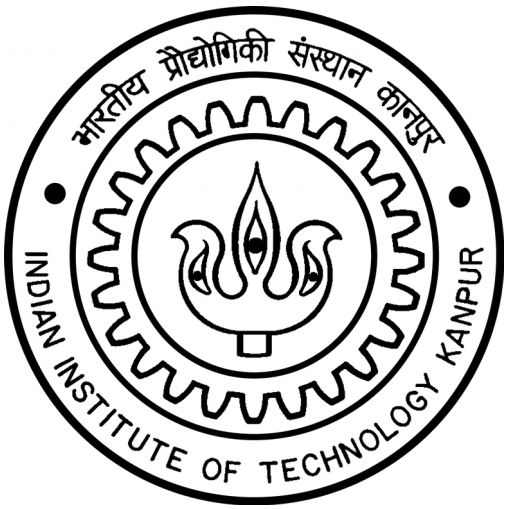


IPV4 TRAFFIC GENERATOR

WINTER INTERNSHIP PROJECT

SUBMITTED BY

SURAMYA GUPTA

STUDENT OF

JAYPEE UNIVERSITY OF ENGINEERING & TECHNOLOGY,
GUNA, M.P.

CERTIFICATE

This is to certify that, Ms. Suramya Gupta, student of Jaypee University Of Engineering & Technology worked for her winter internship project on **IPV4 TRAFFIC GENERATOR** under my supervision during the end of Vth semester of Computer Science and Engineering B.tech.

(Mr. Navpreet Singh)
Chief Engineer
Computer Centre
Indian Institute of Technology Kanpur

ACKNOWLEDGEMENT

Winter Internship is a great opportunity for learning and professional development. Therefore, I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it. I am also grateful for having a chance to meet so many wonderful people and professionals who led me through this internship period.

I am grateful to have Mr. Navpreet Singh, Chief Computer Engineer, Computer Centre, IIT Kanpur, for providing me this golden opportunity to work on this project and for his continuous support and guidance.

I also take the opportunity to express my gratitude and indebtedness to Mr. Saurabh Malhotra for his inestimable help, regular supervision of my work and unfailing guidance.

INDEX

Heading	Page Number
Objective	5
Introduction	6
LAMP Architecture	7
SENDIP	18
IPV4	22
Protocols	27
User Interface	38
Result and Application	47

OBJECTIVE

A packet generator or packet builder is a type of software that generates random packets or allows the user to construct detailed custom packets. Depending on the network medium and operating system, packet generators utilize raw sockets, NDIS function calls, or direct access to the network adapter kernel-mode driver.
This is useful for testing implementations of IP stacks for bugs and security vulnerabilities.

The objective of this project is to design and develop a network packet generator that supports basic features such as sending a packet continuous or non-continuous, any protocol such as UDP, TCP, ICMP ,and of respective packet length and payload option could be user-defined or random using any source port and destination port by using ip addresses.

Currently there no packet generators that offer low cost and high performance.The goal is to offer a simple, low cost, and high performance packet generator that can meet the basic testing needs of network industry.

INTRODUCTION

The network research community has a persistent need to evaluate new algorithms, systems and protocols using tools that create a range of test conditions similar to those experienced in live deployment and ensure reproducible results . Having appropriate tools for generating scalable, tunable and representative network traffic is therefore of fundamental importance. Such tools are critical in laboratory test environments where they can be used to evaluate behaviour and performance of new systems and real networking hardware. They are also critical in emulation environments and simulation environments where representative back- ground traffic is needed. Without the capability to consistently create realistic network test conditions, new systems run the risk of unpredictable behaviour and unacceptable performance when deployed in live environments.

A number of successful application-specific workload generators have been developed.These tools typically focus on generating application-level request sequences that result in network traffic that has the same statistical properties as live traffic from the modelled application.

The objective is to a new network traffic generator capable of recreating IP traffic flows (where flow is defined as a series of packets between a given IP/port pair using a specific trans- port protocol) representative of those observed at routers in the Internet. We are aware of no other tools that target representative traffic generation at the IP flow level. Our approach is to abstract flow-level traffic generation into a series of application-independent file transfers that use either TCP or UDP for transport.

LAMP ARCHITECTURE

LAMP is an archetypal model of web service stacks, named as ancronym of the names of its original four open-source components: the Linux Operating System, the Apache HTTP Server, the MYSQL Relational Database Management System (RDBMS), and the PHP Language. The LAMP components are largely interchangeable and not limited to the original selection. As a solution stack, LAMP is suitable for building dynamic web pages and applications.

Linux

Linux is a Unix-like computer operating system assembled under the model of free and open source software development and distribution. Most Linux distributions, as collections of software based around the Linux kernel and often around a package management system, provide complete LAMP setups through their packages. According to W3Techs in October 2013, 58.5% of web server market share was shared between Debian and Ubuntu, while RHEL, Fedora and CentOS together shared 37.3%.

A Linux-based system is a modular Unix-like operating system, deriving much of its basic design from principles established in Unix during the 1970s and 1980s. Such a system uses a monolithic kernel, the Linux kernel, which handles process control, networking, access to the peripherals, and file systems. Device drivers are either integrated directly with the kernel, or added as modules that are loaded while the system is running.

The GNU userland is a key part of most systems based on the Linux kernel, with Android being the notable exception. The Project's implementation of the C library functions as a wrapper for the system calls of the Linux kernel necessary to the kernel-userspace interface, the toolchain is a broad collection of programming tools vital to Linux development (including the compilers used to build the Linux kernel itself), and the coreutils implement many basic Unix tools. The project also develops a popular CLI shell. The graphical user interface (or GUI) used by most Linux systems is built on top of an implementation of the X Window System. More recently, the Linux community seeks to advance to Wayland as the new display server protocol in place of X11. Many other open-source software projects contribute to Linux systems.

Installed components of a Linux system include the following:

- A bootloader, for example GNU GRUB, LILO, SYSLINUX, or Gummiboot. This is a program that loads the Linux kernel into the computer's main memory, by being executed by the computer when it is turned on and after the firmware initialization is performed.
- An init program, such as the traditional sysvinit and the newer systemd, OpenRC and Upstart. This is the first process launched by the Linux kernel, and is at the root of the process tree: in other terms, all processes are launched through init. It starts processes such as system services and login prompts (whether graphical or in terminal mode).
- Software libraries, which contain code that can be used by running processes. On Linux systems using ELF-format executable files, the dynamic linker that

manages use of dynamic libraries is known as ld-linux.so. If the system is set up for the user to compile software themselves, header files will also be included to describe the interface of installed libraries. Besides the most commonly used software library on Linux systems, the GNU C Library (glibc), there are numerous other libraries, such as SDL and Mesa.

- C standard library is the library needed to run C programs on a computer system, with the GNU C Library being the standard. For embedded systems, alternatives such as the EGLIBC (a glibc fork once used by Debian) and uClibc (which was designed for uClinux) have been developed, although both are no longer maintained. Android uses its own C library, Bionic.
- Basic Unix commands, with GNU coreutils being the standard implementation. Alternatives exist for embedded systems, such as the copyleft BusyBox, and the BSD-licensed Toybox.
- Widget toolkits are the libraries used to build graphical user interfaces (GUIs) for software applications. Numerous widget toolkits are available, including GTK+ and Clutter developed by the GNOME project, Qt developed by the Qt Project and led by Digia, and Enlightenment Foundation Libraries (EFL) developed primarily by the Enlightenment team.
- A package management system, such as dpkg and RPM. Alternatively packages can be compiled from binary or source tarballs.
- User interface programs such as command shells or windowing environments.

Apache

The role of LAMP's web server has been traditionally supplied by Apache, and has since included other web servers such as Nginx.
The Apache HTTP Server has been the most popular web server on the public Internet. In June 2013, Netcraft estimated that Apache served 54.2% of all active websites and 53.3% of the top servers across all domains.[6] In June 2014, Apache was estimated to serve 52.27% of all active websites, followed by nginx with 14.36%.[7]
Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. Released under the Apache License, Apache is open-source software. A wide variety of features are supported, and many of them are implemented as compiled modules which extend the core functionality of Apache. These can range from server-side programming language support to authentication schemes.

FEATURES

Apache supports a variety of features, many implemented as compiled modules which extend the core functionality. These can range from server-side programming language support to authentication schemes. Some common language interfaces support Perl, Python, Tcl and PHP. Popular authentication modules include mod_access, mod_auth, mod_digest, and mod_auth_digest, the successor to mod_digest. A sample of other features include Secure Sockets Layer and Transport Layer Security support (mod_ssl), a proxy module (mod_proxy), a URL rewriting

module (mod_rewrite), custom log files (mod_log_config), and filtering support (mod_include and mod_ext_filter). Popular compression methods on Apache include the external extension module, mod_gzip, implemented to help with reduction of the size (weight) of Web pages served over HTTP. ModSecurity is an open source intrusion detection and prevention engine for Web applications. Apache logs can be analyzed through a Web browser using free scripts, such as AWStats/W3Perl or Visitors.
Virtual hosting allows one Apache installation to serve many different Web sites. For example, one machine with one Apache installation could simultaneously serve www.example.com, www.example.org, test47.test-server.example.edu, etc.
Apache features configurable error messages, DBMS-based authentication databases, and content negotiation. It is also supported by several graphical user interfaces (GUIs).
It supports password authentication and digital certificate authentication. Because the source code is freely available, anyone can adapt the server for specific needs, and there is a large public library of Apache add-ons.

