ABSTRACT

Honeypot is an exciting new technology with enormous potential for the security community. It is resource which is intended to be attacked and compromised to gain more information about the attacker and his attack techniques.

They are a highly flexible tool that comes in many shapes and sizes. This paper deals with understanding what a honeypot actually is ,and how it works.

There are different varieties of honeypots. Based on their category they have different applications. This paper gives an insight into the use of honeypots in productive as well as educative environments.

This paper also discusses the advantages and disadvantages of honeypots , and what the future hold in store for them.

CONTENTS

1.	INTRODUCTION	3
2.	HONEYPOT BASICS	5
3.	TYPES OF HONEYPOTS	7
4.	VALUE OF HONEYPOT	17
5.	IMPLEMENTATION	22
6.	MERITS AND DEMERITS	26
7.	LEGAL ISSUES	28
8.	FUTURE OF HONEYPOTS	30
9.	CONCLUSION	31
10.	REFERENCES	32

INTRODUCTION

The Internet is growing fast and doubling its number of websites every 53 days and the number of people using the internet is also growing. Hence, global communication is getting more important every day. At the same time, computer crimes are also increasing. Countermeasures are developed to detect or prevent attacks - most of these measures are based on known facts, known attack patterns. Countermeasures such as firewalls and network intrusion detection systems are based on prevention, detection and reaction mechanism; but is there enough information about the enemy?

As in the military, it is important to know, who the enemy is, what kind of strategy he uses, what tools he utilizes and what he is aiming for. Gathering this kind of information is not easy but important. By knowing attack strategies, countermeasure scan be improved and vulnerabilities can be fixed. To gather as much information as possible is one main goal of a honeypot. Generally, such information gathering should be done silently, without alarming an attacker. All the gathered information leads to an advantage on the defending side and can therefore be used on productive systems to prevent attacks.

A honeypot is primarily an instrument for information gathering and learning. Its primary purpose is not to be an ambush for the blackhat community to catch them in action and to press charges against them. The focus lies on a silent collection of as much information as possible about their attack patterns, used programs, purpose of attack and the blackhat community itself. All this information is used to learn more about the blackhat proceedings and motives, as well as their technical knowledge and abilities. This is just a primary purpose of a honeypot. There are a lot of other possibilities for a honeypot - divert

hackers from productive systems or catch a hacker while conducting an attack are just two possible examples. They are not the perfect solution for solving or preventing computer crimes.

Honeypots are hard to maintain and they need operators with good knowledge about operating systems and network security. In the right hands, a honeypot can be an effective tool for information gathering. In the wrong, unexperienced hands, a honeypot can become another infiltrated machine and an instrument for the blackhat community.

This paper will present the basic concepts behind honeypots and also the legal aspects of honeypots.

HONEYPOT BASICS

Honeypots are an exciting new technology with enormous potential for the security community. The concepts were first introduced by several icons in computer security, specifically Cliff Stoll in the book "The Cuckoo's Egg", and Bill Cheswick's paper "An Evening with Bedford." Since then, honeypots have continued to evolve, developing into the powerful security tools they are today.

Honeypots are neither like Firewalls that are used to limit or control the traffic coming into the network and to deter attacks neither is it like IDS (Intrusion Detection Systems) which is used to detect attacks. However it can be used along with these. Honeypots does not solve a specific problem as such, it can be used to deter attacks, to detect attacks, to gather information, to act as an early warning or indication systems etc. They can do everything from detecting encrypted attacks in IPv6 networks to capturing the latest in on-line credit card fraud. It is this flexibility that gives honeypots their true power. It is also this flexibility that can make them challenging to define and understand. The basic definition of honeypots is:

A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource.

The main aim of the honeypot is to lure the hackers or attacker so as to capture their activities. This information proves to be very useful since information can be used to study the vulnerabilities of the system or to study latest techniques used by attackers etc. For this the honeypot will contain enough information (not necessarily real) so that the attackers get tempted. (Hence the name Honeypot – a sweet temptation for attackers)Their value lies

in the bad guys interacting with them. Conceptually almost all honeypots work they same. They are a resource that has no authorized activity, they do not have any production value.

Theoretically, a honeypot should see no traffic because it has no legitimate activity. This means any interaction with a honeypot is most likely unauthorized or malicious activity. Any connection attempts to a honeypot are most likely a probe, attack, or compromise. While this concept sounds very simple (and it is), it is this very simplicity that give honeypots their tremendous advantages (and disadvantages).

TYPES OF HONEYPOTS

Honeypots come in many shapes and sizes, making them difficult to get a grasp of. To better understand honeypots and all the different types, they are broken down into two general categories, low-interaction and high-interaction honeypots. These categories helps to understand what type of honeypot one is dealing with, its strengths, and weaknesses. Interaction defines the level of activity a honeypot allows an attacker.

