IOSR Journal of Business and Management (IOSR-JBM)
e-ISSN: 2278-487X, p-ISSN: 2319-7668
PP 51-61
www.iosrjournals.org

ALERT Generation Intrusion Detection System On Heterogeneous System

Simanta Hazra[1], Siddhartha Chatterjee[2] , Saikat Samanta[3]
[1](*Department of Computer Science & Engineering, Asansol Engineering College, India)*
[2](*Department of Computer Application, DSMS Group of Institution, India)*
[3](*Department of Computer Science & Engineering, Bengal College of Engineering, India)*

Abstract: *Security is one of the most fundamental concerns in today's Enterprise network. An enterprise is composed of heterogeneous entities having varying asset values and attack vulnerabilities. To protect the information resources in an enterprise, packet filtering based firewall rules are deployed and the same time, to detect potential threats in the systems and network, intrusion detection systems are also deployed.*
In a general enterprise, set up these two activities are performed independently. But it was shown in the literature that the dynamic configuration of firewall rules can be achieved through utilization of the alerts generated by the IDS tools. However, the IDS systems normally generates large number of alerts that results in blocking of a large number of sites by the firewalls.
In this work we propose a mechanism by which firewall rules are updated by capturing the alerts generated by IDS, but the sites are not blocked for all the information resources. Whether a site will be blocked for an information resource depends on Risk Rate of the resource. If the Risk Rate for an information resource exceeds a predefine threshold value then the site will be blocked for that particular resource. However the site will be available to all other resources having Risk Rate less than the threshold value. This includes the user experience of the network without using the attack vulnerabilities.
Keywords : *IDS, Snort, Firewall, Risk Management, Asset Management, My-SQL, PHP, Perl.*

I. Introduction

To protect information assets, there are various security measures undertaken such as deployment of packet filters (Firewall), Encryption of data, creation of virtual private network (VPN), user authentication and their access policies (Access Control Server or Network Access Control) ,vulnerably assignment and so on. Although all these tools are use to protect the systems, yet there is need for continuous monitoring of the systems and network so that if intrusion occurs it can be easily detected. For that, intrusion detection system (IDS) has been added as a new security measure in today's enterprise network.

The National Institute of Standards and Technology publication titled "Guide to Intrusion Detection and Prevention Systems (IDPS)" defines intrusion detection as "Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices". [10][11]

A typical enterprise setting for Intrusion Detection and their counter measure is shown in Fig.1 are normally done through firewall configuration. Commonly, an open source IDS tool, named Snort [1] is used to generate alerts when malicious activity is detected on the network.

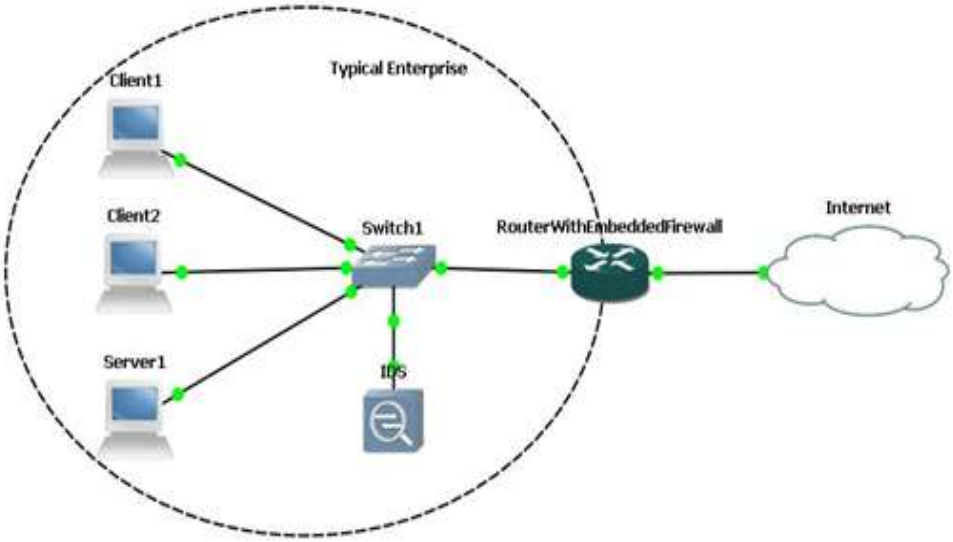


Fig.1 Typical Enterprise Scenario

Most of the work done until now uses this technique. Muhammad et. al. [16] suggested a scheme based on the above Principle. To protect network from intrusion they update the ACL's or Firewall rules configuration based on alerts generated by Snort and consequently the corresponding sites are permanently blocked. A severe weakness of the scheme stems from the fact that even a fine tuned Snort may generate too many alerts blocking unacceptably large number of sites which may result in poor user's experience. A similar scheme for FreeBSD based Linux was proposed in [Guardian] [17],[18].

