Chapter – 3
Architecture Setup and Experiment

3.1 Introduction

The first thing required for our research work is an email server. The email server is used for sending and receiving emails to various domains. Using our own email server, we could set up and test various anti-spoofing protocols like SPF, DKIM and DMARC. Also, we could perform experiments and find possible ways to bypass these anti-spoofing protocols which attackers use to spoof an email.

3.2 Setup of Email Server

The technical specifications of our Email Server are as follows:

Domain Name: https://www.mail.mail-server.in

Public IP Address: 117.240.215.158

Local IP Address: 172.24.12.42

Email Server: Modoboa version 1.17.0

Operating System: Ubuntu 20.04.3 LTS

Database Server: PostgreSQL

Web Server: Nginx

Other Important Packages Installed: Dovecot, Postfix, OpenDKIM

Modoboa is a free and open-source email hosting and management platform that integrates with the Postfix SMTP server and the Dovecot IMAP/POP3 server [74]. Modoboa is written in Python and licensed under the terms of the ISC license. The main elements of Modoboa are as follows [25]:

1. By default, Modoboa serves the webmail client and web-based admin panel via the Nginx web server.
2. Compatible with Postfix and Dovecot.
3. Support MySQL/MariaDB, or PostgreSQL database.

4. Easily create unlimited mailboxes and unlimited mail domains in a web-based admin panel.
5. Easily create email alias in the web-based admin panel.
6. The webmail client includes an easy-to-use message filter that allows you to sort messages into multiple folders.
7. It can help you protect your domain reputation by monitoring email blacklists and generating DMARC reports, so your emails have a better chance to land in the inbox instead of the spam folder.
8. Includes amavis frontend to block spam and detect viruses in email.
9. Calendar and address book.
10. Integration with Let's Encrypt.
11. A policy daemon for Postfix that allows you to define daily sending limits for domains and individual accounts.
12. Includes AutoMX to allow end-users to easily configure mail account in a desktop or mobile mail client.

Setup of email server is done in following steps:

3.2.1 Installation of Email Server

Following steps were used for the installation of Modoboa on ubuntu 20.04 [75]:

1. Preparing the System

Modoboa relies on other tools, some of which require compilation; therefore, we'll need a compiler and a few C libraries [76]. Be careful to install the following system packages: python3-dev, libxml2-dev, libxslt-dev, libjpeg-dev, librrd-dev, rrdtool, libffi-dev, libssl-dev, pkg-config, and libcairo2-dev

Then, as demonstrated in the commands below, create a dedicated system user and install a virtual environment to install the application, which will isolate it (and its dependencies) from the rest of your system.

```
# apt-get install virtualenv python3-pip
# useradd modoboa  # create a dedicated user
# sudo -u modoboa -i bash    # log in as the newly created user
$ virtualenv --python python3 ./env  # create the virtual environment
$ source ./env/bin/activate          # activate the virtual environment
```

2. Modoboa

Next, install the Modoboa using the following command:

```
(env)$ pip install modoboa
```

3. Database Server

Modoboa is compatible with the following databases:

1. PostgreSQL
2. MySQL / MariaDB
3. SQLite

We selected PostgreSQL as our database server and installed using the following commands:

```
(env)$ pip install psycopg[binary]>=3.1
```

```
# sudo -l -u postgres createuser --no-createdb modoboa
# sudo -l -u postgres createdb --owner=modoboa modoboa
```

Then, create a user and a database. For example, to create the Modoboa database owned by a *Modoboa* user, run the following SQL commands:

```
CREATE DATABASE modoboa;
CREATE USER 'modoboa'@'localhost' IDENTIFIED BY 'my-strong-password-here';
GRANT ALL PRIVILEGES ON modoboa.* TO 'modoboa'@'localhost';
```

modoboa-admin.py is a command-line program for deploying a fully functional Modoboa site. To create a new instance in./instance, we just run the following command:

```
(env)$ modoboa-admin.py deploy instance --collectstatic \
         --domain <hostname of your server> --dburl default:<database url>
```

4. Cron Jobs

Modoboa requires a few ongoing jobs to function properly. Create a new file, for example, /etc/cron.d/modoboa, and place the following content inside:

```
#
# Modoboa specific cron jobs
#
PYTHON=<path to Python binary inside the virtual environment>
INSTANCE=<path to Modoboa instance>

# Operations on mailboxes
*     *  *  *  *  <mailbox user>    $PYTHON $INSTANCE/manage.py handle_mailbox_operations

# Generate DKIM keys (they will belong to the user running this job)
*     *  *  *  *  root     umask 077 && $PYTHON $INSTANCE/manage.py modo manage_dkim_keys

# Sessions table cleanup
0     0  *  *  *  root  $PYTHON $INSTANCE/manage.py clearsessions
# Logs table cleanup
0     0  *  *  *  root  $PYTHON $INSTANCE/manage.py cleanlogs
# Logs parsing
*/15  *  *  *  *  root     $PYTHON $INSTANCE/manage.py logparser &> /dev/null
0     *  *  *  *  modoboa  $PYTHON $INSTANCE/manage.py update_statistics
# DNSBL checks
*/30  *  *  *  *  modoboa  $PYTHON $INSTANCE/manage.py modo check_mx
# Public API communication
0     *  *  *  *  modoboa  $PYTHON $INSTANCE/manage.py communicate_with_public_api
```

5. **Policy Daemon**

Modoboa includes a built-in Policy Daemon for Postfix that allows you to set daily sending limitations for domains and accounts. We can start it manually with the following command:

```
(env)> python manage.py policy_daemon
```

6. **RQ Daemon**

Modoboa employs RQ as a job handler for asynchronous operations. We can start it manually with the following command:

```
(env)> rq worker high default low
```

7. **Web Server**

Before setting up Nginx, first download and install *uwsgi* using following command:

```
apt install uwsgi uwsgi-plugin-python3
```

To install the nginx server use the following commands:

```
sudo apt update
sudo apt install nginx
```

Here is a sample nginx configuration:

```
upstream modoboa {
  server unix:<uwsgi_socket_path> fail_timeout=0;
}

server {
    listen 80;
    listen [::]:80;
    server_name <hostname>;
    rewrite ^ https://$server_name$request_uri? permanent;
}

server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name <hostname>;
    root <modoboa_instance_path>;

    ssl_certificate  <ssl certificate for your site>;
    ssl_certificate_key  <ssl certificate key for your site>;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers "ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-
    ssl_prefer_server_ciphers on;
    ssl_session_cache shared:SSL:10m;
    ssl_verify_depth 3;
    ssl_dhparam /etc/nginx/dhparam.pem;

    client_max_body_size 10M;

    access_log /var/log/nginx/modoboa-access.log;
    error_log /var/log/nginx/modoboa-error.log;

    location /sitestatic/ {
        try_files $uri $uri/ =404;
    }

    location /media/ {
        try_files $uri $uri/ =404;
    }

    location ^~ /new-admin {
        alias  <modoboa_instance_path>/frontend/;
        index  index.html;

        expires -1;
        add_header Pragma "no-cache";
        add_header Cache-Control "no-store, no-cache, must-revalidate, post-c

        try_files $uri $uri/ /index.html = 404;
    }

    location / {
        include uwsgi_params;
        uwsgi_param UWSGI_SCRIPT instance.wsgi:application;
        uwsgi_pass modoboa;
    }
    %{extra_config}
}
```

8. Dovecot

Modoboa requires Dovecot 2+ to function, and dovecot's configuration is stored in /etc/dovecot; all paths will be relative to this directory. To install dovecot, run the following command::

```
apt install dovecot-imapd dovecot-lmtpd dovecot-managesieved dovecot-sieve
```

First, edit the conf.d/10-mail.conf and set the mail location variable:

```
# maildir
mail_location = maildir:~/Maildir
```

Edit the inbox namespace of 15-mailboxes.conf to have dovecot automatically create the regular folders when an account is created.

```
mailbox Drafts {
  auto = subscribe
  special_use = \Drafts
}
mailbox Junk {
  auto = subscribe
  special_use = \Junk
}
mailbox Sent {
  auto = subscribe
  special_use = \Sent
}
mailbox Trash {
  auto = subscribe
  special_use = \Trash
}
```

Dovecot 2.1+ guarantees that all special mailboxes are immediately established for new accounts.

To make authentication function, modify the conf.d/10-auth.conf file and uncomment the following line at the end:

```
#!include auth-system.conf.ext
!include auth-sql.conf.ext
#!include auth-ldap.conf.ext
#!include auth-passwdfile.conf.ext
#!include auth-checkpassword.conf.ext
#!include auth-vpopmail.conf.ext
#!include auth-static.conf.ext
```

Next, we need to install “Postgres” connector for dovecot using following command:

sudo apt install dovecot-pgsql

Postfix communicates with Dovecot using the Local Mail Transport Protocol (LMTP). First, check the protocol settings (usually found in dovecot.conf). It should be similar to the example below:

```
protocols = imap pop3 lmtp
```

Then, open the conf.d/10-master.conf, look for lmtp service definition and add the following content inside:

```
service lmtp {
  # stuff before
  unix_listener /var/spool/postfix/private/dovecot-lmtp {
    mode = 0600
    user = postfix
    group = postfix
  }
  # stuff after
}
```

Modoboa allows users to specify filtering rules using the web interface. To do so, “Manage Sieve” must be activated on your server. In conf.d/20-managesieve.conf, make sure the following lines are uncommented:

```
protocols = $protocols sieve

service managesieve-login {
  # ...
}

service managesieve {
  # ...
}

protocol sieve {
  # ...
}
```

9. Postfix

First, we must generate configuration files (or map files) that Postfix will utilize to lookup Modoboa tables. To produce the necessary map files and put them in a directory, use the following command:

```
> cd <modoboa_instance_path>
> python manage.py generate_postfix_maps --destdir <directory>
```

10. OpenDKIM

Modoboa generates DKIM keys for hosted domains but does not sign or verify messages. To do this, we'll need specialist software like OpenDKIM.