Low-interaction honeypots have limited interaction, they normally work by emulating services and operating systems. Attacker activity is limited to the level of emulation by the honeypot. For example, an emulated FTP service listening on port 21 may just emulate a FTP login, or it may support a variety of additional FTP commands. The advantages of a low-interaction honeypot is their simplicity. These honeypots tend to be easier to deploy and maintain, with minimal risk. Usually they involve installing software, selecting the operating systems and services you want to emulate and monitor, and letting the honeypot go from there. This plug and play approach makes deploying them very easy for most organizations. Also, the emulated services mitigate risk by containing the attacker's activity, the attacker never has access to an operating system to attack or harm others. The main disadvantages with low interaction honeypots is that they log only limited information and are designed to capture known activity. The emulated services can only do so much. Also, its easier for an attacker to detect a low-interaction honeypot, no matter how good the emulation is, skilled attacker can eventually detect their presence. Examples of low-interaction honeypots include Specter, Honeyd, and KFSensor.

High-interaction honeypots are different, they are usually complex solutions as they involve real operating systems and applications. Nothing is emulated, the attackers are given the real thing. If one wants a Linux honeypot running an FTP server, they build a real Linux system running a real FTP server. The advantages with such a solution are two fold. First, extensive amounts of information are captured. By giving attackers real systems to interact with, one can learn the full extent of the attackers behavior, everything from new rootkits to international IRC sessions. The second advantage is high-interaction honeypots make no assumptions on how an attacker will behave. Instead, they provide an open environment that captures all activity. This allows high-interaction solutions to learn behavior one otherwise would not expect. An excellent example of this is how a Honeynet captured encoded back door commands on a non-standard IP protocol . However, this also increases the risk of the honeypot as attackers can use these real operating system to attack non-honeypot systems. As result, additional technologies have to be implemented that prevent the attacker from harming other non-honeypot systems. In general, high-interaction honeypots can do everything low-interaction honeypots can do and much more. However, they can be more complex to deploy and maintain. Examples of high-interaction honeypots include Symantec Decoy Server and Honeynets.

Low-interaction	*High-interaction*
Solution emulates operating systems and services.	No emulation, real OS and services are provided.
• Easy to install and deploy. • Captures limited amounts of information.	• Can capture far more information • Can be complex to install or deploy • Increased risk, as attackers are

• Minimal risk, as the emulated services controls attackers.	provided real OS to interact with.

Some people also classify honeypots as low,mid and high interaction honeypots; where mid-interaction honeypots are those with their interaction level between that of low and high interaction honeypots.

A few examples of honeypots and their varieties are:

BackOfficer Friendly

BOF (as it is commonly called) is a very simple but highly useful honeypot developed by Marcus Ranum and crew at NFR. It is an excellent example of a low interaction honeypot.

It is a great way to introduce a beginner to the concepts and value of honeypots. BOF is a program that runs on most Window based operating system. All it can do is emulate some basic services, such as http, ftp, telnet, mail, or BackOrrifice. Whenever some attempts to connect to one of the ports BOF is listening to, it will then log the attempt. BOF also has the option of "faking replies", which gives the attacker something to connect to. This way one can log http attacks, telnet brute force logins, or a variety of other activity (Screenshot). The value in BOF is in detection, similar to a burglar alarm. It can monitor only a limited number of ports, but these ports often represent the most commonly scanned and targeted services.

Specter

Specter is a commercial product and it is another 'low interaction' production honeypot. It is similar to BOF in that it emulates services, but it can emulate a far greater

range of services and functionality. In addition, not only can it emulate services, but emulate a variety of operating systems. Similar to BOF, it is easy to implement and low risk. Specter works by installing on a Windows system. The risk is reduced as there is no real operating system for the attacker to interact with. For example, Specter can emulate a web server or telnet server of the any operating system. When an attacker connects, it is then prompted with an http header or login banner. The attacker can then attempt to gather web pages or login to the system. This activity is captured and recorded by Specter, however there is little else the attacker can do. There is no real application for the attacker to interact with, instead just some limited, emulated functionality. Specters value lies in detection. It can quickly and easily determine who is looking for what. As a honeypot, it reduces both false positives and false negatives, simplifying the detection process. Specter also supports a variety of alerting and logging mechanisms. You can see an example of this functionality in a screen shot of Specter.

One of the unique features of Specter is that it also allows for information gathering, or the automated ability to gather more information about the attacker. Some of this information gathering is relatively passive, such as Whois or DNS lookups. However, some of this research is active, such as port scanning the attacker.

