Asian Journal of Information Technology 6 (7): 801-805, 2007
ISSN: 1682-3915

Capturing and Analysing the Intruder Attacks Using a Platform Independent Honeypot Deployment

[1]V. Maheswari and [2]P.E. Sankaranarayanan
[1]Department of Master of Computer Applications
[1,2]Sathyabama University, Chennai, 119, India

Abstract: This study deals with the implementation of a Java based honeypot under a client server architecture model in which the client alerts the server whenever a new attack is encountered. The honeypot is developed in Java that makes it easier to be run under various platforms. Our honeypot architecture is designed as an alerting tool that sends the alert message to the server periodically. This helps the server to trap the attacks and analyse them and get the intrusion information about the hackers. The implementation also has a web based packet viewer interface, which makes the user to easily understand and view the various packets which comes through the network traffic and get the details about the possible attacks. The intrusion detection system has been designed as a GUI SNORT console, which makes easier to view the existing rules and information about the packets.

Key words: Intrusion, attacks, honeypot, hackers

INTRODUCTION

Honeypots: When more and more traditional services has become web based and as the Internet usage has also increased, the attacks and intrusion to web application system has also become more and more popular. Honeypot is a valuable security tool to view the intruder's activities. This can be used as a network deception tool that deceives the intruders and records their activities. It can be active or passive in nature (Spitzner, 2004). But the firewall and IPS are completely passive in nature. This makes honeypot more useful than other security tools. At the same time industry and academia also show growing interest in honeypot related research activities (Mcgrew *et al*., 2006). Our implementation is a low interaction Java based honeypot developed under client server architecture model. This can be run under various platforms like UNIX/LINUX and Windows. This implementation also includes a GUI SNORT console that interfaces the header information and packet information to update the existing rules and create new rules. Based on these an alerting message is sent to the server, when a new rule is generated and this is done periodically. Web based packet viewer gives the detailed analysis of packets and makes it easier to view the information on the packets. A separate interface for each type of protocol has been developed.

The honeypot was run under a client machine in the network and the incoming packets were captured by the client honeypot and after analyzing their header, the information was stored using MySql database. This provides a valuable resource for analyzing the attacks from vulnerable ports and unauthorized IP address using a web based packet viewer. Our implementation also includes a GUI SNORT console to set the rules and modify them based on the new attacks. The new rules as and when encountered are updated only manually in our implementation and not automatically.

Related work: Honeypot is a security device that is designed to lure malicious activity to it. Capturing such malicious activity allows for studying it to understand the operation and motivation of attackers and subsequently helps to better secure computers and network. A lot of existing honeypots are in usage ranging from low interaction honeypots to high interaction honeypots based on their interaction. There are many free and commercial honeypots each built of their own kind and type. These honeypots range from simple low-involvement to risky high-involvement. Honeyd is a low-to-medium-interaction honeypot system (Provos, 2004). Installed on a Unix system it listens on the Network Interface Card (NIC) for incoming ARP requests. Bait'n'Switch is a honeypot response mechanism that distracts the attacker from the valuable targets. Bait'n'Switch is realized as a snort in-line extension. Honeynet is a high interaction honeypot. Instead of simulating a single vulnerable host, a honeypot simulates

Corresponding Author: Maheswari, Plot no. 8, Adinaath Avenue, Santhoshapuram, Gowrivakkam (PO), Chennai, Tamilnadu, India, Pin code-601 302

a complete network and amplifies the detection and analysation possibilities. Now methods of data capture and data control proposed by Honeynet project shows greater flexibility and higher access control ability, which can be applied on both production honeypot and research honeypot The Deception Toolkit (DTK) is a set of free tools (mostly written in perl) designed by Fred Cohen. DTK uses deception to counter attacks.