Because key-related information is kept in Modoboa's database, we must describe how OpenDKIM will access it.

First, install the needed extra packages on your machine (libopendbx1-*), and then enter the following SQL view into Modoboa's database:

```
CREATE OR REPLACE VIEW dkim AS (
  SELECT id, name as domain_name, dkim_private_key_path AS private_key_path,
       dkim_key_selector AS selector
  FROM admin_domain WHERE enable_dkim
);
```

We may find OpenDKIM's configuration file at /etc/opendkim.conf. Add the following content to it and eventually, reload OpenDKIM.

```
KeyTable              dsn:<driver>://<user>:<password>@<db host>/<db name>/table=dkim?keycol=id?c
SigningTable          dsn:<driver>://<user>:<password>@<db host>/<db name>/table=dkim?keycol=doma
Socket                inet:12345@localhost
```

Add the following lines to the /etc/postfix/main.cf file and reload postfix:

```
smtpd_milters = inet:127.0.0.1:12345
non_smtpd_milters = inet:127.0.0.1:12345
milter_default_action = accept
milter_content_timeout = 30s
```

3.2.2 Reverse DNS (rDNS) setup

If a cop pulls you over, the first thing they will ask for is your license and registration. If the name on the vehicle registration matches the name on the driver's license, the cop can identify the car owner. Reverse DNS operates in the same way [77]. The rDNS is a type of email authentication that is used to match our mail server IP to your host name as shown in *Figure 21*.

Figure 21: Reverse DNS process flow

When we send an email, your recipient's mail server checks to determine if the Sending IP address matches the domain name specified in the HELO command. This is also known as "HELO to IP" [77].

A mail server sends the HELO instructions to start the email-sending operation. It is used to match your domain name to your Mail Server IP address. rDNS is crucial because it lends credibility to the mail server that sends emails and serves as an additional layer of email authentication. rDNS allows us to separate legitimate email senders from compromised servers that disseminate spam [77].

Another advantage of rDNS is that we do not have to accept the entire message body to perform a rDNS lookup. We simply need the information in your message header, which is sent at the start of an email chain. If the email fails rDNS, the email server can decline it, saving server resources and keeping spam from reaching the inbox. [77].

Several major mailbox providers, including Gmail, Microsoft, and Yahoo, will block emails from mail servers that do not have proper rDNS entries. Additionally, some SMTP servers are set up to reject emails when the rDNS does not match the HELO. However, keep in mind that mailbox providers prioritize total IP address and domain reputation when selecting where emails should be sent. rDNS is just one of the variables that will define where our message should be sent [77].

Type	IP Address	Domain Name	TTL
PTR	117.240.215.158 Unknown (AS9829)	mail-server.in	60 min

Figure 22: Result of reverse DNS lookup

To set up rDNS, our sending IP must have a pointer (PTR) record in your DNS that resolves to a valid host name. A pointer record, or PTR record, converts an IP address into a fully qualified domain name (FQDN). It is the opposite of the A record and is used for reverse DNS lookups, which can aid in spam filtering. This can be done with the help of the Internet Service Provider (ISP) who is responsible for supplying us the live IP address which is user for the mail server. The reverse DNS entry can only be done by the ISP providing the public IP address which is BSNL in our case.

So, with the help of BSNL we setup reverse DNS entry for IP address 117.240.215.158 to point to our mail server i.e. mail-server.in as seen in *Figure 22*.

After successful completion of all the above steps, our mail server is ready for use as shown in *Figure 23*.

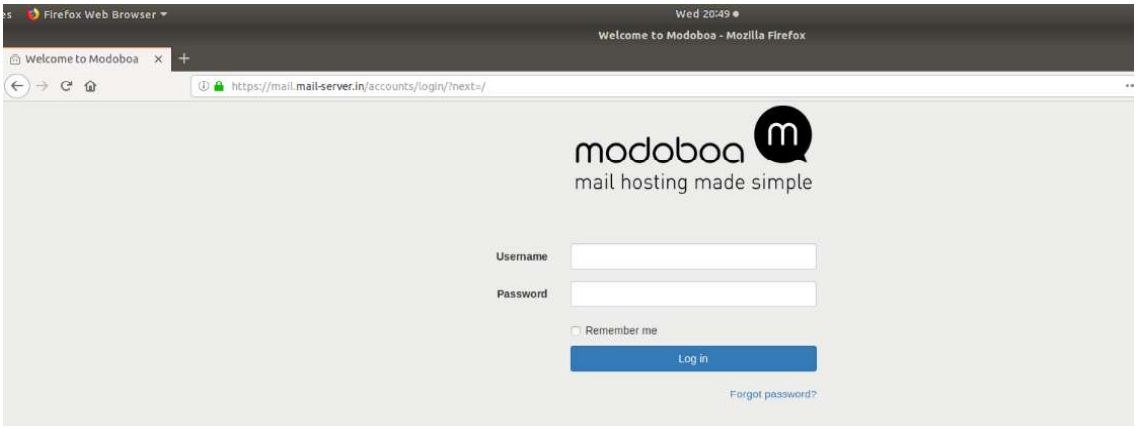


Figure 23: mail-server.in email server home page

3.2.3 Setup of DKIM key for signing emails

DKIM (DomainKeys Identified Mail) uses a private key to digitally sign emails sent from your domain. Receiving SMTP servers check the signature against the public key included in the DNS DKIM record.

We triggered DKIM signing when we added the domain name to the Modoboa admin panel earlier, thus the signature is complete. The only thing left to do is set up a DKIM entry in DNS Manager. First, visit the Modoboa admin page and select your domain name. *Figure 24* illustrates how to click the Show key button in the DNS section.

Figure 24: DKIM key setup in Modoboa

The public key will be revealed. There are two formats. We simply need the Bind/named format. Go to your DNS management, add a TXT record, and type modoboa._domainkey into the Name field. (Remember how we used Modoboa as the selector while adding

domain names in the admin panel. Copy everything inside the parenthesis and paste it into the value area. Remove all double quotes. Our DNS manager may ask you to remove more incorrect characters, such as carriage returns.

3.2.4 Modification in Configuration Files

1. **Modification in configuration files to send email to outside domain**

   By default, the Modoboa email server using postfix will send emails only to its local domains configured on the local mail server and not to the actual domains outsider over the internet. Following steps are to be executed for disabling local email delivery:

   - open /etc/opendkim/main.cf
   - remove $myhostname or $mydomain from mydestination
   - comment following lines

     virtual_mailbox_domains=$virtual_mailbox_maps,hash:/var/spool/postfix/plesk/virtual_domains

     virtual_alias_maps = $virtual_maps, hash:/var/spool/postfix/plesk/virtual

     virtual_mailbox_maps = hash:/var/spool/postfix/plesk/vmailbox
   - Restart postfix

2. **Modification in configuration files to sign fake domain email**

   By default, the mail server will sign only the domain specified in the “From” field of the sending email. But for testing of spoofing of email, we modified the signing table used by OpenDKIM so that it signs the email message key of with whatever domain we wish instead of the original key.

