iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 4 **Issue:** IX **Month of publication:** September 2016

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

Volume 4 Issue IX, September 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET).

Simulation Development of Linux Firewall

Rekha Pandey[1], Aman Arora[2]
[1]Research Scholar, [2]Assistant Professor
Department Computer Science, MIET Kurukshetra

Abstract: ***Open-source systems are not going to replace offerings from commercial vendors. They do, however, offer an increasingly viable alternative. Open-source solutions should be evaluated side-by-side with commercial as part of a tender process. Judged on their merits, open-source solutions may prove to be the best solution for many organizations. Eventually, all tools available will prove to be lacking in some area, whether it is additional functionality or a specific feature. In these cases, having the capability to build open source tool is extremely beneficial.***
In this work we design and implement a gateway filter tool based on various network parameters. The system works as an exhaustive firewall. It has support for both logging on the basis of packet's protocol type.
Index term- TCP, IP, NAT, Virtual File System, Linux

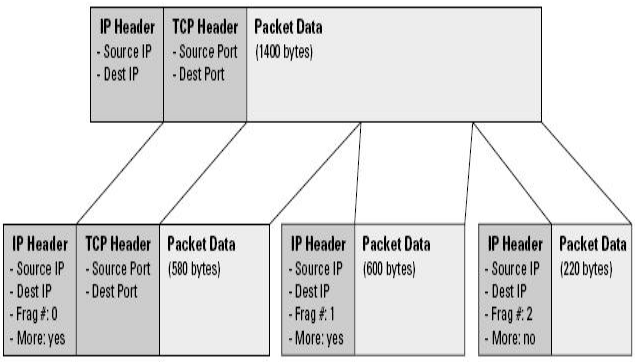
I. INTRODUCTION

The term firewall doesn't accurately describe its function. A real firewall is a barrier to prevent fires[2[from spreading from one room or building to another. A real firewall blocks fires completely. On the other hand, the firewalls should inspect all "fires" and let some pass through while blocking others. Sure, the Internet is hot, but who came up with this term?
A term that more accurately describes the function of the Internet firewall products is doorman. The firewall (or doorman) is the security guard that sits behind a desk near the front entrance of a large office building and screens everybody who wants to come inside. Depending on the type of office, the guard may also screen or inspect people who are leaving the building.

II. FRAGMENTS

IP network traffic travels over all kinds of network segments between the sender and the destination. Not all of these segments or links may allow the same maximum packet size. The maximum packet size is called the Maximum Transmission Unit (MTU) of the network. If a larger IP packet has to cross a network link that allows only a smaller size, the original IP packet can be broken into smaller IP packets and continue. These smaller packets are called IP fragments and are shown in Figure 2.1. Each of these IP fragments has its own IP header that contains the source and final destination IP addresses, as well as a fragment position number, but only a small part of the original TCP information.

Figure 2.1 TCP Information



Two aspects of fragments are important:

A. To speed up things after crossing the network link that allows only a smaller size, the IP fragments are not reassembled again at the other side but travel independently to the final destination. There, they are reunited again in order to form the original IP packet.

B. Each IP fragment contains only a part of the original TCP information. Therefore, only the first fragment contains the TCP part that shows the TCP port number. The other fragments carry the remaining TCP information but not the TCP port number.

What's the poor firewall to do? The arriving IP[3] fragments, except the first one, contain no indication of a TCP port number, so the packet filters can't make a decision based on that. Blocking the second and subsequent fragments disallows all network packets that have passed a network link with a small maximum packet size. Reassembling the packet itself and making a decision based on the complete IP packet means that the firewall is accepting all these fragments and storing them until all fragments have arrived and then continue. This opens up a strong possibility that a hacker can make the firewall do a lot of intensive work, especially if the hacker never sends the last packet. The firewall may be so busy with sorting out all these small packets that it can't focus on other tasks. This is called a denial-of-service attack. Letting the second and subsequent fragments pass the firewall may be the solution, but this strategy also has a disadvantage. The first fragment can be inspected and is possibly blocked. The final-destination computer on the internal network knows that if the first fragment never arrives, it should not reassemble the fragments that did come through and use the fragment anyway. Some implementations of TCP/IP make the mistake of reassembling the fragments, and hackers capitalize on this mistake by sending a complete IP packet that is disguised as a fragment. The firewall allows the packet to pass through, thus relying on the absence of the first fragment. The final-destination computer receives this self-advertised fragment and processes it as a complete IP packet! Because the firewall doesn't block second and subsequent fragments, the hacker is able to send packets to computers on the internal network unchecked.

