koivu-OPNsense.koivu.local

Firewall: Log Files: Live View

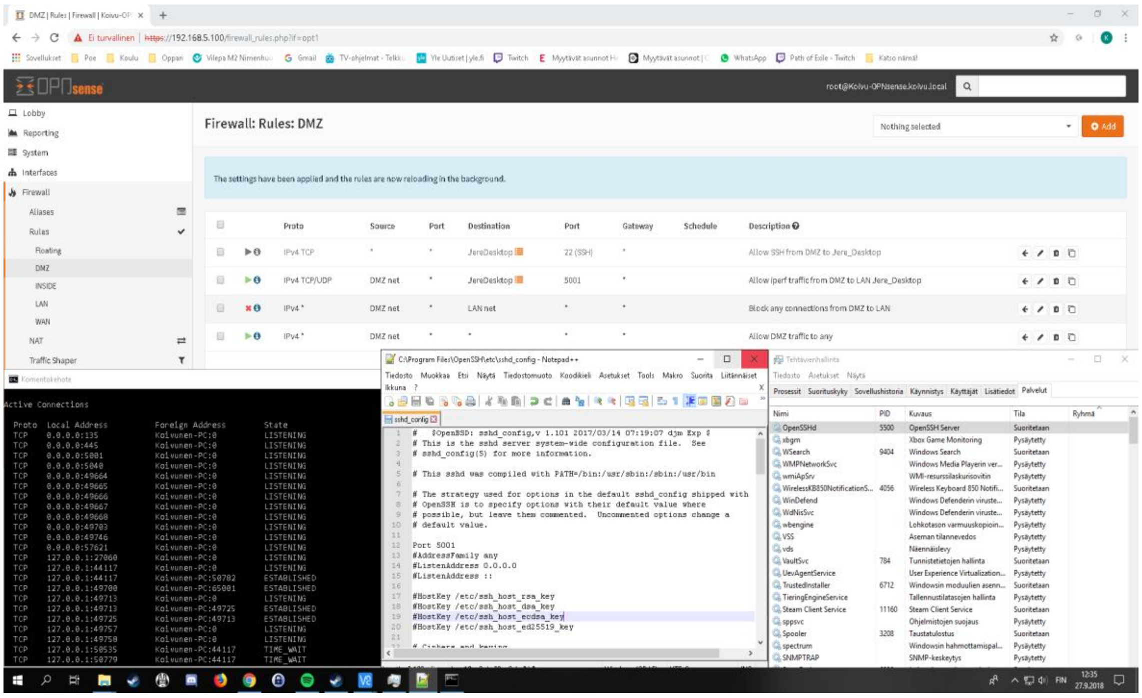
filter | 25 | Auto refresh

Interface	Time	Source	Destination	Proto	Label
wan	Sep 27 12:41:06	192.168.10.64:24340	193.166.4.60:123	udp	let out anything from firewall host itself
DMZ	Sep 27 12:41:06	192.168.4.20:123	193.166.4.60:123	udp	USER_RULE: Allow DMZ traffic to any
lan	Sep 27 12:41:04	192.168.4.20:51584	192.168.5.20:5001	tcp	let out anything from firewall host itself
DMZ	Sep 27 12:41:04	192.168.4.20:51584	192.168.5.20:5001	tcp	USER_RULE: Allow iperf traffic from DMZ to LAN Jere_Desktop

OPNsense (c) 2014-2018 Deciso B.V.

Figure 19. Firewall log entry for allowing SSH connection over port 5001

The firewall rules and SSH server configuration file along with a netstat output showing which ports the host is listening are shown in figure 20.



DMZ | Rules | Firewall | Koivu-OP

Ei turvallinen | https://192.168.5.100/firewall_rules.php?if=opt1

root@Koivu-OPNsense.koivu.local

Firewall: Rules: DMZ

Nothing selected | Add

The settings have been applied and the rules are now reloading in the background.

Proto	Source	Port	Destination	Port	Gateway	Schedule	Description
IPv4 TCP	*	*	JereDesktop	22 (SSH)	*		Allow SSH from DMZ to Jere_Desktop
IPv4 TCP/UDP	DMZ net	*	JereDesktop	5001	*		Allow iperf traffic from DMZ to LAN Jere_Desktop
IPv4 *	DMZ net	*	LAN net	*	*		Block any connections from DMZ to LAN
IPv4 *	DMZ net	*	*	*	*		Allow DMZ traffic to any

Komentokehote

```
Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            Koivunen-PC:0          LISTENING
  TCP    0.0.0.0:445            Koivunen-PC:0          LISTENING
  TCP    0.0.0.0:5001           Koivunen-PC:0          LISTENING
  TCP    0.0.0.0:5040           Koivunen-PC:0          LISTENING
  TCP    0.0.0.0:49664          Koivunen-PC:0          LISTENING
  TCP    0.0.0.0:49665          Koivunen-PC:0          LISTENING
  TCP    0.0.0.0:49666          Koivunen-PC:0          LISTENING
  TCP    0.0.0.0:49667          Koivunen-PC:0          LISTENING
  TCP    0.0.0.0:49668          Koivunen-PC:0          LISTENING
  TCP    0.0.0.0:49703          Koivunen-PC:0          LISTENING
  TCP    0.0.0.0:49746          Koivunen-PC:0          LISTENING
  TCP    0.0.0.0:57621          Koivunen-PC:0          LISTENING
  TCP    127.0.0.1:27060        Koivunen-PC:0          LISTENING
  TCP    127.0.0.1:44117        Koivunen-PC:0          LISTENING
  TCP    127.0.0.1:44117        Koivunen-PC:50782      ESTABLISHED
  TCP    127.0.0.1:49700        Koivunen-PC:65001      ESTABLISHED
  TCP    127.0.0.1:49713        Koivunen-PC:0          LISTENING
  TCP    127.0.0.1:49713        Koivunen-PC:49725      ESTABLISHED
  TCP    127.0.0.1:49725        Koivunen-PC:49713      ESTABLISHED
  TCP    127.0.0.1:49757        Koivunen-PC:0          LISTENING
  TCP    127.0.0.1:49758        Koivunen-PC:0          LISTENING
  TCP    127.0.0.1:50535        Koivunen-PC:44117      TIME_WAIT
  TCP    127.0.0.1:50779        Koivunen-PC:44117      TIME_WAIT
```

