SECURE SOFTWARE DEVELOPMENT LIFE CYCLE

SOCIETY HUB

Vineet Rao - 161070001

Ameya Daddikar - 161070015

Aim

To implement a security framework (secure software lifecycle) for Society Management System (SocietyHub).

Phases	**Description:** Society Hub is a web platform for housing societies to delegate a number of management responsibilities to. It aids in easy storage and retrieval of information related to various parts of the society, along with an account system for added security.

Analysis	• Account system for each flat. • Access to profile details, and issues raised. • Viewing of latest notices pertaining to the society • Completely responsive web design, and can easily be scaled along with the database. • Privileges can be given to administrator and program manager only so that anyone else can not misuse those privileges. • Special accounts for Community Members and Society Admins that can access Society Admin Pages to manage the society	
Design	Input Validation	• Validate and sanitize all input data • Ensure validation on trusted system (server) • Prevent unsanitized data from being passed directly to query/command evaluation engines • Always use minimal user-input data
	Authentication and Password Management	• All access to a society's data and pages must be secured with authentication • All authentication must be performed on a trusted system • Ensure any failure is secure • Store passwords in a hashed manner, on a centralized system.
	Authorization	• Assign roles to users (member, building manager, society manager, committee member, etc.)
	Sensitive details management	• Sensitive identifiable data must be protected • Show only to authorized users • Try to store hashes were actual data is not needed

	Session management	● Create session identifier for every login, at trusted system ● Ensure finite time limit for session ● Invalidate session identifier on logout, regardless of timeout
	Cryptographic practices	● Use hashing for storage of sensitive data like passwords ● Prefer encrypted communication (HTTPS) ● Ensure that master secrets are stored securely (especially confiedential credentials) ● Use compliant cryptographic modules. ● Use only module functionality for cryptographic functions like random number generation, hashing, signature, etc.
	Exception management	● Exception should be handled without causing program failure ● No sensitive data should be displayed to the user ● Maintain logs of every exception
	Database Security	● Use prepared statements ● Ensure variables used as parameters are strongly type ● Use minimum connection time ● Where possible, use views and stored procedures as opposed to base tables
Implementation	Input Validation	1. Conduct all data validation on server side. 2. If any potentially hazardous characters must be allowed as input, be sure that you implement additional controls like output encoding, secure task specific APIs and accounting for the utilisation of that data throughout the application . Examples of common hazardous characters include: < > " ' % ( ) & + \ \' \" 3. Validate all client provided data before processing, including all parameters, URLs and HTTP header content (e.g. Cookie names and values). Be sure to include automated post backs from JavaScript, Flash or other embedded code 4. Verify that header values in both requests and responses contain only ASCII characters 5. Utilize canonicalization to address double encoding or other forms of obfuscation attacks)

	Authentication	1. Require authentication for all pages and resources, except those specifically intended to be public 2. Establish and utilize standard, tested, authentication services whenever possible 3. Use a centralized implementation for all authentication controls, including libraries that call external authentication services 4. If your application manages a credential store, it should ensure that only cryptographically strong oneway salted hashes of passwords are stored and that the table/file that stores the passwords and keys is write-able only by the application. (Do not use the MD5 algorithm if it can be avoided) 5. Authentication failure responses should not indicate which part of the authentication data was incorrect 6. Use only HTTP POST requests to transmit authentication credentials 7. Enforce password complexity requirements established by policy or regulation. Authentication credentials should be sufficient to withstand attacks that are typical of the threats in the deployed environment. (e.g., requiring the use of alphabetic as well as numeric and/or special characters) 8. Password reset and changing operations require the same level of controls as account creation and authentication. 9. If using email based resets, only send email to a pre-registered address with a temporary link/password 10. Notify users when a password reset occurs 11. Implement monitoring to identify attacks against multiple user accounts, utilizing the same password. This attack pattern is used to bypass standard lockouts, when user IDs can be harvested or guessed

	Authorization	1. Use only trusted system objects, e.g. server side session objects, for making access authorization decisions 2. Use a single site-wide component to check access authorization. This includes libraries that call external authorization services 3. Enforce authorization controls on every request, including those made by server side scripts, "includes" and requests from rich client-side technologies like AJAX and Flash 4. Segregate privileged logic from other application code 5. Restrict access to files or other resources, including those outside the application's direct control, to only authorized users 6. Restrict access to protected URLs to only authorized users 7. Restrict access to protected functions to only authorized users 8. Restrict access to services to only authorized users 9. Restrict access to application data to only authorized users 10. Restrict access to user and data attributes and policy information used by access controls 11. If state data must be stored on the client, use encryption and integrity checking on the server side to catch state tampering. 12. Use the "referer" header as a supplemental check only, it should never be the sole authorization check, as it is can be spoofed 13. If long authenticated sessions are allowed, periodically re-validate a user's authorization to ensure that their privileges have not changed and if they have, log the user out and force them to re-authenticate

	Sensitive management	1. Do not disclose sensitive information in error responses, including system details, session identifiers or account information 2. Protect all cached or temporary copies of sensitive data stored on the server from unauthorized access and purge those temporary working files a soon as they are no longer required. 3. Encrypt highly sensitive stored information, like authentication verification data, even on the server side. Always use well vetted algorithms 4. Do not include sensitive information in HTTP GET request parameters 5. Disable auto complete features on forms expected to contain sensitive information, including authentication 6. Disable client side caching on pages containing sensitive information. Cache-Control: no-store, may be used in conjunction with the HTTP header control "Pragma: no-cache", which is less effective, but is HTTP/1.0 backward compatible
	Session management	1. Use the server or framework's session management controls. The application should only recognize these session identifiers as valid 2. Session identifier creation must always be done on a trusted system (e.g., The server) 3. Session management controls should use well vetted algorithms that ensure sufficiently random session identifiers 4. Logout functionality should fully terminate the associated session or connection 5. Generate a new session identifier if the connection security changes from HTTP to HTTPS, as can occur during authentication. Within an application, it is recommended to consistently utilize HTTPS rather than switching between HTTP to HTTPS. 6. Protect server side session data from unauthorized access, by other users of the server, by implementing appropriate access controls on the server 7. Generate a new session identifier on any re-authentication 8. Supplement standard session management for sensitive server-side operations, like account management, by utilizing per-session strong random

		tokens or parameters. This method can be used to prevent Cross Site Request Forgery attacks
	Cryptographic practices	1. All cryptographic functions used to protect secrets from the application user must be implemented on a trusted system (e.g., The server) 2. Protect master secrets from unauthorized access 3. Cryptographic modules should fail securely 4. All random numbers, random file names, random GUIDs, and random strings should be generated using the cryptographic module's approved random number generator when these random values are intended to be un-guessable 5. Cryptographic modules used by the application should be compliant to FIPS 140-2 or an equivalent standard. (See http://csrc.nist.gov/groups/STM/cmvp/validation.html)
	Exception management	1. Do not disclose sensitive information in error responses, including system details, session identifiers or account information 2. Use error handlers that do not display debugging or stack trace information 3. Properly free allocated memory when error conditions occur 4. All logging controls should be implemented on a trusted system (e.g., The server) 5. Error handling logic associated with security controls should deny access by default
Testing	● Test based on test data and cases generated from UML diagrams. ● Perform vulnerability scanning for common ones, and known vulnerabilities in used libraries ● Test variety of input data, valid as well as invalid ● Make requests to non-existent pages, use wrong HTTP method, to check error messages ● Attempt unauthorized activities with a different authentication, like adding buildings from a normal member account.	

Deployment	Network Threats: All network guards like firewall, application firewall, honey-pot and IDS should be updated otherwise following threats are present Information gathering, Sniffing or eavesdropping , spoofing, Session hijacking, Denial of service Server Threats : Server on which I am going to deploy "SocietyHub" should be secure otherwise following threats are possible Viruses , Trojan horse and worms Foot printing Password cracking Denial of service Arbitrary code execution Unauthorized access. Common Procedures: 1. Remove test code or any functionality not intended for production, prior to deployment 2. Collaboration Between Development and Operations 3. Build & Release Automation 4. Minimize the Amount of Change 5. Create and Test SQL Change Scripts 6. Setup Synthetic Transactions Tests 7. Setup network guards like firewalls, application firewalls, IDS, etc.
Maintenance	● Raise alerts on repeated attempts to access unauthorized requests. ● All tables used in "SocietyHub" should be updated properly. ● Allow society Admins to raise issue with website to block some user if necessary. ● Regular system backups and checks

Experiment 2

Name	Ameya S. Daddikar
College I.D.	161070015
Course	Btech. Computer Engineering

Aim

To study and implement OWASP attacks.

Theory

Injection

Injection flaws, such as SQL, QS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data ran trick the interpreter into executing unintended commands or accessing data without proper Authorization.

Mitigation

- SQL injection filtering works in a similar way to email's spam filters. Database firewalls detect SQL injections based on the number of invalid queries from host, while there are OR and UNION blocks inside of request, or others.
- With most development platforms, parameterized statements that work with parameters can be used(sometimes called placeholders or bind variables) instead of embedding user input in the statement. A placeholder can only store a value of the given type and not an arbitrary SQL fragments. Hence the SQL injection would simply be treated as a strange (and probably invalid) parameter value.

Broken Authentication and Session Management

Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.

Mitigation

- **Password Strength** - passwords should have restrictions that require a minimum size and complexity for the password.
- **Password Storage** - All passwords must be stored in either hashed or encrypted form to protect them from exposure, regardless of where they are stored.

Cross-Site Scripting (XSS)

XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser, which can hijack user sessions, deface websites, or redirect the user to malicious sites.

Mitigation

- Using frameworks that automatically escape XSS by design, such as the latest Ruby on Rails, ReactJS. Learn the limitations of each framework's XSS protection and appropriately handle the use cases which are not covered.
- Escaping untrusted HTTP request data based on the context in the HTML output.

Broken Access control

Broken access control occurs if a user is able to access unauthorized resources, this can be
access to restricted pages, database, directories et cetera. Applications have various account types depending on the users: admins, operators and reporting groups etc. One common problem is that the developers restrict the privileges just on the UI side and not on the server side. If exploited, each user can have admin rights.

Mitigation

- Disable Client Side Caching – Many users access web applications from shared computers located in libraries, schools, airports, and other public access points. Browsers frequently cache web pages that can be accessed by attackers to gain access to otherwise inaccessible parts of sites. Developers should use multiple mechanisms, including HTTP headers and meta tags, to be sure that pages containing sensitive information are not cached by user's browsers.
- Forced Browsing Past Access Control Checks – many sites require users to pass certain checks before being granted access to certain URLs that are typically 'deeper' down in the site. These checks must not be bypassable by a user that simply skips over the page with the security check.

Security Misconfiguration

Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.

Mitigation

- A minimal platform without any unnecessary features, components, documentation, and samples. Remove or do not install unused features and frameworks.

- A segmented application architecture that provides effective, secure separation between components or tenants, with segmentation, containerization, or cloud security groups (ACLs).

Sensitive Data Exposure

Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

Mitigation

- Classify data processed, stored or transmitted by an application. Identify which data is sensitive according to privacy laws, regulatory requirements, or business needs.
- Ensure up-to-date and strong standard algorithms, protocols, and keys are in place; use proper key management.

XML External Entity

An XML External Entity attack is a type of attack against an application that parses XML input. This attack occurs when XML input containing a reference to an external entity is processed by a weakly configured XML parser. This attack may lead to the disclosure of confidential data, denial of service, server side request forgery, port scanning from the perspective of the machine where the parser is located, and other system impacts.

Mitigation

- Since the whole XML document is communicated from an untrusted client, it's not usually possible to selectively validate or escape tainted data within the system identifier in the DTD. Therefore, the XML processor should be configured to use a local static DTD and disallow any declared DTD included in the XML document.

Cross-Site Request Forgery (CSRF)

A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application things are legitimate requests from the victim.

Mitigation

- The preferred option is to include the unique token in a hidden field. This causes the value to be sent in the body of the HTTP request, avoiding its inclusion in the URL, which is more prone to exposure.
- Requiring the user to reauthenticate, or prove they are a user (e.g., via a CAPTCHA) can also protect against CSRF.

Using Components with Known Vulnerabilities

Components, such as libraries, frameworks and other software modules, almost run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.

Mitigation

- Remove unused dependencies, unnecessary features, components, files, and documentation.
- Only obtain components from official sources over secure links. Prefer signed packages to reduce the chance of including a modified, malicious component.

Insufficient logging and monitoring

With all the countermeasures in place attacks still happen and that gets noticed only after an incident has happened. If undetected the attackers could have compromised the systems long back and gained persistence. To ensure the malicious intent of the attackers gets noticed beforehand, it is essential to log all the activity and monitor it for any suspicious behavior.

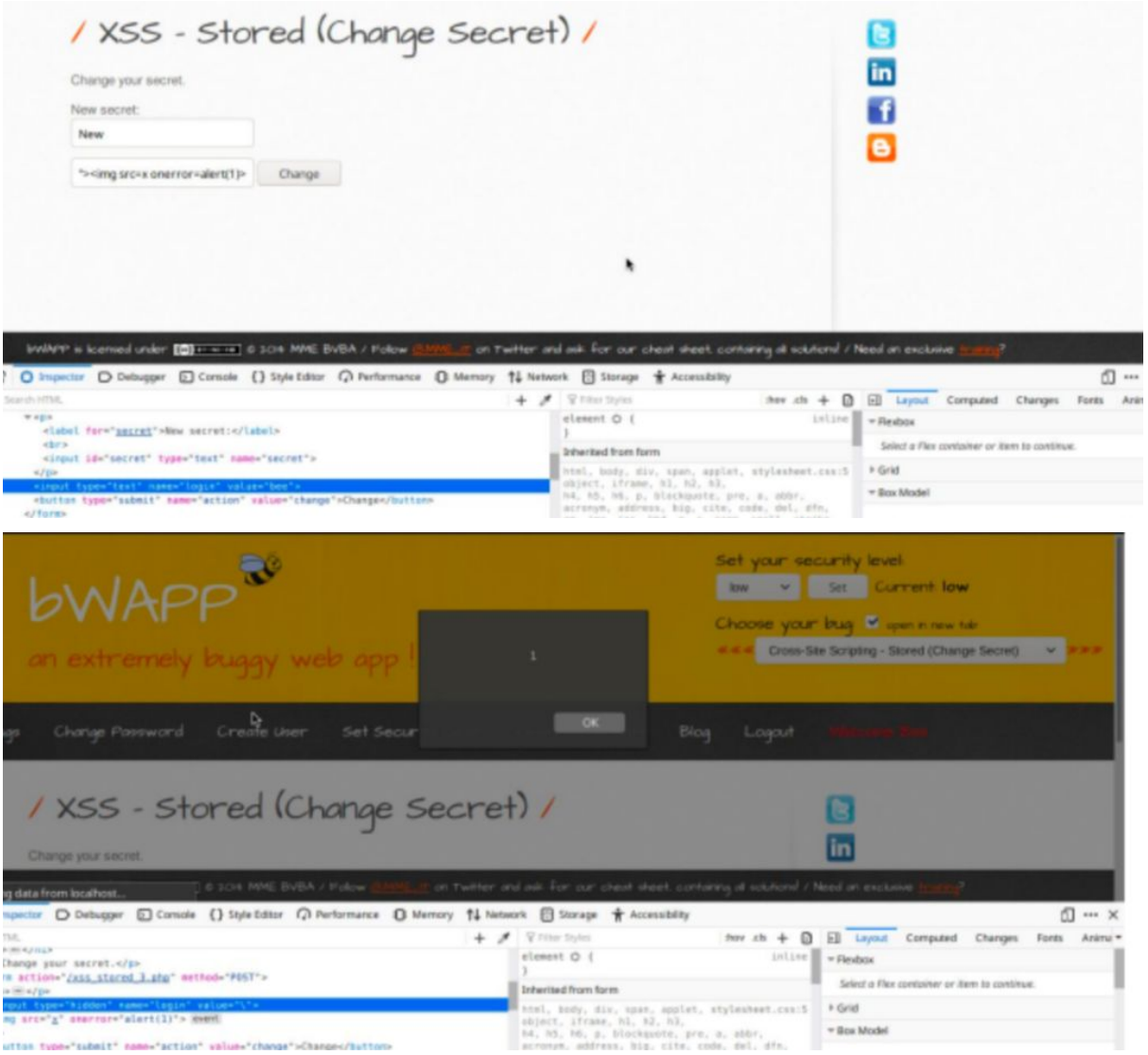
Mitigation

- Ensure that logs are generated in a format that can be easily consumed by a centralized log management solutions.
- Establish effective monitoring and alerting such that suspicious activities are detected and responded to in a timely fashion.

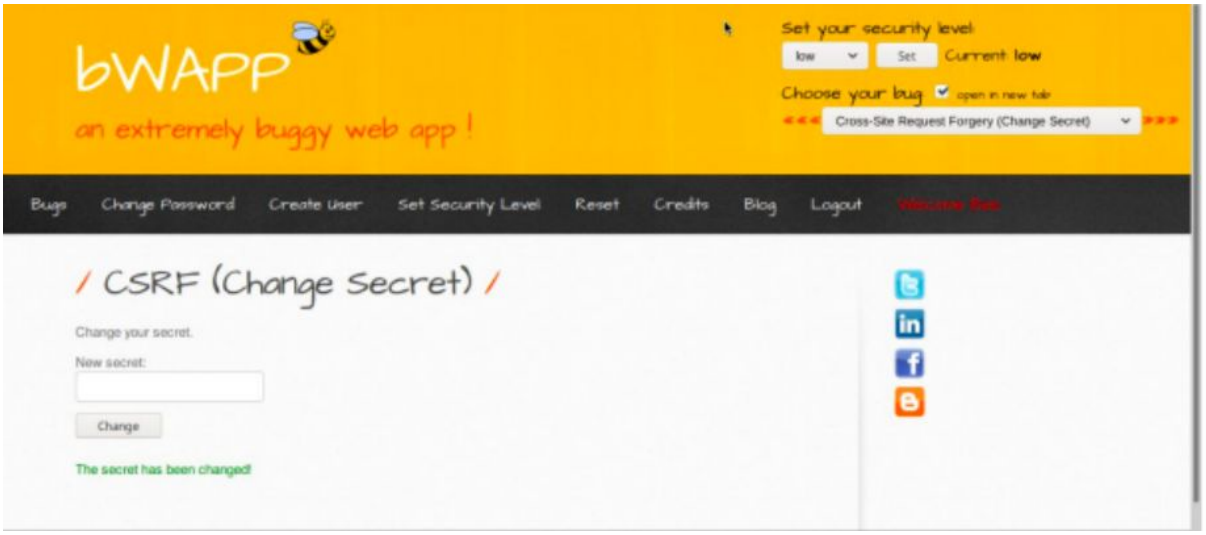
Outputs

SQL Injection

XSS Attack



CSRF Attack



Conclusion

Thus we have studied and replicated OWASP attacks in a controlled and monitored environment to get a better understanding of their working and possible mitigation strategies.

Experiment 3

Aim

To study network analysis and monitoring tools such as: Wire shark, Nmap, Hping.

