# KIT304 Server Administration and Security Assurance

# Tutorial 18

## Goal

In this tutorial you will configure the firewalls on CentOS and Windows Server 2016, and also install a popular network intrusion detection system.

## Introduction

In recent tutorials you have focused on penetration testing – finding the flaws in the security of a server, and attempting to breach it. Ultimately, this is a process that is undertaken to discover whether or not the security policies that are in place are actually achieving their intended objectives.

In today's tutorial you will spend most of your time getting some practical exposure to one of the most important items in your security arsenal from a network perspective: the firewall.

"Firewall" may sound like an unusual name for a piece of network security infrastructure. The name originates from a physical barrier that is often used in building construction that limits the spread of a fire into other parts of a building. In the context of network security, it is the embodiment of a security policy, and dictates which packets are allowed to enter or exit a network or host.

Firewalls come in various forms. They can exist as physical devices, much like routers, that sit between the internal and external network and process all incoming (and in some cases, outgoing) packets. Or they can exist as a software service on a single machine, controlling access to that machine alone. Some, including those you will look at today, are low level packet filters, while others examine traffic at higher levels of the OSI stack, and may control access based on content.

Generally, firewalls operate on the *default deny* principle. Rules are put in place to define what is permissible and everything else is denied. This can be costly from the perspective of usability, but it is efficient in terms of dealing with threats. In other security contexts, where risk is assessed differently, they may do the opposite and use a *default allow* approach, blocking specific traffic or content, but allowing everything else.

## Activity 1 – The CentOS Firewall

1. Launch **CentOS** and **Kali**. When they've booted up, log into them both and give them both an IP address in the **`192.168.1.x`** network using the steps below as a guide.

On **Kali Linux**: Log in with the username **root** and the password **toor**. You've set Kali's IP address multiple times in the past – the command to enter in the terminal window, is:

```
ip a add ipAddress/24 dev eth0
```

On **CentOS**:

a) log in with the username **student** and the password **student**
b) Click on the **Power** icon menu at the upper right of the screen, and then click on the **Ethernet (ens33) Off** entry to expose a submenu. Choose the **Wired Settings** option.
c) In the network settings panel that opens, press the "gear" icon to the right of the **Ethernet (ens33) 1000 MB/s** connection and then click on the **IPv4** tab.
d) Change **IPv4 Method** from *Automatic (DHCP)* to *Manual*.
e) Set the IP address and subnet mask appropriately, and set the gateway to 0.0.0.0.
f) Click **Apply**
g) Turn the Wired Connection **Off** and then **On** again to force the changes to be applied
h) close the settings window

After setting both IP addresses, check that you can ping one machine from the other.

2. To establish a baseline of what is visible about the CentOS system, you're going to scan the first 1000 ports from **Kali** using **nmap** (this will take several minutes to run – be patient):

```
nmap -sT -p 1-1000 ipAddressOfCentOS
```

> ➢ Note: You can check the progress of an **nmap** scan by pressing the up-arrow key.

How many ports appear open from Kali? ______________________________

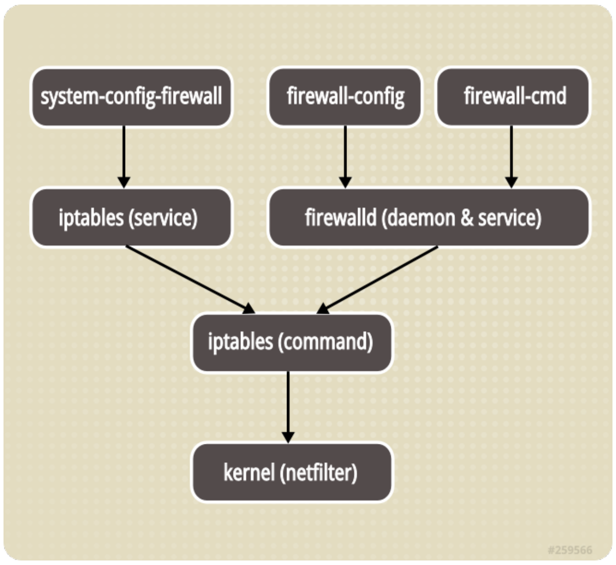
What is the state of the other ~1000 ports? ______________________________

3. The state you see for the remaining ports means that **nmap** can't tell whether those ports are open or closed as there is another item of network infrastructure in the way, most commonly a firewall. It could be a firewall running on the computer, or it could be a router between the two systems acting as a firewall. In the case of CentOS today, it is a software-based firewall.