C:\Program Files\OpenSSH\etc\sshd_config - Notepad++

```
#	$OpenBSD: sshd_config,v 1.101 2017/03/14 07:19:07 djm Exp $
# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/bin:/usr/sbin:/sbin:/usr/bin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options change a
# default value.

Port 5001
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh_host_rsa_key
#HostKey /etc/ssh_host_dsa_key
#HostKey /etc/ssh_host_ecdsa_key
#HostKey /etc/ssh_host_ed25519_key

# Ciphers and keying
```

Tehtävienhallinta

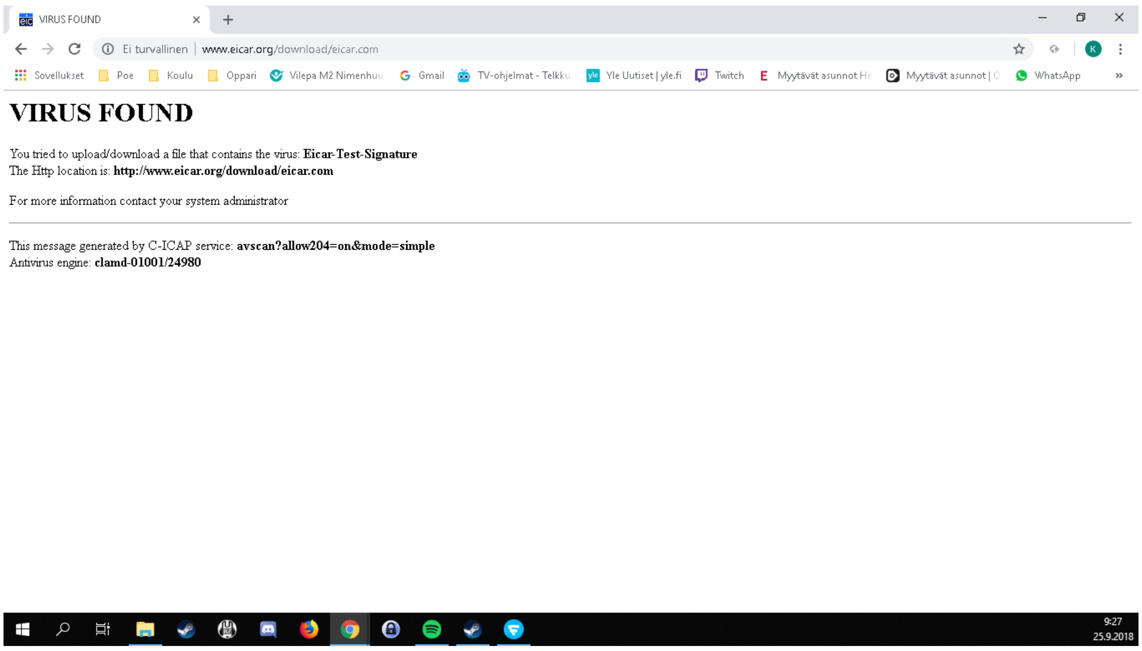
Nimi	PID	Kuvaus	Tila	Ryhmä
OpenSSHd	5500	OpenSSH Server	Suoritetaan	
xbgm		Xbox Game Monitoring	Pysäytetty	
WSearch	9404	Windows Search	Suoritetaan	
WMPNetworkSvc		Windows Media Playerin ver...	Pysäytetty	
wmiApSrv		WMI-resurssilaskurisovitin	Pysäytetty	
WirelessKB850NotificationS...	4056	Wireless Keyboard 850 Notifi...	Suoritetaan	
WinDefend		Windows Defenderin viruste...	Pysäytetty	
WdNisSvc		Windows Defenderin viruste...	Pysäytetty	
wbengine		Lohkotason varmuuskopioin...	Pysäytetty	
VSS		Aseman tilannevedos	Pysäytetty	
vds		Näennäislevy	Pysäytetty	
VaultSvc	784	Tunnistetietojen hallinta	Suoritetaan	
UevAgentService		User Experience Virtualization...	Pysäytetty	
TrustedInstaller	6712	Windowsin moduulien asenn...	Suoritetaan	
TieringEngineService		Tallennustilatasojen hallinta	Pysäytetty	
Steam Client Service	11160	Steam Client Service	Suoritetaan	
sppsvc		Ohjelmistojen suojaus	Pysäytetty	
Spooler	3208	Taustatulostus	Suoritetaan	
spectrum		Windowsin hahmottamispal...	Pysäytetty	
SNMPTRAP		SNMP-keskeytys	Pysäytetty	

Lobby
Reporting
System
Interfaces
Firewall
Aliases
Rules
Floating
DMZ
INSIDE
LAN
WAN
NAT
Traffic Shaper
Tiedosto Muokkaa Etsi Näytä Tiedostomuoto Koodikieli Asetukset Tools Makro Suorita Liitännäiset Ikkuna ?
sshd_config
Tiedosto Asetukset Näytä
Prosessit Suorituskyky Sovellushistoria Käynnistys Käyttäjät Lisätiedot Palvelut
12:35
27.9.2018

Figure 20. Firewall rules, netstat output from the SSH server, SSH server configuration file and task manager view of the OpenSSH server

The rule allowing SSH on the OPNsense is grayed out which means that the rule is disabled and if traffic is not specifically allowed it is denied by default. The window on the middle in the bottom of the screen is the SSH server configuration file and this is where the port that the SSH server is listening to is defined.