III. NETWORK ADDRESS TRANSLATION

Originally, Network Address Translation, or NAT, was introduced to save IP addresses in use on the Internet. An IP address is 32 bits long and with that number of bits[5], you can have only about four billion different IP addresses. Because many companies have claimed large blocks of IP addresses, the available IP numbers were quickly becoming depleted. In May 1994, RFC1631 suggested what was then thought to be a short-term solution — NAT. As it turned out, NAT offered several unexpected advantages, as you'll soon discover. With NAT, all computers on the internal network can use a private range of IP addresses, such as 10.0.0.0/8, which is not in use on the Internet. When they make a connection to the outside world, the NAT computer replaces the private IP address, for example, 10.65.1.7 — listed as Source IP address in the IP packet — with its own public IP address, 23.1.8.3, and sends the packet on its way. The destination computer on the Internet thinks the original sender is 23.1.8.3, and sends a return packet back to this IP address. The NAT computer receives a packet for 23.1.8.3 and replaces the Destination IP address with the original 10.65.1.7 to travel the last leg on the internal network, as shown in Figure 3.1. NAT may as well have been called Network Address Replacing.

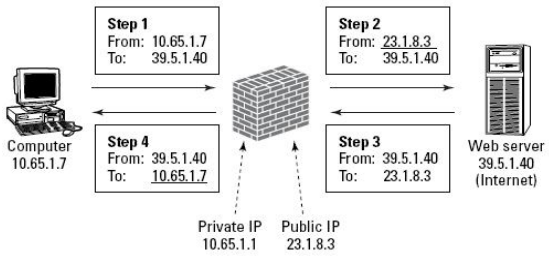


Fig. 3.1Network Address Translation

Finally, what if more than one computer on the internal network wants to use the NAT computer to communicate with the Internet? The 1994 RFC1631 document proposed to solve this by letting the NAT computer have multiple public IP addresses and using one for every concurrent connection from the internal computers to the Internet. In every modern implementation of NAT, this can just as easily be solved by not only changing the Source IP address to 23.1.8.3, but by replacing the source port number with an unused port number above 1023 as well. All the NAT computer has to do is keep a list of which port number temporarily belongs to which requesting internal network computer. Technically, the technique to replace ports is called Network Address Port Translation (NAPT), but everybody just says NAT. Nearly 65,000 port numbers are available, so in theory, one NAT computer can handle thousands of internal network computers.

Volume 4 Issue IX, September 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET).

IV. PURPOSED WORK

A. Conceptual Architecture of the Linux Kernel

The Linux operating system is composed of four major subsystems:

User Applications -- the set of applications in use on a particular Linux system will be different depending on what the computer system is used for, but typical examples include a word-processing application and a web-browser.

O/S Services -- these are services that are typically considered part of the operating system (a windowing system, command shell, etc.); also, the programming interface to the kernel (compiler tool and library) is included in this subsystem.

Linux Kernel -- this is the main area of interest in this paper; the kernel abstracts and mediates access to the hardware resources, including the CPU.

Hardware Controllers -- this subsystem is comprised of all the possible physical devices in a Linux installation; for example, the CPU, memory hardware, hard disks, and network hardware are all members of this subsystem

The Linux kernel presents a virtual machine interface to user processes. Processes are written without needing any knowledge of what physical hardware is installed on a computer -- the Linux kernel abstracts all hardware into a consistent virtual interface. In addition, Linux supports multi-tasking in a manner that is transparent to user processes: each process can act as though it is the only process on the computer, with exclusive use of main memory and other hardware resources. The kernel actually runs several processes concurrently, and is responsible for mediating access to hardware resources so that each process has fair access while inter-process security is maintained.