Theory

Wireshark

Wireshark is the world's foremost and widely-used network protocol analyzer. It lets you see what's happening on your network at a microscopic level and is the de facto (and often de jure) standard across many commercial and non-profit enterprises, government agencies, and educational institutions. Wireshark development thrives thanks to the volunteer contributions of networking experts around the globe and is the continuation of a project started by Gerald Combs in 1998.

Wireshark Features

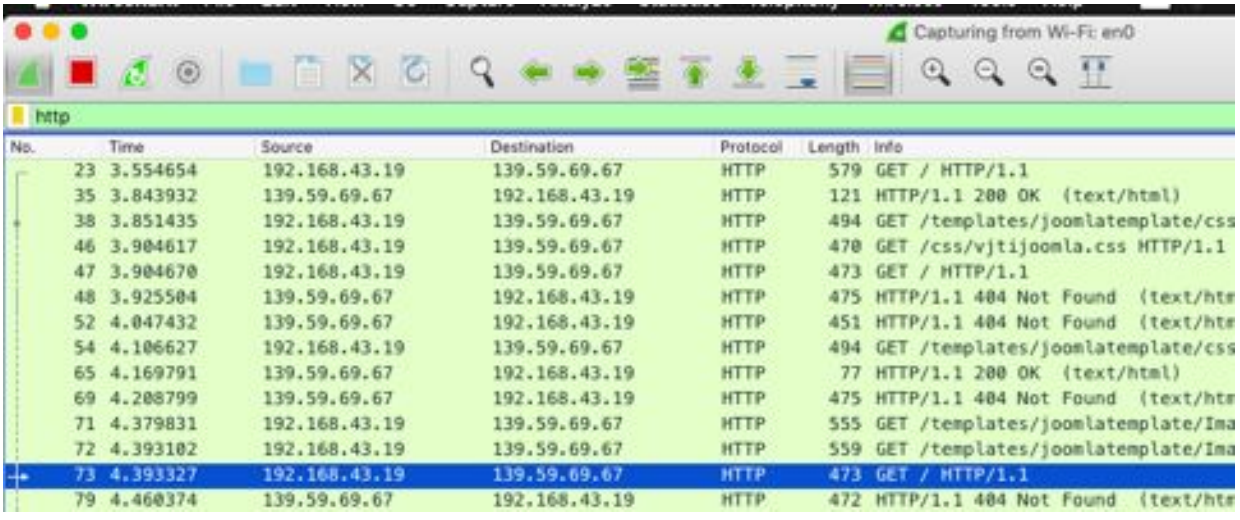
- Deep inspection of hundreds of protocols, with more being added all the time
- Live capture and offline analysis
- Standard three-pane packet browser
- Multi-platform: Runs on Windows, Linux, macOS, Solaris, FreeBSD, NetBSD, and many others
- Captured network data can be browsed via a GUI, or via the TTY-mode TShark utility
- The most powerful display filters in the industry
- Rich VoIP analysis
- Read/write many different capture file formats: tcpdump (libpcap), Pcap NG, Catapult DCT2000, Cisco Secure IDS iplog, Microsoft Network Monitor, Network General Sniffer® (compressed and uncompressed), Sniffer® Pro, and NetXray®, Network Instruments Observer, NetScreen snoop, Novell LANalyzer, RADCOM WAN/LAN Analyzer, Shomiti/Finisar Surveyor, Tektronix K12xx, Visual Networks Visual UpTime, WildPackets EtherPeek/TokenPeek/AiroPeek, and many others
- Capture files compressed with gzip can be decompressed on the fly
- Live data can be read from Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, FDDI, and others (depending on your platform)
- Decryption support for many protocols, including IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP, and WPA/WPA2
- Coloring rules can be applied to the packet list for quick, intuitive analysis
- Output can be exported to XML, PostScript®, CSV, or plain text

TEST SERVER	1. http://vjti.ac.in 2. https://en.wikipedia.org/ 3. http://norvig.com/big.txt
BROWSER/ HTTP SOFTWARE	1. Chrome Version 75.0.3770.142 (Official Build) (64-bit) 2. Postman Version 7.3.4 (7.3.4)
TEST PLATFORM	macOS Mojave 10.14.6

Q1. The basic HTTP GET response interaction:

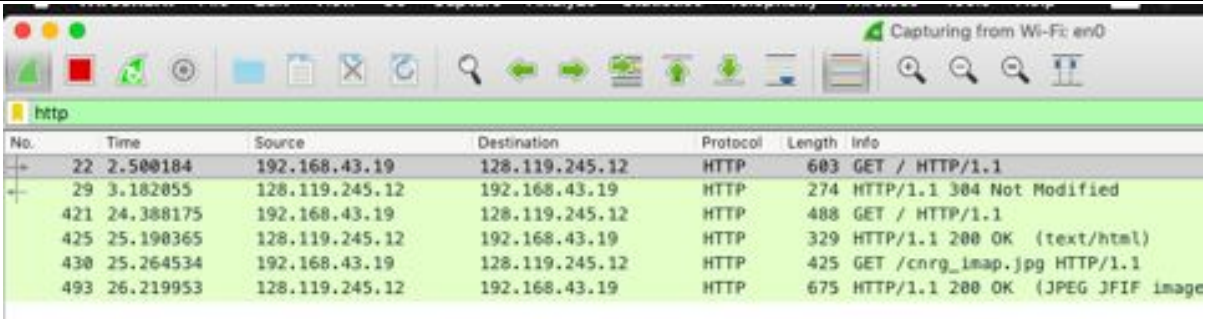
- Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?

Browser is using HTTP 1.1 ; Server responds with HTTP 1.1



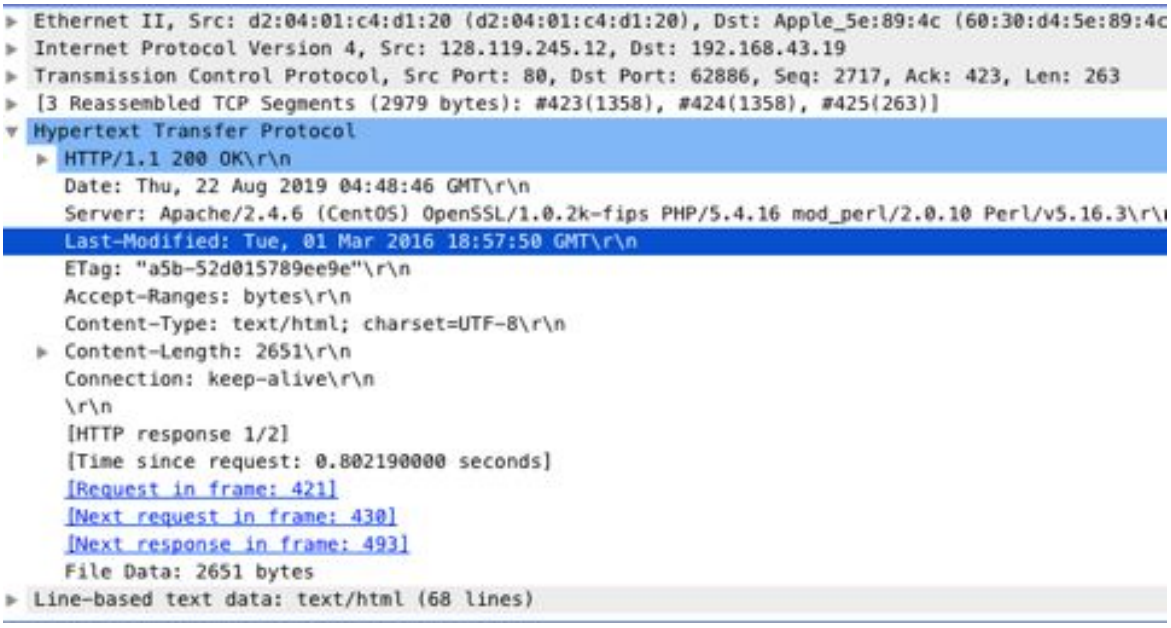
- What is the IP address of your computer? Of the gaia.cs.umass.edu server?

CLIENT IP	192.168.43.19
SERVER IP	128.119.245.12



- When was the HTML file that you are retrieving last modified at the server?

“Last-Modified: Tue, 01 Mar 2016 18:57:50 GMT\r\n”



- How many bytes of content are being returned to your browser?

2651 bytes (refer image above)

Q2. The HTTP CONDITIONAL GET/response interaction:

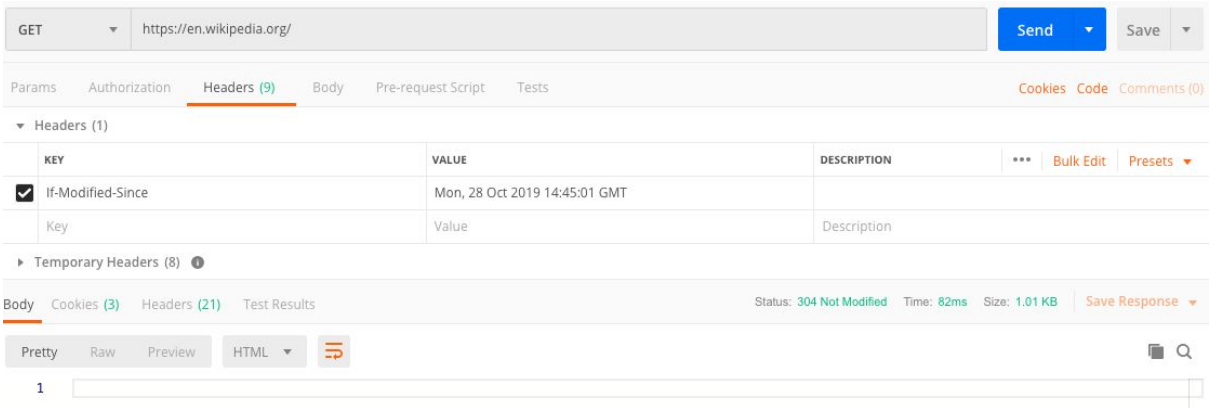
- Inspect the content of the server response. Did the server explicitly return the contents of the file?

NO

- What is the HTTP status code and phrase returned from the server in response to the second HTTP GET? Did the server explicitly return the contents of the file?

304 NOT MODIFIED; NO

(HTTP CONDITIONAL REQUEST SCREENSHOT of the Postman GUI)



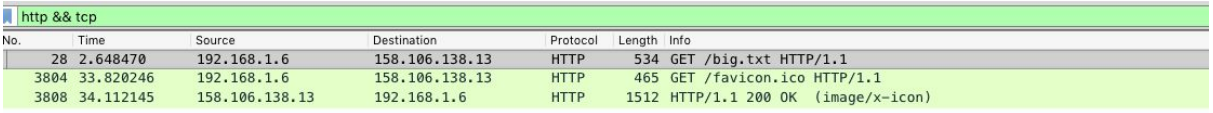
Q3. Retrieving Long Documents:

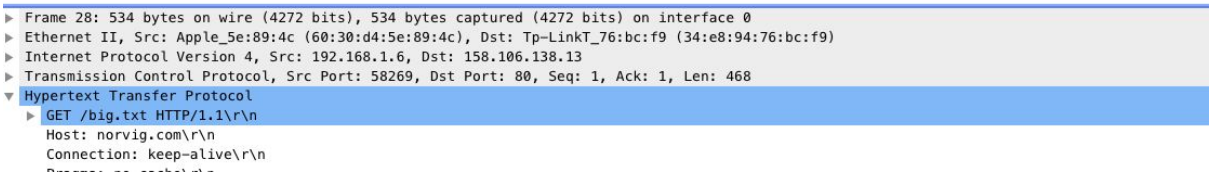
- How many HTTP GET request messages were sent by your browser?

1 request only

- What is the status code and phrase associated with the response to the HTTP GET request?

200 OK





Nmap

Nmap ("Network Mapper") is a free and open source license utility for network discovery and security auditing. Many systems and network administrators also find it useful for tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime.

Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. It was designed to rapidly scan large networks, but works fine against single hosts.

Nmap runs on all major computer operating systems, and official binary packages are available for Linux, Windows, and Mac OS X.

Nmap Features

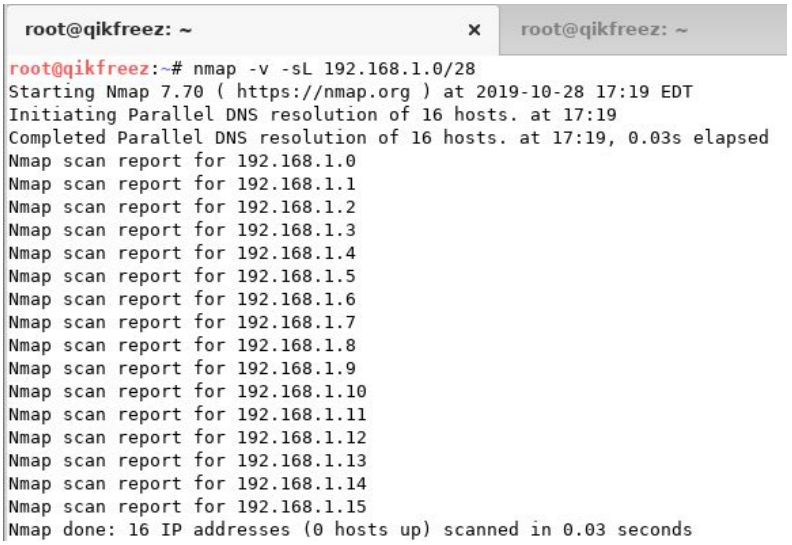
- **Flexible**: Supports dozens of advanced techniques for mapping out networks filled with IP filters, firewalls, routers, and other obstacles. This includes many port scanning mechanisms (both TCP & UDP), OS detection, version detection, ping sweeps, and more.
- **Powerful**: Nmap has been used to scan huge networks of literally hundreds of thousands of machines.

- **Portable**: Most operating systems are supported, including Linux, Microsoft Windows, FreeBSD, OpenBSD, Solaris, IRIX, Mac OS X, HP-UX, NetBSD, Sun OS, Amiga, and more.
- **Easy**: While Nmap offers a rich set of advanced features for power users, you can start out as simply as "nmap -v -A *targethost*". Both traditional command line and graphical (GUI) versions are available to suit your preference. Binaries are available for those who do not wish to compile Nmap from source.
- **Free**: The primary goals of the Nmap Project is to help make the Internet a little more secure and to provide administrators/auditors/hackers with an advanced tool for exploring their networks. Nmap is available for free download, and also comes with full source code that you may modify and redistribute under the terms of the license.
- **Well Documented**: Significant effort has been put into comprehensive and up-to-date man pages, whitepapers, tutorials, and even a whole book.
- **Supported**: While Nmap comes with no warranty, it is well supported by a vibrant community of developers and users. Most of this interaction occurs on the Nmap mailing lists. Most bug reports and questions should be sent to the nmap-dev list, but only after you read the guidelines.
- **Acclaimed**: Nmap has won numerous awards, including "Information Security Product of the Year" by Linux Journal, Info World and Codetalker Digest. It has been featured in hundreds of magazine articles, several movies, dozens of books, and one comic book series.
- **Popular**: Thousands of people download Nmap every day, and it is included with many operating systems (Redhat Linux, Debian Linux, Gentoo, FreeBSD, OpenBSD, etc). It is among the top ten (out of 30,000) programs at the Freshmeat.Net repository. This is important because it lends Nmap its vibrant development and user support communities.

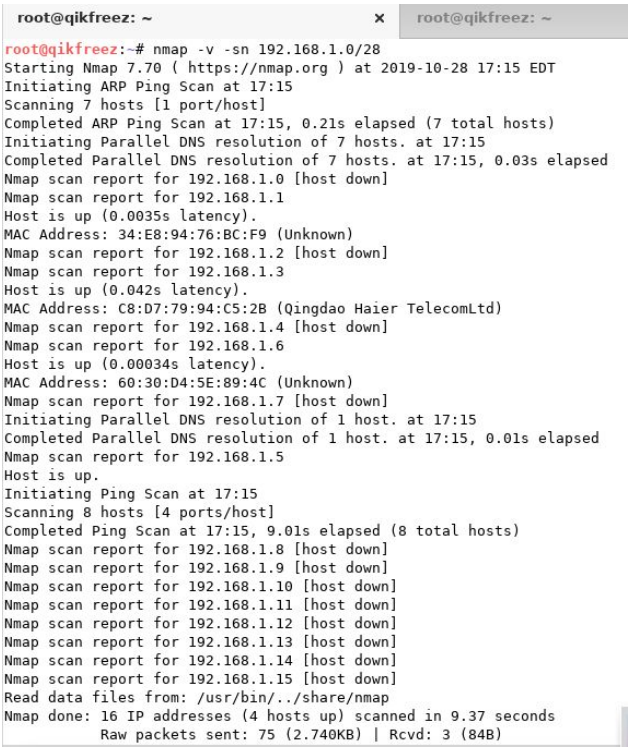
Nmap Commands

1. Detect details about hosts in the network

-sL (List Scan)
The list scan is a degenerate form of host discovery that simply lists each host of the network(s) specified, without sending any packets to the target hosts. By default, Nmap still does reverse-DNS resolution on the hosts to learn their names.



-sn (No port scan)
This option tells Nmap not to do a port scan after host discovery, and only print out the available hosts that responded to the host discovery probes. This is often known as a “ping scan”, but you can also request that traceroute and NSE host scripts be run.



