International Journal of Computer Applications (0975 – 8887)
Volume 174 – No. 15, January 2021

Distributed Firewalls Mechanism for the Resolution of Packets Forwarding Problems in Computer Networks using RSA-CRT Technique

Bukola Fatimah Balogun
Department of Computer Science
Kwara State University, Malete
Nigeria

ABSTRACT

A firewall is a security barrier that is installed between a private network and the public networks (or Internet) at the point of entry to filter all incoming and outgoing packets across it. Firewalls have the responsibility of reviewing any incoming or outgoing packets and reach the decisions about accepting or discarding them. On the basis of contradictory laws, firewalls make decision for each packet in resolving conflicts as the first rule matching the packet. The conventional procedure for constructing firewalls involves a series of rules though inefficient due to the problem of continuity (difficulty in correctly ordering the rules); the completeness problem (difficulty in ensuring thorough scrutiny of all traffic types); and the compactness problem (difficult to keep the number of rules small) because some rules may be redundant and some rules may be combined into one rule. Again, traditional firewalls rely on the concepts of restricted network topology for its operations, and regulation of points of network entry. In particular, firewalls depend on the presumption that node on one side of the entry point of the firewall is to be trusted, and that node on the other side is an enemy. However, this presumption is possible theoretically because of recent advances and access to the Internet. This paper proposes a resolution of packets forwarding problems on computer networks based on distributed firewalls mechanism. The Iptables serve as a policy language, and system management tools. The outcomes showed considerable specification and distribution of resolution policy on Linux operating systems.

General Terms

Network Security, Distributed Firewall System

Keywords

Network, Firewall, Iptables, Security, Policy, RSA, CRT

1. INTRODUCTION

Computers and Networking are inseparable components. Several sensitive transactions occur every second, and today computers are often used for transactions rather than data processing. This had to provide the corrective steps to eradicate and prevent viruses, prevent hacking of data, and to provide authenticated data transfer [7]. A firewall is a device or series of instruments designed to accept or reject network transmissions based on a set of rules and regulations that are often used to protect networks from unauthorized access while allowing legitimate communications to pass through or during sensitive data transmission, which is situated between two networks that filter traffic between them employing some security policies [13]. A firewall is an efficient way to shield a local device or network systems from network-based intrusion attacks while also providing access to the outside world through wide area networks and the internet. A firewall is a series of modules, interposed between two networks, which filters traffic between them according to certain security policies [9]. Distributed firewalls defend the network by securing sensitive endpoints of the network, exactly where hackers want to get in. It filters internet and internal network traffic because the most disruptive and expensive hacking attacks often originate from inside the company.
Distributed firewall was introduced due to shortfalls in traditional firewall such as over-reliance on the topology of the network. The distributed firewall was first conceived by Sir Steve M. Bellovin in the 2000, who wrote and developed the distributed firewall by means of a keynote trust management for authentication and distribution of security policies in a central management system [11]. The foremost distributed firewall was unable to timely update the users' security policies, limited connectivity for filtering IP-addresses, and the overly consideration for controlling TCP connections.

The aim of this paper is to resolve packets forwarding problems on computer networks by means of the distributed firewalls mechanism.

2. LITERATURE REVIEW

2.1 Overview of the Concepts of Firewall

The lack of total security solutions serves as motivation for a network with multilayers to be constructed contemporarily to create an obstacle for violating certain activities [1]. The information protection act for a network seeks to protect data stored on computers such as servers as depicted in Figure 1.

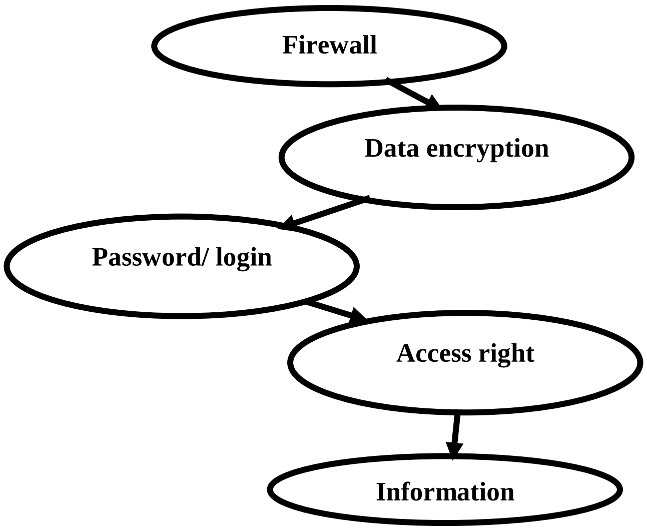


Figure 1: The structure of network security layers [12].

Network servers have several security layers in place to enhance data and information protection capabilities [1]. From Figure 1, the innermost protective layer is access right. This layer regulates network (information) resources and rights (what users can do with those resources). This control applies to files, directories, and partitions. The next layer restricts account access including usernames and passwords (Password/Login) this is a frequently used technique of protection due to its simplicity, the fact that it is economical and highly effective.