Here we propose a framework that assigns a specific asset value to each individual information resource based on its importance to the enterprise. When an attack is generated by Snort, the severity of the attack for each individual information resource is evaluated based on frequency of alerts relevant to the resource and its asset value. If the severity for a particular resources higher than a predefined threshold value then the site is blocked for that resource. But for the resources for which the severity is less than the threshold value the sites remains available to them. However for such resources the incoming packets are now deeply inspected.

In a true sense, our Snort base system is an NIDS (Network base Intrusion Detection System) rather than NIPS (Network base Intrusion Prevention System). In contrast to NIDS analysis which operates in promiscuous mode, as our Snort system, which is passive and external to data flow, NIPS analysis which operates in inline mode must be accurate or sessions can be disrupted if the frames are incorrectly dropped rather than it forwarded by the hardware.
Moreover our software will run in every operating system because the primary requirements of the software such as snort [1], mysql [3], php [7], perl [6] is now available with every operating system and can be installed in every operating systems.

Our system is generated from free open source software, which doesn't require any money at all to build IDS in a network. Is that is not a great?

Note that there is no difference between IDS rule generation module and ACL rule generation module. In fact it can be work for any route like Cisco [2], Juniper, D-link etc. as well as through IPTABLES in Linux also with slide variation in their respective rule generation.

II. Outline Of Our Scheme

In a Typical Enterprise, as shown in Fig. 2, resources are composed of servers and clients and to replace a server more money is required than a client system. So the server has more asset value than client. Also form the information they contain server has more asset value than a client.

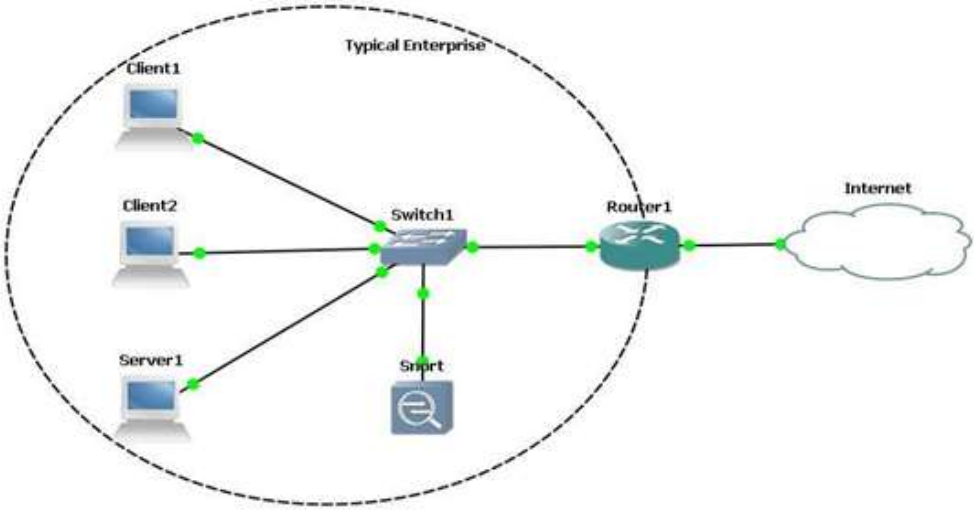


Fig. 2 Typica Fig.2 Typical Enterprise Setup

The main aim of the study is to, when an attack is generated by Snort, the severity of the attack for each individual information resource is evaluated based on frequency of alerts relevant to the resource and its asset value.

Now, if the severity for a particular resource higher than a predefined threshold value then the site is blocked for that resource. But for the resources for which the severity is less than the threshold value the sites remains available to them. However for such resources the incoming packets are now deeply inspected.

To design our scheme, we propose an integrated frame work as shown in Fig. 3, using existing tools and connecting codes that consist of the following modules.

1. Intrusion detection module: Snort will generate alerts to indicate potential intrusion. Alerts are stored in the MYSQL database.
2. Severity of attack checking module: The severity of the attack for each individual information resource is evaluated based on frequency of alerts relevant to the resource and its asset value.
3. IDS rule generation module: Read the alerts from the database and generate corresponding IDS rules to make router more alert on the attack by deeply packet inspection.
4. ACL rule generation module: Read the alerts from the database and generate corresponding ACL rules to block the potential intrusions.
5. Router configuration module: Access the router and execute the rules on the router to configure the IDS rules or ACL rules to deeply inspect the packet or block the connection.