4. For the next part of this tutorial, you're going to explore the **CentOS** firewall. To do this, you will need to become the administrator. Open a CentOS terminal window, and enter the command **su**. You'll be prompted for the root user's password, which is **toor**

5. The CentOS firewall can be managed in several ways:

- The "high-level" approach is to use the **`firewall-cmd`** command, which configures the firewall daemon **`firewalld`**. This daemon interacts with an underlying IP filtering system used in many Unix systems called **`iptables`**, which in turn sits on top of **`netfilter`** – the framework in the kernel that supports packet filtering and related activities.
- The "low-level" approach is to configure the **`iptables`** rules directly. The diagram below shows how the various components are related:


$$\begin{tikzcd}
\text{system-config-firewall} \arrow[d] & \text{firewall-config} \arrow[d] & \text{firewall-cmd} \arrow[dl] \\
\text{iptables (service)} \arrow[dr] & \text{firewalld (daemon \& service)} \arrow[d] & \\
& \text{iptables (command)} \arrow[d] & \\
& \text{kernel (netfilter)} &
\end{tikzcd}$$

- **`iptables`** (service) stores its rules in a text-based format in **`/etc/sysconfig/iptables`**
- **`firewalld`** stores its rules in XML format in **`/etc/firewalld/`**

6. The technique you'll use today is via **`iptables`** commands. These are simple rules that are used to determine how network packets are filtered. To see the list of default rules that already exist, enter the following command in the CentOS terminal, as root:

```
iptables -L
```

   While they won't mean much to you at this point, by the end of this tutorial you should have a better understanding of how to read those rules, and what effect each of them has.

7. Next, you're going to remove all of those rules and then slowly add some back so that you can see how the filtering system operates, one step at a time. To start, enter these three commands:

```
systemctl stop firewalld
iptables -I INPUT -j ACCEPT
```

   The first line turns off the existing firewall, and the next line is a rule that causes all inbound packets to be accepted.

Repeat the **nmap** scan that you did previously (on Kali) and you will see that another service (port) was open, but the firewall was previously filtering traffic to it. With the firewall effectively opened up, that service should now be visible to Kali. What is the port number and name of that service?

______________________________________________________________________________

8. Now enter the commands as detailed below on **CentOS** to set up the firewall so that it is a little more discriminating than just allowing every packet through:

   a) **`iptables -F`**
   Flush all existing rules so you start with a clean state from which to add new rules.

   b) **`iptables -A INPUT -i lo -j ACCEPT`**
   The **`-A`** switch means **append** (or add) a rule to a specific chain – in this case, the INPUT chain. The **`-i`** switch (for **`interface`**) specifies that this rule is relevant to packets destined for the **`lo`** (localhost, 127.0.0.1) interface. The **`-j`** switch (for **jump**) specifies the action to take for packets matching the rule – in this case the **`ACCEPT`** action. So this rule allows all incoming packets destined for the localhost interface to be accepted. This is generally required as many software applications expect to be able to communicate with the localhost adaptor.

   c) **`iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT`**
   This is the rule that does most of the work, and again appends (**`-A`**) it to the **`INPUT`** chain. Here you are using the **`-m`** switch (for **match**) to load the **`state`** module. The state module is able to examine the state of a packet and determine if it is NEW, ESTABLISHED or RELATED. NEW refers to incoming packets that are new incoming connections that weren't initiated by the host system. ESTABLISHED and RELATED refer to incoming packets that are part of an already established connection or related to an already established connection. In other words, this rule is allowing incoming packets that are part of, or related to, a connection that is already established.

   d) **`iptables -A INPUT -p tcp --dport 22 -j ACCEPT`**
   This adds a rule allowing incoming SSH connections (on **`tcp`** port **`22`**).

   e) **`iptables -P INPUT DROP`**
   The **`-P`** (for **policy**) switch sets the default policy on the specified chain. In this case, you're setting the default policy on the INPUT chain to DROP, meaning that if an incoming packet does not match one of the preceding rules it will be dropped. If you were connected to your CentOS host remotely via SSH, and had not added the rule at (d) above, you would have locked yourself out of the system at this point. Fortunately, you're logged in to the GUI via the console today, so you can't actually lock yourself out of the system.

f) **`iptables -P FORWARD DROP`**
This command sets the default policy on the FORWARD chain to DROP. The FORWARD chain is used to pass packets *through* your computer, enabling it to act as a router – that's not something you want to do in this tutorial, hence this rule.

g) **`iptables -P OUTPUT ACCEPT`**
This sets the default policy on the OUTPUT chain to ACCEPT, which means that all outgoing traffic is permitted (implying that you trust your users).

h) **`iptables -L -v`**
This lists (**`-L`**) the rules verbosely (**`-v`**) for the iptables service, so you can check the rules that you've just added.

