Intrusion Detection Management System for eCommerce security

Jens Lichtenberg[1] and Jorge Marx Gómez[2]

[1] Ohio University, Athens, OH, USA,

[2] Otto-von-Guericke-Universität, Magdeburg, Germany,

Abstract. One of the main problems in eCommerce applications and all other systems handling confidential information in general, is the matter of security. This paper introduces the idea of an intrusion detection management system to support the security. Intrusion detection per se, is the act of detecting an unauthorized intrusion by a computer or a network from the inside or the outside of the affected system, making an intrusion the attempt to compromise or otherwise do harm to other network devices. Next to the normal intrusion detection system an Intrusion Management System applies different Intrusion Detection Systems to not only detect a threat but also analyze it and propose counter measures to avoid the compromise of the guarded system. For the treatment plan, depending on the analysis, a multitude of counter measures is identified and ranked. The counter measure identification is done using data mining techniques on a counter measure repository, the final ranking through sorting algorithms. Of the numerous data mining techniques applicable for diagnostic or analytic purposes the nearest neighbor and the correlation coefficient techniques have been implemented. A feasibility study has shown that an analyzer can match a problem against a solution repository and find the optimal treatment suggestions, applied with a ranking, in an acceptable short period of time. Future work will include the analysis of attack characteristics and goals, and the interaction between system manager, response planning and execution module and the attack analyzer. Furthermore the counter measure repository will be evaluated and updated.

1 Introduction

An ecommerce system can be regarded as a distributed real-time system. Being an assortment of different systems, some for interaction purposes, some for customer support and some only for backup purposes, the distributed ecommerce system has to manage the different resources to provide the best quality of service. The main research in distributed systems has focused recently on the development of powerful middleware architectures and strong allocation algorithm ([2],[3],[4],[5],[6],[7]). Most of the resource management architectures lack an integrated security component.

To address this problem, this paper builds upon the SECURE-RM idea developed by the CIDDS workgroup at Ohio University, described in chapter “Related Work”. The module proposed in this paper, enables the Secure-RM architecture to compare attack pattern with solution ideas and find the best solution for an impending attack.

Lichtenberg J. and Marx Gómez J. (2005).
Intrusion Detection Management System for eCommerce security.
In *Proceedings of the 5th International Workshop on Pattern Recognition in Information Systems*, pages 134-143
DOI: 10.5220/0002520101340143

The matter of security in distributed systems in general and ecommerce systems in particular is still a very serious problem. The integration of a solution identification module into a secure resource management environment, would provide the ecommerce platform with the normal agility gained through the resource allocations by the resource manager and also with another security layer. The identification and defeat of attacks on the ecommerce service provider side, helps him build trust and might even attract customers that are currently scared off because of popular security breeches (Postbank scandal in Germany, Late summer 2004).

2 Related Work

2.1 Secure-RM

Ecommerce systems, as much as other distributed real-time systems, have little tolerance for missing deadlines. If a distributed real-time system is attacked, the systems performance may decrease since it is utilized increasingly and anomalously. This could result to a system wide failure or malfunctioning. Since most of the mentioned resource management systems lack the security component, Ohio University's CIDDS research group proposed an architecture called SECURE-RM ([8]).

SECURE-RM, as shown in Figure 1, employs multiple Intrusion detection systems, gathering information about pending attacks. This information is analyzed by the Security Management, which together with the "normal" resource management grasps the impact of the intrusion and derives an action advice for the resource allocation enactor.

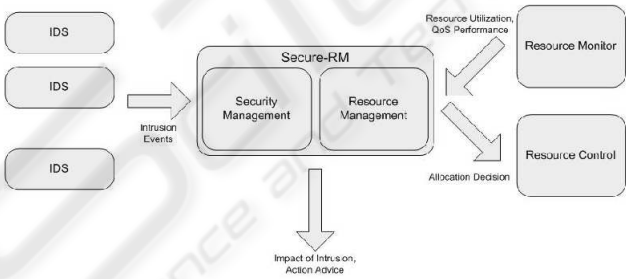

Fig. 1. SECURE-RM

2.2 Intrusion Detection Systems

An intrusion according to Webster is "the act of wrongfully entering upon, seizing, or taking possession of the property of another" ([9]). In computer science terms, intrusion detection is the act of detecting an unauthorized intrusion by a computer or a network from the inside or the outside of the affected system, making an intrusion the attempt