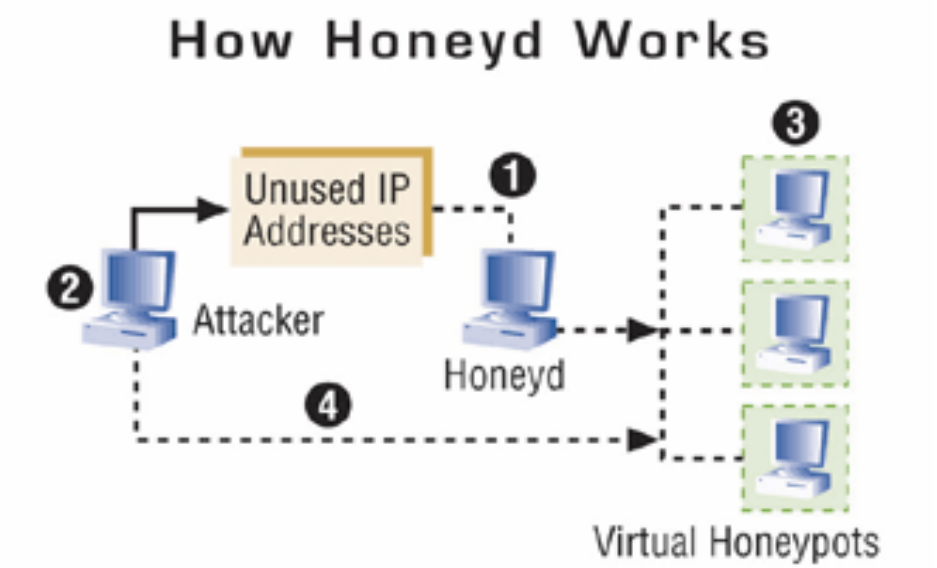
Homemade Honeypots

Another common honeypot is homemade. These honeypots tend to be low interaction. Their purpose is usually to capture specific activity, such as Worms or scanning activity. These can be used as production or research honeypots, depending on their purpose. Once again, there is not much for the attacker to interact with, however the risk is reduced because there is less damage the attacker can do. One common example is creating a service that listens on port 80 (http) capturing all traffic to and from the port. This is

commonly done to capture Worm attacks Homemade honeypots can be modified to do (and emulate) much more, requiring a higher level of involvement, and incurring a higher level of risk. For example, FreeBSD has a jail functionality, allowing an administrator to create a controlled environment within the operating system. The attacker can then interact with this controlled environment. The value here is the more the attacker can do, the more can be potentially learned. However, care must be taken, as the more functionality the attacker can interact with, the more can go wrong, with the honeypot potentially compromised.

Honeyd

Created by Niels Provos, Honeyd is an extremely powerful, OpenSource honeypot. Designed to run on Unix systems, it can emulate over 400 different operating systems and thousands of different computers, all at the same time. Honeyd introduces some exciting new features. First, not only does it emulate operating systems at the application level, like Specter, but it also emulates operating systems at the IP stack level. This means when someone Nmaps the honeypot, both the service and IP stack behave as the emulated operating system. Currently no other honeypot has this capability (CyberCop Sting did have this capability, but is no longer available). Second, Honeyd can emulate hundreds if not thousands of different computers all at the same time. While most honeypots can only emulate one computer at any point in time, Honeyd can assume the identity of thousands of different IP addresses. Third, as an OpenSource solution, not only is it free to use, but it will expotentially grow as members of the security community develop and contribute code.



Honeyd monitors unused IP space ❶. When an attacker ❷ probes an unused IP, Honeyd detects the probe, takes over that IP via ARP spoofing, then creates a virtual honeypot ❸ for the attacker to interact with (Honeyd can create multiple virtual honeypots to fool attackers on all unused addresses). The attacker is fooled into thinking he is interacting with a successful hacked system ❹. In addition, Honeyd automatically updates its list of unused IPs as systems are added or removed from the network.

Honeyd is primarily used for detecting attacks. It works by monitoring IP addresses that are unused, that have no system assigned to them. Whenever an attacker attempts to probe or attack an non-existant system, Honeyd, through Arp spoofing, assumes the IP address of the victim and then interacts with the attacker through emulated services. These emulates services are nothing more then scripts that react to predetermined actions. For example, a script can be developed to behave like a Telnet service for a Cisco router, with

the Cisco IOS login interface. Honeyd's emulated services are also Open Source, so anyone can develop and use their own. The scripts can be written in almost any language, such as shell or Perl. Once connected, the attacker believes they are interacting with a real system. Not only can Honeyd dynamically interact with attackers, but it can detect activity on any port. Most low interaction honeypots are limited to detecting attacks only on the ports that have emulated services listening on. Honeyd is different, it detects and logs connections made to any port, regardless if there is a service listening. The combined capabilities of assuming the identity of non-existant systems, and the ability to detect activity on any port, gives Honeyd incredible value as a tool to detect unauthorized activity. I highly encourage people to check it out, and if possible to contribute new emulated services.