9. To summarise what you've done above, you:

- on the **`INPUT`** chain
  - allow internal traffic to **`localhost`** to be processed
  - allow incoming traffic for connections established by local applications (to remote systems) to be processed
  - allow incoming **`SSH`** connections
  - drop all other traffic
- on the **`FORWARD`** chain, drop all traffic
- on the **`OUTPUT`** chain, pass all traffic

10. Next, turn on the web server on CentOS by entering the following command:

```
service httpd start
```

You can confirm that port 80 is now listening for connections by viewing the netstat output:

```
netstat -ant
```

Make sure you can find the line that represents the web server listening in the output (and recall from tutorial 16 that you'll only see the service listening in the tcp6 section because Apache uses IPv6 sockets to handle IPv4 and IPv6 connections). You can also confirm that the web server is running by opening Firefox on CentOS (under **Applications**) and visiting the web site **localhost**. You should see a simple "It works!" message from the web server.

11. Now attempt to access the **CentOS** web server from **Kali** in the **Firefox** browser (it is located near the top of the quick launch bar on the left side of the screen). Remember you're not visiting localhost now – you need to enter the IP address of CentOS in the address bar. You should find the connection eventually times out. What is the status of port 80 (the web server's listening port) on CentOS when you scan it with **`nmap`** with Kali?

______________________________________________________________________________

12. Enter an **`iptables`** rule for **CentOS** that will open port 80. Write your rule below:

    ______________________________________________________________________________

    Confirm that your rule works by loading the webpage on **Kali**, and scanning it with **`nmap`** again.

## Activity 2 – The Windows Firewall

Many of you will have interacted with the Windows firewall previously on your own Windows system(s), typically when adding permission for an application to go through it. Based on your Unix experience in the previous activity, hopefully you now understand that what you are actually doing is opening up the TCP port being used by that application so that its traffic can get out of your machine and onto the network.

13. Launch the **Windows Server 2016** virtual machine. Select **Send Ctrl-Alt-Del** from the VMWare **Virtual Machine** menu to get a login prompt on the server, and when prompted for the login password, enter **ToorToor1**. After it logs in, give it an IP address in your virtual network. Use the same approach you've used previously for setting the IP address for Windows 10.

14. The Windows Firewall is on by default, and its default configuration allows traffic to more ports than the CentOS default firewall does. Scan the Windows server from **Kali** – how many of the first 1000 are open?

    ______________________________________________________________________________

    From the set of open ports you should be able to see that **Kerberos** is one of the protocols that is running on the Windows Server. You'll have a lecture on this protocol in week 13 – Kerberos is a network authentication system that uses *tickets* to allow nodes communicating over the network to prove their identity to each another in a secure manner – it is an early example of a single-sign-on solution, and is used widely in Windows networks.

15. In the Server Manager window (which should already be open), select the **Tools** menu (located in the upper right of the window) and select the **Windows Firewall with Advanced Security** option. On the panel on the left side of the window, click on **Inbound Rules**, and then **Outbound Rules** to get a sense for how many rules are currently in place on the system. You are able to filter the list using the various **Filter** options in the **Actions** panel on the right-hand side of the window.

    You can also sort the rules by column. If you sort by the **Local Port** column you can see that there are ports that are flagged as **Enabled** that you did not see in the **`nmap`** scan from Kali. What might the reason be for this?

______________________________________________________________________

______________________________________________________________________

16. Next you will create a rule to see how to enforce a policy through the firewall. You want to block a particular IP address from accessing your system on port 80, while not restricting other systems. ***Before you do this***, open web browsers on both **CentOS** and **Kali**, and enter the IP address of your Windows Server into the address bars of both to confirm that the Windows web server is functioning normally.

