# ADAPTIVE FIREWALLING IN AN UNIX ENVIRONMENT

**Octavian Stefan**

*Automation and Applied Informatics Department,*
*Politehnica University of Timisoara*
*Timisoara, Romania*
*octavian.stefan@aut.upt.ro*

Abstract: With the rapid growth of computer networks, security has become an essential aspect of all operating systems. Rules based firewalls are used to deny unwanted access, but those firewalls are vulnerable to unknown attack types missing from the rules database. This paper propose a different firewalling approach, using a firewall with dynamic rules set altering possibility and an intrusion detection system which combines a misuse detection component, an anomaly detection component and a log analysis component.

Keywords: network-centric computing, artificial intelligence and expert systems.

## 1. INTRODUCTION

There are some well known advantages in having a computer connected to the Internet, but there are some drawbacks too, because the highly connected computing world provides malicious users with the necessarily means for their destructive purposes. One tool for preventing unauthorized access to a host or a network is the firewall. Most firewalls have a user defined rules set which is loaded when the firewall is activated. Based on those rules, a firewall can deny, allow or proxy the network traffic. Even with a good rule set, the system can be vulnerable to unknown or unlisted attack types. Some firewalls permit a user intervention, when an unknown situation occurs and ask the system administrator for a decision, but this approach is almost in all situations unacceptable.

This paper proposes a different approach, an adaptive firewall, where new rules are permanently added or removed based on the decisions made by an intrusion detection component.

Intrusion detection schemes based on traffic analysis can be divided in two categories: misuse and anomaly detection schemes.

The misuse method relies on a set of labeled data, a traffic pattern for each different type of attack, upon which the learning algorithm is trained. This method is incapable of detecting new and undocumented types of attack, but it is very efficient for the ones it knows.

The anomaly method builds databases of normal traffic data and then tries to detect any anomaly related to this data. For the training part this method needs a set of normal data, without any trace of attack traffic, because the model may not detect this attack type in the future.

The anomaly detection technique relies upon two axioms: “The majority of the network connections are normal traffic. Only X% of traffic is malicious”. (Leung and Leckie, 1994) and “The attack traffic is statistically different from normal traffic” (Leung and Leckie, 1994).

Attacks fall into four categories: probing – is a class of attacks where an attacker scans a network to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on the network can use the information to look for exploits; denial of service – is a class of

attacks where an attacker makes some computing or memory resource too busy or too full to handle legitimate requests, thus denying legitimate users access to a machine. There are different ways to launch DoS attacks: by abusing the computers legitimate features, by targeting the implementation bugs or by exploiting the system configurations; user to root attacks – is a class of attacks where an attacker starts out with access to a normal user account on the system and is able to exploit vulnerability to gain root access to the system. Most common exploits in this category are regular buffer overflows, which are caused by regular programming mistakes and environment assumptions; remote to user attack – is a class of attacks where an attacker sends packets to a machine over a network, then exploits machine's vulnerability to illegally gain local user access as a user. (Mukkamala et al., 2003).

Not all types of network related attacks are detectable by analyzing the traffic, an example being a user to root attack made from within a ssh tunnel. This type of attacks requires a log and maybe a file system and user commands analysis to detect.

The rest of the paper is organized as follows: in section 2 it is described every component of the adaptive firewall, in 3 the system is tested separately for each component, in 4 the results are discussed and in 5 some conclusions are set.

## 2. THE ADAPTIVE FIREWALL

The first component is the actual firewall, for which I used the OpenBSD's packet filter (PF). I used PF for its online anchor mechanism (Holland et al., 2007).

The misuse component is an expert system being able to learn, beside the traffic patterns for different types of already known attacks, the patterns discovered using the anomaly detection component. The patterns contain sets of packet headers information for different types of low level protocols from the TCP/IP network model like Ethernet header, IP header, TCP header, UDP header, ICMP header. Incoming traffic is matched against these patterns and if a perfect match is found it triggers an action resulting in a deny access rule for the source IP address being injected into the firewall online rules using the anchor mechanism.

The log analysis component scans the online log for known log entries of certain attacks, like unknown user login attempt or incorrect password. If the log system entries match the database entries for this component again an action resulting in a deny access rule for the specific IP address is triggered.

There are a lot of proposed algorithms for anomaly intrusion detection systems in the specialized literature: using payload algorithms (Wang and Stolfo, 2004), genetic algorithms (Li,2004), visual methods (Goodall,2006), adaptive regression splines (Mukkamala et al.,2003), text categorization techniques (Liao and Vemuri,2002), hashing functions and clustering (Mahoney and Chan,2001), neural networks (Ryan et al.,1998), (Debar et al.,1992), (Fox et al.,1990) and (Frank 1994) , with promising experimental results denoting that an anomaly detection component may be used with success in an adaptive firewall.

