International Journal of Research
Available at https://edupediapublications.org/journals

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue-17
December 2017

Efficient Network Intrusion Detection System Using Boyer Moore Algorithm

Srinivas Kalime
Assistant Professor, Department of CSE Jayamukhi Institute of Technological Sciences, Narsampet
Warangal, Telangana, India

Abstract: *Network intrusion detection system is a retrofit approach for providing a sense of security in existing computers and data networks, while allowing them to operate in their current open mode. The goal of a network intrusion detection system is to identify, preferably in real time, unauthorized use, misuse and abuse of computer systems by insiders as well as from outside perpetrators.*

At the heart of every network intrusion detection system is packet inspection which employs nothing but string matching. This string matching is the bottleneck of performance for the whole network intrusion detection system. Thus, the need to increase the performance of string matching cannot be more exemplified.

Meanwhile, aiming at several key modules of intrusion detection system, a detailed analysis of packet capturing module, protocol dispensation module, feature matching module, log evidence module and Intrusion retort module is also given in this paper.

Index Terms- **Network intrusion detection system; packet Capturing module, Boyer-Moore**

I. INTRODUCTION

As with the rapid development of computer networks and distributed system, network security becomes more and more important nowadays. As available field in network security, intrusion detection arouses people's interest. There are some methods for detecting intrusions i.e. anomaly detection and misuse detection. The former is the main field in intrusion detection system (IDS) research.

A network intrusion detection system is an intrusion detection system. That tries to detect malicious activity such as denial of service attack, port scans, network flooding or attempts to crack into computers by monitoring network traffic. [2]

A network intrusion detection system reads all incoming packets and tries to find suspicious patterns known as signatures or rules. These rules are decided by a network administrator while the configuration and deployment of the network intrusion detection system based on the security and network policies of the organization. For instance, if it is observed that a particular TCP connection requests connection to a large number of ports, then it can be assumed that there is someone who is trying to conduct a port scan of all/most of the computers of the network. [2] .A network intrusion detection system is not limited to inspecting the incoming network traffic only. Patterns and outgoing intrusion can also be found from the outgoing or local traffic as well. Some attacks might also come from the inside of the monitored network, as in trusted host attack.

Intrusion detection is defined to be the problem of identifying individuals who are using a computer system without authorization (i.e., *crackers*) and those who have legitimate access to the system but are exceeding their Privileges (i.e., the *insider threat)*

At the heart of every modern network intrusion detection system there is a string matching algorithm. The network intrusion detection system uses the string matching algorithm to compare the payload of the network packet and/or flow against the pattern entries of the intrusion detection rules, which are a part of every network having a network intrusion detection system.

String matching needs significant memory and time requirements. In fact, the performance of all network intrusion detection systems depends almost entirely on the performance of the string matching algorithm

System Architecture

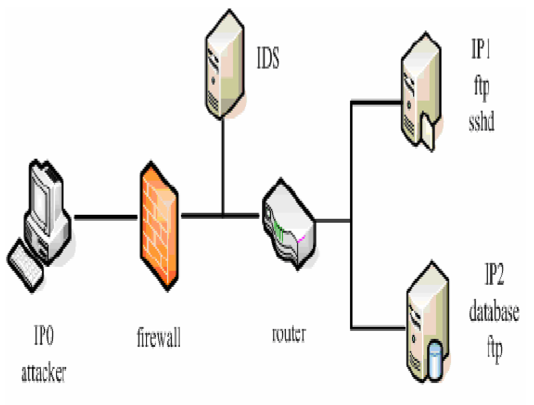


Figure 1. Example network

Related and Previous Work:

As an important aspect of network security, evaluating the computer security through the analysis to the computer network is very important and could protect us from being hacked. Vulnerability scanning is a traditional way to conduct network security analysis. This method can check whether or not there are any known vulnerabilities. This technique is just suitable to check system security qualitatively partially but cannot check a whole system. The ways to find the complex attack paths or list which can lead to changes of the system status are presented by analyzing the security model. For example, the earliest concept of attack graph a method named Privilege Graph is developed. The other model provides for modeling chains of network exploits. Some researchers analyzed these Unix-based systems security using model-checking technique.