This is the full duty of the administrator to monitor, handle, and carry out other users' activities. Data encryption methods are utilized in the third layer using certain algorithms to evade hackers' decryption operation even without an encryption key causing data losses. The outermost layer prevents the violation, intrusions, filters unwanted outgoing or incoming information packets [12].

2.1.1 Structure of a Firewall

A firewall is either a host or hardware-based. A hardware-based firewall is usually customized network boxes, such as routers or switches having special software and hardware installed. In practice, this kind of firewall is expensive, complex, and difficult to configure. The distinguishing feature of a hardware-based firewall and a host-based firewall is ease of deployment by individuals and small organizations. A network-based firewall is considered to be a piece of software running on an individual's personal computer (PC), notebook, or server. It is configured to allow or restrict data transferred on a network based on a set of rules. A firewall is used to secure a network from intrusions and concurrently permit legitimate data across it [5]. Again, a firewall should have at least two network traffics, one for public network activities such as the Internet, and another one for a private network [4]. At this point in time, it acts as a gate controlling outgoing/incoming data streams of an Intranet as shown in Figure 2.

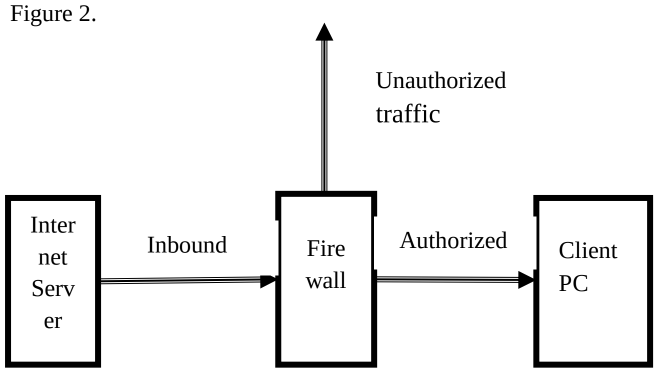


Figure 2: A typical structure of a firewall [3].

2.1.2 Functions of a Firewall

Traditionally, the workings of a firewall rely on the characteristics and types. The basic technical functions of firewalls include [4]:

1. Protect resources.
2. Act as an intermediary.
3. Record and report on events.
4. Authentication access.
5. Manage and control network traffic.

2.1.3 Distributed-Firewall

A host of resident firewalls make up a distributed firewall when centrally designed and controlled [10]. The security policy is still centrally established in this architecture but policy implementation takes place at any endpoint (hosts, routers, etc.). The centralized policy sets out what connectivity may or may not be allowed. That policy will then be distributed to all endpoints, where it is enforced.

2.1.4 Component of the distributed firewall

1. **Security policy language**: used to create policies for each firewall. These policies are the collections of rules, which guide the firewall for evaluating the network traffic and also defines which inbound and outbound connections are allowed or rejected [8].
2. **Policy distribution scheme:** The scheme is used to allow policy control from a central point. This policy is consulted before processing the incoming or outgoing messages. It should guarantee the integrity of the policy during transmission. It can be either directly pushed to end systems or pulled when necessary with the implementation [8].
3. **IPSec:** it offers encryption at the network level that is used to protect network traffic and policy transmission. This also has a more critical role to provide a way to cryptographically verify the information sender. Senders can then have a specific check of their certificate. The objective is to develop and evaluate protocols that resolve the control of adversaries and apply to different aspects of information security, such as data confidentiality, data integrity, authentication, and non-repudiation [8].

2.2 Related Works

Authors in [14] propounded a strategy for checking whether a firewall responds correctly with respect to a security policy given in a high-level declarative language. The strategy is applied in satisfiability solver modulo theories (SMT). These works are limited to the problem of disunity avoidance and do not consider verifying whether a firewall responds correctly with respect to a given security policy.

[6] proposed a system that uses both a network and a firewall working together to provide security in IPv6 systems. This approach is targeted for IPv6 packets that contain the Encapsulated Security Payload, or ESP extension header. Implementation of a dual firewall approach between host and network systems coordinated by a central administrative server. Host firewalls inspect decrypted content and pass information back to the network firewall and administrative server. Distributed firewall approaches are an interesting approach considering the new security threats presented by IPv6 and especially encrypted packets.

[15] proposed a self-adaptive distributed firewall, using the model as a means to structure the various tasks involved in network security management, using a vulnerability analysis system to detect vulnerable hosts on the network infrastructure. There are other obvious drawbacks to the use of NVTs in this system. Every NVT is modified according to the NVT Feed SADF is only able to react to known vulnerabilities that have been published to NVT Feed.

[16] suggested a framework and infrastructure in a virtual machine making use of distributed firewall architecture. The system in one embodiment consists of receiving a packet of a gateway computer firewall interfacing a local area network (LAN) and an external network at an input-output (IO) module; Where the firewall contains at least one IO module performing the IO functionality of the firewall and at least one security processing module performing the security functionality of the firewall, and each of the IO module and at

least one security processing module operates on a virtual machine and is controlled by a firewall controller.