2. Show all the open ports

--open
Only show open (or possibly open) ports

```
root@qikfreez:~# nmap --open google.com
Starting Nmap 7.70 ( https://nmap.org ) at 2019-10-28 17:33 EDT
Nmap scan report for google.com (172.217.160.174)
Host is up (0.015s latency).
Other addresses for google.com (not scanned): 2404:6800:4009:80a::200e
rDNS record for 172.217.160.174: bom05s12-in-f14.1e100.net
Not shown: 998 filtered ports
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 4.99 seconds
root@qikfreez:~#
```

3. Scan IP Portal

-sN; -sF; -sX (TCP NULL, FIN, and Xmas scans)
These three scan types exploit a subtle loophole in the TCP RFC to differentiate between open and closed ports.

```
root@qikfreez:~# nmap -sF google.com
Starting Nmap 7.70 ( https://nmap.org ) at 2019-10-28 17:51 EDT
Nmap scan report for google.com (172.217.160.174)
Host is up (0.014s latency).
Other addresses for google.com (not scanned): 2404:6800:4009:80a::200e
rDNS record for 172.217.160.174: bom05s12-in-f14.1e100.net
All 1000 scanned ports on google.com (172.217.160.174) are open|filtered

Nmap done: 1 IP address (1 host up) scanned in 4.11 seconds
```

4. Find port ranges

-p <port ranges>
Only scan specific ports.

```
root@qikfreez:~# nmap -p 1-600 192.168.1.1
Starting Nmap 7.70 ( https://nmap.org ) at 2019-10-28 17:53 EDT
Nmap scan report for 192.168.1.1
Host is up (0.0038s latency).
Not shown: 597 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
23/tcp open  telnet
80/tcp open  http
MAC Address: 34:E8:94:76:BC:F9 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.20 seconds
```

5. Find protocol list

-sO
IP protocol scan allows you to determine which IP protocols (TCP, ICMP, IGMP, etc.) are supported by target machines.

```
root@qikfreez:~# nmap -sO google.com
Starting Nmap 7.70 ( https://nmap.org ) at 2019-10-28 17:35 EDT
Nmap scan report for google.com (172.217.160.174)
Host is up (0.015s latency).
Other addresses for google.com (not scanned): 2404:6800:4009:80a::200e
rDNS record for 172.217.160.174: bom05s12-in-f14.1e100.net
Not shown: 254 open|filtered protocols
PROTOCOL STATE SERVICE
1        open  icmp
6        open  tcp

Nmap done: 1 IP address (1 host up) scanned in 2.99 seconds
root@qikfreez:~# %
```

6. For virus and os detection

-O: Enable OS detection
--osscan-limit: Limit OS detection to promising targets
--osscan-guess: Guess OS more aggressively

```
root@qikfreez:~# nmap -O google.com
Starting Nmap 7.70 ( https://nmap.org ) at 2019-10-28 17:46 EDT
Nmap scan report for google.com (172.217.160.174)
Host is up (0.014s latency).
Other addresses for google.com (not scanned): 2404:6800:4009:80a::200e
rDNS record for 172.217.160.174: bom05s12-in-f14.1e100.net
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1
 closed port
OS fingerprint not ideal because: Missing a closed TCP port so results incomplete
No OS matches for host

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.53 seconds
```

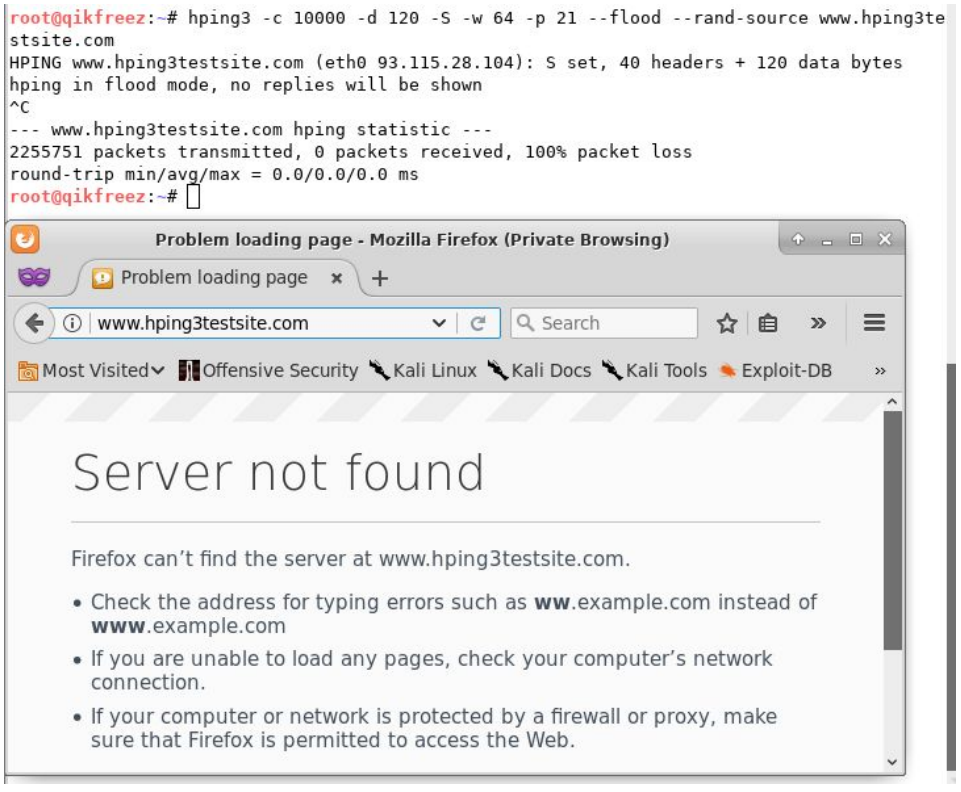
Hping3

hping3 is a network tool able to send custom TCP/IP packets and to display target replies like ping program does with ICMP replies. hping3 handle fragmentation, arbitrary packets body and size and can be used in order to transfer files encapsulated under supported protocols.

Hping3 Features:

- Test firewall rules
- Advanced port scanning
- Test net performance using different protocols, packet size, TOS (type of service) and fragmentation
- Path MTU discovery
- Transferring files between even really fascist firewall rules.
- Traceroute-like under different protocols.
- Firewalk-like usage
- Remote OS fingerprinting
- TCP/IP stack auditing.

Performing DOS attack using Hping3



Conclusion

We've explored multiple options and configurations of the tools Wireshark, Nmap & Hping3 and analyzed public and private network environments.

Experiment 4

Name	Ameya S. Daddikar
College I.D.	161070015
Course	Btech. Computer Engineering

Aim

To study TCP/IP vulnerabilities, attacks and defence mechanism.

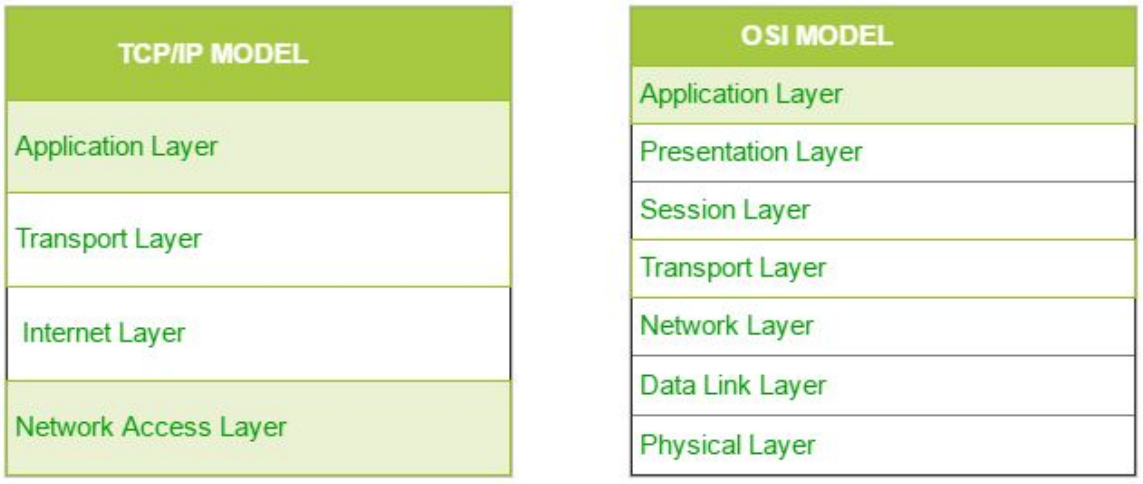
Theory

The TCP/ IP model

The OSI Model is just a reference/logical model. It was designed to describe the functions of the communication system by dividing the communication procedure into smaller and simpler components. But when we talk about the TCP/IP model, it was designed and developed by Department of Defense (DoD) in the 1960s and is based on standard protocols. It stands for Transmission Control Protocol/Internet Protocol. The TCP/IP model is a concise version of the OSI model. It contains four layers, unlike seven layers in the OSI model. The layers are:

- Process/Application Layer
- Host-to-Host/Transport Layer
- Internet Layer
- Network Access/Link Layer

The diagrammatic comparison of the TCP/IP and OSI model is as follows :



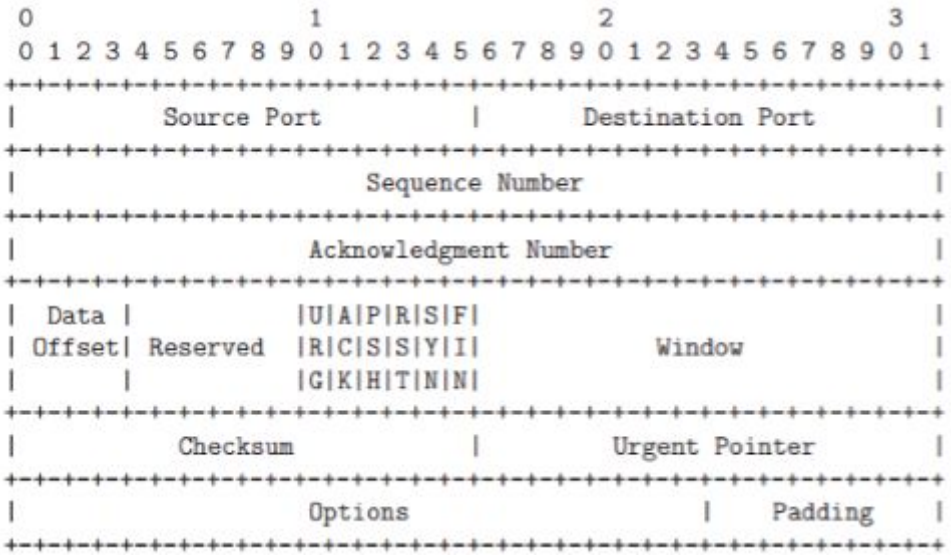
The Transport Layer

Through handshaking and acknowledgments, TCP provides a reliable communication link between two hosts on the internet. When we say that a TCP connection is reliable, we mean that the sender's TCP always knows whether or not a packet reached the receiver's TCP.

If the sender's TCP does not receive an acknowledgment that its packet had reached the destination, the sender's TCP simply re-sends the packet. Additionally, certain data integrity checks on the transmitted packets are carried out at the receiver to ensure that the receiver's TCP accepts only error-free packets.

A TCP connection is full-duplex, meaning that a TCP connection simultaneously supports two byte-streams, one for each direction of a communication link.
TCP includes both a flow control mechanism and a congestion control mechanism.
Flow control means that the receiver's TCP is able to control the size of the segment dispatched by the sender's TCP.This the receiver's TCP accomplishes by putting to use the Window field of an acknowledgment packet.

Congestion control means that the sender's TCP varies the rate at which it places the packets on the wire on the basis of the traffic congestion on the route between the sender and the receiver. The sender TCP can measure traffic congestion by measuring the rate at which the ICMP source-quench messages are received from the routers



IP Protocol

The Internet Protocol (or IP as it generally known), is the network layer of the Internet. IP provides a connection-less service.

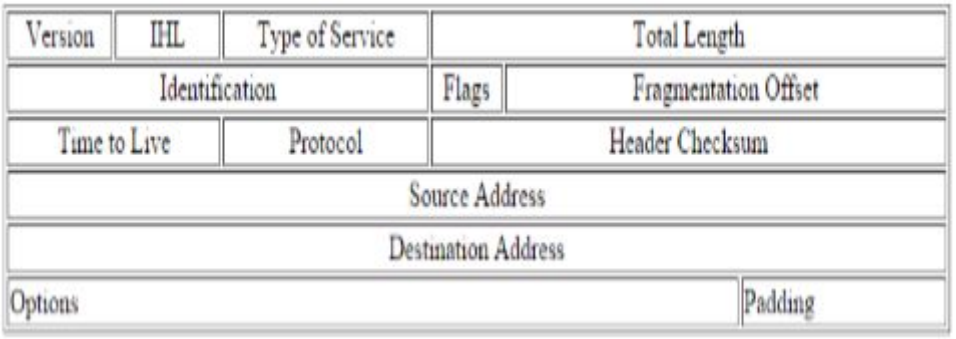
The job of IP is to route and send a packet to the packet's destination. IP provides no guarantee whatsoever, for the packets it tries to deliver.

The IP packets are usually termed datagrams. The datagrams go through a series of routers before they reach the destination.

At each node that the datagram passes through, the node determines the next hop for the datagram and routes it to the next hop.

Since the network is dynamic, it is possible that two datagrams from the same source take different paths to make it to the destination. Since the network has variable delays, it is not guaranteed that the datagrams will be received in sequence. IP only tries for a best-effort delivery.

It does not take care of lost packets; this is left to the higher layer protocols. There is no state maintained between two datagrams; in other words, IP is connection-less.



TCP/ IP Vulnerabilities

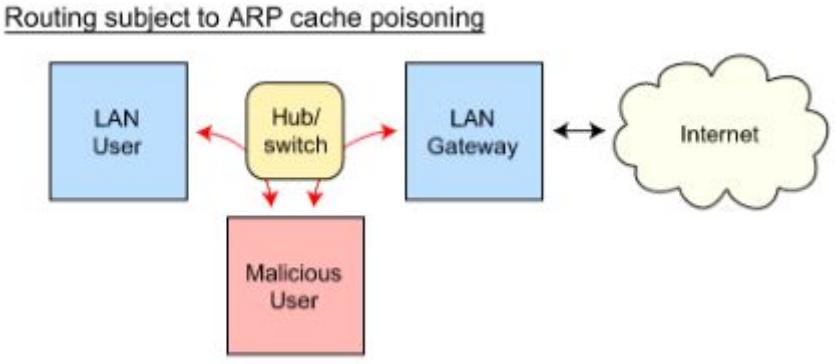
ARP Spoofing

ARP spoofing is a type of attack in which a malicious actor sends falsified ARP (Address Resolution Protocol) messages over a local area network.

This results in the linking of an attacker's MAC address with the IP address of a legitimate computer or server on the network.

Once the attacker's MAC address is connected to an authentic IP address, the attacker will begin receiving any data that is intended for that IP address.

ARP spoofing can enable malicious parties to intercept, modify or even stop data in-transit. ARP spoofing attacks can only occur on local area networks that utilize the Address Resolution Protocol.



Port scanning

Port Scanning is one of the most popular techniques attackers use to discover services that they can exploit to break into systems.

All systems that are connected to a LAN or the Internet via a modem run services that listen to well-known and not so well-known ports.

By port scanning, the attacker can find the following information about the targeted systems: what services are running, what users own those services, whether anonymous logins are supported, and whether certain network services require authentication.

Port scanning is accomplished by sending a message to each port, one at a time. The kind of response received indicates whether the port is used and can be probed for further weaknesses.

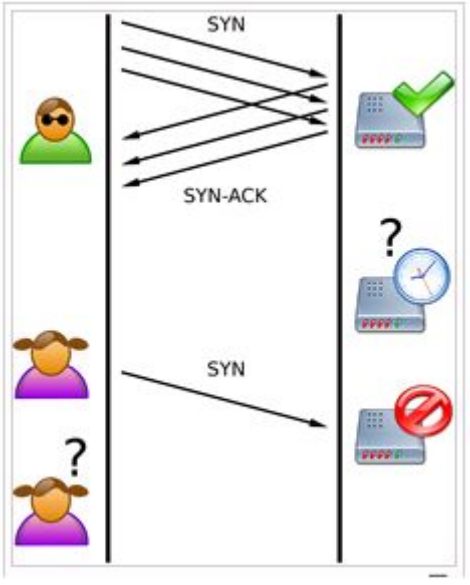
Port scanners are important to network security technicians because they can reveal possible security vulnerabilities on the targeted system.
Port Scan Techniques

- Address Resolution Protocol (ARP)
- TCP connect
- TCP SYN
- TCP FIN

TCP syn flood attack

TCP SYN flood is a type of Distributed Denial of Service (DDoS) attack that exploits part of the normal TCP three-way handshake to consume resources on the targeted server and render it unresponsive.

Essentially, with SYN flood DDoS, the offender sends TCP connection requests faster than the targeted machine can process them, causing network saturation.



The malicious client either does not send the expected ACK, or—if the IP address is spoofed—never receives the SYN-ACK in the first place. Either way, the server under attack will wait for acknowledgement of its SYN-ACK packet for some time.

During this time, the server cannot close down the connection by sending an RST packet, and the connection stays open.

Before the connection can time out, another SYN packet will arrive. This leaves an increasingly large number of connections half-open – and indeed SYN Food attacks are also referred to as “half-open” attacks.

Eventually, as the server’s connection overflow tables fill, service to legitimate clients will be denied, and the server may even malfunction or crash.

The attacker sends several packets but does not send the "ACK" back to the server. The connections are hence half-opened and consuming server resources. Alice, a legitimate user, tries to connect but the server refuses to open a connection resulting in a denial of service.

IP spoofing

IP address spoofing is one of the most frequently used spoofing attack methods. In an IP address spoofing attack, an attacker sends IP packets from a false (or “spoofed”) source address in order to disguise itself.

Denial-of-service attacks often use IP spoofing to overload networks and devices with packets that appear to be from legitimate source IP addresses.

IP spoofing is the action of masking a computer IP address so that it looks like it is authentic.

During this masking process, the fake IP address sends what appears to be a malevolent message coupled with an IP address that appears to be authentic and trusted.

In IP spoofing, IP headers are masked through a form of Transmission Control Protocol (TCP) in which spoofers discover and then manipulate vital information contained in the IP header such as IP address and source and destination information.

Types of spoofing attacks:

- Non-Blind Spoofing
- Blind Spoofing
- Man In the Middle Attack
- Denial of Service Attack
- DNS Spoofing

Defense Mechanisms

Some simple prevention mechanisms like password protecting the system to avoid unauthorized use have become widely popular.

1 Firewalls - Firewalls are systems designed to prevent unauthorized access to or from a network. A firewall is a dedicated appliance or software running on a system which inspects network traffic passing through it and denies or permits passage based on a set of rules. Firewalls can be implemented in both hardware and software or a combination of both. Firewalls can be of the following types:- Packet filter:- It inspects each packet entering or leaving the network and rejects or accepts based on defined rules. It is effective and transparent but difficult to configure. IP spoofing can be easily done for packet filter firewalls. Application Gateway:- Decision to allow or disallow depends upon specific application for e.g. ftp, Telnet etc. It is very effective but imposes performance degradation. Circuit-level Gateway:- It applies security mechanism when a TCP or UDP connection is established. After the connection establishment no further checking is done and packets could flow between hosts. Proxy server:- It sits between the client and server. A client requires some services such as a file, connection web page or other resources available on a different server. The proxy server validates the request with its filter rules and after the request is validated by the filter, the proxy provides the resources by connecting to the relevant servers and requesting services on behalf of clients. Some of the commonly used firewalls are :- Netfilter: It is an open source, firewall written in C that supports different IPV4 protocols and can be used with command line interface [10]. IPFilter: is an open source firewall that supports both IPv4 and IPv6. It works on different types of operating systems like AIX, BSD/OS, and some other flavours of BSD and Solaris.

2. A virtual private network (VPN) – A VPN is a private network that uses a public network such as internet to connect remote sites or users together. Instead of using a dedicated, real world connection such as leased line VPN uses “virtual” connections routed through the internet from the company’s private network to the remote site. It is implemented as an additional logical layer on top of an existing larger network.

3. Authentication - Computer Security authentication means verifying the identity of a user logging onto a network. Authentication is the process of determining whether the person is genuinely the person whose identity he or she is claiming to be. In other words authentication is the process of verification of the identity of a user.

4. Intrusion Detection System (IDS) – An intrusion detection system is a software / hardware designed to detect some unwanted attempts to access, manipulate and/or disable computer system. These attempts are generally generated from a network such as internet. It monitors network and/or system activities for malicious activities or policy violations. It is the process of monitoring the events occurring in a system or networks

5. Intrusion Prevention System (IPS) An Intrusion Prevention System (IPS) uses rule based detection technique for detecting malicious traffic and preventing attacks. IPS is the advancement of intrusion detection system IDS.