Our anomaly detection component will use a neural network for anomaly detection, but with a different approach, using packet header data as input data.
Studying in detail a set of possible attacks we can state that almost all network attacks use a modified packet header, with legal or illegal field values, for the purpose of exploiting hidden vulnerabilities in networks. Other attacks, like for example the syn flood attack in Figure 1, use massive amount of identical packets, issue easy to report by studding the packet header.

I will only use low level protocol headers (Ethernet, IP, TCP, UDP and ICMP) fields, because using a different neural network for every application layer protocol would result in an oversized detection algorithm with extensive memory and computational needs without noticeable positive results in the detection.

Unfortunately, the TCP/IP network model low level protocols headers design is the principal problem for today's network attacks. Figure 2 and Figure 3 show examples of an IP header and a TCP header. For optimizing the detection algorithm, I will merge the one bit fields into 1 byte fields for the neural network input data.

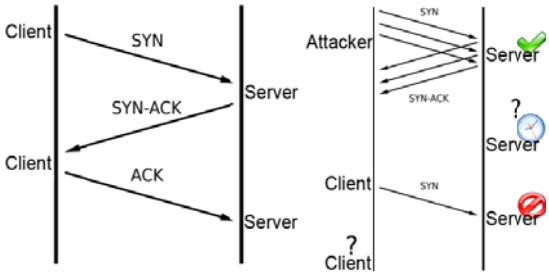


Fig. 1. Syn flood example

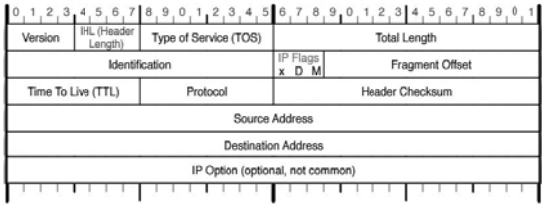


Fig.2. IP header

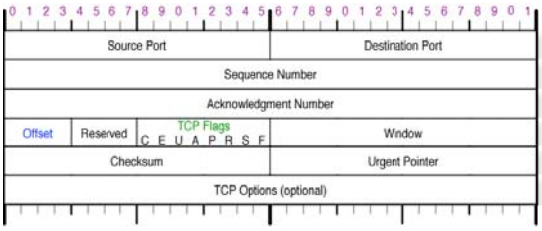


Fig.3. TCP header

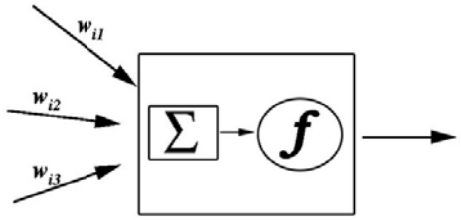


Fig.4. Mathematical representation of a neuron

Artificial neural networks are computational models with the ability to learn, adapt and organize data.

An artificial neural network consists of a pool of simple processing units called neurons communicating with each other over a large number of weighted connections.

Figure 4 describes a mathematical representation of a neuron. Each neuron input has an associated weight which can be modified to model synaptic learning. The neuron output computes a function of the weighted sum of inputs.

I use back-propagation neural networks for the anomaly component, which are very good in finding out the nonlinear correlation between inputs and outputs.

Every packet header type has its own neural network.

The number of neurons for each level depends on the packet header's field number.

Empirically, it can be demonstrated that the additional computation need for a network with more than one hidden layer is not justified by the additional detection rate, so I used a single hidden layer neural network like in figure 5.

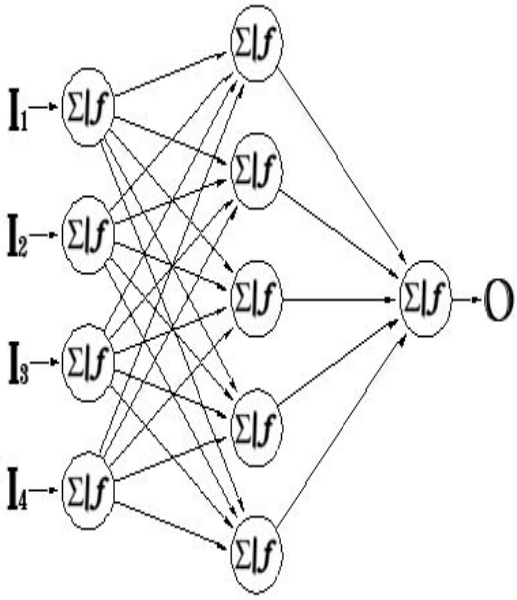


Fig.5 Neural network with one hidden layer and a superposition neuron

## 3. EXPERIMENTS

I conducted the experiments in a controlled, small environment using a total of 25 computers in a local area network. The operating systems used were OpenBSD and FreeDSB. After a training period of 10 days, when the network was injected with normal application traffic, I started the injection of attack simulation traffic for each intrusion detection system component. Results were collected and a conclusion for each component has been made. For the misuse component and for the log analysis component an online pattern database was used.