[8] proposed a system using a policy definition language (known as the Key Note) and using IPsec for secure traffic between hosts. The system can allow or reject traffic intended for a particular system depending on the policy to be followed. Remote end-user devices should be protected to prevent them from being used as entry points in the enterprise network; the limitations of the system are inability to update the user in timely manner.
In particular, the most relevant work is that of [8]. The approach is based on a three-layer system: a high-level policy language, an intermediate-level language (keynote) used by their mechanism, and the actual mechanism enforcing the policy. In the proposed system, a RSA-CRT algorithm was used for the cryptography and authentication, an Iptable was used for its policy formation and enforcement.

3. RESEARCH METHODOLOGY

3.1 Proposed System

This system uses an asymmetric cryptography algorithm known as Rivest, Shamir and Adleman (RSA) and Residue Number System (RNS) to encrypt and decrypt information to provide security services such as Privacy, Data Integrity, and Message Authentication. Also, iptables which is a command-line firewall utility that uses policy chains to allow or block traffic is employed. When a connection tries to establish itself on the system, iptables look for a rule in its list to match it to. If it doesn’t find one, it resorts to the default action.

The Iptables is written in python programming language and the operating system used is Linux (Ubuntu). Confidentiality of information is achieved using the new cryptographic algorithm for the encryption and decryption of information. And since it makes use of two keys which is, the public and private key, only authorized recipients and sender have access to the information.

The implementation of the proposed system includes:

1. A policy language that states what sort of connection is permitted or prohibited, written with the iptables
2. A system to update the users on changes on the rules on the iptables (sender-client)
3. IPSEC, the network-level encryption mechanism for Internet Protocols (TCP, UDP, etc.).

The basic idea is simple. A compiler translates the policy language into some internal format. The system management software distributes this policy file to all hosts that are protected by the firewall. And incoming packets are accepted or rejected by each **inside** host, according to both the policy and the cryptographically-verified identity of each sender

3.2 System Design

The system design will be divided into two forms including:

1. The encryption part (RSA) that involves the communication between the client and server
2. On a successful connection, the credentials are provided to the client

3.2.1. RSA and Chinese Remainder Theorem

The security of the RSA is based on the difficulty of factoring large integers. The encryption and decryption process of the RSA algorithm requires modular exponentiation. This chapter will not go in-depth about how RSA works since we used the python basic authentication form.

The decryption must be faster in RSA-CRT than the decryption of the original RSA, the CRT enables the RSA algorithm to be implemented efficiently.

Given the input, m, lift it to the e-th (or d-th power modulo p and modulo q). The intermediate results are then combined with some constant predefined by adding and multiplying to determine the final result, the execution time is less than four times because the modular exponentiation is performed on half the size of n. The RSA decryption complexity M = Cd mod n relies directly on the size of d and n.

The exponent decryption d determines the number of multiplications of the modular exponentiation required, and the modulus n determines the size of the intermediate result, thus reducing the size of both d and n is considered an important advantage in the Chinese Remainder Theorem (CRT). Where the factors of the modulus N (P and Q) are supposed to be known. By CRT, the computation of M =CD (mod N) can be partitioned into two parts:

MP =CPDP (mod P)
Mq =CqDq (mod q)
Where CP =C (mod P)
DP = D (mod P-1)
Cq =C (mod q)
Dq = D (mod q-1)
Then using Chinese Remainder Theorem,
And find a solution M=MP (mod p) = Cd (mod p)
M=Mq (mod q) = Cd (mod q).
This reduces the time of computation since DP, DQ <D, and CP, CQ < C. In fact, their size is almost half the original size. There are several ways that this can be obtained from the original plaintext, M using the CRT. The comparison of the features of RSA and RSA-CRT is presented in Table 1.

Table 1. A comparison of RSA and RSA-CRT

RSA	RSA using CRT
Process very slow	Process is faster
It has less security	More secured security
Using encryption, decryption the required time is more	Using encryption, decryption the required time is less
One public and a private key is used	One public and a private key is used

3.3. System Requirements

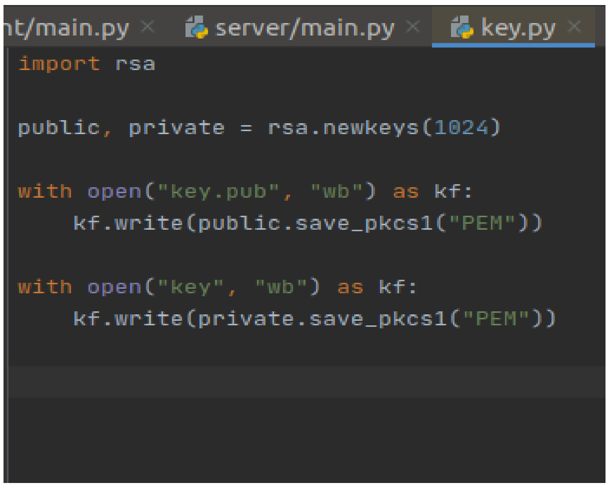
These are the requirements needed for the user to be able to operate on this proposed system;

1. Operating System: Linux operating system
2. Processor RAM: minimum of 4GB
3. Processor Speed: minimum of 1.8GHz
4. Language: Python programming language and any Python IDE

4. RESULTS

4.1 System Functionality

The proposed system implementation of RNS-based encryption and decryption keys is presented in Figure 2.