6. Some popular IDS being used are :- Snort: It is an open source IDS that works on application layer and network layer. It can detect and prevent different attacks like buffer overflow, denial of service attack, port scan, SMB probes and some other attacks.

7. Some commonly used mitigating techniques against IP Spoofing include use of encrypted session in router, using Access Control List for applying the security policies, application of defence mechanisms of upper layers.

8 Counteracting Ping of Death Attack – Techniques like changing the LAN IP address, use of filtering devices such as routers and dedicated firewall to drop all incoming (ICMP) packets are commonly used to defend against such attacks.

9. Mitigating Smurf Attack – For countering a smurf the commonly used techniques include “state-full” inspection at firewall and to deny external ICMP traffic access to the internal network.

10. Countermeasures for Long File or User Name Attacks – Such attacks can be countered by configuring the network filtering device to automatically drop the traffic which contains file names and usernames that are more than 255 characters long.

11. SYN Attack Countermeasures - Identifying the source IP Addresses of the attack packets and then using a firewall or router to block all traffic from this source.

Conclusion

Thus we studied the mechanism and vulnerabilities of the TCP/ IP architecture.

Experiment 5

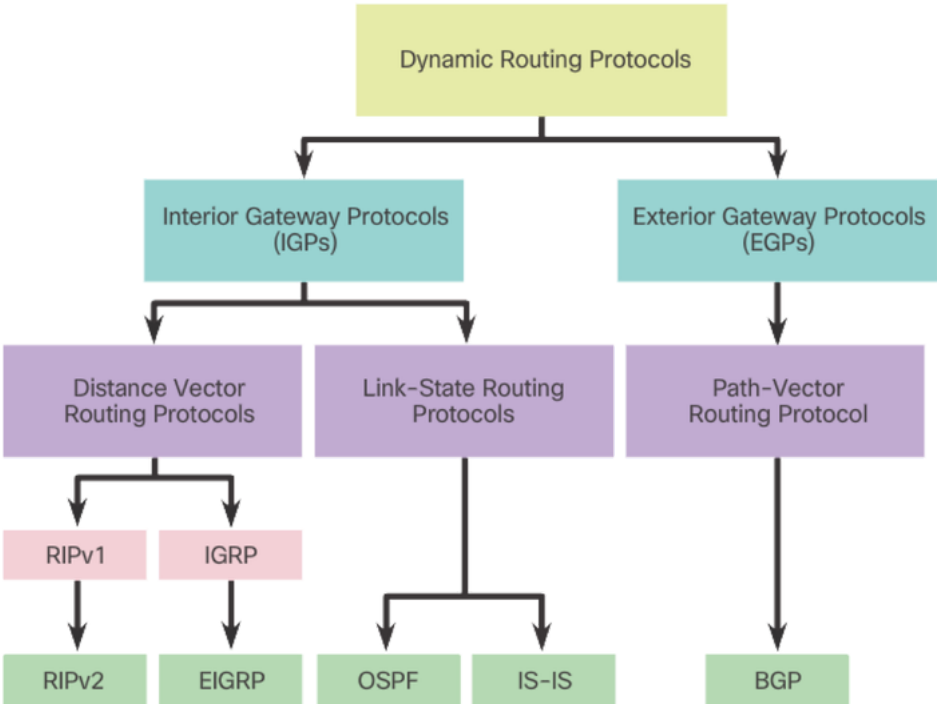
Name	Ameya S. Daddikar
College I.D.	161070015
Course	Btech. Computer Engineering

Aim

To study Routing Protocol attacks and defence mechanism.

Theory

Given below is a classification of various routing protocols. We will be looking for exploits and defenses for RIP, BGP and OSPF protocols.



RIP

Routing Information Protocol developed in 1980s used in small/medium networks, RIP is a distance vector routing protocol which use hop count as a routing metric, rip is able to route information across the network up to 15 hops. RIP support IPv6 in his newest version (RIPv2).

RIPv1 is extremely vulnerable to serious attack.

Attacks

The basic idea is to keep issuing fake RIP Response messages, containing basically whatever we need (in terms of routes). The other routers will eventually insert our malicious entries in they tables and start routing the packets accordingly.

Normally, RIP Response messages are sent to the dedicated multicast address 224.0.0.9, but the same messages seem to be accepted even when their destination IP address is a specific router. This is great, from an attacker point of view, because it allows he or she to be surgical in the injection process.

Other attacks

1. Forging RIP messages
   Spoofing source address and sending invalid routes, altering traffic flow.
   - Traffic Hijacking
   - Traffic Monitoring
   - Redirecting traffic from trusted to untrusted.
2. Obtaining Cleartext RIPv2 "password" when sent across network.
   Using retrieved password to send authenticated updates to RIPv2 routers, altering traffic flow with consequences listed above.

Defenses

- Disabling RIPv1 and using RIPv2 with MD5 authentication.
- Enabling MD5 based authentication for RIPv2
- Disabling RIP completely and using OSPF with MD5 authentication as interior gateway protocol. OSPF is the suggested IGP.

OSPF

Open shortest Path first developed in 1998 is a link state routing protocol that use metric as a routing metric, this protocol in enable to discover the network using identification messages (LSA).

There are three version of the protocol:

- OSPFv1: described in RFC 1131 (has never gone beyond the experimental phase, as far as I know);
- OSPFv2: described in RFC 2328, it superseeds v1 and it is the version deployed worldwide for dealing with IPv4 networks;
- OSPFv3: it supports IPv6 and is described in RFC 6340.

Attacks

There are various attacks known against OSPF (see on Google Scholar), but they are normally able to “only” generate a DoS condition.

In 2011, Alex Kirshon, Dima Gonikman, Dr. Gabi Nakibly demonstrated an attack, during the Black Hat USA, that was able to circumvent the fightback packets.

The idea behind the attack is quite simple.

Imagine two neighbours routers, A and B. The attacker generates two packets, one sent to the victim router A to trigger the fightback and another sent, at the very same time, to the router B inside which the malicious routes have to be injected. The fightback packet, sent by A is received by B after the malicious one and it is discarded because seen as identical, even though the content of the two packets is different.

This is possible because two OSPF LSA (Link State Advertisement) packets are considered identical if they have:

- the same sequence number;
- the same 16 bits checksum value;
- approximately the same age (within a 15 minutes time difference).

The sequence number and the age are quite easy to spot and fake, while the checksum has to be calculated. Luckily, it can be predicted, because it is calculated on LSA fields that have values that can be inferred in advance.

Other attack

Forging OSPF messages : Can be somewhat difficult but theoretically possible if no authentication required or cleartext password obtained.

Solution

Enable MD5 Authentication in OSPF implementation.

BGP and EGP

Border Gateway Protocol is the internet standard External Gateway
Protocol (EGP) was designed to exchange information and routing updates between different autonomous system, BGP is a Path vector Protocol and it makes routing decisions based on path, network policies, or rule-sets.

Attacks: BGP Hijacking

BGP hijacking is an illicit process of taking control of a group of IP prefixes assigned to a potential victim. Either intentionally or accidentally, it is achieved by changing paths used for forwarding network traffic, exploiting the weaknesses of BGP. The aim of this blog post is to explore these weaknesses and to discuss possible countermeasures.

- Partial BGP Hijacking
  A partial BGP hijacking occurs when two origin Autonomous Systems announce an identical IP prefix with the same prefix length. The BGP best path selection rules, such as preferring the shortest AS path, determine which path is the best.

- Complete BGP HIjacking
  The complete BGP hijacking occurs when an attacker announces de-aggregated thus a more specific IP prefix than the actual owner of the prefix.

Filtering IP prefixes on Tier 3 ISP and customer side, however, can reduce occurrence of BGP hijacks.

Defenses

Limit AS_PATH in Announced Prefixes
We can limit the AS_PATH in announced prefixes using BGP AS path filter. The regular expression ^$ in ACL statement matches empty AS_PATH thus it allows only locally announced prefixes being sent to ISP.

ip as-path access-list 1 permit ^$

router bgp 64502
neighbor 200.1.1.1 filter-list 1 out

The lines above are applied on the customer router (AS64502) towards ISP (AS64500) BGP peer address 200.1.1.1. The AS64502 is added to ASP_PATH after the filter is applied. The configuration prevents customer AS64502 to become transit AS in case of a multihomed connection. As a result, traffic sent from another ASs is not routed through customer but uses a high-speed link of upstream providers instead.

The ISP can also configure the AS_PATH filter towards customer BGP router 200.1.1.2.

ip as-path access-list 1 permit ^64502$

router bgp 64500
neighbor 200.1.1.2 filter-list 1 in

Announce Only Owned Prefixes
Now we create a prefix-list on a customer router that permits the announcement of only the assigned prefix 199.1.1.0/24. The list is applied toward ISP router. All other prefixes are not being sent.

ip prefix-list filter_out seq 10 permit 199.1.1.0/24

router bgp 64502
neighbor 200.1.1.1 prefix-list filter_out out

The ISP should only accept prefixes which have been assigned or allocated to its customer.

ip prefix-list as64502in seq 10 permit 199.1.1.0/24

router bgp 64500
neighbor 200.1.1.2 prefix-list as64502in in

Filter Own Prefixes and Accept only Prefixes with Length /24 and Less
Customers do not need to know about the path to their own prefixes so they should filter them. However, filtering the single prefix 199.1.1.0/24 is not sufficient. If someone announces customer prefix with the longer prefix length than /24 it would be installed into customer routing table. For this reason, we need to specify the prefix length to 32. The sequence 10 denies customer prefix 199.1.1.0 within the length from 24 to 32. The sequence 20 accept only prefixes that are not denied by a rule 10 and their prefix length is /24 and less.

ip prefix-list filter_in seq 10 deny 199.1.1.0/24 le 32
ip prefix-list filter_in seq 20 permit 0.0.0.0/0 le 24router bgp 64502
neighbor 200.1.1.1 prefix-list filter_in in

Filter Default Route
Unless customers do not need a default route they should block it. Sequence 10 denies a default prefix. All other routes are matched and permitted by sequence 20.

ip prefix-list filter_defaultin seq 10 deny 0.0.0.0/0
ip prefix-list filter_defaultin seq 20 permit 0.0.0.0/0 le 32router bgp 64502

Detecting BGP Hijacking
When the IP prefixes are hijacked, connection might be redirected and discarded as in the Pakistan Telecom incident. In this case detection of hijacking is an easy task since a service becomes unavailable. As for the BGP Man in the Middle attacks, when data might be intercepted or modified, detection is not so straightforward because the connection is working. BGP hijacking, however, can still be detected since the BGP AS_PATH attribute gets changed. Moreover, network traffic takes different (not optimal) path which leads to degraded performance and the increased round-trip time (RTT). Providers' Looking Glass (LG) servers or Route Views are great tools to discover a change in the routing paths.

Conclusion

Thus the routing protocol and the possible attacks on them are studied. Also their defence mechanisms are understood.

EXPERIMENT 6

Ameya Daddikar - 161070015

AIM:

To implement a firewall using GUFW.

THEORY:

Firewalls are programs used to provide security between an internal and an external network (which may be the Internet). They monitor incoming and outgoing traffic and decide to allow or reject traffic depending upon policies, learnt patterns, heuristics etc. In its simplest implementation, a firewall will have a set of rules determining whether particular traffic should be allowed through based on criteria like protocol, port and source/destination IP address.

The various firewalls are:

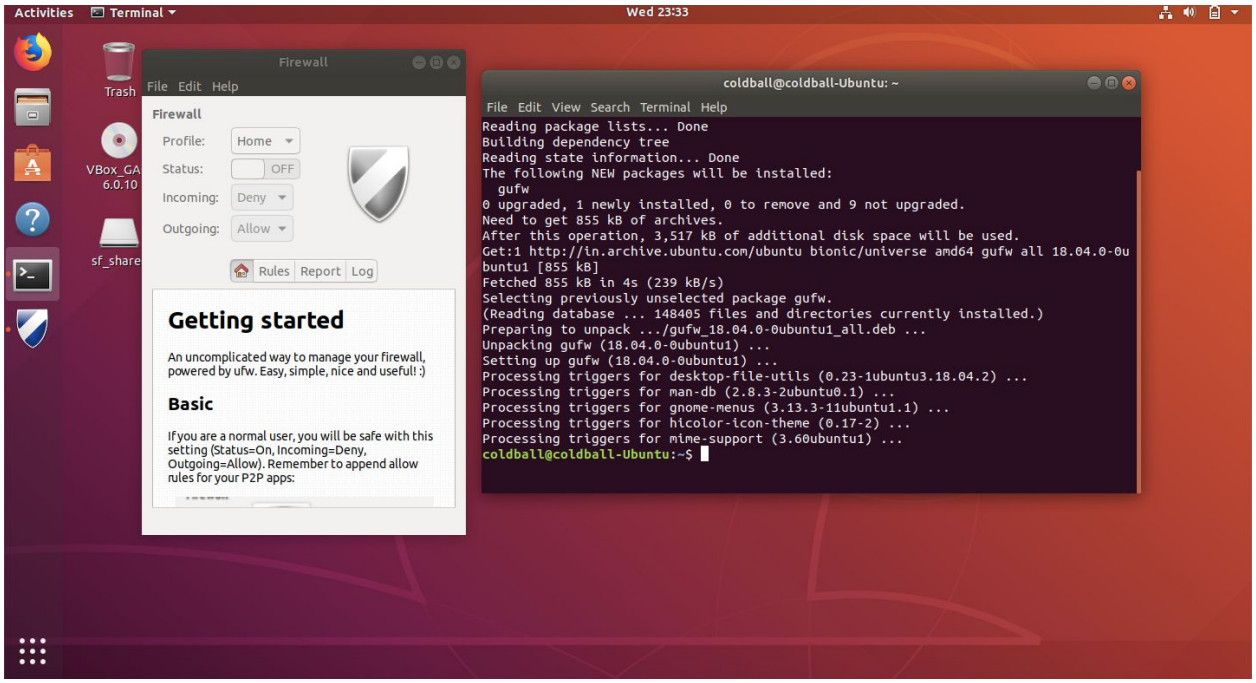
- Packet filters:
  They act by simply inspecting packets. If the packets follow the rules, they are allowed in or out. This is the simplest type, and does not provide protection against unknown vulnerabilities without default deny, in which case availability may be reduced.
- Circuit-level gateways
  It monitors TCP handshaking among packets across trusted clients or servers and untrusted hosts. It thus determines whether a requested session is legitimate. A trusted client requests a service, and the gateway accepts this request. On behalf of the client, the gateway opens a connection to the requested untrusted host and then closely monitors the TCP handshaking that follows. It works at the session layer of the OSI model, transport layer of TCP/IP
- Application gateways:
  These work at the application layer. They can understand the packets coming in and decide if an unwanted service is attempting to break in.

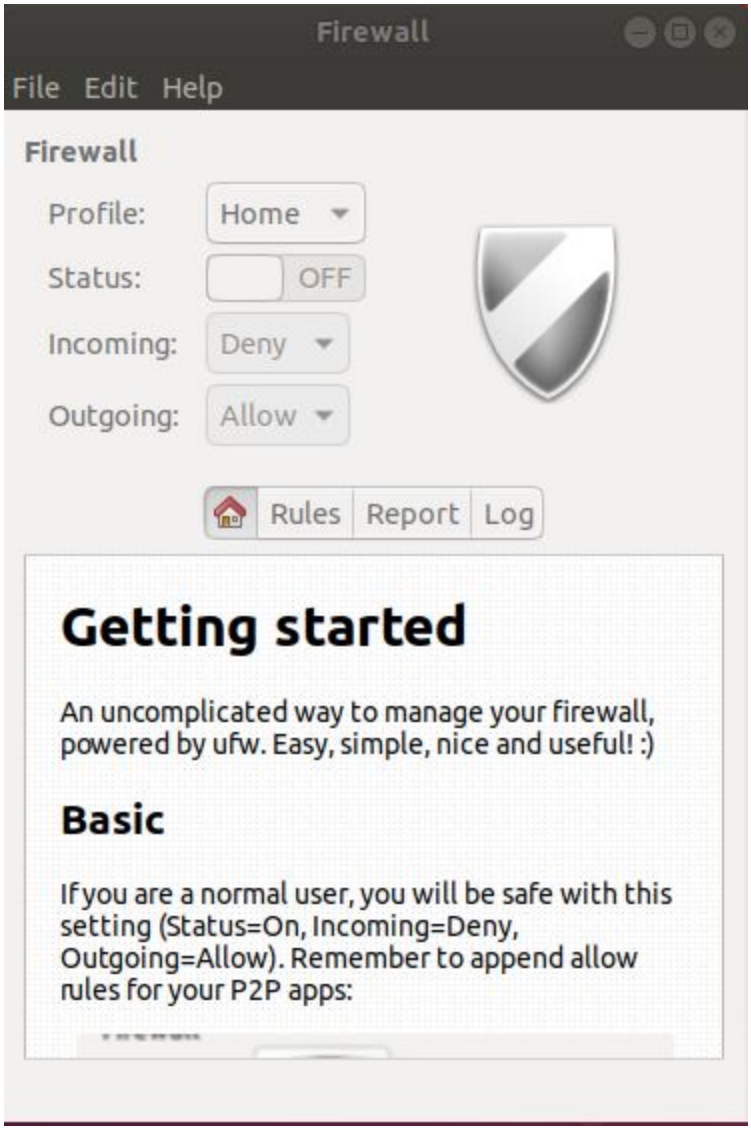
PRACTICAL:

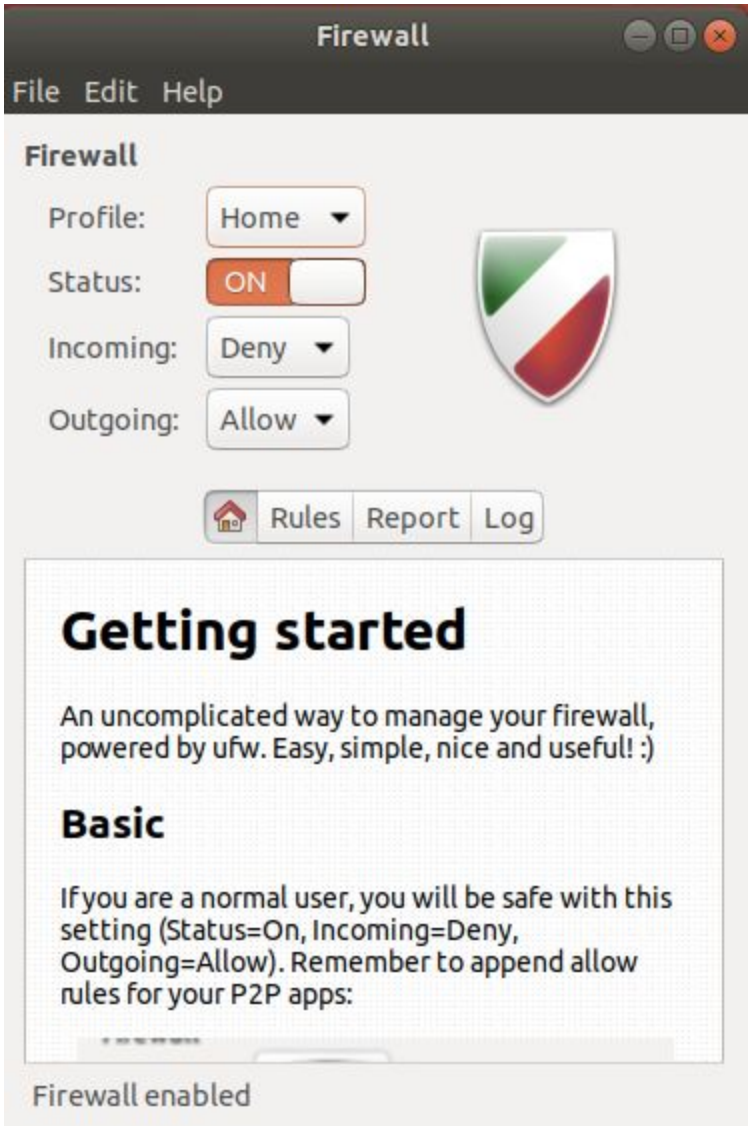
GUFW is a graphical utility for managing Uncomplicated Firewall (**UFW**), which is a frontend to iptables, the basic framework for firewalls.