B. Network Interface Architecture

1) Goals: The network subsystem allows Linux systems to connect to other systems over a network. There are a number of possible hardware devices that are supported, and a number of network protocols that can be used. The network subsystem abstracts both of these implementation details so that user processes and other kernel subsystems can access the network without necessarily knowing what physical devices or protocol is being used.

2) Modules

a) Network device drivers communicate[6] with the hardware devices. There is one device driver module for each possible hardware device.

b) The device independent interface module provides a consistent view of all of the hardware devices so that higher levels in the subsystem don't need specific knowledge of the hardware in use.

c) The network protocol modules are responsible for implementing each of the possible network transport protocols.

d) The protocol independent interface module provides an interface that is independent of hardware devices and network protocol. This is the interface module that is used by other kernel subsystems to access the network without having a dependency on particular protocols or hardware.

Finally, the system calls interface module restricts the exported routines that user processes can access.

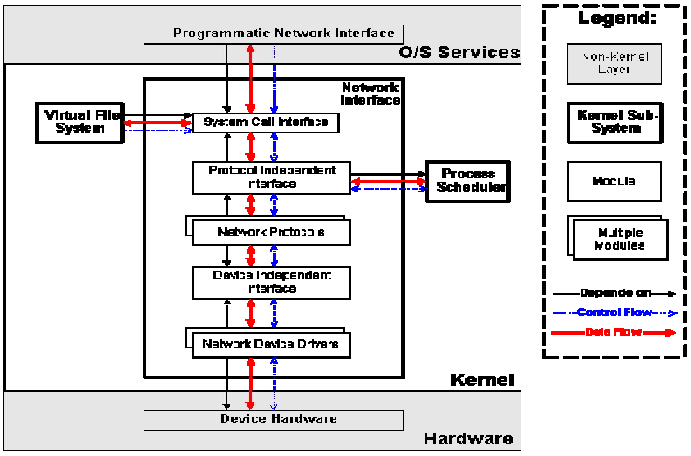


Figure 4.1 : Network Interface Subsystem in Context

Volume 4 Issue IX, September 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET).

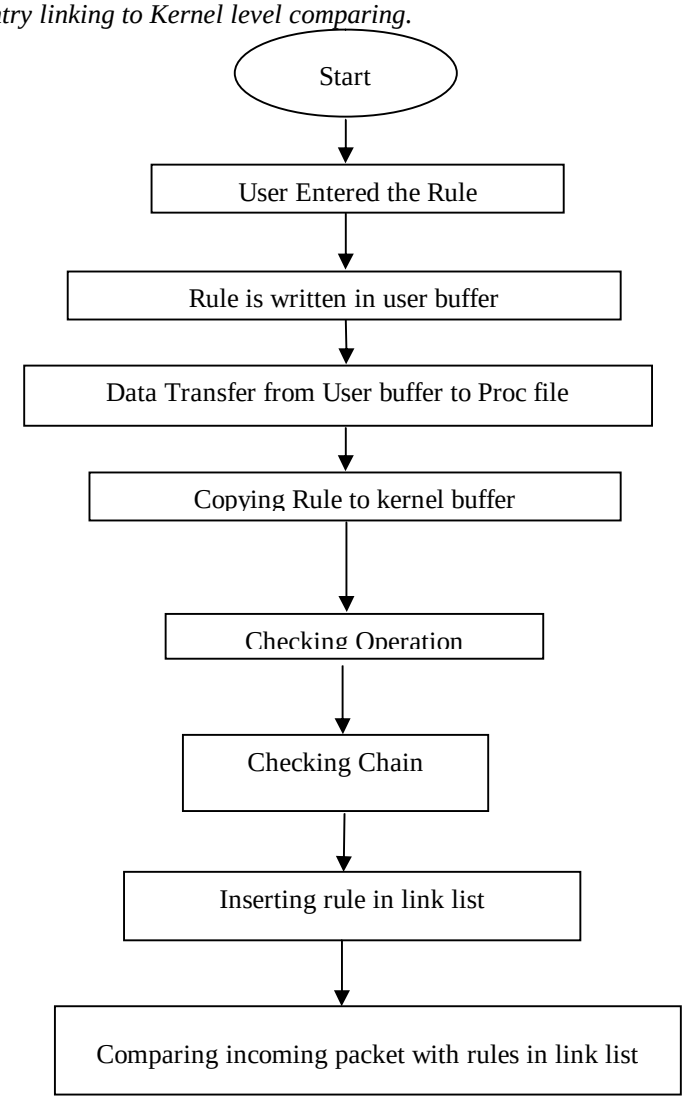
The conceptual architecture of the Linux kernel has proved its success; essential factors for this success were the provision for the organization of developers, and the provision for system extensibility. The Linux kernel architecture was required to support a large number of independent volunteer developers. This requirement suggested that the system portions that require the most development -- the hardware device drivers and the file and network protocols -- be implemented in an extensible fashion. The Linux architect chose to make these systems be extensible using a data abstraction technique: each hardware device driver is implemented as a separate module that supports a common interface. In this way, a single developer can add a new device driver, with minimal interaction required with other developers of the Linux kernel. The success of the kernel implementation by a large number of volunteer developers proves the correctness of this strategy.