   For example, by modifying the config files we could send an email from abc@msubaroda.ac.in signed by the private key of the domain mail-server.in instead of the private key for the domain msubaroda.ac.in. A preview of the modified configuration file of OpenDKIM used by our email server for signing emails sent via postfix can be seen in *Figure 25*. The configuration file of OpenDKIM describes the path to Key Table and Signing Table. The Key Table maps domain name with the respective private key available.

```
# This file was automatically installed on 2020-11-07T16:13:50.504943
# This is a basic configuration that can easily be adapted to suit a standard
# installation. For more advanced options, see opendkim.conf(5) and/or
# /usr/share/doc/opendkim/examples/opendkim.conf.sample.

# Log to syslog
Syslog                  yes
SyslogSuccess           Yes
LogWhy                  Yes
LogResults  Yes

# Required to use local socket with MTAs that access the socket as a non-
# privileged user (e.g. Postfix)
UMask                   007

# Sign for example.com with key in /etc/dkimkeys/dkim.key using
# selector '2007' (e.g. 2007._domainkey.example.com)
#Domain         mail-server.in
KeyTable        /var/lib/dkim/dkim_key_table
SigningTable    refile:/var/lib/dkim/dkim_signing_table
#KeyList                /var/lib/dkim/dkim_domains.key
#Selector       modoboa
```

Figure 25: Preview of OpenDKIM configuration file

As seen in *Figure 26,* keyname2 has a mapping of domain name “mail-server.in” with the selector i.e. “modoboa” and the location of the private key i.e. /var/lib/dkim/mail-server.in.pem.

```
keyname1 gujaratbloodgroup.com:modoboa:/var/lib/dkim/gujaratbloodgroup.com.pem
keyname2 mail-server.in:modoboa:/var/lib/dkim/mail-server.in.pem
keyname3 msubaroda.ac.in:modoboa:/var/lib/dkim/msubaroda.ac.in.pem
```

Figure 26: Preview of OpenDKIM Key Table

Now, to sign an email sent from a different domain name than that of our email server, with private key of our own domain name we can modify the signing table. As seen in *Figure 27*, except the first domain, we mapped all the other domains with “keyname2” which indicates that private key of “mail-server.in” domain is to be used for signing all the emails sending from any of those mentioned in the signing table. Wildcard character “*” is used to indicate the same working for any email id for the specified domain names.

```
*@gujaratbloodgroup.com keyname1
*@mail-server.in keyname2
*@msubaroda.ac.in keyname2
*@gmail.com keyname2
*@yahoo.co.in keyname2
*@yahoo.com keyname2
*@gujarattourism.com keyname2
*@testab.com keyname2
*@outlook.com keyname2
*@orthocarehospital.in keyname2
```

Figure 27: Preview of OpenDKIM Signing Table

3.3 Implementation of Anti-Spoofing Protocols

After the successful setup of our own email server, we set up all the three anti-spoofing algorithms in our email server. For this we published respective entries of SPF, DKIM and DMARC in DNS of our domain mail-server.in as shown in *Figure 28*, *Figure 29 and Figure 30* respectively. For SPF, we mentioned live IP address of our email server i.e. 117.240.215.158 which will be used to inform the receiving email server that any email coming from an email ID of "mail-server.in" domain must only come from the IP address mentioned in our SPF record.

mail-server.in.	3600	IN	TXT	v=spf1 ip4:117.240.215.158

Figure 28: DNS entry of SPF for domain mail-server.in

In DKIM record, we inserted the public key previously generated from OpenDKIM with the selector "modoboa". This public key can be used by the receiving email servers to decrypt the emails claiming to have come from any email id of our email server "mail-server.in"

modoboa._domainkey.mail-server.in.	14400	IN	TXT	v=DKIM1; k=rsa; p=MIICIjANBgk qhkiG9w0BAQEFAAOCAg8AMIICCg KCAgEAl2yboh/jZ7GiAlUdhK1Lukr6 RX2sRwxSmxOYK55sJfE2lMSKD29 45AHOWodb5OTWhZ+hj0Ognpb1e /quRyAEVIQLKdTS+Dotn93ngtapQ 6tEiODjf41bCfquIBOJtRaNe0KjweiZ YN3ogMLKIfT46fmgw6ww+OH2F4 pgHXGFXLZW9FXLcn21H9lhLopSz mv9npQYWxGrCABgjDO9f9isweGj WmNegCD88rcugIk5kzxcgW0XAR2 EoU7IiqUVOHx4TlHC0VMiNagbpeo 0K+dENeC3DU/h6Z1RmRZQPgtRi4 vFZ4jzUhr534GBgWX3UQC4cTsnG yRq00isBKkjF1WW1P1yT0EBHkP7d UP3IFYJTl7s9y0mjUXjtTAp610Iu4Q scMwsKZLE4UxE4PDeMP2gfMYQw mipbnb9e9EwrXeasZ0seTZPPFDrm c4VdlczVOqwiq2OSGt7KHMjxNKj8 ZmIQSvvolS/XrtGtZ21iAU+Tl6iFpB F/WgW7X0HTsRQVTT3MNCCGRkoP NryFwPczOqWH9bw2f7GL3CGLBtjX 43PE36MQ1c+ppmUaWbzIfOzDP6L Gbt2XqGW0Xte4+9iTsEeye4ak6cD BS7+uv4EC4OcMPCCGlxTEUzRkTV gFt02UOv79YO6Kbvgq4q8dy2H7lL x1dpdwxq2w+JGOVLy0SB+8XkCA wEAAQ==

Figure 29: DNS entry of DKIM for domain mail-server.in

_dmarc.mail-server.in.	14400	IN	TXT	v=DMARC1;p=none;sp=none;adkim=r;aspf=r;

Figure 30: DNS entry of DMARC for domain mail-server.in

The DMARC policy that we used for initial testing purpose mentions p=none means no action is to be taken in case of failure of SPF and/or DKIM of an email from our domain. The entry of "sp=none" represents the same policy of no action for the subdomains of "mail-server.in". Also, "adkim=r" and "aspf=r" means a relaxed policy for alignment of domain names in SPF and DKIM both.

3.4 Testing of Delivery of Email

After the successful implementation of all the three anti-spoofing protocols, SPF, DKIM and DMARC to our email server we tested the delivery of emails to various other email servers sent from our own email server. On our email server we added various other domain names also for testing email spoofing and we were successfully able to send spoofed email to the inbox of recipients under various conditions.

3.4.1 Regular email from domain not having any anti-spoofing protocols

From: prashant@mail-server.in

To: prashantchauhan25@gmail.com

SPF, DKIM, DMARC: Not Published for domain mail-server.in

First, we selected our own email server i.e. "mail-server.in" with none of the anti-spoofing protocols mentioned in the DNS of our domain.

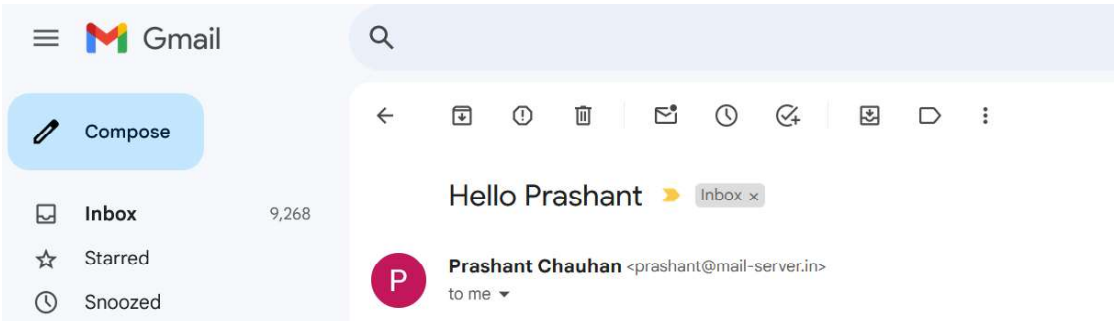
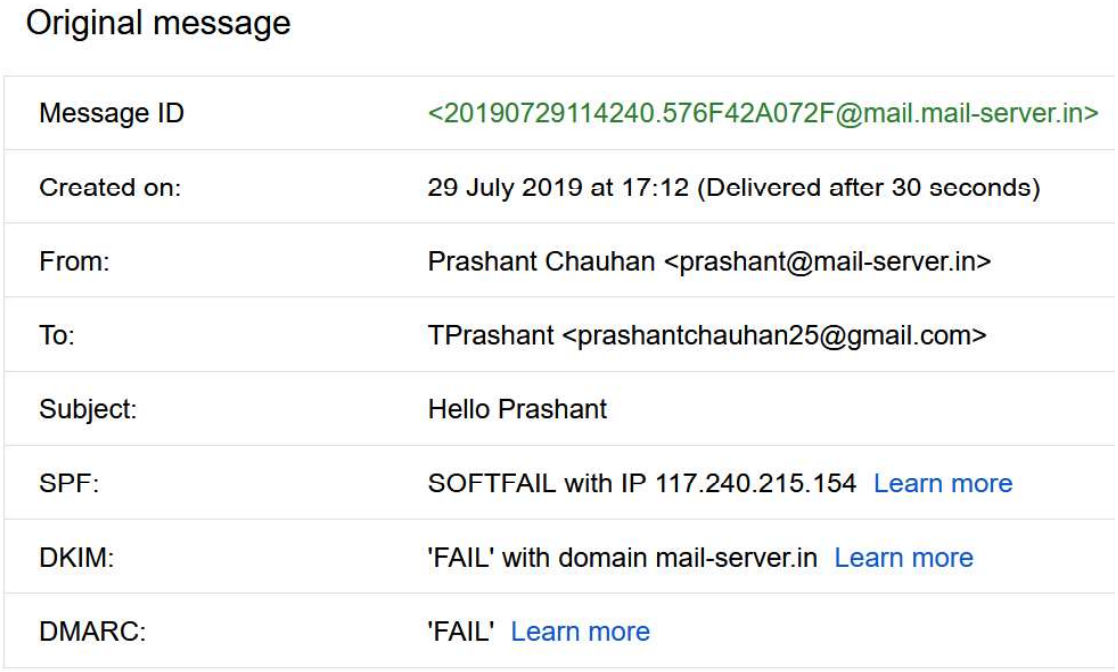


Figure 31: Regular email delivery in inbox for domain "mail-server.in" without having any anti-spoofing protocols

As seen in *Figure 31*, our email was successfully sent to the inbox of the recipient user. Since at this moment we had not published SPF, DKIM and DMARC records in our DNS, none of these tests were passed for the received email as can be seen in *Figure 32* and Figure 33.