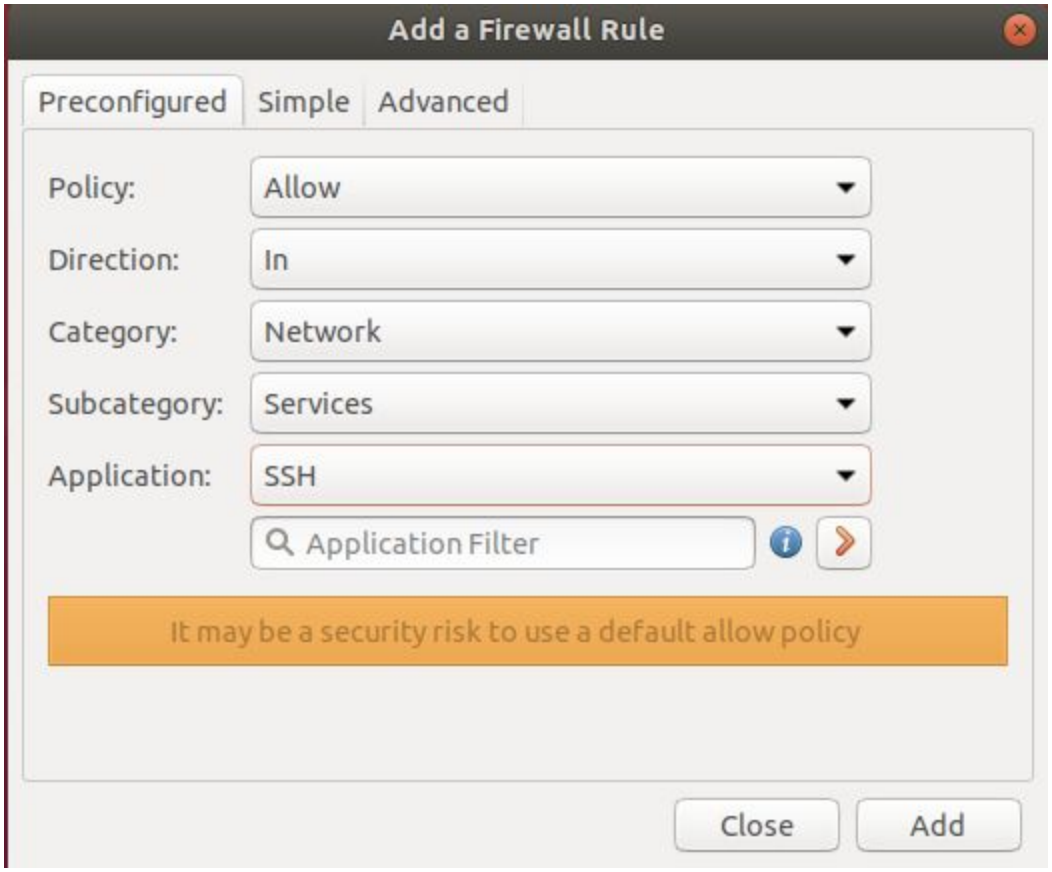
- Install with
  sudo apt install gufw -y
- Turn Status to On
- There are three profiles:
  - Home
  - Public
  - Office

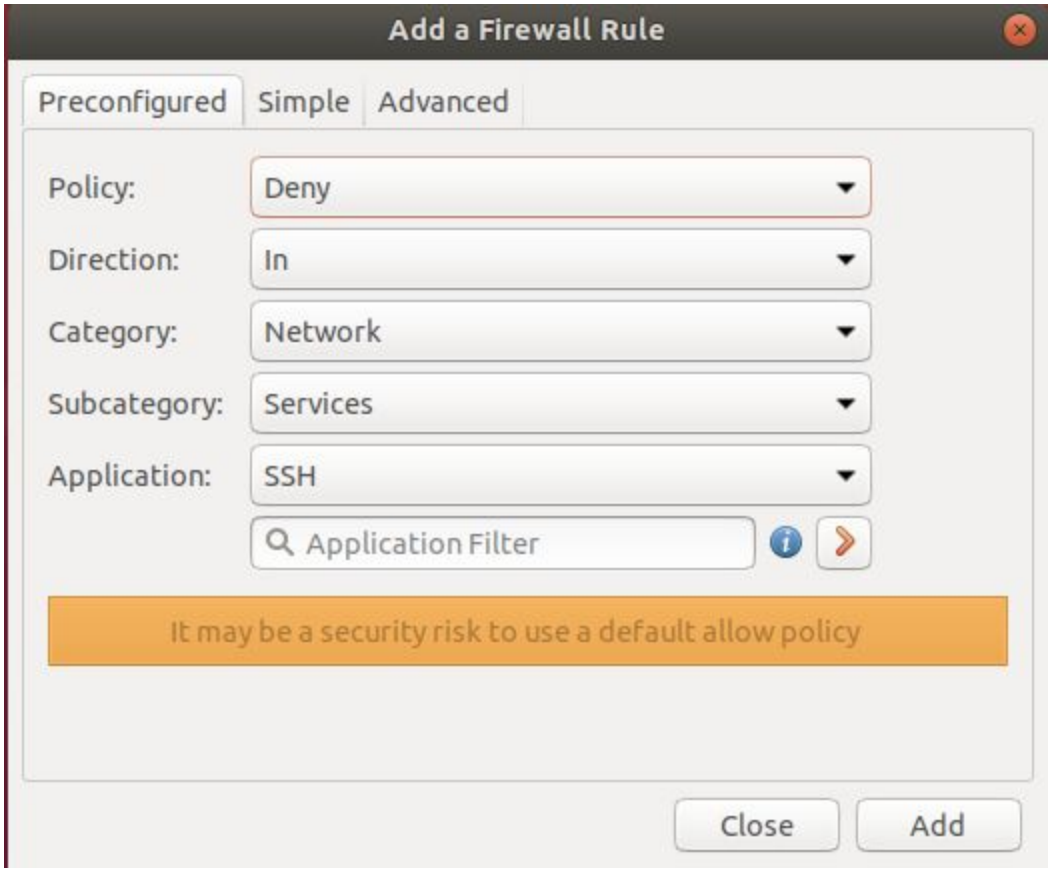
- Select Rules tab
- Select + to add
- Allow, Deny, Reject and Limit are the policies.
    - **Allow:** allows any entry traffic to a port
    - **Deny:** denies any entry traffic to a port
    - **Reject:** denies any entry traffic to a port and informs the requester about the rejection
    - **Limit:** denies entry traffic if an IP address has attempted to initiate 6 or more connections in the last 30 seconds
- Direction may be incoming/outgoing
- Category, Subcategory and application select the scope of the rule
- Finally, press add




Firewall
File Edit Help
Firewall
Profile:
Home
Status:
OFF
Incoming:
Deny
Outgoing:
Allow
Rules
Report
Log
Getting started
An uncomplicated way to manage your firewall, powered by ufw. Easy, simple, nice and useful! :)
Basic
If you are a normal user, you will be safe with this setting (Status=On, Incoming=Deny, Outgoing=Allow). Remember to append allow rules for your P2P apps:


Firewall
File Edit Help
Firewall
Profile:
Home
Status:
ON
Incoming:
Deny
Outgoing:
Allow
Rules
Report
Log
Getting started
An uncomplicated way to manage your firewall, powered by ufw. Easy, simple, nice and useful! :)
Basic
If you are a normal user, you will be safe with this setting (Status=On, Incoming=Deny, Outgoing=Allow). Remember to append allow rules for your P2P apps:
Firewall enabled

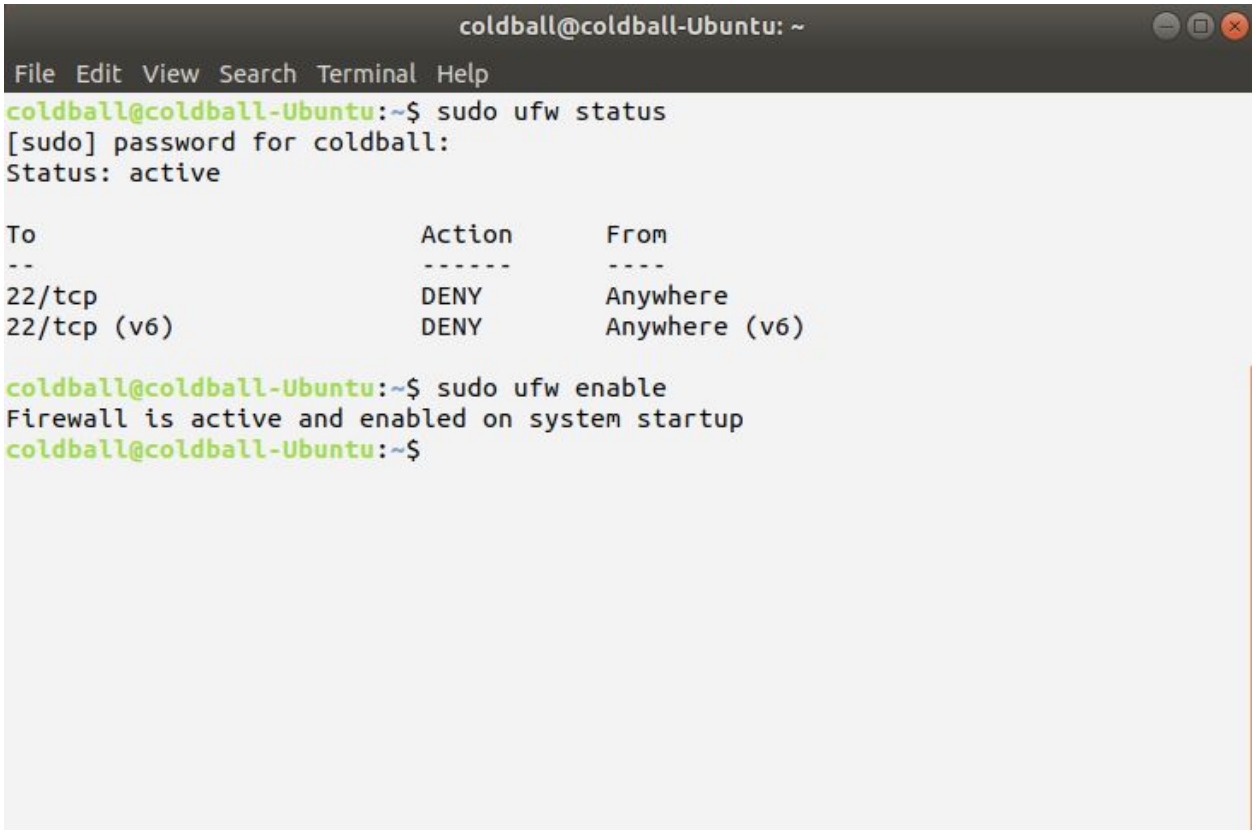

Add a Firewall Rule
Preconfigured
Simple
Advanced
Policy:
Allow
Direction:
In
Category:
Network
Subcategory:
Services
Application:
SSH
Application Filter
It may be a security risk to use a default allow policy
Close
Add

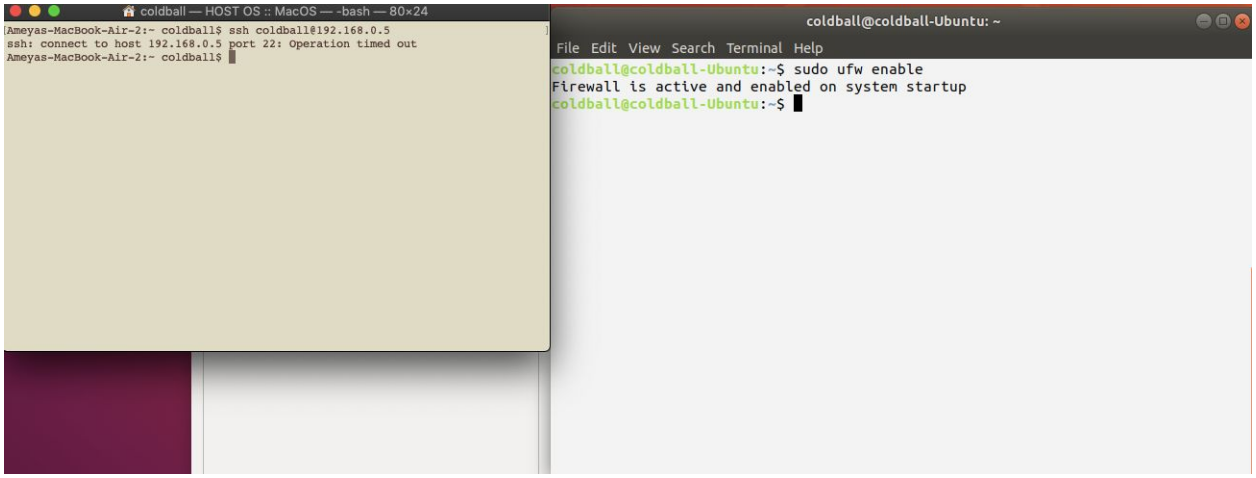


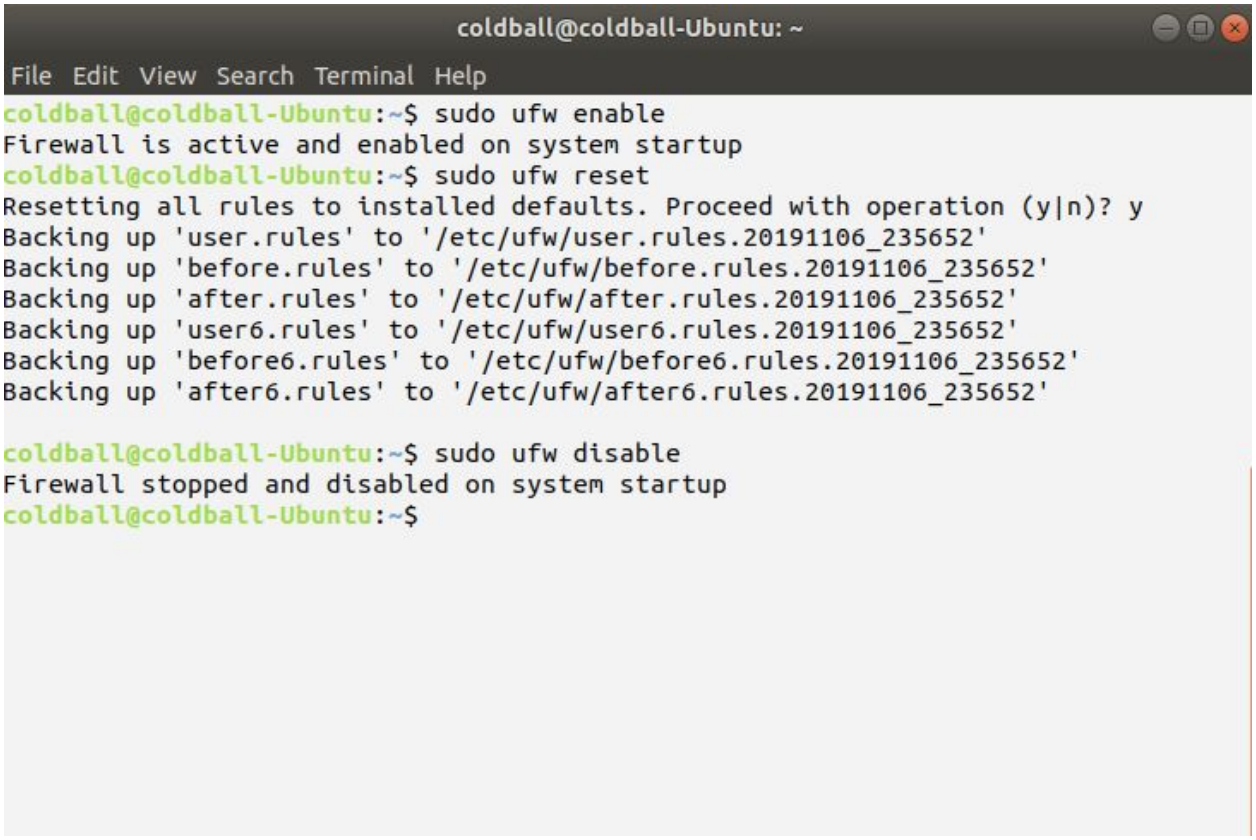

```
coldball@coldball-Ubuntu: ~
File  Edit  View  Search  Terminal  Help
coldball@coldball-Ubuntu:~$ sudo ufw status
[sudo] password for coldball:
Status: active

To                         Action      From
--                         ------      ----
22/tcp                     DENY        Anywhere
22/tcp (v6)                DENY        Anywhere (v6)

coldball@coldball-Ubuntu:~$
```


coldball@coldball-Ubuntu: ~
File Edit View Search Terminal Help
coldball@coldball-Ubuntu:~$ sudo ufw status
[sudo] password for coldball:
Status: active
To Action From
-- ------ ----
22/tcp DENY Anywhere
22/tcp (v6) DENY Anywhere (v6)
coldball@coldball-Ubuntu:~$ sudo ufw enable
Firewall is active and enabled on system startup
coldball@coldball-Ubuntu:~$


coldball — HOST OS :: MacOS — -bash — 80×24
Ameyas-MacBook-Air-2:~ coldball$ ssh coldball@192.168.0.5
ssh: connect to host 192.168.0.5 port 22: Operation timed out
Ameyas-MacBook-Air-2:~ coldball$
coldball@coldball-Ubuntu: ~
File Edit View Search Terminal Help
coldball@coldball-Ubuntu:~$ sudo ufw enable
Firewall is active and enabled on system startup
coldball@coldball-Ubuntu:~$



CONCLUSION:

Hence, a firewall is implemented in GUFW

Experiment 7

Name	Ameya S. Daddikar
College I.D.	161070015
Course	Btech. Computer Engineering

Aim

To study and implement an IDS (Intrusion Detection System) using an open source tool (SNORT).