3) A Virtual File System: Under Linux, all data are stored as files. Most users are familiar with the two primary types [7]of files: text and binary. But the /proc/ directory contains another type of file called a virtual file. It is for this reason that /proc/ is often referred to as a virtual file system.

These virtual files have unique qualities. Most of them are listed as zero bytes in size and yet when one is viewed, it can contain a large amount of information. In addition, most of the time and date settings on virtual files reflect the current time and date, indicative of the fact they are constantly updated.

Virtual files such as /proc/interrupts, /proc/meminfo, /proc/mounts, and /proc/partitions provide an up-to-the-moment glimpse of the system's hardware. Others, like the /proc/filesystems file and the /proc/sys/ directory provide system configuration information and interfaces.

4) Flow Chart of User Level entry linking to Kernel level comparing.



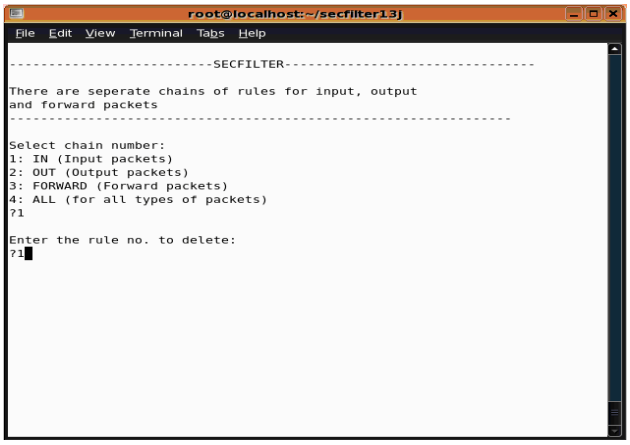
Volume 4 Issue IX, September 2016
ISSN: 2321-9653

V. SIMULATION RESULTS

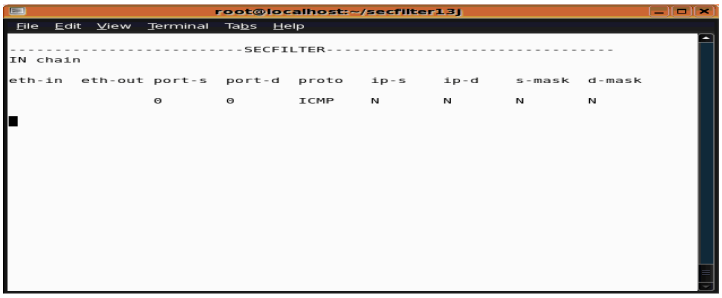
Snapshots 5.1 Snapshots ping before inserting rule.