Original message

Message ID	<20190729114240.576F42A072F@mail.mail-server.in>
Created on:	29 July 2019 at 17:12 (Delivered after 30 seconds)
From:	Prashant Chauhan <prashant@mail-server.in>
To:	TPrashant <prashantchauhan25@gmail.com>
Subject:	Hello Prashant
SPF:	SOFTFAIL with IP 117.240.215.154 Learn more
DKIM:	'FAIL' with domain mail-server.in Learn more
DMARC:	'FAIL' Learn more

Figure 32: Raw email summary for domain "mail-server.in" without having any anti-spoofing protocols

```
Received-SPF: softfail (google.com: domain of transitioning prashant@mail-server.in does not designate 117.240.215.154 as
permitted sender) client-ip=117.240.215.154;
Authentication-Results: mx.google.com;
       dkim=temperror (no key for signature) header.i=@mail-server.in header.s=default header.b=jCU1OghS;
       spf=softfail (google.com: domain of transitioning prashant@mail-server.in does not designate 117.240.215.154 as
permitted sender) smtp.mailfrom=prashant@mail-server.in;
       dmarc=fail (p=NONE sp=NONE dis=NONE) header.from=mail-server.in
Received: from localhost (mail.mail-server.in [127.0.0.1]) by mail.mail-server.in (Postfix) with ESMTP id 576F42A072F for
```

Figure 33: Raw email details for domain "mail-server.in" without having any anti-spoofing protocols

3.4.2 Regular email from domain having all three anti-spoofing protocols

From: prashant@mail-server.in
To: prashantchauhan25@gmail.com
SPF, DKIM, DMARC: Published for domain mail-server.in

Next, we published all three anti-spoofing protocols i.e. SPF, DKIM and DMARC in our DNS records for the domain "mail-server.in" and then checked the delivery of email sent from our email server.

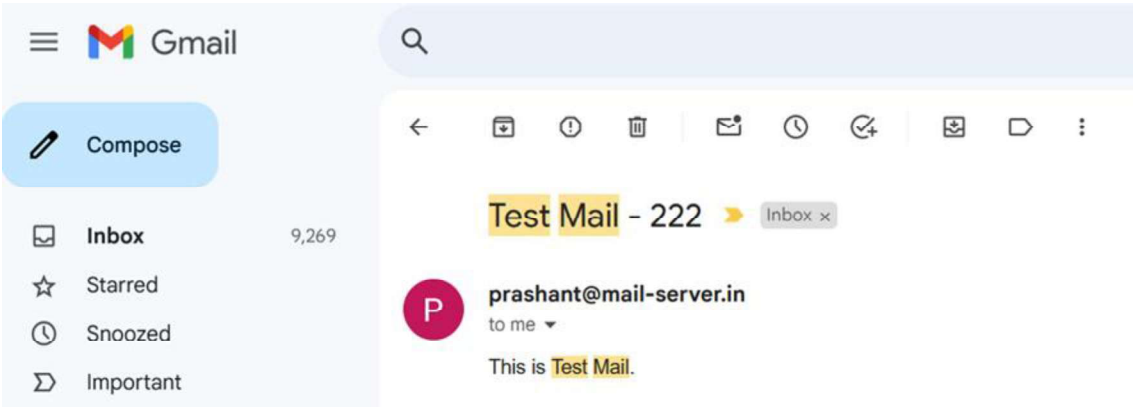


Figure 34: Regular email delivery in inbox for domain "mail-server.in" with all 3 anti-spoofing protocols

Original message

Message ID	<60f67b1f.1c69fb81.b72d0.5498SMTPIN_ADDED_MISSING@mx.google.com>
Created on:	20 July 2021 at 12:58 (Delivered after 5 seconds)
From:	prashant@mail-server.in
To:	prashantchauhan25@gmail.com
Subject:	Test Mail - 222
SPF:	PASS with IP 117.240.215.158 Learn more
DKIM:	'PASS' with domain mail-server.in Learn more
DMARC:	'PASS' Learn more

Figure 35: Raw email summary for domain "mail-server.in" with all 3 anti-spoofing protocols

As seen in *Figure 34*, our regular email sent from the email ID "prashant@mail-server.in" to "prashantchauhan25@gmail.com" was successfully sent to the inbox of the recipient user. When seen in detail of the received email, we found that all the three tests for SPF, DKIM and DMARC were passed for the email as seen in *Figure 35 and* Figure 36.

```
Authentication-Results: mx.google.com;
       dkim=pass header.i=@mail-server.in header.s=modoboa header.b=CD48BmJs;
       spf=pass (google.com: domain of prashant@mail-server.in designates 117.240.215.158 as permitted sender)
smtp.mailfrom=prashant@mail-server.in;
       dmarc=pass (p=QUARANTINE sp=QUARANTINE dis=NONE) header.from=mail-server.in
Message-ID: <60f67b1f.1c69fb81.b72d0.5498SMTPIN_ADDED_MISSING@mx.google.com>
Received: from localhost (localhost [127.0.0.1]) by mail-server.in (Postfix) with ESMTP id 8D0DC1402FB; Tue, 20 Jul 2021
12:58:28 +0530 (IST)
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed; d=mail-server.in; s=modoboa; t=1626766108;
```

Figure 36: Raw email details for domain "mail-server.in" with all 3 anti-spoofing protocols

3.4.3 Spoofed email from domain not having any anti-spoofing protocols

From: Prashant.chauhan-cse@msubaroda.ac.in
To: prashantchauhan25@gmail.com
SPF, DKIM, DMARC: Not Published for domain msubaroda.ac.in

After successfully testing email delivery for regular emails, we modified config files as already mentioned and tested for delivery of spoofed emails. First, we selected the domain "msubaroda.ac.in" which had none of the three anti-spoofing protocols published in the DNS of the domain. As seen in *Figure 37*, we were successfully able to send email from "prashant.chauhan-cse@msubaroda.ac.in", a domain not having any of the three anti-spoofing protocols published.

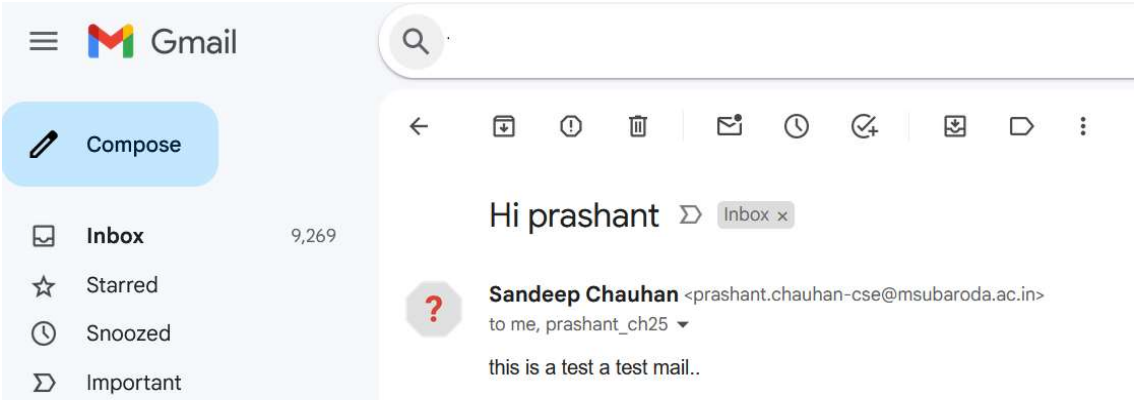


Figure 37: Spoofed email delivery in inbox for domain "msubaroda.ac.in" without having any anti-spoofing protocols

Original message

Message ID	<20190409112501.2500.18573@mail.mail-server.in>
Created on:	9 April 2019 at 16:55 (Delivered after 11 seconds)
From:	Sandeep Chauhan <prashant.chauhan-cse@msubaroda.ac.in>
To:	prashantchauhan25@gmail.com, prashant_ch25@yahoo.co.in
Subject:	Hi prashant
SPF:	NEUTRAL with IP 117.240.215.154 Learn more

Figure 38: Spoofed email summary for domain "msubaroda.ac.in" without having any anti-spoofing protocols

```
Received-SPF: neutral (google.com: 117.240.215.154 is neither permitted nor denied by best guess record for domain of
prashant.chauhan-cse@msubaroda.ac.in) client-ip=117.240.215.154;
Authentication-Results: mx.google.com;
       spf=neutral (google.com: 117.240.215.154 is neither permitted nor denied by best guess record for domain of
prashant.chauhan-cse@msubaroda.ac.in) smtp.mailfrom=prashant.chauhan-cse@msubaroda.ac.in
Received: from localhost (mail.mail-server.in [127.0.0.1]) by mail.mail-server.in (Postfix) with ESMTP id EBBAA2A34DA; Tue,
  9 Apr 2019 16:55:05 +0530 (IST)
```

Figure 39: Spoofed email details for domain "msubaroda.ac.in" without having any anti-spoofing protocols

As seen in *Figure 38* and Figure 39, the spoofed email sent by us from the domain "msubaroda.ac.in", didn't pass any of the three anti-spoofing protocols SPF, DKIM and DMARC. DKIM and DMARC tests were not even checked as no entry in DNS has been done for these.