MYSQL

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality. MySQL is a central component of the LAMP open-source web application software stack (and other "AMP" stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python". Applications that use the MySQL database include: TYPO3, MODx, Joomla, WordPress, Simple Machines Forum, phpBB, MyBB, and Drupal. MySQL is also used in many high-profile, large-scale websites, including Google (though not for searches), Facebook, Twitter, Flickr, and YouTube.

Features

MySQL is offered under two different editions: the open source MySQL Community Server and the proprietary Enterprise Server. MySQL Enterprise Server is differentiated by a series of proprietary extensions which install as server plugins, but otherwise shares the version numbering system and is built from the same code base.

Major features as available in MySQL 5.6:

- A broad subset of ANSI SQL 99, as well as extensions
- Cross-platform support
- Stored procedures, using a procedural language that closely adheres to SQL/PSM
- Triggers
- Cursors
- Updatable views
- Online Data Definition Language (DDL) when using the InnoDB Storage Engine.
- Information schema
- Performance Schema that collects and aggregates statistics about server execution and query performance for monitoring purposes.
- A set of SQL Mode options to control runtime behavior, including a strict mode to better adhere to SQL standards.
- X/Open XA distributed transaction processing (DTP) support; two phase commit as part of this, using the default InnoDB storage engine
- Transactions with savepoints when using the default InnoDB Storage Engine. The NDB Cluster Storage Engine also supports transactions.
- ACID compliance when using InnoDB and NDB Cluster Storage Engines
- SSL support
- Query caching
- Sub-SELECTs (i.e. nested SELECTs)
- Built-in replication support (i.e., master-master replication and master-slave replication) with one master per slave, many slaves per master. Multi-master replication is provided in MySQL Cluster, and multi-master support can be added to unclustered configurations using Galera Cluster.

- Full-text indexing and searching
- Embedded database library
- Unicode support
- Partitioned tables with pruning of partitions in optimizer
- Shared-nothing clustering through MySQL Cluster
- Multiple storage engines, allowing one to choose the one that is most effective for each table in the application.
- Native storage engines InnoDB, MyISAM, Merge, Memory (heap), Federated, Archive, CSV, Blackhole, NDB Cluster.
- Commit grouping, gathering multiple transactions from multiple connections together to increase the number of commits per second.

The developers release minor updates of the MySQL Server approximately every two months. The sources can be obtained from MySQL's website or from MySQL's GitHub repository, both under the GPL license.

Limitations

When using some storage engines other than the default of InnoDB, MySQL does not comply with the full SQL standard for some of the implemented functionality, including foreign key references. Check constraints are parsed but ignored by all storage engines.

Up until MySQL 5.7, triggers are limited to one per action / timing, meaning that at most one trigger can be defined to be executed after an `INSERT` operation, and one before `INSERT` on the same table. No triggers can be defined on views.

MySQL database's inbuilt functions like `UNIX_TIMESTAMP()` will return `0` after 03:14:07 UTC on 19 January 2038. Recently, there had been an attempt to solve the problem which had been assigned to the internal queue.

PHP

PHP: Hypertext Preprocessor (or simply PHP) is a server-side scripting language designed for Web development, but also used as a general-purpose programming language. It was originally created by Rasmus Lerdorf in 1994, the PHP reference implementation is now produced by The PHP Group. PHP originally stood for Personal Home Page, but it now stands for the recursive acronym PHP: Hypertext Preprocessor.
PHP code may be embedded into HTML code, or it can be used in combination with various web template systems, web content management systems, and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalone graphical applications.
The standard PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web

servers on almost every operating system and platform, free of charge.

PHP is a general-purpose scripting language that is especially suited to server-side web development, in which case PHP generally runs on a web server. Any PHP code in a requested file is executed by the PHP runtime, usually to create dynamic web page content or dynamic images used on websites or elsewhere. It can also be used for command-line scripting and client-side graphical user interface (GUI) applications. PHP can be deployed on most web servers, many operating systems and platforms, and can be used with many relational database management systems (RDBMS). Most web hosting providers support PHP for use by their clients. It is available free of charge, and the PHP Group provides the complete source code for users to build, customize and extend for their own use.

Dynamic web page: example of server-side scripting (PHP and MySQL).

PHP acts primarily as a filter, taking input from a file or stream containing text and/or PHP instructions and outputting another stream of data. Most commonly the output will be HTML, although it could be JSON, XML or binary data such as image or audio formats. Since PHP 4, the PHP parser compiles input to produce bytecode for processing by the Zend Engine, giving improved performance over its interpreter predecessor.
Originally designed to create dynamic web pages, PHP now focuses mainly on server-side scripting, and it is similar to other server-side scripting languages that provide dynamic

content from a web server to a client, such as Microsoft's ASP.NET, Sun Microsystems' JavaServer Pages, and mod_perl. PHP has also attracted the development of many software frameworks that provide building blocks and a design structure to promote rapid application development (RAD). Some of these include PRADO, CakePHP, Symfony, CodeIgniter, Laravel, Yii Framework, Phalcon and Zend Framework, offering features similar to other web frameworks.

The LAMP architecture has become popular in the web industry as a way of deploying web applications.PHP is commonly used as the P in this bundle alongside Linux, Apache and MySQL, although the P may also refer to Python, Perl, or some mix of the three. Similar packages, WAMP and MAMP, are also available for Windows and OS X, with the first letter standing for the respective operating system. Although both PHP and Apache are provided as part of the Mac OS X base install, users of these packages seek a simpler installation mechanism that can be more easily kept up to date.

SENDIP

SendIP is a command-line tool to send arbitrary IP packets. It has a large number of options to specify the content of every header of a RIP, RIPng, BGP, TCP, UDP, ICMP, or raw IPv4/IPv6 packet. It also allows any data to be added to the packet. Checksums can be calculated automatically, but if you wish to send out wrong checksums, that is supported too.

Installing sendip on ubuntu

Open the terminal and run the following command

sudo apt-get install sendip

Syntax

sendip [-v] [-d data] [-h] [-f datafile] [-p module] [module options] hostname

-d data -- add this data as a string to the end of the packet.

Data can be rN to generate N random(ish) data bytes; 0x or 0X followed by hex digits; 0 followed by octal digits; any other stream of bytes.

-f datafile -- read packet data from file

-h -- print this message

-p module -- load the specified module

-v -- be verbose

IPV4

-is x -- Source IP address Default: 127.0.0.1
-id x -- Desitnation IP address Default: Correct
-ih x -- IP header length Default: Correct
-iv x -- IP version Default: 4
-iy x -- IP type of service Default: 0
-il x -- Total IP packet length Default: Correct
-ii x -- IP packet ID Default: Random
-ifr x -- IP reservced flag Default: 0 (options are 0,1,r)
-ifd x -- IP don't fragment flag Default: 0 (options are 0,1,r)
-ifm x -- IP more fragments flag Default: 0 (options are 0,1,r)
-if x -- IP fragment offset Default: 0
-it x -- IP time to live Default: 255
-ip x -- IP protcol Default: 0, or set by underlying protocol
-ic x -- IP checksum Default: Correct
-ionum x -- IP option as string of hex bytes
-ioeol -- IP option: end of list
-ionop -- IP option: no-op
-iorr x -- IP option: record route. Format: pointer:addr1:addr2:…
-iots x -- IP option: timestamp. Format: pointer:overflow:flag:
(ip1:)ts1:(ip2:)ts2:...
-iolsr x -- IP option: loose source route. Format:
pointer:addr1:addr2:...
-iosid x -- IP option: stream identifier
-iossr x -- IP option: strict source route. Format:
pointer:addr1:addr2:...