After the training period was completed and after I tested each component in the controlled environment, I used a demilitarized zone network for testing the whole system in a real life environment.

## 4. RESULTS

For a more conclusive result I tested each intrusion detection component separate.

Table 1 presents the misuse component reaction to some randomly chosen attacks. As we can see the component works very well with a 98% detection rate for known attacks. In the case of bruteforceroot attack it could not detect the attack because the traffic was encapsulated in encrypted packets by the ssh daemon.

Table 1. Misuse component detection rate for some randomly chosen attacks

| Attack | Description | Det. rate |
|---|---|---|
| apache2 | DOS, HTTP overflow in apache web server | 1/1 |
| arppoison | DOS, spoofed ARP with bad IP/Ethernet map | 3/3 |
| bruteforceftp | R2L, FTP password guessing | 50/50 |
| bruteforcetelnet | R2L, telnet password guessing | 40/40 |
| bruteforceroot | U2R,root password guessing inside a ssh session | 0/40 |
| ipsweep | Probe, tests for valid IP addresses | 10/10 |
| mailbomb | DOS, mail server flood | |
| named | R2L, DNS buffer overflow | 2/2 |
| neptune | DOS, SYN Flood | 3/3 |
| nmap | Probe, tests for listening ports | 48/50 |
| pod | DoS, oversized IP packet | 5/5 |
| sendmail | R2L, SMTP buffer overflow | 3/3 |
| smurf | DOS, distributed ICMP echo reply flood | 5/5 |
| teardrop | DOS, overlapping IP fragments | 4/4 |
| udpstorm | DOS, echo/charge loop using spoofed request | 2/2 |

Table 2. Misuse component detection rate for some randomly chosen attacks

| Attack | Description | Det. Rate |
|---|---|---|
| apache2 | DOS, HTTP overflow in apache web server | 0/1 |
| arppoison | DOS, spoofed ARP with bad IP/Ethernet map | 0/3 |
| bruteforceftp | R2L, FTP password guessing | 50/50 |
| bruteforcetelnet | R2L, telnet password guessing | 40/40 |
| bruteforceroot | U2R,root password guessing inside a ssh session | 40/40 |
| ipsweep | Probe, tests for valid IP addresses | 0/10 |
| mailbomb | DOS, mail server flood | |
| named | R2L, DNS buffer overflow | 0/2 |
| neptune | DOS, SYN Flood | 0/3 |
| nmap | Probe, tests for listening ports | 0/50 |
| pod | DoS, oversized IP packet | 0/5 |

Table 2 contains the results for the log analysis component. Because this component doesn't analyze traffic but the system logs, it cannot react to unlogged attack events as we can see from table 2.

This component comes as an add-on for the two traffic related detection components, but it is very useful for attacks like bruteforceroot in a ssh tunnel, attacks transparent for the other components.

For an operation system with a good log process this component has a very important role in the intrusion detection.

The anomaly detection component results are presented in Table 3. As we can see, it has a success rate of 61 % in finding attacks.

The anomaly detection component injected some false alarms into the system which could lead to normal traffic filter. The number of false alarms was very low, only a few per day. This is something we have to consider acceptable for the system in order to use the adaptive firewall.

The computational need for our components was very low. In our case on dual processor systems the CPU use was under 0,1 %.

Table 3. Anomaly component detection rate for some randomly chosen attacks

| Attack | Description | Det. Rate |
|---|---|---|
| apache2 | DOS, HTTP overflow in apache web server | 1/4 |
| arppoison | DOS, spoofed ARP with bad IP/Ethernet map | 2/3 |
| bruteforceftp | R2L, FTP password guessing | 47/50 |
| bruteforcetelnet | R2L, telnet password guessing | 36/40 |
| bruteforceroot | U2R,root password guessing inside a ssh s. | 2/40 |
| ipsweep | Probe, tests for valid IP addresses | 10/10 |
| mailbomb | DOS, mail server flood | |
| named | R2L, DNS buffer overflow | 2/2 |
| neptune | DOS, SYN Flood | 2/3 |
| nmap | Probe, tests for listening ports | 25/50 |
| pod | DOS, oversized IP packet | 1/5 |
| sendmail | R2L, SMTP buffer overflow | 1/3 |
| smurf | DOS, distributed ICMP echo reply flood | 2/5 |
| teardrop | DOS, overlapping IP fragments | 1/4 |
| udpstorm | DOS, echo/charge loop using spoofed request | 2/2 |

