Lab #1

Computer Networks

Aim: Introduction to computer networks

Description:

What is a Network?

A Network in the world of computers is said to be a collection of interconnected hosts, via some shared media which can be wired or wireless. A computer network enables its hosts to share and exchange data and information over the media. Network can be a Local Area Network spanned across an office or Metro Area Network spanned across a city or Wide Area Network which can be spanned across cities and provinces.

A computer network can be as simple as two PCs connected together via a single copper cable or it can be grown up to the complexity where every computer in this world is connected to every other, called the Internet. A network then includes more and more components to reach its ultimate goal of data exchange. Below is a brief description of the components involved in computer network:

1. Hosts - Hosts are said to be situated at ultimate end of the network, i.e. a host is a source of information and another host will be the destination. Information flows end to end between hosts. A host can be a user's PC, an internet Server, a database server etc.

2. Media - If wired, then it can be copper cable, fiber optic cable, and coaxial cable. If wireless, it can be free-to-air radio frequency or some special wireless band. Wireless frequencies can be used to interconnect remote sites too.

3. Hub - A hub is a multiport repeater and it is used to connect hosts in a LAN segment. Because of low throughputs hubs are now rarely used. Hub works on Layer-1 (Physical Layer) of OSI Model.

4. Switch - A Switch is a multiport bridge and is used to connect hosts in a LAN segment. Switches are much faster than Hubs and operate on wire speed. Switch works on Layer-2 (Data Link Layer), but Layer-3 (Network Layer) switches are also available.

5. Router - A router is Layer-3 (Network Layer) device which makes routing decisions for the data/information sent for some remote destination. Routers make the core of any interconnected network and the Internet.

6. Gateways - A software or combination of software and hardware put together, works for exchanging data among networks which are using different protocols for sharing data.

7. Firewall - Software or combination of software and hardware, used to protect users data from unintended recipients on the network/internet.

All components in a network ultimately serve the hosts.

Host Addressing

Communication between hosts can happen only if they can identify each other on the network. In a single collision domain (where every packet sent on the segment by one host is heard by every other host) hosts can communicate directly via MAC address.

MAC address is a factory coded 48-bits hardware address which can also uniquely identify a host. But if a host wants to communicate with a remote host, i.e. not in the same segment or logically not connected, then some means of addressing is required to identify the remote host uniquely. A logical address is given to all hosts connected to Internet and this logical address is called "*Internet Protocol Address*".

Routers

1. Everyone uses an Internetwork router to connect to the Internet. A router's first job is to route, transparently and seamlessly directing packets from one network to another.But a router can do much more. First of all, if you know how to describe "bad" behavior, a router can look for it in Internetwork traffic. For example, if you can associate certain IP addresses with the network interfaces of a router, the router can tell you if an outside computer is pretending to be inside your network--a classic IP spoofing attack.
2. Routers can also be configured to address source-routed address requests in packets. These are packets that basically say, "You see where it has my IP address here in this field? Well, when you send packets back to me, don't check your routing table or anyone else's to send me the packet. Instead, send it to this other address here."
3. Another security feature of routers is the ability to filter. Filtering applies policy to packets, declaring what is permitted and denied by using rules that specify...
    i. Network interface: Which network did this packet come from?
    ii. Source: What IP address did it come from?
    iii. Destination: Where does it want to go?
    iv. Packet type
    v. Protocol: What language to talk--for example, HTTP for Web traffic or SMTP for e-mail.
    vi. Port to use: matches the packet with a particular service running on a computer--for example, e-mail is usually on port 25, Web runs over port 80.

Conclusion:

Thus by performing this practical we learned about the basics of Computer Networks.

Lab #2

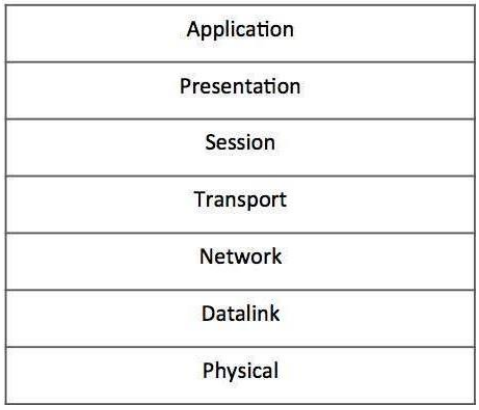
OSI Model

Aim: Introduction to OSI Model

Description:

OSI Model

1. The International Standard Organization has a well-defined model for Communication Systems known as Open System Interconnection, or the OSI Model. This layered model is a conceptualized view of how one system should communicate with the other, using various protocols defined in each layer. Further, each layer is designated to a well-defined part of communication system. For example, the Physical layer defines all the components of physical nature, i.e. wires, frequencies, pulse codes, voltage transmission etc. of a communication system.
2. The OSI Model has the following seven layers:



Application Layer (Layer-7): This is where the user application sits that needs to transfer data between or among hosts. For example: HTTP, file transfer application (FTP) and electronic mail etc.

Presentation Layer (Layer-6): This layer helps to understand data representation in one form on a host to other host in their native representation. Data from the sender is converted to on-the-wire data (general standard format) and at the receiver's end it is converted to the native representation of the receiver.

Session Layer (Layer-5): This layer provides session management capabilities between hosts. For example, if some host needs password verification for access and if credentials are provided then for that session password verification does not happen again. This layer can assist in synchronization, dialog control and critical operation management (e.g., an online bank transaction).

Transport Layer (Layer-4): This layer provides end to end data delivery among hosts. This layer takes data from the above layer and breaks it into smaller units called Segments and then gives it to the Network layer for transmission.

Network Layer (Layer-3): This layer helps to uniquely identify hosts beyond the subnets and defines the path which the packets will follow or be routed to reach the destination.

Data Link Layer (Layer-2): This layer takes the raw transmission data (signal, pulses etc.) from the Physical Layer and makes Data Frames, and sends that to the upper layer and vice versa. This layer also checks any transmission errors and sorts it out accordingly.

Physical Layer (Layer-1): This layer deals with hardware technology and actual communication mechanism such as signaling, voltage, cable type and length, etc.

Conclusion:
Thus by performing this practical we learned about layers of the OSI Model and their functionality.

Lab #3

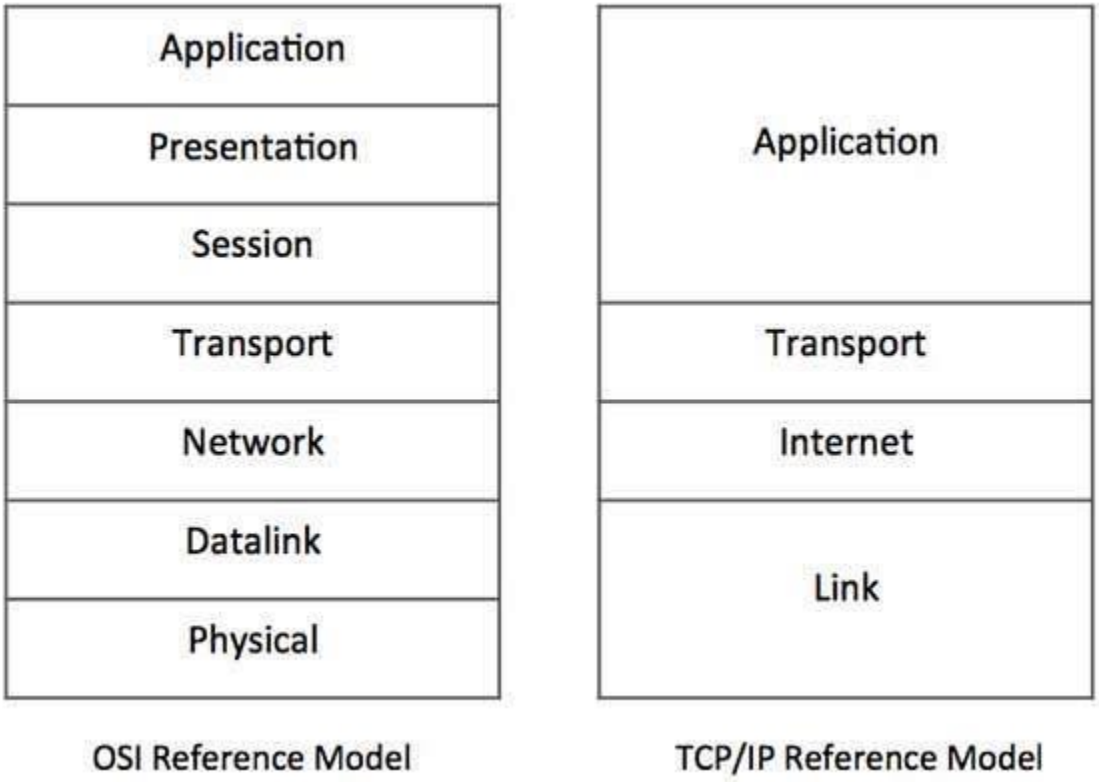
TCP-IP Model

Aim: Introduction to TCP-IP Model, IPv4s

Description:

TCP/IP Suite

A majority of the internet uses a protocol suite called the *Internet Protocol Suite* also known as the *TCP/IP protocol suite*. This suite is a combination of protocols which encompasses a number of different protocols for different purpose and need. Because the two major protocols in this suite are TCP (Transmission Control Protocol) and IP (Internet Protocol), this is commonly termed as TCP/IP Protocol suite. This protocol suite has its own reference model which it follows over the internet. In contrast with the OSI model, this model of protocols contains fewer layers.



This model is indifferent to the actual hardware implementation, i.e. the physical layer of OSI Model. This is why this model can be implemented on almost all underlying technologies. Transport and Internet layers correspond to the same peer layers. All three top layers of OSI Model are compressed together in single Application layer of TCP/IP Model.

Internet Protocol

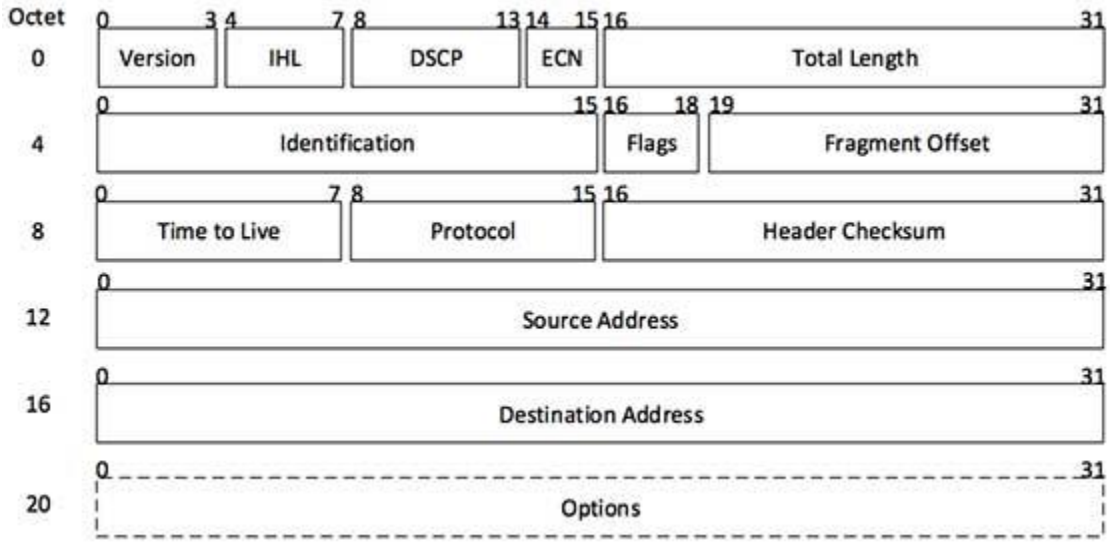
Internet Protocol is one of the major protocols in the TCP/IP protocols suite. This protocol works at the network layer of the OSI model and at the Internet layer of the TCP/IP model. Thus this protocol has the responsibility of identifying hosts based upon their logical addresses and to route data among them over the underlying network.

IP provides a mechanism to uniquely identify hosts by an IP addressing scheme. IP uses best effort delivery, i.e. it does not guarantee that packets would be delivered to the destined host, but it will do its best to reach the destination. Internet Protocol version 4 uses 32-bit logical address.

Internet Protocol being a layer-3 protocol (OSI) takes data Segments from layer-4 (Transport) and divides it into packets. IP packet encapsulates data unit received from above layer and add to its own header information.

IP Encapsulation

The encapsulated data is referred to as IP Payload. IP header contains all the necessary information to deliver the packet at the other end.



IP Header

IP header includes many relevant information including Version Number, which, in this context, is 4. Other details are as follows:

1. **Version:** Version no. of Internet Protocol used (e.g. IPv4).
2. **IHL:** Internet Header Length; Length of entire IP header.
3. **DSCP:** Differentiated Services Code Point; this is Type of Service.
4. **ECN:** Explicit Congestion Notification; It carries information about the congestion seen in the route.
5. **Total Length:** Length of entire IP Packet (including IP header and IP Payload).

6. **Identification:** If IP packet is fragmented during the transmission, all the fragments contain same identification number. to identify original IP packet they belong to.

7. **Flags:** As required by the network resources, if IP Packet is too large to handle, these 'flags' tells if they can be fragmented or not. In this 3-bit flag, the MSB is always set to '0'.

8. **Fragment Offset:** This offset tells the exact position of the fragment in the original IP Packet.
9. **Time to Live:** To avoid looping in the network, every packet is sent with some TTL value set, which tells the network how many routers (hops) this packet can cross. At each hop, its value is decremented by one and when the value reaches zero, the packet is discarded.

10. **Protocol:** Tells the Network layer at the destination host, to which Protocol this packet belongs to, i.e. the next level Protocol. For example protocol number of ICMP is 1, TCP is 6 and UDP is 17.

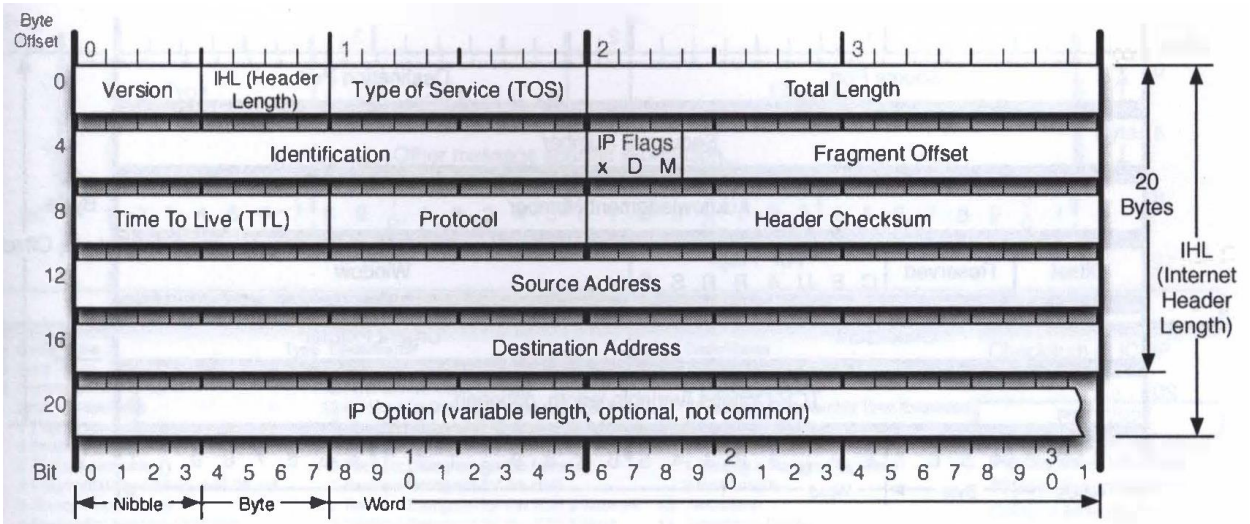
11. **Header Checksum:** This field is used to keep checksum value of entire header which is then used to check if the packet is received error-free.