Mantrap

Produced by Recourse, Mantrap is a commercial honeypot. Instead of emulating services, Mantrap creates up to four sub-systems, often called 'jails'. These 'jails' are logically discrete operating systems separated from a master operating system (see Diagram.) Security administrators can modify these jails just as they normally would with any operating system, to include installing applications of their choice, such as an Oracle database or Apache web server. This makes the honeypot far more flexible, as it can do much more. The attacker has a full operating system to interact with, and a variety

of applications to attack. All of this activity is then captured and recorded. Not only can we detect port scans and telnet logins, but we can capture rootkits, application level attacks, IRC chat session, and a variety of other threats. However, just as far more can be learned, so can more go wrong. Once compromised, the attacker can used that fully functional operating system to attack others. Care must be taken to mitigate this risk. As such, it can be

categorized this as a mid-high level of interaction. Also, these honeypots can be used as either a production honeypot (used both in detection and reaction) or a research honeypot to learn more about threats. There are limitations to this solution. The biggest one is that we are limited to only what the vendor supplies us. Currently, Mantrap only exists on Solaris operating system.

Honeynets

Honeynets represent the extreme of research honeypots. They are high interaction honeypots, one can learn a great deal, however they also have the highest level of risk.

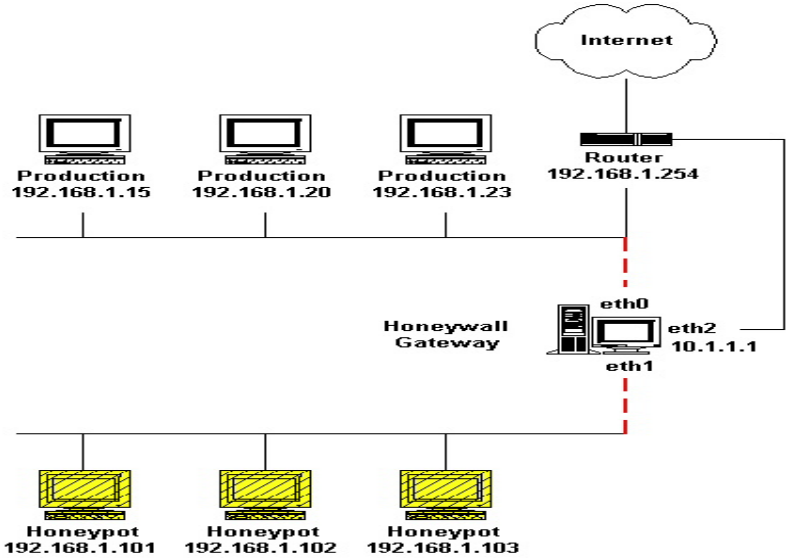


Fig: A honeynet

Their primary value lies in research, gaining information on threats that exist in the Internet community today. A Honeynet is a network of production systems. Unlike many of the honeypots discussed so far, nothing is emulated. Little or no modifications are made to the honeypots. The idea is to have an architecture that creates a highly controlled network, one where all activity is controlled and captured. Within this network we place our

intended victims, real computers running real applications. The bad guys find, attack, and break into these systems on their own initiative. When they do, they do not realize they are within a Honeynet. This gives the attackers a full range of systems, applications, and functionality to attack. All of their activity, from encrypted SSH sessions to emails and files uploads, are captured without them knowing it. This is done by inserting kernel modules on the victim systems that capture all of the attacker's actions. From this we can learn a great deal, not only their tools and tactics, but their methods of communication, group organization, and motives. However, with this capability comes a great deal of risk. A variety of measures must be taken to ensure that once compromised, a Honeynet cannot be used to attack others. Honeynets do this using a Honeywall gateway. This gateway allows inbound traffic to the victim systems, but controls the outbound traffic using intrusion prevention technologies. This gives the attacker the flexibility to interact with the victim systems, but prevents the attacker from harming other non-Honeynet computers. Honeynets are primarily research honeypots. They could be used as production honeypots, specifically for detection or reaction, however it is most likely not worth the time and effort

We have reviewed six different types of honeypots. No one honeypot is better than the other, each one has its advantages and disadvantages, it all depends on what is to be achieved. To more easily define the capabilities of honeypots, we have categorized them based on their level of interaction. The greater interaction an attacker has, the more we can learn, but the greater the risk. For example, BOF and Specter represent low interactions honeypots. They are easy to deploy and have minimal risk. However, they are limited to emulating specific services and operating systems, used primarily for detection. Mantrap and Honeynets represent mid-to-high interaction honeypots. They can give far greater depth of information, however more work and greater risk is involved

Sometimes, honeypots are also classified as Hardware based and Software based honeypots.

