Honeypot based Secure Network System

Yogendra Kumar Jain
Head of the Department
Computer Science & Engineering
Samrat Ashok Technological Institute Vidisha, M.P., India

Surabhi Singh
Research Scholar
Computer Science & Engineering Department
Samrat Ashok Technological Institute Vidisha, M.P., India

Abstract-
A honeypot is a non-production system, design to interact with cyber-attackers to collect intelligence on attack techniques and behaviors. There has been great amount of work done in the field of network intrusion detection over the past three decades. With networks getting faster and with the increasing dependence on the Internet both at the personal and commercial level, intrusion detection becomes a challenging process. The challenge here is not only to be able to actively monitor large numbers of systems, but also to be able to react quickly to different events. Before deploying a honeypot it is advisable to have a clear idea of what the honeypot should and should not do. There should be clear understanding of the operating systems to be used and services (like a web server, ftp server etc) a honeypot will run. The risks involved should be taken into consideration and methods to tackle or reduce these risks should be understood. It is also advisable to have a plan on what to do should the honeypot be compromised. In case of production honeypots, a honeypot policy addressing security issues should be documented. Any legal issues with respect to the honeypots or their functioning should also be taken into consideration. In this paper we explain the relatively new concept of "honeypot." Honeypots are a computer specifically designed to help learn the motives, skills and techniques of the hacker community and also describes in depth the concepts of honeypots and their contribution to the field of network security. The paper then proposes and designs an intrusion detection tool based on some of the existing intrusion detection techniques and the concept of honeypots.

Keywords- Secure Network, Honeypot, Intrusion Detection.

I. INTRODUCTION

An intruder can be defined as somebody attempting to break into an existing computer. This identity is popularly termed as a hacker, blackhat or cracker. The number of computers connected to a network and the Internet is increasing with every day. When combined with the increase in networking speed has made intrusion detection a challenging process. System administrators now days have to deal with larger number of systems connected to the networks that provide a variety of services. The challenge here is not only to be able to actively monitor all the systems but also to be able to react quickly to different events. Traditionally intrusion detection involved a defensive approach where systems were either dedicated computers like firewalls or host based detection systems aimed at detecting attacks or preventing them. These systems existed as a part of the commercial/in-use networks and used techniques like pattern matching or anomaly detection. Another type of security systems are system integrity checkers, which are, typically host based. The problem that these systems face is that they are running on computers, which are in use on a daily basis. These systems usually have to deal with large number of connections and data transfers which results in huge log files and also makes it difficult to differentiate between normal traffic and intrusion attempts accurately. Many of these systems are also known to generate many false positives or in some cases false negatives. Moreover these systems provide very little insight to the tools and methods employed by the blackhat community [1] and [3].
An outsider attack is an attack from a person who is not a member of the organization. Usually the intruder is a hacker whose intensions are to cause harm or mischief. We can classify this intruder into two types, one who has something to gain by the intrusion and the other a curious person trying to probe the security of the system. The first type is popularly termed as a "cracker". Crackers attack web-sites or database servers in an attempt to gain critical information such as credit card or social security information. Some try to deface government web-sites or deny normal service and may be backed by political motive. The second type is the "hacker" who can be