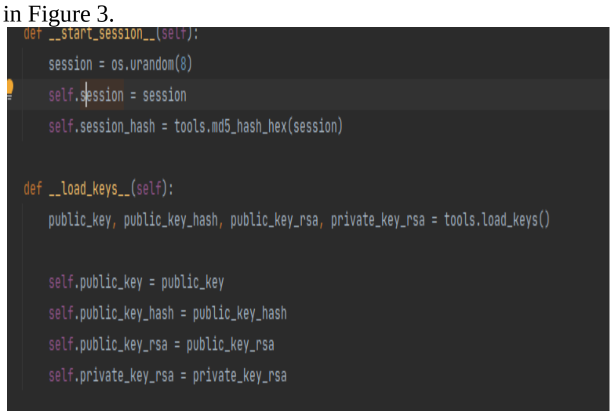
```python
import rsa

public, private = rsa.newkeys(1024)

with open("key.pub", "wb") as kf:
    kf.write(public.save_pkcs1("PEM"))

with open("key", "wb") as kf:
    kf.write(private.save_pkcs1("PEM"))
```

Figure 2: Generating your public and private keys

From Figure 2, the stage involved in generating private and public keys, but, only the public key will be sent over the Internet to the admin/server after generation of public and private keys through emails or any form**.**

4.2. Generation of RSA key pairs and hashing

This stage involves the generation of RSA key pairs which is, the private and public key and hashing it and this would be done using the public key and it is done at both ends as shown in Figure 3.

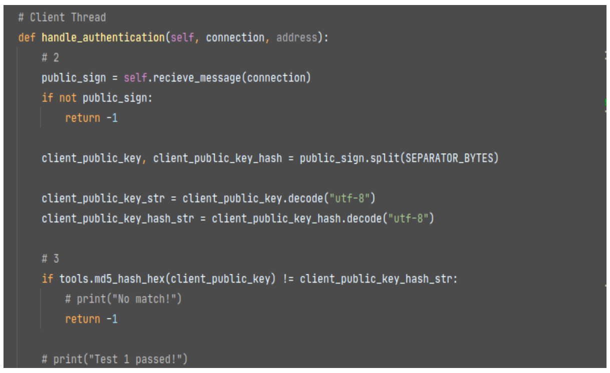


```python
def __start_session__(self):
    session = os.urandom(8)
    self.session = session
    self.session_hash = tools.md5_hash_hex(session)

def __load_keys__(self):
    public_key, public_key_hash, public_key_rsa, private_key_rsa = tools.load_keys()

    self.public_key = public_key
    self.public_key_hash = public_key_hash
    self.public_key_rsa = public_key_rsa
    self.private_key_rsa = private_key_rsa
```

Figure 3: Generation of RSA key pairs and hashing

In Figure 3, suppose client A uses the public key of the server to send the admin an encrypted message, in that message he/she can claim to be Client A, node might even be an intruder, but the admin/server has no way of verifying if it is the right person since everyone can see her public key, in order for the admin to verify the origin of the message, RSA can be used to sign a message as depicted in Figure 4.

.



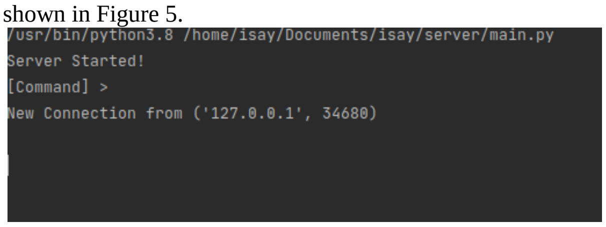
```python
# Client Thread
def handle_authentication(self, connection, address):
    # 2
    public_sign = self.recieve_message(connection)
    if not public_sign:
        return -1

    client_public_key, client_public_key_hash = public_sign.split(SEPARATOR_BYTES)

    client_public_key_str = client_public_key.decode("utf-8")
    client_public_key_hash_str = client_public_key_hash.decode("utf-8")

    # 3
    if tools.md5_hash_hex(client_public_key) != client_public_key_hash_str:
        # print("No match!")
        return -1

    # print("Test 1 passed!")
```

Figure 4: Hashed message and signature.

In Figure 4, suppose that client A wishes to send a message to the server, node can use its private key to do so, by producing a hash value of this signed message and attaches it as a "signature" to the message. When the server/admin receives the message, and uses the same hash algorithm in conjunction with client A public key if the hash value corresponds with that of client A hash value then he knows it is from the right person.