Moving on to the antivirus test this is where the OPNsense came up with very positive results. OPNsense can be configured to act as a middleman for web traffic. This means that the client is not communicating with the web server directly instead it connects to the OPNsense where the traffic is decrypted and encrypted and then sent to the web server. This enables OPNsense to detect viruses that are hidden in encrypted traffic like HTTPS. This feature was confirmed to work by downloading the Eicar test files. Below in figure 21 is the block page shown after attempting to download the unencrypted Eicar test file.



VIRUS FOUND

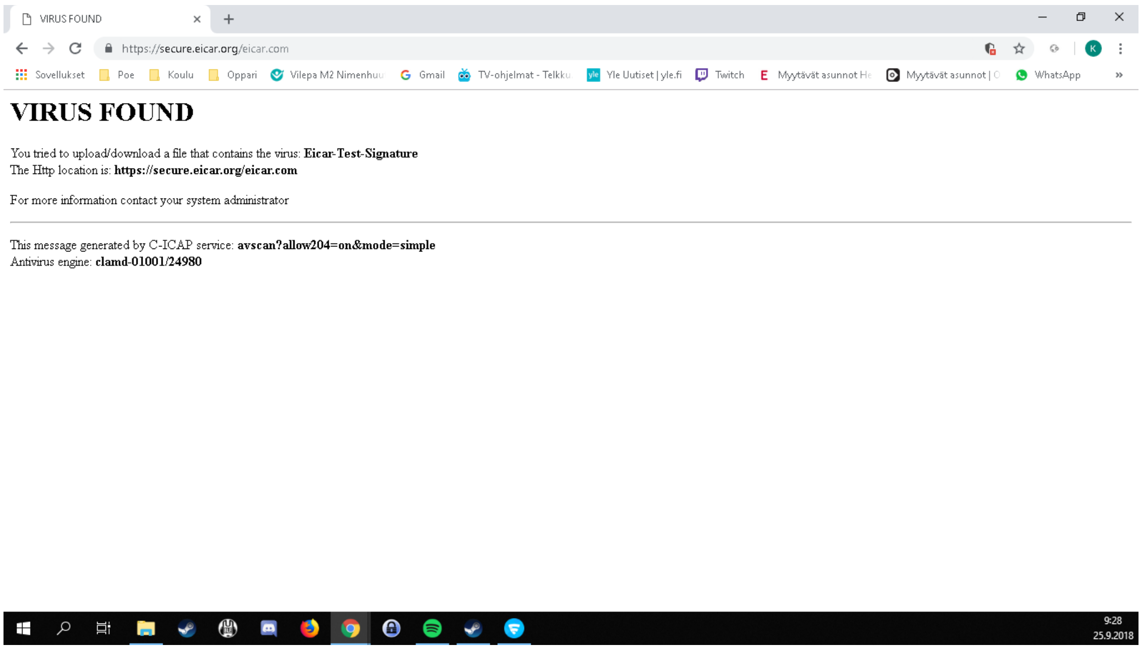
You tried to upload/download a file that contains the virus: **Eicar-Test-Signature**
The Http location is: **http://www.eicar.org/download/eicar.com**

For more information contact your system administrator

This message generated by C-ICAP service: **avscan?allow204=on&mode=simple**
Antivirus engine: **clamd-01001/24980**

Figure 21. Block page after downloading an unencrypted Eicar test file

The encrypted test file was also detected as seen in figure 22.



VIRUS FOUND

You tried to upload/download a file that contains the virus: **Eicar-Test-Signature**
The Http location is: **https://secure.eicar.org/eicar.com**

For more information contact your system administrator

This message generated by C-ICAP service: **avscan?allow204=on&mode=simple**
Antivirus engine: **clamd-01001/24980**

Figure 22. Block page after downloading an encrypted Eicar test file

OPNsense uses the open source virus scanning tool ClamAV to provide antivirus functionalities.

Since OPNsense has the capability to act as a middleman and decrypt traffic it was also able to restrict web content based on predefined categories for example advertisement or gambling. Once the SSL decryption was configured and the web filter was enabled the advertisements were efficiently filtered out from the various web pages. The filtering of web traffic on OPNsense is done transparently so there is no need for configuring clients separately.

The IDS and IPS abilities were tested by performing vulnerability scans against the OPNsense. The IPS was not able to prevent these scans nor was it able to block the DoS attack carried out on the firewall. The log files had no entries concerning these tests even though the signatures for DDoS and SYN scans were enabled on the IPS.

The DoS stress test almost fully utilized the OPNsense system resources. The CPU which is responsible for processing the flooding of packets was heavily drained during the test. Figure 23 shows how much CPU resources were used when a full-HD stream, a file download and a DoS attack were ongoing.