UDP

-us x -- UDP source port Default: 0
-ud x -- UDP destination port Default: 0
-ul x -- UDP packet legnth Default: Correct
-uc x -- UDP checksum Default: Correct

TCP

-ts x -- TCP source port Default: 0
-td x -- TCP destination port Default: 0
-tn x -- TCP sequence number Default: Random
-ta x -- TCP ack number Default: 0
-tt x -- TCP data offset Default: Correct
-tr x -- TCP header reserved field EXCLUDING ECN and CWR bits Default: 0
-tfe x -- TCP ECN bit Default: 0 (options are 0,1,r)
-tfc x -- TCP CWR bit Default: 0 (options are 0,1,r)
-tfu x -- TCP URG bit Default: 0, or 1 if -tu specified (options are 0,1,r)
-tfa x -- TCP ACK bit Default: 0, or 1 if -ta specified (options are 0,1,r)
-tfp x -- TCP PSH bit Default: 0 (options are 0,1,r)
-tfr x -- TCP RST bit Default: 0 (options are 0,1,r)
-tfs x -- TCP SYN bit Default: 1 (options are 0,1,r)
-tff x -- TCP FIN bit Default: 0 (options are 0,1,r)
-tw x -- TCP window size Default: 65535
-tc x -- TCP checksum Default: Correct
-tu x -- TCP urgent pointer Default: 0
-tonum x -- TCP option as string of hex bytes
-toeol -- TCP option: end of list

-tonop -- TCP option: no op
-tomss x -- TCP option: maximum segment size
-towscale x -- TCP option: window scale
-tosackok -- TCP option: allow selective ack
-tosack x -- TCP option: selective ack (rfc2018), format is l_edge1:r_edge1,l_edge2:r_edge2...
-tots x -- TCP option: timestamp (rfc1323), format is tsval:tsecr

ICMP

-ct x -ICMP message type Default: ICMP_ECHO
-cd x -- ICMP code Default: 0
-cc x -- ICMP checksum Default: Correct

IPV4

Internet Protocol version 4 (IPv4) is the fourth version of the Internet Protocol (IP). It is one of the core protocols of standards-based internetworking methods in the Internet, and was the first version deployed for production in the ARPANET in 1983. It still routes most Internet traffic today,despite the ongoing deployment of a successor protocol, IPv6. IPv4 is described in IETF publication RFC 791 (September 1981), replacing an earlier definition (RFC 760, January 1980).
IPv4 is a connectionless protocol for use on packet-switched networks. It operates on a best effort delivery model, in that it does not guarantee delivery, nor does it assure proper sequencing or avoidance of duplicate delivery. These aspects, including data integrity, are addressed by an upper layer transport protocol, such as the Transmission Control Protocol (TCP).

Addressing

IPv4 uses 32-bit addresses which limits the address space to 4294967296 (232) addresses.
IPv4 reserves special address blocks for private networks (~18 million addresses) and multicast addresses (~270 million addresses).

Packet structure

An IP packet consists of a header section and a data section.
An IP packet has no data checksum or any other footer after the data section. Typically the link layer encapsulates IP packets in frames with a CRC footer that detects most errors, and typically the end-to-end TCP layer checksum detects most other errors.
Header

The IPv4 packet header consists of 14 fields, of which 13 are required. The 14th field is optional and aptly named: options. The fields in the header are packed with the most significant byte first, and for the diagram and discussion, the most significant bits are considered to come first (MSB 0 bit numbering). The most significant bit is numbered 0, so the version field is actually found in the four most significant bits of the first byte.

Version

The first header field in an IP packet is the four-bit version field. For IPv4, this is always equal to 4.

Internet Header Length (IHL)

The Internet Header Length (IHL) field has 4 bits, which is the number of 32-bit words. Since an IPv4 header may contain a variable number of options, this field specifies the size of the header (this also coincides with the offset to the data). The minimum value for this field is 5,[23] which indicates a length of 5 × 32 bits = 160 bits = 20 bytes. As a 4-bit field, the maximum value is 15 words (15 × 32 bits, or 480 bits = 60 bytes).

Differentiated Services Code Point (DSCP)

Originally defined as the Type of service (ToS) field. This field is now defined by RFC 2474 (updated by RFC 3168 and RFC 3260) for Differentiated services (DiffServ). New technologies are emerging that require real-time data streaming and therefore make use of the DSCP field. An example is Voice over IP (VoIP), which is used for interactive data voice exchange.

Explicit Congestion Notification (ECN)

This field is defined in RFC 3168 and allows end-to-end notification of network congestion without dropping packets. ECN

is an optional feature that is only used when both endpoints support it and are willing to use it. It is only effective when supported by the underlying network.

Total Length

This 16-bit field defines the entire packet size in bytes, including header and data. The minimum size is 20 bytes (header without data) and the maximum is 65,535 bytes. All hosts are required to be able to reassemble datagrams of size up to 576 bytes, but most modern hosts handle much larger packets. Sometimes links impose further restrictions on the packet size, in which case datagrams must be fragmented. Fragmentation in IPv4 is handled in either the host or in routers.

Identification

This field is an identification field and is primarily used for uniquely identifying the group of fragments of a single IP datagram. Some experimental work has suggested using the ID field for other purposes, such as for adding packet-tracing information to help trace datagrams with spoofed source addresses,[24] but RFC 6864 now prohibits any such use.

Flags

A three-bit field follows and is used to control or identify fragments. They are (in order, from most significant to least significant):

- bit 0: Reserved; must be zero.[note 1]
- bit 1: Don't Fragment (DF)
- bit 2: More Fragments (MF)

If the DF flag is set, and fragmentation is required to route the packet, then the packet is dropped. This can be used when sending packets to a host that does not have sufficient resources to handle fragmentation. It can also be used for Path MTU Discovery, either automatically by the host IP software, or manually using

diagnostic tools such as ping or traceroute. For unfragmented packets, the MF flag is cleared. For fragmented packets, all fragments except the last have the MF flag set. The last fragment has a non-zero Fragment Offset field, differentiating it from an unfragmented packet.

Fragment Offset

The fragment offset field is measured in units of eight-byte blocks. It is 13 bits long and specifies the offset of a particular fragment relative to the beginning of the original unfragmented IP datagram. The first fragment has an offset of zero. This allows a maximum offset of (213 – 1) × 8 = 65,528 bytes, which would exceed the maximum IP packet length of 65,535 bytes with the header length included (65,528 + 20 = 65,548 bytes).

Time To Live (TTL)

An eight-bit time to live field helps prevent datagrams from persisting (e.g. going in circles) on an internet. This field limits a datagram's lifetime. It is specified in seconds, but time intervals less than 1 second are rounded up to 1. In practice, the field has become a hop count—when the datagram arrives at a router, the router decrements the TTL field by one. When the TTL field hits zero, the router discards the packet and typically sends an ICMP Time Exceeded message to the sender. The program traceroute uses these ICMP Time Exceeded messages to print the routers used by packets to go from the source to the destination.

Protocol

This field defines the protocol used in the data portion of the IP datagram. The Internet Assigned Numbers Authority maintains a list of IP protocol numbers which was originally defined in RFC 790.

Header Checksum:

Main article: IPv4 header checksum

The 16-bit checksum field is used for error-checking of the header. When a packet arrives at a router, the router calculates the checksum of the header and compares it to the checksum field. If the values do not match, the router discards the packet. Errors in the data field must be handled by the encapsulated protocol. Both UDP and TCP have checksum fields.
When a packet arrives at a router, the router decreases the TTL field. Consequently, the router must calculate a new checksum.

Source address

This field is the IPv4 address of the sender of the packet. Note that this address may be changed in transit by a network address translation device.

Destination address

This field is the IPv4 address of the receiver of the packet. As with the source address, this may be changed in transit by a network address translation device.

PROTOCOLS

UDP

In computer networking, the User Datagram Protocol (UDP) is one of the core members of the Internet protocol suite. The protocol was designed by David P. Reed in 1980 and formally defined in RFC 768. With UDP, computer applications can send messages, in this case referred to as datagrams, to other hosts on an Internet Protocol (IP) network. Prior communications are not required in order to set up communication channels or data paths. UDP uses a simple connectionless communication model with a minimum of protocol mechanism. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram. It has no handshaking dialogues, and thus exposes the user's program to any unreliability of the underlying network; There is no guarantee of delivery, ordering, or duplicate protection. If error-correction facilities are needed at the network interface level, an application may use Transmission Control Protocol (TCP) or Stream Control Transmission Protocol (SCTP) which are designed for this purpose.
UDP is suitable for purposes where error checking and correction are either not necessary or are performed in the application; UDP avoids the overhead of such processing in the protocol stack. Time-sensitive applications often use UDP because dropping packets is preferable to waiting for packets delayed due to retransmission, which may not be an option in a real-time system.

UDP is a minimal message-oriented transport layer protocol that is documented in RFC 768. UDP provides no guarantees to the upper layer protocol for message delivery and the UDP layer retains no state of UDP messages once sent. For this reason, UDP sometimes is referred to as Unreliable Datagram Protocol.
A number of UDP's attributes make it especially suited for certain applications.

- It is transaction-oriented, suitable for simple query-response protocols such as the Domain Name System or the Network Time Protocol.
- It provides datagrams, suitable for modeling other protocols such as IP tunneling or Remote Procedure Call and the Network File System.
- It is simple, suitable for bootstrapping or other purposes without a full protocol stack, such as the DHCP and Trivial File Transfer Protocol.
- It is stateless, suitable for very large numbers of clients, such as in streaming media applications such as IPTV.
- The lack of retransmission delays makes it suitable for real-time applications such as Voice over IP, online games, and many protocols built on top of the Real Time Streaming Protocol.
- Because it supports multicast, it is suitable for broadcast information such as in many kinds of service discovery and shared information such as broadcast time or Routing Information Protocol.