further broken down into two types: - an extremely intelligent computer knowledgeable person or a “script kiddie”. An intelligent hacker is one who studies protocols and algorithms and tries to detect vulnerabilities in them. There is nothing malicious about this type although his curiosity and intent is often criticized by many security analysts as irresponsible behavior. The “script kiddie” is the intruder with limited skills but the one who uses automated computer programs or who exploits code downloaded from the Internet. Needless to say the “script kiddie is the most common type of intruder. This “script kiddie” is one of the reasons why “security by means of obscurity” will not work. If you think you are hidden from the world since you are not advertising any services and you think no one would be interested in you then you are wrong. The primary aim of this intruder is to compromise as many systems as possible. He is aided by the easy-to-use tools that scan a range of IP addresses looking for a vulnerable computer. So it’s just a matter of time that any system on the Internet will get probed and, if found vulnerable, then “hacked into”. Networks face even a bigger threat, since all the intruder has to do is compromise one insignificant system in the network and use it to attack the more important systems. A honeypot is a program, machine, or system put on a network as bait for attackers. The idea is to deceive the attacker by making the honeypot seem like a legitimate system. Honeypots are typically virtual machines that emulate real machines by feigning running services and open ports, services which one might find on a typical machine on a network. These running services are meant to attract the attention of attackers so that they spend valuable time and resources will be used to try to exploit the machine while the attacker is being monitored and recorded by the honeypot. The idea behind these systems is to provide systems or services that deceive the intruder. Such systems help in learning the methods that intruders use and they also can be viewed as a decoy to distract hackers from the real systems and services. Honeypots can be classified as deception systems. By definition a honeypot is *“a security resource whose value lies in being probed, attacked or compromised”*. Honeypots can be used as tools to gather information which can be used to enforce and strengthen existing intrusion detection tools or network firewalls. Honeypots should not be viewed as a solution to network security; they should be seen as an aid to it. We look at the objectives behind the deployment of honeypots, their uses and security and legal issues involved with it. We also look at the setup of a network of honeypots and present some analysis based on the information gathered from it. We summarize by presenting a survey of existing honeypot technologies. In this thesis we look at the new concept of honeypots and their application in intrusion detection systems. As a part of the thesis project a network of honeypots was designed and implemented. The honeypots were kept online for a period of time and any network communication or events related to it was recorded and analyzed [2] and [4].

II. BACKGROUND

A honeypot is a program, machine, or system put on a network as bait for attackers. The idea is to deceive the attacker by making the honeypot seem like a legitimate system. Honeypots are typically virtual machines that emulate real machines by feigning running services and open ports, services which one might find on a typical machine on a network. Currently there are two types of honeypots classified according to their use-:

Research Honeypots:
As the name suggests these honeypots are deployed and used by researchers or curious individuals. These are used to gain knowledge about the methods used by the blackhat community. They help security researchers learn more about attack methods and help in designing better security tools. They can also help us detect new attack methods or bugs in existing protocols or software. They can also be used to strengthen or verify existing intrusion detection systems. They can provide valuable data which can be used to perform forensic or statistical analysis.

Production Honeypots:
These honeypots are deployed by organizations as a part of their security infrastructure. These add value to the security measures of an organization. These honeypots can be used to refine an organization’s security policies and validate its intrusion detection systems. Production honeypots can provide warnings ahead of an actual attack. For example, lots of HTTP scans detected by honeypot is an indicator that a new http exploit might be in the wild. Normally commercial servers have to deal with large amounts of traffic and it is not always possible for intrusion detection systems to detect all suspicious activity. Honeypots can function as early warning systems and provide hints and directions to security administrators on what to lookout for.
The real value of a honeypot lies in it being probed, scanned and even compromised, So it should be made accessible to computers on the Internet or at least as accessible as other computers on the network. As far as possible the system should behave as a normal system on the Internet and should not show any signs of it being monitored or of it being a honeypot. Even though we want the honeypot to be compromised it shouldn’t pose a threat to other systems on the Internet. To achieve this, network traffic leaving the honeypot should be regulated and monitored. This is the most critical part of the entire setup. We do want honeypots to be hacked but we

don’t want to be liable for any damage caused to other systems via the honeypot. Logging is paramount to the working and success of a honeypot. The idea here is to let the attacker completely take over the system and record all possible information about the techniques used to compromise the system. One can also monitor the activities and events that happen after he succeeds in compromising the system, thought it should be done in careful way without harming other systems or networks. It is advisable to have multiple layers of logging on a honeypot. The better the information is gathered, the better the analysis can be performed. Multiple layers not only provide more information but also help in relating/confirming information between different layers. Even redundant layers can be helpful in cases where the blackhat detects the honeypot and tries to clear his traces in the logs. The logs should be checked on daily basis and, if possible, even more frequently. There are many questions that need to be answered beforehand with regard to the possibility of a honeypot being compromised. How do we find out the honeypot is compromised? How quickly will we be alerted? How do we backup the compromised system for analysis? What is the next step? Do we let the hacker know about the existence of the honeypot? Do we allow the attacker to continue? If yes, how do we restrict damage to other computers? The answers to all these questions should be carefully thought out and planned [1], [6] and [7].