12. **Source Address:** 32-bit address of the Sender (or source) of the packet.

13. **Destination Address:** 32-bit address of the Receiver (or destination) of the packet.

14. **Options:** This is optional field, which is used if the value of IHL is greater than 5. These options may contain values for options such as Security, Record Route, Time Stamp, etc.

IPv4 Header



Version

Version of IP Protocol. 4 and 6 are valid. This diagram represents version 4 structure only.

Header Length

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

Protocol

IP Protocol ID. Including (but not limited to):

1 ICMP	17 UDP	57 SKIP
2 IGMP	47 GRE	88 EIGRP
6 TCP	50 ESP	89 OSPF
9 IGRP	51 AH	115 L2TP

Total Length

Total length of IP datagram, or IP fragment if fragmented. Measured in Bytes.

Fragment Offset

Fragment offset from start of IP datagram. Measured in 8 byte (2 words, 64 bits) increments. If IP datagram is fragmented, fragment size (Total Length) must be a multiple of 8 bytes.

Header Checksum

Checksum of entire IP header

IP Flags

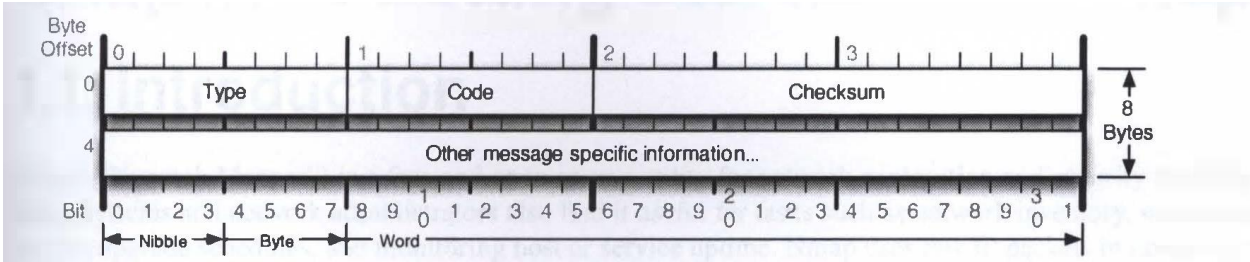
x D M

x 0x80 reserved (evil bit)
D 0x40 Do Not Fragment
M 0x20 More Fragments follow

RFC 791

Please refer to RFC 791 for the complete Internet Protocol (IP) Specification.

ICMP Header



ICMP Message Types

Type	Code/Name
0	Echo Reply
3	Destination Unreachable
	0 Net Unreachable
	1 Host Unreachable
	2 Protocol Unreachable
	3 Port Unreachable
	4 Fragmentation required, and DF set
	5 Source Route Failed
	6 Destination Network Unknown
	7 Destination Host Unknown
	8 Source Host Isolated
	9 Network Administratively Prohibited
	10 Host Administratively Prohibited
	11 Network Unreachable for TOS
3	Destination Unreachable (continued)
	12 Host Unreachable for TOS
	13 Communication Administratively Prohibited
4	Source Quench
5	Redirect
	0 Redirect Datagram for the Network
	1 Redirect Datagram for the Host
	2 Redirect Datagram for the TOS & Network
	3 Redirect Datagram for the TOS & Host
8	Echo
9	Router Advertisement
10	Router Selection
11	Time Exceded
	0 TTL Exceeded
	1 Fragment Reassembly Time Exceeded
12	Parameter Problem
	0 Pointer Problem
	1 Missing a Required Operand
	2 Bad Length
13	Timestamp
14	Timestamp Reply
15	Information Request
16	Information Reply
17	Address Mask Request
18	Address Mask Reply
30	Traceroute

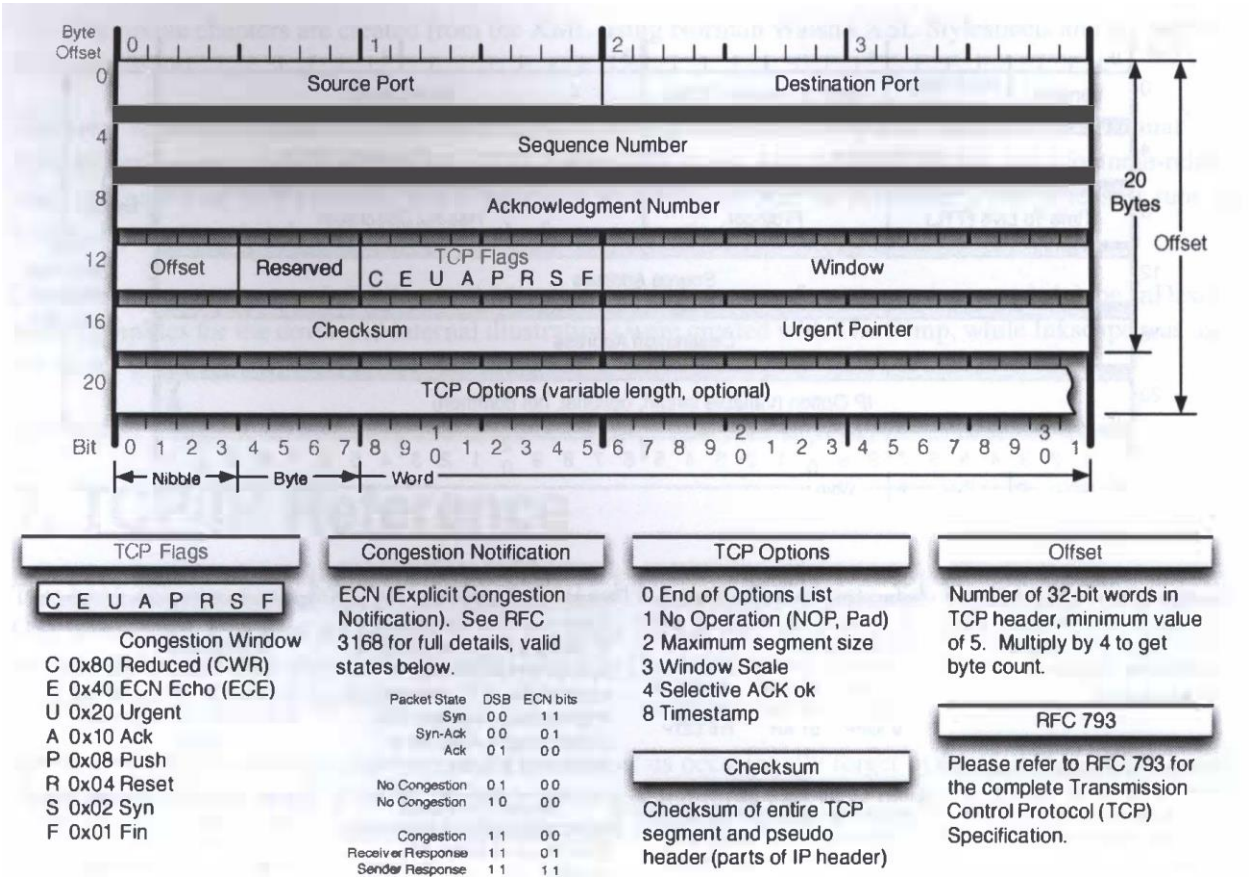
Checksum

Checksum of ICMP header

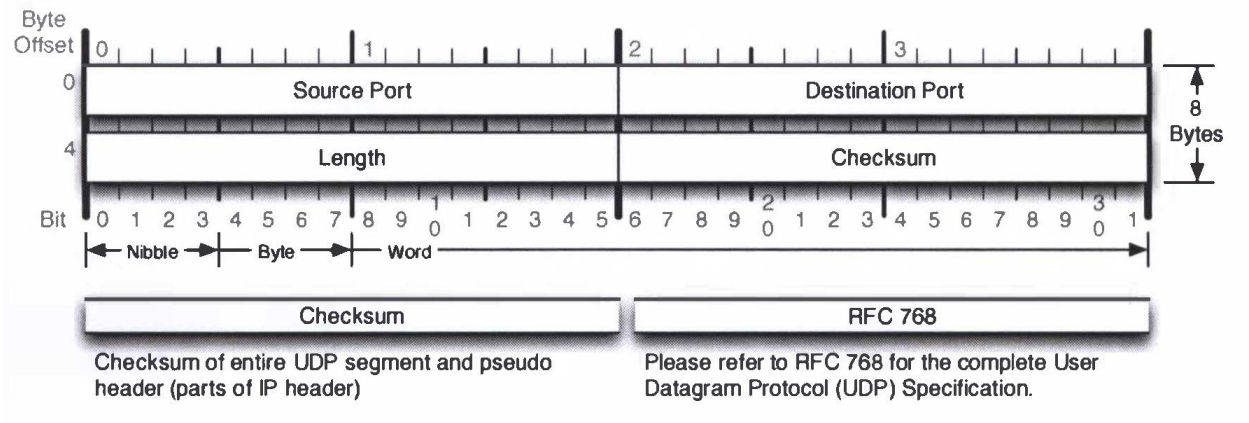
RFC 792

Please refer to RFC 792 for the Internet Control Message protocol (ICMP) specification.

TCP Header



UDP Header



TCP/IP State Transition Diagram (RFC793)

A connection progresses through a series of states during its lifetime. The states are: LISTEN, SYN-SENT, SYNRECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT, and the fictional state CLOSED.CLOSED is fictional because it represents the state when there is no TCB, and therefore, no connection.

Briefly the meanings of the states are:

LISTEN represents waiting for a connection request from any remote TCP and port.

SYN-SENT represents waiting for a matching connection request after having sent a connection request.

SYN-RECEIVED represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request.

ESTABLISHED represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.

FIN-WAIT-1 represents waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.

FIN-WAIT-2 represents waiting for a connection termination request from the remote TCP.

CLOSE-WAIT represents waiting for a connection termination request from the local user.

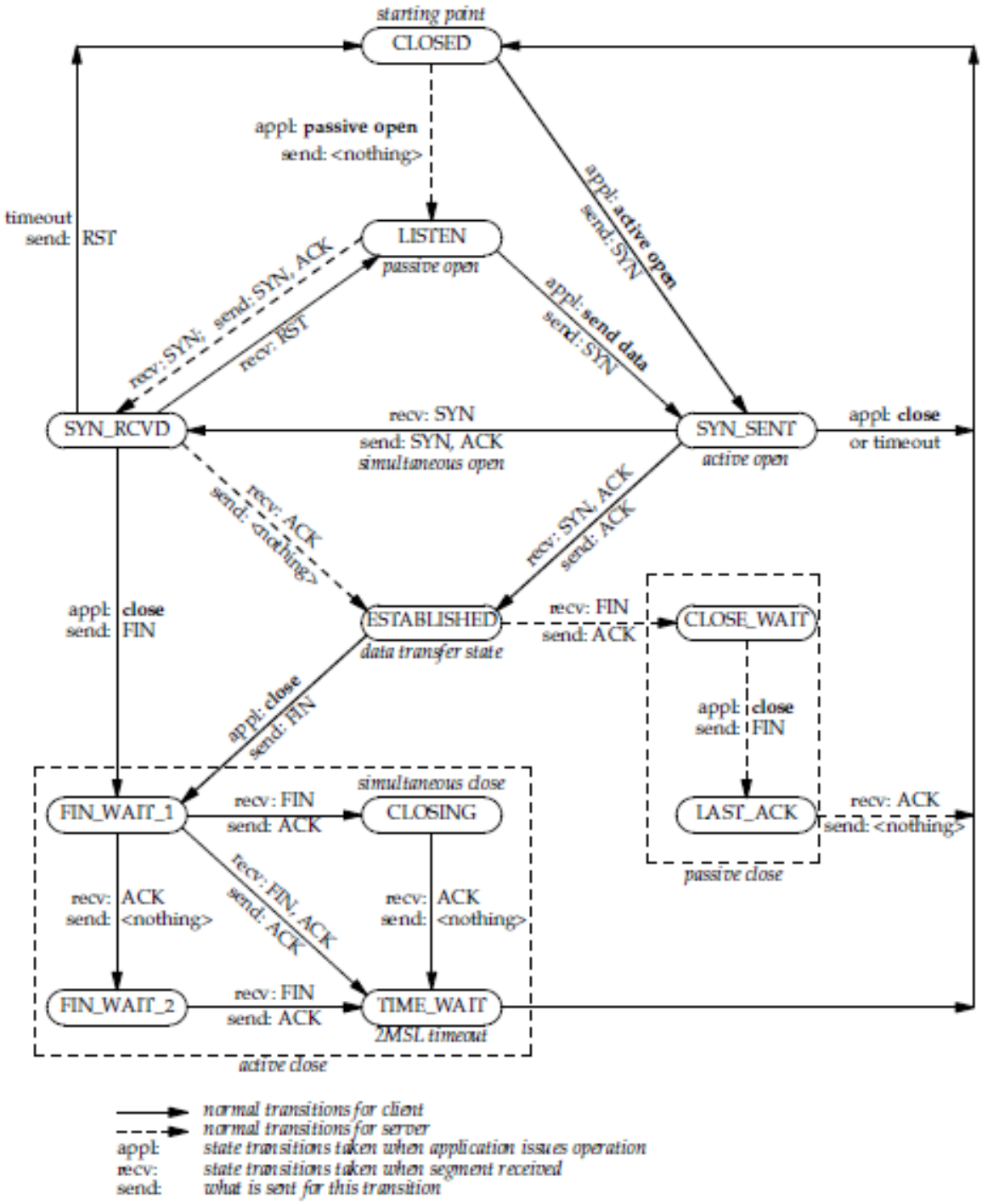
CLOSING represents waiting for a connection termination request acknowledgment from the remote TCP.

LAST-ACK represents waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).

TIME-WAIT represents waiting for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request.

CLOSED represents no connection state at all.

A TCP connection progresses from one state to another in response to events. The events are the user calls, OPEN, SEND, RECEIVE, CLOSE, ABORT, and STATUS; the incoming segments, particularly those containing the SYN, ACK, RST and FIN flags; and timeouts.



TCP state transition diagram.

Conclusion:

Thus by performing this practical we learned about layers of the TCP-IP Model, headers of IPv4, TCP, UDP, ICMP protocols and TCP-IP state transition diagram.