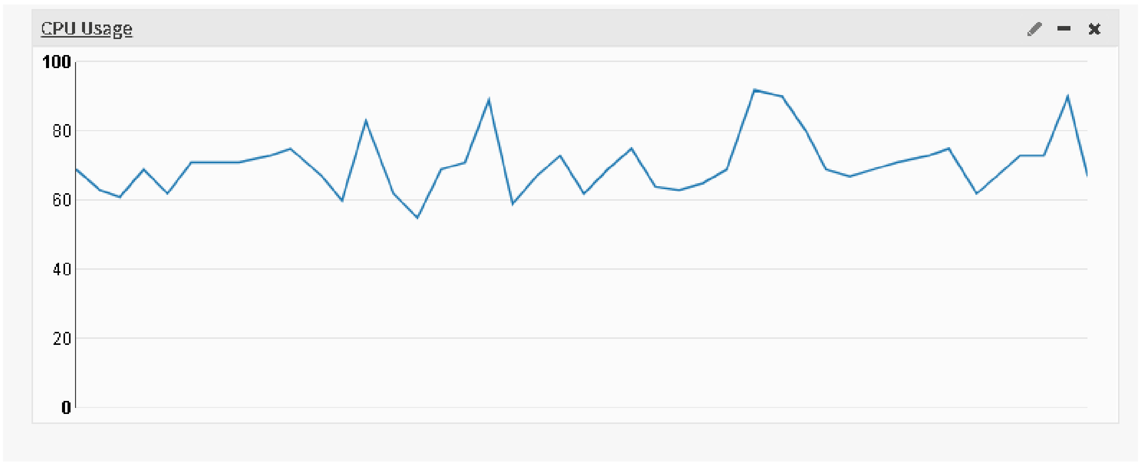
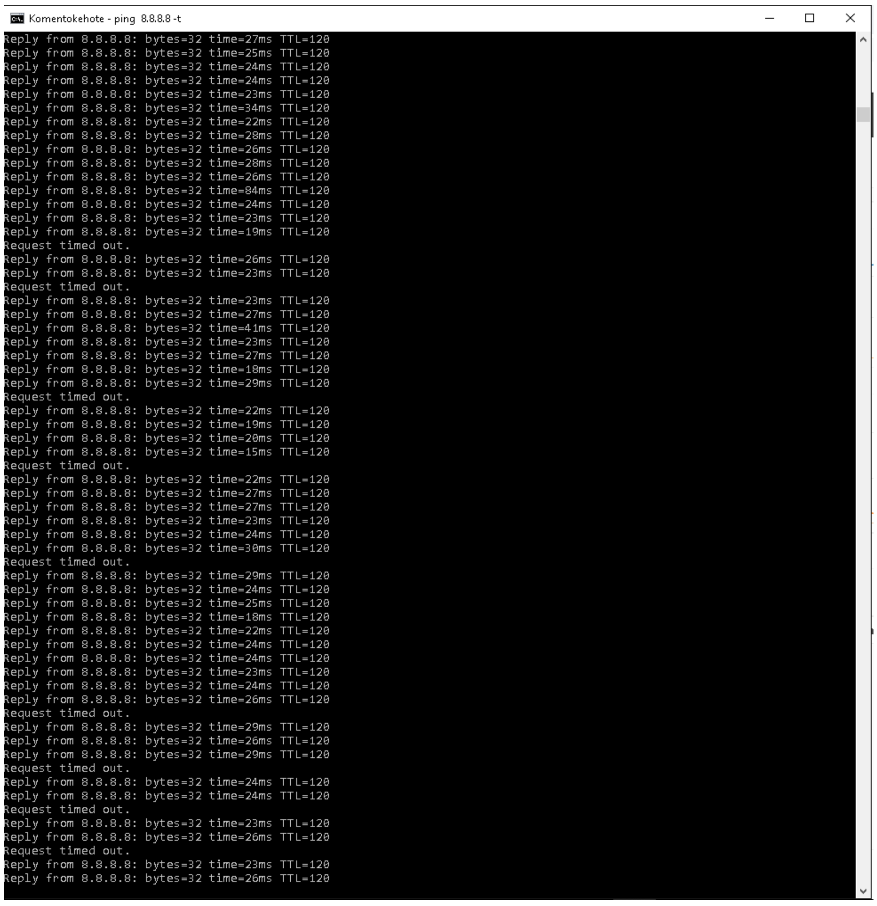


Figure 23. CPU usage during full-HD stream, file download and DoS

The stress test caused the download speed to drop to 1 Mbps and the ping test indicated that packets started dropping in addition to an increased latency as shown in figure 24.



Komentokehote - ping 8.8.8.8 -t

```
Reply from 8.8.8.8: bytes=32 time=27ms TTL=120
Reply from 8.8.8.8: bytes=32 time=25ms TTL=120
Reply from 8.8.8.8: bytes=32 time=24ms TTL=120
Reply from 8.8.8.8: bytes=32 time=24ms TTL=120
Reply from 8.8.8.8: bytes=32 time=23ms TTL=120
Reply from 8.8.8.8: bytes=32 time=34ms TTL=120
Reply from 8.8.8.8: bytes=32 time=22ms TTL=120
Reply from 8.8.8.8: bytes=32 time=28ms TTL=120
Reply from 8.8.8.8: bytes=32 time=26ms TTL=120
Reply from 8.8.8.8: bytes=32 time=28ms TTL=120
Reply from 8.8.8.8: bytes=32 time=26ms TTL=120
Reply from 8.8.8.8: bytes=32 time=84ms TTL=120
Reply from 8.8.8.8: bytes=32 time=24ms TTL=120
Reply from 8.8.8.8: bytes=32 time=23ms TTL=120
Reply from 8.8.8.8: bytes=32 time=19ms TTL=120
Request timed out.
Reply from 8.8.8.8: bytes=32 time=26ms TTL=120
Reply from 8.8.8.8: bytes=32 time=23ms TTL=120
Request timed out.
Reply from 8.8.8.8: bytes=32 time=23ms TTL=120
Reply from 8.8.8.8: bytes=32 time=27ms TTL=120
Reply from 8.8.8.8: bytes=32 time=41ms TTL=120
Reply from 8.8.8.8: bytes=32 time=23ms TTL=120
Reply from 8.8.8.8: bytes=32 time=27ms TTL=120
Reply from 8.8.8.8: bytes=32 time=18ms TTL=120
Reply from 8.8.8.8: bytes=32 time=29ms TTL=120
Request timed out.
Reply from 8.8.8.8: bytes=32 time=22ms TTL=120
Reply from 8.8.8.8: bytes=32 time=19ms TTL=120
Reply from 8.8.8.8: bytes=32 time=20ms TTL=120
Reply from 8.8.8.8: bytes=32 time=15ms TTL=120
Request timed out.
Reply from 8.8.8.8: bytes=32 time=22ms TTL=120
Reply from 8.8.8.8: bytes=32 time=27ms TTL=120
Reply from 8.8.8.8: bytes=32 time=27ms TTL=120
Reply from 8.8.8.8: bytes=32 time=23ms TTL=120
Reply from 8.8.8.8: bytes=32 time=24ms TTL=120
Reply from 8.8.8.8: bytes=32 time=30ms TTL=120
Request timed out.
Reply from 8.8.8.8: bytes=32 time=29ms TTL=120
Reply from 8.8.8.8: bytes=32 time=24ms TTL=120
Reply from 8.8.8.8: bytes=32 time=25ms TTL=120
Reply from 8.8.8.8: bytes=32 time=18ms TTL=120
Reply from 8.8.8.8: bytes=32 time=22ms TTL=120
Reply from 8.8.8.8: bytes=32 time=24ms TTL=120
Reply from 8.8.8.8: bytes=32 time=24ms TTL=120
Reply from 8.8.8.8: bytes=32 time=23ms TTL=120
Reply from 8.8.8.8: bytes=32 time=24ms TTL=120
Reply from 8.8.8.8: bytes=32 time=26ms TTL=120
Request timed out.
Reply from 8.8.8.8: bytes=32 time=29ms TTL=120
Reply from 8.8.8.8: bytes=32 time=26ms TTL=120
Reply from 8.8.8.8: bytes=32 time=29ms TTL=120
Request timed out.
Reply from 8.8.8.8: bytes=32 time=24ms TTL=120
Reply from 8.8.8.8: bytes=32 time=24ms TTL=120
Request timed out.
Reply from 8.8.8.8: bytes=32 time=23ms TTL=120
Reply from 8.8.8.8: bytes=32 time=26ms TTL=120
Request timed out.
Reply from 8.8.8.8: bytes=32 time=23ms TTL=120
Reply from 8.8.8.8: bytes=32 time=26ms TTL=120
```