Security Issues:
Honeypots don’t provide security (they are not a securing tool) for an organization but if implemented and used correctly they enhance existing security policies and techniques. Honeypots can be said to generate a certain degree of security risk and it is the administrator’s responsibility to deal with it. The level of security risk depends on their implementation and deployment. There are two views of how honeypot systems should handle its security risks.

- ➔ Honeypots that fake or simulate: There are honeypot tools that simulate or fake services or even fake vulnerabilities. They deceive any attacker to think they are accessing one particular system or service. A properly designed tool can be helpful in gathering more information about a variety of servers and systems. Such systems are easier to deploy and can be used as alerting systems and are less likely to be used for further illegal activities.

- ➔ Honeypots that are real systems: This is a viewpoint that states that honeypots should not be anything different from actual systems since the main idea is to secure the systems that are in use. These honeypots don’t fake or simulate anything and are implemented using actual systems and servers that are in use in the real world. Such honeypots reduce the chances of the hacker knowing that he is on a honeypot. These honeypots have a high risk factor and cannot be deployed everywhere. They need a controlled environment and administrative expertise.

A compromised honeypot is a potential risk to other computers on the network or for that matter the Internet. Many systems are compromised and used in attacks such as Denial of Service. The honeypot must be constantly supervised at regular intervals. A network dedicated to honeypots helps not only in supervising honeypots but also helps in detecting attacks and restricting the honeypot from being used to attack other computers. Honeypots don’t guarantee every attack will be detected. Honeypots can only detect attacks from traffic directed at them. So a smart hacker who detects a honeypot in a network that he is trying to compromise will avoid sending any traffic to the honeypot. If this happens the honeypot will be completely oblivious of any ongoing attacks on other computers in the network. Honeypots that run services with known bugs or have user created holes don’t help in adding any extra knowledge but can be used to gather statistical data or reveal identities of blackhat or blackhat systems.
An administrator could be charged with negligence if he intentionally or un-intentionally allows a compromised honeypot to be used to attack other systems. Also any information (false or genuine) that the hackers gain from the honeypot can sometimes adversely affect the organization [3], [5] and [8].

Legal issues:
To start with, a honeypot should be seen as an instrument of learning. Though there is a viewpoint that honeypots can be used to “trap” hackers. Such an idea can be considered as an entrapment. The legal definition of entrapment is *“Entrapment is the conception and planning of an offense by an officer, and his procurement of its commission by one who would not have perpetrated it except for the trickery, persuasion, or fraud of the officers."*
This legal definition applies only to law-enforcement, so organizations or educational institutions cannot be charged with entrapment. The key to establishing entrapment is “predisposition” would the attacker have committed the crime without “encouragement activity”. Also as long as one doesn’t entice the hacker in any way it cannot be considered entrapment. The issue of privacy is also of concern with respect to the monitoring

and intercepting of communication. Honeypots are systems intended to be used by nobody. They do not provide user accounts or services of any kind to the public and thus should not be violating any privacy laws. Also privacy laws change from country to country and should be taken into consideration before deploying honeypots. Honeypots come with a certain degree of liability. Administrators or researchers who deploy honeypots are responsible for any security threats that the honeypots pose. As such an administrator is liable for any compromised system that is under his supervision.

Role of Honeypots in Network Security:
Honeypots and related technologies have generated great deal of interest in the past two years. Honeypots can be considered to be one of the latest technologies in network security today. Project Honeynet is actively involved with deployment and study of honeypots. Honeypots are used extensively in research and it's a matter of time that they will be used in production environments as well.