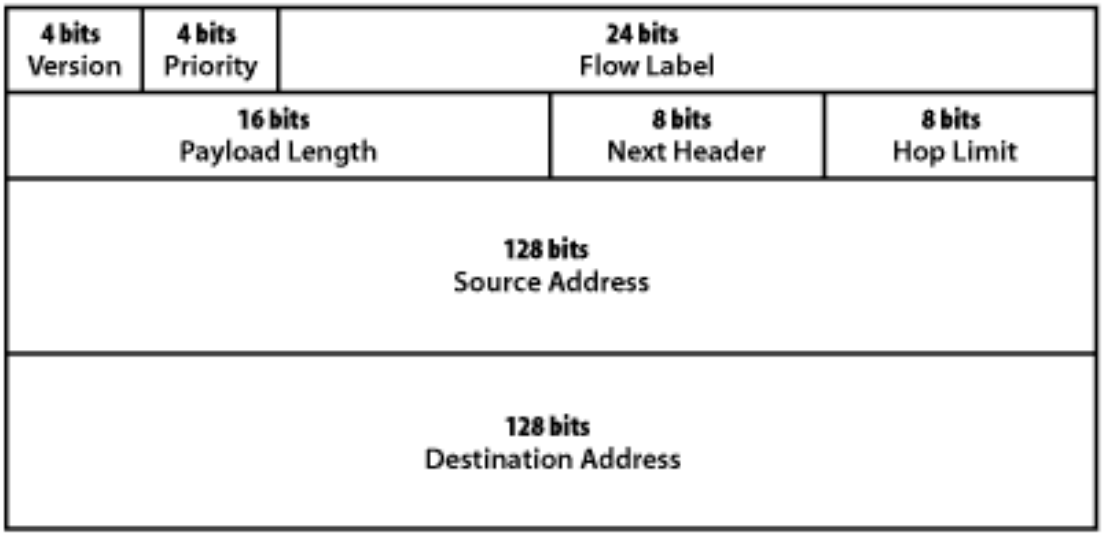
Lab #4

IPv6

Aim: Introduction to IPv6 standard

Description:

Header



Working

1. IPv6 is the Internet's next-generation protocol, designed to replace the current Internet Protocol.
2. IPv6 is a network layer protocol that enables data communications over a packet switched network.
3. Packet switching involves the sending and receiving of data in packets between two nodes in a network.
4. The working standard for the IPv6 protocol was published by the Internet Engineering Task Force (IETF) in 1998. col, IP Version 4.
5. Expanded Addressing Capabilities IPv6 increases the IP address size from 32 bits to 128 bits, to support more levels of addressing hierarchy, a much greater number of addressable nodes, and simpler auto-configuration of addresses.
6. The scalability of multicast routing is improved by adding a "scope" field to multicast addresses.
7. And a new type of address called an "anycast address" is defined, used to send a packet to any one of a group of nodes.

Comparison of IPv4 and IPv6

PARAMETER	IPv4	IPv6
Version	4th version of IP with 4-bytes (32-bit).	6th version of IP with 16-bytes (128-bit).
Address	32-bit addressing scheme.	128-bit addressing scheme.
Packet Size	Required 576 bytes and fragmentation optional	Required 1280 bytes and without fragmentation optional.
Packet fragmentation	Sending hosts and routers.	Only Sending hosts.
Packet header	I) for quality of services handling and it doesn't identify packet flow.	I) for quality of services handling and it identify packet flow.
	II) Include checksum and option up to 40-bytes	II) Doesn't include checksum and header used for optional data.
Broadcast	Yes	No
Multicast:	Yes	Yes

Conclusion:

Thus by performing this practical we learned about layers of the OSI Model and their functionality.

Lab #5

Nmap-1

Aim: Introduction to Nmap
Software: Nmap, ZenMap
Description:

1. Nmap ("Network Mapper") is an open source tool for network exploration and security auditing. It was designed to rapidly scan large networks, although it works fine against single hosts. 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. While Nmap is commonly used for security audits, many systems and network administrators find it useful for routine tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime.
2. The output from Nmap is a list of scanned targets, with supplemental information on each depending on the options used. Key among that information is the "interesting ports table". That table lists the port number and protocol, service name, and state. The state is either open, filtered, closed, or unfiltered.
    i. Open means that an application on the target machine is listening for connections/packets on that port.
    ii. Filtered means that a firewall, filter, or other network obstacle is blocking the port so that Nmap cannot tell whether it is open or closed.
    iii. Closed ports have no application listening on them, though they could open up at any time. Ports are classified as unfiltered when they are responsive to Nmap's probes, but Nmap cannot determine whether they are open or closed.
3. Nmap reports the state combinations `open|filtered` & `closed|filtered` when it cannot determine which of the two states describe a port. The port table may also include software version details when version detection has been requested. When an IP protocol scan is requested (- so), Nmap provides information on supported IP protocols rather than listening ports.
4. In addition to the interesting ports table, Nmap can provide further information on targets, including reverse DNS names, operating system guesses, device types, and MAC addresses.

Name

nmap — Network exploration tool and security / port scanner

Synopsis

`nmap [ <Scan Type> ...] [ <Options> ] { <target specification> }`

15.1. Description

Nmap ("Network Mapper") is an open source tool for network exploration and security auditing. It was designed to rapidly scan large networks, although it works fine against single hosts. 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. While Nmap is commonly used for security audits, many systems and network administrators find it useful for routine tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime.

The output from Nmap is a list of scanned targets, with supplemental information on each depending on the options used. Key among that information is the "interesting ports table". That table lists the port number and protocol, service name, and state. The state is either `open`, `filtered`, `closed`, or `unfiltered`. `Open` means that an application on the target machine is listening for connections/packets on that port. `Filtered` means that a firewall, filter, or other network obstacle is blocking the port so that Nmap cannot tell whether it is `open` or `closed`. `Closed` ports have no application listening on them, though they could open up at any time. Ports are classified as `unfiltered` when they are responsive to Nmap's probes, but Nmap cannot determine whether they are open or closed. Nmap reports the state combinations `open|filtered` and `closed|filtered` when it cannot determine which of the two states describe a port. The port table may also include software version details when version detection has been requested. When an IP protocol scan is requested (`-sO`), Nmap provides information on supported IP protocols rather than listening ports.

In addition to the interesting ports table, Nmap can provide further information on targets, including reverse DNS names, operating system guesses, device types, and MAC addresses.

15.2. Options Summary

This options summary is printed when Nmap is run with no arguments, and the latest version is always available at *http://nmap.org/data/nmap.usage.txt*. It helps people remember the most common options, but is no substitute for the in-depth documentation in the rest of this manual. Some obscure options aren't even included here.

```
Nmap 4.76 ( http://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sP: Ping Scan - go no further than determining if host is online
  -PN: Treat all hosts as online -- skip host discovery
  -PS/PA/PU [portlist]: TCP SYN/ACK or UDP discovery to given ports
```

-PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
-PO [protocol list]: IP Protocol Ping
-n/-R: Never do DNS resolution/Always resolve [default: sometimes]
--dns-servers <serv1[,serv2],...>: Specify custom DNS servers
--system-dns: Use OS's DNS resolver
SCAN TECHNIQUES:
-sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
-sU: UDP Scan
-sN/sF/sX: TCP Null, FIN, and Xmas scans
--scanflags <flags>: Customize TCP scan flags
-sI <zombie host[:probeport]>: Idle scan
-sO: IP protocol scan
-b <FTP relay host>: FTP bounce scan
--traceroute: Trace hop path to each host
--reason: Display the reason a port is in a particular state
PORT SPECIFICATION AND SCAN ORDER:
-p <port ranges>: Only scan specified ports
Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080
-F: Fast mode - Scan fewer ports than the default scan
-r: Scan ports consecutively - don't randomize
--top-ports <number>: Scan <number> most common ports
--port-ratio <ratio>: Scan ports more common than <ratio>
SERVICE/VERSION DETECTION:
-sV: Probe open ports to determine service/version info
--version-intensity <level>: Set from 0 (light) to 9 (try all probes)
--version-light: Limit to most likely probes (intensity 2)
--version-all: Try every single probe (intensity 9)
--version-trace: Show detailed version scan activity (for debugging)
SCRIPT SCAN:
-sC: equivalent to --script=default
--script=<Lua scripts>: <Lua scripts> is a comma separated list of
directories, script-files or script-categories
--script-args=<n1=v1,[n2=v2,...]>: provide arguments to scripts
--script-trace: Show all data sent and received
--script-updatedb: Update the script database.
OS DETECTION:
-O: Enable OS detection
--osscan-limit: Limit OS detection to promising targets
--osscan-guess: Guess OS more aggressively
TIMING AND PERFORMANCE:
Options which take <time> are in milliseconds, unless you append 's'
(seconds), 'm' (minutes), or 'h' (hours) to the value (e.g. 30m).
-T[0-5]: Set timing template (higher is faster)
--min-hostgroup/max-hostgroup <size>: Parallel host scan group sizes
--min-parallelism/max-parallelism <time>: Probe parallelization
--min-rtt-timeout/max-rtt-timeout/initial-rtt-timeout <time>: Specifies
probe round trip time.
--max-retries <tries>: Caps number of port scan probe retransmissions.
--host-timeout <time>: Give up on target after this long

--scan-delay/--max-scan-delay <time>: Adjust delay between probes
--min-rate <number>: Send packets no slower than <number> per second
--max-rate <number>: Send packets no faster than <number> per second
FIREWALL/IDS EVASION AND SPOOFING:
-f; --mtu <val>: fragment packets (optionally w/given MTU)
-D <decoy1,decoy2[,ME],...>: Cloak a scan with decoys
-S <IP_Address>: Spoof source address
-e <iface>: Use specified interface
-g/--source-port <portnum>: Use given port number
--data-length <num>: Append random data to sent packets
--ip-options <options>: Send packets with specified ip options
--ttl <val>: Set IP time-to-live field
--spoof-mac <mac address/prefix/vendor name>: Spoof your MAC address
--badsum: Send packets with a bogus TCP/UDP checksum
OUTPUT:
-oN/-oX/-oS/-oG <file>: Output scan in normal, XML, sl<rIpt kIddi3,
and Grepable format, respectively, to the given filename.
-oA <basename>: Output in the three major formats at once
-v: Increase verbosity level (use twice or more for greater effect)
-d[level]: Set or increase debugging level (Up to 9 is meaningful)
--open: Only show open (or possibly open) ports
--packet-trace: Show all packets sent and received
--iflist: Print host interfaces and routes (for debugging)
--log-errors: Log errors/warnings to the normal-format output file
--append-output: Append to rather than clobber specified output files
--resume <filename>: Resume an aborted scan
--stylesheet <path/URL>: XSL stylesheet to transform XML output to HTML
--webxml: Reference stylesheet from Nmap.Org for more portable XML
--no-stylesheet: Prevent associating of XSL stylesheet w/XML output
MISC:
-6: Enable IPv6 scanning
-A: Enables OS detection and Version detection, Script scanning and Traceroute
--datadir <dirname>: Specify custom Nmap data file location
--send-eth/--send-ip: Send using raw ethernet frames or IP packets
--privileged: Assume that the user is fully privileged
--unprivileged: Assume the user lacks raw socket privileges
-V: Print version number
-h: Print this help summary page.
EXAMPLES:
nmap -v -A scanme.nmap.org
nmap -v -sP 192.168.0.0/16 10.0.0.0/8
nmap -v -iR 10000 -PN -p 80

15.3. Target Specification

Everything on the Nmap command-line that isn't an option (or option argument) is treated as a target host specification. The simplest case is to specify a target IP address or hostname for scanning.

Sometimes you wish to scan a whole network of adjacent hosts. For this, Nmap supports CIDR-style addressing. You can append /*<numbits>* to an IP address or hostname and Nmap will scan every IP address for which the first *<numbits>* are the same as for the reference IP or hostname given. For example, 192.168.10.0/24 would scan the 256 hosts between 192.168.10.0 (binary: `11000000 10101000 00001010 00000000`) and 192.168.10.255 (binary: `11000000 10101000 00001010 11111111`), inclusive. 192.168.10.40/24 would do exactly the same thing. Given that the host scanme.nmap.org is at the IP address 64.13.134.52, the specification scanme.nmap.org/16 would scan the 65,536 IP addresses between 64.13.0.0 and 64.13.255.255. The smallest allowed value is /0, which scans the whole Internet. The largest value is /32, which scans just the named host or IP address because all address bits are fixed.

CIDR notation is short but not always flexible enough. For example, you might want to scan 192.168.0.0/16 but skip any IPs ending with .0 or .255 because they are commonly broadcast addresses. Nmap supports this through octet range addressing. Rather than specify a normal IP address, you can specify a comma separated list of numbers or ranges for each octet. For example, 192.168.0-255.1-254 will skip all addresses in the range that end in .0 and or .255. Ranges need not be limited to the final octets: the specifier 0-255.0-255.13.37 will perform an Internet-wide scan for all IP addresses ending in 13.37. This sort of broad sampling can be useful for Internet surveys and research.

IPv6 addresses can only be specified by their fully qualified IPv6 address or hostname. CIDR and octet ranges aren't supported for IPv6 because they are rarely useful.

Nmap accepts multiple host specifications on the command line, and they don't need to be the same type. The command **nmap scanme.nmap.org 192.168.0.0/16 10.0.0,1,3-7.0-255** does what you would expect.

While targets are usually specified on the command lines, the following options are also available to control target selection:

`-iL` *<inputfilename>* (Input from list)
: Reads target specifications from *<inputfilename>*. Passing a huge list of hosts is often awkward on the command line, yet it is a common desire. For example, your DHCP server might export a list of 10,000 current leases that you wish to scan. Or maybe you want to scan all IP addresses *except* for those to locate hosts using unauthorized static IP addresses. Simply generate the list of hosts to scan and pass that filename to Nmap as an argument to the `-iL` option. Entries can be in any of the formats accepted by Nmap on the command line (IP address, hostname, CIDR, IPv6, or octet ranges). Each entry must be separated by one or more spaces, tabs, or newlines. You can specify a hyphen (`-`) as the filename if you want Nmap to read hosts from standard input rather than an actual file.

`-iR` *<num hosts>* (Choose random targets)
: For Internet-wide surveys and other research, you may want to choose targets at random. The *<num hosts>* argument tells Nmap how many IPs to generate. Undesirable IPs such as those in certain private, multicast, or unallocated address ranges are automatically skipped. The argument `0` can be specified for a never-ending scan. Keep in mind that some network administrators bristle at unauthorized scans of their networks and may complain. Use this option at your own risk! If you find yourself really bored one rainy afternoon, try the command **nmap -sS -PS80 -iR 0 -p 80** to locate random web servers for browsing.

`--exclude` *<host1>*`[,`*<host2>*`[,...]]` (Exclude hosts/networks)
: Specifies a comma-separated list of targets to be excluded from the scan even if they are part of the overall network range you specify. The list you pass in uses normal Nmap syntax, so it can include hostnames, CIDR netblocks, octet ranges, etc. This can be useful when the network you wish to scan

includes untouchable mission-critical servers, systems that are known to react adversely to port scans, or subnets administered by other people.