2. **Packet Capturing Module**:

Packet capturing module can collect data packets from the network flow. If packet capturing module wants to monitor the data which flow through the network card (NIC) but don't belong to its own host, it must round the processing mechanism when the system works normally and access to the network bottom layer directly. first ,set the working mode of network card as promiscuous mode to make it receive data packets on target MAC address not its own MAC address and then access to the data link layer directly to capture relevant data. the data should be filtered by application program not upper layers, such as IP and TCP layer,etc,in this way ,all the data which flow through the network card can be monitored.

3. Protocol Dispensation Module:

This module includes all sub-modules aiming at concrete protocol processing like HTTP, SNMP, POP3, FTP, TCP, UDP, RPC processing module. A single protocol message transmitted from packet capturing module and packet preprocessing module(Which occurs after PCM But Before This) needs to be cached by a protocol processing module and a carriage return character is used to represent the end of this process according to the command end mode stated in the protocol. This can be detected by feature pattern matching, in the process of package, the corresponding treatment of some special keyed characters in concrete client protocol packages can be made, including command character, space character and backspace character,etc in the protocol itself.

Some intrusion behaviors can be found, before passing the data packet to the next module of it. Take TCP protocol processing module as an example, Here the sender can request the receiver more than once, but no response from the receiver, then the sender will understand, the request is attacked by some intruder. then An alarm signal is generated directly to be sent to the attack log record module and IRM,it's not necessary to hand the alarm signal to the pattern matching module., which not only enhances the real time performance of IDS but also reduces the resource burdens of using pattern matching to detect intrusion behaviors.

4. String Search Algorithms

Before we introduce our set wise Boyer-Moore-Horspool string matching algorithm, we briefly review the string matching problem, and the Boyer-Moore and Aho-Corasick approaches. More information can be found in Gus field's book [3] and a comprehensive review of string matching [4]. We focus specially on understanding the main ideas behind standard Boyer-Moore because they are essential to understanding many of the Multiple-pattern algorithms, including our set wise Boyer-Moore-Horspool algorithm.Assume a text string T of length n and a pattern string P of length m,each composed of an ordered set of characters from an common alphabet A. The general problem is to determine the location of P within T, or that T does not contain P. Let the characters of T and P be numbered sequentially from left to right starting with one. If, for a given offset or shift s, every character Pi 2 P1: Pm matches the corresponding character Ti+s 2 T1: Tn, then P occurs at offset s in T. We write this

Equality as Ti+s: Tm+s = Pi: Pm

4.1. Boyer-Moore

Current Snort implementations use the Boyer-Moore string matching algorithm, which is widely regarded as th providing the best average-case performance of any known Algorithm. The algorithm is based on a few key observations that allow single patterns to be found in sub linear time in the average case [5].

The first observation is that character comparisons can be made from right to left Starting at the end of the pattern instead of the beginning. Let e be the endpoint of the Pattern and e = m initially. For j = 0.(m - 1), we compare Te-j and Pm-j . Shifts are logically performed by incrementing e.

Bad Character Heuristic: Second, consider when a character mismatch is found Between Te-j and Pm-j , so that j is the length of the matching suffix of P. If the last Occurrence in P of the character Te-j occurs q characters from the end of P, then there is no point in trying new endpoints less than e + q -j. This *bad character heuristic*

allows us to skip many of the comparisons of the naive algorithm. The amount that we can shift the endpoint is easily computed based on a function B(c) that computes the Distance from the end of P to the last occurrence of character c:

B(c) = min {q | Pm-q = c}

For each character c that does not occur in the pattern at all, B(c) = m. All values for B can be pre-computed and stored in an array of length jAj. If we find a character mismatch at Te-j , then we can safely set the endpoint e to e + B(Te-j) - j and restart the right to left comparison of Te-m+1:::Te and P1:::Pm Note that the last occurrence may not be to the left of the current location e j, and so B(Te-j)-j _ 0. In this case the bad character heuristic provides no insight and the endpoint must be shifted by just one additional position.

Good Suffixes Heuristic: The *good suffix heuristic* is the third key observation. If a mismatch is found in the middle of the pattern at Pm-j , then there is a suffix of 0 or more characters Pm-j+1:::Pm that do match. Thus, there is no point in testing new endpoints e0 that do not cause that same suffix string to match (Te'-j+1: Te' 6= Pmj+1: Pm).