Packet structure

UDP Header

The UDP header consists of 4 fields, each of which is 2 bytes (16 bits). The use of the fields "Checksum" and "Source port" is

optional in IPv4 (pink background in table). In IPv6 only the source port is optional (see below).

Source port number

This field identifies the sender's port when meaningful and should be assumed to be the port to reply to if needed. If not used, then it should be zero. If the source host is the client, the port number is likely to be an ephemeral port number. If the source host is the server, the port number is likely to be a well-known port number.

Destination port number

This field identifies the receiver's port and is required. Similar to source port number, if the client is the destination host then the port number will likely be an ephemeral port number and if the destination host is the server then the port number will likely be a well-known port number.

Length

A field that specifies the length in bytes of the UDP header and UDP data. The minimum length is 8 bytes because that is the length of the header. The field size sets a theoretical limit of 65,535 bytes (8 byte header + 65,527 bytes of data) for a UDP datagram. However the actual limit for the data length, which is imposed by the underlying IPv4 protocol, is 65,507 bytes (65,535 − 8 byte UDP header − 20 byte IP header).
In IPv6 jumbograms it is possible to have UDP packets of size greater than 65,535 bytes.RFC 2675 specifies that the length field is set to zero if the length of the UDP header plus UDP data is greater than 65,535.

Checksum

The checksum field may be used for error-checking of the header and data. This field is optional in IPv4, and mandatory in IPv6.The field carries all-zeros if unused.

TCP

The **Transmission Control Protocol** (**TCP**) is one of the main protocols of the Internet protocol suite. It originated in the initial network implementation in which it complemented the Internet Protocol (IP). Therefore, the entire suite is commonly referred to as *TCP/IP*. TCP provides reliable, ordered, and error-checked delivery of a stream of octets (bytes) between applications running on hosts communicating via an IP network. Major internet applications such as the World Wide Web, email, remote administration, and file transfer rely on TCP. Applications that do not require reliable data stream service may use the User Datagram Protocol (UDP), which provides a connectionless datagram service that emphasizes reduced latency over reliability.

TCP segment structure

Transmission Control Protocol accepts data from a data stream, divides it into chunks, and adds a TCP header creating a TCP segment. The TCP segment is then encapsulated into an Internet Protocol (IP) datagram, and exchanged with peers.

The term *TCP packet* appears in both informal and formal usage, whereas in more precise terminology *segment* refers to the TCP protocol data unit (PDU), *datagram* to the IP PDU, and *frame* to the data link layer PDU:

Processes transmit data by calling on the TCP and passing buffers of data as arguments. The TCP packages the data from these buffers into segments and calls on the internet module [e.g. IP] to transmit each segment to the destination TCP.

A TCP segment consists of a segment *header* and a *data* section. The TCP header contains 10 mandatory fields, and an optional extension field (*Options*, pink background in table).

The data section follows the header. Its contents are the payload data carried for the application. The length of the data section is not specified in the TCP segment header. It can be calculated by subtracting the combined length of the TCP header and the encapsulating IP header from the total IP datagram length (specified in the IP header).

Source port (16 bits)

Identifies the sending port.

Destination port (16 bits)

Identifies the receiving port.

Sequence number (32 bits)

Has a dual role:

- If the SYN flag is set (1), then this is the initial sequence number. The sequence number of the actual first data byte and the acknowledged number in the corresponding ACK are then this sequence number plus 1.
- If the SYN flag is clear (0), then this is the accumulated sequence number of the first data byte of this segment for the current session.

Acknowledgment number (32 bits)

If the ACK flag is set then the value of this field is the next sequence number that the sender of the ACK is expecting. This acknowledges receipt of all prior bytes (if any). The first ACK sent by each end acknowledges the other end's initial sequence number itself, but no data.

Data offset (4 bits)

Specifies the size of the TCP header in 32-bit words. The minimum size header is 5 words and the maximum is 15 words thus giving the minimum size of 20 bytes and maximum of 60

bytes, allowing for up to 40 bytes of options in the header. This field gets its name from the fact that it is also the offset from the start of the TCP segment to the actual data.

Reserved (3 bits)

For future use and should be set to zero.

Flags (9 bits) (aka Control bits)

Contains 9 1-bit flags

- NS (1 bit): ECN-nonce - concealment protection CWR (1 bit): Congestion Window Reduced (CWR) flag is set by the sending host to indicate that it received a TCP segment with the ECE flag set and had responded in congestion control mechanism (added to header by RFC 3168).
- ECE (1 bit): ECN-Echo has a dual role, depending on the value of the SYN flag. It indicates:
- If the SYN flag is set (1), that the TCP peer is ECN capable.
- If the SYN flag is clear (0), that a packet with Congestion Experienced flag set (ECN=11) in the IP header was received during normal transmission (added to header by RFC 3168). This serves as an indication of network congestion (or impending congestion) to the TCP sender.
- URG (1 bit): indicates that the Urgent pointer field is significant
- ACK (1 bit): indicates that the Acknowledgment field is significant. All packets after the initial SYN packet sent by the client should have this flag set.
- PSH (1 bit): Push function. Asks to push the buffered data to the receiving application.
- RST (1 bit): Reset the connection
- SYN (1 bit): Synchronize sequence numbers. Only the first packet sent from each end should have this flag set. Some other flags and fields change meaning based on this flag, and some are only valid when it is set, and others when it is clear.

- FIN (1 bit): Last packet from sender.

Window size (16 bits)

The size of the *receive window*, which specifies the number of window size units (by default, bytes) (beyond the segment identified by the sequence number in the acknowledgment field) that the sender of this segment is currently willing to receive (*see Flow control and Window Scaling*).

Checksum (16 bits)

The 16-bit checksum field is used for error-checking of the header, the Payload and a Pseudo-Header. The Pseudo-Header consists of the Source IP Address, the Destination IP Address, the protocol number for the TCP-Protocol (0x0006) and the length of the TCP-Headers including Payload (in Bytes).

Urgent pointer (16 bits)

if the URG flag is set, then this 16-bit field is an offset from the sequence number indicating the last urgent data byte.

Options (Variable 0–320 bits, divisible by 32)

The length of this field is determined by the data offset field. Options have up to three fields: Option-Kind (1 byte), Option-Length (1 byte), Option-Data (variable). The Option-Kind field indicates the type of option, and is the only field that is not optional. Depending on what kind of option we are dealing with, the next two fields may be set: the Option-Length field indicates the total length of the option, and the Option-Data field contains the value of the option, if applicable. For example, an Option-Kind byte of 0x01 indicates that this is a No-Op option used only for padding, and does not have an Option-Length or Option-Data byte following it. An Option-Kind byte of 0 is the End Of Options option, and is also only one byte. An Option-Kind byte of 0x02 indicates that this is the Maximum Segment Size option, and will

be followed by a byte specifying the length of the MSS field (should be 0x04). This length is the total length of the given options field, including Option-Kind and Option-Length bytes. So while the MSS value is typically expressed in two bytes, the length of the field will be 4 bytes (+2 bytes of kind and length). In short, an MSS option field with a value of 0x05B4 will show up as (0x02 0x04 0x05B4) in the TCP options section.

Padding

The TCP header padding is used to ensure that the TCP header ends, and data begins, on a 32 bit boundary. The padding is composed of zeros.

ICMP

The Internet Control Message Protocol (ICMP) is a supporting protocol in the Internet protocol suite. It is used by network devices, including routers, to send error messages and operational information indicating, for example, that a requested service is not available or that a host or router could not be reached.ICMP differs from transport protocols such as TCP and UDP in that it is not typically used to exchange data between systems, nor is it regularly employed by end-user network applications (with the exception of some diagnostic tools like ping and traceroute).

The Internet Control Message Protocol is part of the Internet Protocol Suite, as defined in RFC 792. ICMP messages are typically used for diagnostic or control purposes or generated in response to errors in IP operations (as specified in RFC 1122). ICMP errors are directed to the source IP address of the originating packet.

For example, every device (such as an intermediate router) forwarding an IP datagram first decrements the time to live (TTL) field in the IP header by one. If the resulting TTL is 0, the packet is discarded and an ICMP time exceeded in transit message is sent to the datagram's source address.
ICMP uses the basic support of IP as if it were a higher level protocol, however, ICMP is actually an integral part of IP. Although ICMP messages are contained within standard IP packets, ICMP messages are usually processed as a special case, distinguished from normal IP processing. In many cases, it is necessary to inspect the contents of the ICMP message and deliver the appropriate error message to the application responsible for transmission of the IP packet that prompted the sending of the ICMP message.
Many commonly used network utilities are based on ICMP messages. The traceroute command can be implemented by transmitting IP datagrams with specially set IP TTL header fields, and looking for ICMP time exceeded in transit and Destination unreachable messages generated in response. The related ping utility is implemented using the ICMP echo request and echo reply messages.