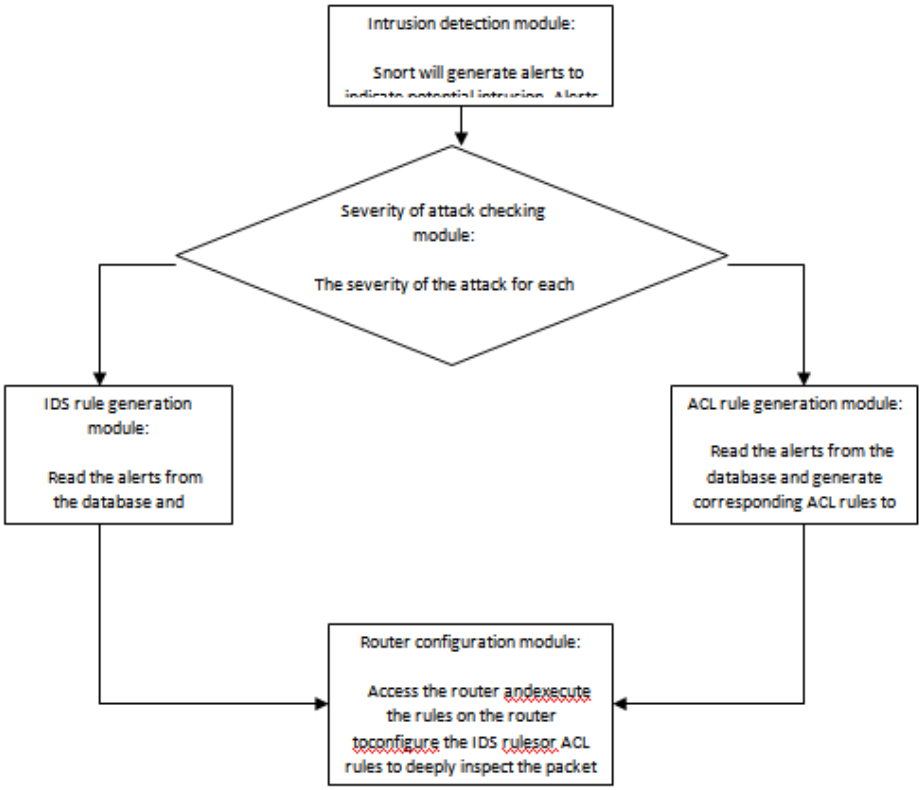


Fig.3 Integrated Frame Work

As shown in Fig 3, this is accomplished by generating ACL or IDS rules on the basis of alerts generated by Snort and then execute them on a router to block the potential intrusion or to do access after some deeply inspection of the packets.

The details of these 5 modules are discussed below:

1. Intrusion detection module:
2. Snort is used as an intrusion detection module to generate alerts for the potential intrusions. The alerts are automatically logged by Snort to My-SQL database from where they are read by the proposed module and are used to prevent the potential intrusion.

For alerts to be logged to a My-SQL database, Snort should be built with My-SQL capability at the time of compiling it. Once we have installed Snort with My-SQL capability and properly configured it using its configuration file, providing correct path for the rules, then whenever Snort will run in IDS mode, all of the alerts will be logged to a My-SQL database. This database then can be used to generate ACL or IDS rule for every alert logged to the database, which in a fine tuned IDS system represents a potential attack.

It was very important that Snort should be fine-tuned for the network because only then the false alarms will be minimum and almost all the alerts indicate potential intrusions. Current stable version of Snort rules was used for the experiments.

3. Severity of attack checking module:

Severity of attack checking is done on the basis of illegitimate traffic as well as Severity of the attack. If produced alerts show that attack has happened from the illegitimate source then it is better to block the traffic rather than deeply packet inspection. Then ACL rule generation module will be perform next rather than IDS rule generation module.

Severity of attack calculation is done by parameters like, like hood or frequency of the Threat occurrence vs. the vulnerability they exploit. We consider it as Security Concern function.

So, Security Concern = F (Threat, Vulnerability)

In our case we always consider Vulnerability=1 for reducing the complexity. So

Security Concern = F (Frequency of the Thread occurrence or Number of Alerts)

Risk Rate = F (Asset Value, Security Concern)

So a Risk Rate denotes the Severity of the attack. As it is shown by Fig.4.

Asset Name	Asset value (b)	Number of Alerts (c)	Measures of Risk Rate

Server A	10	9	90
Client B	2	34	68
Client C	3	30	90
Client D	4	20	80
Server E	8	10	80
Server F	7	10	70

- Value of asset is determined in terms of replacement or reconstruction costs
- Costs are converted onto qualitative scale

Fig.4 Risk Rate Calculation

Hence, proposed system can work at its best to inspect legitimate traffic, to make router more alert on the attack by deeply packet inspection or to block illegitimate traffic, the potential intrusions.

