

# Simulation of a Secure Corporate Ecosystem

*A culminating experience report  
submitted in partial fulfilment of the  
requirements for award of the degree  
Of*

MASTER OF SCIENCE

In

CYBERSECURITY

Submitted by:

Parag Mhatre (801073873)



DEPARTMENT OF SOFTWARE AND INFORMATION SYSTEMS

UNIVERSITY OF NORTH CAROLINA CHARLOTTE

CHARLOTTE, NC 28223

SPRING – 2020

## Table of Contents

1.	Project Summary	3
2.	Project Details	4
3.	Simulate the corporate environment	4
4.	Setting up the various enterprise services	6
5.	Configuring the Single Sign-On and securing the network communications	7
6.	Install and secure the web server	8
7.	Place and configure the network firewall	10
8.	Reflection	12

## Project Summary

This project was a part of the Competitive Cyber Defence course under Dr. Thomas Moyer. This course was recently introduced into the curriculum at University of North Carolina at Charlotte in the department of Software and Information systems. This course aims to provide the students with hands-on experience with designing, deploying, securing, and defending enterprise network services. Topics taught in the course include: securing network communication, single-sign-on services, firewall and IDS deployment, security policy design and development, log analysis, securing critical network infrastructure, network access control policies, penetration testing tools, secure information flow, and secrets management. Students are expected to demonstrate their ability to defend these services against adversary attacks.

In today's world, any corporate infrastructure faces a number of challenges and security consists of a huge part of those challenges. Many times, a lot of functions used by the employees require them to use third party services which would traditionally require them to login to every third party service that they want to use. Therefore, to have a single point of login, we use a Single Sign-On which allows us to login once and use that authentication to establish a session to every other service that we want to use.

If an attacker tries to intrude a network, a firewall is the go-to appliance to stop the intruders and malicious connections. The type, placement and configuration of a firewall is equally important to make sure of its effectiveness. Thus, we use two firewalls in this project, iptables - a stateful packet filtering firewall and ModSecurity - a web application firewall. We also decide the placements and configurations of these firewalls.

We perform the above functionalities on a simulated corporate infrastructure that we create on a Xubuntu virtual machine using LXD system container manager that builds on top of LXC. We create separate containers to simulate different machines like the SSH server, LDAP database server, Kerberos server, DNS server, Web server, client machine, etc.

The amount of tools and technologies that I was exposed to when I was pursuing this project has set me on a course to be a better cybersecurity professional than I was before taking this course. For example, many parts of the project required extensive research and testing to come up with solutions against problems that we faced. This is a very important skill that is required in real world cybersecurity jobs. Researching and coming up with innovative solutions even when there is very little or confusing documentation available is something that we expect to deal with every day.

## Project Details

This project consists of five phases and the objective of the project is to configure basic enterprise network services, implement Single Sign-On using LDAP and Kerberos, configure a web server and to set up two firewalls in the network.

1. Simulate the corporate environment
2. Setting up the various enterprise services
3. Configuring the Single Sign-On and securing the network communications
4. Install and secure the web server
5. Place and configure the network firewall

The five phases are explained below in detail along with all the tasks and actions that were performed in each phase along with figures wherever required.

### 1. Simulate the corporate environment

To simulate a corporate environment, we install Xubuntu Linux as a virtual machine. In this machine, we use LXD container management platform to create multiple containers that represent various machines (servers) required in a corporate ecosystem.

We create multiple containers to simulate corporate assets - DNS server, DNS backup server, system log server, network file system, LDAP server, Kerberos server, mail server, test machine, SSH server, backup test machine and a client machine with IP addresses as shown in the Fig 1 below.

We use containers instead of using VM as they have a performance benefit. Usage of a few important machines is explained below.

Host	IP Address	Purpose
ns1	10.0.3.11	Primary DNS Server
ns2	10.0.3.12	Secondary DNS Server
syslog	10.0.3.14	Logging server
nfs	10.0.3.15	NFS server
ldap	10.0.3.16	LDAP server
krb	10.0.3.17	Kerberos server
mail	10.0.3.18	Mail server
server	10.0.3.100	Test Server
client	10.0.3.101	Test Client
ssh	10.0.3.102	SSH Server
client2	10.0.3.250	Second test client

Fig 1. List of containers

DNS Server: It is a server that contains a database of hostnames and their associated public IP addresses.