Theory

Intrusion Detection System

An Intrusion Detection System (IDS) is a device, typically another separate computer, that monitors activity to identify malicious or suspicious events.

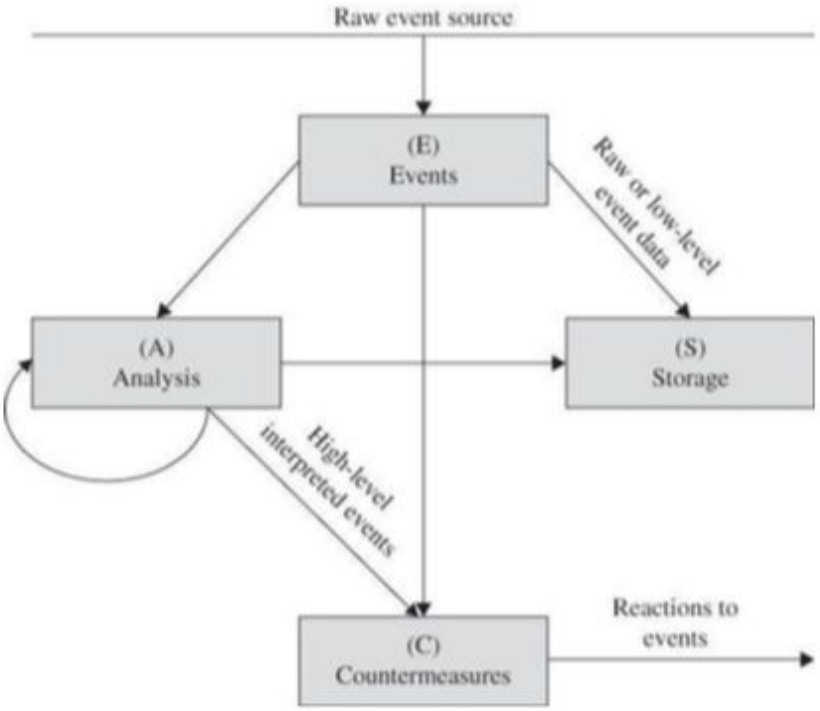


FIGURE 6-64 Model of an Intrusion Detection System

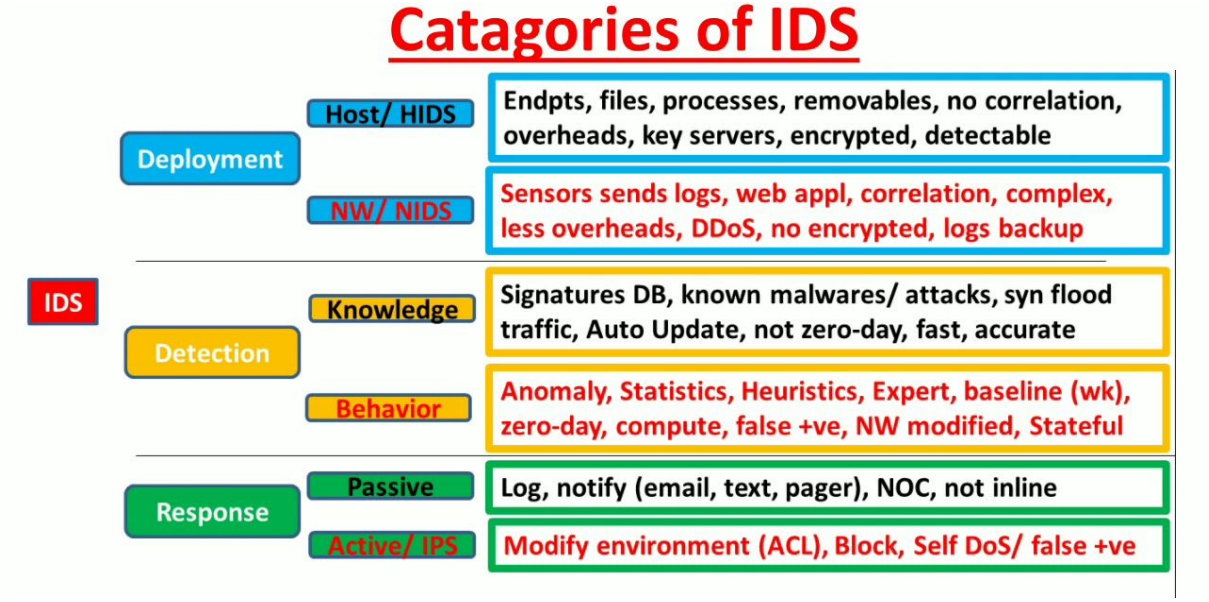
The components in the figure above are the four basic elements of an intrusion detection system, based on the Common Intrusion Detection Framework. An IDS receives raw

inputs from sensors. It saves those inputs, analyzes them, and takes some controlling action.

IDSs perform a variety of functions:

- Monitoring users and system activity
- Auditing system configuration for vulnerabilities and misconfigurations
- Assessing the integrity of critical system and data files
- Recognizing known attack patterns in system activity
- Identifying abnormal activity through statistical analysis
- Managing audit trails and highlighting user violation of policy or normal activity
- Correcting system configuration errors
- Installing and operating traps to record information about intruders

Categories of IDS



SNORT

Snort is an open source network intrusion prevention system, capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis, content searching/matching, and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more.

Snort has three primary uses:

1. A straight packet sniffer like tcpdump
2. A packet logger (useful for network traffic debugging, etc)
3. A full blown network intrusion prevention system.

Snort Rules

Rules are a different methodology for performing detection, which bring the advantage of 0-day detection to the table. Unlike signatures, rules are based on detecting the actual vulnerability, not an exploit or a unique piece of data. Developing a rule requires an acute understanding of how the vulnerability actually works.

Community rules refer to all rules that have been submitted by members of the open source community or Snort Integrators. These rules are freely available to all Snort users and are governed by the GPLv2.

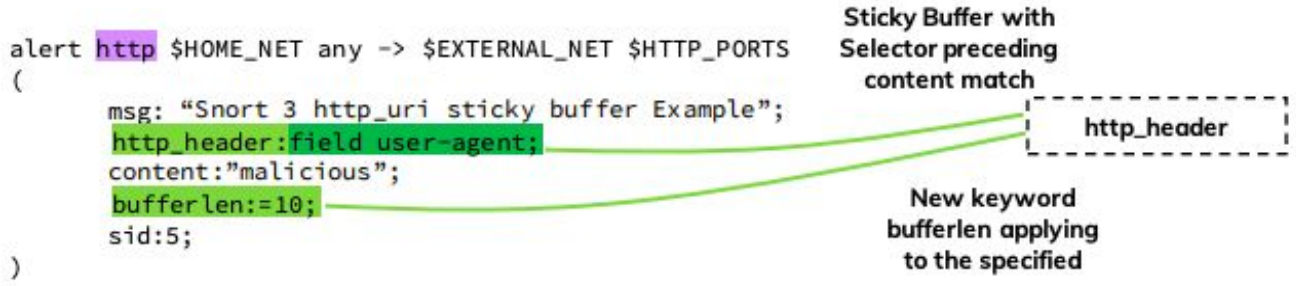
Rule Headers

The rule header follows a specific format:

Action Protocol Networks Ports Direction Operator Networks Ports

Examples:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (RULE_OPTIONS)
alert udp $EXTERNAL_NET $FILE_DATA_PORTS -> $HOME_NET any (RULE_OPTIONS)
```



Output

my.rules

snort.conf

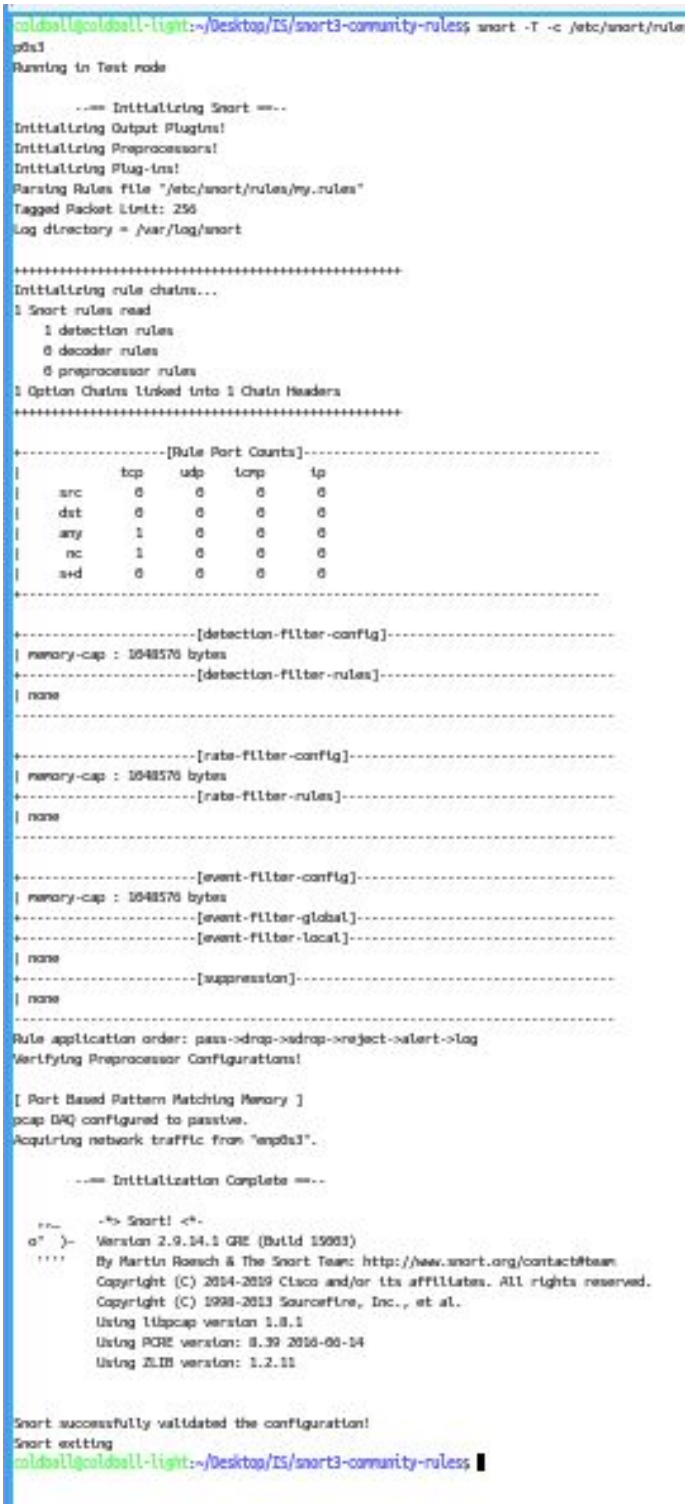
```
# Setup the network addresses you are protecting
ipvar HOME_NET 10.0.2.0/24