Datagram structure

The ICMP packet is encapsulated in an IPv4 packet. The packet consists of header and data sections.

Header

The ICMP header starts after the IPv4 header and is identified by IP protocol number '1'.All ICMP packets have an 8-byte header and variable-sized data section. The first 4 bytes of the header have fixed format, while the last 4 bytes depend on the type/code of that ICMP packet.

Checksum

Error checking data, calculated from the ICMP header and data, with value 0 substituted for this field. The Internet Checksum is used.Rest of Header

Four-bytes field, contents vary based on the ICMP type and code.
Data

ICMP error messages contain a data section that includes a copy of the entire IPv4 header, plus at least the first eight bytes of data from the IPv4 packet that caused the error message. The maximum length of ICMP error messages is 576 bytes.[3] This data is used by the host to match the message to the appropriate process. If a higher level protocol uses port numbers, they are assumed to be in the first eight bytes of the original datagram's data.

The variable size of the ICMP packet data section has been exploited. In the "Ping of death", large or fragmented ping packets are used for denial-of-service attacks. ICMP data can also be used to create covert channels for communication. These channels are known as ICMP tunnels.

Control messages

Control messages are identified by the value in the type field. The code field gives additional context information for the message. Some control messages have been deprecated since the protocol was first introduced.

PAYLOAD

The essential data that is being carried within a packet or other transmission unit. The payload does not include the "overhead" data required to get the packet to its destination. Note that what constitutes the payload may depend on the point-of-view. To a communications layer that needs some of the overhead data to do

its job, the payload is sometimes considered to include the part of the overhead data that this layer handles. However, in more general usage, the payload is the bits that get delivered to the end user at the destination.

Interpacket gap

In computer networking, a minimal pause may be required between network packets or network frames. Depending on the physical layer protocol or encoding used, the pause may be necessary to allow for receiver clock recovery, permitting the receiver to prepare for another packet (e.g. powering up from a low-power state) or another purpose.

sleep — The sleep command is used to delay for a specified amount of time.The sleep command pauses for an amount of time defined by NUMBER.

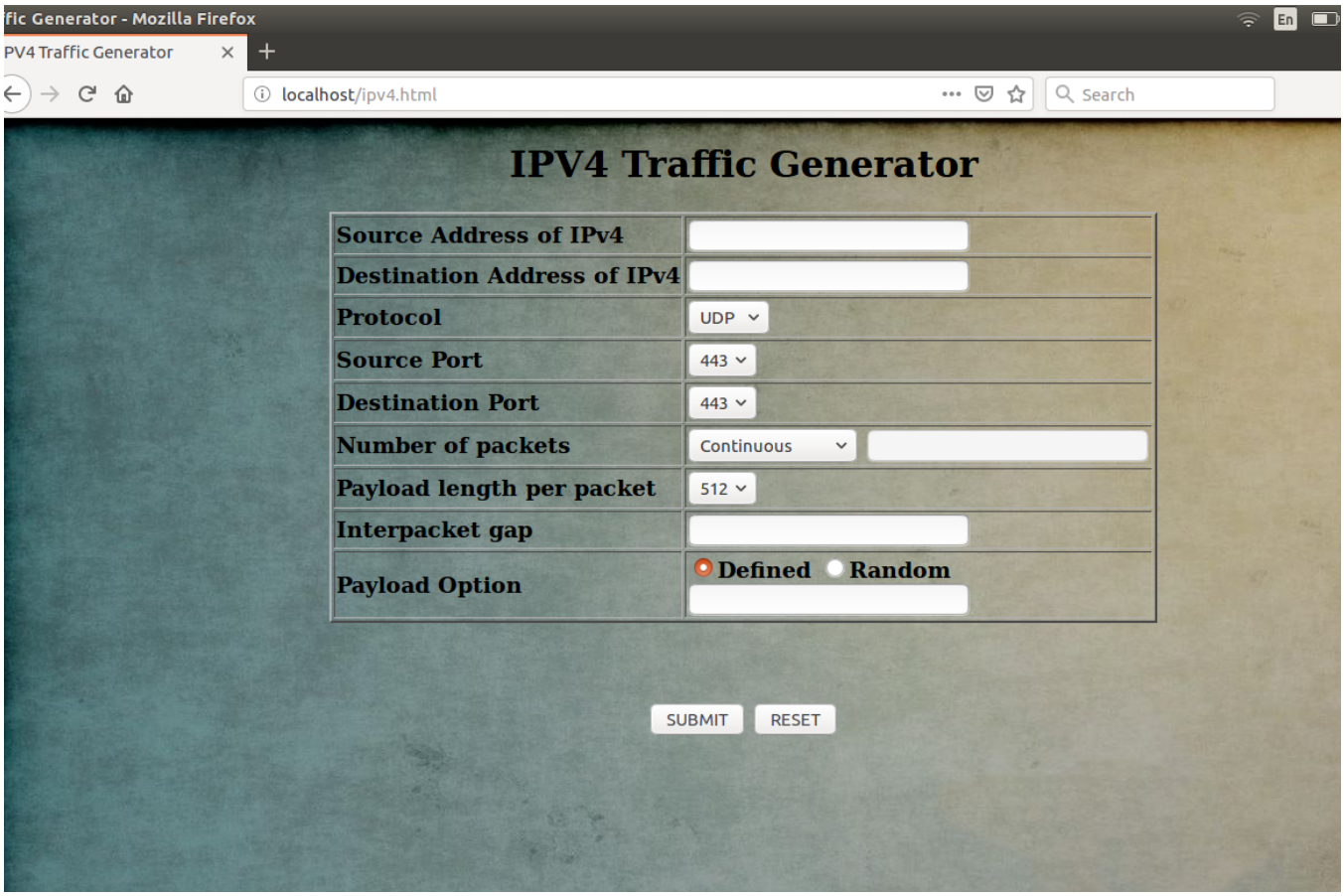
SUFFIX may be "s" for seconds (the default), "m" for minutes, "h" for hours, or "d" for days.
Some implementations require that NUMBER be an integer, but modern Linux implementations allow NUMBER to also be a floating-point value.If more than one NUMBER is specified, sleep delays for the sum of their values.

syntax

sleep NUMBER[SUFFIX]...
sleep OPTION

USER INTERFACE

IPV4 TRAFFIC GENERATOR PAGE



IPV4 TRAFFIC GENERATOR HTML

1

led Document 1 - gedit

Open

```
<!DOCTYPE html>
<head>
<title>IPV4 Traffic Generator</title>
</head>
<body background="1.jpg">
<form method="POST" action="test.php"><center>
        <h1>IPV4 Traffic Generator</h1>
        <h3>
<table border="2px solid " bordercolor="black">
<tr><td>
<label for="src_add">Source Address of IPv4</label></td>
<td>
<input type="text" name="src_add"></td></tr>
<tr><td>
<label for="dest_add">Destination Address of IPv4</label>
</td><td>
<input type="text" name="dest_add"></td></tr>
<tr><td>Protocol</td><td><select name ="prot" onclick="disabledrop(this)">
<option value="udp">UDP</option>
<option value="tcp">TCP</option>
<option value="icmp">ICMP</option></select></td></tr>

<tr><td>Source Port</td><td><select id="srcp" name="srcp">
<option value="443">443</option>
<option value="21">21</option>
<option value="22">22</option>
<option value="53">53</option>
<option value="80">80</option></select></td></tr>

<tr><td>Destination Port</td><td><select id="dstp" name="dstp">
<option value="443">443</option>
<option value="21">21</option>
<option value="22">22</option>
<option value="53">53</option>
<option value="80">80</option></select>
<script>
function disabledrop(a1)
{
var v1=a1.value;
if(v1=="icmp")
```

Plain Text Tab Width: 8 Ln 99, Col

2


led Document 1 - gedit
Open
if(v1=="icmp")
{
document.getElementById('srcp').disabled=true;
document.getElementById('dstp').disabled=true;
}
else
{
document.getElementById('srcp').disabled=false;
document.getElementById('dstp').disabled=false;
}
}
</script>
</td></tr>
<tr><td>Number of packets</td><td><select name="cont" onchange="disablen(this)">
<option value="cont">Continuous</option>
<option value="ncont">Non-continuous</option></select>
<input type="text" name="number" disabled="true" id="t">
<script>
function disablen(a2)
{
var t1=document.getElementById('t')
t1.disabled=a2.value=="cont";
}
</script></td></tr>
<tr><td>Payload length per packet</td>
<td><select name="len">
<option>512</option>
<option>8</option></select>
<tr><td>Interpacket gap</td>
<td>
<input type="text" name="gap"></td></tr>
<tr><td>Specify</td>
<td><input type="radio" name="select" value="own" type="text" onclick="dis(this)" checked="true">Defined
<input type="radio" name="select" value="rand" onclick="dis(this)" >Random
<input type="text" name="po" id="p1">
<script>
function dis(a2)
{
var p=a2.value;
Plain Text
Tab Width: 8