Other machines use this server to convert domain names to IP addresses.

LDAP Server: A server that stores databases which can be updated/modified real-time and that other machines can use to lookup information from the database.

Kerberos Server: Kerberos is a network authentication protocol that uses symmetric key cryptography and requires authorization from a trusted third party to authenticate client-server applications

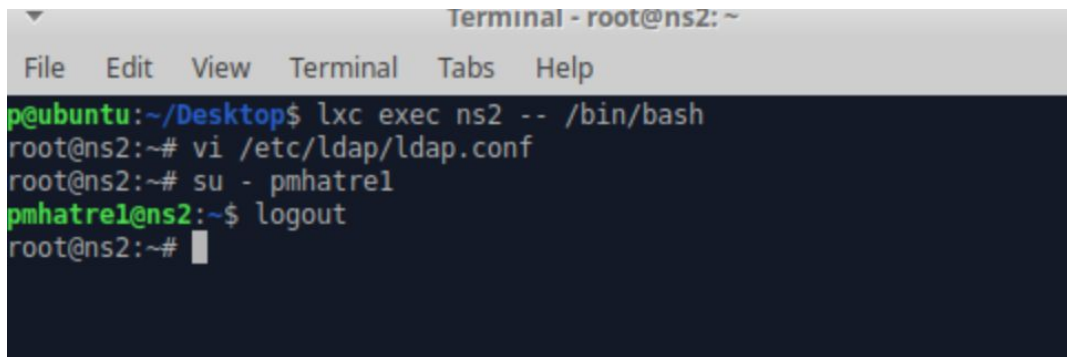
Web Server: A machine that runs software and hosts one or more websites.

Client Machine: This is a machine that simulates any workstations used by employees to perform their day-to-day tasks.

## 2. Setting up the various enterprise services

Once the containers were deployed with IP addresses as shown above, we configure the respective machines so as to enable the services to function successfully. We use the *bind9* software to configure the domain nameservers. This enabled every other machine in the network to access other machines using their domain names instead of the IP addresses.

Next, we configured the LDAP server using the *openldap* software tool. This enabled that specific server to store details about services, devices, etc. in the LDAP database which can be accessed by other machines in the network.

A terminal window titled "Terminal - root@ns2: ~" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows a sequence of commands and prompts: a user logs into a container named 'ns2' via 'lxc exec ns2 -- /bin/bash', becomes root, edits '/etc/ldap/ldap.conf' with 'vi', switches to user 'pmhatrel' with 'su - pmhatrel', logs out with 'logout', and returns to the root prompt.

```
Terminal - root@ns2: ~
File Edit View Terminal Tabs Help
p@ubuntu:~/Desktop$ lxc exec ns2 -- /bin/bash
root@ns2:~# vi /etc/ldap/ldap.conf
root@ns2:~# su - pmhatrel
pmhatrel@ns2:~$ logout
root@ns2:~#
```

Fig 2. LDAP login successful

After the LDAP server, we configured the Kerberos Server using the *krb5* package and set that up to act as an authenticator for users. The Kerberos Server would fetch a user's details from the LDAP database and authenticate a user, thus enabling that user to login on any machine in the network that is set up to use LDAP and Kerberos.

The way Kerberos works is, it authenticates a user with a TGT (Ticket Granting Ticket). This is done by a process where the user's machine signs a request with the user's private key which is verified on the

server end. The key is never actually sent over the network. Once a user is authenticated, the user gets a TGT that the user can send to the TGS (Ticket Granting Service) to get a temporary session key for whichever service that the user wants to access and use. The details of the session and the above details can be seen in the Fig 3 below.

```
Mar 20 05:31:40 krb kadmind[2827]: Setting up TCP socket for address 0.0.0.0.464
Mar 20 05:31:40 krb kadmind[2827]: Setting up TCP socket for address ::.464
Mar 20 05:31:40 krb kadmind[2827]: setsockopt(15,IPV6_V6ONLY,1) worked
Mar 20 05:31:40 krb kadmind[2827]: Setting up RPC socket for address 0.0.0.0.749
Mar 20 05:31:40 krb kadmind[2827]: Setting up RPC socket for address ::.749
Mar 20 05:31:40 krb kadmind[2827]: setsockopt(17,IPV6_V6ONLY,1) worked
Mar 20 05:31:40 krb kadmind[2827]: set up 6 sockets
Mar 20 05:31:40 krb kadmind[2827]: Seeding random number generator
Mar 20 05:31:40 krb kadmind[2827]: kadmind: starting...
Mar 20 05:31:40 krb kadmind[2827]: starting

root@krb:~# kinit pmhatrel
Password for pmhatrel@MHATRE.CCD.LAB:
root@krb:~# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: pmhatrel@MHATRE.CCD.LAB

Valid starting    Expires          Service principal
03/20/19 05:32:16 03/20/19 15:32:16 krbtgt/MHATRE.CCD.LAB@MHATRE.CCD.LAB
    renew until 03/21/19 05:32:15
root@krb:~#
```