Block only activate for illegitimate traffic, while allowing legitimate traffic to enter network easily when the severity of the attack below some threshold value. Here in our proposed system blocking is treated as a lower priority than inspecting.

As today every router is shipped with some sort of IDS capability, generating IDS rules for deeply packet inspection would not be a problem alone with ACL rules generation. Here we refer IDS rule for deeply packet inspection until now every IDS only generate ACL Rules, but our proposed system generate IDS as well as ACL rules based on severity of the attack.

4. IDS rule generation module:

5. IDS rule generation module will be written in PHP and is used to access the database to read the alerts and based on these alerts, IDS rules will be generated. Snort's database has source and destination IP addresses and ports for each and every alert generated by Snort. This information can be easily accessed from the database and used to generate a specific IDS rule to make router more alert on the attack by deeply packet inspection**.**

IDS rule generation module connects to this database and check for any new alerts generated by Snort. If there is any new alert, it queries the database for the "iphdr" table in the database, which contains information about the IP header of the packets that generated the alert. After query, IP header of every alert is fetched. The "Protocol" field in the IP header is checked to find the upper layer protocol and according to the value of the field and corresponding upper layer protocol, other table (icmp=1, tcphdr=6, udphdr=17) are selected to gather additional information about the source of intrusion as shown in Fig. 5

.