4.3 Connection to the server

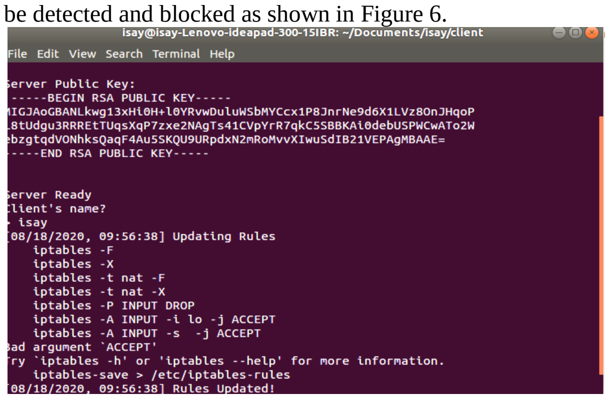
The server is up and running, then it detects a connection from a localhost:127.0.0.1 on port:34680 of the client having the name "isay" connects with the public key of the server, and by default, all connections/security policies passed are denied as shown in Figure 5.



```
/usr/bin/python3.8 /home/isay/Documents/isay/server/main.py
Server Started!
[Command] >
New Connection from ('127.0.0.1', 34680)
```

Figure 5: Connection to server by clients' side

Whenever, an intruder uses the public key of the server and tries to connect or impersonate an individual the intruder will be detected and blocked as shown in Figure 6.

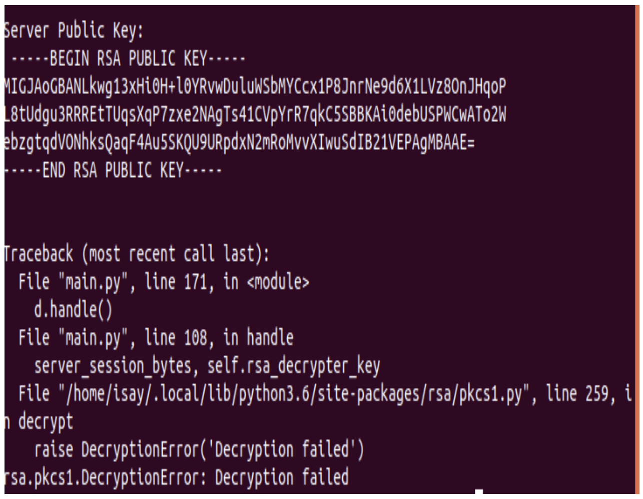


isay@isay-Lenovo-ideapad-300-15IBR: ~/Documents/isay/client
File Edit View Search Terminal Help

```
Server Public Key:
-----BEGIN RSA PUBLIC KEY-----
MIGJAoGBANLkwg13xHi0H+l0YRvwDuluWSbMYCcx1P8JnrNe9d6X1LVz8OnJHqoP
L8tUdgu3RRREtTUqsXqP7zxe2NAgTs41CVpYrR7qkC5SBBKAi0debUSPWCwATo2W
ebzgtqdVONhksQaqF4Au5SKQU9URpdxN2mRoMvvXIwuSdIB21VEPAgMBAAE=
-----END RSA PUBLIC KEY-----


Server Ready
Client's name?
> isay
[08/18/2020, 09:56:38] Updating Rules
    iptables -F
    iptables -X
    iptables -t nat -F
    iptables -t nat -X
    iptables -P INPUT DROP
    iptables -A INPUT -i lo -j ACCEPT
    iptables -A INPUT -s  -j ACCEPT
Bad argument `ACCEPT'
Try `iptables -h' or 'iptables --help' for more information.
    iptables-save > /etc/iptables-rules
[08/18/2020, 09:56:38] Rules Updated!
```

Figure 6: connection to the server

In this case, an intruder trying to gain access will not be able to decrypt the messages, the intruder below uses the public key of the server and try to decrypt the message sent, but the keys are unmatched, so the intruder cannot decrypt the message, the server on seeing the above message immediately blocks and drops the connection as illustrated in Figure 7

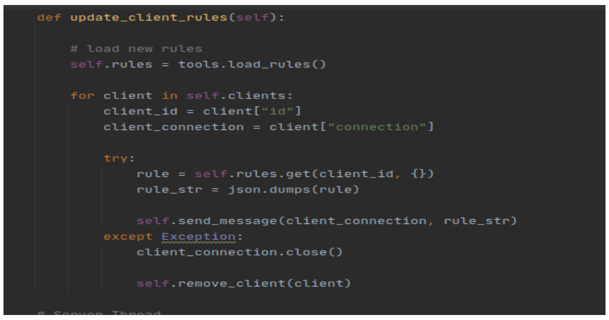


```
Server Public Key:
-----BEGIN RSA PUBLIC KEY-----
MIGJAoGBANLkwg13xHi0H+l0YRvwDuluWSbMYCcx1P8JnrNe9d6X1LVz80nJHqoP
L8tUdgu3RRREtTUqsXqP7zxe2NAgTs41CVpYrR7qkC5SBBKAi0debUSPWCwATo2W
ebzgtqdVONhksQaqF4Au5SKQU9URpdxN2mRoMvvXIwuSdIB21VEPAgMBAAE=
-----END RSA PUBLIC KEY-----


Traceback (most recent call last):
  File "main.py", line 171, in <module>
    d.handle()
  File "main.py", line 108, in handle
    server_session_bytes, self.rsa_decrypter_key
  File "/home/isay/.local/lib/python3.6/site-packages/rsa/pkcs1.py", line 259, i
n decrypt
    raise DecryptionError('Decryption failed')
rsa.pkcs1.DecryptionError: Decryption failed
```

