

A dissertation on

# Deception Techniques Using Honeypots

Prepared by:  
Amit D. Lakhani

Guided by:  
Dr. Kenneth G. Paterson

Information Security Group  
Royal Holloway, University of London  
UK

This dissertation is submitted to Royal Holloway, University of  
London as partial fulfillment for the degree of MSc in  
Information Security

# Acknowledgements

The dissertation on deception techniques using honeypots has given me a wealth of information about the topic and security as a whole. In this regard, I would like to extend my gratitude to all those who have knowingly or unknowingly helped me in preparation of this document.

Firstly, I would like to thank my advisor **Dr. Kenneth G. Paterson** for his ever-willing support and guidance. His comments to a myriad of issues proved to be a great help in preparing this document from start to finish. Also, I would like to thank **Information Security Group** at Royal Holloway, as a whole for giving me this opportunity to interact with him and the rest of faculty.

Secondly, I would like to thank all my sources at the **Honeynet Project** who helped me get up at critical junctures within the dissertation. Whether it be the IDS-honeypot mailing list at insecure.org or through direct emails, all the people were always helpful to guide me in proper direction. Special thanks to Dr. Lance Spitzner for adjoining my name at honeynet.org.

Thirdly, I would like to thank many individuals who have given me support in developing some innovative topics. Matthew Williamson from HP labs, Rakan al-khalil from Columbia University, Augusto Paes de Barros, Richard Salgado are only some names I can cite here. My heartfelt gratitude to all.

Lastly, I would be glad to admit that the dissertation has been a great learning experience and I would certainly look forward to future opportunities like this.

With sincere thanks,

Amit D. Lakhani

## Abstract

Honeypots are a versatile tool for a security practitioner. Of course, they are tools that are meant to be attacked or interacted with to gain more information about attackers, their motives and tools, but they have matured from just that narrow concept.

This dissertation will try to give an analysis of what growth has taken place in this field and how they have grown to cater for various needs within security. As a fundamental issue, the legal issues will be discussed and an attempt will be made to judge their relevance. The core of this dissertation will consist of various deception techniques that can be used using honeypots. Various innovative applications like mobile code throttlers will be cited and the reader will be encouraged to develop newer ideas in this field.

At the end, the conclusion will give a thorough insight into things that need to be kept in mind while deploying this tool as a third line of defence.

# Index

Acknowledgements . . . . .	i
Abstract . . . . .	ii
Motivation . . . . .	iii
<u>Chapter 1</u> : Introduction to honeypots and their types. . . . .	1
The Research View . . . . .	1
The Commercial View . . . . .	3
Comparison . . . . .	5
Honeypots . . . . .	7
Types of honeypots . . . . .	9
Production honeypots . . . . .	9
Research honeypots . . . . .	11
LaBrea . . . . .	12
Honeyd. . . . .	16
Specter for windows . . . . .	18
Mantrap or Symantec Decoy Server . . . . .	21
<u>Chapter 2</u> : Legal Issues in Honeypot Usage . . . . .	24
Entrapment . . . . .	26
Privacy . . . . .	32
Liability. . . . .	37
<u>Chapter 3</u> : Risk Mitigation in honeypot deployment . . . . .	41
Risk mitigation. . . . .	41
<u>Chapter 4</u> : Deception Techniques . . . . .	46
Common deployment strategies . . . . .	46
'Sacrificial Lamb' . . . . .	46
Deception ports . . . . .	47
'Proximity decoys'. . . . .	47
'Redirection Shield' . . . . .	48
'Minefield'. . . . .	49
Deception Techniques:	
1. Simple port listener . . . . .	51
2. As mobile code throttlers . . . . .	56
3. Honeypot Farms . . . . .	61
4. Random Servers . . . . .	67
5. Digital breadcrumbs . . . . .	69

Conclusion . . . . .	74
Bibliography . . . . .	iv
List of Figures . . . . .	v
List of Tables . . . . .	vi

## **Motivation**

ISO/ITU 7498-2 [7] defines Security as:

*The term 'security' is used in the sense of minimizing the vulnerabilities of assets and resources.*

Information Security, as the word says, has always meant securing assets and providing controls and procedures to resist damage or potential impact on the system(s) under consideration. This definition as is understood in security circles has various potential inferences and is typically understood in the defensive sense. Protect the network, protect the server, protect the logs, the list never ends. However, in today's world and systems where you never know where your network starts and ends; because of growing demands, applications and utilities like wireless LANs, remote sites, working from home, VPNs; this approach becomes a bit too cumbersome.

Although, classical security is defined in terms of prevention, detection and reaction [6], the last two terms are often neglected mainly because of legal hassles or for funding reasons. Although, there are well-known technologies like IDS, firewalls etc. to aid to detection and reaction, they themselves sustain from their own weaknesses.