to compromise or otherwise do harm to other network devices. There are two kinds of Intrusion Detection Systems, the first analyzes log files from routers, firewalls, servers and other network devices to detect previously known attack signatures, which have been stored beforehand in the IDS's database ([1]). This system, called rule-based IDS, is fast and very secure in detecting older attacks, however it cannot protect a system against attacks that occur for the first time. The second type of IDS, the behavioral based systems, store a "normal" system behavior into a database and analyzing log files to determine an abnormal system state and certain behavior according to the state. This method allows for the detection of new unknown attacks but is susceptible to false positive and false negative alerts thus dampening the trust in the system. An IDS can, upon detection of an intrusion, issue alarms or alerts and take various kinds of automatic action, ranging from shutting down Internet links or specific servers to launching back-traces. The implementation of an Intrusion Detection System can be software based, hardware based or combined in preinstalled and preconfigured stand-alone IDS boxes. There are three basic types of Intrusion Detection Architectures implementing Intrusion Detection Systems: *Network based IDS*, *Host based IDS* and *Distributed IDS*.

In practice a combination of all three approaches can be implemented to allow for a higher state of security. An IDS can not only serve as an intrusion detector alone, but can also monitor database access, DNS functionality and it can also protect the e-mail servers and be applied as a company policy watch to provide the enterprise with the possibility to enforce copyright and electronic laws.

2.3 SNORT

An example for an intrusion detection system is SNORT ([1]), an efficient, stable, freeware, and open source implementation. It is considered a lightweight intrusion detection system, leaving a small footprint on the system with the possibility to run on various platforms, including Windows and Unix. Providing a real-time IP traffic analysis there are three ways to configure SNORT's core packet sniffer. In the first configuration, it is unaltered set up as a network sniffer, reading packages of the network and displaying them to the screen. The second way is a setup as a package logger, writing transmitted packages to the disks. The last way is a setup up as a network intrusion detection system, analyzing network traffic for matches against a user-definable rule-set, making SNORT a rule-based IDS with the ability to identify:

- CGI scans
- Buffer overflows
- SMB probes
- Unauthorized server services
- OS fingerprinting attemps
- Obfuscation (camouflaging the source code)

Identifying an intrusion, SNORT can record, ignore or alert a system administrator about the unhealthy traffic activities. The logging or alerting methods can be set up using Syslog, XML, plain text or WinPopUps.

The Snort Architecture as shown in Figure 2 consists of a Sniffer, a Preprocessor, a Detection Engine, and an alert and logging module, responsible for the output. The

Sniffer, as SNORT's core component, is responsible for network analysis and troubleshooting. It can analyze and benchmark the performance of the network backbone and can be used to eavesdrop for security risks like clear-text passwords. The raw data packages picked up by the Sniffer are then passed on to the Preprocessor, which formats them according to specific plug-ins, which also check for certain behavior types in the packages and then pass them on to the detection engine. In the detection engine the data is checked against a set of rules. If rules match the data in package, they are send to the alert and logging processor, where they can be passed to different outputs.

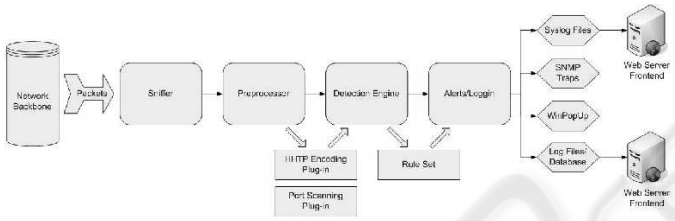

Fig. 2. SNORT Architecture

The open lines or current pitfalls of the SNORT architecture and IDS, in general, are mostly the missing of a number of packages, allowing the chance of intrusion attempts to go unnoticed, and the necessary handling of false positives and negatives, whereas the false negatives are much more dangerous then false negatives, since they can allow the infiltration of a system without notice of the IDS and system administrator.

2.4 INBOUNDS

INBOUNDS, the Integrated Network-Based Ohio University Network Detective Service, is network based real-time intrusion detection systems, developed at Ohio University as a project of Tjaden, Welch and Ostermann ([8]). It detects suspicious behavior, which marks it as a misuse or behavioral detecting IDS. INBOUNDS analyzes information gathered by package sniffers, like TCPTrace, and system host monitors, using those supplied by Ohio Universitys DeSiDeRaTa Resource Management architectures. The usage of system monitors and different package sniffers make INBOUNDS an integrated system, allowing its low overhead and scalability properties. The INBOUNDS architecture consists of five components and is designed to function in a large distributed system, where security and Quality of Service level must be insured dynamically. There are five architectural levels. The Data collection captures, filters, processes, and summarizes system information. It is assembled out of different package sniffers, hosts and network monitors. Each data collection module produces a data stream, which is stored in the historical and current data repository. The data visualization module and the final analysis module can request the data in the current data repository. The historical data repository summarizes the information from the collection module. The data

visualization module allows the user to view a visualization of the data streams in the current and historical repository, to help him making sense of the gathered information. The final Intrusion Detection service recognizes suspicious behavior and notifies the historical data repository and the visualization module to record and display the alert.