Table 4. Adaptive firewall detection rate for real time known attacks

| Attack | Description | Det. Rate |
|---|---|---|
| apache2 | DOS, HTTP overflow in apache web server | 4/4 |
| arppoison | DOS, spoofed ARP with bad IP/Ethernet map | 3/3 |
| bruteforceftp | R2L, FTP password guessing | 5/5 |
| bruteforcetelnet | R2L, telnet password guessing | 4/4 |
| bruteforceroot | U2R,root password guessing inside a ssh session | 4/4 |
| ipsweep | Probe, tests for valid IP addresses | 1/1 |
| mailbomb | DOS, mail server flood | |
| named | R2L, DNS buffer overflow | 2/2 |
| neptune | DOS, SYN Flood | 3/3 |
| nmap | Probe, tests for listening ports | 5/5 |
| pod | DOS, oversized IP packet | 3/3 |
| sendmail | R2L, SMTP buffer overflow | 3/3 |
| smurf | DOS, distributed ICMP echo reply flood | 5/5 |
| teardrop | DOS, overlapping IP fragments | 4/4 |
| udpstorm | DOS, echo/charge loop using spoofed request | 2/2 |

Testing the adaptive firewall in the real time environment had a success rate of 100 % for known attacks types and 67% percent rate for unknown attacks. Success rate was a bit higher in the real life environment because there were some illegal field values for some of the protocols' headers made on purpose or by mistake by the hackers which determined the anomaly component to react.

## 5. CONCLUSIONS

The adaptive firewall has a lot of visible advantages compared with a rule based one. A rule base firewall if is poorly configured shows almost no protection at all, unlike a poorly configured adaptive one which has a 60% protection rate even if it doesn't contain any patterns in the database.

Another aspect is that even if the rule based firewall is excellent configured it can't protect a host or a network for attacks from persons we consider friends at the time of rules definition and even worse it cannot protect our public services for most types of attacks.

An adaptive firewall protects a host or a network for indirect attacks, too. If a person is trying a type of attack he would probably try some other attack after to discover the system vulnerability, but because he was first detected, he will be filtered and he could not try another type of attack on the system.

An adaptive firewall may have some disadvantages because it could generate false alarms and filter traffic it shouldn't filter. Because of that a decision should be made if the adaptive firewall should be used in critical application systems.

Used in conjunction with a rule based firewall, an adaptive firewall could be the best protection a system can get against unwanted traffic.

REFERENCES


Debar, H., M. Becker and D. Siboni (1992). A neural network component for an intrusion detection system. In *Proceedings of the 1992 IEEE Computer Society Symposium on Research in Computer Security and Privacy*, **240-250**

Fox, K. L.,R.R. Henning, J.H. Reed and R. Simonian (1990). A neural network approach towards intrusion detection. In *Proceeding of the 13th National Computer Security Conference*, **125-134**

Frank, J. (1994). Artificial Intelligence and intrusion detection: Current and future directions. In *Proceedings of the 17th National Computer Security Conference*

Goodall, J.R. (2006). Visualizing Network Traffic for Intrusion Detection. *Conference on Designing Interactive Systems*

Holland, N., J. Knight and S. Mestdagh (2007). OpenBSD FAQ. In *OpenBSD User Manual*

Leung, K. and C. Leckie(2005). Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters. In *ACSC 2005*, **333-342**

Li,W. (2004). Using Genetic Algorithm for Network Intrusion Detection. In *Proceedings of the United States Department of Energy Cyber Security Group*. Kansas City

Liao, Y. and V.R. Vemuri (2002). Using Text Categorization Techniques for Intrusion Detection. *11th USENIX Security Symposium*. San Francisco, CA

Mahoney, M.V. and P.K. Chan (2001). PHAD: Pachet Header Anomaly Detection for Identifying Hostile Network Traffic. In *Florida Tech. tehnical report CS-2001-4*

Mukkamala , S. , A.H. Sung and A. Abraham (2003). Intrusion Detection Systems using Adaptive Regression Splines

Ryan, J., M-J Lin and R. Mukkulainen (1998). Intrusion Detection with Neural Networks. In *Advances in Neural Information Processing Systems 10*, **943-949**

Wang, K. and S.J. Stolfo (2004). Anomalous Payload-based Network Intrusion Detection.