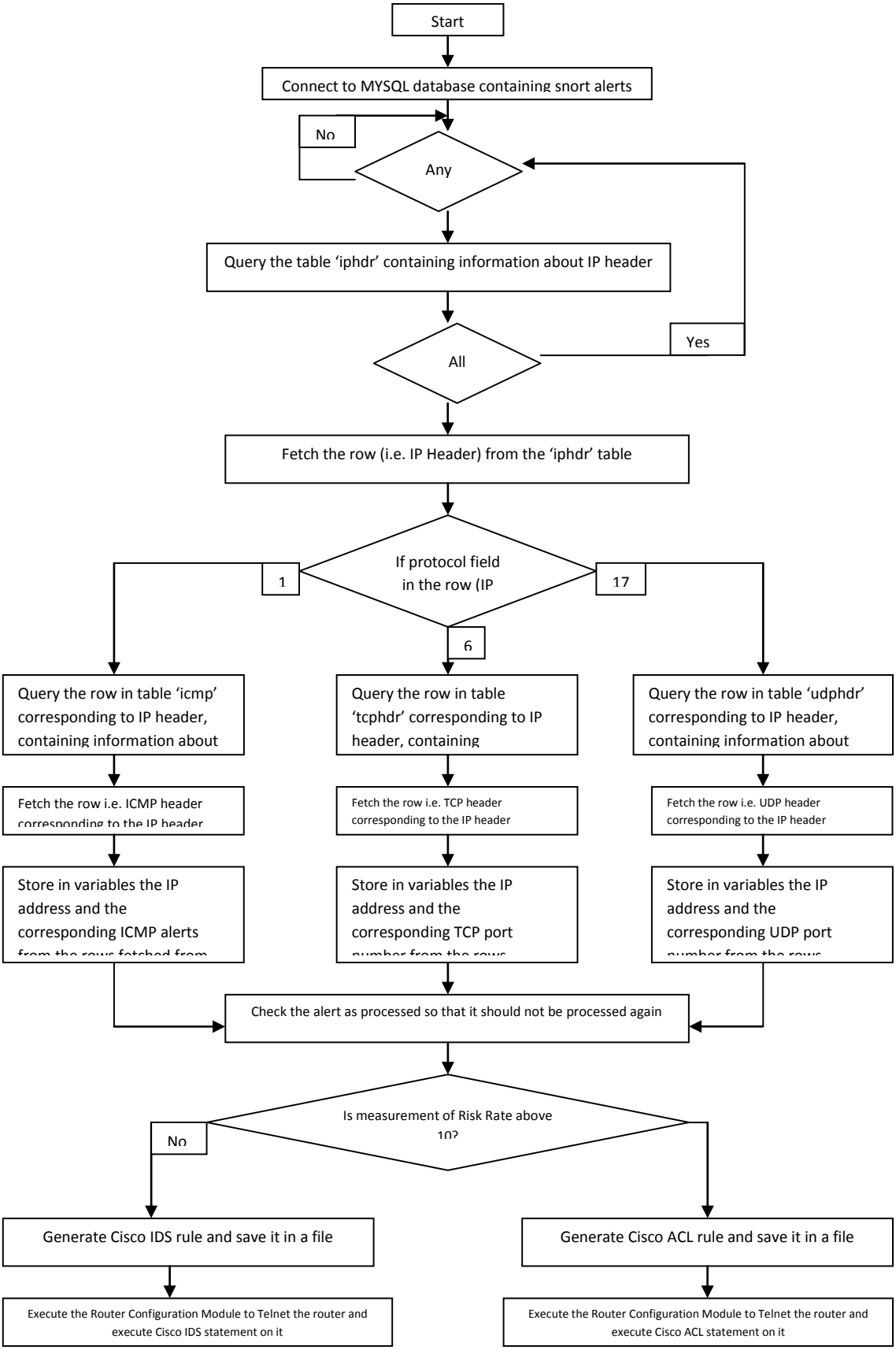


Fig. 5 Dynamic Configuration of IDS or ACL rules on the Router

4. ACL rule generation module: ACL rule generation module will be written in PHP and is used to access the database to read the alerts and based on these alerts, ACL rules will be generated. Snort's database has source and destination IP addresses and ports for each and every alert generated by Snort. This information can be easily accessed from the database and used to generate a specific ACL rule to block the potential intrusions.
ACL rule generation module connects to this database and check for any new alerts generated by Snort. If there is any new alert, it queries the database for the "iphdr" table in the database, which contains information about the IP header of the packets that generated the alert. After query, IP header of every alert is fetched. The "Protocol" field in the IP header is checked to find the upper layer protocol and according to the value of the field and corresponding upper layer protocol, other table (icmp=1, tcphdr=6, udphdr=17) are selected to gather additional information about the source of intrusion.
After generating the ACL rules the router configuration module access the router automatically using telnet and configure the ACL rules on it. ACL rules can also be removed after the attack is over or if the configured ACL rules have some undesired effect on the network. Removal mechanism is also proposed.
6. Router configuration module:
Router configuration module is basically designed to access the router and configure it automatically. Router configuration module written in Perl. For using telnet in a Perl script, Perl Net Cisco telnet [6] module is needed. By using this module, the router can be accessed and commands can be entered to configure the router. First of all we need to instantiate Net::Cisco::Telnet object and specify a timeout in case the expected prompt does not match to the router prompt. All methods used in this module are of the Net::Cisco::Telnet object. To connect to router using telnet open () method will be used. Router configuration module then waits for the vt (virtual terminal) "Password:" prompt on the router. Password is provided by the script to the router and router enters into "User Mode". In "User Mode" we do not have access to configure the router, so, now we should switch to "Privileged Mode". Now the Router configuration module sends "enable" command to the router to switch to "Privileged Exec Mode", the router asks for the "Privileged Exec Mode" password, which the module will provides. Now we have to switch to "Global Configuration Mode", using "configure terminal" command. Now, in this mode the IDS or ACL rules can be configured on the router by simply sending the string (i.e. an IDS rule or an extended ACL rule) passed to Router.
The details of software interface are as shown in Fig.6 below.

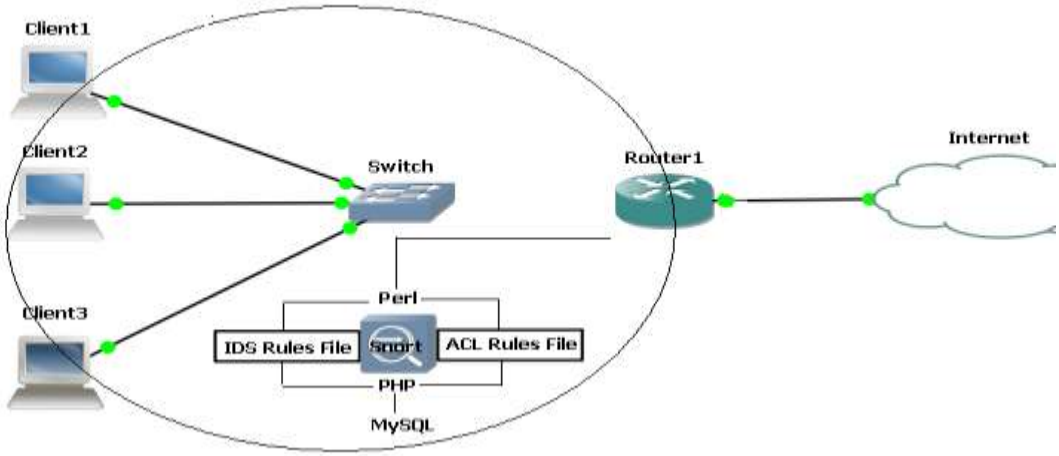


Fig.6Software Interface

III. Implementation Of The Scheme

As depicted in Fig. 1, we have taken three thousands machines each with Dual Core processor with 2 GB RAM, where all machine are connected through several switches and form a LAN for our experiment. Among them thirty machines was use for Server with Windows 2000 installed and the remaining machines was use for Client with Windows XP installed. In another machine I have installed Fedora 14 with Snort version 2.9.0.2 was installed. A Cisco router 3700 with IOS c3725-advsecurityk9-mz.124-3 was connected with the LAN and Internet [8]. The IOS version was chosen with build in IDS capabilities.