Currently the system is implemented at Ohio University for intrusion detection. The main research projects concerning INBOUNDS is the strengthening of the DeSiDeRaTa Inbounds link within the Secure-RM, to allow an insight into attack strategy ([10], [11]). Through the knowledge about the attack signatures a decision maker could be applied to diagnose the effect of applicable counter measures.

3 Modular Framework

This chapter proposes a solution identification module for occurring security problems. The basic structure of the module is shown in Figure 3. An attack is detected and identified by the intrusion detection systems. The attack profile is passed on to the solution identification module, which then matches the attack pattern to different solutions through the means of data mining. Possible solutions are ranked and a solution report is passed on to an enactor component, which has to conduct the necessary measures to secure the system. After the deployment of security measures, the enactor stores positive feedback in the solution repository. Solution algorithms can also be added, deleted or modified manually in the repository.

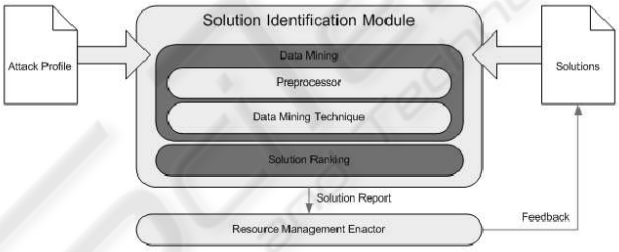

Fig. 3. Module Framework

The Solution Identification Module creates structures of the attack profile and the different solutions. After preprocessing the necessary information the two structures are handed down to the data mining technique. Setting solutions and attack profile in relation, the preliminary results are sorted according to their attack compatibility. The separation of the data mining technique and the actual module allows for fast updates of techniques according to specifications and experimental data and also ensures that the technique used is the best and latest choice for the problem at hand.

To allow the system to stay up to date, the used data mining techniques can be exchanged easily. This exchangeability of the techniques allows for a thorough testing and experimenting, which techniques provide the best performance under everyday usage.

In the context of this research two techniques have been implemented into the “Technique Library”. First, we have the Nearest Neighbor technique, which is an algorithm implementation of the Inductions Decision Tree paradigm. Correlation Coefficient technique, marking the second approach, is part of Multivariate Exploratory Data Analysis. With these two techniques implemented both major parts of Data Mining techniques have been covered, Induction being part of the New Analytic Techniques and Multivariate EDA as a part of Traditional Exploratory Data Analysis.

Built upon the SECURE-RM architectural approach, the module can be used as an Attack Analyzer for an Intrusion Detection Management System, as shown in Figure 6. While an Intrusion Detection System only monitors network activity to detect abnormal behavior or certain behavior signatures indicating an apparent attack on the system, an Intrusion Detection Management System (IDMS) applies multiple Intrusion Detectors. These Detectors send their gathered information, much like a Software or Hardware monitor in a resource management system, to a diagnosis module. The diagnosis module, called Attack Analyzer in an IDMS, detects certain attack behavior from the gathered data and proposes responses to occuring attacks. In tight cooperation with the “Response Planning and Executioning” part of the IDMS the Attack Analyzer provides the System Manager with knowledge about the nature and severity of the attack to allow him to discern between serious dangers to the network or harmless script attacks. The basic structure of an IDMS is shown in Figure 4.

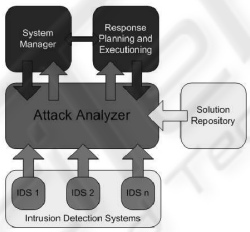

Fig. 4. Intrusion Detection Management System

While the proposed diagnosis module can generally be integrated into an IDMS, it can even be applied on IDS level. The structure of an IDS, like SNORT, can be easily expanded to include a Solution Repository in the means of the proposed module, as shown in Figure 5.

The proposed module can be trained to allow a variety of different responses. To counter an attack on a network structure, an Intrusion Detection Management System can reallocate tasks to other machines, it could shut down parts of the network or the network in total. Likewise the IDMS could invoke counterattacks to retrace the way to the attacker, identifying him to allow for prosecution.