Snapshots 5.2 Inserting rule to block packets on the basis of protocol. (Eg. ping (ICMP packets)).

Snapshots 5.3 Inserting rule in Inbound chain.



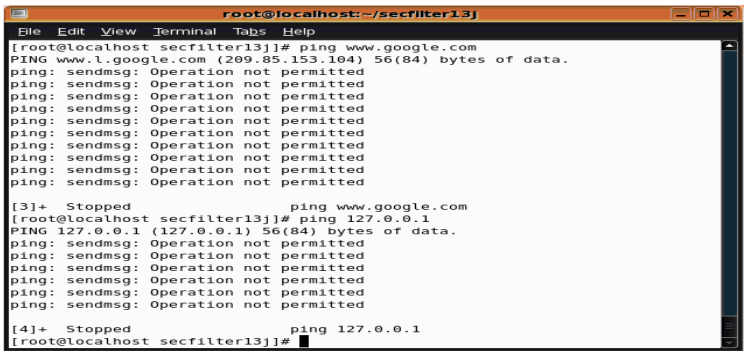
Snapshots 5.4 displaying the rule.



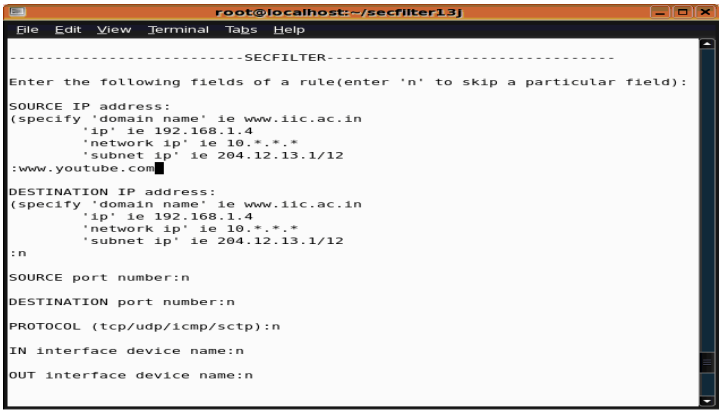
Ping after inserting rule.

Volume 4 Issue IX, September 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET).

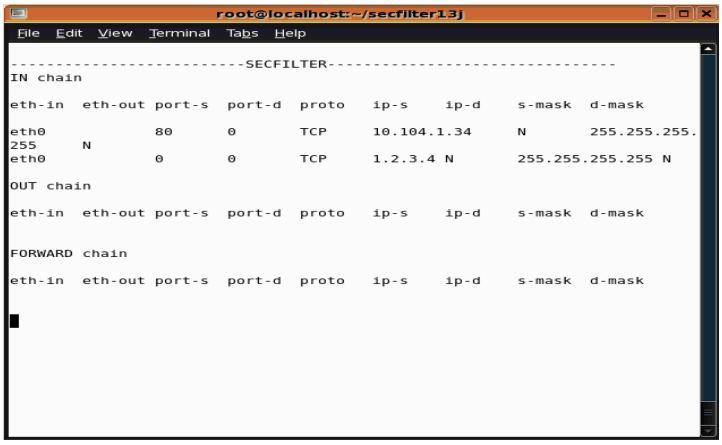
Snapshots 5.5 ICMP packets blocked.



Snapshots 5.6 Inserting rule to block packets on the basis of ip address.
(Eg. www.youtube.com)



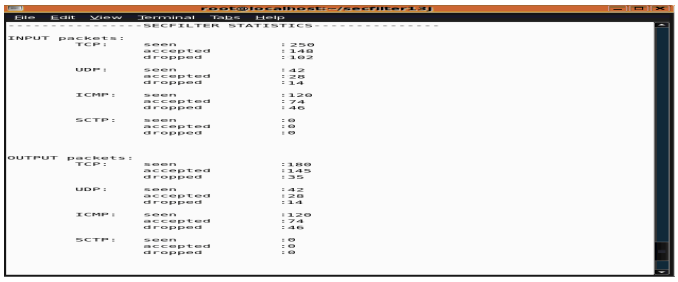
Snapshots 5.7 Displaying the rule.