Figure 7: Client authentication to server

4.4 Distribution of policies

In distributed firewall security policies can be distributed in two ways; the pull and push technique. In the proposed system used here, the push technique is used. The technique is used when policies are updated at the central management, which is the major advantage of a distributed firewall. On default, every user is denied access to the network and all packets are dropped, whenever a security policy is updated, the system admin must update the host immediately. The push technique ensures that all host always have the updated policies at all times, the policy language defines the connections. The server updates the clients as shown in Figure 8.

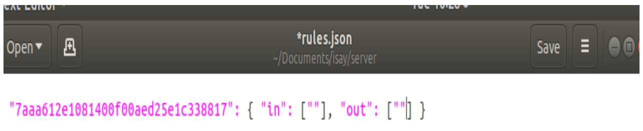


```
def update_client_rules(self):

    # load new rules
    self.rules = tools.load_rules()

    for client in self.clients:
        client_id = client["id"]
        client_connection = client["connection"]

        try:
            rule = self.rules.get(client_id, {})
            rule_str = json.dumps(rule)

            self.send_message(client_connection, rule_str)
        except Exception:
            client_connection.close()

            self.remove_client(client)
```

Figure 8: Distribution of policies

The admin/server knows who is connected, Suppose client A wishes to send a message to the server, the client uses his/her own private key to do so, the client produces a hash value of this signed message and attaches it as a "signature" to the message .when the server/admin receives the message he uses the same hash algorithm in conjunction with client A public key if the hash value corresponds with that of client A hash value then he knows it is from the person, if both hashes correspond then he knows it is "isay" that is trying to connect or maybe "client B".

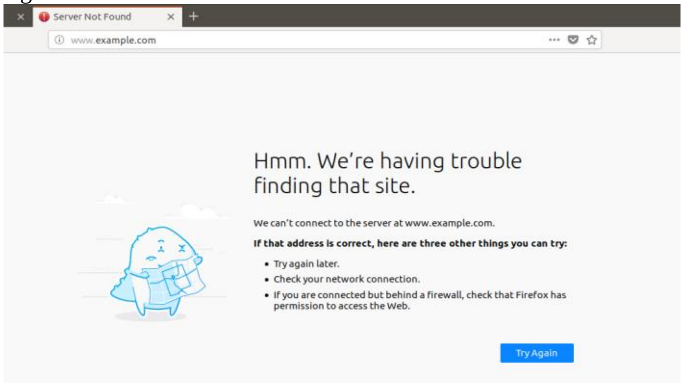
The server hashes the public key of the client and then sends the updated security policies to the expected user, on the server side the rules are stated in a text document as shown in Figure 9



```
"7aaa612e1081400f00aed25e1c338817": { "in": [""], "out": [""] }
```

Figure 9: The rules.json.

The first part of this array list as shown in Figure 10, is the hash of the client public key, and then "IN" and "OUT" are the inbound and outbound connections that are either allowed or denied. On default, as stated earlier all connections are blocked/denied and cannot access the Internet as shown in Figure 10.



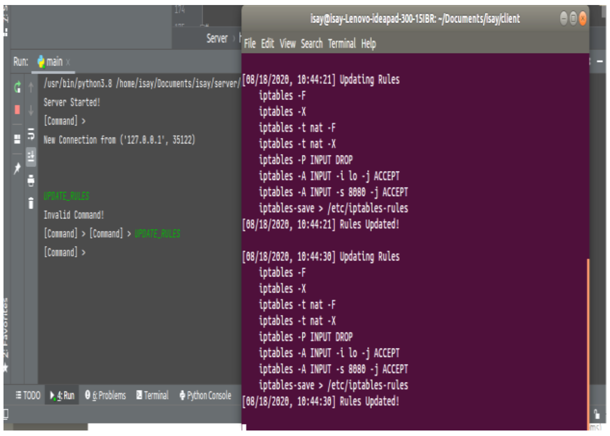
Hmm. We're having trouble finding that site.

We can't connect to the server at www.example.com.

If that address is correct, here are three other things you can try:

- Try again later.
- Check your network connection.
- If you are connected but behind a firewall, check that Firefox has permission to access the Web.