Configuration and Deployment of a Honeynet:
The first step in deploying honeypots is to determine what we want to do with them. In this thesis the purpose was to learn the uses of honeypots and how to effectively deploy honeypots. Simply stated the greater the interaction the more we can learn. A honeynet can be defined as a collection of high interaction honeypots configured in a secured and monitored environment. A honeynet is a network constructed to aid the deployment of honeypots within. The honeypots in a honeynet are normal day to day systems, running the regular servers and services with nothing being emulated of faked. It was decided to have such a carefully constructed environment within which the honeypots could be deployed and monitored effectively.

Security of the gateway:
The security of the gateway was of prime importance since all administration and monitoring activity revolves around it. All services on the gateway except those absolutely required were disabled and the latest patches applied. In order to be able to remotely administer the gateway and firewall a secure shell server was setup on a non-standard port. All unnecessary user accounts and services were disabled and the system was hardened as much as possible. Finally it was tested with *Nessus* a powerful scanner that helps in detecting any security holes in a computer that hackers might exploit. *Nessus* generates detailed reports along with graphs, which can be viewed in *html* format [2] and [10].

The Linux Honeypot:
This honeypot runs Red Hat (www.redhat.com) with basic configuration plus the services that were desired to be monitored. The idea here was to make the system look like a regular system that has a few servers running but nothing that is being used extensively. Honeypots can also be configured to fake activity in the form of logins, emails etc to make them appear as if they are being used daily. It was elected to opt for the other option where the system looks like one that has been installed and configured but for the most part left unattended.

- ➔ **Services-**Web server (http), ftp, SSH (secure shell), mail server and a database services are the most common services used in the real world. So it was decided to run these on the Linux honeypot. Since Redhat Linux distribution comes with *apache* web server it was used as web server running on its default port. The web server was installed with the default configurations and settings. *Apache-tomcat*, a java based web server was also configured on port 8080 with the default out-of-the-box configurations. Other servers that were installed were secure shell (SSH) server, a file transfer protocol (FTP) server, a *mysql* (an open-source database server) server and *sendmail*. All there servers were configured using their default configuration files. The ftp server was setup to accept anonymous connections. The only modifications made to the configuration files were the ones that turned on all the respective logging options.
- ➔ **Modifying syslog-** The first thing that hackers do after compromising a system is disable the system logger and/or delete logs in order to cover their traces. The *syslog* source code was therefore modified to read a configuration file from a non-standard directory with a nonstandard name. This configuration file was setup to send all log messages to a remote *syslog* server. After make the necessary changes to the source code, the compiled binaries (*syslogd* and *klogd)* were renamed to something less conspicuous like *lpd* (a print server). The default *syslog* server was left running without any modifications.

- ➔ **Modifying the Shell-** *Bash* (the default shell on Linux) source code was also modified to send all the shell commands and keystrokes to a separate log file on the gateway computer. A separate client-server setup was developed to send these logs to the gateway. A second layer of *bash* logging was also added by modifying *bash* to spawn a *script* session every time a *bash* command was executed. A *script* session captures both the commands and their output to a file which is then logged to the *syslog* server.

➔ **User accounts-** It was decided to make the honeypot look as if it has been left unattended. So except for a couple of user accounts no other changes were made to the default user accounts. To increase interaction level one can add user accounts with varying degree of password complexity with the hope of a hacker being able to crack the passwords. User activity can also be faked to make it look like a busily used system. This might attract some hackers especially the skilled ones who find it challenging to break into such systems. It might also deter the unskilled "script kiddie" who might back off with the fear of being caught.

➔ **Integrity checking-** After installing all the software and servers *Tripwire* was used to create a database of the md5sums of all the system binaries and configuration files. The database and configuration files were then saved to a floppy disk and tripwire uninstalled. This database will help check which binaries or files were modified if the system gets compromised.

The Windows Honeypot:
Windows professional was selected as the operating environment on the Windows honeypot. As with the Linux honeypot the Windows honeypot was made to look like it had been installed and left alone. The latest patches from Microsoft were installed and the following configurations performed:

➔ **Internet Information Services (IIS)-** IIS is the Windows suite of web server (HTTP), FTP server and SMTP server. Over the years it has been subject to plethora of attacks like *Code-Red*, *Nimda27*. IIS web service, & ftp service with the default options were configured and a few user directories under the root folder of the web/ftp server were created.