The software was written in PHP [13][14] and Perl [12][15], so PHP and Perl were also installed on the snort system which we was use as an IDS to generate the alerts and dealing with that generated alerts. The PHP version was v5.3.6 and the Perl version was v5.12.2.

As the main aim of the software was to configure IDS or ACL rules based on the Snort alerts, so, we used the build in snortrules-snapshot-2861.tar.gz rules to configure our Snort rules which was by default installed in /etc/snort/rules directory. As all the rules will produce too many alerts so we fine tune these rules and create a customize rules in local rules file for our testing purpose. We create a customize rule for Google page access and according to that, the page will be block in Server machine whereas it will open on Client machine after verification through IDS rules.

#alert tcp any any <> any 80 (msg: "Test web activity"; sid:1000001)

MIT DARPA Intrusion detection data set [9] can be use to test our proposed system as it produce too much alert but at first we want to test our software with small number of alerts generated that why we create our own custom rule to generate minimum number of alerts.

IV. Experimental Setup And Observed Result

As Snort will be our main IDS, so we installed snort in Fedora 14 machine with the proper configuration and integration with MYSQL as alerts that will be generated by snort will be store in MYSQL database which we use as a primary source to produce IDS or ACL rules through our software.

To install snort with MYSQL capabilities we first build up the fedora machine with additional software required which I found from sorce forge how to install snort with MYSQL capabilities. We also installed barnyard2 software to graphically see the alerts generated by snort in BASE (Basic Analysis and Security Engine) [4] [5] web pages.

We automated snort to run in IDS mode as like a demon.

The software which we build for IDS or ACL rule generation is open in a web browser. In which we first give information about the IP addresses of server and client along with their asset value which are in the range of 1-10 where costs are converted onto qualitative scale. These IP address is first converted into snort like integer number and store in a separate table IAAR (IP address of Server and Client, Asset Value, Number of Alerts, and their Risk Rate). As we fetch every row from the 'iphdr' table we check it not to process further and increments the Number of Alerts field which shows number of alerts generated for a particular server or client. Then we use this number and Asset Value to calculate the Risk Rate of an alert. If the Risk Rate shows a value above 50 then we are going to generate ACL rules through ACL rule generation module otherwise we generate IDS rules through IDS rule generation module. This threshold value is also taken as an input from the web browser.

All of the ACL rules we create have 15 minutes timeout value so that they will be deleted after 15 minutes when the attack are no longer exist.

Then the snort system was connected with the route through Fast Ethernet interface over where we fire the generated IDS or ACL rules which was created by our software in a file. We actually execute the file through Router configuration Module which contains the generated IDS or ACL rules. At the end of the ACL rules we explicitly specify the permit rules for other traffic. In other cases we create an extended ACL to block the source of the traffic and fire it in the internal interface of the router.

We first see that an alert has been generated by snort for our custom rule,

#alert tcp any any <> any 80 (msg: "Test web activity"; sid:1000001), which we can see from the BASE web interface.

The IDS and ACL rules produce by our software and fire in the router, can be see through "show running-configuration" and "show access-lists" command which show the following result on the router.

IDS Rules:

```
ip ips name MY_IPS list 100
!interface FastEthernet 1/0
 ip ips MY_IPS in
!
access-list 100 permit  ip host 197.218.177.69  host 192.168.1.2
access-list 100 permit  ip host 197.218.177.69  host 192.168.1.3

access-list 100 deny ip any any
!
```

ip ips name MY_IPS list 100 - Creates an IPS rule named MY_IPS and filters it against access list 100. Packets matching a deny statement in the ACL bypass the IPS engine, whereas packets matching a permit statement are scanned with the IPS engine.

ip ips MY_IPS in - Specifies that packets inbound to the interface are scanned with the IPS rule MY_IPS.

ACL Rules:

Extended IP access list 100

deny tcp host 197.218.177.69 host 192.168.1.1

deny tcp host 197.218.177.69 host 192.168.1.1 - A connection from host 197.218.177.69 , which is most probably the Google site, blocked when it was access from the server with an IP address 192.168.1.1

It is clear from the results obtained that all the incoming traffic (as the access lists are configured on the interfaces for incoming traffic) from the above IP addresses, which are the source of potential intrusion will be either blocked when it is designate for the server which has an ip address 192.168.1.1 or will be deeply inspected by IDS of the router when it will be designated for the client and hence our system has successfully prevented intrusion into the network.

It should be noted that only fine tune of a snort rule produce less alerts and accordingly the generation of IDS or ACL rules should be less otherwise everything can be block. In our system blocking is treated as a lower priority than deeply inspection. So more IDS rules has been generated rather than ACL rules which show the significance of our experiment.