Some of the pioneer work done under client honeypot are Honeyclient (Wang, 2006), Honey-monkey (Wang *et al*., 2006) and the client honeypot of the University of Washington (UW) (Moshchuk *et al*., 2006). These client honeypots focus on malicious webservers, which they interact with by driving a web browser on the honeypot system. Honeyclient detects successful attacks by monitoring changes to a list of files, directories and system configuration after the Honeyclient has interacted with a server. Honeymonkey also detects intrusions by monitoring changes to a list of executable files and registry entries, but Honeymonkey goes a step further by adding monitoring of the child processes to its repertoire to detect client side attacks. The UW client honeypot uses event triggers of file system activity, process creation, registry activity and browser crashes to identify client side attacks. All these client honeypots can be classified as high interaction client honeypots because they make use of a real browser within a real operating system environment and monitor the state of the entire system.

A similar work is done by Mai *et al*. (2004) in which they have implemented Honeyd as a Java code and have done the emulation and injection of packets using JNI and tcpdump file was used for intrusion detection. In our earlier work (Maheswari and Sankaranarayanan, 2007) we have developed a Java based honeypot that has been implemented to detect the network intruder's activity through a Multilayered Data control, Data Capture and Data duplication activites. As an extension of our earlier work the Java based honeypot has been used to detect the web-based threats and generate rules using an Intrusion Detection system and send the appropriate messages to server. This implementation is a rule based activity which uses SNORT for rule generation and SNORT has been used since it a freely available tool.

IMPLEMENTATION

The overall architecture of our implementation is given in Fig. 1. It includes a web-based interface to view the packets and also a GUI SNORT console has been designed to view the rules and the header and packet information about the network traffic.

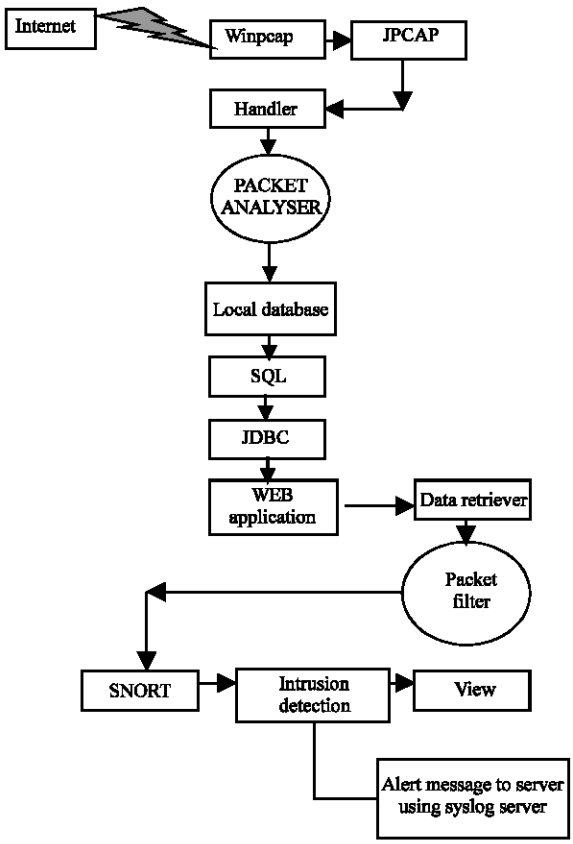

Fig. 1: Overall architectural flow of our implementation

The traffic that comes from outside traffic passes through our honeypot. The honeypot captures all the packets, filters them, analyse them and finally generate an alarm if an attack is encountered. All the packets were captured using the JPCAP/WINCAP (Charles, 2002). All the packets captured were involved in a packet processing thread, used for the different types of packets like TCP, UDP, HTTP FTP etc. The headers of these packets were analysed and were sent to the database module. The database is used as input the SNORT IDS to compare the packets with the existing rules and identify the possible attacks. The packets captured by the JPCAP were stored in MySql database using JDBC. MySql allows Unicode text format that enables the packet to be stored in binary raw format. Each type of packet headers was stored in a separate table for making the viewing and analyzing easier.