Figure 10: The website "example.com" not working



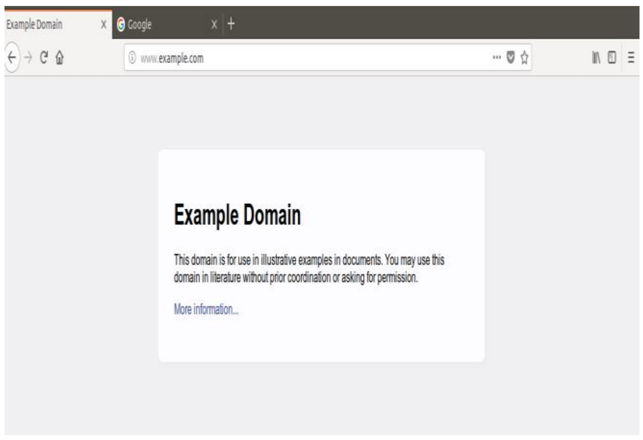
Server
Run: main
isay@isay-Lenovo-ideapad-300-15IBR: ~/Documents/isay/client
File Edit View Search Terminal Help
TODO
4: Run
6: Problems
Terminal
Python Console

```
/usr/bin/python3.8 /home/isay/Documents/isay/server/
Server Started!
[Command] >
New Connection from ('127.0.0.1', 35122)


UPDATE_RULES
Invalid Command!
[Command] > [Command] > UPDATE_RULES
[Command] >
```

```
[08/18/2020, 10:44:21] Updating Rules
    iptables -F
    iptables -X
    iptables -t nat -F
    iptables -t nat -X
    iptables -P INPUT DROP
    iptables -A INPUT -i lo -j ACCEPT
    iptables -A INPUT -s 8080 -j ACCEPT
    iptables-save > /etc/iptables-rules
[08/18/2020, 10:44:21] Rules Updated!

[08/18/2020, 10:44:30] Updating Rules
    iptables -F
    iptables -X
    iptables -t nat -F
    iptables -t nat -X
    iptables -P INPUT DROP
    iptables -A INPUT -i lo -j ACCEPT
    iptables -A INPUT -s 8080 -j ACCEPT
    iptables-save > /etc/iptables-rules
[08/18/2020, 10:44:30] Rules Updated!
```

Figure 11: The update security policies

On default, all connections are blocked as displayed in Figure 10. A user trying to access "example.com" cannot access it. The server updates the security policies (Figure 11) of the user from the "rules.json" file, port 8080 is allowed. Now let's try the same "example.com" again as shown in Figure 12.



Example Domain

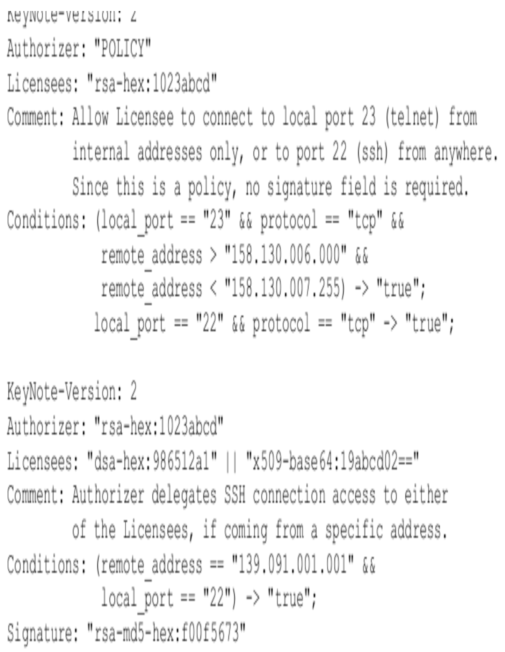
This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.

More information...

Figure 12: The website "example.com" is working

In Figure 12, the previous "examples.com" works perfectly indicative of the operations of a distributed firewall and its policies, it helps with packet filtering by dropping and accepting packet based on some security policies, the major advantage is being defined centrally, so both the insider and the outsiders are seen as a threat, and only with the right policies will you be allowed in.

Application Interacts with Keynote. The Requester is typically a user that authenticates through some application-dependent protocol, and optionally provides credentials. The Verifier needs to determine whether the Requester is allowed to perform the requested action. It is responsible for providing to KeyNote all the necessary information, the local policy, and any credentials. It is also responsible for acting upon KeyNote's response is shown in Figure 13.



```
KeyNote-Version: 2
Authorizer: "POLICY"
Licensees: "rsa-hex:1023abcd"
Comment: Allow Licensee to connect to local port 23 (telnet) from
        internal addresses only, or to port 22 (ssh) from anywhere.
        Since this is a policy, no signature field is required.
Conditions: (local_port == "23" && protocol == "tcp" &&
             remote_address > "158.130.006.000" &&
             remote_address < "158.130.007.255) -> "true";
            local_port == "22" && protocol == "tcp" -> "true";

KeyNote-Version: 2
Authorizer: "rsa-hex:1023abcd"
Licensees: "dsa-hex:986512a1" || "x509-base64:19abcd02=="
Comment: Authorizer delegates SSH connection access to either
        of the Licensees, if coming from a specific address.
Conditions: (remote_address == "139.091.001.001" &&
             local_port == "22") -> "true";
Signature: "rsa-md5-hex:f00f5673"
```

Figure 13: The keynote policies

Example KeyNote Policy and Credential. The local policy allows a particular user (as identified by their public key) to connect access to the telnet port by internal addresses, or to the SSH port from any address. That user then delegates to two other users (keys) the right to connect to SSH from one specific address. Though, the first key can effectively delegate at most the same rights it possesses. KeyNote does not allow rights amplification; any delegation acts as refinement. The above system works in a similar format with my system but security policies are passed in a different format.