A categorization for different attacks and the necessary countermeasures can be found in Figure 6. The actions to be taken are indicated by the grid color, green indicating threats that need no drastic actions but just the allocation of some tasks, yellow for

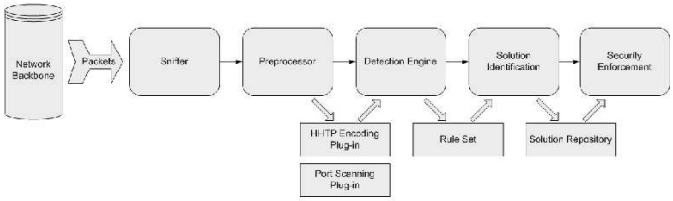

Fig. 5. SNORT Structure including the module

attacks that call for drastic resource allocation and the disabling of the most vulnerable parts of the network, red finally for threats that call for a network-wide shutdown to prevent damage to the system and/or company.

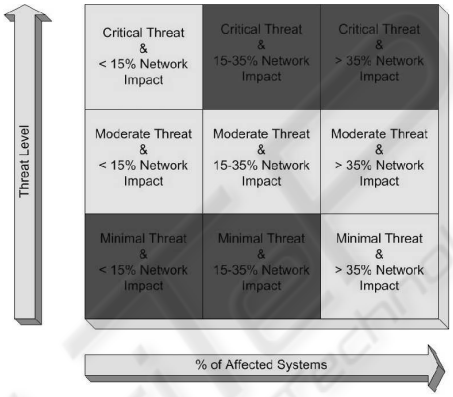

Fig. 6. Threat Taxonomy ([1], p.391)

This threat taxonomy can be implemented in the diagnosis module, using it as the information in the algorithm repository to provide the system manager with the knowledge about the severity of the threat (critical, moderate or minimal) and the proposed action to be taken (disable, log or alert). To allow for a categorization of countermeasures in the taxonomy from threat level information, a simple nearest neighbor technique can be applied to use the module for IDS and IDMS.

4 Experimental Framework

The Performance experiments were conducted on two different machines, running two different Linux distributions, each with two different versions. The first computer was an Ohio University machine running Fedora Core 1, a Red Hat spawn. This machine is part of the Center for Intelligence, Distributed and Dependable Systems Laboratories. It is setup with AMD XP 2000+ and 512 MB RAM. The second machine is part

of a private network and was run under Fedora Core 1 and 2, and Knoppix 3.3 and 3.4. Knoppix is a dynamic Linux distribution, which can be started from CD and is not copied to the hard disk. It is optimized for system recovery, but serves well as a comparison to the hard drive depending Fedora Cores. The second machine is a Pentium 3 1.2 GHz notebook with 256 MB RAM. No other tasks were run on the two machines during the experiments. In the first part of the experiments, the performance of the module is analyzed. To achieve a better insight into the composition of the performance, different setups are compared with changing machines and changing operating systems.

The performance of the two used data mining techniques was measured using the UNIX function times() ([12]). To be able to measure the execution times, each solution identification and ranking run was conducted 1000 times. The results in all these runs were always the same, only differing in their performance, which was added up. To achieve an even better average, these 1000 runs were measured 10 times. The final results for each measurement had to be added up once more and divided by 10 to get the final average for one algorithmical setup.

The performance of the techniques was measured through the CPU time used while executing instruction in the user space, a structure created by the times() function called UTIME. It is measured in CLK_TCK, which is defined in “limits.h” as 3 clock ticks per second. The results of the experiments are presented in Figures 7 to 11. They all show a logarithmically ascending nearest neighbor function. The correlation coefficient distribution tends to be almost linear with a step at around 290 algorithms. Only using Knoppix 3.3, the step is very distinct, while other distributions barely allow for time difference at all. Considering the Fedora Core 1 implementation on the Ohio University computer, it can be said that even here a slight logarithmic or power distribution of the nearest neighbor results can be found, while the correlation coefficient result settle around a linear distribution.

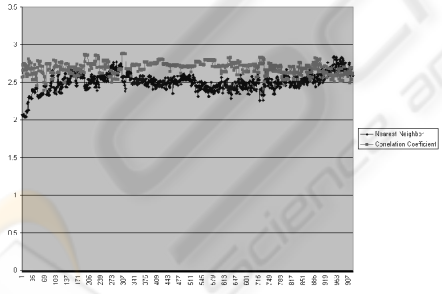

Fig. 7. UTIME Private Fedora Core 1

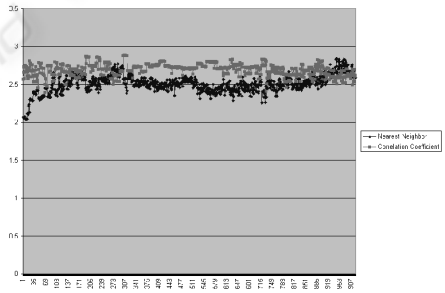

Fig. 8. UTIME Private Fedora Core 2

Analyzing the memory usage of the proposed module, the experiments using Fedora Core 2 show two distinct steps for both the nearest neighbor and the correlation coefficient technique. This means, that up to a certain number of solution algorithms, a constant number of kilobytes is sufficient to execute the identification. Once a certain