Figure 24. Ping test during full-HD stream, file download and DoS

Naturally the CPU temperatures also started rising due to the amount of traffic it had to process which can be seen in figure 25.

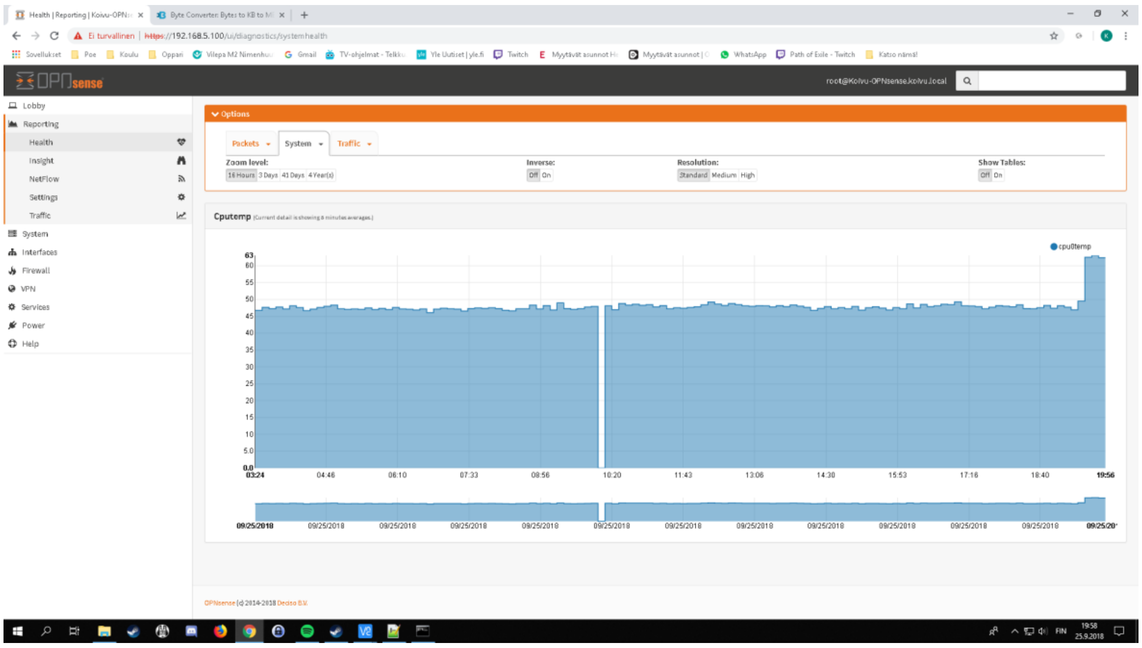


Figure 25. CPU temperatures during stress test

5 Comparison of open source and commercial firewalls

5.1 Differences between IPFire and OPNsense

The tests implemented on the IPFire and OPNsense indicated that both firewalls were able to achieve the same throughput with no additional features installed. The throughput test showed that when the IPS was enabled the throughput drastically decreased on OPNsense. IPFire was able to deliver the same throughput even with the IPS enabled. This was almost double the amount of traffic that OPNsense could handle with IPS enabled. Enabling the IPS on IPFire could be noticed on the throughput tests as an increase in CPU usage. A possible reason for poor IPS throughput on OPNsense could be that the operating system cannot fully utilize the processor resources.

The application detection was practically non-existent on both operating systems. However, the OPNsense has some sort of application awareness that comes with the IPS function. It is possible to block web applications by creating an IPS rule and including an SSL fingerprint of a website in to it. This way it is only possible to block a complete web application instead of only allowing certain features within the web application for example Facebook chat or Google mail.

The OPNsense came on top through the tests that included the built-in proxy functions like web filtering and antivirus detection. IPFire was not capable of SSL decryption and therefore it was only able to detect viruses and filter web content from unencrypted traffic. This is a huge setback because majority of traffic nowadays is encrypted. OPNsense was able to filter web content based on different categories of web pages for example advertisements and gambling. It also detected viruses from different types of encrypted files.

The IPS and stress tests were similar in both distributions. Neither of the tested distributions could detect the common scanning procedures nor were they able to stop or even detect the DoS attack. Both IPFire and OPNsense started dropping data packets and the latency increased during the DoS attack. The quality of the video stream that was playing while the DoS was ongoing degraded, and the download speed was affected negatively. One difference was that the inbound traffic during the DoS, video stream and a file download was 860 Mbps on IPFire while putting OPNsense under same stress it was receiving a maximum of 780 Mbps with IPS disabled in both systems. The fact that neither of the IPS engines was able to detect the DoS attack is a big security risk. The DoS attack implemented in the testing was originated from one host only. If the firewall would be attacked by a DDoS attack which means that multiple hosts would flood the firewall the effects would be devastating.