3.4.4 Spoofed email from domain having all three anti-spoofing protocols

From: prashantchauhan25@gmail.com
To: Prashant.chauhan-cse@msubaroda.ac.in
SPF, DKIM, DMARC: Published for domain gmail.com

Next, we selected the email address "prashantchauhan25@gmail.com" for which SPF, DKIM and DMARC records are already published by the domain "gmail.com".

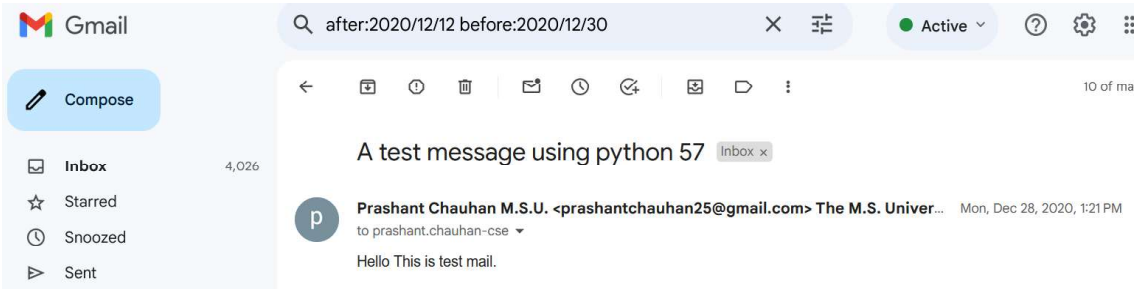
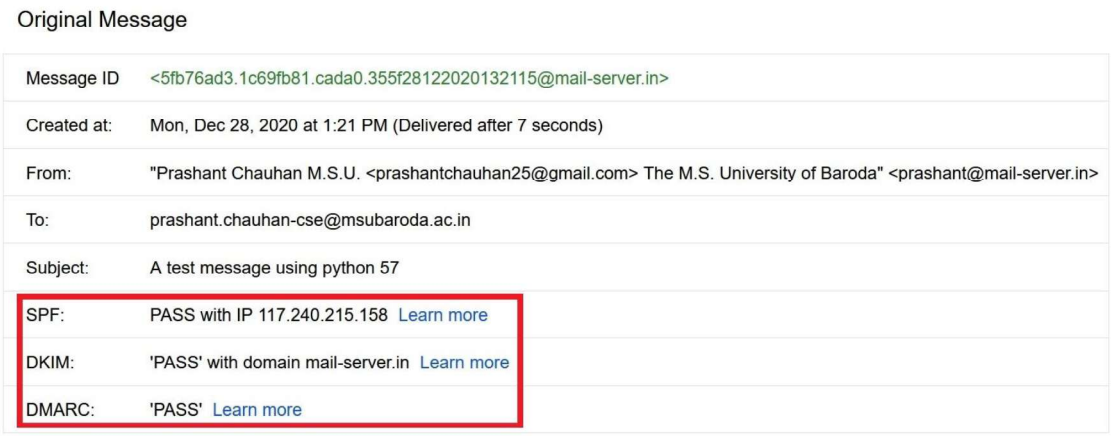


Figure 40: Spoofed email delivery in inbox for domain "gmail.com" with all 3 anti-spoofing protocols

As seen in *Figure 40*, our email was successfully delivered to the inbox of the recipient user. From *Figure 41* and *Figure 42*, we can see that our email successfully passed all the three anti-spoofing protocols SPF, DKIM and DMARC. Also, we used multiple email addresses in the "From" field to pass all the anti-spoofing protocols and to prevent any kind of warning message by the receiving email server. We added the email address "prashant@mail-server.in" in the "From" field so that the receiving email server detects "mail-server.in" and performs all the tests on this domain.


Original Message

Message ID	<5fb76ad3.1c69fb81.cada0.355f28122020132115@mail-server.in>
Created at:	Mon, Dec 28, 2020 at 1:21 PM (Delivered after 7 seconds)
From:	"Prashant Chauhan M.S.U. <prashantchauhan25@gmail.com> The M.S. University of Baroda" <prashant@mail-server.in>
To:	prashant.chauhan-cse@msubaroda.ac.in
Subject:	A test message using python 57
SPF:	PASS with IP 117.240.215.158 Learn more
DKIM:	'PASS' with domain mail-server.in Learn more
DMARC:	'PASS' Learn more

Figure 41: Raw email summary for domain "gmail.com" with all 3 anti-spoofing protocols

```
Received-SPF: pass (google.com: domain of prashant@mail-server.in designates 117.240.215.158 as permitted sender)
ip=117.240.215.158;
Authentication-Results: mx.google.com;
       dkim=pass header.i=@mail-server.in header.s=modoboa header.b=PeqZCsCF;
       spf=pass (google.com: domain of prashant@mail-server.in designates 117.240.215.158 as permitted sender)
smtp.mailfrom=prashant@mail-server.in;
       dmarc=pass (p=QUARANTINE sp=QUARANTINE dis=NONE) header.from=mail-server.in
Received: from localhost (localhost [127.0.0.1]) by mail-server.in (Postfix) with ESMTP id 08342140314 for <prasha
cse@msubaroda.ac.in>; Mon, 28 Dec 2020 13:21:20 +0530 (IST)
```

Figure 42: Raw email details for domain "gmail.com" with all 3 anti-spoofing protocols

3.5 Attacks on Anti-Spoofing Protocols

Since the inception of these anti-spoofing protocols, attackers are continuously trying to find ways to bypass one or more of these protocols and they have been successful to bypass all of these anti-spoofing protocols i.e. Sender Policy Framework (SPF), DomainKeys Identified Mail (DKIM) and Domain-based Message Authentication, Routing and Conformance (DMARC) by adopting different mechanisms.

3.5.1 How to Bypass SPF?

To bypass the SPF, we took advantage of the fact that SMTP uses two separate domains for displaying "mail-from" to users and as actual "mail-from". The "mail-from" written in the email body is used to check the SPF test as well as to display as "sender" to the recipient user. The "mail-from" sent as header is used to get the actual domain the email came from.

As shown in *Figure 43*, we sent "prashant@mail-server" as the actual sender email address in the spoofed email and "prashant@gmail.com" as the spoofed sender email address.

```
sender = 'prashant@mail-server.in'
receivers = ['prashantchauhan25@gmail.com']

message = """From: Prashant Chauhan <prashant@gmail.com>
Subject: Meeting Reminder 12

Sir, reminder again..
Please find below the updated schedule for the meeting:
Date: 10 Nov 2020
Time: 10 am onwards
```

Figure 43: Part of python script used to send spoofed email

As seen in *Figure 44*, the spoofed email address, "prashant@gmail.com" was displayed to the recipient user as the "sender", while the domain "mail-server.in" was used for SPF check which successfully passed as it was already configured for our own IP address.

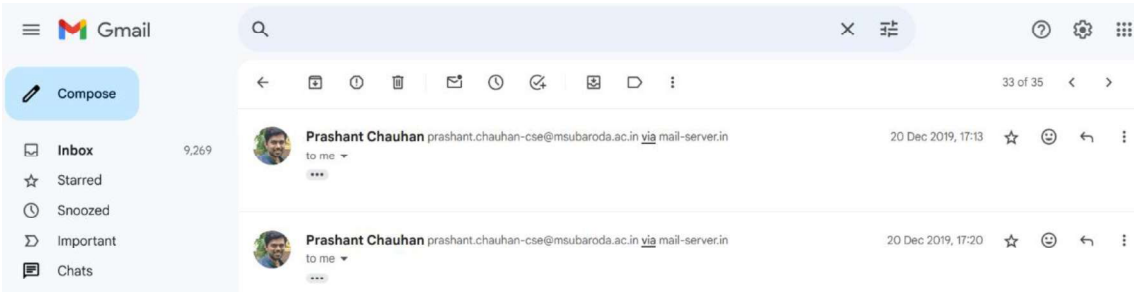


Figure 44: Spoofed email delivery in inbox after bypassing SPF

We can confirm from *Figure 45*, that our spoofed email successfully passed the SPF test as "mail-server.in" domain was used to check the SPF records.

Message ID	<20191220120129.F061F2A3968@mail.mail-server.in>
Created on:	20 December 2019 at 17:31 (Delivered after 63 seconds)
From:	Prashant Chauhan <prashant.chauhan-cse@msubaroda.ac.in>
To:	Prashant <prashantchauhan25@gmail.com>
Subject:	Hello Prashant
SPF:	PASS with IP 117.240.215.158 Learn more

Figure 45: Raw email summary for spoofed email after bypassing SPF

3.5.2 How to Bypass DKIM?

To bypass the DKIM, we added a signature encrypted by our own private key for the domain “mail-server.in”, along with the email. Once, the receiving server receives the email it tried to find the domain which sent the email, and it gets our original email server as that domain name.