`--excludefile <exclude_file>` (Exclude list from file)
This offers the same functionality as the `--exclude` option, except that the excluded targets are provided in a newline, space, or tab delimited `<exclude_file>` rather than on the command line.

Conclusion:
Thus, we've studied all utilities of Nmap and also learned about scan command syntax and their operation.

Lab #6

Nmap-2

Aim: Port Scanning using NMap
Software: Nmap, ZenMap
Description:

Port Scanning

1. Port scanning allows a hacker to determine what services are running on the systems that have been identified. If vulnerable or insecure services are discovered, the hacker may be able to exploit these to gain unauthorized access.
2. There are a total of 65,535 * 2 ports (TCP & UDP). While a complete scan of all these ports may not be practical, an analysis of popular ports should be performed. Many port scanners ping first, so make sure to turn this feature off to avoid missing systems that have blocked ICMP.
3. Popular port scanning programs include: Nmap, Netscan Tools, Superscan and Angry IP Scanner.
4. The port numbers are divided into three ranges:
   1. Well Known Ports (from 0 through 1023)
   2. Registered Ports (from 1024 through 49151)
   3. Dynamic and/or Private Ports (from 49152 through 65535).

Common Scan types

4. TCP Full Connect scan: This type of scan is the most reliable but also the most detectable. It is easily logged and detected because a full connection is established. Open ports reply with a SYN/ACK; closed ports respond with a RST/ACK.

5. TCP SYN scan: This type of scan is known as half-open, because a full TCP connection is not established. This type of scan was originally developed to be stealthy and evade IDS systems, although most now detect it. Open ports reply with a SYN/ACK; closed ports respond with a RST/ACK.

6. TCP FIN scan: Forget trying to set up a connection; this technique jumps straight to the shutdown. This type of scan sends a FIN packet to the target port. Closed ports should send back an RST. This technique is usually effective only on Unix devices.

7. TCP NULL scan: Sure, there should be some type of flag in the packet, but a NULL scan sends a packet with no flags set. If the OS has implemented TCP per RFC 793, closed ports will return an RST.

8. TCP ACK scan: This scan attempts to determine access control list (ACL) rule sets or identify whether stateless inspection is being used. If an ICMP Destination Unreachable, Communication Administrative Prohibited message is returned, the port is considered to be filtered.

9. TCP XMAS scan: just a port scan that has toggled on the FIN, URG, and PSH flags. Closed ports should return an RST.

Nmap Output

1. The output from Nmap is a list of scanned targets, with supplemental information on each depending on the options used. Key among that information is the interesting ports table.

2. Filtered means that a firewall, filter, or other network obstacle is blocking the port so that Nmap cannot tell whether it is open or closed. Closed ports have no application listening on them, though they could open up at any time. Ports are classified as unfiltered when they are responsive to Nmap's probes, but Nmap cannot determine whether they are open or closed. Nmap reports the state combinations open|filtered and closed|filtered when it cannot determine which of the two states describe a port.

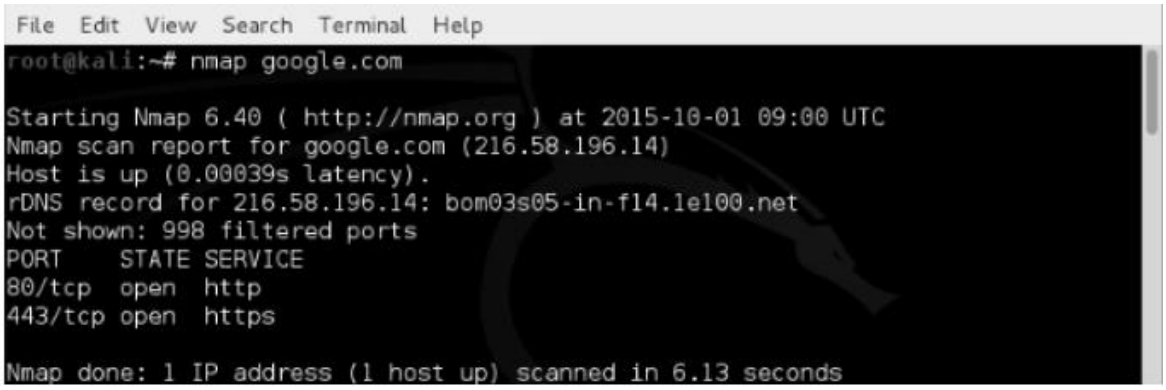
Nmap Scan Options

When we use the command line in the Nmap tool instead of GUI, we need some option which listed with the command to define the type of scan methods. The table below lists some of these options.

Scan Option	Name	Notes
-sS	**TCP SYN**	Stealth scan
-sT	**TCP FULL**	Full connect
-sF	**FIN**	No reply from open port
-sN	**Null**	No flags are set
-sX	**Xmas**	URG,PUSH, and FIN are set
-sP	**Ping**	Performs ping
-sU	**UDP Scan**	Like Null scan
-sA	**ACK**	Performs an ACK scan
-sI	**Idle Scan**	Performs zombie scan

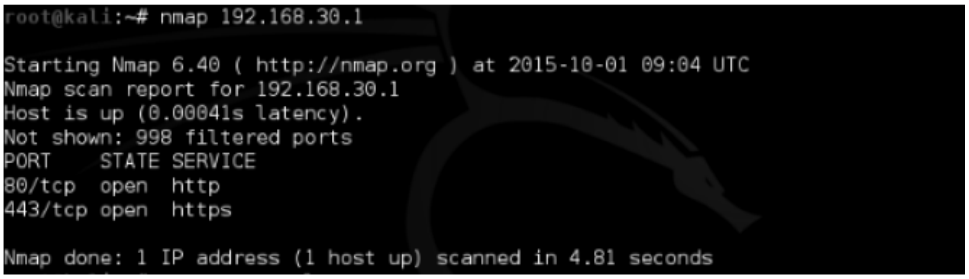
Scan a system with Hostname

The Nmap tool offers various methods to scan a system. Here we use google.com as a hostname.



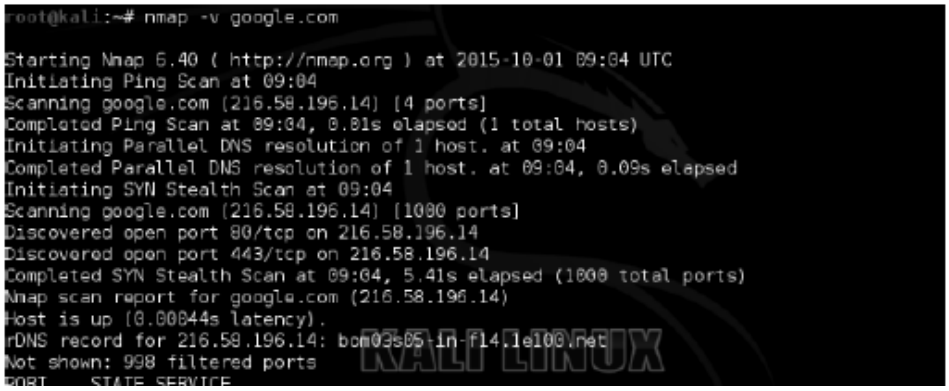
Scan using IP Address

Here we use an IP Address as 192.168.30.1



Scan using "-v" option

It gives more information about the remote host.



Scan multiple hosts

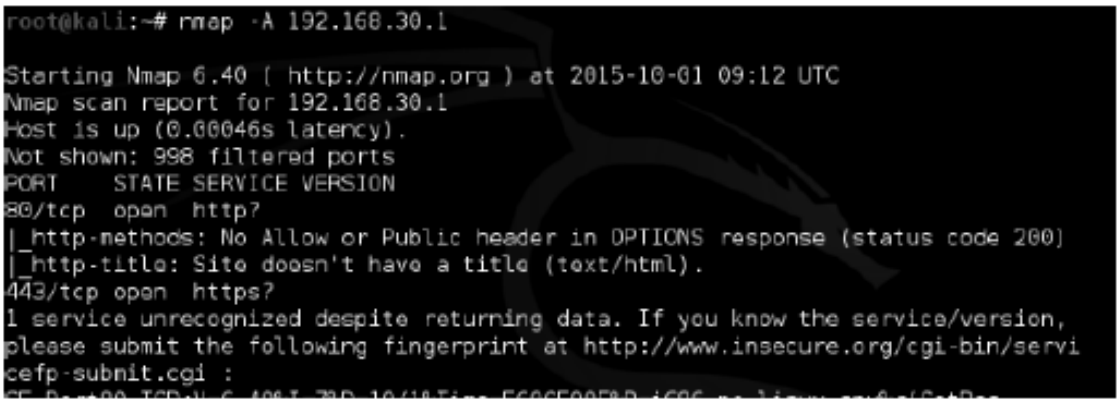
Scan multiple hosts by simply writing the IP Address or hostnames with Nmap.

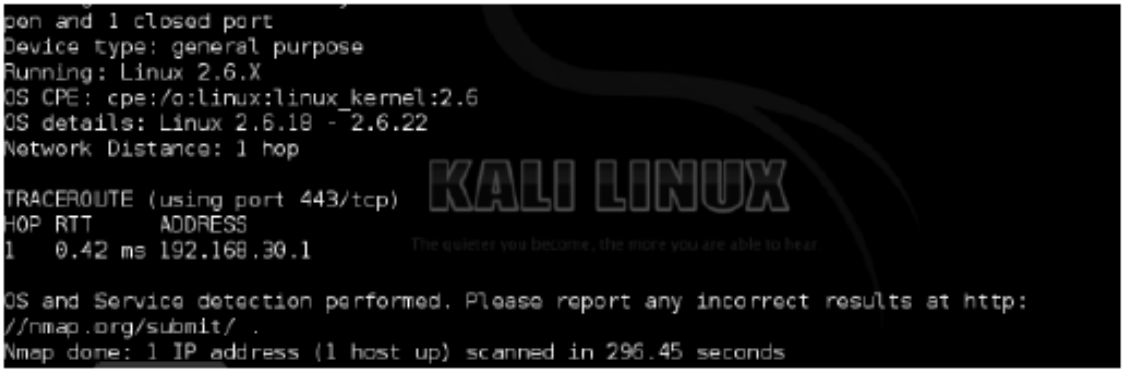
Scan multiple servers using the last octet of IP Address

Scans on multiple IP address by simple specifying last octet of IP address. For example, here to scan on IP addresses 192.168.30.1, 192.168.30.2 and 192.168.30.3.

Scan OS information

With Nmap, you can detect which OS and version is running on the remote host. To enable OS & version detection, script scanning and traceroute, we can use -A option with Nmap.





Conclusion:

Thus, by performing this practical we got to know about the working of Nmap and scanning of Ports using Nmap.

Lab #7

Nmap-3

Aim: TCP/UDP Scanning using NMap
Software: Nmap, ZenMap
Description:

TCP Scanning

The two basic scan types used most in Nmap are:

1. **TCP Connect scanning**
   Socket programming uses a system call named connect to begin a TCP connection to a remote site. If connect succeeds, a connection was made.This allows a basic type of port scan, which attempts to connect to every port in turn, and notes whether or not the connection succeeded. Once the scan is completed, ports to which a connection could be established are listed as open, the rest are said to be closed.

2. **TCP SYN scanning**
   When a TCP connection is made between two systems, a process known as a "three way handshake" occurs. This involves the exchange of three packets, and synchronises the systems with each other.

Conclusion:
Thus by performing this practical we learned about the working of NMap for TCP/UDP Scanning.

Lab #8

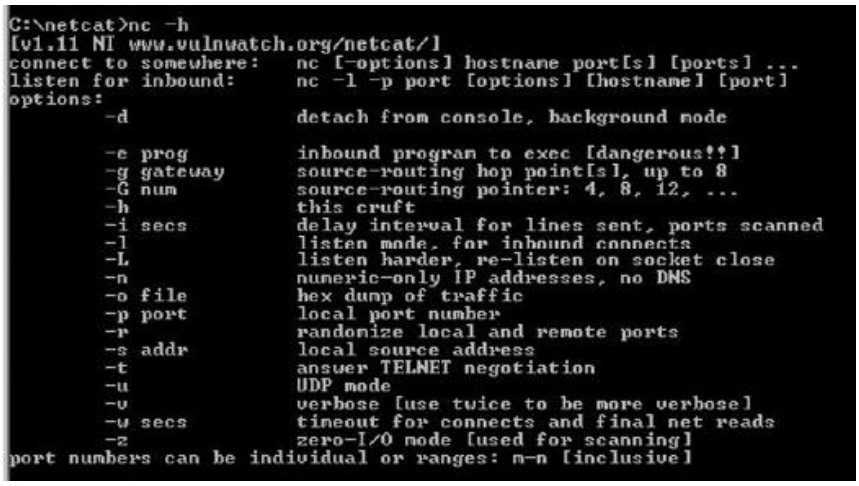
Netcat-1

Aim: Introduction to Netcat
Software: Netcat, Nmap (*Nmap consists of NCat, an quivalent of Netcat)
Description:

Introduction

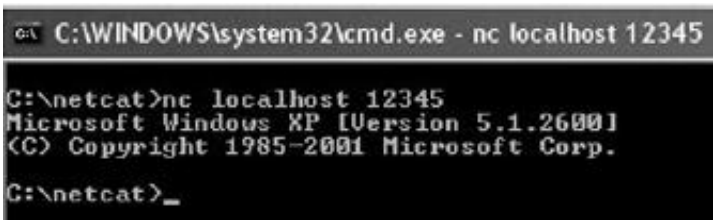
1. Originally released in 1996, Netcat is a networking program designed to read and write data across both Transmission Control Protocol TCP and User Datagram Protocol (UDP) connections using the TCP/Internet Protocol (IP) protocol suite. Netcat is often referred to as a ”Swiss Army knife” utility, and for good reason. Just like the multi-function usefulness of the venerable Swiss Army pocket knife, Netcat’s functionality is helpful as both a standalone program and a back-end tool in a wide range of applications. Some of the many uses of Netcat include port scanning, transferring files, grabbing banners, port listening and redirection, and more nefariously, a backdoor.
2. There is some debate on the origin of the name Netcat, but one of the more common (and believable) explanations is that Netcat is simply a network version of the vulnerable cat program. Just as cat reads and writes information to files, Netcat reads and writes information across network connections. Furthermore, Netcat is specifically designed to behave as cat does.
3. Originally coded for UNIX, and despite not originally being maintained on a regular basis, Netcat has been rewritten into a number of versions and implementations. It has been ported to a number of operating systems, but is most often seen on various Linux distributions as well as Microsoft Windows.

Options



Modes of Operation

Netcat has two primary modes of operation, as a client, and as a server.



Conclusion:

Thus by performing this practical we learned about the working of NMap for TCP/UDP Scanning.