Hardware-based honeypots are servers, switches or routers that have been partially disabled and made attractive with commonly known misconfigurations. They sit on the internal network, serving no purpose but to look real to outsiders. The operating system of each box, however, has been subtly disabled with tweaks that prevent hackers from really taking it over or using it to launch new attacks on other servers.

Software emulation honeypots, on the other hand, are elaborate deception programs that mimic real Linux or other servers and can run on machines as low-power as a 233-MHz PC. Since an intruder is just dancing with a software decoy, at no time does he come close to actually seizing control of the hardware, no matter what the fake prompts seem to indicate. Even if the hacker figures out that it's a software honeypot, the box on which it's running should be so secure or isolated that he couldn't do anything but leave anyway.Software emulation might be more useful for corporate environments where business secrets are being safeguarded.

VALUE OF HONEYPOTS

Now that we have understanding of two general categories of honeypots, we can focus on their value. Specifically, how we can use honeypots. Once again, we have two general categories, honeypots can be used for production purposes or research. When used for production purposes, honeypots are protecting an organization. This would include preventing, detecting, or helping organizations respond to an attack. When used for research purposes, honeypots are being used to collect information. This information has different value to different organizations. Some may want to be studying trends in attacker activity, while others are interested in early warning and prediction, or law enforcement. In general, low-interaction honeypots are often used for production purposes, while high-interaction honeypots are used for research purposes. However, either type of honeypot can be used for either purpose. When used for production purposes, honeypots can protect organizations in one of three ways; prevention, detection, and response. We will take a more in-depth look at how a honeypot can work in all three.

1. ***Prevention*** : Honeypots can help prevent attacks in several ways. The first is against automated attacks, such as worms or auto-rooters. These attacks are based on tools that randomly scan entire networks looking for vulnerable systems. If vulnerable systems are found, these automated tools will then attack and take over the system (with worms self-replicating, copying themselves to the victim). One way that honeypots can help defend against such attacks is slowing their scanning down, potentially even stopping them. Called sticky honeypots, these solutions monitor unused IP space. When probed by such scanning activity, these

honeypots interact with and slow the attacker down. They do this using a variety of TCP tricks, such as a Windows size of zero, putting the attacker into a holding pattern. This is excellent for slowing down or preventing the spread of a worm that has penetrated the internal organization. One such example of a sticky honeypot is LaBrea Tarpit. Sticky honeypots are most often low-interaction solutions (one can almost call them 'no-interaction solutions', as they slow the attacker down to a crawl).

Honeypots can also be used to protect the organization from human attackers. The concept is deception or deterrence. The idea is to confuse an attacker, to make him waste his time and resources interacting with honeypots. Meanwhile, the organization being attacked would detect the attacker's activity and have the time to respond and stop the attacker.

This can be even taken one step farther. If an attacker knows an organization is using honeypots, but does not know which systems are honeypots and which systems are legitimate computers, they may be concerned about being caught by honeypots and decided not to attack your organizations. Thus the honeypot deters the attacker. An example of a honeypot designed to do this is Deception Toolkit, a low-interaction honeypot.

2. ***Detection :*** The second way honeypots can help protect an organization is through detection. Detection is critical, its purpose is to identify a failure or breakdown in prevention. Regardless of how secure an organization is, there will always be failures, if for no other reasons then humans are involved in the process. By detecting an attacker, you can quickly react to them, stopping or mitigating the damage they do. Traditionally, detection has proven extremely difficult to do. Technologies such as IDS sensors and systems logs have proved ineffective for several

reasons. They generate far too much data, large percentage of false positives (i.e. alerts that were generated when the sensor recognized the configured signature of an "attack", but in reality was just valid traffic), inability to detect new attacks, and the inability to work in encrypted or IPv6 environments. Honeypots excel at detection, addressing many of these problems of traditional detection. Since honeypots have no production activity, all connections to and from the honeypot are suspect by nature. By definition, anytime a connection is made to the honeypot, this is most likely an unauthorized probe, scan, or attack. Anytime the honeypot initiates a connection, this most likely means the system was successfully compromised. This helps reduce both false positives and false negatives greatly simplifying the detection process by capturing small data sets of high value, it also captures unknown attacks such as new exploits or polymorphic shellcode, and works in encrypted and IPv6 environments. In general, low-interaction honeypots make the best solutions for detection. They are easier to deploy and maintain then high-interaction honeypots and have reduced risk.