A stronger version of the good suffix heuristic is attributed to Kuipers. This version also skips over reoccurrences of the suffix that are preceded by the same character as Pm-j. There is no point in attempting such shifts since it is known that Te-j 6= Pm-j this optimization does not significantly complicate the preprocessing step, and allows the algorithm to be provably linear even in the worst case [3]. The full Boyer-Moore algorithm shifts by the greater of the values given by the Bad-character and good-suffix heuristics.

Performance Analysis of BM: Clearly the performance of the BM algorithm depends on the characters in the text and pattern strings. Knuth showed that the original Boyer-Moore algorithm performs O(n + rm) comparisons, where r is the number of times the pattern occurs in the text. For r = 0, 6n comparisons are required [6, 4].

Cole showed that Boyer-Moore with the stronger good suffix heuristic requires only 4n comparisons if the pattern does not occur in the text. When the pattern occurs r times in the text, the number of comparisons is _ (rn)[3]. This complex analysis does not even include any advantages caused by the bad character heuristic.

Galil developed a slight modification to Boyer-Moore that allows a proof of O(n) performance regardless of the number of occurrences [7, 3]. In 1986, Apostolic and Giancarlo developed a variant of Boyer-Moore algorithm that requires at most 2n comparisons, but requires O(m) additional space [8, 4, 3]. In 1996, Crochemore and Lecroq showed that the Apostolic-Giancarlo algorithm has a tight bound of at most 1:5n comparisons [9]. This bound is better than the 2n of the KMP algorithm and encroaches on Rivet's lower bound of n-m+1 comparison for any algorithm's worst case performance [10].

Many performance studies of Boyer-Moore and its variants have been performed. Despite worst-case performance that is super-linear, the observed average case performance is typically much better. Horspool studied a version of Boyer-Moore that does not perform the good suffix comparisons. This version is O(nm) in the worst case, but performs comparably to the original Boyer-Moore algorithm in the average case [4].

5. *Log evidence (record) module:*

The log record of intrusion detection system based on attack feature pattern matching can be divided into two types, namely attack log record and protocol operation log record. Attack log record provides essential information for the network manager after intrusion events occur. Protocol operation log record provides a set of comprehensive, operating system-independent and searchable log record, which can be used to audit and trace intrusion behaviors afterwards. when protocol processing module deals with corresponding protocol messages respectively, if an intrusion behavior is found, it will directly calls the attack log record module to make an attack log record; if no intrusion behavior is found, it calls the protocol operation log record module to make a normal protocol operation log record. Similarly, when attack feature pattern matching module (using Boyer Moore) is matched to a certain attack feature, it will also call the attack log record module.

5.1. Attack register module

Attack register (record) module is used to record the necessary information of intrusion behaviors detected by intrusion detection system complete attack log record should include the information below:

occurrence time of intrusion behavior, source address of intruder, source port used by intruder, destination address of intrusion, objective of intrusion and name of port attack feature pattern.

5.2. *Protocol process register module:*

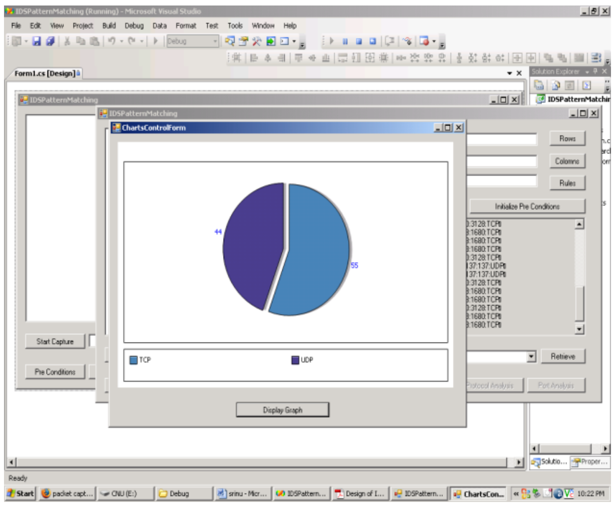
Protocol process register module is a Set of comprehensive and open service protocol-oriented operation log record independent of the log record of protected host, which provides detailed network or host access information for the system manager and can be used to trace the source of intruder or the intrusion detection system based on audit. Because of Different operation contents of various protocols, various protocols have different contents of operation log, generally including the information below:

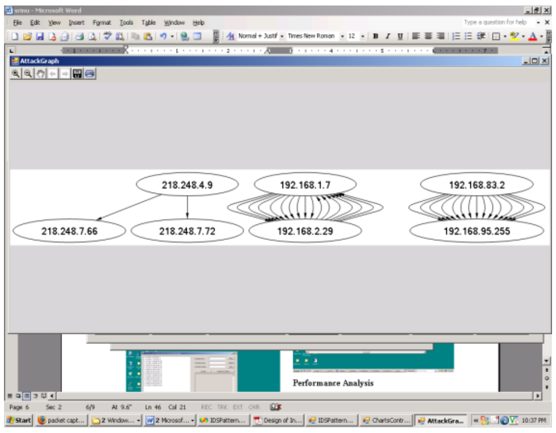
Protocol service name, occurrence time of operation, source address of accessor, source port used by accessor, destination address of access and destination port of access the rest are concrete operation contents of each application protocol.

6. Intrusion retort (response) module:

Intrusion retort module gives corresponding responses for verified intrusion behaviors, including static measures, for example, record attack data, store captured data, send emails and messages to the manager, it can also take some active dynamic preventive measures, cut off the intruded connection and modify the access control of router, for instance. Generally, the corresponding module of system mainly include very important functions, namely Alert(), logtoTable(UNIT SourIP, UNIT DestIP. CTime Timestamp). The function of Alert () is to give real-time warning of intrusion behaviors, the function of logtoTable (UNIT SourIP, UNIT DestIP.

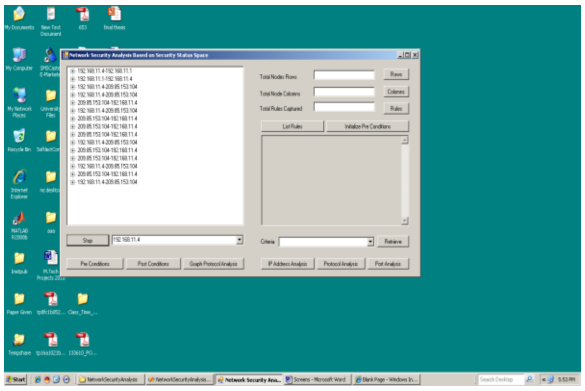
CTime Timestamp) is to set up a log of high danger class intrusion behaviors in the database.





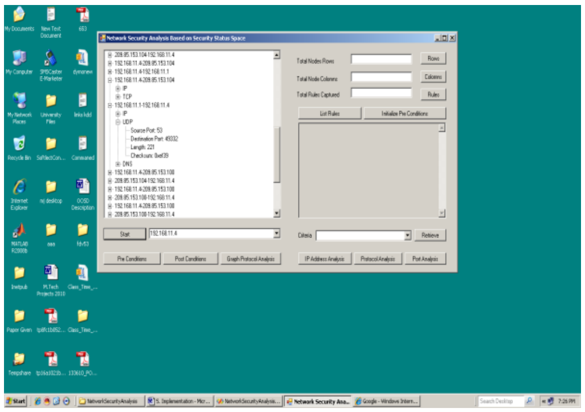
EXPERMENTATION&RESULTS:

The concept of this paper is implemented and different results are shown below


Network Security Analysis Based on Security Status Space

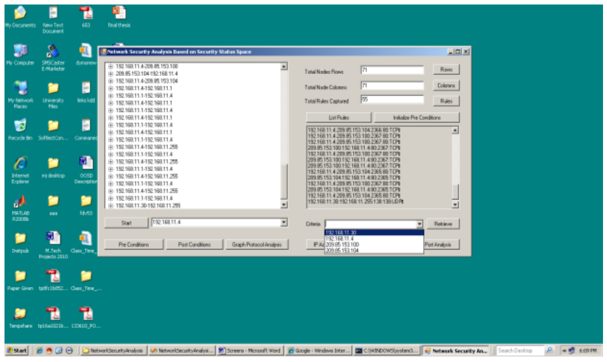
```
192.168.11.4-192.168.11.1
192.168.11.1-192.168.11.4
192.168.11.4-209.85.153.104
192.168.11.4-209.85.153.104
209.85.153.104-192.168.11.4
192.168.11.4-209.85.153.104
209.85.153.104-192.168.11.4
209.85.153.104-192.168.11.4
209.85.153.104-192.168.11.4
192.168.11.4-209.85.153.104
209.85.153.104-192.168.11.4
209.85.153.104-192.168.11.4
192.168.11.4-209.85.153.104
209.85.153.104-192.168.11.4
192.168.11.4-209.85.153.104
```