In general, if there are hundred machines and twenty machines are use for server, then 80 rules are generated for deeply inspection whereas 20 rules is generated for blocking. So percentage of blocking =20/100*100=20 %.

Whereas in our previous system blocking was 100 %. Although we have to compromise some risk factor. So it can be graphically shown as Fig. 7 below.

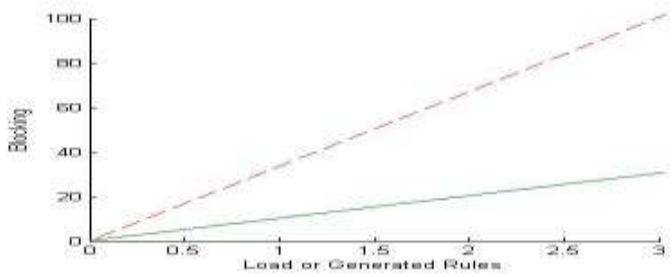
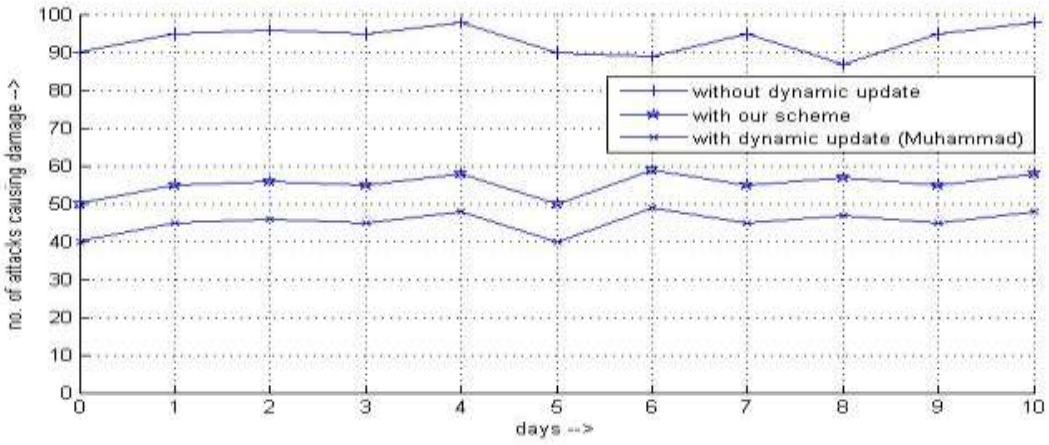


Fig.7 Percentage of Blocking

The system was designed, developed and deployed at the laboratory and its performance was analyzed in contrast to the existing manual IDS based scheme. The results obtain from 3000 machines over a period of 10 days, is given below, with a comparison between old schemes.