Fig 3. Kerberos authentication successful

### 3. Configuring the Single Sign-On and securing the network communications

In this phase, we actually configure the Single Sign-On by setting up the SSH server to use the kerberos and ldap machines to allow an user to login from any machine to a user whose details are stored in the ldap database. A kerberos TGT (Ticket granting ticket) is generated for each user when they attempt to login.

Once, the Single Sign-On is configured and deployed, we focus on securing the network communications in the environment. DNS was implemented when security was not a concern and hence, the default implementation is not secure. Thus, an attacker can easily exploit these properties of the default configuration to fake a DNS response and this poses a threat to the ecosystem. Thus, we implement the DNS security extension called DNSSEC on our servers. This will enable servers to support public-key cryptography such that a client machine on our network can verify the response and trace if it was actually sent from the real DNS server.

Once the DNS servers are secured, we move on to enable LDAPS (S for secure) by using TLS certificates to encrypt all LDAP traffic. We will also have to modify the ldap client containers (in this case, all containers which use LDAP in any way are ldap clients) to communicate with ldap over TLS. In our current configuration, we are using LDAP in collaboration with Kerberos. Thus, we also have to configure kerberos server with the updated settings.

#### 4. Install and secure the web server

We use nginx as our web server engine in this project. An enterprise usually has an internal portal that is hosted on a web server. We will host a WordPress application and use a MySQL database on the backend to support the application. After the nginx server is installed and the WordPress application is set up, we will try and access the test web page to check functionality.

Additionally, since we do not want to end up having two user databases, we will also connect LDAP to WordPress to use the same database for login purposes. WordPress has a LDAP plugin which enables it to connect and use LDAP support to authenticate a user. We also always configure the client machine to



use all the services that we installed and setup. We also use a local CA to assign a certificate to the Wordpress website thereby enabling it to use TLS.

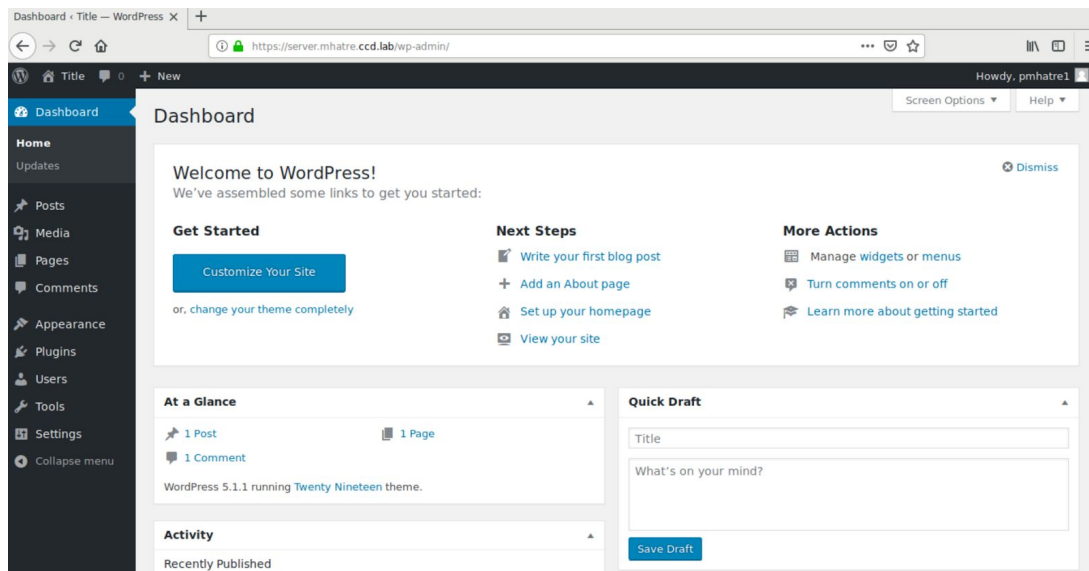
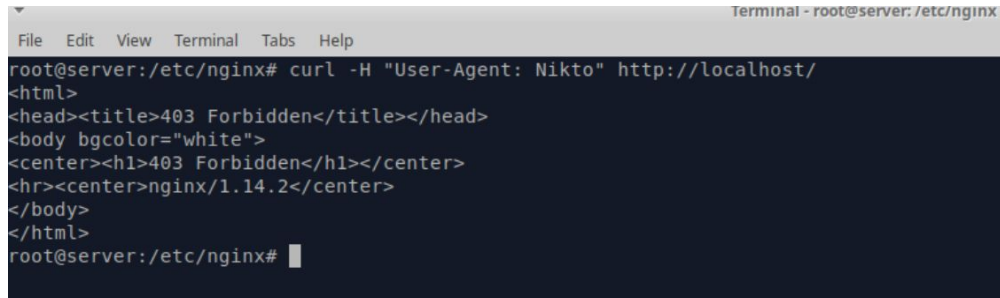