Stop 192.168.11.4


Network Security Analysis Based on Security Status Space

```
209.85.153.104-192.168.11.4
192.168.11.4-209.85.153.104
192.168.11.4-192.168.11.1
192.168.11.4-209.85.153.104
    IP
    TCP
192.168.11.1-192.168.11.4
    IP
    UDP
        Source Port: 53
        Destination Port: 49332
        Length: 221
        Checksum: 0xef39
    DNS
192.168.11.4-209.85.153.100
209.85.153.104-192.168.11.4
192.168.11.4-209.85.153.100
209.85.153.100-192.168.11.4
192.168.11.4-209.85.153.100
209.85.153.100-192.168.11.4
```

Start 192.168.11.4

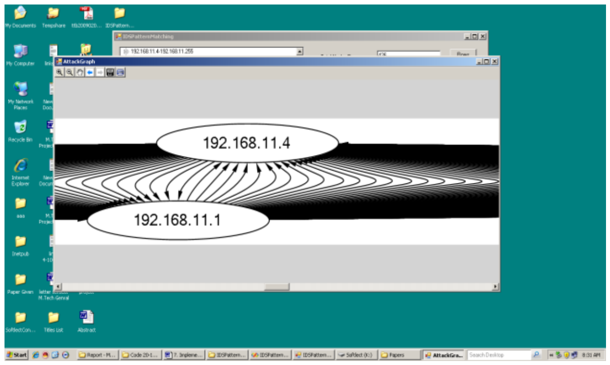

Network Security Analysis Based on Security Status Space

```
192.168.11.4-209.85.153.100
209.85.153.104-192.168.11.4
192.168.11.4-209.85.153.104
192.168.11.4-192.168.11.1
192.168.11.1-192.168.11.4
192.168.11.4-192.168.11.1
192.168.11.1-192.168.11.4
192.168.11.4-192.168.11.1
192.168.11.1-192.168.11.4
192.168.11.4-192.168.11.1
192.168.11.1-192.168.11.4
192.168.11.4-192.168.11.255
192.168.11.1-192.168.11.4
192.168.11.4-192.168.11.255
192.168.11.1-192.168.11.4
192.168.11.4-192.168.11.255
192.168.11.1-192.168.11.4
192.168.11.4-192.168.11.255
192.168.11.1-192.168.11.4
192.168.11.30-192.168.11.255
```

Total Nodes Rows: 71
Total Node Columns: 71
Total Rules Captured: 55

```
192.168.11.4:209.85.153.104:2366:80:TCP
192.168.11.4:209.85.153.100:2367:80:TCP
192.168.11.4:209.85.153.100:2367:80:TCP
209.85.153.100:192.168.11.4:80:2367:TCP
192.168.11.4:209.85.153.100:2367:80:TCP
209.85.153.100:192.168.11.4:80:2367:TCP
209.85.153.100:192.168.11.4:80:2367:TCP
192.168.11.4:209.85.153.104:2365:80:TCP
209.85.153.104:192.168.11.4:80:2365:TCP
192.168.11.4:209.85.153.100:2367:80:TCP
209.85.153.104:192.168.11.4:80:2365:TCP
192.168.11.4:209.85.153.104:2365:80:TCP
192.168.11.30:192.168.11.255:138:138:UDP
```

Start: 192.168.11.4
Criteria: 192.168.11.30 / 192.168.11.4 / 209.85.153.100 / 209.85.153.104



Performance Analysis:

The proposed paper is implemented in .Net technology on a Pentium-III PC with 20 GB hard-disk and 256 MB RAM. The propose paper's concepts shows efficient results and has been efficiently tested on different systems.

IV.Conclusion

The tools to generate attack graph based on security status space for network security analysis are designed and implemented, and the experiment indicates the method is usable and effective. Many related research should be done in the future, the results from network scan tools should be used in the tools. The generating algorithm should be optimized and the method to analyze attack graph should be further studied

Finally, on the basis of this algorithm, an intrusion detection system model based on pattern matching algorithm is put forward. In addition, a detailed analysis of packet capturing module, protocol processing module, feature pattern matching module, log record module and intrusion response module of intrusion detection system is given in this paper.