Fig.8 days vs. no. of attacks



causing damage

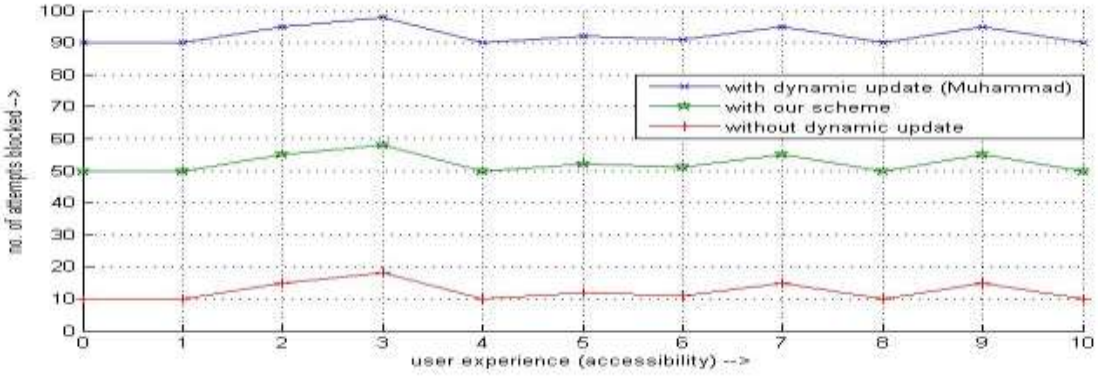


Fig.9 user experience (accessibility) vs. no. of attempts blocked

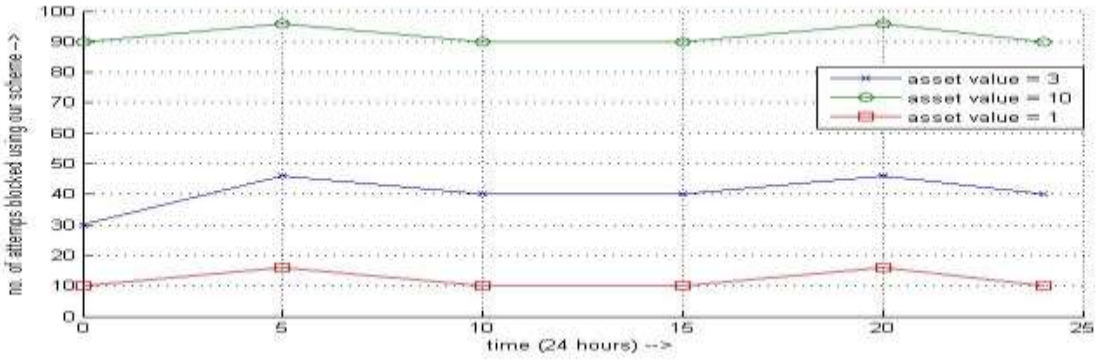


Fig.10 time (24 hours) vs. no. of attempts blocked using our scheme

The proposed system, after deployment, is providing much improved user's experience without hampering the security risks of vital information resources.

V. Conclusion

In this work we propose a mechanism by which firewall rules are updated by capturing the alerts generated by IDS, but the sites are not blocked for all the information resources. Whether a site will be blocked for an information resource depends on Risk Rate of the resource. If the Risk Rate for an information resource exceeds a predefine threshold value then the site will be blocked for that particular resource. However the site will be available to all other resources having Risk Rate less than the threshold value.

The system was designed, developed and deployed at the laboratory and its performance was analyzed in contrast to the existing manual IDS based scheme. The proposed system, after deployment, is providing much improved user's experience without hampering the security risks of vital information resources.

As the proposed system is Signature-based or Rule-based system, it doesn't protect from novel attacks. This may be extended to include Anomaly-based detection so new attacks can be detected.

Moreover, the proposed scheme addresses the problem from a centralized point and for large enterprise it may introduce unacceptable delay and overhead. The problem can be resolved by using multiple IDS distributed over the enterprise and maintaining a distributed database or distributed hash-table.

Acknowledgements

We would like to express our deep sense of gratitude to the organizing committee of DSMS Group of Institution. Special thanks to all the laboratory staff of Asansol Engineering College for their noble help and cooperation to carry out our experiments.

References

[1]. Snort at http://www.snort.org
[2]. Cisco systems at http://www.cisco.com
[3]. MySQL database at http://www.mysql.org
[4]. ACID (Analysis Console for Intrusion Databases) at http://acidlab.sourceforge.net/
[5]. BASE (Basic Analysis and Security Engine) at http://sourceforge.net/projects/secureideas/files/
[6]. Net-Telnet-Cisco 1.10 at http://search.cpan.org/~joshua/Net-Telnet-Cisco-1.10/Cisco.pm
[7]. PHP at http://php.net
[8]. GNS3 at http://www.gns3.net/
[9]. 1998 DARPA Intrusion Detection Evaluation Data Set at http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/1998data.html
[10]. Karen Scarfone, Peter Mell, "*Guide to Intrusion Detection andPrevention Systems (IDPS)*," 2007, Special Publication 800-94,Recommendations of the National Institute of Standards and Technology at http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf
[11]. Intrusion detection FAQ at http://www.sans.org/
[12]. Rafeeq Ur Rehman, "Intrusion Detection Systems with Snort, Advanced IDS Techniques Using Snort, Apache, MySQL, PHP, and ACID", Prentice Hall PTR atwww.phptr.com
[13]. Brad Bulger, Jay Greenspan, and David Wall, "MySQL/PHP Database Applications", Wiley, Second Edition-2004.
[14]. Timothy Boronczyk, Elizabeth Naramore, Jason Gerner,Yann Le Scouarnec, Jeremy Stolz, and Michael K. Glass, "Beginning.PHP.6.Apache.MySQL.6.Web.Development", Wrox,Jan.2009
[15]. Kerry J. Cox, Christopher G,"Managing Security With Snort And Ids Tools", O'Reilly, 1St Edition
[16]. Muhammad Naveed, Shams un Nihar and Mohammad Inayatullah Babar "Network Intrution Preventation by Configuraing ACLs on the Routers, based on Snort IDS alerts" in Proc. 6th International Conference on Emerging Technologies (ICET), 2010, pp. 234-239.
[17]. Guardian security software at http://www.chaotic.org/guardian/
[18]. Peyman Kabiri and Ali A. Ghorbani "Research on Intrusion Detection and Response: A Survey" in Proc. International Journal of Network Security, Vol.1, No.2, PP.84–102, Sep. 2005 at http://isrc.nchu.edu.tw/ijns/