17. In the Windows Firewall window, ensure you're viewing **Inbound Rules** and click **New Rule** in the **Actions** panel on the right-hand side of the window. Then:

    a) in the **Rule Type** tab, select **Custom** and click **Next**
    b) in the **Program** tab, select **All Programs** and press **Next**
    c) in the **Protocols and Ports** tab select **TCP** from the **Protocol type** menu, set the **Local port** to **Specific Ports**, and enter **80** in the local port field, then click **Next**
    d) in the **Scope** tab, in the **Which remote IP addresses does this rule apply to?** section, select **These IP addresses** and then click the **Add** button. Enter the IP address of the Kali system in the text entry area, but as you do so, note that you can block more than just a single IP address – you can block subnets, or even defined ranges of your own choosing. Click **OK**, and then click **Next**
    e) in the **Action** tab, select the **Block the Connection** radio button and click **Next**
    f) in the **Profile** tab, click **Next**
    g) in the **Name** tab, add a name and description for the rule that you have created, and click **Finish**

18. Go back to the browser on **Kali**, hold down the shift key and click the reload icon on the address bar (to force a page refresh). This time, the page shouldn't load and will eventually time out. If you do an `nmap` scan of the Windows Server's port 80, what mode do you see?

    ______________________________________________________________________

    Meanwhile, on **CentOS**, the web page will still load normally as before, unaffected by the rule you just added.

## Activity 3 – Intrusion Detection

Firewalls are useful as a preventative measure. They enforce a policy. However, what happens when they don't work, or a user is undertaking activity that is allowed by the firewall, but is something that probably shouldn't be allowed on the network? Firewalls don't respond in an interactive way. They simply allow or disallow packets based on the rules they have been configured with.

An Intrusion Detection System (IDS) is the next layer in our security-in-depth model. The goal of an IDS is about detection and response. They operate in a similar fashion to anti-virus systems that you may have on your own systems. They are looking for malicious activity, or activity that is anomalous, and alert you when they discover something suspicious.

A **network-based IDS** often resides just inside the firewall at the entry point (or the exit point) of a network. It is invisible to an attacker, as it typically isn’t running on a host that they can easily interact with, and basically just syphons up packets as they travel past. This does mean, however, that while an IDS can see encrypted traffic, it cannot decrypt it, and cannot make judgements about the content. IDSs observe large amounts of traffic, operating in as close to real-time as possible. One potential down-side is that they often don’t have much context about what the target of a packet actually is. For example, is the web traffic it is observing actually going to a webserver?

**Host-based** intrusion detection systems protect just that: the host or endpoint. This includes workstations, servers, mobile devices and so on. They are one of the last layers of defence, and are also one of the best security controls because they have more context than any other layer about the target. They can be configured to look for known types of attacks against a ***host***, but they don’t usually have any context about what is occurring on the rest of the network.

An active area of research is sharing information between different intrusion detection systems and other security infrastructure, and using intelligent agents to correlate information about events that are occurring at multiple locations around the network.

In the following activity you will be installing **`snort`** on Windows Server 2016. Snort is a network-based IDS available for Windows and Unix based systems. It started out as an open-source system in 1998, but the company that developed it was purchased by Cisco in 2009, and they now continue its development.

Unfortunately, the default Snort configuration files are set up assuming Unix file locations and paths. You’ll need to modify the main configuration file to use Windows format paths, and to specify some Windows executables.

19. On **Windows Server 2016** in the **Downloads** folder are two installers (**Snort** and **WinPcap**) a folder of files named **`snortrules-snapshot-29141`**.

20. Run **WinPcap** to install it onto the server, accepting all of the default values when prompted. This program provides packet-capture services for network traffic, and is also required for programs like Wireshark.

21. Run the **Snort** installer, accepting all of the default values when prompted.

22. Copy the ***contents*** of the **`Snortrules-snapshot-29141`** folder (***not the folder itself***) into the **`C:\Snort`** directory that was created by the Snort installer. When prompted, *allow it to* ***Replace the files in the destination***.

23. Click on the Windows **Search** icon at the bottom left of the screen, and enter the text **`ISE`**. From the list of matching entries, click on **Windows PowerShell ISE**. In the **File** menu, select **Open**, navigate to **`C:\Snort\etc`**, and open the **`snort.conf`** file. You need to make a series of changes to this file to work under Windows.

    a) Under **`Step #1`** (line 41) set the **`HOME_NET`** variable to the IP address of your Windows server. For example:

    ```
    ipvar HOME_NET ipAddressOfWindowsServer
    ```

    This specifies that you are only protecting your Windows Server 2016. If you were to set the value to **`192.168.1.0/24`** then you could monitor traffic for the entire subnet, but that would mean that the Kali VM was being protected too, and Snort would ignore malicious packets from Kali to your Windows host, and thus not alert you to their presence. Thus, instead of monitoring a subnet, today you will just monitor the Windows host.

    b) Scroll down to the **`Path to your rules files`** section (line 101), and change the rules so that they match the following (text that must remain unchanged is shown in grey, while new or changed text is shown in black). Note that you have to use Windows-style backslash characters here, not the Unix-style forward slashes that are in the config file:

    ```
    var RULE_PATH c:\snort\rules
    # var SO_RULE_PATH ../so_rules
    var PREPROC_RULE_PATH c:\snort\preproc_rules
    ...
    var WHITE_LIST_PATH c:\snort\rules
    var BLACK_LIST_PATH c:\snort\rules
    ```

    c) On the last line of the **`Step #2`** section (line 182), remove the leading hash that comments the line out, and add a path to the log file:

    ```
    config logdir: c:\snort\log
    ```

    d) Move down to **`Step #4`** (line 238) and modify the first two settings under it as follows:

    ```
    # path to dynamic pre-processor libraries
    dynamicpreprocessor directory c:\snort\lib\snort_dynamicpreprocessor
    # path to base preprocessor engine
    dynamicengine c:\snort\lib\snort_dynamicengine\sf_engine.dll
    # path to dynamic rules libraries
    ```

Note in particular the change in slashes to backslashes.

e) Move down to line 326 and insert a comment character at the start of the line:

```
#   decompress_swf { deflate lzma } \
```

f) In **`Step #5`** find the section that relates to **`portscan`** detection (***not*** the line that introduces the topic, but rather the line immediately below, which should be line 416) and remove the comment character **`#`** at the start of the **`preprocessor`** line:

```
# Portscan detection. For information, see README.sfportscan
preprocessor sfportscan: proto ... (rest of line unchanged)
```

g) Save your changes (control-S), but leave the file open as you may need to make corrections.

24. In the command-line area of the **Windows PowerShell** window enter the following command:

```
cd C:\snort\bin
```

For Snort to function you need to tell it what network interface to monitor. To get a list of the interfaces enter the command:

```
.\snort.exe -W   (note – this is an upper-case “W”)
```

Note that some of the output, while appearing in red, is not an error. Below the red text that was output, you’ll see a list of network interfaces and their index values. On this virtual machine you only have one interface, and its index is **`1`**, however on other computers you may have more than a single interface. The index number of the interface that you want to monitor is passed as a parameter to the **`-i`** option in the following step.

25. Test that your Snort configuration is valid by entering the following command:

```
.\snort.exe -i 1 -c c:\snort\etc\snort.conf -T
```

As this runs it will output quite a few messages, but if your configuration file is correct it will eventually finish and report **`Snort successfully validated the configuration!`** If it fails, it will usually indicate a line number in the configuration file that you need to adjust.

26. Before you start Snort, you’re going to write a new rule. The files in the **`c:\Snort\rules`** folder are ***signatures*** that are matched against traffic captured on the network interface that indicate possible threats or malicious activity. You will create a rule that will alert you when someone pings your system. To do this open the **`local.rules`** file in the **`C:\Snort\rules`** directory in the ISE text editor, and enter the following rule at the bottom of the file:

```
alert icmp any any -> any any (msg:"testing icmp rule"; sid:1000001;)
```

This rule states that an alert should be generated when any **`icmp`** messages appear from any external IP address on any port, to any port on any IP address in the range being monitored. The **`sid`** (signature ID) at the end is a 7 digit number, which is commonly what is used for locally defined rules. Existing rule definitions in public rule libraries tend to use the first 6 digits in an ordered fashion (with gaps for further rules to be added in the future).

27. Save the **`local.rules`** file, and start snort by entering the following command in PowerShell:

    ```
    .\snort.exe -i 1 -c c:\snort\etc\snort.conf -A console
    ```

    This tells snort to run on interface index 1 (your Ethernet port) using **`snort.conf`** as its configuration file, and report alerts (**`-A`**) to the console.

    Once Snort has completed validating its rules, it will cease output on the console, and will be running and monitoring the IP address you provided earlier in the config file.

28. Ping the Windows Server from **Kali** – you should see that for every ping, there's a resulting alert logged on the console by Snort, as specified by the rule that you entered at step 27. Use Control-C to quit the ping process in Kali.

29. Now use **nmap** (on Kali) to execute a Christmas scan (**`-sX`**) of the first 1000 ports on the Windows server and see if Snort detects it. It shouldn't, since you don't yet have any relevant rules to detect this type of scan.