Lab #9

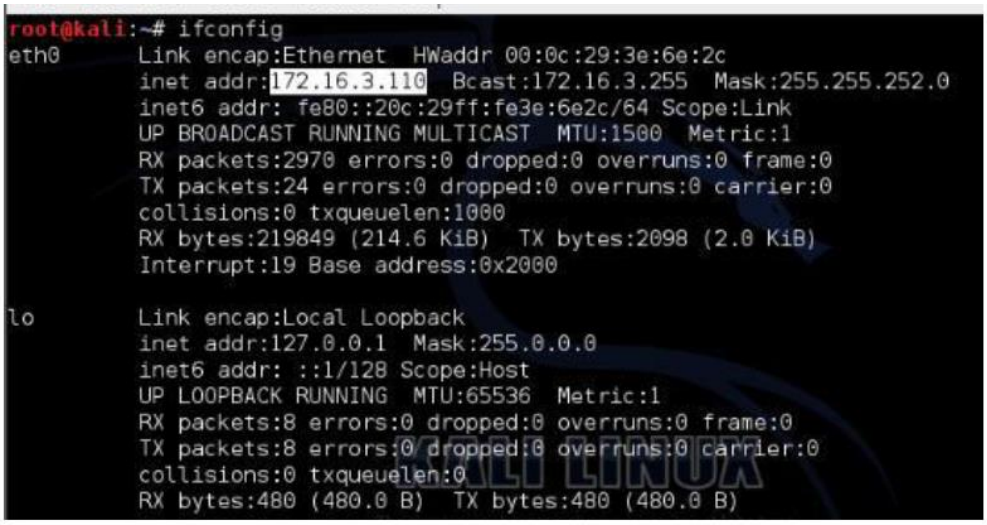
Netcat-2

Aim: TCP/UDP Connectivity Using Netcat
Software: Netcat, Nmap (*Nmap consists of NCat, an quivalent of Netcat)
Description:

Netcat

1. Netcat is a terminal application that is similar to the telnet program but has lot more features. It`s a "power version" of the traditional telnet program. Apart from basic telnet function as it can do various other things like creating socket servers to listen for incoming connections on ports, transfer files from the terminal etc. So it is a small tool that is packed with lots of features. Therefore it`s called the "Swiss-army knife for TCP/IP".
2. Netcat is a computer networking service for reading from and writing network connections using TCP or UDP. Netcat is designed to be a dependable "back-end" device that can be used directly or easily driven by other programs and scripts. At the same time, it is a feature-rich network debugging and investigation tool, since it can produce almost any kind of correlation you would need and has a number of built-in capabilities.

Scan IP Address using ipconfig command

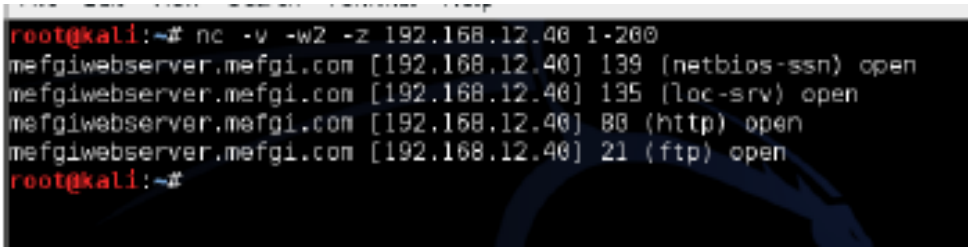


Listening server on port 12348

Now we connect to a server with following details:

192.168.106.128: Server IP Address. 12348: Port no. on which server is listening.

Finding an open port between 1-200



Conclusion:

Thus, by performing this practical we got to know about the Netcat and its connectivity for communication via terminal of Linux.

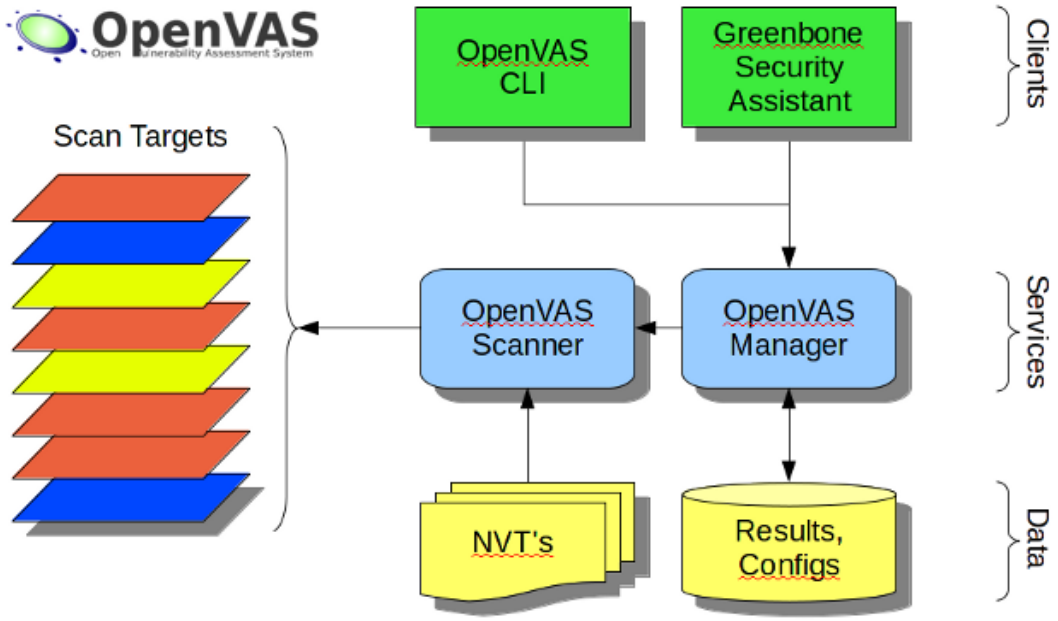
Lab #10

OpenVAS

Aim: Introduction to OpenVAS
Software: OpenVAS
Description:

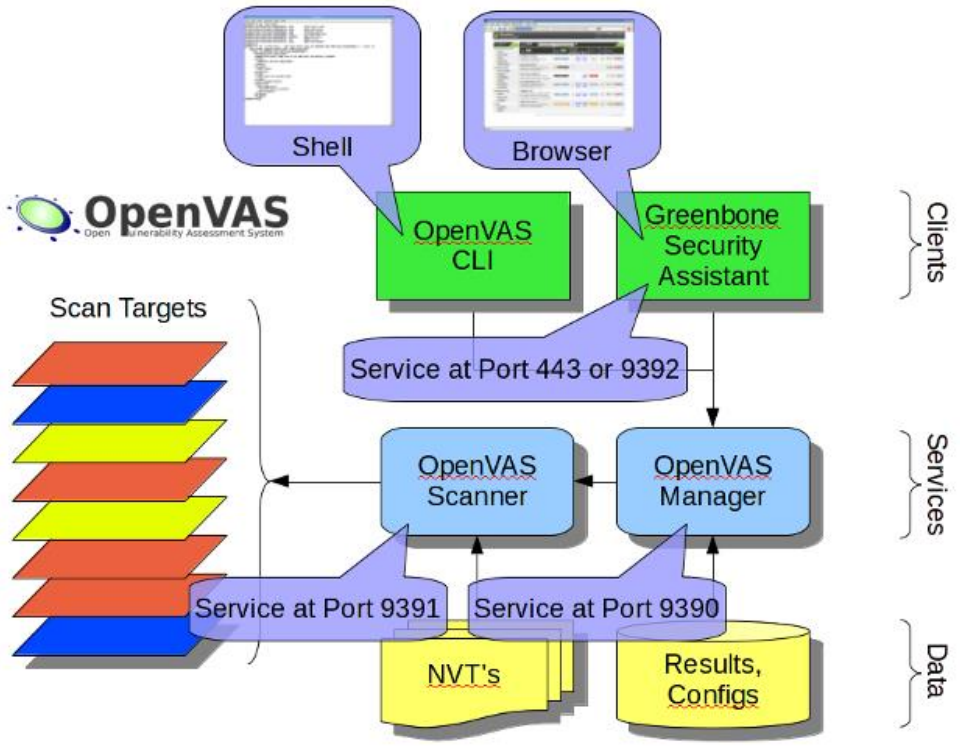
OpenVAS

1. The Open Vulnerability Assessment System (OpenVAS) is a framework of several services and tools offering a comprehensive and powerful vulnerability scanning and vulnerability management solution.
2. The actual security scanner is accompanied with a daily updated feed of Network Vulnerability Tests (NVTs), over 35,000 in total (as of April 2014).
3. All OpenVAS products are Free Software. Most components are licensed under the GNU General Public License (GNU GPL).

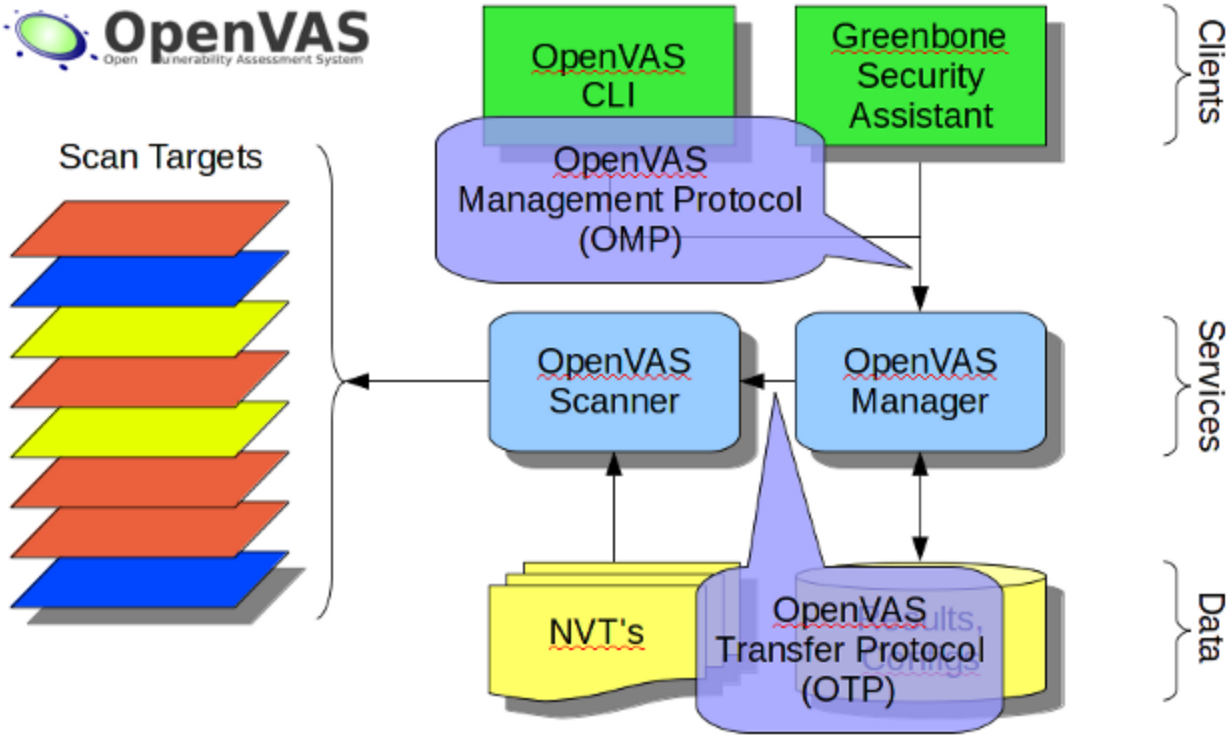


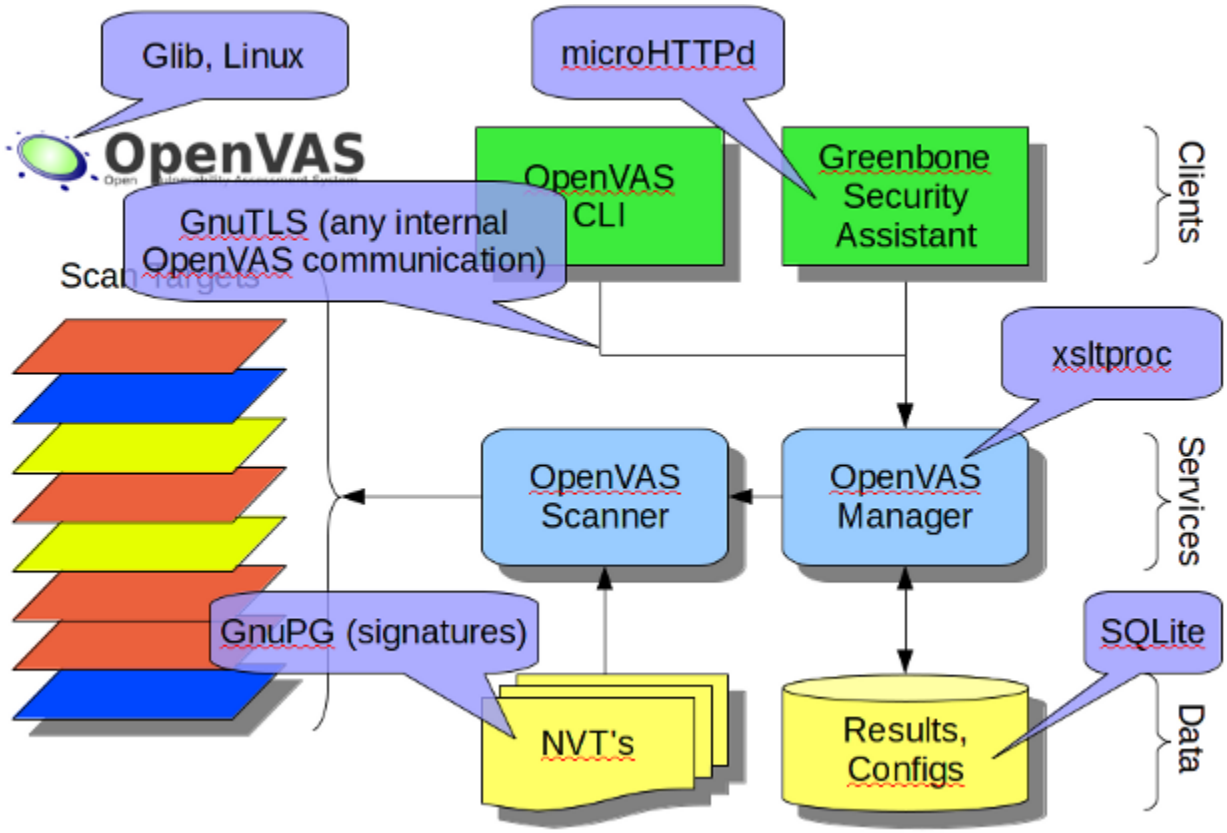
Architecture Overview

1. The Open Vulnerability Assessment System (OpenVAS) is a framework of several services and tools. The core of this SSL-secured service-oriented architecture is the OpenVAS Scanner. The scanner very efficiently executes the actual Network Vulnerability Tests (NVTs) which are served with daily updates via the OpenVAS NVT Feed or via a commercial feed service.