Then, it checks our DNS entry for finding the public key of our domain and tries to decrypt the signature with the public key. As the signature gets decrypted successfully, it confirms that this email came from an authentic user and the DKIM test is passed successfully as seen in *Figure 47*. As seen in *Figure 48*, the domain “mail-server.in” was used for the DKIM test and the domain “msubaroda.ac.in” was used for DMARC test. But since the detected domain has not published DMARC policy in its DNS, DMARC was ignored and our spoofed email successfully makes its way to the inbox of the recipient user.

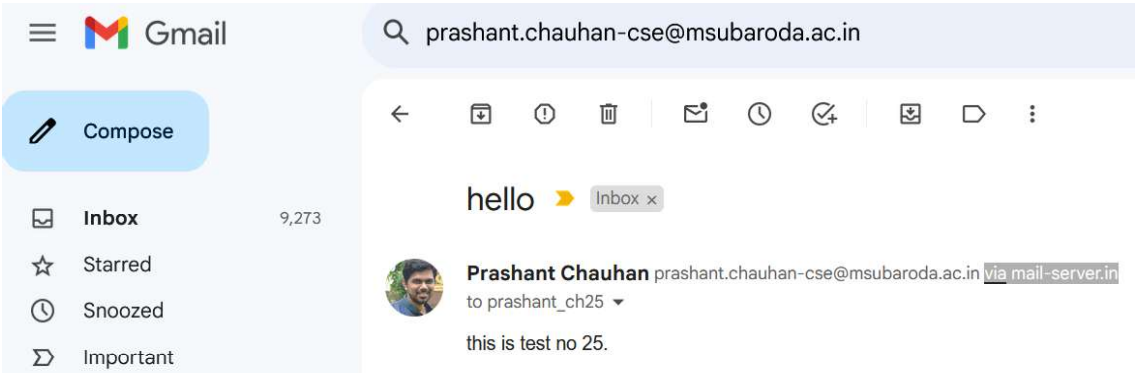


Figure 46: Spoofed email delivery in inbox after bypassing SPF and DKIM

Original message

Message ID	<5fb6e415.1c69fb81.b3556.895dSMTPIN_ADDED_MISSING@mx.google.com>
Created on:	20 November 2020 at 03:00 (Delivered after 4 seconds)
From:	MSU - <prashant.chauhan-cse@msubaroda.ac.in>
To:	prashantchauhan25@gmail.com
Subject:	A test message using python 503
SPF:	PASS with IP 117.240.215.158 Learn more
DKIM:	'PASS' with domain mail-server.in Learn more

Figure 47: Raw email summary for spoofed email after bypassing SPF and DKIM

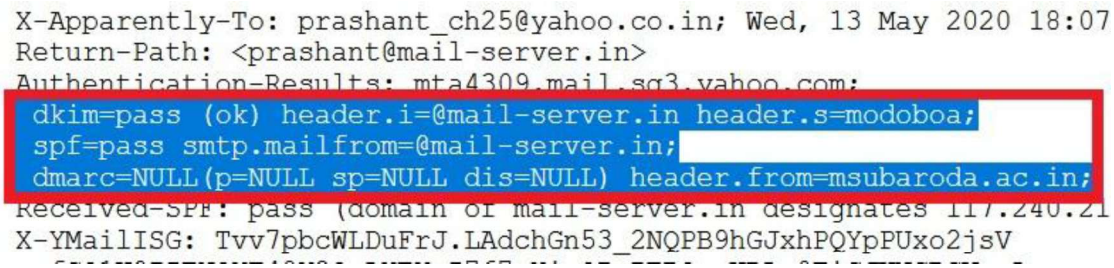


Figure 48: Raw email details for spoofed email after bypassing SPF and DKIM

3.5.3 How to Bypass DMARC?

To bypass the DMARC, we tried to insert multiple email addresses with the "mail-from" field in the email one of them being the spoofed email address and the other one email address of our own email server as shown in *Figure 49*. The spoofed email successfully delivered to yahoo mail server and Gmail server as seen in *Figure 50* and *Figure 51* respectively.

```
sender = 'prashant@mail-server.in'
receivers = ['prashantchauhan25@gmail.com']

message = """From: Prashant Chauhan <prashant@gmail.com>
<prashant@mail-server.in>
To: <prashantchauhan25@gmail.com>
Subject: Meeting Reminder 12
```

Figure 49: Part of python script used to send spoofed email with multiple email ids

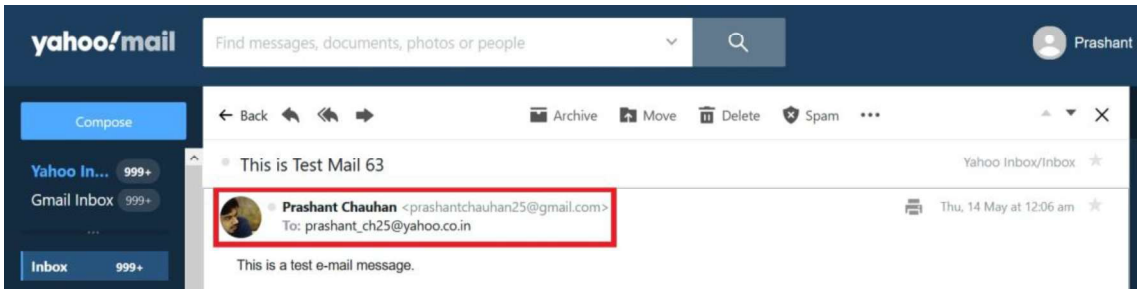


Figure 50: Spoofed email delivered in inbox of "yahoo.co.in" after bypassing SPF, DKIM and DMARC

As one of the email addresses in the "mail-from" field is the actual originating email address, the receiving email server performs DMARC test on our original email address resulting in passing of DMARC test.

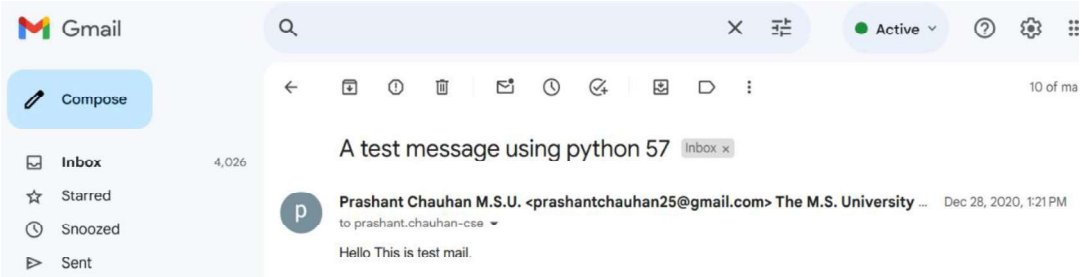


Figure 51: Spoofed email delivered in inbox of "gmail.com" after bypassing SPF, DKIM and DMARC

Knowing the fact that whatever is written in the "From" header of the email message is displayed to the recipient as the sender of the email, the name in the "From" field is purposefully written longer so that the recipient user is unable to view the second email address written after the spoofed email address as can be seen in *Figure 51*. Only if the users check the email details, they may find the second email address written in the "From" header as seen in *Figure 52*.

Original Message

Message ID	<5fb76ad3.1c69fb81.cada0.355f28122020132115@mail-server.in>
Created at:	Mon, Dec 28, 2020 at 1:21 PM (Delivered after 7 seconds)
From:	"Prashant Chauhan M.S.U. <prashantchauhan25@gmail.com> The M.S. University of Baroda" <prashant@mail-server.in>
To:	prashant.chauhan-cse@msubaroda.ac.in
Subject:	A test message using python 57
SPF:	PASS with IP 117.240.215.158 Learn more
DKIM:	'PASS' with domain mail-server.in Learn more
DMARC:	'PASS' Learn more

Figure 52: Raw email summary for spoofed email after bypassing SPF, DKIM and DMARC

```
Received: from mail-server.in (mail-server.in. [117.240.215.158])
        by mx.google.com with ESMTPS id s23si13542116pjs.153.2020.12.27.23.31.29
        for <prashantchauhan25@gmail.com>
        (version=TLS1_3 cipher=TLS_AES_256_GCM_SHA384 bits=256/256);
        Sun, 27 Dec 2020 23:31:30 -0800 (PST)
Received-SPF: pass (google.com: domain of prashant@mail-server.in designates 117.240.215.158 as permitted sender)
ip=117.240.215.158;
Authentication-Results: mx.google.com;
       dkim=pass header.i=@mail-server.in header.s=modoboa header.b=hpJMnLCy;
       spf=pass (google.com: domain of prashant@mail-server.in designates 117.240.215.158 as permitted sender)
smtp.mailfrom=prashant@mail-server.in;
       dmarc=pass (p=QUARANTINE sp=QUARANTINE dis=NONE) header.from=mail-server.in
Received: from localhost (localhost [127.0.0.1]) by mail-server.in (Postfix) with ESMTP id 254BF140314 for
```

Figure 53: Raw email details for spoofed email after bypassing SPF, DKIM and DMARC

Since one of the email addresses in the "mail-from" field is same as the one used for checking SPF and DKIM, identifier alignment is also passed even in case of spoofed email resulting in passing of DMARC test as can be seen in *Figure 52* and Figure 53.