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET any

# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules

# If you are using reputation preprocessor set these
# Currently there is a bug with relative paths, they are relative to where snort is
# not relative to snort.conf like the above variables
# This is completely inconsistent with how other vars work, BUG 89986
# Set the absolute path appropriately
var WHITE_LIST_PATH /etc/snort/rules/iplists
var BLACK_LIST_PATH /etc/snort/rules/iplists
```

Testing SNORT rules file my.rules



Conclusion

Thus we deployed SNORT on the VM's network to listen for a simple set of SNORT rules and alert any matches on the console.

Experiment 8

Name	Ameya S. Daddikar
College I.D.	161070015
Course	Btech. Computer Engineering

Aim

Create the environment for the database application. Perform database administration and performance of database Application.(grant, revoke).

Theory

Oracle Database (Oracle XE)

Oracle XE (eXpress Edition) is Oracle Corporation's free to use and distribute database edition. XE is available for Windows and Linux and can be downloaded free of charge from Oracle TechNet. Linux RPM's are also available for easy deployment on Linux servers.

Features

Multitenant: Get isolation, agility, and economies of scale by managing multiple Pluggable Databases inside your Oracle Multitenant Container Database

In-Memory: Support real-time analytics, business intelligence, and reports by keeping your important data in the Oracle Database In-Memory column store

Partitioning: Enhance performance, availability, and manageability of your database with data partitioning that meets diverse business requirements

Advanced Analytics: Get valuable insights and deliver predictions from your data using Data Mining SQL, R programming, and the Oracle Data Miner UI

Advanced Security: Protect your sensitive data at the source and build end-to-end encrypted apps with layers of security including Oracle Transparent Data Encryption and Data Redaction.

Required Resources

- Up to 12 GB of user data
- Up to 2 GB of database RAM
- Up to 2 CPU threads
- Up to 3 Pluggable Databases

User Privileges and Roles

A user privilege is a right to execute a particular type of SQL statement, or a right to access another user's object. The types of privileges are defined by Oracle.

Roles, on the other hand, are created by users (usually administrators) and are used to group together privileges or other roles. They are a means of facilitating the granting of multiple privileges or roles to users.

User Roles

A role groups several privileges and roles, so that they can be granted to and revoked from users simultaneously. A role must be enabled for a user before it can be used by the user.

Oracle provides some predefined roles to help in database administration. These roles, listed in the table below, are automatically defined for Oracle databases when you run the standard scripts that are part of database creation. You can grant privileges and roles to, and revoke privileges and roles from, these predefined roles in the same way as you do with any role you define.

Role Name	Created By (Script)	Description
CONNECT	SQL.BSQ	Includes the following system privileges: ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE VIEW
RESOURCE	SQL.BSQ	Includes the following system privileges: CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE
DBA	SQL.BSQ	All system privileges WITH ADMIN OPTION
Note: The previous three roles are provided to maintain compatibility with previous versions of Oracle and may not be created automatically in future versions of Oracle. Oracle Corporation recommends that you design your own roles for database security, rather than relying on these roles.		
EXP_FULL_DATABASE	CATEXP.SQL	Provides the privileges required to perform full and incremental database exports. Includes: SELECT ANY TABLE, BACKUP ANY TABLE, EXECUTE ANY PROCEDURE, EXECUTE ANY TYPE, ADMINISTER RESOURCE MANAGER, and INSERT, DELETE, and UPDATE on the tables SYS.INCVID, SYS.INCFIL, and SYS.INCEXP. Also the following roles: EXECUTE_CATALOG_ROLE and SELECT_CATALOG_ROLE.
IMP_FULL_DATABASE	CATEXP.SQL	Provides the privileges required to perform full database imports. Includes an extensive list of system privileges (use view DBA_SYS_PRIVS to view privileges) and the following roles: EXECUTE_CATALOG_ROLE and SELECT_CATALOG_ROLE.
DELETE_CATALOG_ROLE	SQL.BSQ	Provides DELETE privilege on the system audit table (AUD$)
EXECUTE_CATALOG_ROLE	SQL.BSQ	Provides EXECUTE privilege on objects in the data dictionary. Also, HS_ADMIN_ROLE.
SELECT_CATALOG_ROLE	SQL.BSQ	Provides SELECT privilege on objects in the data dictionary. Also, HS_ADMIN_ROLE.
RECOVERY_CATALOG_OWNER	CATALOG.SQL	Provides privileges for owner of the recovery catalog. Includes: CREATE SESSION, ALTER SESSION, CREATE SYNONYM, CREATE VIEW, CREATE DATABASE LINK, CREATE TABLE, CREATE CLUSTER, CREATE SEQUENCE, CREATE TRIGGER, and CREATE PROCEDURE
HS_ADMIN_ROLE	CATHS.SQL	Used to protect access to the HS (Heterogeneous Services) data dictionary tables (grants SELECT) and packages (grants EXECUTE). It is granted to SELECT_CATALOG_ROLE and EXECUTE_CATALOG_ROLE such that users with generic data dictionary access also can access the HS data dictionary.
AQ_USER_ROLE	CATQUEUE.SQL	Obsoleted, but kept mainly for release 8.0 compatibility. Provides execute privilege on DBMS_AQ and DBMS_AQIN.
AQ_ADMINISTRATOR_ROLE	CATQUEUE.SQL	Provides privileges to administer Advance Queuing. Includes ENQUEUE ANY QUEUE, DEQUEUE ANY QUEUE, and MANAGE ANY QUEUE, SELECT privileges on AQ tables and EXECUTE privileges on AQ packages.
SNMPAGENT	CATSNMP.SQL	This role is used by Enterprise Manager/Intelligent Agent. Includes ANALYZE ANY and grants SELECT on various views.

Creating Roles

You can create a role using the CREATE ROLE statement, but you must have the CREATE ROLE system privilege to do so. Typically, only security administrators have this system privilege.

Immediately after creation, a role has no privileges associated with it. To associate privileges with a new role, you must grant privileges or other roles to the new role.

You must give each role you create a unique name among existing usernames and role names of the database. Roles are not contained in the schema of any user. In a database that uses a multibyte character set, Oracle recommends that each role name contain at

least one single-byte character. If a role name contains only multibyte characters, the encrypted role name/password combination is considerably less secure.

The following statement creates the clerk role, which is authorized by the database using the password bicentennial:

CREATE ROLE clerk IDENTIFIED BY bicentennial;

The IDENTIFIED BY clause specifies how the user must be authorized before the role can be enabled for use by a specific user to which it has been granted. If this clause is not specified, or NOT IDENTIFIED is specified, then no authorization is required when the role is enabled. Roles can be specified to be authorized by:

- The database using a password
- An application using a specified package
- Externally by the operating system, network, or other external source
- Globally by an enterprise directory service

Later, you can set or change the authorization method for a role using the ALTER ROLE statement. The following statement alters the clerk role to specify that the user must have been authorized by an external source before enabling the role:

ALTER ROLE clerk IDENTIFIED EXTERNALLY;

Dropping Roles

In some cases, it may be appropriate to drop a role from the database. The security domains of all users and roles granted a dropped role are immediately changed to reflect the absence of the dropped role's privileges. All indirectly granted roles of the dropped role are also removed from affected security domains. Dropping a role automatically removes the role from all users' default role lists.

Because the creation of objects is not dependent on the privileges received through a role, tables and other objects are not dropped when a role is dropped.

You can drop a role using the SQL statement DROP ROLE. To drop a role, you must have the DROP ANY ROLE system privilege or have been granted the role with the ADMIN OPTION.

The following statement drops the role CLERK:

DROP ROLE clerk;

Granting System Privileges and Roles

You can grant system privileges and roles to other users and roles using the GRANT statement. The following privileges are required:

To grant a system privilege, you must have been granted the system privilege with the ADMIN OPTION or have been granted the GRANT ANY PRIVILEGE system privilege.

To grant a role, you must have been granted the role with the ADMIN OPTION or have been granted the GRANT ANY ROLE system privilege.

Note: You cannot grant a roll that is IDENTIFIED GLOBALLY to anything. The granting (and revoking) of global roles is controlled entirely by the enterprise directory service.

The following statement grants the system privilege CREATE SESSION and the accts_pay role to the user jward:

GRANT CREATE SESSION, accts_pay TO jward;

A user or role that is granted a privilege or role specifying the WITH ADMIN OPTION clause has several expanded capabilities:

- The grantee can grant or revoke the system privilege or role to or from any user or other role in the database. Users cannot revoke a role from themselves.
- The grantee can further grant the system privilege or role with the ADMIN OPTION.
- The grantee of a role can alter or drop the role.

In the following statement, the security administrator grants the new_dba role to michael:

GRANT new_dba TO michael WITH ADMIN OPTION;

Oracle allows you to create a new user with the GRANT statement. If you specify a password using the IDENTIFIED BY clause, and the username/password does not exist in the database, a new user with that username and password is created. The following example creates ssmith as a new user while granting ssmith the CONNECT system privilege:

GRANT CONNECT TO ssmith IDENTIFIED BY p1q2r3;

Revoking User Privileges and Roles

You can revoke system privileges and roles using the SQL statement REVOKE.

Any user with the ADMIN OPTION for a system privilege or role can revoke the privilege or role from any other database user or role. The revoker does not have to be the user that originally granted the privilege or role. Users with GRANT ANY ROLE can revoke any role.

The following statement revokes the CREATE TABLE system privilege and the accts_rec role from tsmith:

REVOKE CREATE TABLE, accts_rec FROM tsmith;

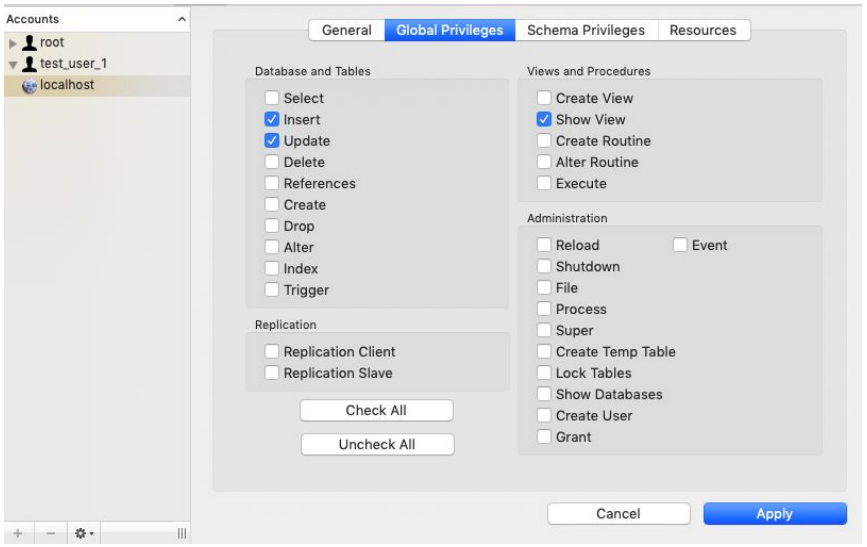
Depending on what is granted or revoked, a grant or revoke takes effect at different times:

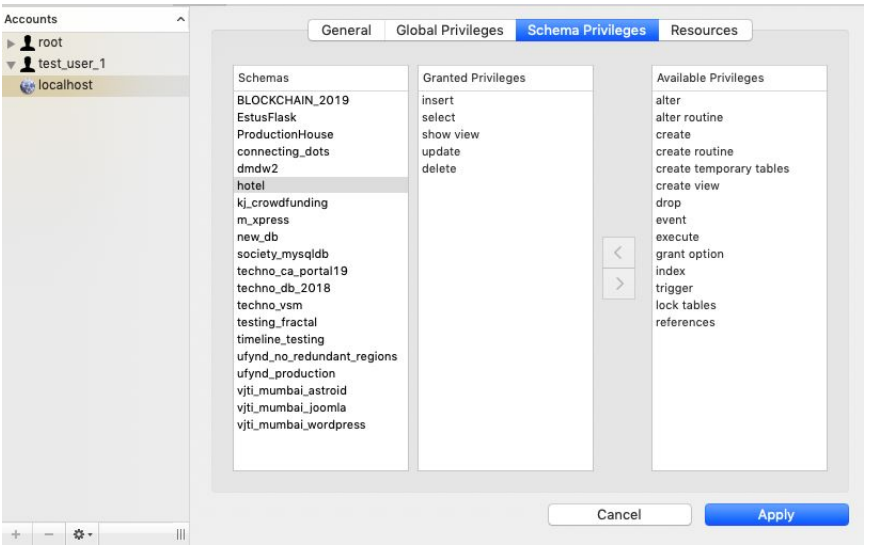
When Do Grants and Revokes Take Effect?

All grants/revokes of system and object privileges to anything (users, roles, and PUBLIC) are immediately observed.

- All grants/revokes of roles to anything (users, other roles, PUBLIC) are only observed when a current user session issues a SET ROLE statement to re-enable the role after the grant/revoke, or when a new user session is created after the grant/revoke.
- You can see which roles are currently enabled by examining the SESSION_ROLES data dictionary view.

Output





Conclusion

Thus, we reviewed the Privileges and Roles in an Oracle Database environment.

Experiment 9

Name	Ameya S. Daddikar
College I.D.	161070015
Course	Btech. Computer Engineering

Aim

To study the hardening of Linux OS.

Theory

Wireshark

Most systems have confidential data that needs to be protected. To safeguard this data, we need to secure our Linux system. But how to properly harden a Linux system? To do this, we start by with physical security measures to prevent unauthorized people from access the system in the first place. Next is doing the installation the right way, so we have a solid foundation. Finally, we will apply a set of common security measures. After we are finished, our server or desktop system should be better protected.

It is believed, Linux is already secure by default.

One of the myths about Linux is that it is secure, as it is not susceptible to viruses or other forms of malware. This is partially true, as Linux uses the foundations of the original UNIX operating system. Processes are separated and a normal user is restricted in what he or she can do on the system. Still, Linux is not perfectly secure by default. One of the reasons is the Linux distributions that package the GNU/Linux kernel and the related software. They have to choose between usability, performance, and security.

With the difficult choices that Linux distributions have to make, we can be sure of compromises. These compromises typically result in a lower level of security. What about malware for Linux? That is definitely a myth. The Linux platform also has its fair share of backdoors, rootkits, works, and even ransomware. That is one of the reasons why it is important to do system hardening, security auditing, and checking for compliance with technical guidelines.