    To detect **nmap** scans in Snort, open and edit the **`scan.rules`** file in the **`rules`** folder, and add some or all of the following rules to it:

    ```
    alert tcp any any -> any any (msg:"SYN FIN Scan"; flags: SF; sid:9000000;)
    alert tcp any any -> any any (msg:"FIN Scan"; flags: F; sid:9000001;)
    alert tcp any any -> any any (msg:"NULL Scan"; flags: 0; sid:9000002;)
    alert tcp any any -> any any (msg:"XMAS Scan"; flags: FPU; sid:9000003;)
    alert tcp any any -> any any (msg:"Full XMAS Scan"; flags: SRAFPU; sid:9000004;)
    ```

    Save your edits, force Snort to quit with a Control-C, and then relaunch it to confirm that it can now detect **nmap** scans too (it may take a few seconds for the first of them to appear in the log).

## Snort Rules

You've only scratched the surface of what can be detected with Snort, and only by creating very simple local rules. In production scenarios, the use of up-to-date rules is critical to be able to detect and be alerted to new threats as they emerge and are documented by malware researchers. The process of creating new rules to detect these threats is complex and ongoing, and as a result, there are two sets of rules you can use for Snort. The **Community Ruleset** is developed and contributed by snort users freely, and is freely available to all users. The **Snort Subscriber Rule Set** is developed by paid employees of the Snort service, and is only available to those who either register with the service, or pay for a Snort subscription.

Given its 20+ year development history, Snort's rule format is now recognised as a *de facto* standard, and is supported by many other intrusion detection and protection systems.

## Conclusion

In KIT111 or KIT201 you learned about Firewalls and Intrusion Detection Systems from a theoretical perspective. Today you configured and used them in practice (albeit in a very simple way). They are not simple infrastructure elements that you set up and leave – they require on-going maintenance and monitoring, and real-world systems can be quite complex to configure and use.

A real-world network will often have multiple subnets, and each of those subnets may well have different policies applied to them at the firewall and the intrusion detection system. Similarly, some hosts on the network will be visible to the outside world, and those externally-visible components will require more aggressive monitoring by the IDS. This layering of policies complicates network management but provides a much stronger defence against reconnaissance operations and system exploitation from bad actors.

## Skills to Master

This week's skills that you need to master to progress through the unit include:

- using the **`iptables`** commands on CentOS to perform basic firewall configuration operations
- using the Windows firewall interface to explore the current firewall configuration, and to add new rules
- configuring Windows to use Snort, including adding local rules to detect basic network activity like pings and **`nmap`** scans

## Appendix: The DMZ (Additional Reading)

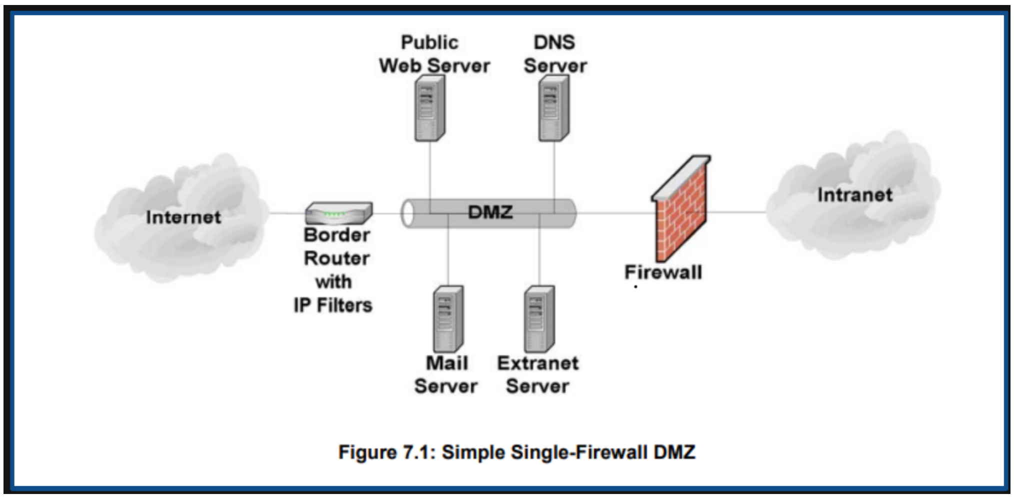
The purpose of the reconnaissance phase of penetration testing is to gain an understanding of what the target network and service infrastructure is comprised of and how it interconnects. All elements are key to effectively predicting and then attacking the target's defensive and security resources.