In contrast, the attackers have so many advantages in their arsenal [57,41]:

**Element of surprise:** Attackers can always develop and attack with exploits at any time, day or place and launch them. Zero-day discovery needs the defenders to be on-guard 24/7 and this is not feasible for manually interfaced systems.

**Many-to-one vs one-to-many:** While attackers can focus on one system, the defenders have to focus on all their systems. Same goes for patches as well, while defenders have to patch all their systems and keep track of all the vulnerabilities, the attackers have a front-edge of just finding one unpatched system and exploiting it.

With these advantages in their hands, the meaning that security is hard to implement and why cannot it be attained 100%, becomes clear. With a radical change in philosophy of looking at Information Security enters 'honeypots'- the technology that started from a debatable stand to present scenario where they have matured as not only academic but also as a commercially viable solution to security.

This work is a feasibility analysis and study of honeypots and various techniques that can be used alongside honeypots. The implications of these analyses are narrated as we go and carry a heavy discussion and thinking on this varied tool we have in our hand, which when used according to the pre-defined security criteria can give fantastic results and allows us to meet our security objectives.

## Chapter 1

### Introduction to Honeypots and their types

In security scenarios, there have always been two potent players. The one that tries to destroy the systems and networks and the one that tries to protect them. These are formerly called the blackhats and whitehats respectively, but this is a radical view seen by personnel following a military setup. In this instance, we can not miss the stand of The HoneyNet Project initiated by Lance Spitzner[45], by and large it is the only proper research honeypot group we have today. And it is true to think to a great extent that it is a correct viewpoint to see security as a whole. But commercially, the logic may not work. At this very junction, let us see these two conflicting views. They both do need security in their own ways for protecting their resources but their approaches are different. Also, let us compare the ideas of these and see how the implementation of honeypots may or may not be viable to security industry and security research.

#### ***The Research view***

In the research setup the black hats are thought of as malicious attackers who want nothing but reputation, money, fame, secrets (military and commercial) etc. They use specific tools for these purposes and the sole aim of the research group is to learn about these tools and their implementation [17]. As is said in the military world, if you want to secure from your enemy you have to know your enemy. And it is true to think that in this open network, Internet, it is hard to know who your enemies are and what are they trying to accomplish.

Also, they divide the attackers into different categories based on their skill sets like –

- Novices
- Script kiddies and
- Skilled black hat.

The novices have either no or meagre knowledge of the tools they are going to use and the methodology they use may not be standard, which in most cases gets them into trouble. They have very basic to none know-how of the computer networks as a whole and try to learn it as they gain experience. But this learning curve usually ends when they either get caught due to careless scanning or canned software usage.

The script kiddies are similar but they have some basic knowledge of the system as a whole. They might know some internal contacts within the network they are going to attack or may have a network topology of the system. However, they do not develop their own tools and use the tools as-and-when available. They also use a tried-and-tested formula of scanning, reconnaissance, conquer and attack i.e. their attacks have a common standard pattern within them.

The skilled black hats are the victims; research honeypots are designed for. They develop their own tools based on the network they are going to attack or some common tools that they can share with each other. These are the most dangerous of attackers and sometimes might even get the system they attack into a lot of monetary loss. Also, if they even doubt being monitored they might leave no trace of their presence or destroy the whole system completely. Normally, these are disgruntled system administrators, experts or technical gurus.

However, it is believed that the first two categories are dangerous because they may strike any target that their tool can exploit and thus leaves an

element of surprise of the next victim. But they do not prosper because of their lack of dedication to attack on one particular target [35,17].

Once the enemy is categorised into a particular category the next step is to learn about which tools they are using and how they are using. For this, a sacrificial system – a honeypot - is often designed in it's most basic form and kept for being hacked. In the present scenario, it would not take longer to get attacked (15 seconds is a record for the fastest scan and 43 for getting attacked) and soon you can observe the very nature by which the black hats try and gain access to your system. If there seems like an overdose of the system getting too compromised the plug is pulled off and the attacker detached.

Thus, the requirements of the research group are quite simple – they want to learn more about the attackers, they want to know what tools and tactics they use and how they use it, they also might want to prosecute the hackers by providing a clear understanding of the legal issues.

### ***The Commercial View***

In the commercial view, the basic nature is to protect the assets. A common approach in here is performing a risk analysis for the whole system and establishing levels of significance to each asset. CRAMM (CCTA Risk Analysis and Management Method) is one such tool. Once the assets are established vulnerabilities are searched for in them and a thorough table of all the possible threat scenarios is presented. This asset, vulnerability assessment and threat analysis develops a residual risk present in the system, even when the safeguards are applied. It is then left to the owner to decide whether the risks are accepted and carried on or whether controls, policies, procedures are placed in order to mitigate them. But a common point to note in this whole methodology is the fact that there is no concern of the attacker.