Areas	Core	Resources	Services	Environment
System Hardening	Boot Process Containers Frameworks **Kernel** Service Manager Virtualization	Accounting **Authentication** Cryptography Logging Network Software Storage Time	Database Mail Middleware Monitoring Printing **Shell** **Web**	Forensics Incident Response Malware Risks Security Monitoring System Integrity
Security Auditing				
Compliance				

What is System Hardening?

To improve the security level of a system, we take different types of measures. This could be the removal of an existing system service or uninstall some software components.

System hardening is the process of doing the 'right' things. The goal is to enhance the security level of the system. There are many aspects to securing a system properly. Yet, the basics are similar for most operating systems. So the system hardening process for Linux desktop and servers is that that special.

Core principles of System Hardening

If we would put a microscope on system hardening, we could split the process into a few core principles. These include the principle of least privilege, segmentation, and reduction. Principle of least privilege

The principle of least privileges means that we give users and processes the bare minimum of permission to do their job. It is similar to granting a visitor access to a building. We could give full access to the building, including all sensitive areas. The other option is to only allow our guest to access a single floor where they need to be.

Examples:

- When read-only access is enough, don't give write permissions
- Don't allow executable code in memory areas that are flagged as data segments
- Don't run applications as the root user, instead use a non-privileged user account

Segmentation

The next principle is that we split bigger areas into smaller ones. If we look at that building again, we have split it into multiple floors. Each floor can be further divided into different zones. Maybe our visitor is only allowed on floor 4, in the blue zone. If we translate this to Linux security, this principle would apply to memory usage. Each process can only access their own memory segments.

Reduction

This principle aims to remove something that is not strictly needed for the system to work. It looks like the principle of least privilege, yet focuses on preventing something in the first place. A process that does not have to run, should be stopped. Similar for unneeded user accounts or sensitive data that is no longer being used.

System Hardening steps

Overview of hardening steps:

1. Install security updates and patches
2. Use strong passwords
3. Bind processes to localhost
4. Implement a firewall
5. Keep things clean
6. Security configurations
7. Limit access
8. Monitor our systems
9. Create backups (and test)
10. Perform system auditing

1. Install security updates and patches

Most weaknesses in systems are caused by flaws in software. These flaws we call vulnerabilities. Proper care for software patch management help with reducing a lot of the related risks. The activity of installing updates often has a low risk, especially when starting with the security patches first. Most Linux distributions have the option to limit what packages we want to upgrade (all, security only, per package). Make sure that our security
updates are installed as soon as they come available. It goes without saying, before we implement something, test it first on a (virtual) test system.
Depending on our Linux distribution there might be a way to implement security patches automatically, like unattended upgrades on Debian and Ubuntu. This makes software patch management a lot easier!

2. Use strong passwords

The main gateway to a system is by logging in as a valid user with the related password of that account. Strong passwords make it more difficult for tools to guess the password and let malicious people walk in via the front door. A strong password consists of a variety of characters (alphanumeric, numbers, special like percent, space, or even Unicode characters).

3. Bind processes to localhost

Not all services have to be available via the network. For example, when running a local instance of MySQL on our web server, let it only listen on a local socket or bind to localhost (127.0.0.1). Then configure our application to connect via this local address, which is typically already the default.

4. Implement a firewall

Only allowed traffic should in an ideal situation reach our system. To achieve this, implement a firewall solution like iptables, or the newer nftables.
When creating a policy for our firewall, consider using a “deny all, allow some” policy. So we deny all traffic by default, then define what kind of traffic we want to allow. This is especially useful for incoming traffic, to prevent sharing services we didn’t intend to share.

5. Keep things clean

Everything installed on a system which doesn’t belong there can only negatively impact our machine. It will also increase our backups (and restore times). Or they might contain vulnerabilities. A clean system is often a more healthy and secure system. Therefore minimization is a great method in the process of Linux hardening.
Actionable tasks include:
・Delete unused package
・Clean up old home directories and remove the users 6. Secure configurations
Most applications have one or more security measures available to protect against some forms of threats to the software or system. Look at the man page for any options and test these options carefully.

7. Limit access

Only allow access to the machine for authorized users. Does someone really need access or are alternative methods possible to give the user what he or she wants?

8. Monitor our systems

Most intrusions are undetected, due to lack of monitoring. Implement normal system monitoring and implement monitoring on security events. For example, the use of the Linux audit framework increased detection rates of suspected events.

9. Create backups (and test)

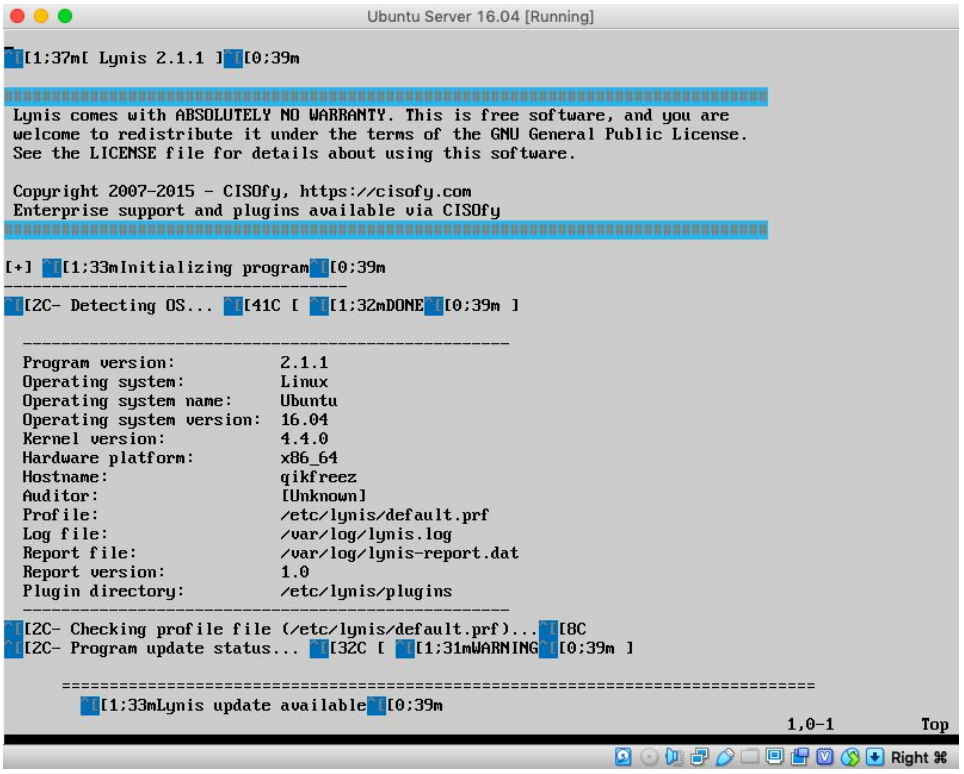
Regularly make a backup of system data. This can prevent data loss. Even more important, test our backups. Having a backup is nice, but it is the restore that really counts!
Backups can be done with existing system tools like tar and scp. Another option to spare bandwidth is synchronizing data with tools like rsync. If we rather want to use a backup program, we can consider Amanda or Bacula.

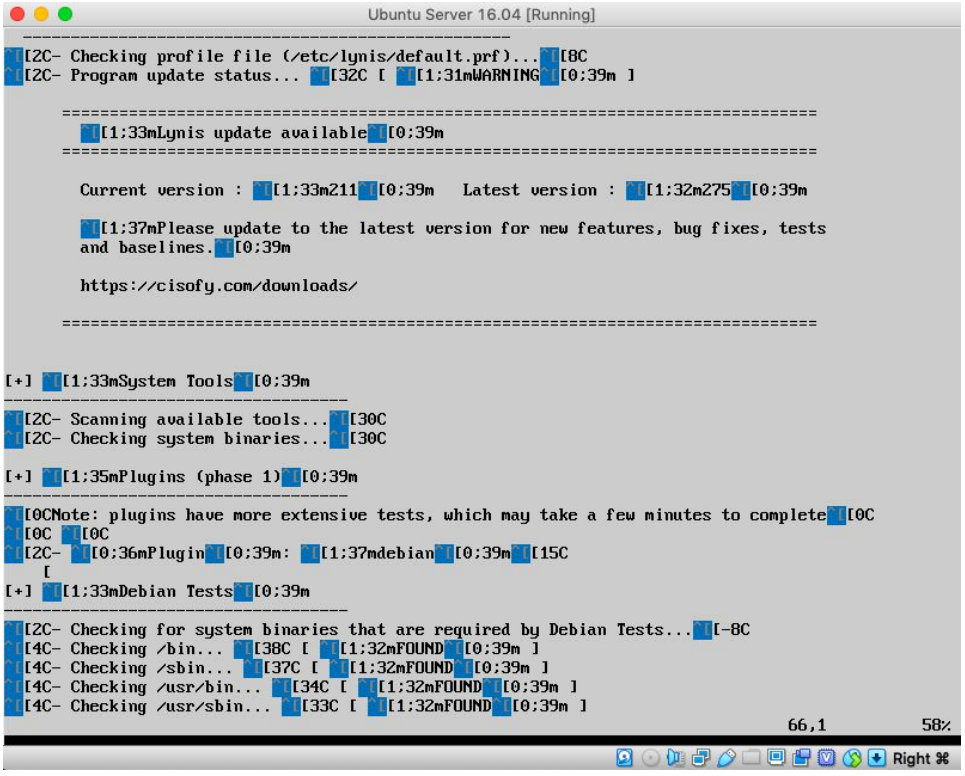
10. Perform system auditing

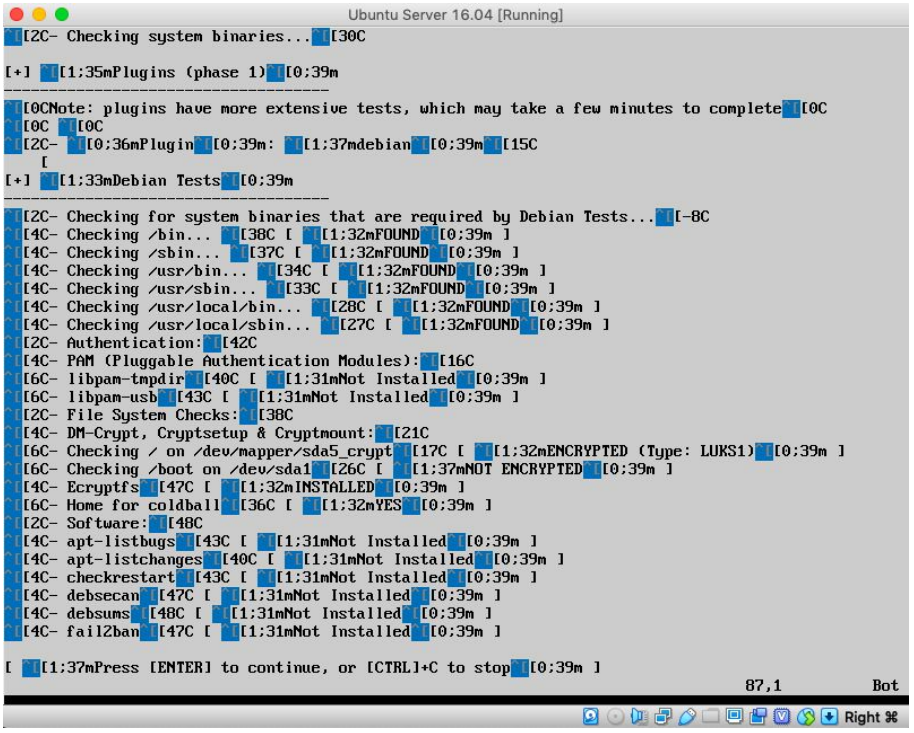
Use a security tool like Lynis to perform a regular audit of our system. Any findings are shown on the screen and also stored in a data file for further analysis. With an extensive log file, it allows to use all available data and plan next actions for further system hardening.
Lynis runs on almost all Linux systems or Unix flavors. It only requires a normal shell. Root permissions are preferred, yet not needed. The security tool is free to use and open source software (FOSS).

Console output of lynis --check-all

(performs a full test of the system, printing the result of each test to stdout)







Conclusion

Linux hardening is the process of doing the ‘right’ things for linux OS. The goal is to enhance the security level of the system. There are many aspects to securing a system properly. principle of least privilege, segmentation, and reduction are the core principles of hardening.

Experiment 10

Name	Ameya S. Daddikar
College I.D.	161070015
Course	Btech. Computer Engineering

Aim

To study and perform different types of DoS attacks on a website using PENTMENU.

Theory

Denial of Service (DoS)

The Denial of Service (DoS) attack is focused on making a resource (site, application, server) unavailable for the purpose it was designed. There are many ways to make a service unavailable for legitimate users by manipulating network packets, programming, logical, or resources handling vulnerabilities, among others. If a service receives a very large number of requests, it may cease to be available to legitimate users. In the same way, a service may stop if a programming vulnerability is exploited, or the way the service handles resources it uses.
Sometimes the attacker can inject and execute arbitrary code while performing a DoS attack in order to access critical information or execute commands on the server. Denial-of-service attacks significantly degrade the service quality experienced by legitimate users. These attacks introduce large response delays, excessive losses, and service interruptions, resulting in direct impact on availability.

Risk Factors

Risk factors can break down into multiple categories. Two principle sources of risk include inadequate resources and non-technical threat motivators.

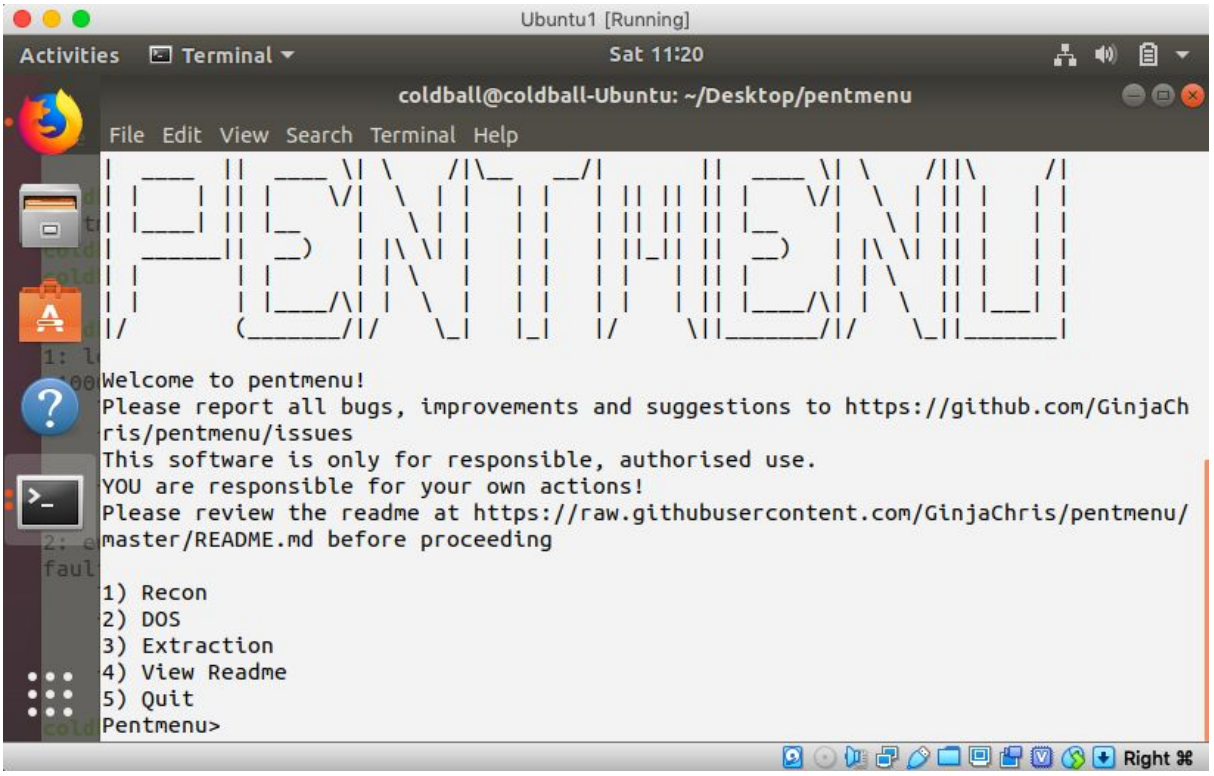
The first example of a risk factor, inadequate resources, requires attention if system architecture was not designed to meet traffic demand overflows. This risk reduces the difficulty of successfully executing a DoS attack and can, left unchecked, result in DoS symptoms absent an actual attack.

The second example and perhaps the largest risk factor is not technical and is in the domain of public relations or strategic communications. An organization should avoid taking action that can make them a target of a DoS attack unless the benefits of doing so outweigh the potential costs or mitigating controls are in place.

Other risk factors may also exist depending on the specific environment.

Pentmenu

Pentmenu is a bash select menu for quick and easy network recon and DOS attacks Sudo is implemented where necessary. Tested on Debian and Arch.



Requirements

- bash
- sudo
- curl
- netcat (must support '-k' option, openbsd variant recommended)
- hping3 (or nping can be used as a substitute for flood attacks)
- openssl
- stunnel
- nmap
- whois (not essential but preferred)
- nslookup (or 'host')
- ike-scan

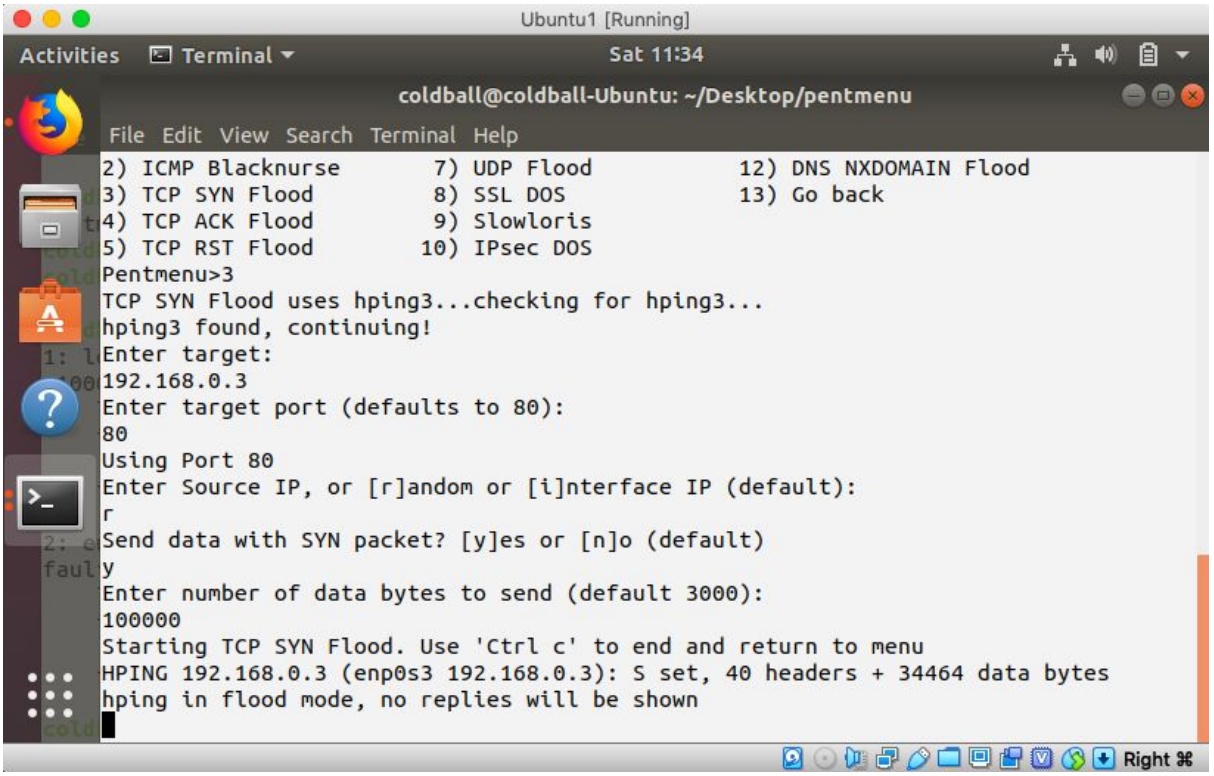
DOS Modules

TCP Syn Flood

TCP SYN Flood - sends a flood of TCP SYN packets using hping3. If hping3 is not found, it attempts to use the nmap-nping utility instead. Hping3 is preferred since it sends packets as fast as possible. Options are provided to use a source IP of your interface, or specify (spoof) a source IP, or spoof a random source IP for each packet. Optionally, you can add

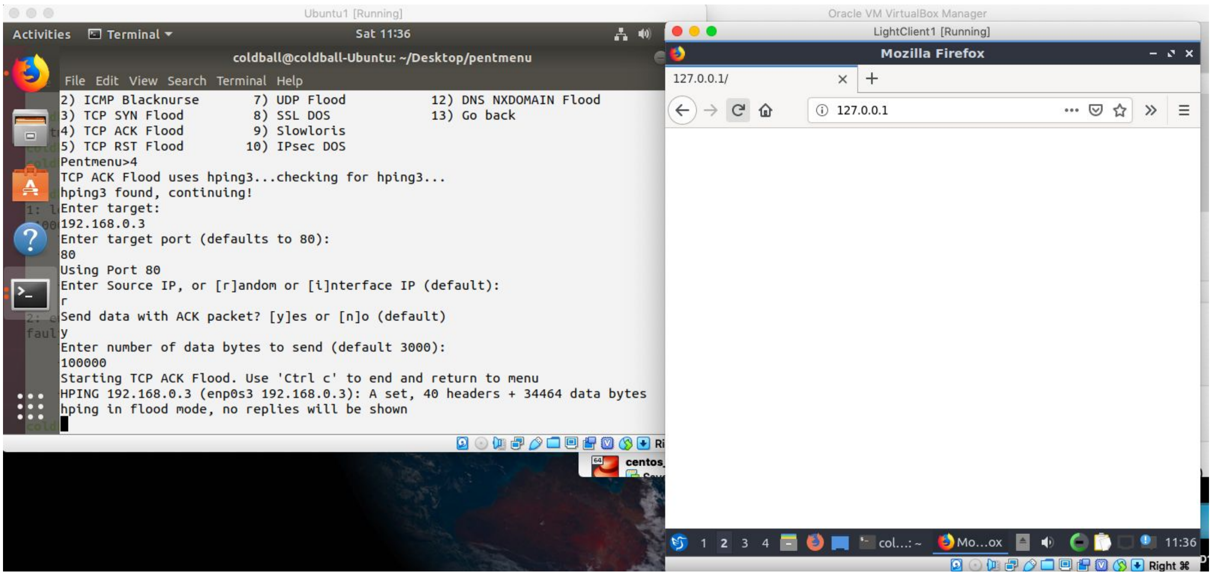
data to the SYN packet. All SYN packets have the fragmentation bit set and use hpings virtual MTU of 16 bytes, guaranteeing fragmentation. Falling back to nmap-nping means sending X number of packets per second until Y number of packets is sent and only allows the use of interface IP or a specified (spoofed) source IP.

A TCP SYN flood is unlikely to break a server, but is a good way to test switch/router/firewall infrastructure and state tables. Note that whilst hping will report the outbound interface and IP which might make you think script does not work as expected, the source IP will be set as specified; review a packet capture of the traffic if in doubt! Since the source is definable, it is simple to launch a LAND attack for example (see https://en.wikipedia.org/wiki/LAND). The ability to set the source also allows, for example, sending SYN packets to one target and forcing the SYN-ACK responses to a second target.



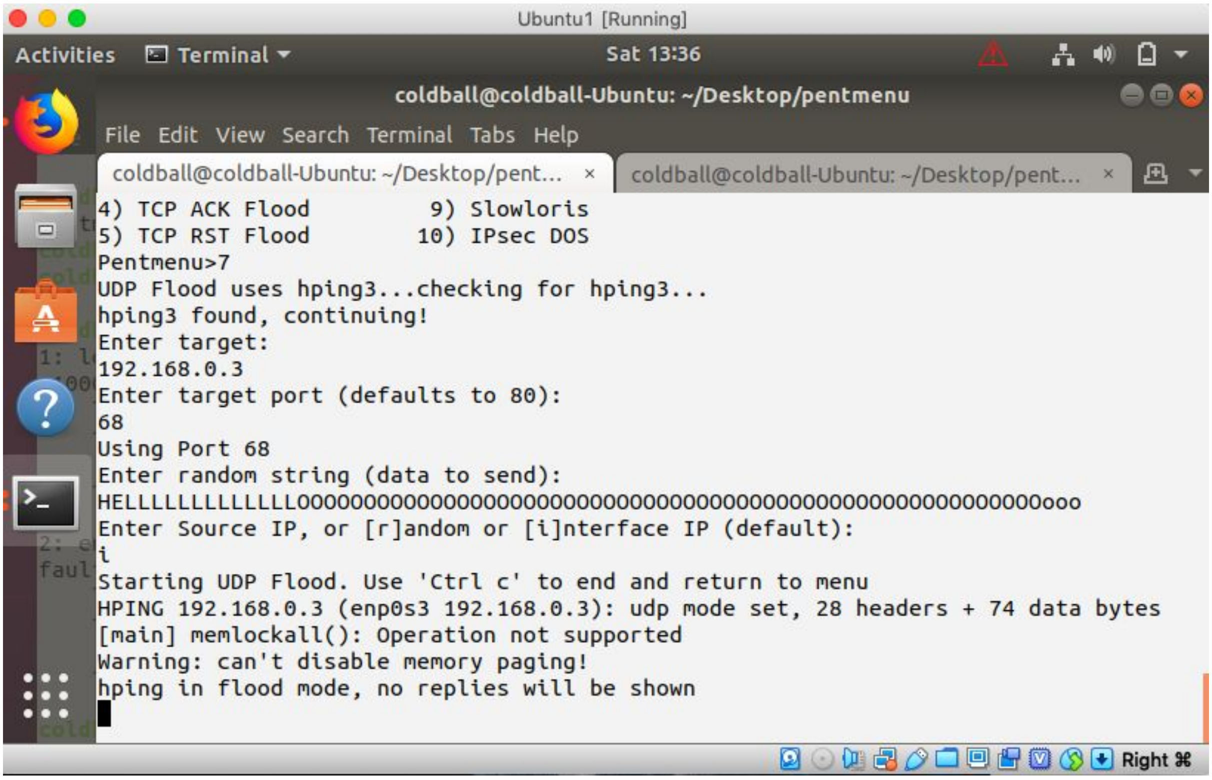
TCP ACK Flood

Offers the same options as the SYN flood, but sets the ACK (Acknowledgement) TCP flag instead. Some systems will spend excessive CPU cycles processing such packets. If the source IP is set to that of an established connection, it is possible that an established connection can be disrupted by this 'blind' TCP ACK Flood. This attack is considered 'blind' because it does not take into account any details of any established connection (like sequence or acknowledgement numbers).



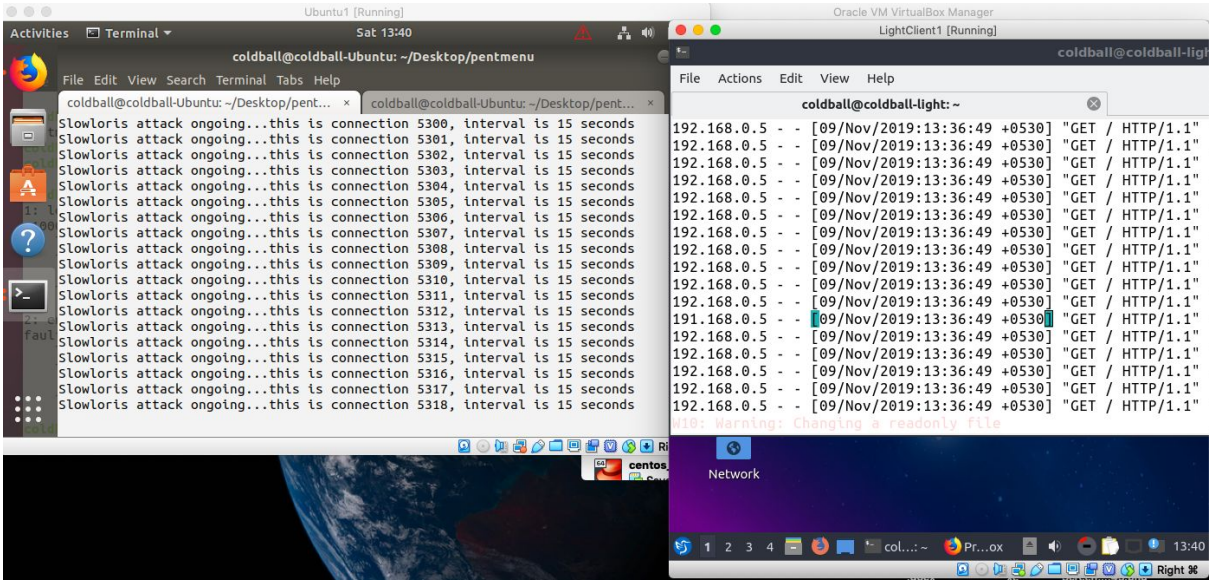
UDP Flood

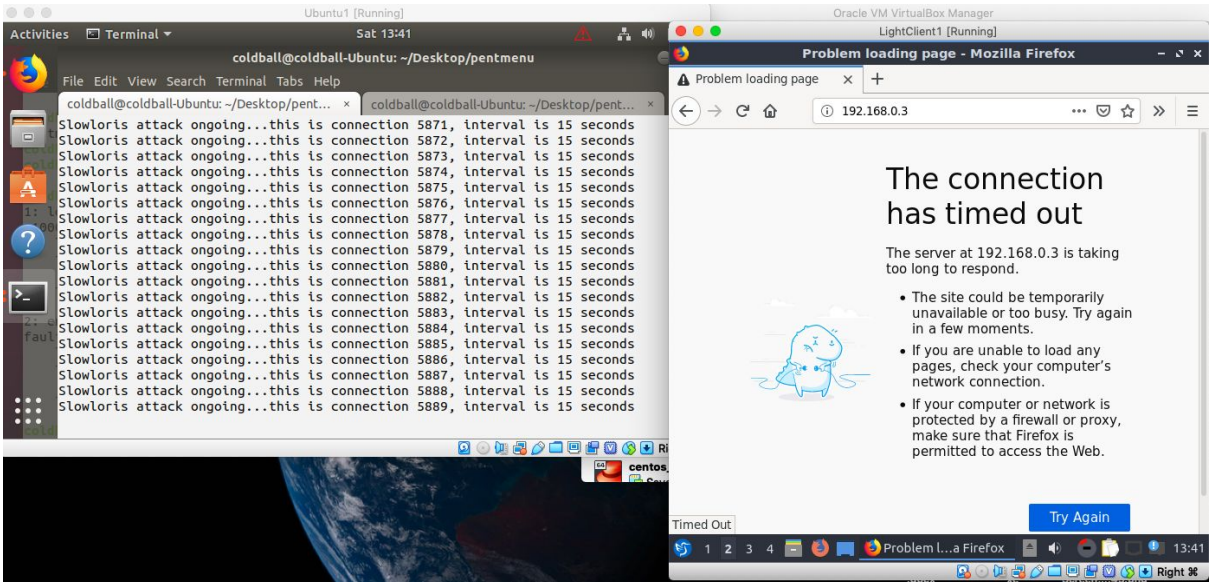
Much like the TCP SYN Flood but instead sends UDP packets to the specified host:port. Like the TCP SYN Flood function, hping3 is used but if it is not found, it attempts to use nmap-nping instead. All options are the same as TCP SYN Flood, except you must specify data to send in the UDP packets. Again, this is a good way to check switch/router throughput or to test VOIP systems.



Slowloris

Slowlorisis an application layer attack which operates by utilizing partial HTTP requests. The attack functions by opening connections to a targeted Web server and then keeping those connections open as long as it can.





Conclusion

Thus pentmenu DoS module was used to perform different DoS attacks on a website using command line.