3.6 Comparison of Genuine and Spoofed Emails

3.6.1 Authentic Genuine Email

The genuine email sent from the domain “msubaroda.ac.in”, using the official Gmail server, passed only the DKIM test as seen in *Figure 54*. This is because at this moment the domain “msubaroda.ac.in” has not published any of the anti-spoofing protocols in their DNS. But since “msubaroda.ac.in” is using Gmail services for email, the DKIM is setup by the Gmail server itself, the DKIM test was passed by the authentic email sent from the genuine email account as seen in *Figure 54*.

Message ID	<CAHSz8NN5LjVSvTz+AdR3BZB4F1HsiyX_M40Dp7XYJ10wNBSRFw@mail.gmail.com>
Created on:	28 December 2020 at 13:51 (Delivered after 11 seconds)
From:	Prashant Chauhan <prashant.chauhan-cse@msubaroda.ac.in>
To:	prashantchauhan25@gmail.com
Subject:	Test Mail 62
SPF:	NEUTRAL with IP 209.85.220.41 Learn more
DKIM:	'PASS' with domain msubaroda-ac-in.20150623.gappssmtp.com Learn more

Figure 54: Raw email summary for genuine email sent from domain “msubaroda.ac.in”

```
 by atlas203.free.mail.sg3.yahoo.com with SMTPs; Mon, 29 Jun 2020 05:07:47 +000
X-Originating-Ip: [209.85.208.53]
Received-SPF: none (domain of msubaroda.ac.in does not designate permitted send
Authentication-Results: atlas203.free.mail.sg3.yahoo.com;
 dkim=pass header.i=@msubaroda-ac-in.20150623.gappssmtp.com header.s=20150623;
 spf=none smtp.mailfrom=msubaroda.ac.in;
 dmarc=unknown
X-Apparently-To: prashant_ch25@yahoo.co.in; Mon, 29 Jun 2020 05:07:48 +0000
X-YMailISG: f4hEagoWLDuOVSAnXuGuF4Eu0eUpyPebtCzTywRGWNHkU4O_
 uXDhZqvLPZlYsA2kGi_bQpBTaYcUDmd6Kr674KvyMkVK59lS08NJkgP_mks0
 tCvkT46gdr9BoFXq24Mk_JPLgKT.d_GqWVArNSYMbbmhDBX21Qa8VM70y15i
 Nt1A80u0OuPT9WqIelAYzKq_OWiA5peDVeTNBxibpwISOY375Myw_vpRZ3we
```

Figure 55: Raw email details for genuine email sent from domain “msubaroda.ac.in”

From *Figure 55*, we can see that “spf=none” and “dmarc=unknown” means no record for SPF and DMARC were found for the domain “msubaroda.ac.in” in its DNS. So even in case of genuine email from “msubaroda.ac.in”, SPF and DMARC were not passed and only DKIM was passed.

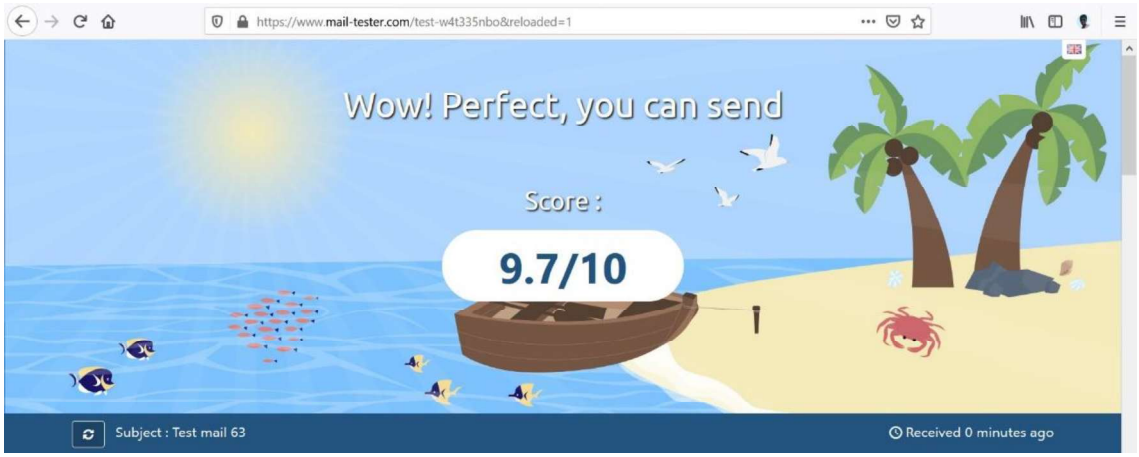


Figure 56: Email delivery report for spoofed email from domain "gmail.com"

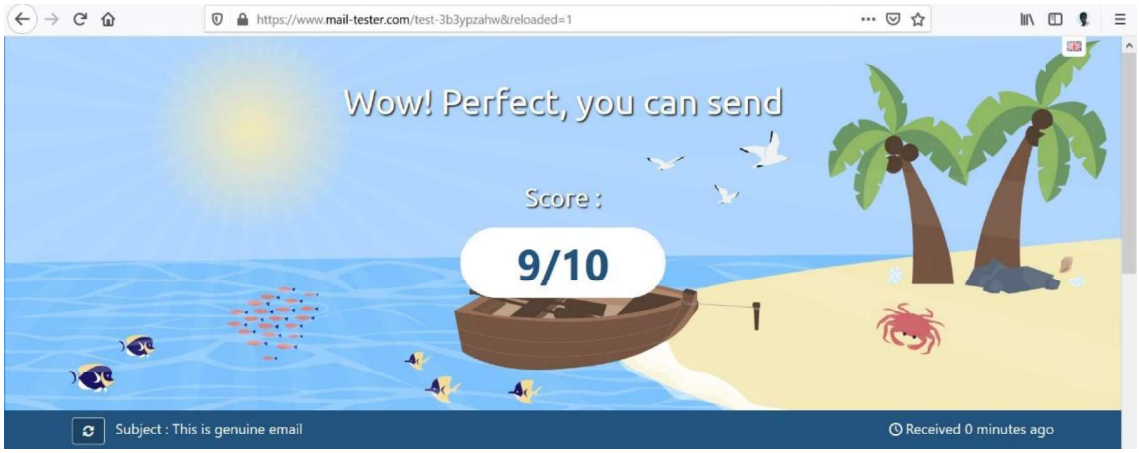


Figure 57: Email delivery report for genuine email from domain "msubaroda.ac.in"

We tested email delivery score using online website "https://www.mail-tester.com/". A score of 9.7 was found for the genuine email sent from the official Gmail account as seen in *Figure 56* and a score of 9 was found for the genuine email sent from the webmail of "msubaroda.ac.in" as seen in *Figure 57*.

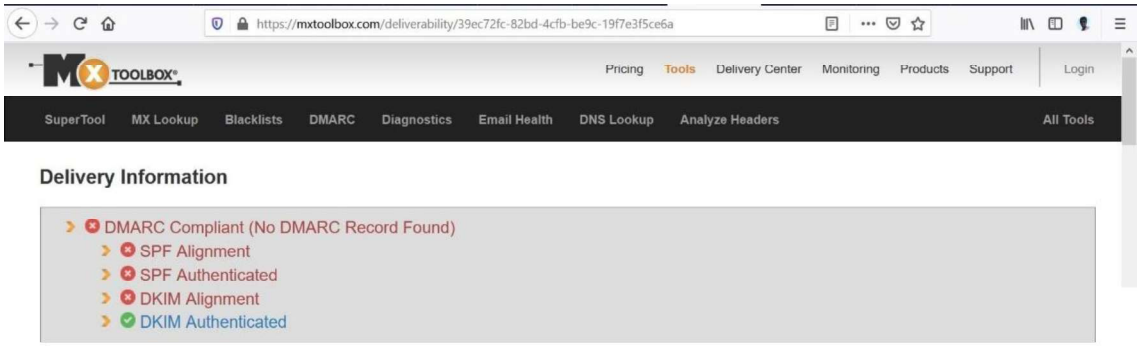


Figure 58: Email delivery information for genuine email from domain "msubaroda.ac.in"

We also tested the genuine email sent from the domain “msubaroda.ac.in” on another online portal *MxToolBox* to further check the authenticity of the email sent. As seen in *Figure 58*, we found that the authentic email sent from the official webmail of “msubaroda.ac.in” passed only the DKIM authentication while the SPF alignment, SPF authentication and DKIM alignment tests were failed for the genuine email.

3.6.2 Spoofed Fake Email

Once all the tests were performed for the authentic genuine emails, we begin tests on our fake spoofed emails. And we were successfully able to spoof email addresses for almost all kinds of domains. Whether a domain had published all anti-spoofing protocols or none of them, we were successful in bypassing all the three anti-spoofing protocols SPF, DKIM and DMARC.

As seen in *Figure 59*, our spoofed email passed SPF, DKIM and DMARC tests for the email address from the domain “msubaroda.ac.in”. Also, we were able to pass SPF, DKIM and DMARC tests for the emails from the domain “gmail.com” as seen in *Figure 60*.