3. ***Response :*** The third and final way a honeypot can help protect an organization is in reponse. Once an organization has detected a failure, how do they respond? This can often be one of the greatest challenges an organization faces. There is often little information on who the attacker is, how they got in, or how much damage they have done. In these situations detailed information on the attacker's activity are critical. There are two problems compounding incidence response. First, often the very systems compromised cannot be taken offline to analyze. Production systems, such as an organization's mail server, are so critical that even though its been hacked, security professionals may not be able to take the system down and do a proper forensic analysis. Instead, they are limited to analyze the live system while still

providing production services. This cripples the ability to analyze what happened, how much damage the attacker has done, and even if the attacker has broken into other systems. The other problem is even if the system is pulled offline, there is so much data pollution it can be very difficult to determine what the bad guy did. By data pollution, I mean there has been so much activity (user's logging in, mail accounts read, files written to databases, etc) it can be difficult to determine what is normal day-to-day activity, and what is the attacker. Honeypots can help address both problems. Honeypots make an excellent incident resonse tool, as they can quickly and easily be taken offline for a full forensic analysis, without impacting day-to-day business operations. Also, the only activity a honeypot captures is unauthorized or malicious activity. This makes hacked honeypots much easier to analyze then hacked production systems, as any data you retrieve from a honeypot is most likely related to the attacker. The value honeypots provide here is quickly giving organizations the in-depth information they need to rapidly and effectively respond to an incident. In general, high-interaction honeypots make the best solution for response. To respond to an intruder, you need in-depth knowledge on what they did, how they broke in, and the tools they used. For that type of data you most likely need the capabilities of a high-interaction honeypot.

Up to this point we have been talking about how honeypots can be used to protect an organization. We will now talk about a different use for honeypots, research.

Honeypots are extremely powerful, not only can they be used to protect your organization, but they can be used to gain extensive information on threats, information few other technologies are capable of gathering. One of the greatest

problems security professionals face is a lack of information or intelligence on cyber threats. How can we defend against an enemy when we don't even know who that enemy is? For centuries military organizations have depended on information to better understand who their enemy is and how to defend against them. Why should information security be any different?

Research honeypots address this by collecting information on threats. This information can then be used for a variety of purposes, including trend analysis, identifying new tools or methods, identifying attackers and their communities, early warning and prediction, or motivations. One of the most well known examples of using honeypots for research is the work done by the Honeynet Project, an all volunteer, non-profit security research organization. All of the data they collect is with Honeynet distributed around the world. As threats are constantly changing, this information is proving more and more critical.

IMPLEMENTATION

Honeypot Location

A honeypot does not need a certain surrounding environment as it is a standard server with no special needs.A honeypot can be placed anywhere a server could be placed. But certainly, some places are better for certain approaches as others.

A honeypot can be used on the Internet as well as the intranet, based on the needed service. Placing a honeypot on the intranet can be useful if the detection of some bad guys inside a private network is wished. It is especially important to set the internal thrust for a honeypot as low as possible as this system could be compromised, probably without immediate knowledge.

If the main concern is the Internet, a honeypot can be placed at two locations:

- In front of the firewall (Internet)
- DMZ
- Behind the firewall (intranet)

Each approach has its advantages as well as disadvantages. Sometimes it is even impossible to choose freely as placing a server in front of a firewall is simply not possible or not wished.

By placing the honeypot in front of a firewall , the risk for the internal network does not increase. The danger of having a compromised system behind the firewall is eliminated. A honeypot will attract and generate a lot of unwished traffic like portscans or attack patterns. By placing a honeypot outside the firewall, such events do not get logged by

the firewall and an internal IDS system will not generate alerts. Otherwise, a lot of alerts would be generated on the firewall or IDS.Probably the biggest advantage is that

the firewall or IDS, as well as any other resources, have not to be adjusted as the honeypot is outside the firewall and viewed as any other machine on the external network. The disadvantage of placing a honeypot in front of the firewall is that internal attackers cannot be located or trapped that easy, especially if the firewall limits outbound traffic and therefore limits the traffic to the honeypot.

Placing a honeypot inside a DMZ seems a good solution as long as the other systems inside the DMZ can be secured against the honeypot. Most DMZs are not fully accessible as only needed services are allowed to pass the firewall. In such a case,placing the honeypot in front of the firewall should be favored as opening all corresponding ports on the firewall is too time consuming and risky.