Packet viewer module: A JSP based packet viewer was incorporated in this project. This web application allows the packets to be viewed as html tables. Separate interfaces for each type of protocol have been developed. By clicking on appropriate hyperlinks the packets are viewed.

Intrusion detection system: The captured packets were analysed using a Intrusion detection system implemented using SNORT. The GUI SNORT console is developed for this module. From this interface the header information and the packet information can be verified and identified. The SNORT interface has an inbuilt provision to alert & log for particular type of intrusion or SNORT rule. The rules are made and saved manually into the database, which is internally integrated on each packet captured. On identifying the packet as an intrusion a continuous log will be made and an alert message is sent to the source system.

Alerting system: The alerting messages were saved in a Syslog file to be processed by the Syslog server. We have implemented this using a Kiwi Syslog daemon, which was used to filter the messages text and forward it to the server via Email (SMTP).

RESULTS AND DISCUSSION

Simulation environment: Internet traffic on 50 computers in a LAN is successfully monitored and analyzed which gives 99% accurate results. Packet Handler, analyzer, synchronization is tested for maximum performance using exceptions. The entire setup was tested for all types of packets from both unauthorized IP address and unauthorized ports. These packets were generated and were redirected to honeypot by the router and the packets of each type were analysed using the existing SNORT rules. An alarm message was sent out if any rule has been found. Later these rules were manually updated in the SNORT database.

Simulation results: The data which has been collected by the honeypot was analysed for all possible attacks, virus and worm attacks. SNORT was used to identify these attacks and the database, which was stored in the Mysql format, was given as input to the SNORT IDS system. The output generated by the SNORT console is given in Fig. 2 and 3. It shows how the rules and alerts can be defined using the user friendly screen. Figure 4 shows a sample of packets captured and their information.

Some of the packets which were captured by SNORT are from the ports like 80, 21, 443, 1434, 3127, 137, 139. Figure 3 shows the number of attacks that has been observed under the various ports. The unauthorized IP address was also recorded using Telnet service. The graph shows that there was more number of attacks on port 80 that is on the web services. This port is vulnerable towards attacks formed against any application using this communication port, such as Internet Explorer, Windows Media Player etc. From the log file, it shows the traces of spam mails. A worm attack, Mydoom worm was also captured in the port 3127. Another rare attack was on port number 443, which was a denial of service attack. The alert messages were generated by the SNORT and if it is any worm or virus attack a message was sent to the server using the Syslog server.

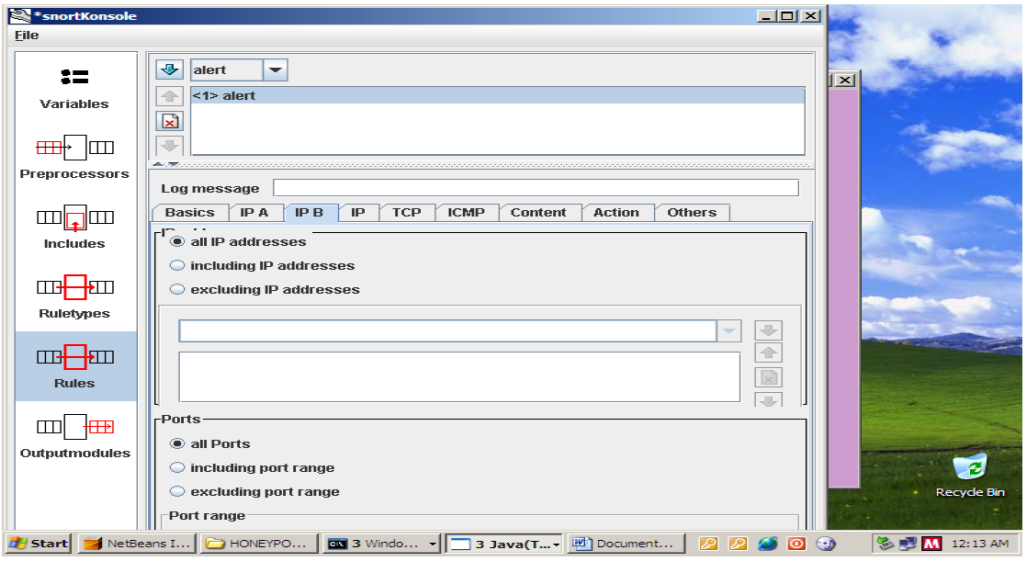

Fig. 2: SNORT console showing the RULES

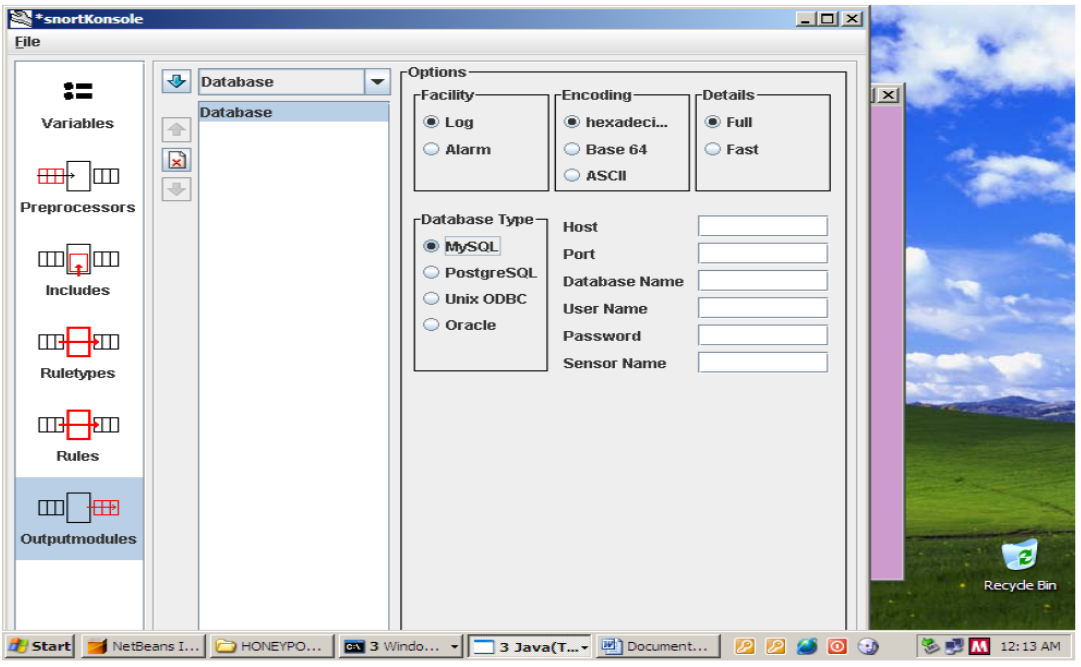

Fig. 3: SNORT console showing the Output module options

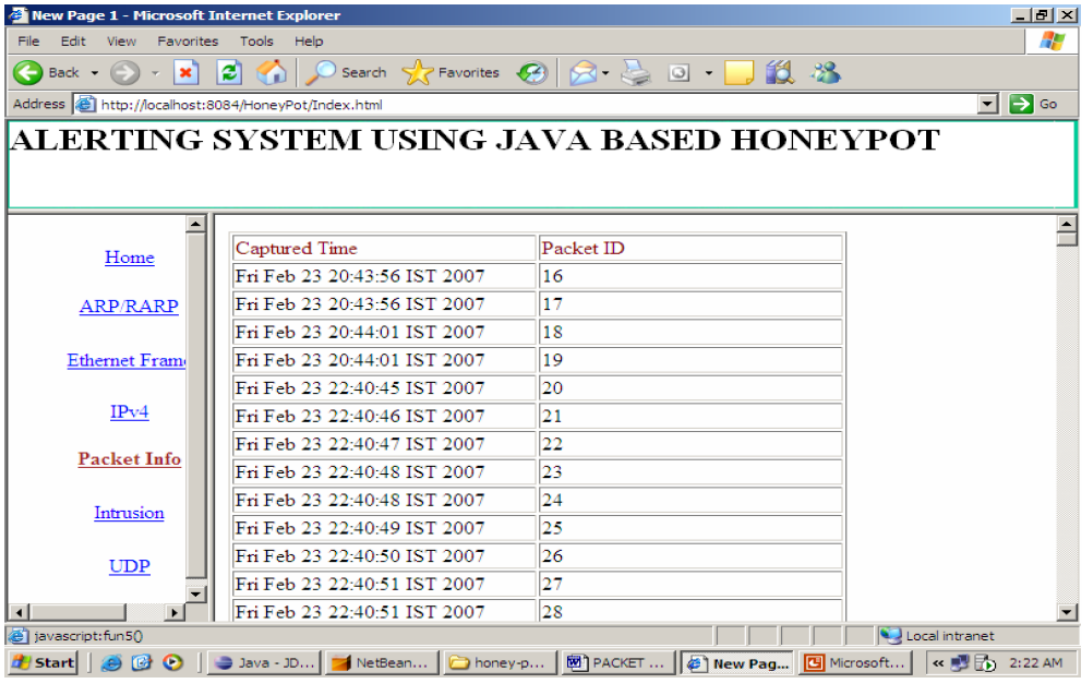

Fig. 4: Sample screen on Packet Capturing

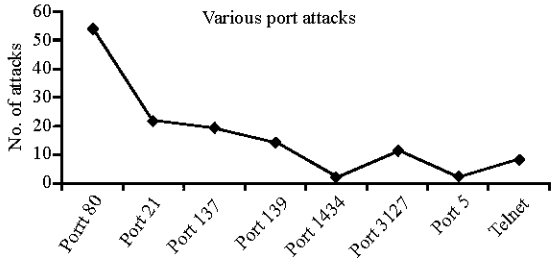

Fig. 5: Port attacks on various ports

CONCLUSION

In our implementation, we have developed a honeypot as a platform independent tool and it was used to capture and analyse the various attacks using the SNORT intrusion detection system. A web based packet viewer was also used to view the packets and a GUI based SNORT console was also used to analyse the packets.

FUTURE ENHANCEMENTS

In our implementation, we have not included the latest rules for virus worm detection. This implementation can be exclusively designed for worm detection, so that the prevention of worms can be made easier. The alerting message is sent to the server only periodically. This can be done based on the severity and the type of attack encountered and automatically. We have not emulated all the protocols and this can be extended to the operating system also.

REFERENCES

Charles, P., 2002. JPcap-Java Packet Capture Library. http://sourceforge.net/projects/jpcap

Mai, Y., R. Upadrashta and X. Su, 2004. J-Honeypot: A Java-Based Network Deception Tool with Monitoring and Intrusion Detection, itcc, International Conference on Information Technology: Coding and Computing (ITCC) 1: 804.

Maheswari, V. and P.E. Sankaranarayanan, 2007. Defeating hackers through a Java based honeypot deployment, Information Technology Journal, (under printing).

Mcgrew, R., B. Rayford and J.R. Vaughn, 2006. Experiences with honeypot systems: Development, deploymentand analysis. Proceedings of the 39th Hawaii International Conference on System Sciences.

Moshchuk, A., T. Bragin, S.D. Gribble and H.M. Levy, 2006. A Crawler-based Study of Spyware on the Web. In 13th Annual Network and Distributed System Secu-rity Symposium (San Diego).

Provos, N., 2004. A Virtual Honeypot Framework, 13th USENIX Security Symosium, San Diego, CA.

Spitzner, L., 2004. The honeynet project: Trapping the hackers, IEEE. Security and Privacy, pp: 15-23.

Wang, K., 2006. Honeyclient, Version 0.1.1. Available from http://www.honeyclient.org/

Wang, Y.M., D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen and S. King, 2006. Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites That Exploit Browser Vulnerabilities. In 13th Annual Network and Distributed System Security Symposium.