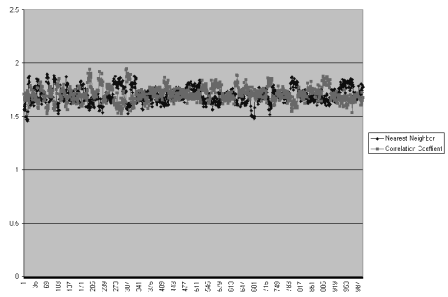

Fig. 9. UTIME OU Fedora Core 1

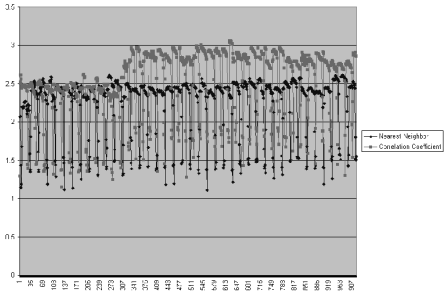

Fig. 10. UTIME Private Knoppix 3.3

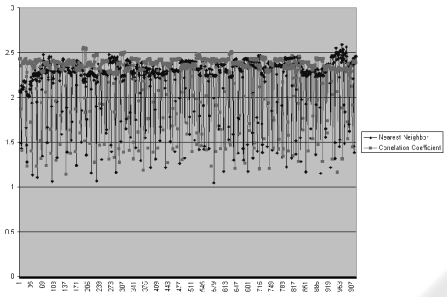

Fig. 11. UTIME Private Knoppix 3.4

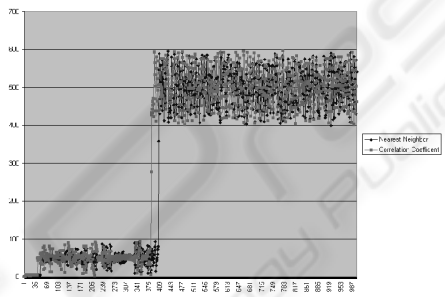

Fig. 12. Memory Usage Fedora Core 2

level is reached however, the system assigns another block of kilobytes to the process. This leads to the steps shown in Figure 12.

5 Conclusions

This paper shows that a lightweight Intrusion Detection Management Systems can be found using the architectural ideas of SECURE-RM. The module presents the system administrator with fast solution suggestion to an apparent problem, giving him a multitude of choices to respond to a threat. On an easier level, the module could be implemented into a normal intrusion detection system, to present the user, not only with a threat warning, but also with countermeasure proposals. The module, henceforth, not only affects resource management systems, but can also be applied to its security spawn and even purely security-oriented systems.

The future work concerning this project will include the implementation of more data mining techniques into the technique library, the implementation of system manager interfaces to allow for easier setup and management, and finally the integration of the module into a live resource management system to emphasize the SECURE-RM approach.

References

1. BEALE, J., FOSTER, J. C. and POSLUNS, J. (2003) Snort 2.0 Intrusion Detection
2. BRANDT, S. et al. (1998): A dynamic quality of service middleware agent for mediating application resource usage
3. FLEEMAN, D. et al. (2002): Quality-based Adaptive Resource Management Architecture (QUARMA): A CORBA Resource Management Service
4. HABAN, D. and SHIN, K. G. (1990): Applications of real-time monitoring for scheduling tasks with random execution times
5. LEE, C., SIEWIOREK, D. and RAJKUMAR, R. (1997): A Resource Allocation Model for QoS Management IEEE
6. LEE, C. and SIEWIOREK, D. (1998): An Approach for Quality of Service Management
7. MOERLAND, T. (2002) Resource Management and Scheduling http://www.liacs.nl/home/llexx/gc/rm.pdf
8. TJADEN, B. et al. (2000): INBOUNDS: The integrated, Network-Based Ohio University Network Detective Service Webster's Online Dictionary
9. WEBSTER (2001): Webster's Online Dictionary (2001)
10. WELCH, L. R. (1998): Specification, Modeling, Analysis of Dynamic Real-Time Systems
11. WELCH, L. R. and SHIRAZI, Behrooz A. (1998): Distributed, Scalable, Dependable Real-Time Systems: Middleware Services and Applications
12. WOLF, F. (2004) Performance Measurement 1, Class Notes