2. The OpenVAS Manager is the central service that consolidates plain vulnerability scanning into a full vulnerability management solution. The Manager controls the Scanner via OTP (OpenVAS Transfer Protocol) and itself offers the XML-based, stateless OpenVAS Management Protocol (OMP).
3. All intelligence is implemented in the Manager so that it is possible to implement various lean clients that will behave consistently e.g. with regard to filtering or sorting scan results. The Manager also controls a SQL database (sqlite-based) where all configuration and scan result data is centrally stored. Finally, Manager also handles user management includiung access control with groups and roles.
4. Different OMP clients are available: The Greenbone Security Assistant (GSA) is a lean web service offering a user interface for web browsers. GSA uses XSL transformation stylesheet that converts OMP responses into HTML.
5. OpenVAS CLI contains the command line tool "omp" which allows to create batch processes to drive OpenVAS Manager. Another tool of this package is a Nagios plugin.
6. The OpenVAS Scanner offers the communication protocol OTP (OpenVAS Transfer Protocol) which allows to control the scan execution. This protocol is subject to be eventually replaced and thus it is not recommended to develop OTP clients.


OpenVAS
Open Vulnerability Assessment System
OpenVAS CLI
Greenbone Security Assistant
Clients
Scan Targets
OpenVAS Management Protocol (OMP)
OpenVAS Scanner
OpenVAS Manager
Services
NVT's
OpenVAS Transfer Protocol (OTP)
Data

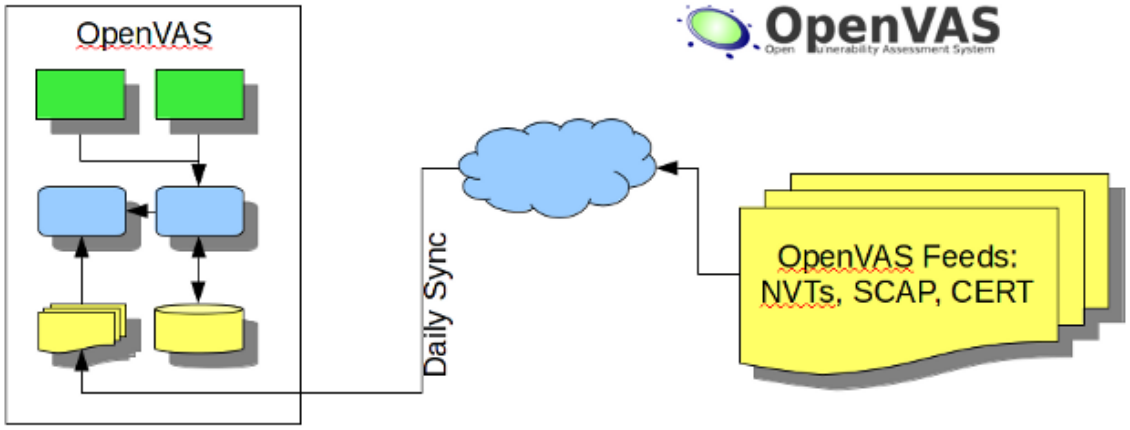

Glib, Linux
microHTTPd
OpenVAS
Open Vulnerability Assessment System
OpenVAS CLI
Greenbone Security Assistant
Clients
GnuTLS (any internal OpenVAS communication)
Scan Targets
xsltproc
OpenVAS Scanner
OpenVAS Manager
Services
GnuPG (signatures)
SQLite
NVT's
Results, Configs
Data

Feature overview

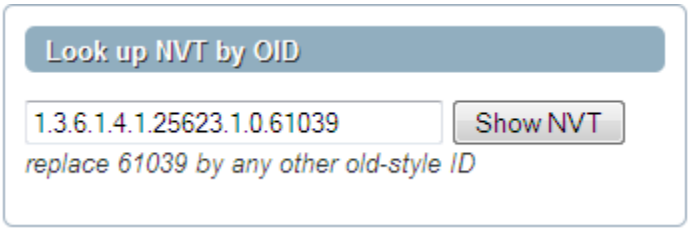
1. OpenVAS Scanner
    - Many target hosts are scanned concurrently
    - OpenVAS Transfer Protocol (OTP)
    - SSL support for OTP (always)
    - WMI support (optional)
2. OpenVAS Manager
    - OpenVAS Management Protocol (OMP)
    - SQL Database (sqlite) for configurations and scan results
    - SSL support for OMP (always)
    - Many concurrent scans tasks (many OpenVAS Scanners)
    - Notes management for scan results
    - False Positive management for scan results
    - Scheduled scans
    - Flexible escalators upon status of a scan task
    - Stop, Pause and Resume of scan tasks
    - Master-Slave Mode to control many instances from a central one
    - Reports Format Plugin Framework with various plugins for: XML, HTML, LateX, etc.
    - User Management
    - Feed status view
    - Feed synchronisation
3. Greenbone Security Assistant (GSA)
    - Client for OMP and OAP
    - HTTP and HTTPS
    - Web server on its own (microhttpd), thus no extra web server required
    - Integrated online-help system
    - Multi-language support
4. OpenVAS CLI
    - Client for OMP

About OpenVAS NVT Feed

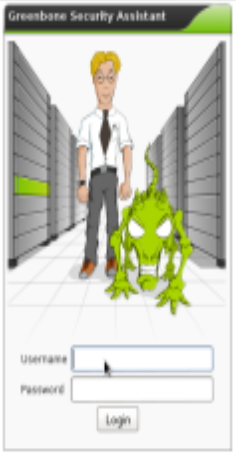
1. The OpenVAS project maintains a public feed of Network Vulnerability Tests (NVTs). It contains more than 35,000 NVTs (as of April 2014), growing on a daily basis. This feed is configured as the default for OpenVAS.
2. For online-synchronisation use the command **openvas-nvt-sync** to update your local NVTs with the newest ones from the feed service. The command allows *rsync*, *wget* or *curl* as transfer method.
3. For offline-updates it is also possible to download the whole Feed content as a single archive file (around 14 MByte). However, it is recommended to use the rsync-synchronisation routine because it downloads only changes and therefore is tremendously faster after the very first full download.



4. The feed is usually updated weekly. The files of the OpenVAS NVT Feed are signed by the "OpenVAS: Transfer Integrity" certificate. The presence of this signature **does not indicate any judgement or quality control** of the script itself. It is only intended to assist you in verifying the integrity of the NVT files after transfer. Thus, a valid signature only means that the script has not been modified on the way between the OpenVAS distribution point and your OpenVAS installation. See the notes at the bottom of the overview on Trusted NVTs for more information on this certificate.

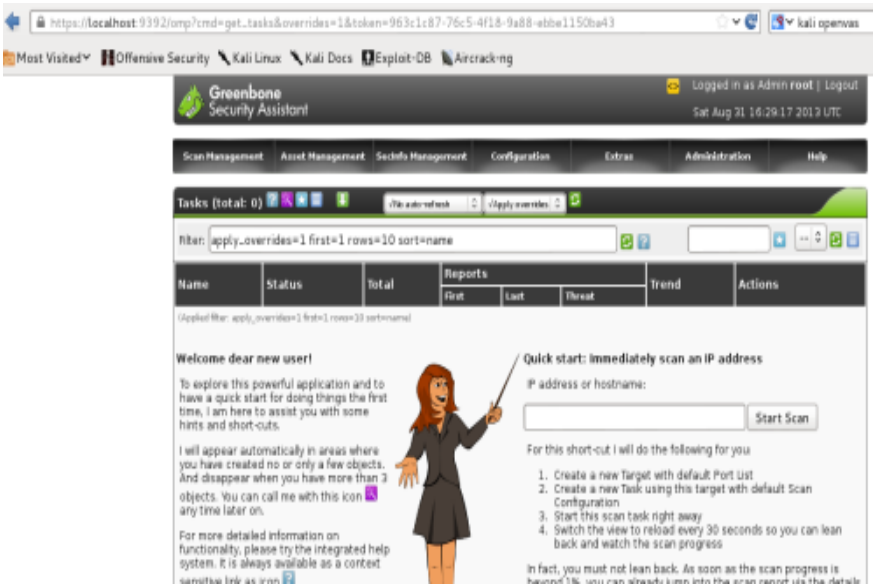


OpenVAS Login Box



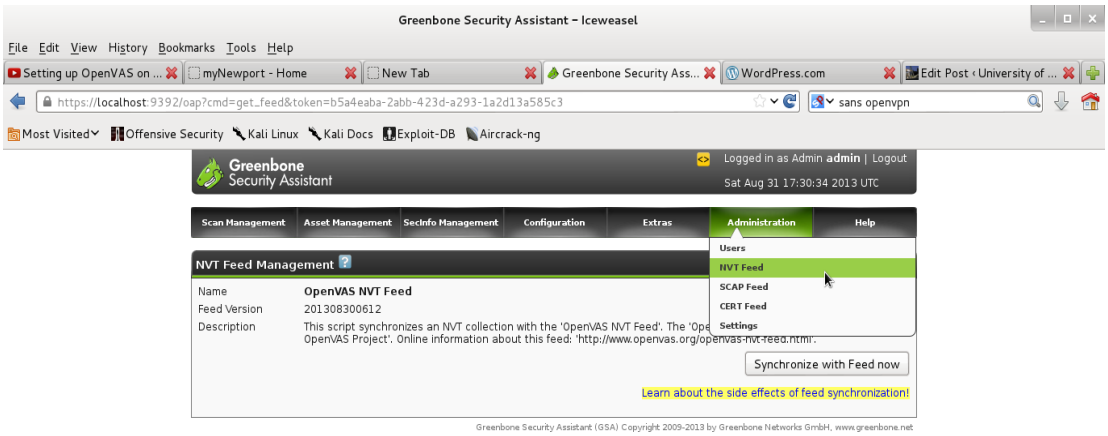
Default username = admin
Password (whatever you entered during setup)
******

OpenVAS Security Assistant screen (Hermione Granger wizard appears)



Update your Vulnerability Database Feeds

Administration > NVT Feed > Synchronise with Feed Now



This step is critical. if you do not update the vulnerability database feeds, it will generate errors later on.

Administration > NVT Feed

Administration > SCAP Database Feed (these are xml files for the reports)