A honeypot behind a firewall can introduce new security risks to the internal network, especially if the internal network is not secured against the honeypot through additional firewalls. This could be a special problem if the IP’ s are used for authentication. It is important to distinguish between a setup where the firewall enables access to the honeypot or where access from the Internet is denied. By placing the honeypot behind a firewall, it is inevitable to adjust the firewall rules if access from the Internet should be permitted. The biggest problem arises as soon as the internal honeypot is compromised by an external attacker. He gains the possibility to access the internal network through the honeypot. This traffic will be unstopped by the firewall as it is regarded as traffic to the honeypot only, which in turn is granted. Securing an internal honeypot is therefore

mandatory, especially if it is a high-involvement honeypot. With an internal honeypot it is also possible to detect a misconfigured firewall which forwards

unwanted traffic from the Internet to the internal network. The main reason for placing a honeypot behind a firewall could be to detect internal attackers.

The best solution would be to run a honeypot in its own DMZ, therefore with a preliminary firewall. The firewall could be connected directly to the Internet or intranet, depending on the goal. This attempt enables tight control as well as a flexible environment with maximal security.

How does a Honeypot Gather Information

Obviously a honeypot must capture data in an area that is not accessible to an attacker. Data capture happens on a number of levels.

Firewall Logs—A Packet Sniffer (or similar IDS sensor)—The IDS should be configured to passively monitor network traffic (for an added level of invisibility, one might set the system up to have no IP address or, in some instances, the sniffer could be configured to completely lack an IP stack). This will capture all cleartext communication, and can read keystrokes.

Local and Remote Logs—These should be set up just as it would on any other system, and will possibly be disabled, deleted, or modified by an experienced hacker, but plenty of useful information will still be available from all the previous capture methods.

Remotely Forwarded Logs—Will capture data on a remote log and then instantly forward the data to a system even further out of the range of the attacker,so that the attacker cannot be warned that all his activities are watched or try to modify the captured data.

Limiting Outbound Attacks

To protect oneself from any sort of third party liabilities, an individual deploying a honeypot will likely want some kind of safeguard. Firewalls can be configured to let an unlimited number of inbound connections, while limiting outbound connections to a specific number (be it 10 outbound connections, or 50). This method lacks flexibility, and could shut an attacker out at a critical point (in the middle of an IRC session, or before they have retrieved all of their tools). A more flexible option is as follows: a system configured as a layer 2 bridge (which will lack all TCP activity, thus being harder to detect). The system can be configured to monitor all activity and can utilize a signature database to distinguish a known attack from any non-aggressive activity (and instead of blocking the attack, it can simply add some data to the packet to render it ineffectual). It can also throttle bandwidth (to quench a DDoS attack). This is a very effective way to protect other systems; however, it will not block unknown or new attacks.

Putting the Honey into the Pot

An advanced honeypot is a fully functional OS, and therefore can be filled with financial information, e-mails with passwords for other honeypots, databases of fake customers—anything that might motivate an attacker to compromise the system. An individual could set up a web server that explains that the law services of so and so and so and so from San Francisco are currently setting up their systems to do online consultation for big banks and other big businesses. A whole network of honeypots sits in a secure environment behind a firewall that an attacker would need to break through. The network might have loads of fake data and e-mail; a large playing field for an advanced hacker to Wander through.

MERITS AND DEMERITS

Merits: Honeypots have a large number of merits in its favour. They are :

- *Small data sets of high value:* Honeypots collect small amounts of information. Instead of logging a one GB of data a day, they can log only one MB of data a day. Instead of generating 10,000 alerts a day, they can generate only 10 alerts a day. Remember, honeypots only capture bad activity, any interaction with a honeypot is most likely unauthorized or malicious activity. As such, honeypots reduce 'noise' by collectin only small data sets, but information of high value, as it is only the bad guys. This means its much easier (and cheaper) to analyze the data a honeypot collects and derive value from it.
- *New tools and tactics:* Honeypots are designed to capture anything thrown at them, including tools or tactics never seen before.
- *Minimal resources:* Honeypots require minimal resources, they only capture bad activity. This means an old Pentium computer with 128MB of RAM can easily handle an entire class B network sitting off an OC-12 network.
- *Encryption or IPv6:* Unlike most security technologies (such as IDS systems) honeypots work fine in encrypted or IPv6 environments. It does not matter what the bad guys throw at a honeypot, the honeypot will detect and capture it.
- *Information:* Honeypots can collect in-depth information that few, if any other technologies can match.
- *Simplicty:* Finally, honeypots are conceptually very simple. There are no fancy algorithms to develop, state tables to maintain, or signatures to update. The simpler a technology, the less likely there will be mistakes or misconfigurations.

Demerits: Like any technology, honeyopts also have their weaknesses. It is because of this they do not replace any current technology, but work with existing technologies.