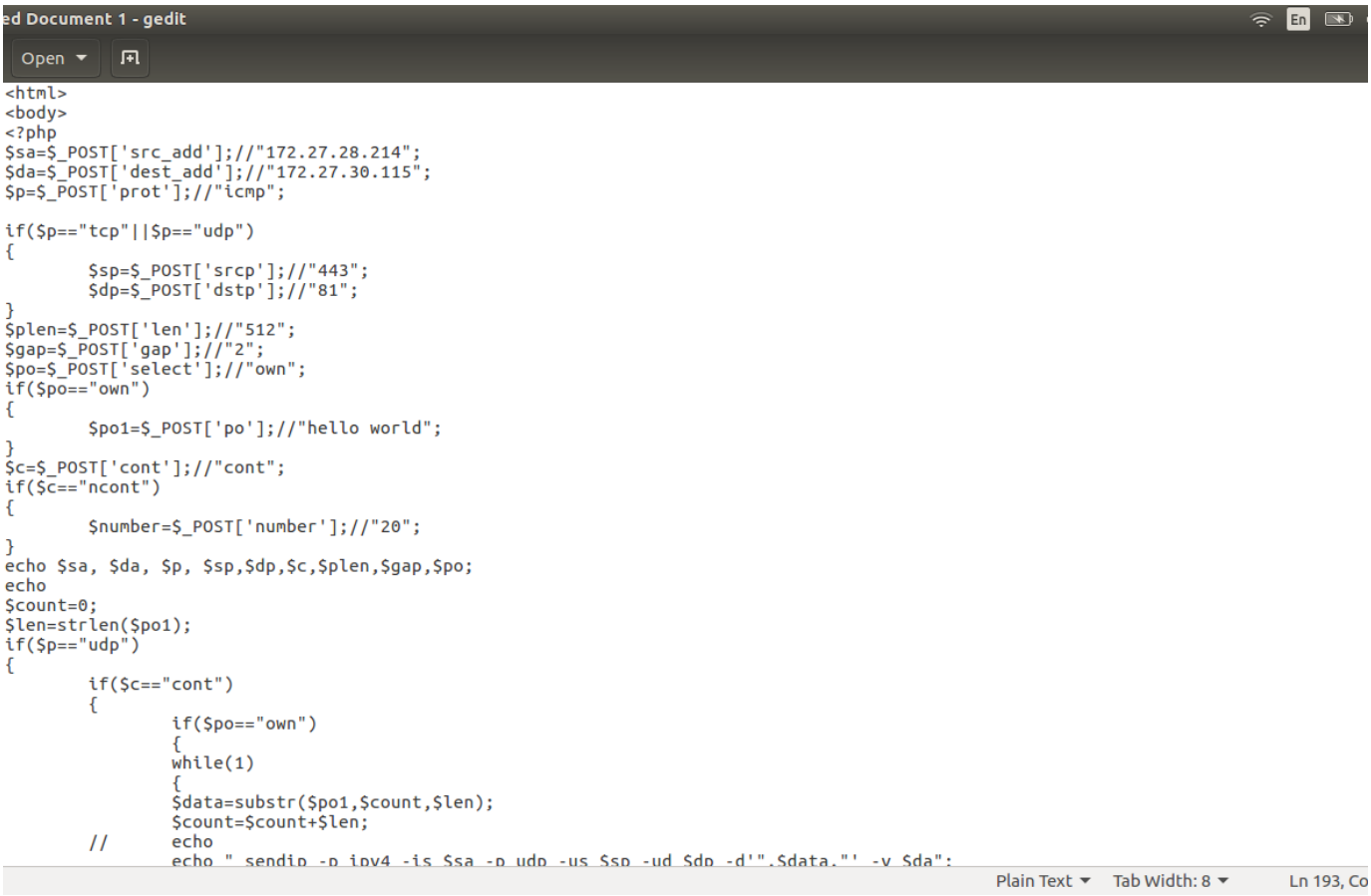
3


ed Document 1 - gedit
Open
}
</script></td></tr>
<tr><td>Payload length per packet</td>
<td><select name="len">
<option>512</option>
<option>8</option></select>
<tr><td>Interpacket gap</td>
<td>
<input type="text" name="gap"></td></tr>
<tr><td>Payload Option</td>
<td><input type="radio" name="select" value="own" type="text" onclick="dis(this)" checked="true">Defined
<input type="radio" name="select" value="rand" onclick="dis(this)" >Random
<input type="text" name="po" id="p1">
<script>
function dis(a2)
{
var p=a2.value;
if(p=="a2")
{
document.getElementById("p1").disabled=false;
}
else if(p=="rand")
{
document.getElementById("p1").disabled=true;
}
}
</script>
</td></tr>
</table>

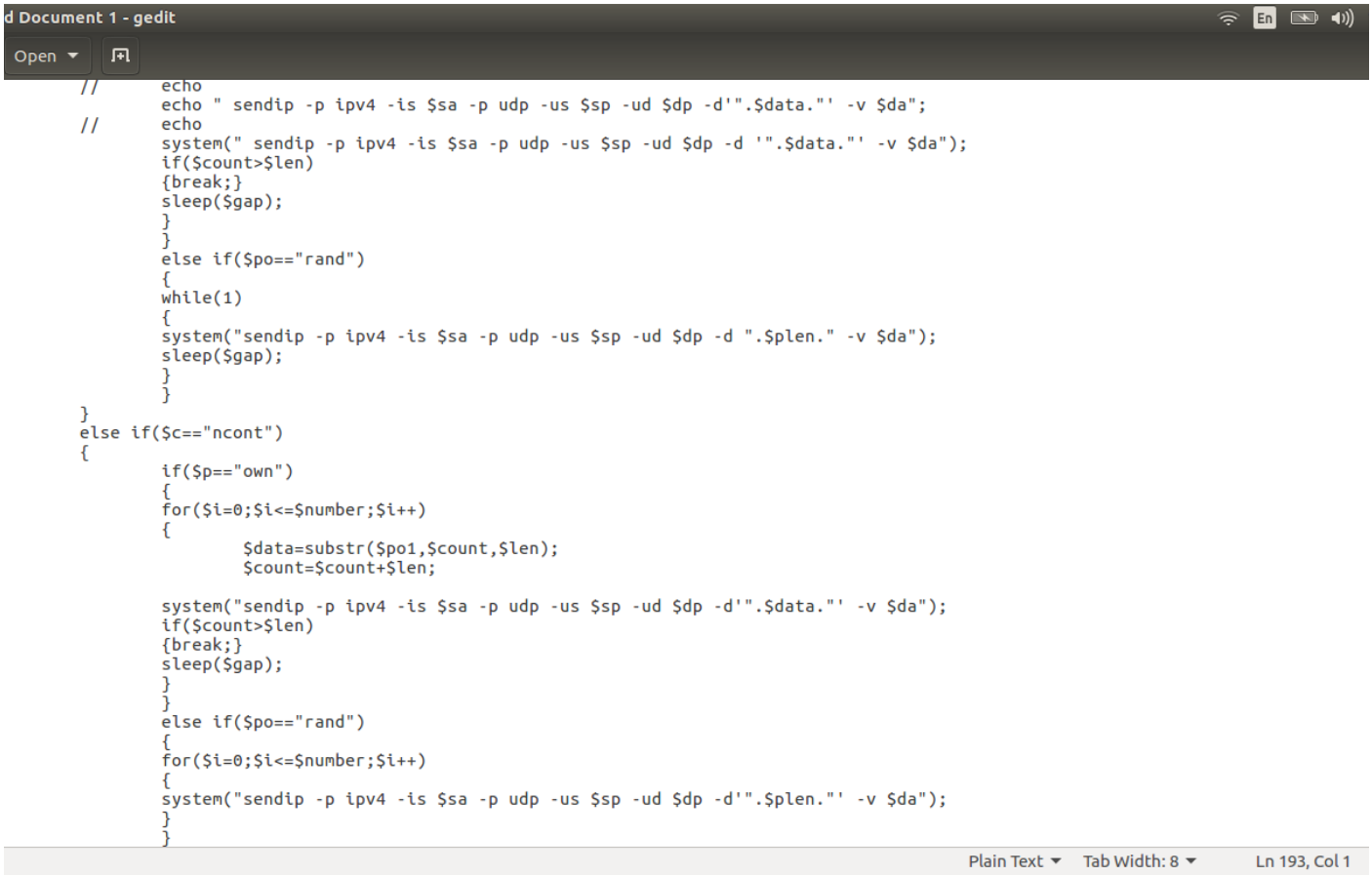
<button type="sub">SUBMIT</button>
<button type="reset">RESET</button>
</h3>
</form>
</body>
</html>
Plain Text
Tab Width: 8
Ln 37, Col 1

test PHP

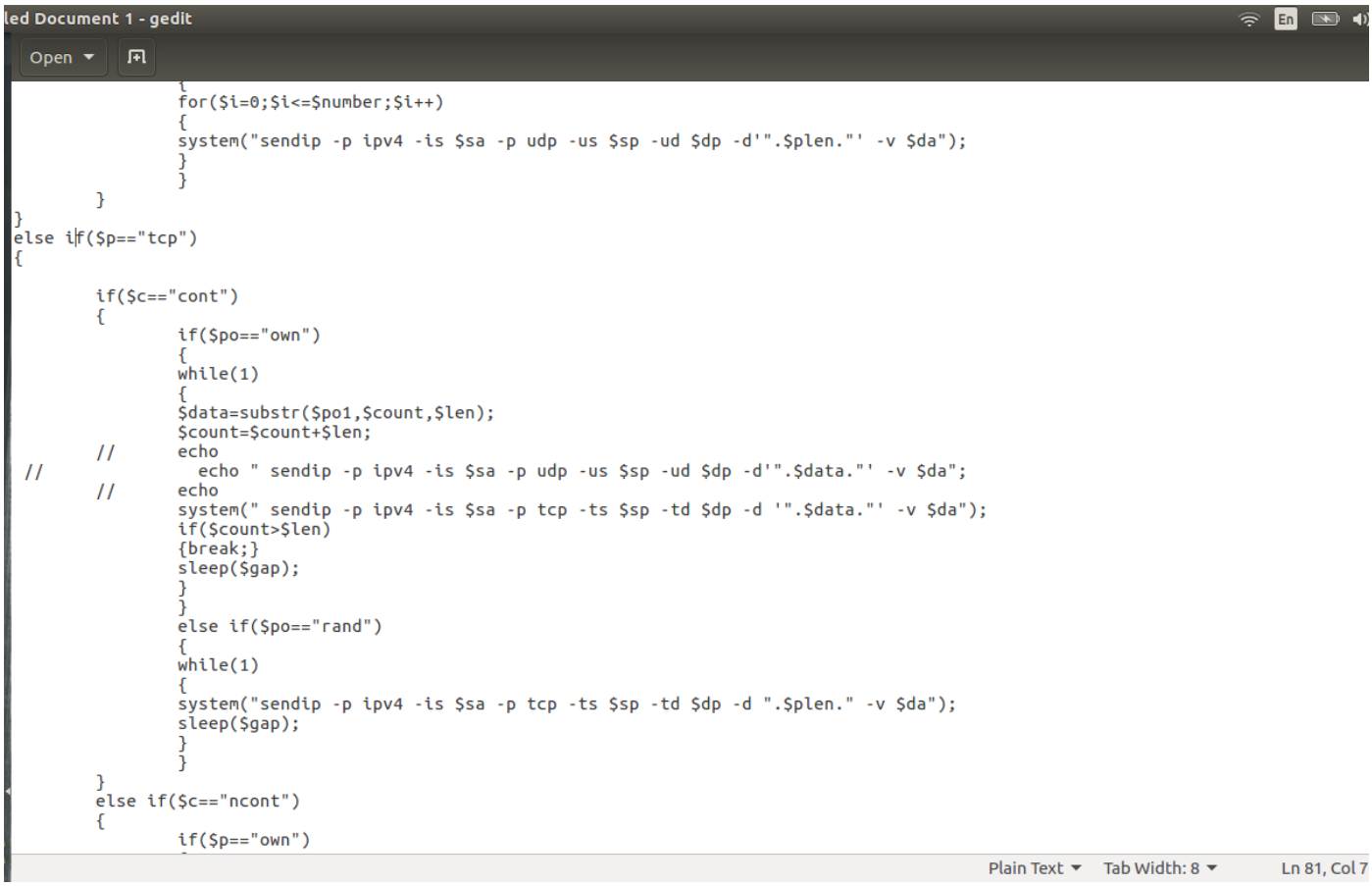
1


ed Document 1 - gedit
Open
<html>
<body>
<?php
$sa=$_POST['src_add'];//"172.27.28.214";
$da=$_POST['dest_add'];//"172.27.30.115";
$p=$_POST['prot'];//"icmp";
if($p=="tcp"||$p=="udp")
{
$sp=$_POST['srcp'];//"443";
$dp=$_POST['dstp'];//"81";
}
$plen=$_POST['len'];//"512";
$gap=$_POST['gap'];//"2";
$po=$_POST['select'];//"own";
if($po=="own")
{
$po1=$_POST['po'];//"hello world";
}
$c=$_POST['cont'];//"cont";
if($c=="ncont")
{
$number=$_POST['number'];//"20";
}
echo $sa, $da, $p, $sp,$dp,$c,$plen,$gap,$po;
echo
$count=0;
$len=strlen($po1);
if($p=="udp")
{
if($c=="cont")
{
if($po=="own")
{
while(1)
{
$data=substr($po1,$count,$len);
$count=$count+$len;
//
echo
Plain Text
Tab Width: 8
Ln 193, Co