As Internet services evolved in the late 1990's it was soon realised that servers opened up logical paths into otherwise physically secure and private networks and systems. Once a server is compromised, an attacker may be able to use it as a base for further reconnaissance and attacks, working their way deeper into an ICT ecosystem. As a consequence, the perimeter security model was born. This establishes the notions that:

- the external (i.e., the public Internet) is "untrusted"
- the internal corporate network (often called the Intranet) is considered the "trusted" or secure network, and
- other companies (such as supply-chain or manufacturing partners) requiring a direct connection but not fully trusted are termed "extranets"

All are connected in a tightly controlled subnetwork called the De-Militarized Zone (DMZ)

DMZ's are designed specifically to business requirements and involve placing firewall(s) between the public Internet and the internal network. This creates physical and logical network segment(s) that can only be traversed through rules and policies.



*Scarfone et. al., 2007*

- the Border/Edge router is where ISP services are terminated
- the DMZ hosts all externally accessible business components and servers such as Web, DNS, Mail and VoIP/video

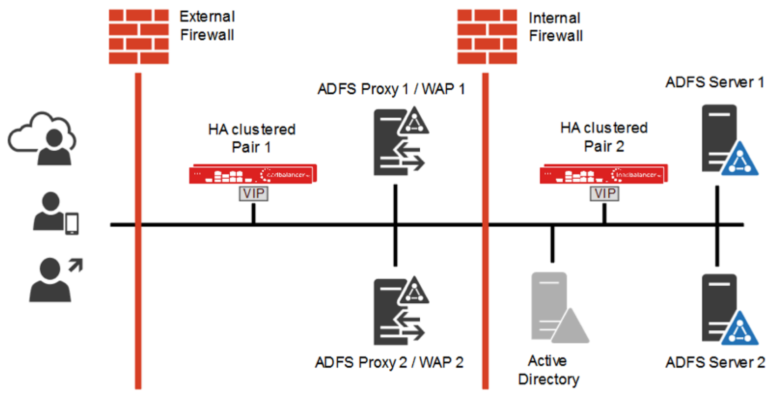
The simple DMZ is where the existing border router is used with access control lists (ACL's) to restrict certain addresses (MAC and or IP) and/or, types of traffic (port numbers) through the DMZ. This is only appropriate for small businesses that face minimal threat and don't need extensive security resources.

The basic weakness in the approach is that while the router is able to protect against most network layer attacks, it is not "aware" of the actual server hardware, OS, application(s) and services. Thus, it cannot protect against attacks aimed specifically targeting a server/service. Also, the router cannot provide application load balancing or virus scanning of incoming email, for example.

A more secure approach to creating a DMZ network is to use dual firewalls, in which two firewalls are deployed with the DMZ network positioned between them.

- the external or perimeter firewall is configured to allow external traffic destined to the DMZ and proxy servers only
- the internal firewall only allows traffic from the DMZ and proxies into the internal network.

For example, using Microsoft Active Directory Federated Services:



*loadbalancer.org*

This is considered to be more secure as both would need to be compromised before an attacker could access the internal resources.

The DMZ allows businesses to separate the network so that security controls can be tuned specifically for each service. For example, rather than have publicly accessible Web/Application Servers, secure web services could be configured to direct all SSL/TLS encryption and termination to a dedicated SSL Offload server and then only allow sessions from the SSL Offload server to be actioned by a dedicated internal HTTPS (TCP:443) server (this configuration is often referred to as a **jump server**).