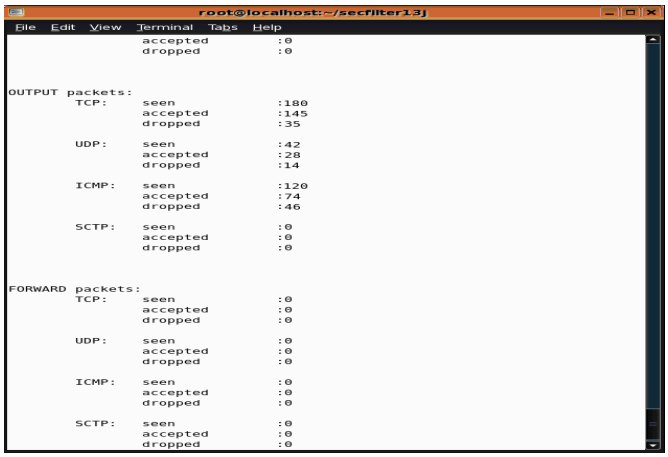
Snapshots 5.8 Site www.youtube.com is blocked.

Volume 4 Issue IX, September 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET).

Snapshots 5.9 Log file of INPUT packets and OUTPUT packets.



Snapshots 5.10 Log file of FORWARD packets.



VI. CONCLUSION AND FUTURE SCOPE

This application is perfect for running on a server which could be a network gateway. The most of developed has involved network layer programming Therefore, it is more like a non-application layer application.

We hope the solution should continue its making popularity among beginners as well as advanced users like system administrators.

A. Future Scope

We would surely like to take this work further onto a bigger domain of application as well as keep it on the level of current open standards that keeping it up to latest technologies. We know how fast new standards are coming in this area, so it is sometimes right to say what we have developed today is going to be out of application after few months that is support for the product is also a major issue.

We would like following to be extended to existing system:

1) A work on user interface so that it suites to even beginners.

2) Extensions for the application layer protocols so that it interacts with the user even at the application. Right now it is working on network layer only.

REFERENCES

[1] Intanagonwiwat, R. Govindan, D. Estrin, Directed Diffusion: Ascalable and robust communication paradigm for sensor networks,In Proc. 6th Annual International Conference on Mobile Computingand Networking, Boston, Massachusetts, 2000, pp. 56-67.

[2] N. Sadagopan, B. Krishnamachari, A. Helmy, Active query forwardingin sensor networks (ACQUIRE),Ad Hoc Networks, 2005, Vol.3, Issue 1, pp. 91-113.

[3] J. Kulik, W. R. Heinzelman, H. Balakrishnan, Adaptive protocols for informationdissemination in wireless sensor networks,In Proc. 5th AnnualACM/IEEE International Conference on Mobile Computing andNetworking, Massachusetts Institute of Technology, USA, Washington,1999, pp. 174-185.

[4] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, An application-specific protocol architecture for wireless microsensor networks,IEEE Trans. Wireless Commun., 2002, 1 (4), 60-70.

[5] S. Lindsey, C.Raghavendra, PEGASIS: Power-Efficient GAthering in Sensor Information Systems,In Proc. IEEE Aerospace Conference, USA, Montana, 2002, Vol. 3, pp. 1125-1130.

[6] B. Blum, T. He, S. Son, J. Stankovic, IGF: A state-free robust communication protocol for wireless sensor networks,TechnicalReport CS-2003-11, Department of Computer Science, University of Virginia, USA, 2003.