Fig 4. TLS on nginx server with WordPress successful

Once the web server is up and ready to use, we install a WAF (Web application Firewall) - ModSecurity to secure the web server. We download, install and test the functioning of ModSecurity on the web server. This process can be a bit lengthy as ModSecurity has to be cloned from its github and compiled from its source code and we have to use the nginx connector that enables nginx to integrate with ModSecurity library. Then we have to modify some nginx files and include ModSecurity in them to enable nginx to load ModSecurity and point it to the ModSecurity rules files. The final step towards completing this process is to test the working of ModSecurity.

To do this, we install a test rule and check if the rule actually blocks the test connection as required by the test rule. Once this is done, we import and install the OWASP (Open Web Application Security Project)

Core rule set on ModSecurity. In practice, the companies create their own configurations on top of the OWASP Core rule set as it is a recognized rule set that blocks a lot of the common attacks.

A terminal window titled "Terminal - root@server: /etc/nginx" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The prompt is "root@server:/etc/nginx#". The command entered is "curl -H 'User-Agent: Nikto' http://localhost/". The output is an HTML response from nginx: <html><head><title>403 Forbidden</title></head><body bgcolor='white'><center><h1>403 Forbidden</h1></center><hr><center>nginx/1.14.2</center></body></html>. The prompt returns to "root@server:/etc/nginx#".

```
Terminal - root@server: /etc/nginx
File Edit View Terminal Tabs Help
root@server:/etc/nginx# curl -H "User-Agent: Nikto" http://localhost/
<html>
<head><title>403 Forbidden</title></head>
<body bgcolor="white">
<center><h1>403 Forbidden</h1></center>
<hr><center>nginx/1.14.2</center>
</body>
</html>
root@server:/etc/nginx#
```

Fig 5. OWASP core rule set works

## 5. Place and configure the network firewall

We use iptables - The Linux kernel firewall as the firewall for this ecosystem. We decide that the placement of the firewall would be external facing (towards the internet) and hence setup the firewall on the Xubuntu machine which in this scenario simulates like the gateway for the container based ecosystem. Now, we set the default policy of the firewall to block all incoming connections and we move to allow only the required connections. The required connections from the firewall are as follows:

- Access to ports used by SSH, HTTP, HTTPS and port 2222 on the firewall (Xubuntu)
- Redirect incoming HTTP and HTTPS requests to the server container
- Redirect incoming SSH requests to the server container
- Redirect incoming requests to port 2222 to the client container's SSH service