➔ **Other Services -** The *mysql* server package for Windows environment was installed. *Apache- Tomcat*, a java based web server was installed and configured on port 8080 with the default settings. All the default Windows services like net logon, netbios, and remote procedure call were left unchanged. All these service have some security flaws. Netlogon supports authentication of account logon events for computers in a domain. Netbios over the years has had many flaws such as "null session flaw".

➔ **Shares-** A share on Windows is a resource like directory, printer etc that has been made available to other systems. This share can then be accessed remotely from other computers. Many viruses and worms scan for Windows shares. So a few shares were created on the Windows honeypot and gave the Windows group "Everyone"(any user) read permissions on these shares.

➔ **Event Logging-** All the security and auditing options available in Windows environment were enabled. A *Perl script* periodically collects the event logs and sends them to the remote log-collecting server. Few PHP scripts were written which allowed access to the Windows event logs via a web browser [1], [8], [9] and [11].

Attracting Hackers:
A question on "attracting hackers" posted on a honeypot mailing list at securityfocus.com received many interesting replies. Many people seem to think that there is no need to attract hackers and that putting a system on the Internet is sufficient. Also attracting might not be a good idea as it might result in a security threat to other computers in the network. It was decided not to do anything special to attract hackers. Enticing hackers is a debatable topic in the honeypot community and many honeypot researchers feel that a honeypot should never actively entice a hacker but, by definition, it should passively wait for probes, scans and attacks. In this thesis, the honeypots that could have been mainly done the following point:

• Having the honeypot access IRC networks, especially hacker-related networks, will definitely attract hackers. Some of the attacks are done using the IRC client itself.
• Scan known hacker networks from the honeypot to see if you can get them to retaliate. This will definitely attract them though this will expose your network to attacks like denial of service.
• Lastly don't patch known bugs or install un-patched versions of software. With an unpatched system, you will catch script kiddies who just ran some automated hack tool or read about a way to hack in.

III. PROPOSED TECHNIQUES

Important computers such as servers are usually protected, patched and updated and maintained better than computers such as test servers, workstations in school labs, desktops used by organizational staff etc. These ubiquitous computers are the ones that administrators find it difficult to secure. If you think a system is hidden from the world or is not an important and that it will be left alone you are wrong. In fact computers which are not regularly monitored are the first ones to be compromised. There are many reasons why these computers will be attacked. A "Script kiddie" picks random computers to try out his exploit tools and code. A more experienced hacker will want to use the computer in order to cover his tracks before attacking more important computer like commercial servers. Another use of such unattended computers is to use them in a "Denial of Service" attack. Organizations also face a considerable degree of security risk from within their own network. Recent CERT reports show that about 71% of the attacks were instigated by insiders. The element of the malicious insider poses an even bigger threat. With more and more computers/networks to secure, an NIDS should be easy to use, both in installation and configuration, since many network administrators are concerned with securingAnd managing a larger number of computers systems. Alerting is also an important factor with an intrusion detection system. An intrusion detection system should provide a reasonably good alerting mechanism such as email of some other network based mechanism. Another thing that would help an administrator is having a central management system using which he can not only view logs and alerts but also configure the intrusion detection system. Having all these issues in mind we proposed an intrusion detection system which takes into account the issues described above. Since there are quite a few freely available intrusion detection systems, the idea behind this project was to incorporate some of these tools and add some new techniques into an intrusion detection system package. The network administrator who has to manage a reasonably large number of computers in the same local area network is the main user that this product intends to target. Honeypots provided the deception systems which would work for generating early warnings. Another use of the honeypots was in conjunction with the tracing module. The honeypots would provide information to the tracing module which can then be used to trace the attacker back to the source. The honeypot services are just simulations and not real servers and hence don't have any security concerns. They merely act as a decoy and an early warning system. The following services are simulated-

- ➔ HTTP – Fake web server versions, web-pages and error messages.
- ➔ FTP – Fake ftp sessions, logins and error messages.
- ➔ POP3 - Simple pop3 commands and messages.
- ➔ SSH & TELNET – Fake SSH and TELNET servers.

The first and foremost requirement is the intrusion detection technique itself. The most common and widely used technique in intrusion detection is signature-based pattern matching. The idea behind this technique is to simply scan all network packets either on a per-host basis or the entire network itself and match these packets with known attack patterns usually called attack signatures. If a network packet matches a known attack then trigger an alert or perform some function to prevent it. This project aims at incorporating some of the tools already in use along with adding some of the newer concepts in intrusion detection. The intrusion detection system should be extendable in terms of attack signatures and detection rules and have the ability to add custom rules. *Snort,* an open source tool, was the intrusion detection tool of choice for this thesis work.

Logging Mechanism

The clients should be capable of both text based logging and logging to a database. The text based logging helps in deployment of clients with minimum dependences and requirements. Database logging helps in better storage, adds flexibility in terms of logging, and also allows expendability in terms of further processing of the logs.

Alerting Mechanism

Alerting methods can be email alerts, local system alarms. The frequency of emails and their content can be configured.

Tracing the Attack Source

Another valuable feature is to detect the source of the attack. There are several passive and active methods that can be used to trace an attacker back to the source. Among the active scanning tools *nmap,* is probably the most popular and feely available was the tool of choice.

Configuration of the Package

The intrusion detection mechanism itself should be configurable on a per client basis. The configuration can also be loaded using configuration files.

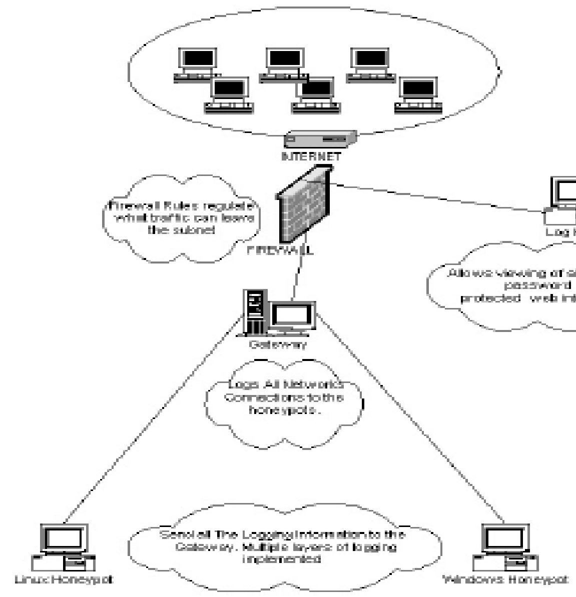
INTERNET
Firewall Rules regulate what traffic can leave the subnet
FIREWALL
Log Host
Allows viewing of all logs via password protected web interface
Gateway
Logs All Networks Connections to the honeypots.
Send all The Logging Information to the Gateway. Multiple layers of logging implemented
Linux Honeypot
Windows Honeypot

Figure 1 Working of the honeynet

Working:
The idea behind the entire package was to put together a set of tools that collectively work as an intrusion detection system and also as an early warning system. The honeypot module can be used to simulate many services. Some of the services or servers simulated as a part of this thesis were-

- Apache web server
- IIS web server
- Three different kinds of FTP severs
- A simple telnet server
- A simple SSH (Secure shell) server
- A POP3 server

These services can be configured to run on any port and the level of interaction can also be controlled. The module listens on the configured ports and carries out communication with the potential attacker. Since the system is not providing any service at that particular port, any traffic received on that port can be termed suspect. The module then logs the information and alerts the administrator. The module also tries to gather information about the attacker's IP address by analyzing the traffic and sending queries back to that IP address. Since these are not actually servers they do not pose a security threat and they can be turned off as per requirements. More services can be added and also the level of interaction can be increased. *Snort* works in parallel to the honeypots by analyzing traffic and matching it with a rule database. The honeypots act as a triggering mechanism while *snort* provides the intrusion detection functionality.
The package then sends the logged information via email alerts to the administrator. Both *snort* and the honeypots log to in text format as well as to a database. A Web based front-end is also available to view the logs and perform query on these logs. The following screen shot show the information that honeypot module logs. In the first row of the logs we can see that a FTP session was established between the host and the attacker. The module records the commands in the ftp session were logged and a trace of the attacker's IP address.

IV. RESULT

The honeypots have never been compromised so we are yet to see a complete intrusion but nevertheless the honeypots recorded enough data to show that computers today are not safe from attackers. These honeypots were behind multiple networks and were not providing any public services, nor were they advertised in any way. Having the honeypot sit on a commercial ISP network would invite more hack attempts. Nevertheless it really doesn't matter where your computers live, they are bound to be probed, scanned and attacked. The honeypots were online for five months. Five months logging produced interesting and significant results discussed below. This clearly shows that any computer on the Internet is not safe from probes, scans and attempted exploits. In general portscans occurred simultaneously on both honeypots indicating that the scans were generated by scripts or tools. Exploit attempts were directed only to the relevant systems. For example the mod-ssl exploit was directed only at the Linux honeypot which was running apache. Even though the Windows honeypot was online for less time it received more connection attempts. There could be many reasons for this statistic. Windows is the most used OS among personal desktops. Often they are not fully patched and updated and easy to exploit.

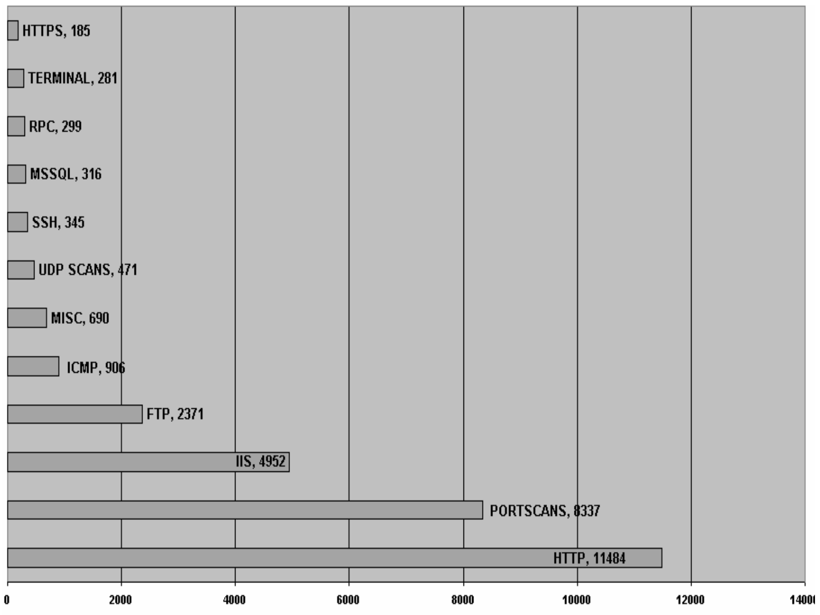

Figure 2 shows the plot of Category Vs Number of Occurrences.

HTTP exploit attempts are the highest among the connections that the honeypots received. There were many port scans and attempts to gather information such as server type or version. There are many tools in the wild that scan for a range of IP addresses looking for vulnerable web server versions.

V. CONCLUSION AND FUTURE WORK

In this work, we explored the concept of honeypots in depth and saw how it might be useful to the field of network security. The concept of honeypots is an important addition to the security field. Honeypots offer an offensive approach to intrusion detection and prevention. Most importantly, they serve as a learning tool for system administrators and also involved studying issues concerning intrusion detection systems the challenges that these systems faced. The Internet has become indispensable both at the organizational and personal level and so it will be the case with security systems. The use of honeypots and related technologies is on the rise. As awareness and interest in honeypots increases so will its use in an organization as a security tool. There is scope for development of honeypot tools which facilitate the different aspects of honeypots like logging, tracing back to the source etc. System modules for sophisticated keystroke logging, better filtering tools and utilities to capture encrypted traffic are a few things that could be worked on. One can even consider an out-of-the-box honeypot distribution with a modified kernel to make it easy for system administrators to deploy honeypots.