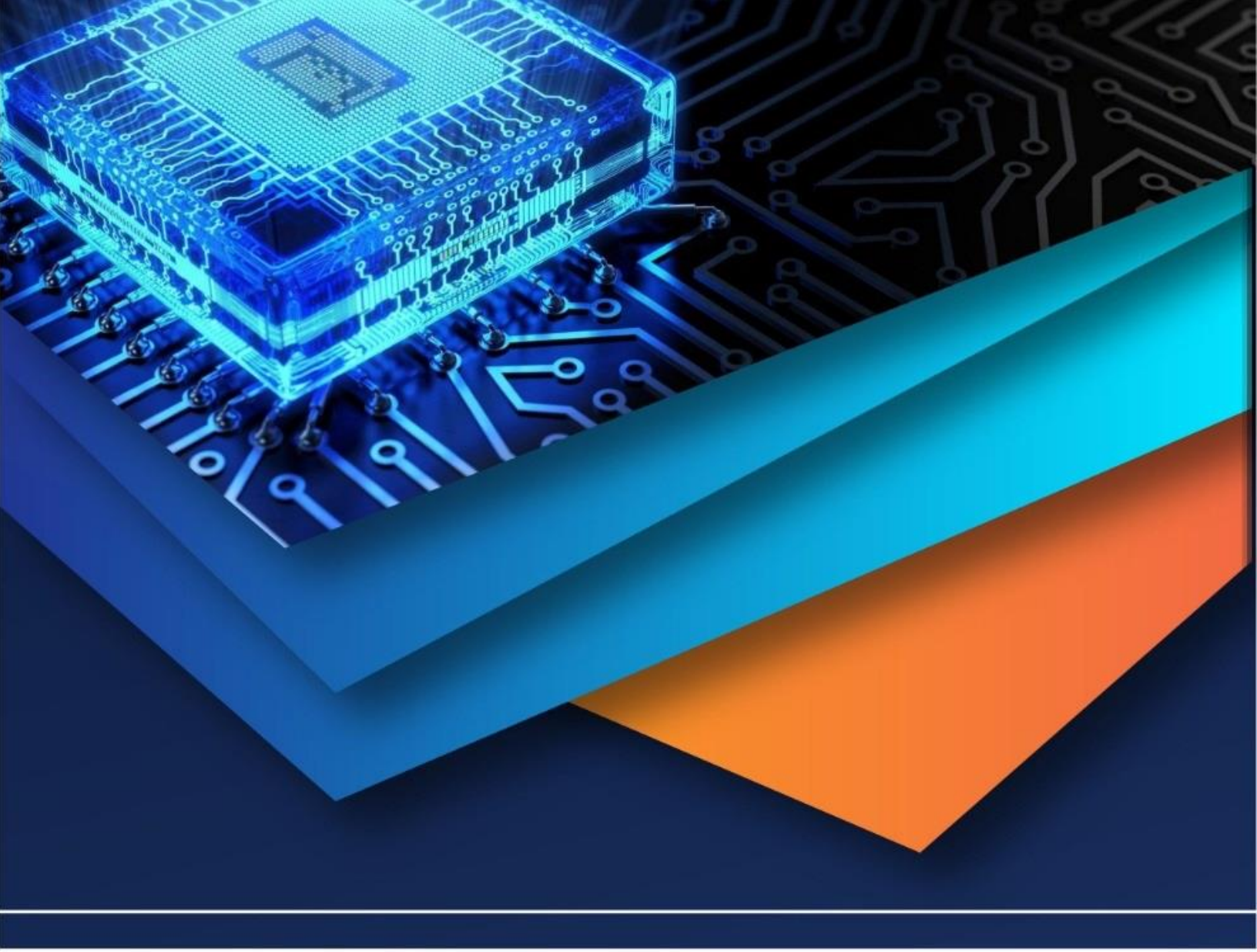
[7] O. Younis and S. Fahmy, HEED: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks, IEEE Transactions on Mobile Computing,2004, 3 (4), 366-379.

[8] A. Manjeshwar, D. Agrawal, TEEN: A routing protocol for enhanced efficiency in wireless sensor networks,In Proc. 15th International Parallel and Distributed Processing Symposium (IPDPS'01) Workshops, USA, California, 2001, pp. 2009-2015.

[9] A. Manjeshwar, D. Agrawal, "APTEEN: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks," In Proc. International Parallel and Distributed Processing Symposium, Florida, 2002, pp. 195-202.

[10] www.netfilter.org

[11] www.linuxjournal.com



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY