An Advanced Honeypot System for Efficient Capture and Analysis of Network Attack Traffic

Balaji Darapareddy#1, Vijayadeep Gummadi#2

[1] M.Tech (CSE),Gudlavalleru Engineering College, Gudlavalleru
[2] Associate Professor(CSE), Gudlavalleru Engineering College, Gudlavalleru.

ABSTRACT:

A Honeypot is an information system resource used to divert attackers and hackers away from critical resources as well as a tool to study an attacker's methods. One of the most widely used tools is honeyd for creating honeypots. The logs generated by honeyd can grow very large in size when there is heavy attack traffic in the system, thus consuming a lot of disk space. The huge log size poses difficulty when they are processed and analyzed by security analysts as they consume a lot of time and resources. We propose a system which addresses these issues. It has two important modules. The first module is to capture packets in the network ie either lan or web. The second module is a analyzer the captured packets in order to generate summarized captured packet information and graphs for the security administrators. This application also monitors packet information regarding web traffic. The experimental results show that the space required by log file reduces significantly and reports generated dynamically as per user needs.

I INTRODUCTION

By increasing the usage of the Internet and implementing commonly used tasks through it, the concept of distributed applications has been considerably grown. Currently firewall and Intrusion Detection Systems (IDS) have been practically developed to block variety of threats through incoming
port connections.

INTRUSION DETECTION SYSTEMS (IDS)

Intrusion detection is the process of monitoring computers or networks for unauthorized entrance or activity. IDS can also be used to monitor network traffic, thereby detecting if a system is being targeted by a network attack. There are two basic types of intrusion detection: host-based (HIDS) and network-based (NIDS). Each has a distinct approach to monitoring and securing data, and each has distinct advantages and disadvantages. Host-based IDSs examine data held on individual computers that serve as hosts; they are highly effective for detecting insider abuses. Examples of host-based IDS implementations include Windows NT/2000 Security Event Logs, and UNIX Syslog. On the other hand, Network based intrusion detection systems analyze data packets that travel over the actual network. These packets are examined and sometimes compared with empirical data to verify whether they are of malicious or benign nature [2]. **An** example of NIDS is Snort **[3],** which is an open source network intrusion detection system that performs real-time traffic analysis. It can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, and OS fingerprinting attempts. There are two basic techniques used to detect intruders: anomaly detection, misuse detection (signature detection). Anomaly Detection is designed to uncover abnormal patterns of behavior, the IDS establish a baseline of normal usage patterns, and anything that widely deviates from it is flagged as a possible intrusion [5]. Misuse Detection, commonly called signature detection, uses specifically known patterns of unauthorized behavior to predict and detect subsequent similar attempts. These specific patterns are called signatures. Therefore in case of Misuse Detection at the heart of IDS is the attack signature. The signatures can be generated through approaches like Network Grapping / Pattern Matching, Protocol Decode/Analysis, Heuristic and Honeypot. Current intrusion detection systems often work as misuse detectors, where the packets in the monitored network are compared against a repository of signatures that define characteristics of an intrusion. Successful matching causes alerts to be fired. The signature often consists of one or more specific binary patterns found in a given network packet. The signature can be described as a Boolean relation called rule [6]. An intrusion detection system is able to recognize an attack only when it knows a signature for this attack, and thus require continuous updates of their signature database. Also continuous research to analyze new attacks and find their signatures is a must.
Moreover, a slight change in the attack scenario may be enough to alter the attack signature and thus fool a signature filter. They are consequently vulnerable to polymorphic attacks and other evasion techniques which are expected to grow in the near future. At present, the creation of these signatures is a tedious process that requires detailed

knowledge of each software exploit that is to be captured and a large pool of ASCII-log data to analyze.

Honeypots

The honeypot has emerged as an effective tool for observing and understanding intruder's toolkits, tactics, and motivations. A honeypot suspects every packet transmitted to/from it, giving it the ability to collect highly concentrated and less noisy datasets for network attack analysis. "A honeypot is an information system resource whose alue lies in unauthorized or illicit use of that resource" [6]. Honeypot is an exciting technology with great potential for the field of network security. It can be understood as a resource used to divert attackers and hackers away from critical resources i.e. it is an observed trap. It can also be used to study an attacker's methods and tools. The value of a Honeypot lies in unauthorized and illicit use. Neither any authorized activity runs on these resources nor do they have any production value i.e. no legitimate activity is carried out. It provides a large amount of valuable information for analysis and can detect variety of attacks, working even within encrypted environment. It acts as a cherished observation and early warning tool but on the contrary it should be used with caution as it has risks associated with it.

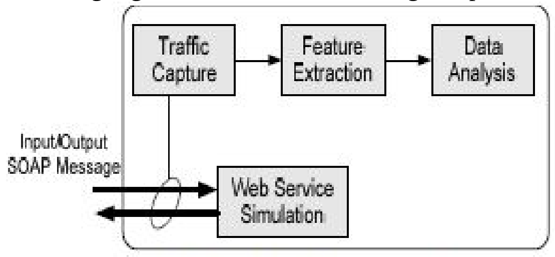
II BACKGROUND AND RELATED WORK

Network Deception using Honeypots

A honeypot is an excellent deception tool to use on a network. First, a honeypot allows us to present controlled information to our adversaries. Second, a honeypot allows us to collect information about our attackers. Finally, a honeypot can serve as a deterrent against future attacks. There are plenty of honeypot programs out there, from relatively free ones like Back-Officer Friendly [7], to commercial ones like Symantec Decoy Server [8]. As mentioned before, the Honeyd program created by Niels Provos is a low-interaction honeypot that emulates single or multiple hosts on the network with different operating system signatures and
services on a single machine. The current implementation of the Honeyd program is built using the C language and is made for BSD, Linux, or other variant of UNIX operating systems. It utilizes the OS fingerprints signature file from NMAP [6] and Xprobe2 rc1 [3] to spoof responses to OS scans. Furthermore, the Honeyd program is
capable of simulating many network environment variables such as packet delay, congestion, routing, and forwarding. The WS Honeypot is a high interaction honeypot. It provides real web services to ensure a real interaction with attackers. The services offered by the honeypot can be deployed by using two technologies, Axis or .Net. The administrator of the honeypot can customize his own web service, or he can simply use an automated tool integrated into the honeypot that can create from a WSDL (Web Services Description Language) file a real service that can be deployed in the honeypot. This tool offers flexibility in the behavior of the honeypot and in the choice of its features. Honeypots store collected information in log files. This
information is used to learn new techniques, tools and motivations of hackers to better protect the production systems against attacks. The main problem related to logging is the great amount of data that has to be analyzed by a human expert. Much of these collected data represent the normal behavior of the system and don't have any relation with attacks. So the human expert will be overwhelmed with a large amount of data in audit trails and may fail to notice severe threats. For this reason, we choose to use, in the WS Honeypot, machine learning techniques to analyze data and detect attacks in a semi-automatic way. These techniques help us to learn the normal behavior of activities in the honeypot, and consequently, detect any significant deviation from this normal behavior and provide it to the expert to decide whether it constitutes a true positive.

Existing Architecture

The role of the WS Honeypot is to simulate the behavior of a Web service. It incorporates automated tools to capture and analyze clients activities, especially those issued from attackers. The existing system architecture is shown in the following Figure. It consists of following components:



The WS Honeypot architecture

Traffic Capture

Traffic logging is an important task to collect and classify client activities. This component includes traffic capturing mechanisms and monitoring tools to intercept and parse requests and responses for Web services simulated on the WS Honeypot. The Web service requests are formatted in XML language and encapsulated in SOAP messages which use HTTP as the transport protocol. The content inspection of these messages is necessary to detect attacks.

Whenever a connection attempt is made by an attacker in unused IP space, honeyd which continuously monitors it acts being a victim. It interacts with the attacker and logs the malicious activities. To illustrate in case a telnet server is emulated then attacker's login and password along with other commands can certainly be logged. The emulated services are deterministic in his or her functioning and behave because we are part of a predetermined solution for documented actions. In the event the action is unknown then a mistkea message is made. Benefits associated with emulating various operating systems and services are to the point it assists the honeypot to blend together with the production network so we can learn about various attack methods used on various services. To illustrate with a Win2000 server an IIS webserver will be emulated with a Linux system an Apache webserver. Honeyd mimics legitimate services through direct manipulation of all the network stack of the designated OS. Some of these are TCP, UDP, and ICMP. Honeyd was developed this fashion to take care of popular network security scanning tools an example would be Nmap [8].

Each computer-system among the Honeyd virtual network emulates an OS “personality”, and that is an explicit match associated with an Nmap or Xprobe[13] prescribed OS “signature”. The signatures are offered inside an Nmap and Xprobe fingerprint file. The files enumerate the flag sequence of TCP/IP communication that would be used to distinguish different OS platforms. Therefore, when Nmap scans Honeyd it detects the personality signature and recognizes it, subsequently identifying the OS[6]. A number of features used in Honeyd are simulation of large network topologies, setting configurable network characteristics like latency, loss and bandwidth etc, integration of physical machines into network topology, support of multiple entry routers to serve multiple networks,tunneling for using a distributed networks. Time consuming and annoying work of search through logs generated by honeyd was simplified by a wide range of tools which were developed for creating summary of honeyd logs.
Honeysum [9] is a kind of name tool designed in Perl. It may generate summary of honeyd logs based upon Internet protocol addresses, protocols, ports etc. It generates top use of source and port; also it shows array of connections per hour. It is capable of generating summary both in text and also HTML showing outcomes in type of graphs. HoneyView [10] was another such tool. It is faster than honeysum while it possesses a php based web interface which provides better consumer experience.
Swamill [11] is yet another such software although it's not limited to only honeyd. It truly is universal log file analysis and reporting software. It might process log files generated by many security tools, sniffers, applications etc. The analyzers take some processing time to generate the reports. This can be time intensive when logs are of large size, and that is usually in the case of heavy attack scenarios like DDoS. The large size of the log also requires lot of space on the disk. Hence reducing this log sizes are another important issue. These issues are addressed by the proposed model.

Problem in existing honeypot:

- Difficulty in analyzing the log records due to large data.
- Need to configure in each and every host inside the network.
- Huge number.of log files posses difficulty when they are processed and analyzed by security analysts as they consume a lot of time and resources.
- Existing system doesn’t provide summarized information about each host based on protocols.
- Existing system provide static graphical representation.
- Packet information is stored in textual format which is not secured.
- Existing system is suffering from categorizing the normal and abnormal behavior of a system when Network environment is too complex.

III. PROPOSED FRAMEWORK

Below figure 1 represents overall architecture of the present system.First this sytem captures all packet and then each host packet information is stored in the database for future reference. After storing in the database the real time analyzer gives the real time packets information in the form of graphical representation dynamically.
Functional Modules
• Enabling NIC interface
• Installing Win Pcap library
• J-HoneyPot captures packets by the help of Win Pcap library.
• Java Native Interface is required to run our advanced J-HoneyPot system.
• Capturing the packets ,Processing and storing them in database
• Dynamic graphical representation with honeypot.

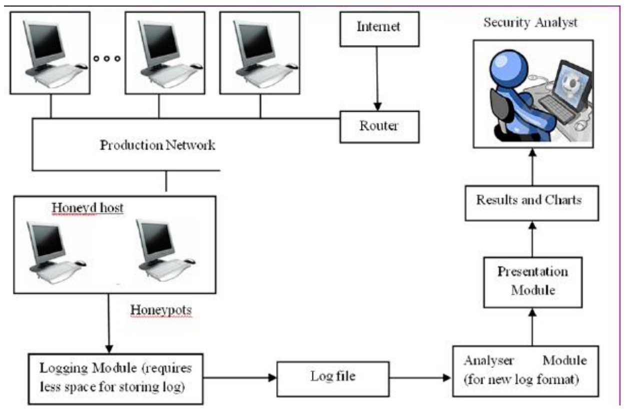


Fig 1: Proposed Architecture

Win Pcap library
• Win Pcap library is a powerful set of library files which are used to perform various tasks like
– scanning available network adaptors
– obtaining information about the network adaptors.
– capture network packets using one of the network interface cards of the computer, filter the captured packets, to obtain only the desired ones.
The implementers of the pcap API wrote it in C, so other languages such as Java, .NET languages, and scripting languages generally use Pcap library to interact with NIC cards. Therefore the WinPcap or LibPcap library is essential for running efficient J-HoneyPot application on Windows and Linux environments.

Processing and storing

The processing of the log is done by changing the format of the records slightly. A couple of new fields are added that help in doing this job. A number of records are considered at a time and they are processed to find whether they belong to a particular flow. The processing module waits for these numbers of records and then starts execution. By flow we mean that the packets have same source IP address, source port, destination IP address, destination port number, flags (if any). Suppose x records are found belonging to a particular flow. The time stamp of first record is the first field, the time in the time-stamp of the last record is added as the second field and number of records x comes in as the third field. Rest whole record is copied down. If a flow contains only one record then the original record is copied as it is. Thus, in the worst case also the log is not going to increase. From the knowledge of attack traffic and the tools that are used to attack it can be easily verified that it is very less probable case. Hence the system is going to decrease the log size in most of the cases without losing information.The new record format is shown in Fig .

2011-01-18-11:09:39.6092 11:09:40.0283 45 tcp(6) - 192.168.111.105 119 192.168.111.212 3515: 52 A [Windows]

Algorithm to filter packets

Step 1: open the interface
Step 2: Start capturing packets
• for each packet pack
a) set filter=‘ TCP or IP’
b) temp[]=capturesetfilter(filter)
c) if(temp[]==‘TCP’)
d) store pack dest port, seq, src port, syn to DB
else
e)store identifier(v4.0), dest port, src port, sync to DB.
Setp 3: Sort DB according to sequence number in the TCP table.
Step 4: Sort the DB according to IP addresses.
Step 5: End

Algorithm to capture n/w packets

Step 1:Get list of all network interfaces and store them in NetworkInterface[]
Step 2: Get each Network Interface name and its MAC addresses in the NetworkInterface[]
Step 3: Choose NetworkInterface to capture packets in promiscuous mode.
(In non-promiscuous mode, when a NIC receives a frame, it normally drops it unless the frame is addressed to that NIC's MAC address or is a broadcast or multicast frame, thus in Promiscuous mode allowing the computer to read frames intended for other machines or network devices)
Step 4: Set no.of Packets to capture. (infinate -1)
Step 5: Print the packets in the console.
Step 6: End

Presentation Module

The files generated above were given as input to the Gnuplot tool. The graphs thus obtained can be used by the security analyst to strengthen the security of the production network. The pictorial representation is much easier to comprehend and have been used since a very long time by other analyzers as well. At this point it can be pointed out that the already existing analyzers can also process the log file if the code is changed slightly to incorporate the change in format of the log records.Info-packet The Info-packet is a standardised representation of packet header fields of any encapsulating protocol. For any protocol, an Info-packet contains fields that are common to every protocol and also fields that are protocol specific. Common fields in an Info-packet include the packet header size, the packet size (data + packet header). Protocol specific fields for IP, for example, include source and destination IP addresses, while for TCP, for example, they include the source and destination ports, see table . Info-packets mainly contain integers and strings. These data types are easier to manipulate and so making the task of packet processing lighter
.

Protocol:	1P
Total length.:	1500
Encap. Protocol:	17
Version:	4
Data length.:	1480
Time To Live:	255
IP Source:	141.35.14.26
IP Destination:	141.35.14.31
Header length:	20

Table 1: Some Contents of an IP Info-packet

This system takes the network data packet capture, and then carry on the analysis and the statistics. In the capture, it also can be set the necessary capture filter to reduce the system burden according to demand. The captured data can be promptly preserved for future use. Figure illustrates the Program flow diagram.

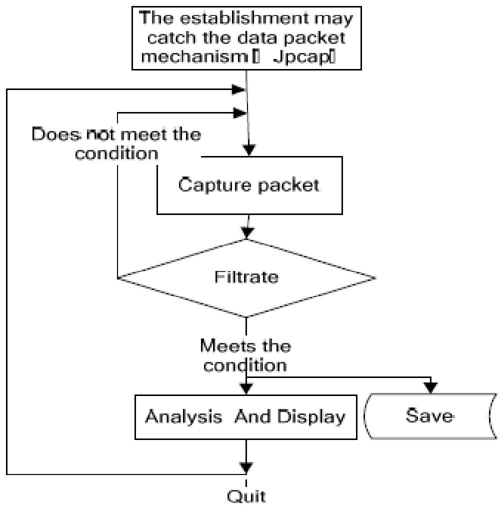

The establishment may catch the data packet mechanism Jpcap
Does not meet the condition
Capture packet
Filtrate
Meets the condition
Analysis And Display
Save
Quit

Fig 2: Packet capturing in jpcap

Capturing, preservation and analysis of data packets is the key to program. Using HONEYPOT for capturing and analysis data packet general need following process: Capturing, preservation and analysis of data packets is the key to program. Using HONEYPOT for capturing and analysis data packet general need following process:
(1) Creates an executive interface. When a packet is captured, the method of data packets analysis in PacketReceiver interface defined is invoked to analyze it by HONEYPOTCaptor.processPacket (). public class HONEYPOTTip implements PacketReceiver
(2) Searches device, returns to a string to represent Adapter NetworkInterface[] devices = HONEYPOTCaptor. getDeviceList();
(3) Select monitoring device NetworkInterface deviceName = devices[0];
(4) Turns on the adapter, if calls successfully, returns to a type of HONEYPOTCaptor object. It realizes through HONEYPOTCaptor.openDevice() method. The method needs four parameters: the device is to be opened, the largest number of bytes read from the device once, the device will be set to indicate whether the promiscuous mode Boolean

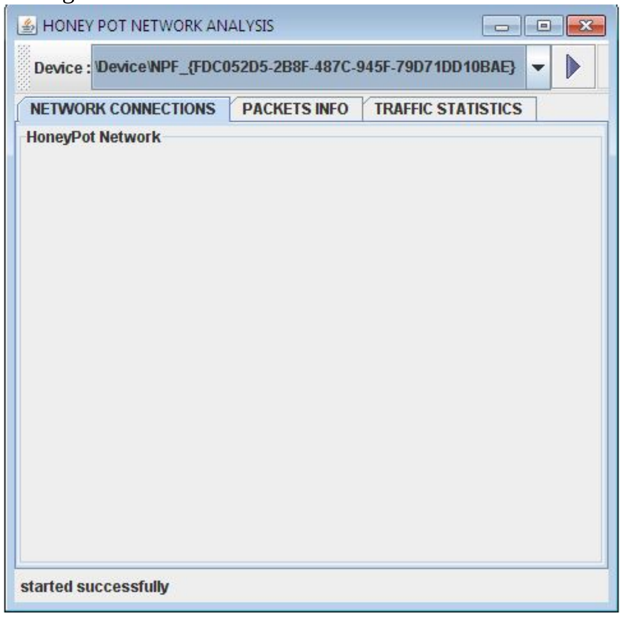
(true, false) value and the subsequent call processPacket () method to use to the value of overtime. HONEYPOTCaptor HONEYPOT = HONEYPOTCaptor.openDevice (deviceName, 2000, true, 20);

(5) Starts to capture data packets. Call processPacket () or loopPacket () started listening. These two modes are with two parameters: The capture most large package number may be - 1 (note there is no restriction), an instance of the class carries out PacketReceiver. HONEYPOT. processPacket (-1, new HONEYPOTTip());

(6) Stops the capture, closes the device HONEYPOT.close();

IV. EXPERIMENTAL RESULTS

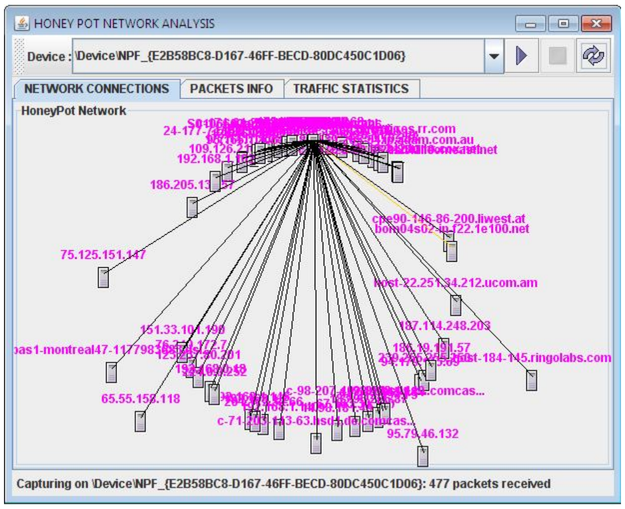
Proposed program is used to develop progam Honeypot for data packet capture analysis. Choose the sub-menu "start capture" under "capture data packets" menu , pop-up dialog box of the choice device and the establishment capture package option, in this selected by the device for the capture and set network card promiscuous mode, and determine the length and the filter information of the data packet captured, then start the data packet capture. Figure demonstrates an Ethernet data frame captured. Source MAC address is 00:02:3f:02:3b:ed, Destination MAC address is 00: 11: 11 :02:59:e8, it means that the frame is sent from host A to host B. Version of 4 means that the version of the IP protocol is IPv4. Header length of 20, means the header length of degree is 20 bytes. Total length of 1500 indicates that the IP datagram (or fragment) total length is 1500 bytes. Identification of Oxebb2(60338) indicates that the IP datagram (or fragment) identification is 60338. Flag segment as Flags is a three-bit field. The first bit is reserved.



Device : \Device\NPF_{FDC052D5-2B8F-487C-945F-79D71DD10BAE}

started successfully

Fig 3: Selecting adaptor for packets capture



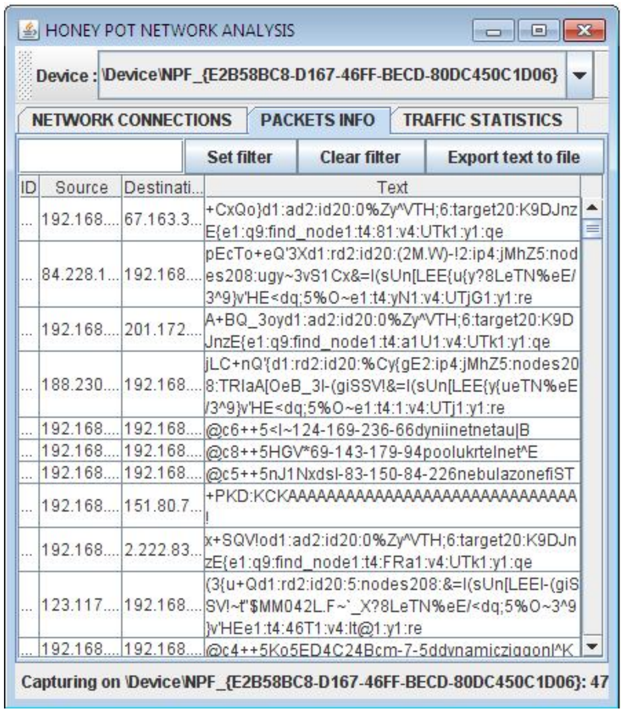
Device : \Device\NPF_{E2B58BC8-D167-46FF-BECD-80DC450C1D06}

Capturing on \Device\NPF_{E2B58BC8-D167-46FF-BECD-80DC450C1D06}: 477 packets received

Fig 4: Proposed Honeypot capturing network.

Ethernet Frame Captured

The second bit is called the do not fragment bit, its value is 0 means the datagram can be fragmented if necessary. The third bit is called the more fragment bit, its value is 1 means the datagram is not the last fragment, there are more fragments after this one. Fragment offset of 1480 shows the relative position of this fragment with respect to the whole datagram is 1480 bytes, that is the former fragment contains the first 1480 bytes data of the higher protocol; Time to live of 64 means the IP datagram's lifetime is 64 hops; Protocol of ICMP(OxOl) indicates that the higher level protocol that uses the services of the IP layer is ICMP, this field specifies the final destination protocol is to which the IP datagram should be delivered. Header checksum is the error detection method used by most TCP/IP protocols, its value is Oxe746. Source of 192.168.0.10 and Destination of .168.0.20 indicates that the IP datagram(or fragment) is sent from host A to host B. Data of 1480 indicates that the fragment carries 1480 bytes data of the higher layer. Test to ascertain discrepancies between the captured results and IP fragmentation principle again, as ICMP carries 3000 bytes, coupled with 8 bytes of its header, the total bytes is 3008, with the MTU of data link layer protocol in Ethernet is 1500, then get rid of 20 bytes IP header, each fragment can only get 1480 bytes, the offset of the data in the original datagram measured in units of eight bytes. so 3008 bytes are divided into three parts: 1480, 1480, 48, then the length of the three datagram fragment should be 1500, 1500, 68 after plusing 20 bytes of the IP header, that is consistent with the results of the captured data completely.



HONEY POT NETWORK ANALYSIS

Device : \Device\NPF_{E2B58BC8-D167-46FF-BECD-80DC450C1D06}

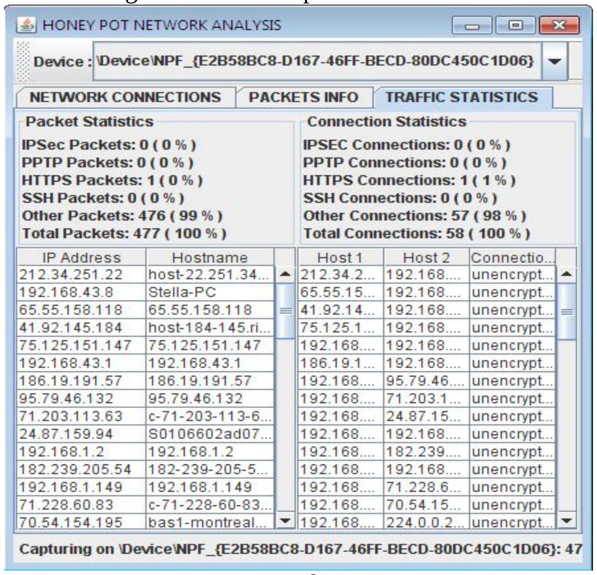
NETWORK CONNECTIONS | PACKETS INFO | TRAFFIC STATISTICS

Set filter | Clear filter | Export text to file

ID	Source	Destinati...	Text
...	192.168....	67.163.3...	+CxQo}d1:ad2:id20:0%Zy^VTH;6:target20:K9DJnzE{e1:q9:find_node1:t4:81:v4:UTk1:y1:qe
...	84.228.1...	192.168....	pEcTo+eQ'3Xd1:rd2:id20:(2M.W)-!2:ip4:jMhZ5:nodes208:ugy~3vS1Cx&=l(sUn[LEE{u{y?8LeTN%eE/3^9}v'HE<dq;5%O~e1:t4:yN1:v4:UTjG1:y1:re
...	192.168....	201.172....	A+BQ_3oyd1:ad2:id20:0%Zy^VTH;6:target20:K9DJnzE{e1:q9:find_node1:t4:a1U1:v4:UTk1:y1:qe
...	188.230....	192.168....	jLC+nQ'{d1:rd2:id20:%Cy{gE2:ip4:jMhZ5:nodes208:TRIaA[OeB_3l-(giSSV!&=l(sUn[LEE{y{ueTN%eE/3^9}v'HE<dq;5%O~e1:t4:1:v4:UTj1:y1:re
...	192.168....	192.168....	@c6++5<l~124-169-236-66dyniinetnetau|B
...	192.168....	192.168....	@c8++5HGV*69-143-179-94poolukrtelnet^E
...	192.168....	192.168....	@c5++5nJ1Nxdsl-83-150-84-226nebulazonefiST
...	192.168....	151.80.7...	+PKD:KCKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA!
...	192.168....	2.222.83...	x+SQV!od1:ad2:id20:0%Zy^VTH;6:target20:K9DJnzE{e1:q9:find_node1:t4:FRa1:v4:UTk1:y1:qe
...	123.117....	192.168....	(3{u+Qd1:rd2:id20:5:nodes208:&=l(sUn[LEEl-(giSSV!~t"$MM042L.F~`_X?8LeTN%eE/<dq;5%O~3^9}v'HEe1:t4:46T1:v4:lt@1:y1:re
...	192.168....	192.168....	@c4++5Ko5ED4C24Bcm-7-5ddynamicziggonl^K

Capturing on \Device\NPF_{E2B58BC8-D167-46FF-BECD-80DC450C1D06}: 47

Fig 5: Summarized packet information



HONEY POT NETWORK ANALYSIS

Device : \Device\NPF_{E2B58BC8-D167-46FF-BECD-80DC450C1D06}

NETWORK CONNECTIONS | PACKETS INFO | TRAFFIC STATISTICS

Packet Statistics

IPSec Packets: 0 (0 %)
PPTP Packets: 0 (0 %)
HTTPS Packets: 1 (0 %)
SSH Packets: 0 (0 %)
Other Packets: 476 (99 %)
Total Packets: 477 (100 %)

Connection Statistics

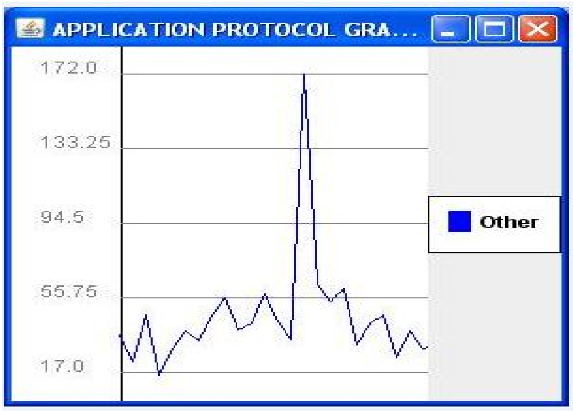
IPSEC Connections: 0 (0 %)
PPTP Connections: 0 (0 %)
HTTPS Connections: 1 (1 %)
SSH Connections: 0 (0 %)
Other Connections: 57 (98 %)
Total Connections: 58 (100 %)

IP Address	Hostname
212.34.251.22	host-22.251.34...
192.168.43.8	Stella-PC
65.55.158.118	65.55.158.118
41.92.145.184	host-184-145.ri...
75.125.151.147	75.125.151.147
192.168.43.1	192.168.43.1
186.19.191.57	186.19.191.57
95.79.46.132	95.79.46.132
71.203.113.63	c-71-203-113-6...
24.87.159.94	S0106602ad07...
192.168.1.2	192.168.1.2
182.239.205.54	182-239-205-5...
192.168.1.149	192.168.1.149
71.228.60.83	c-71-228-60-83...
70.54.154.195	bas1-montreal...

Host 1	Host 2	Connectio...
212.34.2...	192.168....	unencrypt...
65.55.15...	192.168....	unencrypt...
41.92.14...	192.168....	unencrypt...
75.125.1...	192.168....	unencrypt...
192.168....	192.168....	unencrypt...
186.19.1...	192.168....	unencrypt...
192.168....	95.79.46....	unencrypt...
192.168....	71.203.1...	unencrypt...
192.168....	24.87.15...	unencrypt...
192.168....	192.168....	unencrypt...
192.168....	182.239....	unencrypt...
192.168....	192.168....	unencrypt...
192.168....	71.228.6...	unencrypt...
192.168....	70.54.15...	unencrypt...
192.168....	224.0.0.2...	unencrypt...

Capturing on \Device\NPF_{E2B58BC8-D167-46FF-BECD-80DC450C1D06}: 47

Fig 6: LAN packets statistics



Our tests show that the proposed system can manage to improve the detection accuracy in comparison with the conventional honeyd and IDS in the following figure. As it is clear, in the proposed system the objective is to recheck the attacking packets in order to avoid the false positives while both honeyd and IDS have their own false positive ratio which is definitely greater than what we gained as a cooperation of these two systems.

V. CONCLUSION AND FUTURE WORK

Proposed packet HONEYPOT is effectively enhanced by corporating features like making the packet HONEYPOT program platform independent, filtering the packets using filter table, filtering the suspect content from the network traffic and gather and report network statistics. A packet HONEYPOT is not just only for an admin's tool. It can be used for network traffic monitoring, traffic analysis, troubleshooting and other useful purposes.In future this honeypot can be embedded in real time websites so that it gives effective detection rates for packet attacks and spoofing types of attacks. In future this honeypot is extended to find web application vulnerabilities for electronic applications.

REFERENCES:

[1] Provos, N., Honeyd - Network Rhapsody for You. 2002-2003, Center for Information Technology Integration - Computer Science Department of University of Michigan. http://www.citi.umich.edu/u/provos/hon eyd/

[2] Roesch, M., Snort - The Open Source Network Intrusion Detection System. 2003. http://www.snort.org/

[3] Song, D., libdnet. 2003. http://libdnet.sourceforge.net/

[4] Spitzner, L., Honeypots: Tracking Hackers. 2002: Addison-Wesley Pub Co. 480.

[5] Spitzner, L., Definition and value of Honeypots, in Tracking Hackers. 2003. http://www.trackinghackers. com/papers/honeypots.html

[6] Heberlein, L.T., G. Dim, K. Levilt, B. Mukhejee, J. Wood, and D. Wolber, I' A network security monitor,'' Proc., 1990 Symposium on Research in Security and Privacy, pp. 296-304, Oakland, CA, May 1990

[7] Staniford-Chen S., S. Cheung, R Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle, " GPICG- A Graph-Based Intrusion Detection System for Large Networks," The 19th National Information Systems Security Conference

[8I Anton Chuvakin, "Honeynets: High Value Security Data": Analysis of real attacks launched at a honeypot, Network Security, vol. 2003, Issue 8, pp. 11-15, August 2003.

[9] L. Spitzner, "Honeytokens: The Other Honeypot.," in Internet: http://www.Securityfocus. com/infocus/1713, 2003.

[10] Honeyd, http://www.honeyd.org/, 2008.