In terms of how easy it is to install and maintain the firewall distributions chosen for this thesis OPNsense seemed to be the better of the two. First off, the installation was simple on both distributions and every step of the installation had clear guidance on both systems. The OPNsense got stuck on the installation process a couple times but, in the end, it went through. Overall the installation was a bit easier on the IPFire and there were no errors during the process. After the initial setup all the additional security features had to be installed to be able to test how the features perform. This is where a bug on OPNsense was found. Rebooting OPNsense sometimes caused the interface assignments to reset which caused the internal network to lose connectivity. This made configuration changes very inconvenient because they often require a reboot of the system.

Both distributions in this thesis have documentation that helps administrators enable new features on the firewall. The OPNsense documentation is much clearer and more detailed compared to the documentation of IPFire. OPNsense has step-by-step guides on how to enable new features and the effects of configurations are clearly stated. The configuration was also much less time consuming on OPNsense due to excellent documentation. In addition, OPNsense documentation seems to be updated more frequently. In general, there was a lot of features on IPFire that were not fully documented especially most of the add-on features.

The general usage of these firewalls was moderately easy though OPNsense had a slight edge on this field. There were differences in the organization of the firewalls. IPFire was missing the ability to create address objects and address groups and this causes inconvenience in managing the firewall. Reviewing the logs of different events also felt more informative on OPNsense. The firewall log monitor on OPNsense showed exactly what rule is blocking or allowing traffic at a specific time which was missing from the IPFire. Also, the filtering of traffic log was much more advanced on OPNsense in fact IPFire did not have this functionality at all. Also, the way the UI was organized on OPNsense was more refined and aesthetic than what is was on IPFire. One thing where the IPFire stood out was how it displayed general system information for example CPU usage, memory usage and traffic flows. The graphs were very detailed and did not contain any excess information. Some of the graphs can be seen in chapter 4.4 of this thesis. Another important thing that increases usability is the automation of tasks. In OPNsense UI there is a specific section for Cron jobs where an administrator can configure automated tasks to occur on certain time frames. For example, updates can be scheduled to run every night at 01:00. This is also possible on IPFire, but it is not included in the GUI, so these tasks must be scheduled from the CLI which requires more know-how. An important note was also made about the configuration changes made to the firewalls. On OPNsense every configuration change made to the firewall is saved as a snapshot which means that if someone makes a configuration error the firewall can be reverted to the previous configuration that was working. It is also possible to take backups from different snapshots and the snapshots can be compared with each other to see what changes have been made between the two. This is a feature that IPFire does not have and it’s a feature that can help resolve many issues.

Unfortunately, the high availability was not tested in this thesis. To implement high availability, it requires two identical devices and those were not available in this thesis. The high availability functions were still researched to give an idea which firewall would better suit a production environment. It turned out that OPNsense provided better readiness for implementing high availability because in addition to the failover and state table synchronization it was able to exchange and maintain the configurations between the active and the passive device with the use of XMLRPC sync. [28] IPFire basically has the same functionality that OPNsense has but it uses an addon called Keepalived which uses VRRP to implement the failover function. To maintain all the ongoing connections that are stored in the state table when switching to a passive firewall IPFire uses an addon called Conntrackd. The ability to synchronize configuration changes from the active device to the passive device is not available for IPFire. Once again, the configuration of high availability is much more difficult on IPFire because it cannot be done from the GUI. [29]

Both firewalls had site-to-site VPN and remote VPN capabilities. The configuration of the VPN tunnels is done with the help of documentation and there are no configuration wizards available to make the configuration easier. As stated earlier in this thesis, the configuration of OPNsense is generally easier and configuring the VPN is no exception. The throughput values of the VPN tunnels were not measured in this thesis due to lack of time.

The power consumption of the firewall devices was not measured on this thesis because there were no tools required to perform this. It is still worth noting that OPNsense had different power usage modes that could be configured to save power or to make the system perform with maximum resources which also consumes more power. IPFire had no means of affecting the power usage.

5.2 Comparison to commercial firewalls

An important factor when comparing commercial firewalls to open source firewalls is the pricing of similar devices. For reference an entry level firewall designed for smaller environments from Palo Alto costs roughly one thousand euros [30]. This model is the PA-220 and it is somewhat comparable to the models available from open source firewall vendors. Both IPFire and OPNsense have their own hardware appliances that are for sale with different specifications. A similar model to PA-220 from OPNsense costs around 650 euros and from IPFire it costs approximately 750 euros [31] [32]. Concerning the IPFire and OPNsense devices a customer has all the features available for these prices. For the Palo Alto appliance many of the features require licenses before they can be enabled. For example, threat prevention requires a separate license before it can be enabled on the firewall. Several other features also require licensing on Palo Alto firewalls like URL filtering and support license [33]. To give an idea of the license costs a one-year threat prevention license for PA-220 costs 214 euros [34]. This means that the real costs from using commercial firewalls comes from the licenses rather than what the actual hardware costs. This is where the real savings are made when using open source firewalls since they do not require licenses.

The cost savings also come with the expense of the security features being less advanced and responsive. The open source firewalls fall behind when comparing the databases which are used to identify attack patterns, viruses, application level traffic, web content and much more. Open source firewalls do not possess the readiness to deal with zero-day attacks which means that if a new attack pattern is carried out on a firewall it has no means to block or identify the attack because it does not exist in the databases used to identify malicious traffic. Commercial firewalls like Palo Alto can prevent zero-day attacks by sending unknown files to a test cloud where the file is executed and if it is deemed harmful it is added to a database of known malicious attack patterns. These kinds of databases are constantly updated and the response time for dealing with different kinds of threats is greatly improved. The licenses are expensive for a good reason and most features on commercial firewalls are more advanced and provide higher security.

Another big difference on commercial and open source firewalls is the usability. When managing big network environments with thousands of users the amount of firewall rules, addresses and networks grows too big to handle if there is no means of organizing them. Commercial firewalls can be used to group security rules according to personal preferences for example rules that apply to a DMZ zone can be grouped or assigned a tag which makes it easier to find them amongst other rules. Another handy way of organizing the firewall rules is to separate NAT rules from security rules. In the end when there are several security administrators it comes down to creating a common policy which dictates how the firewall should be managed. Also, the traffic monitors are more advanced in many ways. Traffic can be filtered by a variety of different attributes like source and destination IP addresses, translated addresses, services and so on. An administrator can search for address objects, address groups, networks and easily view the security policies in which they are used. The same way one can search the traffic log, policies can also be filtered to make them easier to find. These kinds of advanced search functions make management more effective and less time consuming. The GUI on most commercial firewalls is also designed with more thought which makes them superior. It should also be mentioned that FQDN objects cannot be created on some of the open source firewalls.

The hardware of commercial firewalls also differs from the open source firewalls. Firewall vendors have specifically designed hardware that serves firewalling purposes. The components are designed to provide low latency with reduced load on the CPU also keeping in mind the power usage of the device. An example of the CPU architecture on Palo Alto firewalls is the separation of management and data handling for different processors. These are called the management plane and the data plane. The data plane is responsible for processing the actual traffic flowing through the firewall while the management plane takes care of the processes used to manage the firewall. This architecture prevents the normal traffic from affecting the management of the firewall for example in the event of a DoS. This feature cannot be found on the open source firewalls. Otherwise the hardware specifications are quite similar when comparing for example the Palo Alto PA-220 to corresponding models from IPFire and OPNsense. The OPNsense specifications did not include the power usage statistics but IPFire and Palo Alto both had a power usage of around 20 Watts. What was a bit surprising was that both IPFire and OPNsense boxes had better throughput than the Palo Alto PA-220. The fact that IPFire had the best throughput with similar hardware supports the test results acquired on this thesis. Also, it is hard to say anything about the quality of the hardware provided by open source hardware providers. One could assume the lifetime of the commercial firewalls is longer than what it is on the open source hardware.

Thinking about the future it is good to address the IPv6 support of open source firewalls. Commercial firewalls support IPv6 implementations but that is not the case with all open source options. OPNsense provides full IPv6 functionality but IPFire doesn’t officially support it yet. As of now IPFire can be manually configured to support IPv6 addressing but it requires a lot of expertise with the Linux operating system. Version 3 of IPFire is said to possess IPv6 support in the future.

6 State of open source firewalls

After installing, configuring and testing the open source firewalls in this thesis it is clear that open source firewalls generally are dragging behind when compared to commercial firewalls. Still there are some impressive features available in them considering they are free of charge. All the basic functions that modern companies require are there, but it feels like both tested firewalls were lacking in some field. For instance, IPFire excelled on the throughput but was lacking on ease of use and OPNsense on the other hand had poor throughput but had an intuitive UI and excellent content filtering capabilities. For a firewall to be a viable option for use in companies it must reliably manage to perform all tasks it is given. Having tested and used the open source firewalls for several months it can be stated that this does not apply for open source firewalls.

It has to be mentioned that the open source firewalls are developing fast and they constantly get updates and new features. Maybe the biggest drawback in open source firewalls at the time of writing this thesis is the absence of application level control of traffic. Recent developments are possibly going to make a big change in this field as well. While researching the open source firewalls a new add-on was discovered that is available for open source firewalls that would enable application level detection. A Company called Sunny Valley Networks offers a service called Sensei that can be installed on open source firewalls as an add-on. The Sensei is introduced to bring next generation capabilities to current open source firewalls. The Sensei promises application level control, full transparent TLS inspection, cloud application control and cloud-based threat intelligence databases to detect zero-day attacks [35]. If features like these are added to the existing open source firewalls along with other improvements, they could really start competing with some of the commercial firewalls out there. With these kinds of development steps in the open source field the existing firewall vendors must come up with new innovations to stand out in the market. As of now the open source firewalls could fit small companies that are not ready to spend thousands of euros to enhance their security.

7 Conclusion

The objective of this study was to clarify the present state of open source firewalls by testing and comparing two open source firewall distributions that are based on different operating systems. The key findings of this study were documented and used to evaluate the weaknesses in open source firewalls compared to commercial firewalls from known vendors.

The comparison of IPFire and OPNsense revealed that OPNsense is more suitable from a business point of view. The main things that gave OPNsense the edge were ease of use, SSL decryption, better support for high availability and comprehensive documentation. While reviewing the firewalls, an important note was made concerning their throughput and the way the hardware was utilized on high stress situations. It seemed that the Linux operating system was more efficient in utilizing the full potential of the CPU. This conclusion was made on the basis of observing how the firewalls CPU core frequencies acted when the firewall was under stress. During the tests on IPFire, the hardware graphs on the firewall showed that all the CPU cores were using their full potential. OPNsense did not have similar graphs to support the fact that it would have all the CPU cores working at full speeds. IPFire also had better results from the throughput tests which suggests that the Linux operating system is optimized better. Overall the Linux operating system was proven to be more reliable.

To summarize the weak points of open source firewalls, the threat prevention capabilities were worse than what was implied. The testing proved that the IPS functions and application detection were severely insufficient which makes commercial firewalls superior. Other central differences in commercial firewalls were also custom designed hardware and effortless management. Although commercial firewalls come with great expenses, they still seemed to be the better option in most cases.

References

1 Network > Zones. Online material. https://www.paloaltonetworks.com/documentation/71/pan-os/web-interface-help/network/network-zones#_56070. Accessed 15.11.2018.