REFERENCES

[1] Yaser Alosefer and Omer Rana, “Honeyware: a web-based low interaction client honeypot”, Third IEEE International Conference on Software Testing, Verification, and Validation Workshops (ICSTW), pp. 410 – 417, 2010.

[2] Xiaoyan Sun, Yang Wang, Jie Ren, Yuefei Zhu and Shengli Liu, “Collecting Internet Malware Based on Client-side Honeypot”, 9th IEEE International Conference for Young Computer Scientists (ICVCS 2008), pp. 1493 – 1498, 2008.

[3] C. H. Nick Jap, P. Blanchfield, and K. S. Daniel Su, “The use of honeypot approach in software-based application protection for shareware programs”, IEEE International Conference on Computing & Informatics, (ICOCI '06), pp. 1-7, 2006.

[4] Jian Bao and Chang-peng Ji, and Mo Gao, “Research on network security of defense based on Honeypot”, IEEE International Conference on Computer Application and System Modeling (ICCASM), vol. 10, pp. V10-299 - V10-302, 2010.

[5] Anjali Sardana, R. C. Joshi, “Honeypot Based Routing to Mitigate DDoS Attacks on Servers at ISP Level”, IEEE International Symposiums on Information Processing (ISIP), pp. 505-509, 2008.
[6] Guanlin Chen, Hui Yao and Zebing Wang, “Research of Wireless Intrusion Prevention Systems based on Plan Recognition and Honeypot”, IEEE International Conference on Wireless Communications & Signal Processing (WCSP), pp. 1-5, 2009.
[7] Chao-Hsi Yeh and Chung-Huang Yang, “Design and Implementation of Honeypot Systems Based on Open-Source Software”, IEEE International Conference on Intelligence and Security Informatics (ISI), 265-266, 2008.
[8] Babak Khosravifar, Maziar Gomrokchi, Jamal Bentahar, “A Multi-Agent-based Approach to Improve Intrusion Detection Systems False Alarm Ratio by Using Honeypot”, IEEE International Conference on Advanced Information Networking and Applications Workshops, pp. 97 – 102, 2009.
[9] Wei Li-feng, Wang Xiao-bin, “Research on Honeypot Information Fusion Based on Game Theory”, Second IEEE International Conference on Computer Research and Development, pp. 803 – 806, 2010,
[10] Xinliang Wang, Fang Liu, LuYing Chen, Zhenming Lei, “Research for Scan Detection Algorithm of High-Speed Links Based on Honeypot”, 2nd IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC), pp. 66-70, 2010.
[11] Yun Yang and Jia Mi, “Design and Implementation of Distributed Intrusion Detection System based on Honeypot”, 2nd IEEE International Conference on Computer Engineering and Technology (ICCET), vol. 6, pp. V6-260 - V6-263, 2010.

AUTHORS PROFILE

Dr. Yogendra Kumar Jain presently working as head of the department, Computer Science & Engineering at Samrat Ashok Technological Institute Vidisha M.P India. The degree of B.E. (Hons) secured in E&I from SATI Vidisha in 1991, M.E. (Hons) in Digital Tech. & Instrumentation from SGSITS, DAVV Indore (M.P), India in 1999. The Ph. D. degree has been awarded from Rajiv Gandhi Technical University, Bhopal (M.P.) India in 2010. Research Interest includes Image Processing, Image compression, Network Security, Watermarking, Data Mining. Published more than 40 Research papers in various Journals/Conferences, which include 10 research papers in International Journals. **E-mail**: ykjain_p@yahoo.co.in.

Mrs. Surabhi Singh presently pursuing M.Tech of the department, Computer Science & Engineering at Samrat Ashok Technological Institute, Vidisha, M.P., India. The degree of M.C.A from Madhavan Institute of Technical Education & Science (Gwalior) affiliated to Rajive Gandhi Technical University, Bhopal, Madhya Pradesh, India. Research Interest includes Network Security, Data Mining. **E-mail:** surabhi241079@gmail.com.