REFERENCES:

[1].ZHANG Hu, "Design of Intrusion Detection System Based on a New Pattern Matching Algorithm". College of Information Technology Anhui University Finance & Economics Bengbu, China

978-0-7695-3521-0/09 $25.00 © 2009 IEEE

DOI 10.1109/ICCET.2009.244.

[2].Wikipedia.Network Intrusion Detection System. http://en.wikipedia.org/wiki/Network_intrusion_detection_system.

[3].D.Gusfield,*Algorithms on Strings, Trees, and Sequences*, Cambridge University Press,

[4].G.Stephen,String Searching Algorithms, World Scientific, 1994.

[5].R.S.Boyer and J. S. Moore,"A fast string searching algorithm," Communications of the ACM, vol. 20, no. 10, pp. 762–772, Oct. 1977.

[6].D.Knuth, J. Morris, and V. Pratt, "Fast pattern matching in strings," SIAM Journal on Computing, vol. 6, no. 2, pp. 323–50, June 1977.

[7]. Z. Galil, "On improving the worst case running time of the Boyer-Moore string searching algorithm," *Communications of the ACM*, vol. 22, no. 9, pp. 505–8, 1979.

[8]. A. Apostolico and R. Giancarlo, "The Boyer-Moore-Galil string searching strategies revisited,"SIAM Journal on Computing, vol. 15, no. 1, pp. 98–105, Feb. 1986.

[9]. M. Crochemore, C. Hancart, and T. Lecroq, "A unifying look at the Apostolico-Giancarlo string matching algorithm," Journal of Discrete Algorithms, 2000.

[10]. Ronald L. Rivest, "On the worst-case behavior of string-searching algorithms," SIAM Journal on Computing, vol. 6, no. 4, pp. 669–674, Dec. 1977.

[11]. R.Ritchey and P.Ammann, "Using model checking to analyze network vulnerabilities", In Proceedings of the IEEE Symposium on Security and Privacy, MAY 2001, pp 156-165.

[12]. P.Ammann, D. Wijesekera and S. Kaushik, "Scalable, Graph-Based Network Vulnerability Analysis", In Proceedings of CCS 2002: 9th ACM Conference on Computer and Communications Security, Washington, DC, November 2002.

[13]. L.Swiler, C.Philips, D.Ellis, and S.Chakerian, "Compter-Attack Graph Generation Tool", In Proceeding of the DARPA Information Survivability Conference & Exposition II, Anaheim, California, June 2001.

[14]. Rodolphe Ortalo, Yves Deswarte and Affiliate, "Experimenting with Quantitative Evaluation Tools for Monitoring Operational Security", IEEE Transactions on Software Engineering, Vol.25, Sept./Oct. 1999, pp.633-650.

[15]. S. Templeton & K. Levitt, "A requires/provides model for computer attacks", In Proceedings of the New Security Paradigms Workshop, Cork, Ireland, September 2000.

[16].Ulf lindvist, Phillip Brentano and Doug Mansur, "IDS Motivation,architecture,and An Early Prototype",Computer Security
Laboratory, US Davis: 160-171.

[17].R. Bayer, Symmetric binary B-trees: Data structure and maintenance algorithms, Acta Information, 1972:34-56.

[18].Breslauer D, Efficient String Algorithmic [Ph.D. Thesis], Computer Science Department. Columbia University, NY: 231-255.

[19].Robert Graham, Protocol Analysis and Command Parsing vs. Pattern Matching in IntrusionDetectionSystem,
http://www.networkice.com

[20].Neil Desai,Increasing Performance in HighSpeed IDS,
http://www.linuxsecurity.com/resource
file/intrusion_detection/increasing_Performance_in
high Speed_ NIDS.pdf

[21].Moore,Strother J.The Boyer_Moore Fast String SearchingAlgiruthm. Texas University.http://www.uteax.edu/moore/bestideas/st
ring-Searching/index.html

[22]. Jai Sundar Balasubramaniyan, Jose Omar Garcia-Fernandez,DavidIsacoff, Eugene Spafford, Diego Zamboni, "An architecture for intrusion detection using autonomous agents", CERIAS Technical
Report 98/05, June 11. 1998:34-53.