2 Cisco Networking Academy: CCNA Security Chapter 4. https://static-course-assets.s3.amazonaws.com/CCNAS2/en/index.html#4 Accessed 23.3.2018.

3 Stateless Firewall Filter Overview. 2018. Online material. https://www.juniper.net/documentation/en_US/junos/topics/concept/firewall-filter-overview.html Accessed 23.3.2018.

4 Firewall and types. 2009. Online material. https://supportforums.cisco.com/t5/security-documents/firewall-and-types/ta-p/3112038 Accessed 23.3.2018.

5 Transmission Control Protocol Darpa Internet Program Protocol Specification. 1981. Online material. https://tools.ietf.org/html/rfc793#section-1.5 Accessed 15.11.2018.

6 How does a firewall track UDP? 2013. Online material. <https://supportforums.cisco.com/t5/firewalling/how-does-a-firewall-track-udp/td-p/2354302> Accessed 23.3.2018.

7 Next-Generation Firewalls (NGFWs). Online material. https://www.gartner.com/it-glossary/next-generation-firewalls-ngfws Accessed 5.4.2018.

8 Next-Generation Firewall Defined by Gartner. 2009. Online material. https://researchcenter.paloaltonetworks.com/2009/10/next-generation-firewall-defined-by-gartner/ Accessed 5.4.2018.

9 Defining the Next-Generation Firewall by Gartner. 2009. PDF-file. https://www.tomsnetworking.de/uploads/media/Gartner-Defining_next_generation_firewall.pdf Accessed 5.4.2018.

10 Palo Alto: Cybersecurity Survival Guide. PDF-file.

11 What is unified threat management? Online material. https://www.juniper.net/us/en/products-services/what-is/utm/ Accessed 15.11.2018.

12 Cisco Networking Academy: CCNA Security Chapter 8. https://static-course-assets.s3.amazonaws.com/CCNAS2/en/index.html#8 Accessed 16.4.2018.

13 SSL VPN Security. Online material. https://www.cisco.com/c/en/us/about/security-center/ssl-vpn-security.html Accessed 16.4.2018.

14 How Does SSL/TLS Work? Online material. https://www.websecurity.symantec.com/security-topics/how-does-ssl-handshake-work Accessed 16.4.2018.

15 Cisco Networking Academy: Routing and Switching Essentials Chapter 10. https://static-course-assets.s3.amazonaws.com/RSE503/en/index.html#10 Accessed 7.4.2018.

16 Domain Name System. 2018. Online material. https://en.wikipedia.org/wiki/Domain_Name_System Accessed 7.4.2018.

17 Root Servers. Online material. https://www.iana.org/domains/root/servers Accessed 7.4.2018.

18 Security Aspects and Benefits of Web Proxy in An Organization. 2017. Online material. https://www.securitycommunity.tcs.com/infosecsoapbox/articles/2017/04/06/security-aspects-and-benefits-web-proxy-organization?page=6 Accessed 15.11.2018.

19 Network Time Protocol. 2018. Online material. https://en.wikipedia.org/wiki/Network_Time_Protocol Accessed 4.5.2018.

20 PANB_Course_Manual_2018_09 from Elisa Santa Monica Networks. 2018. Javelin reader file with restricted access.

21 Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6. 2010. Online material. https://tools.ietf.org/html/rfc5798#section-3 Accessed 15.11.2018.

22 Common Address Redundancy Protocol (CARP). Online material. https://www.freebsd.org/doc/handbook/carp.html Accessed 15.11.2018.

23 Best free Linux firewalls of 2018. 2018. Online material. https://www.techradar.com/news/best-free-linux-firewall Accessed 7.4.2018.

24 iPerf - The ultimate speed test tool for TCP, UDP and SCTP. Online material. <https://iperf.fr/> Accessed 20.7.2018.

25 Anti-Malware Test file. Online material. http://2016.eicar.org/85-0-Download.html Accessed 15.11.2018.

26 Port Scanning Techniques. Online material. https://nmap.org/book/man-port-scanning-techniques.html Accessed 23.10.2018.

27 hping3(8) - Linux man page. Online material. https://linux.die.net/man/8/hping3 Accessed 17.9.2018.

28 Configure CARP. Online material. https://wiki.opnsense.org/manual/how-tos/carp.html Accessed 2.11.2018.

29 Firewall HA with conntrackd and keepalived. 2013. Online material. https://backreference.org/2013/04/03/firewall-ha-with-conntrackd-and-keepalived/ Accessed 2.11.2018.

30 Palo Alto PA-220 Next Generation Firewall System. Online material. https://www.ade24.de/Palo-Alto-PA-220-1 Accessed 6.11.2018.

31 OPNsense A10 Quad Core SSD Desktop Gen2. Online material. https://www.deciso.com/product-catalog/DEC630/ Accessed 6.11.2018.

32 lightningwirelabs.com-ipfire-appliances.pdf. 2017. PDF-file. https://www.lightningwirelabs.com/assets/docs/lightningwirelabs.com-ipfire-appliances.pdf Accessed 6.11.2018.

33 Activate Licenses and Subscriptions. Online material. https://www.paloaltonetworks.com/documentation/81/pan-os/pan-os/getting-started/activate-licenses-and-subscriptions Accessed 6.11.2018.

34 Palo Alto Threat Prevention Subscription PA-220. Online material. https://www.ade24.de/epages/63712130.sf/en_GB/?ObjectPath=/Shops/63712130/Products/PA-220-TP Accessed 6.11.2018.

35 Sunny Valley Networks. Online material. <https://www.sunnyvalley.io/sensei> Accessed 6.11.2018.

Table for conclusions of testing and comparison

🟥 Not available 🟧 Average 🟩 Excellent

Attribute	IPFire	OPNsense
Usability	🟧	🟩
Application detection	🟥	🟥
Threat prevention	🟧	🟧
Documentation	🟧	🟩
Power usage	🟩	🟧
Throughput	🟩	🟧
IPv6 support	🟧	🟩
Web filter	🟧	🟩