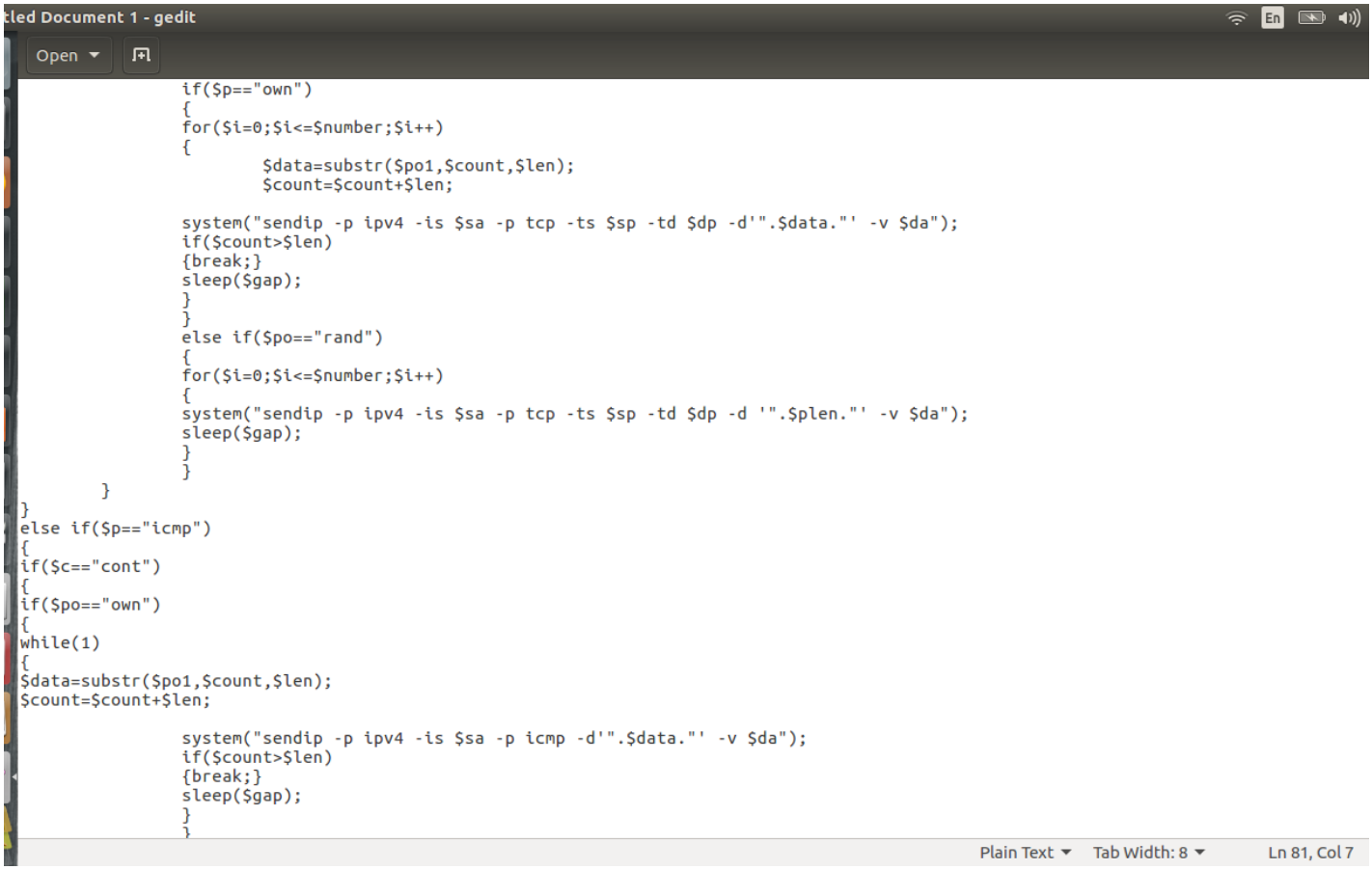
2


d Document 1 - gedit
Open
//	echo
echo " sendip -p ipv4 -is $sa -p udp -us $sp -ud $dp -d'".$data."' -v $da";
//	echo
system(" sendip -p ipv4 -is $sa -p udp -us $sp -ud $dp -d '".$data."' -v $da");
if($count>$len)
{break;}
sleep($gap);
}
}
else if($po=="rand")
{
while(1)
{
system("sendip -p ipv4 -is $sa -p udp -us $sp -ud $dp -d ".$plen." -v $da");
sleep($gap);
}
}
}
else if($c=="ncont")
{
if($p=="own")
{
for($i=0;$i<=$number;$i++)
{
$data=substr($po1,$count,$len);
$count=$count+$len;
system("sendip -p ipv4 -is $sa -p udp -us $sp -ud $dp -d'".$data."' -v $da");
if($count>$len)
{break;}
sleep($gap);
}
}
else if($po=="rand")
{
for($i=0;$i<=$number;$i++)
{
system("sendip -p ipv4 -is $sa -p udp -us $sp -ud $dp -d'".$plen."' -v $da");
}
}
Plain Text
Tab Width: 8
Ln 193, Col 1

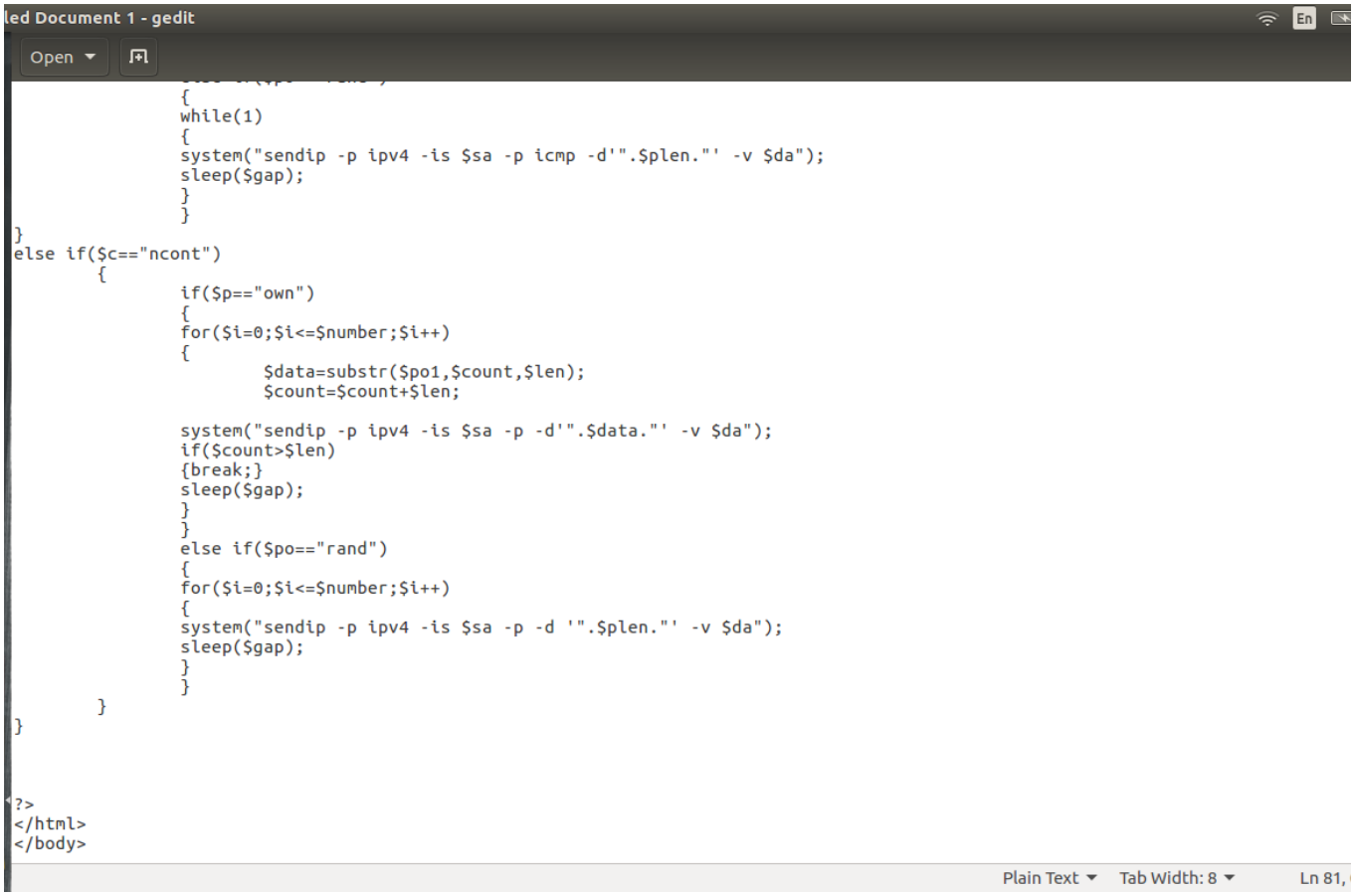
3


led Document 1 - gedit
Open
for($i=0;$i<=$number;$i++)
system("sendip -p ipv4 -is $sa -p udp -us $sp -ud $dp -d'".$plen."' -v $da");
else if($p=="tcp")
if($c=="cont")
if($po=="own")
while(1)
$data=substr($po1,$count,$len);
$count=$count+$len;
echo " sendip -p ipv4 -is $sa -p udp -us $sp -ud $dp -d'".$data."' -v $da";
system(" sendip -p ipv4 -is $sa -p tcp -ts $sp -td $dp -d '".$data."' -v $da");
if($count>$len)
{break;}
sleep($gap);
else if($po=="rand")
system("sendip -p ipv4 -is $sa -p tcp -ts $sp -td $dp -d ".$plen." -v $da");
else if($c=="ncont")
if($p=="own")
Plain Text
Tab Width: 8
Ln 81, Col 7

4


tled Document 1 - gedit
Open
if($p=="own")
{
for($i=0;$i<=$number;$i++)
{
$data=substr($po1,$count,$len);
$count=$count+$len;
system("sendip -p ipv4 -is $sa -p tcp -ts $sp -td $dp -d'".$data."' -v $da");
if($count>$len)
{break;}
sleep($gap);
}
}
else if($po=="rand")
{
for($i=0;$i<=$number;$i++)
{
system("sendip -p ipv4 -is $sa -p tcp -ts $sp -td $dp -d '".$plen."' -v $da");
sleep($gap);
}
}
}
}
else if($p=="icmp")
{
if($c=="cont")
{
if($po=="own")
{
while(1)
{
$data=substr($po1,$count,$len);
$count=$count+$len;
system("sendip -p ipv4 -is $sa -p icmp -d'".$data."' -v $da");
if($count>$len)
{break;}
sleep($gap);
}
}
Plain Text
Tab Width: 8
Ln 81, Col 7

5


led Document 1 - gedit
Open
{
while(1)
{
system("sendip -p ipv4 -is $sa -p icmp -d'".$plen."' -v $da");
sleep($gap);
}
}
}
else if($c=="ncont")
{
if($p=="own")
{
for($i=0;$i<=$number;$i++)
{
$data=substr($po1,$count,$len);
$count=$count+$len;
system("sendip -p ipv4 -is $sa -p -d'".$data."' -v $da");
if($count>$len)
{break;}
sleep($gap);
}
}
else if($po=="rand")
{
for($i=0;$i<=$number;$i++)
{
system("sendip -p ipv4 -is $sa -p -d '".$plen."' -v $da");
sleep($gap);
}
}
}
}
?>
</html>
</body>
Plain Text
Tab Width: 8
Ln 81,

RESULT and APPLICATIONS

The IPV4 Traffic Generator was successfully developed. We have developed a network packet generator that supports basic features such as sending a packet continuous or non-continuous, any protocol such as UDP, TCP, ICMP ,and of respective packet length and payload option could be user-defined or random using any source port and destination port by using ip addresses.The internet facility of any organization has to be monitored regularly so as to check for any cases of misuse and misapplication.

So to keep a track of such things, all the data being transmitted from one terminal or system to another has to be recorded and analyzed. It can be used as a utility which can be installed on any server machine and thus is useful in finding the faults; misuse, hacking or any legal or illegal problem held at any attached terminal and can be removed in no time. It gives us a tool, utility software which can give us solution to many networking problems in no time. With this only its objective area widens and so the scope.

The end result of this project is that it will act as a utility tool and can be easily installed on any application server and there it can be used to send packets using ip addresses and other details such as protocol, ports etc. from one system to another.