- *Limited view:* Honeypots can only track and capture activity that directly interacts with them. Honeypots will not capture attacks against other systems, unless the attacker or threat interacts with the honeypots also.
- *Risk:* All security technologies have risk. Firewalls have risk of being penetrated, encryption has the risk of being broken, IDS sensors have the risk of failing to detect attacks. Honeypots are no different, they have risk also. Specifically, honeypots have the risk of being taken over by the bad guy and being used to harm other systems. This risk various for different honeypots. Depending on the type of honeypot, it can have no more risk then an IDS sensor, while some honeypots have a great deal of risk.

LEGAL ISSUES

In the past there has been some confusion on what are the legal issues with honeypots. There are several reasons for this. First, honeypots are relatively new. Second, honeypots come in many different shapes and sizes and accomplish different goals. Based on the different uses of honeypots different legal issues apply. Last, there are no precedents for honeypots. There are no legal cases recorded on the issues. The law is developed through cases. Without cases directly on point, we are left trying to predict, based on cases in other contexts, how courts will treat honeypots. Until a judge gives a court order, we will really never know.

With honeypots, there are three main issues that are commonly discussed: entrapment, privacy, and liability.

- Liability: You can potentially be held liable if your honepyot is used to attack or harm other systems or organizations. This risk is the greatest with Research honeypots.
- Privacy: Honeypots can capture extensive amounts of information about attackers, which can potentially violate their privacy. Once again, this risk is primarily with Research honeypots. However in case of honeypot there is exemption. It means that security technologies can collect information on people (and attackers), as long as that technology is being used to protect or secure your environment. In other words, these technologies are now exempt from privacy restrictions. For example, an IDS sensor that is used for detection and captures network activity is doing so to detect (and thus enable organizations to respond to) unauthorized activity. Such a technology is most likely not considered a violation of privacy.

- Entrapment: For some odd reason, many people are concerned with the issue of entrapment. Entrapment, by definition is "a law-enforcement officer's or government agent's inducement of a person to commit a crime, by means of fraud or undue persuasion, in an attempt to later bring a criminal prosecution against that person." Think about it, entrapment is when you coerce or induce someone to do something they would not normally do. Honeypots do not induce anyone. Attackers find and break into honeypots on their own initiative. People often question the idea of creating targets of high value, for example honeypots that are ecommerce sites or advertised as having government secrets. Even then, such honeypots are most likely not a form of entrapment as you are not coercing them into breaking into the honeypot. The bad guy has already decided to commit unauthorized activity, one is merely providing a different target for the blackhat to attack. Therefore, in most cases involving honeypots, entrapment is not an issue.

FUTURE OF HONEYPOTS

Mr. Lance spitzner who has played a major role in the development of honeypots has made certain predictions about the future of honeypots. They are as follows:

- *Government projects*: Currently honeypots are mainly used by organizations, to detect intruders within the organization as well as against external threats and to protect the organization. In future, honeypots will play a major role in the government projects, especially by the military, to gain information about the enemy, and those trying to get the government secrets.
- *Ease of use*: In future honeypots will most probably appear in prepackaged solutions, which will be easier to administer and maintain. People will be able to install and develop honeypots at home and without difficulty.
- *Closer integration*: Currently honeypots are used along with other technologies such as firewall, tripwire, IDS etc. As technologies are developing, in future honeypots will be used in closer integration with them. For example honeypots are being developed for WI-FI or wireless computers. However the development is still under research.
- *Specific purpose*: Already certain features such as honeytokens are under development to target honeypots only for a specific purpose. Eg: catching only those attempting credit card fraud etc.
- Honeypots will be used widely for expanding research applications in future.

CONCLUSION

This paper has given an in depth knowledge about honeypots and their contributions to the security community. A honeypot is just a tool. How one uses this tool is upto them.

Honeypots are in their infancy and new ideas and technologies will surface in the next time. At the same time as honeypots are getting more advanced, hackers will also develop methods to detect such systems. A regular arms race could start between the good guys and the blackhat community.

Let' s hope that such a technology will be used to restore the peace and prosperity of the world and not to give the world a devastating end.

REFERENCES

- Spitzner, Lance.

  "Honey pots Tracking Hackers". Addison-Wesley: Boston,2002

- Spitzner, Lance.

  "The value of Honey pots, Part Two: Honeypot Solutions and legal Issues"

  10 Nov. 2002

  <http://online.securityfocus.com/infocus/1498>

- Spitzner, Lance.

  "Know Your Enemy: Honey nets 18 Sep. 2002.

  <http://project.honeynet.org/papers/honeynet/>.

- "Honey pots-Turn the table on hackers" June 30,2003

  <www.itmanagement.earthweb.com/secu/article.php/143 6291>

- <www.tracking-hackers.com >

- Posted By: Brian Hatch

  Published By: *New Order*, 1/6/2003 11:36

  <www.linuxsecurity.com>