Original message

Message ID	<5fb76ad3.1c69fb81.cada0.355f28122020130122@mail-server.in>
Created on:	28 December 2020 at 13:01 (Delivered after 8 seconds)
From:	"Prashant Chauhan <prashant.chauhan-cse@msubaroda.ac.in> The M.S. University of Baroda" <prashant@mail-server.in>
To:	prashantchauhan25@gmail.com
Subject:	A test message using python 52
SPF:	PASS with IP 117.240.215.158 Learn more
DKIM:	'PASS' with domain mail-server.in Learn more
DMARC:	'PASS' Learn more

Figure 59: Raw email summary for spoofed email from domain “msubaroda.ac.in”

Like the genuine email, we also tested email delivery score for the spoofed emails sent through our email server using the online website https://www.mail-tester.com/. A score of 9.9 was found for the spoofed email sent from our email server in the name of “msubaroda.ac.in” as seen in *Figure 61* and a score of 7.6 was found for the spoofed email sent from our email server from the domain “gmail.com” as seen in *Figure 62*.

Original Message

Message ID	<5fb76ad3.1c69fb81.cada0.355f28122020132115@mail-server.in>
Created at:	Mon, Dec 28, 2020 at 1:21 PM (Delivered after 7 seconds)
From:	"Prashant Chauhan M.S.U. <prashantchauhan25@gmail.com> The M.S. University of Baroda" <prashant@mail-server.in>
To:	prashant.chauhan-cse@msubaroda.ac.in
Subject:	A test message using python 57
SPF:	PASS with IP 117.240.215.158 Learn more
DKIM:	'PASS' with domain mail-server.in Learn more
DMARC:	'PASS' Learn more

Figure 60: Raw email summary for spoofed email from domain "gmail.com"

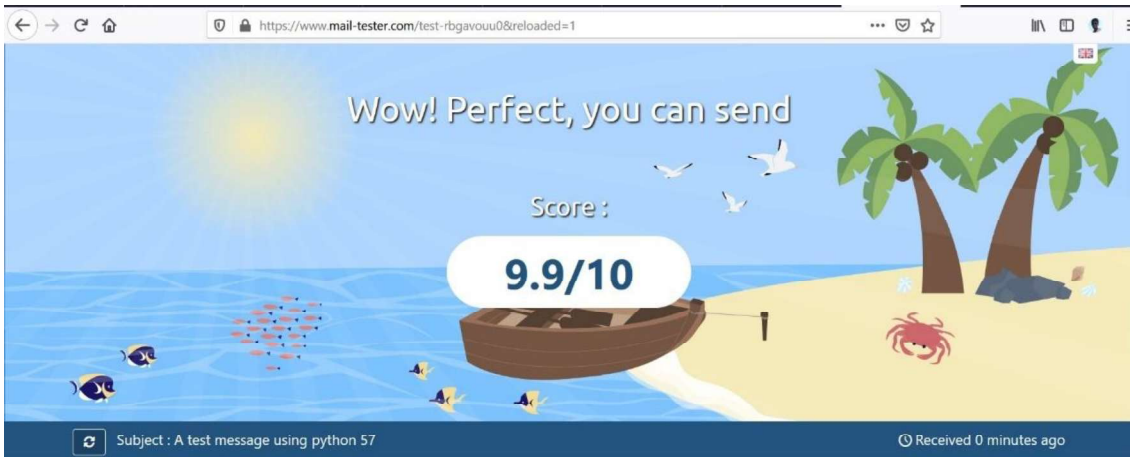


Figure 61: Email delivery report for spoofed email from domain "msubaroda.ac.in"

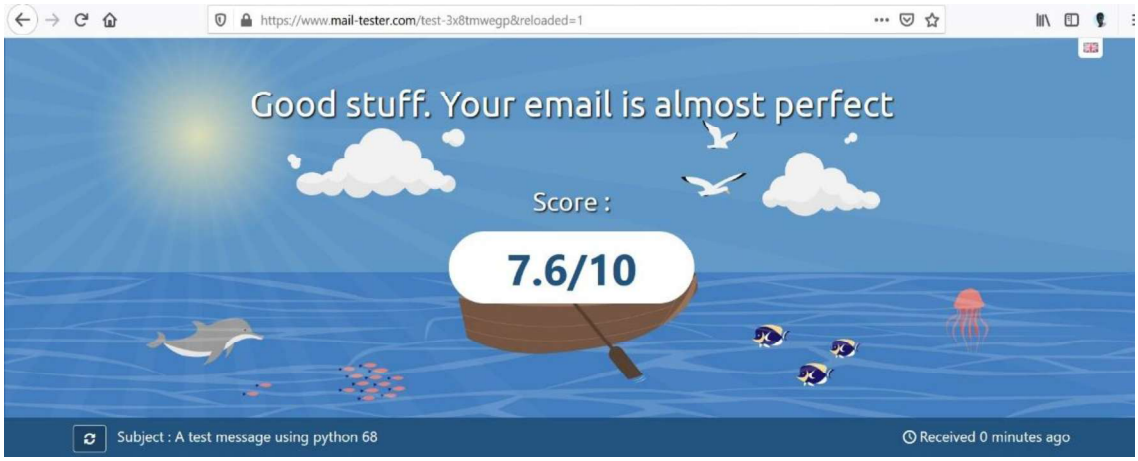


Figure 62: Email delivery report for spoofed email from domain "gmail.com"

We also tested the spoofed email sent from the domain “msubaroda.ac.in” on the online portal *MxToolBox* to further check the authenticity of the sent email. As seen in *Figure 63*, we found that the spoofed email sent from our email server for the domain “msubaroda.ac.in” passed all the tests namely SPF alignment, SPF authentication and DKIM alignment and DKIM authentication which was not a case even in case of genuine email.

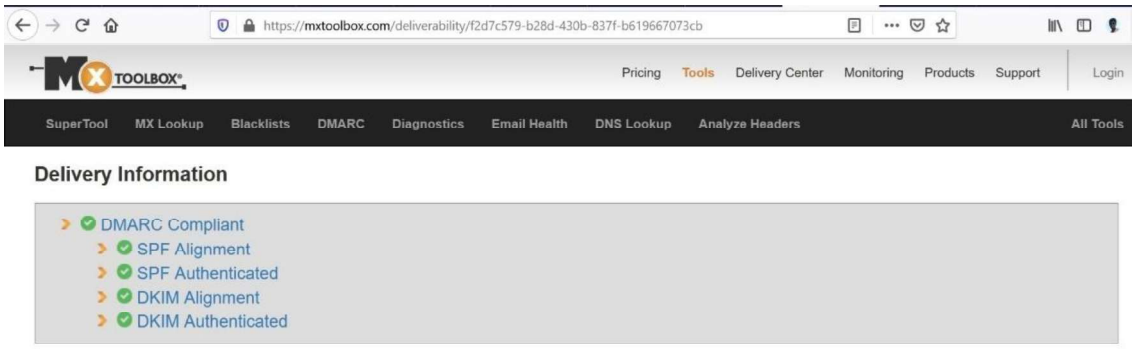


Figure 63: Email delivery information for spoofed email from domain “msubaroda.ac.in”

By comparing the results as seen in *Figure 54* and Figure 59, we can clearly say that a spoofed email from the domain “msubaroda.ac.in” sent by our email server passes all the three anti-spoofing protocols while the genuine email passes only DKIM test. So we can say that for the recipient email server, our spoofed email looks even more authentic than the actual genuine email.

Also by comparing the results as seen in *Figure 57* and Figure 61, we find that email delivery score of spoofed email for the domain “msubaroda.ac.in” is 9.9 which is more than the original genuine email i.e. 9 only. Though the email delivery score for the domain ”gmail.com” for spoofed email is only 7.6 as compared to a score 9.7 for genuine email, we still able to successfully deliver our spoofed email in inbox of the recipient user passing all the three anti-spoofing protocols.

As seen in *Figure 50*, even for the spoofed email the profile picture of the spoofed sender is displayed in the inbox of the recipient user which further gives confidence to the recipient of the authenticity of the email, which should not have happened.

3.7 Conclusion and Summary

In this chapter, we described the procedure for the setup of our own email server including installation of needed packages and modification to various configuration files. We then discussed the ways to bypass the three anti-spoofing protocols currently in use i.e. SPF, DKIM and DMARC. By various modifications to the email server, we were successfully able to send spoofed email to various other recipient email servers. We also listed various tests performed on the genuine as well as spoofed emails to find out whether our spoofed emails actually work or not.

From the comparison of various tests on authentic as well as spoofed email, we concluded that it is still easily possible to spoof an email, to successfully deliver spoofed email to the inbox of the recipient user and to successfully bypass all the three anti-spoofing protocols SPF, DKIM and DMARC. We also found one issue in implementation of DMARC policy that the Quarantine/ Reject policy is not enabled by majority of the email servers like Gmail. This is done by the email servers so as not to affect their deliverability of emails in case of wrong test.

With our tests, we also found that in many cases our spoofed email was found to be more authentic even than the original genuine email. The attackers and spoofers take advantage of this facts and try to victimize various users for their own benefits.