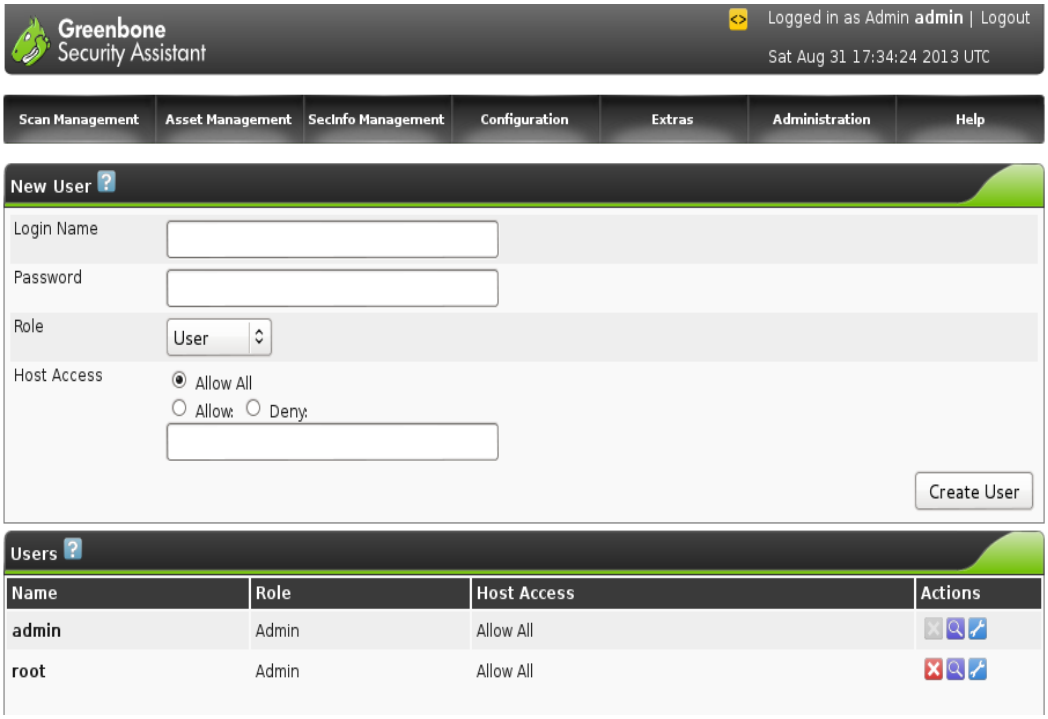
Administration > Cert Feed

****

Add Users

Administration > Users

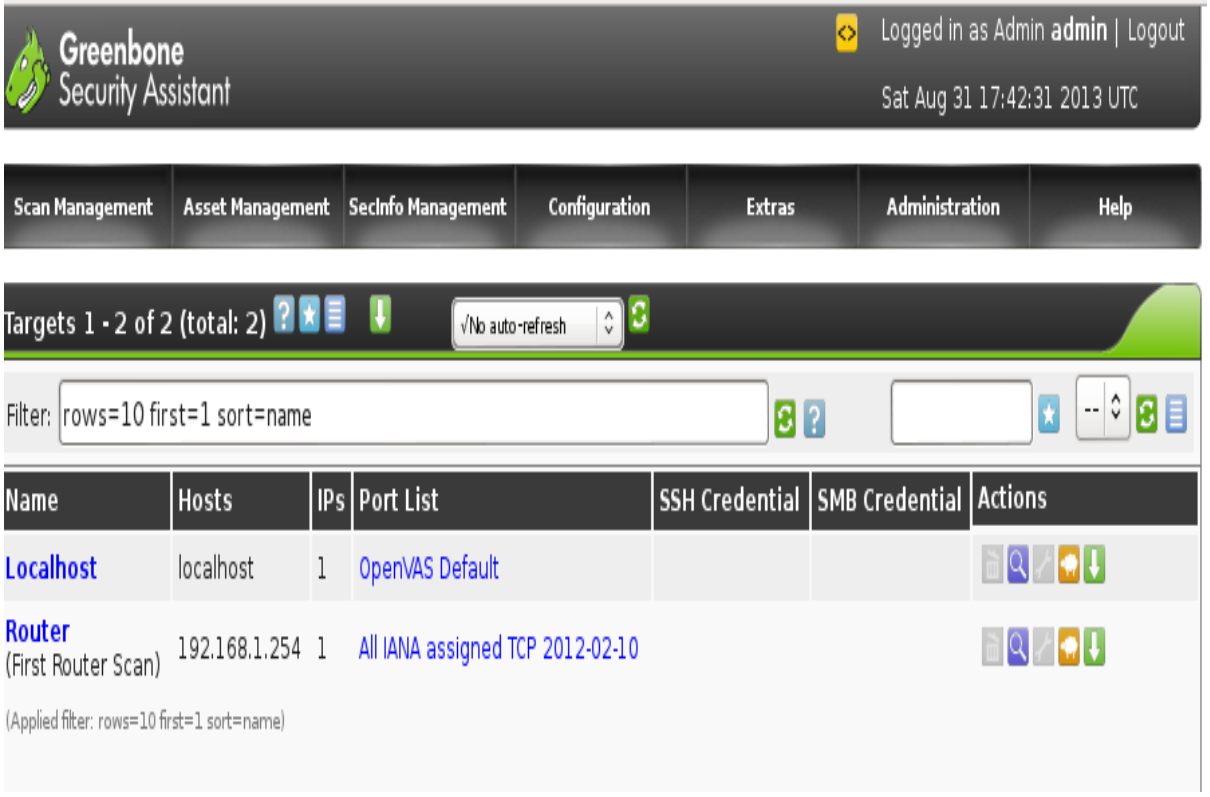
Add Users



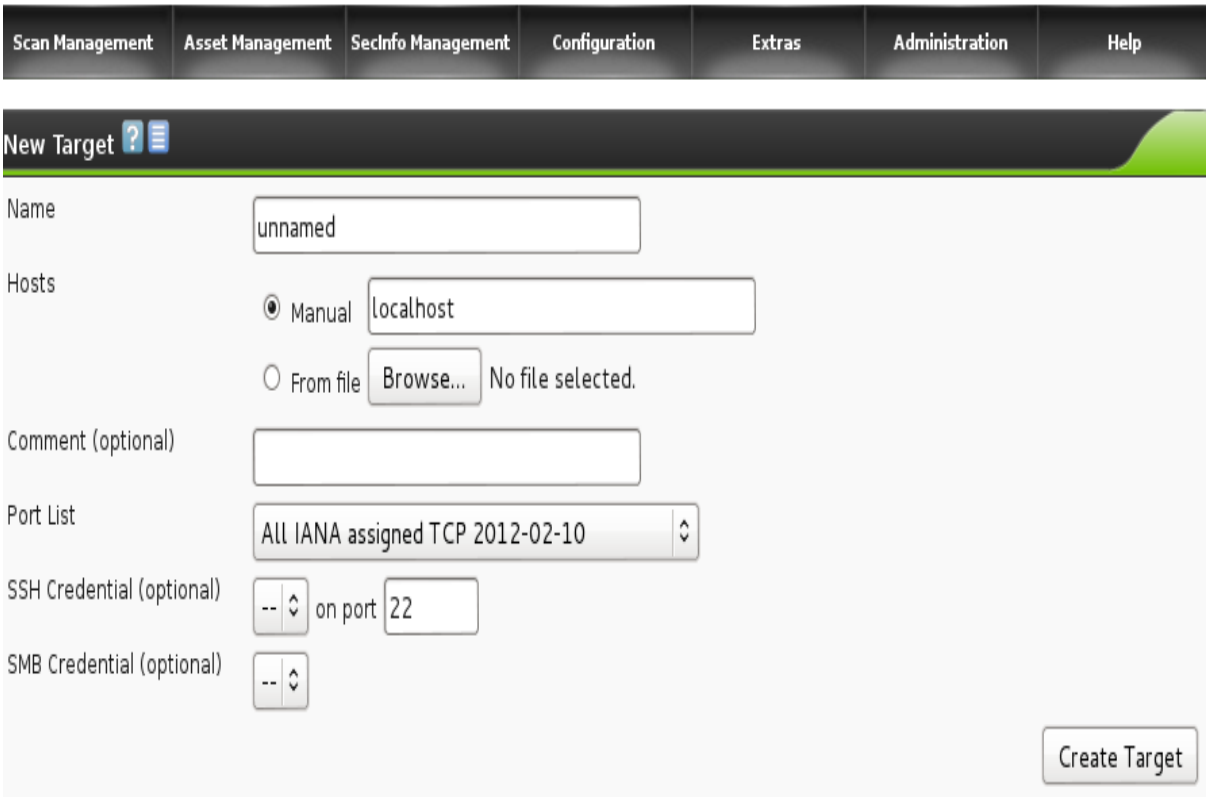
Set Targets to Scan

Configuration > Targets

Localhost will be there by default.
Add your router as a target eg 192.168.1.1 or 192.168.1.254

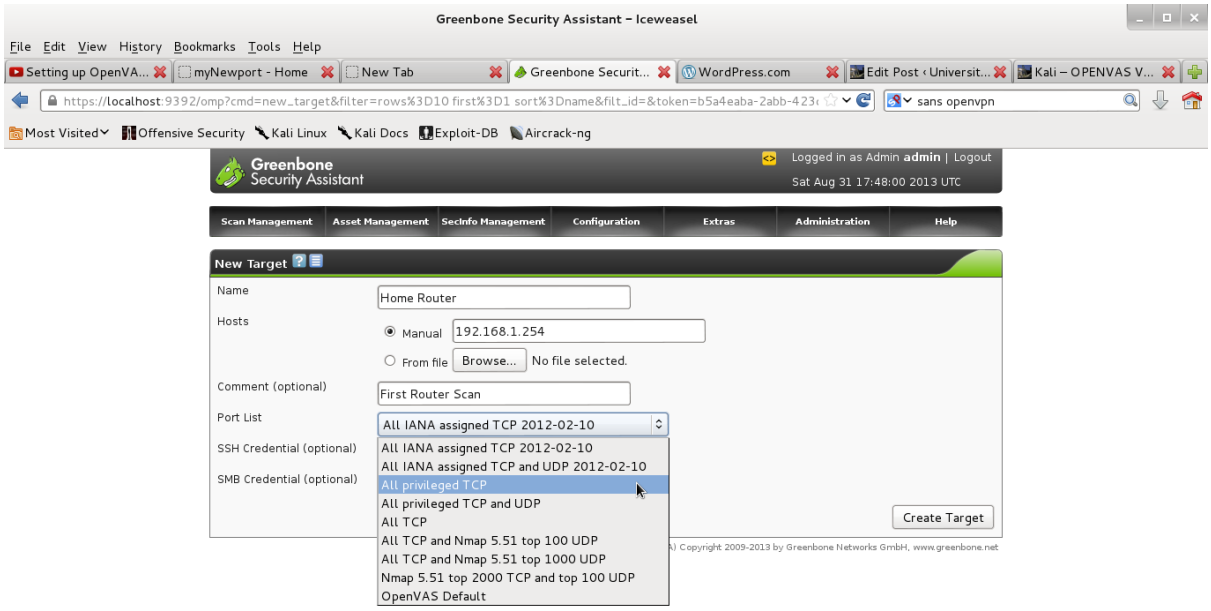


Look for the Blue box with a White star – click the star
White star = New Target



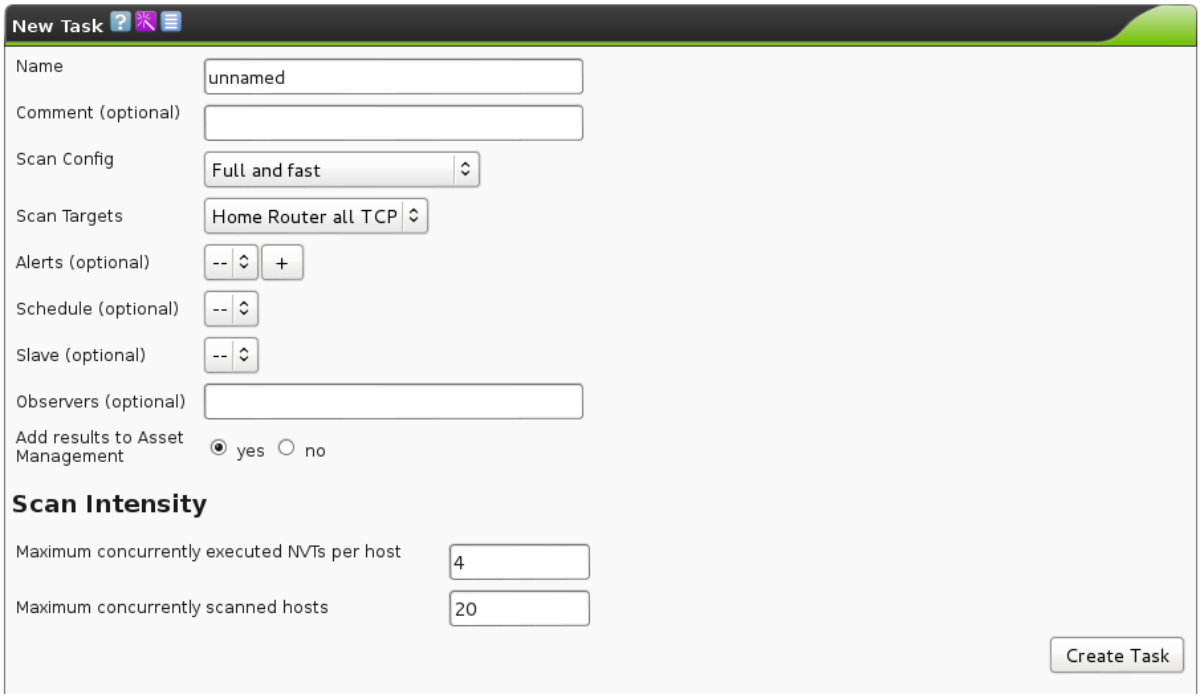
Enter IP of Router, and port options (eg all TCP)

Create Target Button



Step 6 – Create a Task

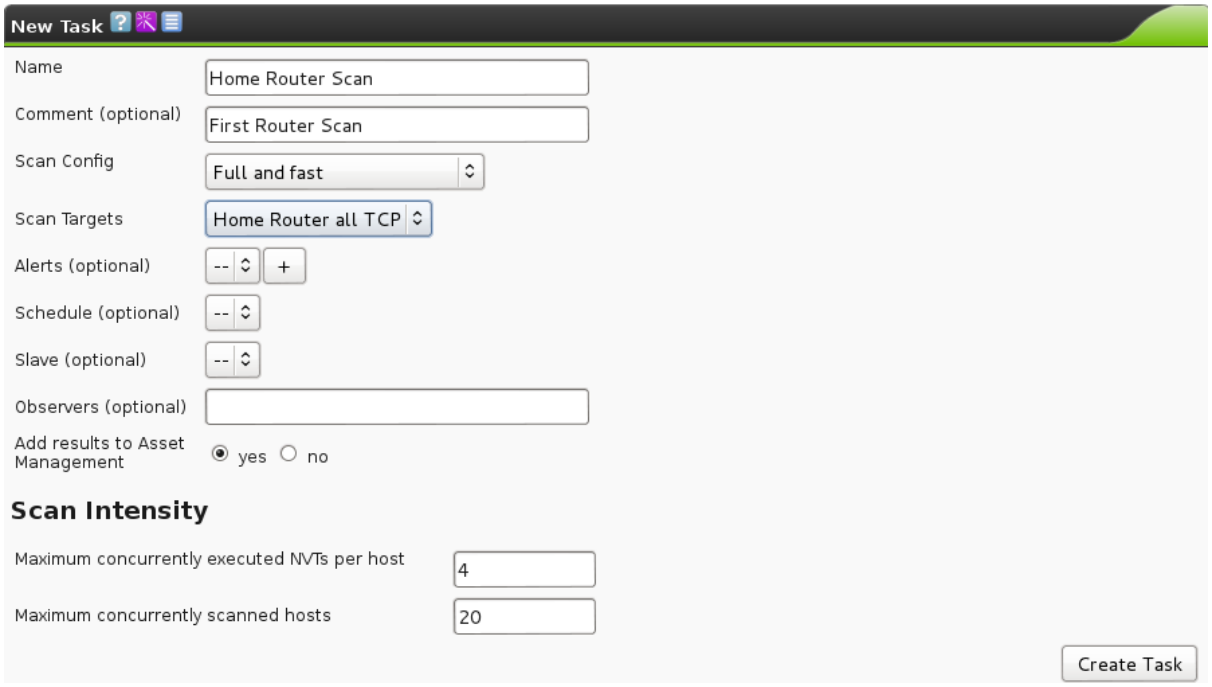
Scan Management > New Task



Home Router scan

Create Task Button

Scan Config = Full and Fast

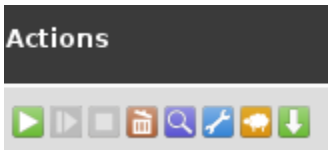


New status (Green)

Name	Status	Total	Reports			Trend	Actions
			First	Last	Threat		
Home Router Scan (First Router Scan)	New						
Router (First Router Scan)	Done	1		Aug 6 2013	High		

(Applied filter: apply_overrides=1 first=1 rows=10 sort=name)