Once the above configuration is done, no other communication except the HTTP, HTTPS requests and SSH requests on port 22 and 2222, will be allowed in the corporate environment. If required, we can also specify which IP addresses can send the requests for the above.

```
p@paddict: ~  
File Edit View Search Terminal Help  
p@paddict:~$ ssh pmhatre1@192.168.176.129  
pmhatre1@192.168.176.129's password:  
Last login: Mon Apr  1 12:55:46 2019 from 192.168.176.1  
pmhatre1@server:~$ logout  
Connection to 192.168.176.129 closed.  
p@paddict:~$ ssh pmhatre1@192.168.176.129 -p 2222  
pmhatre1@192.168.176.129's password:  
Last login: Mon Apr  1 13:11:22 2019 from 192.168.176.1  
pmhatre1@client:~$ logout  
Connection to 192.168.176.129 closed.  
p@paddict:~$
```

Fig 6. iptables firewall rules work

Thus, in the above five phases, we have simulated a corporate ecosystem, configured various services to enable DNS, login and authentication - LDAP and Kerberos, implemented Single Sign-On, installed web server and WordPress, integrated ModSecurity firewall with nginx web server, decided, placed and configured iptables firewall to complete this project.

## Reflection

This project, performed as a part of the “**Competitive Cyber Defence**” course under **Dr. Thomas Moyer**, taught me to understand and implement real world tools that are used in today’s corporate world. Almost all corporate networks, universities, groups, etc. have a Single Sign-On implementation and house multiple firewalls at various locations in the network.

While performing this project, I learned to set up local DNS servers and to implement public key encryption on the nameserver communication. I also learned what the LDAP and Kerberos protocols are and how both of these protocols can be used together to achieve a Single Sign-On login system. I learned to actually implement these protocols using OpenLDAP and Krb5 packages in a Linux environment. Additionally, I implemented and configured the TLS encryption layer over the OpenLDAP and Krb5 packages. I learned how to configure client machines to leverage and use the DNS, LDAP and Kerberos servers to resolve domain names and authenticate a user.

Another part of the project was to setup a WordPress application on a nginx server block to simulate a real world web site and I learned how to configure and use nginx servers, create a local CA (Certificate Authority) and use the local CA to sign certificates so that the web server can securely communicate with encrypted packets with the other machines in the network. Then I also understood what WAF (Web Application Firewall) is and learned to use one WAF - ModSecurity. I learned how to integrate ModSecurity with nginx server. This task was especially difficult because there is very little and confusing documentation on the integration of ModSecurity and nginx server. Once I understood how to integrate ModSecurity and the nginx servers, I also imported the OWASP core rule set and tested its functionality.

Once this was done, the final phase was to deploy a network based packet filtering firewall - “iptables” in the network. Understanding this firewall and deploying it was a big learning curve and it also helped me a lot in one of my other courses. This will also help me in my real world career as designing, configuring and maintaining a firewall is very important in enterprise security.

One very important skill that I gained while doing this project is “Troubleshooting”. This skill is very important and a good troubleshooter can always get a task done quickly with more efficiency. Troubleshooting requires various skills to be used together in conjunction. Skills that are useful in this task are - having an eye for detail, always observing the outcome of whatever you are performing, trying to understand why a particular action lead to a particular outcome - i.e. trying to understand how a particular system works on the lower level, researching about the issues you face - online, documentation, etc. and coming up with solutions that can either work or fail. The idea here is to fail multiple times till you succeed once.

I was also able to leverage the techniques and technologies I learned in other coursework to complete this project. I learned the purpose and usage of Certificate Authorities and the use of TLS to secure communications in “**ITIS 6240: Applied Cryptography**”. I also learned the use of various authentication mechanisms in “**ITIS 6010: Wireless Security**” that lead me to better understand the functioning of the project as a whole. Additionally, I had learned various of the security principles and usage of protocols in the course “**ITIS 6200: Principles of Information Security and Privacy**”.

Additionally, I was able to use the skills I learned in this class to design a Project for a class that I worked as a Teaching Assistant for. I created a Firewall based activity - “**Monitor, control and redirect the**

**traffic flowing via a smart router**". I was also able to use the skills in other courses, most notably **"ITIS 6167: Network Security"**. I also used the things I learned here when I participated in the UNC Charlotte's Blue Team at the Department of Energy's CyberForce Competition at Oak Ridge National Laboratory.

This project evolved a keen interest in me and I realized how complex an enterprise's network can be. This led to me gaining interest in securing and exploitation of the corporate network and I am today certain that I would love to work in a Red team that tries to find faults with an enterprise's network in order to make it more secure.