5. CONCLUSION

This paper proposed target resolution policies and credentials in a distributed manner with RSA-CRT algorithm. This security policies and implementation were performed in Python and Iptables. It was found that at endpoints, various shortcomings of traditional firewalls are overcome including:

1. Denial-of-service attack mitigation is more effective at the network ingress points (depending on the particular kind of attack).
2. Intrusion detection systems are more effective when located at a traditional firewall, where complete traffic information is available.
3. Security is no longer dependent on restricting the network topology. This allows considerable flexibility in defining the "security perimeter," which can easily be extended to safely include remote hosts and networks
4. Since we no longer solely depend on a single firewall for protection, we eliminate a performance bottleneck.
5. Filtering of certain protocols (such as FTP) which was difficult when done on a traditional firewall, becomes significantly easier, since all the relevant information is present at the decision point, that is, the end host.
6. The number of outside connections the protected network is no longer a cause for administration nightmares. Adding or removing links has no impact on the security of the network.
7. End-to-end encryption is made possible without sacrificing security, as was the case with traditional firewalls. In fact, end-to-end encryption greatly improves the security of the distributed firewall.

However, further experimentation is needed to determine the robustness, efficiency of this proposed packet resolution architecture in networks.

6. REFERENCES

[1] Cwalinski, R. (2019). *An SDN-based Approach to Protect Communication Between Virtual Machines*. 262–265.

[2] Da Costa Júnior, E., da Silva, C., Pinheiro, M. *et al.* (2018). A new approach to deploy a self-adaptive distributed firewall. *J Internet Serv Appl.*, 9**(**12), https://doi.org/10.1186/s13174-018-0083-6

[3] Fraser, B. (2016). Networking Group. RFC 2196. Site security handbook, https://www.ietf.org/rfc/rfc2196.txt.

[4] Hu, H., Han, W., Kyung, S., Wang, J., Ahn, G., Zhao, Z., & Li, H. (2019). Computers & Security Towards a reliable firewall for software-defined networks. *Computers & Security*, *87*, 101597. https://doi.org/10.1016/j.cose.2019.101597

[5] Jmal, R., & Fourati, L. C. (2020). Distributed software defined information centric networking. *Int. J. High Performance Computing and Networking*, *16*(1), 14–25.

[6] Lai, Y., Jiang, G., Li, J., & Yang, Z. (2009, February). Design and implementation of distributed firewall system

for IPv6. In 2009 International Conference on Communication Software and Networks, pp. 428-432. IEEE.

[7] Nife, F. N., & Kotulski, Z. (2020). Application ‑ Aware Firewall Mechanism for Software. *Journal of Network and Systems Management*, *28*(3), 605–626. https://doi.org/10.1007/s10922-020-09518-z

[8] Pandikumar, T., & Gidey, M. (2017). Data Security in LAN Using Distributed Firewall. *Int. Research Journal of Engineering and Technology, 04*(05), 867-873.

[9] Prabakaran, S. (2019). *Stateful firewall - enabled software - defined network with distributed controllers : A network performance study*. (October), 1–17. https://doi.org/10.1002/dac.4237

[10] Sahay, R., Meng, W., & Jensen, C. D. (2019). The application of Software Defined Networking on securing computer networks : A survey. *Journal of Network and Computer Applications*, *131*, 89–108. https://doi.org/10.1016/j.jnca.2019.01.019

[11] Ioannidis, S., Keromytis, A. D., Bellovin, S. M., & Smith, J. M. (2003). Implementing a Distributed Firewall http://www.cis.upenn.edu/~angelos/Papers/df.pdf

[12] Vacca, J. R. (2007). Practical Internet security. USA: Springer.

[13] [13] Valenza, F., & Cheminod, M. (2020). An Optimized Firewall Anomaly Resolution. *Journal of Internet Services and Information Security*, *10*(1), 22–37. https://doi.org/10.22667/JISIS.2020.02.29.022

[14] Youssef, N. B., Bouhoula, A., & Jacquemard, F. (2009). Automatic Verification of Conformance of Firewall Configurations to Security Policies. In *IEEE Symposium on Computers and Communications*, 526–531, IEEE Computer Society Press.

[15] da Costa Junior, E., da Silva, C., Pinheiro, M. et al. A new approach to deploy a self-adaptive distributed firewall. *J Internet Serv Appl* **9,** 12 (2018). https://doi.org/10.1186/s13174-018-0083-6.

[16] Shieh, Choung-Yaw. (2013). Distributed firewall architecture using virtual machines.