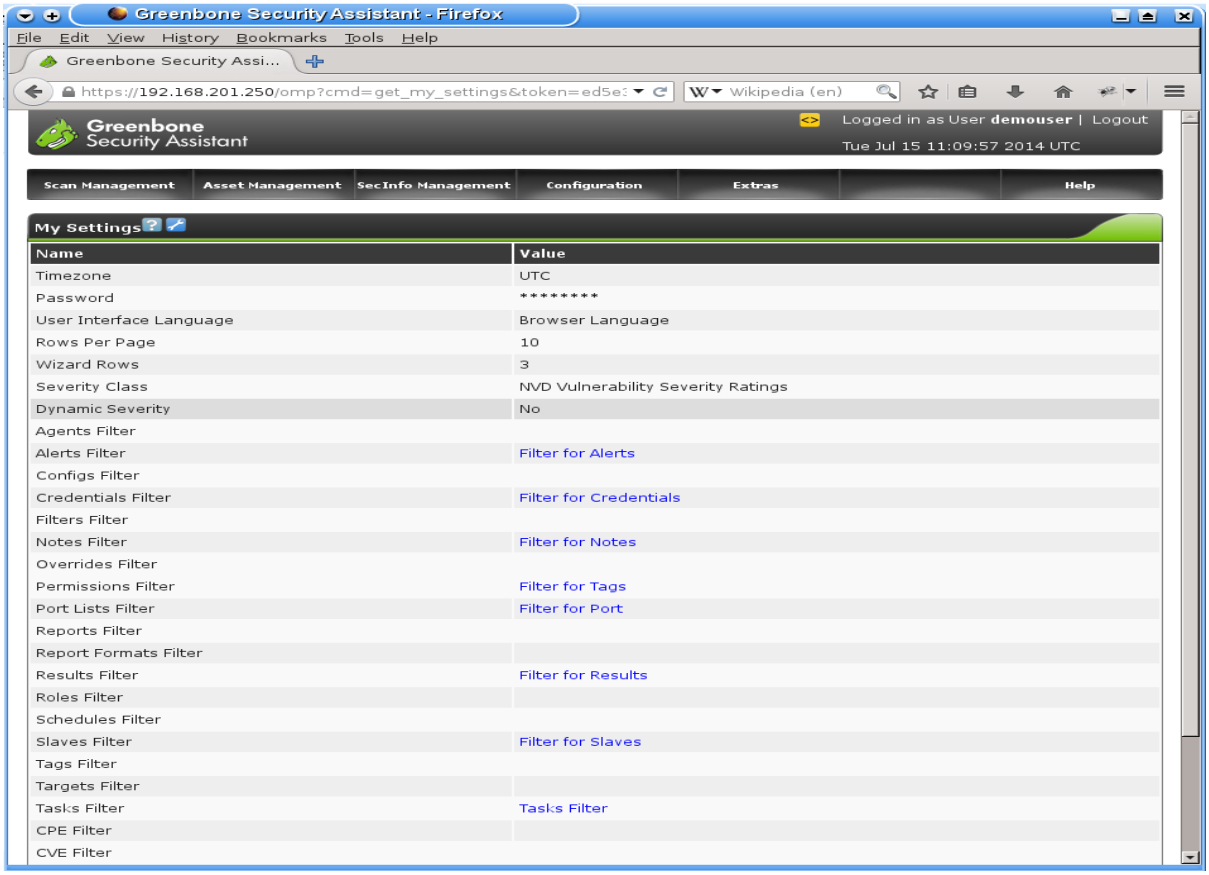
Green Arrow to Run this new task

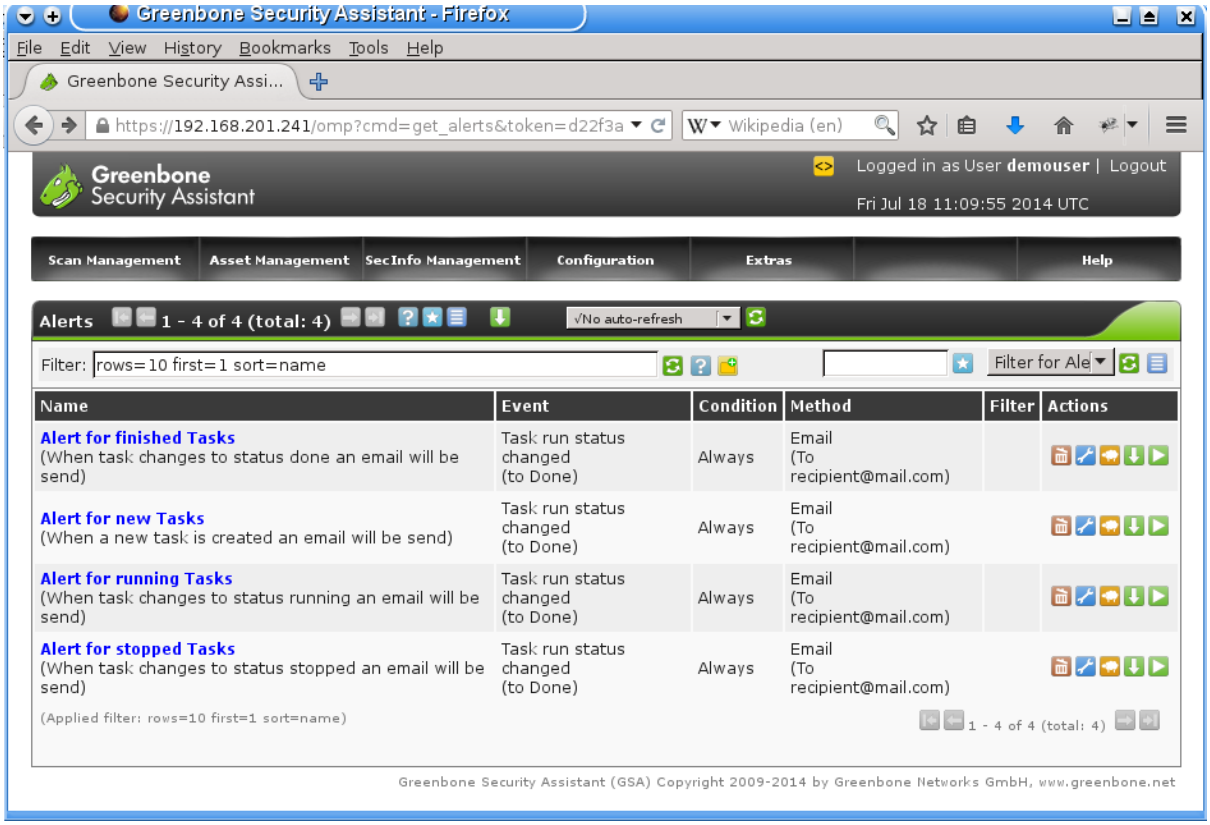


To watch LIVE

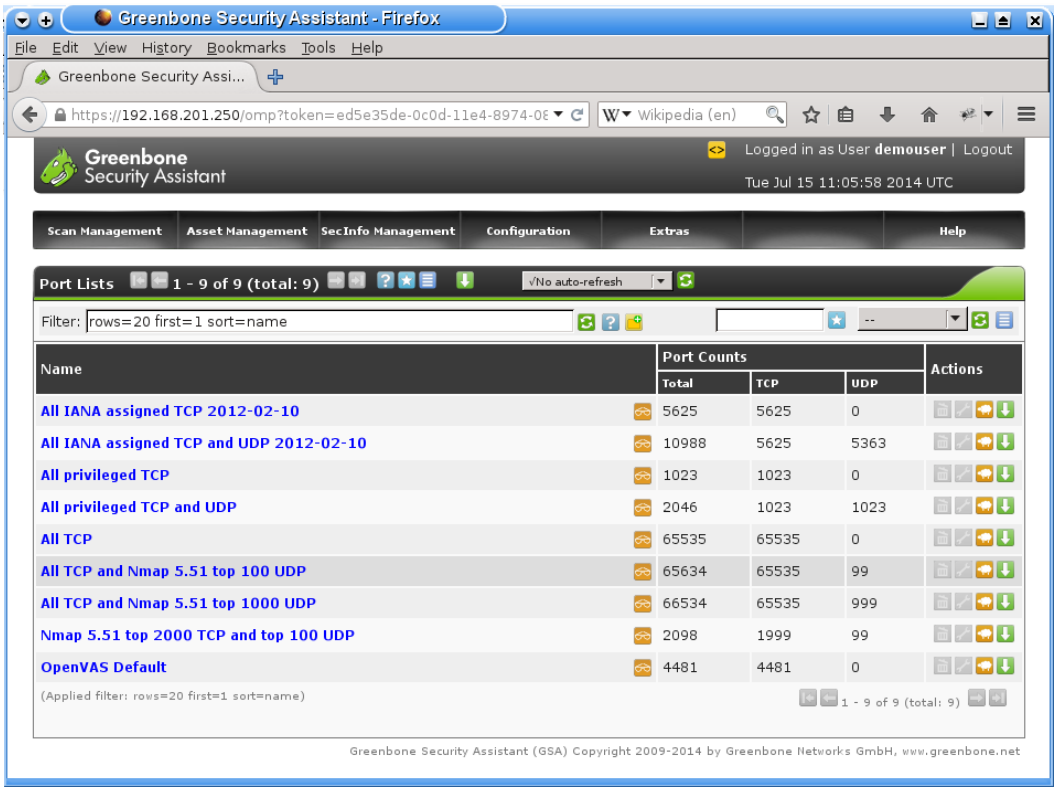
Set No Refresh dropdown box – to 30 seconds

Other Activities




Greenbone Security Assistant - Firefox
File Edit View History Bookmarks Tools Help
Greenbone Security Assi...
https://192.168.201.241/omp?cmd=get_alerts&token=d22f3a
Wikipedia (en)
Greenbone Security Assistant
Logged in as User demouser | Logout
Fri Jul 18 11:09:55 2014 UTC
Scan Management
Asset Management
SecInfo Management
Configuration
Extras
Help
Alerts
1 - 4 of 4 (total: 4)
No auto-refresh
Filter: rows=10 first=1 sort=name
Filter for Ale
Name
Event
Condition
Method
Filter
Actions
Alert for finished Tasks
(When task changes to status done an email will be send)
Task run status changed (to Done)
Always
Email (To recipient@mail.com)
Alert for new Tasks
(When a new task is created an email will be send)
Task run status changed (to Done)
Always
Email (To recipient@mail.com)
Alert for running Tasks
(When task changes to status running an email will be send)
Task run status changed (to Done)
Always
Email (To recipient@mail.com)
Alert for stopped Tasks
(When task changes to status stopped an email will be send)
Task run status changed (to Done)
Always
Email (To recipient@mail.com)
(Applied filter: rows=10 first=1 sort=name)
1 - 4 of 4 (total: 4)
Greenbone Security Assistant (GSA) Copyright 2009-2014 by Greenbone Networks GmbH, www.greenbone.net

Port List


Greenbone Security Assistant - Firefox
File Edit View History Bookmarks Tools Help
Greenbone Security Assi...
https://192.168.201.250/omp?token=ed5e35de-0c0d-11e4-8974-08
Wikipedia (en)
Greenbone Security Assistant
Logged in as User demouser | Logout
Tue Jul 15 11:05:58 2014 UTC
Scan Management
Asset Management
SecInfo Management
Configuration
Extras
Help
Port Lists
1 - 9 of 9 (total: 9)
No auto-refresh
Filter: rows=20 first=1 sort=name
Name
Port Counts
Total
TCP
UDP
Actions
All IANA assigned TCP 2012-02-10 5625 5625 0
All IANA assigned TCP and UDP 2012-02-10 10988 5625 5363
All privileged TCP 1023 1023 0
All privileged TCP and UDP 2046 1023 1023
All TCP 65535 65535 0
All TCP and Nmap 5.51 top 100 UDP 65634 65535 99
All TCP and Nmap 5.51 top 1000 UDP 66534 65535 999
Nmap 5.51 top 2000 TCP and top 100 UDP 2098 1999 99
OpenVAS Default 4481 4481 0
(Applied filter: rows=20 first=1 sort=name)
1 - 9 of 9 (total: 9)
Greenbone Security Assistant (GSA) Copyright 2009-2014 by Greenbone Networks GmbH, www.greenbone.net

Conclusion

Thus we have studied about OpenVAS system.

Lab #11

DVWA

Aim: Introduction to DVWA
Software: DVWA
Description:

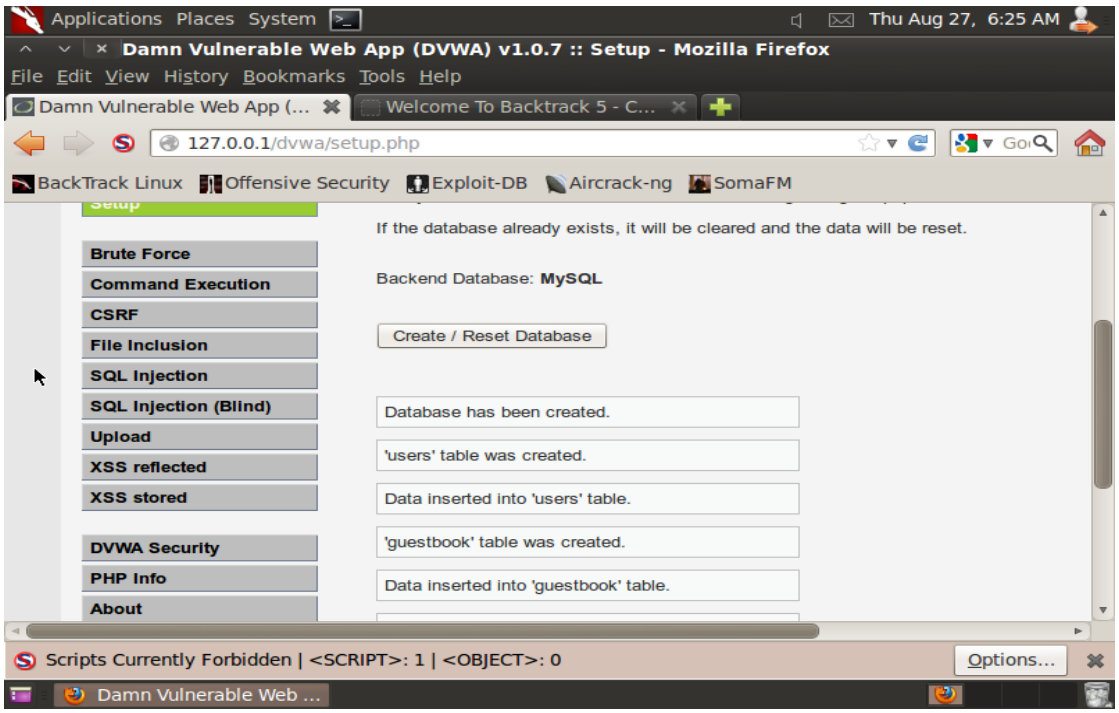
DVWA

1. Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable.
2. 2. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment.

Configuring DVWA

Follow steps below to configure DVWA

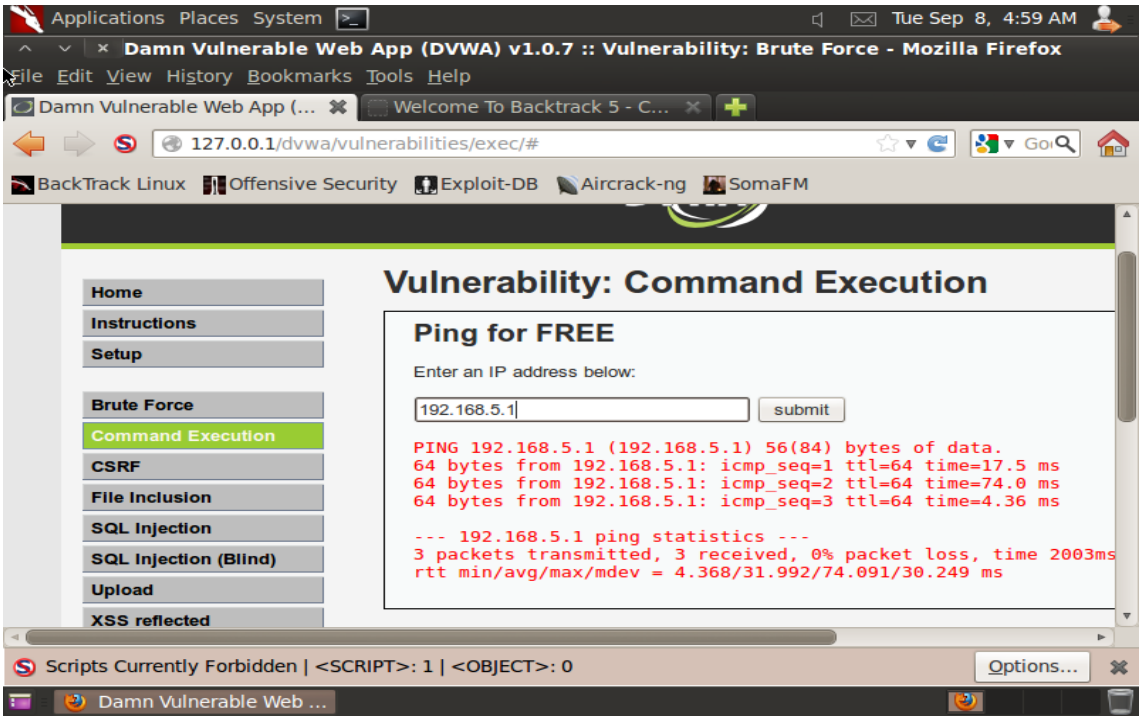
- Create Database



- Login with Username :- admin
  Password: password

- Click on DVWA Security
- Select Low
- Click Submit

- Click on Command Execution
- 192.168.1.106
- Click Submit



- 192.168.1.106; cat /etc/passwd
- Click Submit

Conclusion

Thus we have studied about DVWA system.