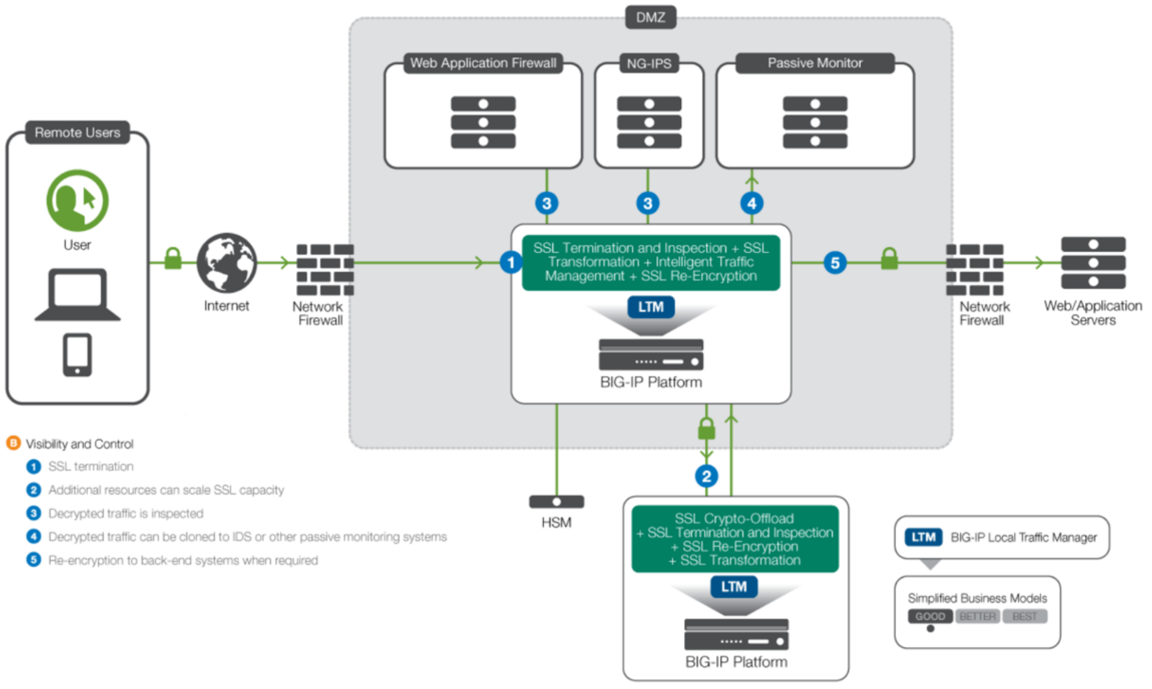

DMZ
Web Application Firewall
NG-IPS
Passive Monitor
Remote Users
User
Internet
Network Firewall
SSL Termination and Inspection + SSL Transformation + Intelligent Traffic Management + SSL Re-Encryption
LTM
BIG-IP Platform
Network Firewall
Web/Application Servers
Visibility and Control
SSL termination
Additional resources can scale SSL capacity
Decrypted traffic is inspected
Decrypted traffic can be cloned to IDS or other passive monitoring systems
Re-encryption to back-end systems when required
HSM
SSL Crypto-Offload + SSL Termination and Inspection + SSL Re-Encryption + SSL Transformation
LTM
BIG-IP Platform
LTM BIG-IP Local Traffic Manager
Simplified Business Models
GOOD
BETTER
BEST

Figure 6: Many different systems, including web analytics, benefit from SSL crypto offload.

***f5.com, 2014***

As the data volumes and variety of services have increased, so network architectures have needed to evolve in order to allow for cloud services and high availability servers providing greater service flexibility, availability and redundancy.

Architectures often now include multiple layers, for example:

- TCP/IP Layer 2 - Network
  - Hot Standby Router Protocol (HSRP)
    - A Cisco proprietary protocol which provides redundancy for a local subnetwork.
    - With HSRP, two or more routers share a Virtual IP (VIP) address to appear as one router.

- TCP/IP Layer 3 – Transport
  - Load Balancers such as F5.
    - TCP and UDP streams can be separated/directed to specific internal servers.
    - Proxy services - used on the Edge or inside the DMZ to translate port numbers to internal servers.

- TCP/IP Layer 4 – Server Clustering (2 or more Application servers to work in unison)
  - Fail-over Clusters
    - Consist of 2 or more servers having a separate "heartbeat" connection between the hosts.
    - The heartbeat is used to monitor whether all the services are still in use: as soon as a service on one machine breaks down the other machine(s) take over.
  - Load-Balanced Server clustering
    - When a request is received, the cluster checks which machine is the least busy and then sends the request to that machine.

    - Load-balancing cluster is also a Fail-over cluster but with the extra functionality and often, multiple hosts in multiple zones.
- High Performance Computing (HPC) clustering
    - Where servers are configured specifically to provide data centres and Cloud service providers the extreme elasticity and performance they need.

Failover and Load Balanced clusters are commonly used for small to medium sized business web-farms, databases and firewalls. Businesses requiring 99.99999% availability (3.16 seconds downtime per year…) or higher (!) require complex combinations of both load balancing and HPC.

These high service requirements are also often deploying a “Zero Trust” environment where there is no static Trusted zone and **ALL** traffic is deemed potentially hostile.

## Further Reading

https://searchsecurity.techtarget.com/definition/DMZ

https://www.oreilly.com/library/view/zero-trust-networks/9781491962183/ch01.html

Scarfone, Karen et al, 2007, ***NIST Special Publication 800-45 Version 2, Guidelines on Electronic Mail Security***, https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-45ver2.pdf

loadbalancer.org (author unspecified), ***Load balancing Microsoft ADFS***, https://www.loadbalancer.org/applications/microsoft-adfs/

f5.com (author unspecified), 2014, ***The F5 SSL Reference Architecture***, https://www.f5.com/services/resources/white-papers/the-f5-ssl-reference-architecture