The systems are analysed against known threats and vulnerabilities and not against individuals and growing threats or technological advances.

It is these residual risks that can either get the systems under consideration into being compromised. Also, if the risk analysis is not up-to-date the new vulnerabilities that have crept up cannot be dealt with and leaves a great margin of error for the systems. But all the measures listed for prevention of systems are adopted and a thorough management policy for implementing security is established. If the policy is right, the methodology is right, then there are very less chances that the systems might get attacked.

Also, it is of least importance to find the attacker(s) and prosecute them. This is because of following reasons:

- 1) *Inefficient use of man-power*: Rather than wasting their time and money in prosecuting a 15-year-old kid the system owners think it much more convenient to vest their man-power in designing a new risk analysis tool for their system.
- 2) *Complicating legal issues*: Due to legal issues associated and myriad of laws governing different state-country scenarios, the matter is just closed and forgotten. Also, there is a constant gap between technological advances and the catching up of legal community with these changes. So usually computer system misuse cases are by far the first ones to be ruled in a number of jurisdictions.
- 3) *Reputational damage*: Another point of concern is reputational damage. No system owner wants to loose it's customer base just because it was attacked and it exposed it out by going to law enforcement and found the attacker. Even at the end if the culprit is found, the customers' confidence on the company decreases. So when business ideas are

matched with security implications there has to be conflicts and these conflicts might allow a window of opportunity for the attackers.

- 4) *No returns for losses*: Unless the system owners can prove the absence of 'due care' by the amplifying network the attacker has used, and get the returns from system owners of that network, there is no way the end attacker will be able to pay the losses a company suffers due to the attack. For example, in February 2000 Yahoo!, eBay, Amazon etc websites were attacked by a 15-year-old Canadian by the nickname MafiaBoy and suffered at least \$ 3.1 billion of losses. Unless these firms can prosecute the middle network belonging to some other firms for downstream liability there is no way they can get this huge sum from a teenager.

Thus, from the above points it is clear how the commercial world reacts to security and what are their primary concerns regarding their assets. They basically want a good protection mechanism, a solid legal background if they want to prosecute, a rigid punishment guideline, and possibly returns for their losses if they do get attacked. If this is achieved they might be able to sacrifice the reputational damage they might suffer but it might improve as well by prosecuting the misfeasor.

### ***Comparison***

Observing the two views, it becomes clear that there are chances of both the views getting mixed but lets first reckon the merits and demerits of each of them.

#### Points for research view:

- It is of course good to know your enemy and see under what situations and how they use certain tools and thus give a simulated version of attackers attacking a target.

- New and unknown attacks can be visualised and possible exploits can be seen. The term zero-day discovery is apt for honeypots.
- Does not rely on known set of risks and threats.
- Educates and makes aware of various current threats growing within the black hat community. A recent trend of credit card frauds was exposed and a technical paper was produced by the HoneyNet Group for the same. [46]

#### Points against research view:

- Technically, the trade-off for research and time invested for this activity may not be a viable for a business scenario.
- More concerned about prosecuting attackers, a view not adopted by business firms. Thus, it looks more government or law enforcement oriented.
- Sits on the edge of what is right and what may not be right. Tracing an attacker to origin may include steps that might cross the line between what is right and what is not.
- High skill set is required for maintenance of the honeypots. Also, training for this may not be provided as there occurs no formal training due to changing attacks.
- It suffers from an imbalance – the higher the interactivity the greater the risk; the lower the risk, the shallower the data [3]

#### Points for commercial view:

- Adopting this view guarantees you are on the safer side of security. There is no concern that law can question you if you maintain suitable standards and work ethics. It ensures you are on the right side of the line.
- Can be called efficient use of manpower as you are not investing your man-hours in research, which may not reap any results at the end.

- Standard skill set required which is available or can be given suitable training.

Points against commercial view:

- If not totally, relies heavily on known vulnerabilities and attack signatures, so new attacks can not be identified and prevented.
- Depends on correctness of risk analysis methods and all threats being identified. Human-errors, software bugs (if automated) are source of errors that might give drastic consequences.
- Has no idea of who and where the attacker is operating from and what are his motives. Only prevention may not help you if you do not know which files or what assets the attacker is after.

Having seen the above list, it becomes clear that both views have their own merits and it depends solely on the individual or the company as a whole of which methodology they adopt. There are risks and dangers in both of them and the best practice is to formulate an approach one wants to take towards security and implement the view as required.

**Honeypots:**

Having seen the conflicting views and their merits, seeking a common solution to all the requirements can be a laborious task. In comes honeypots, a system that can be moulded to supplement any of the views as chosen. Also, as is seen later in types of honeypots there happen to be two broad types of honeypots - production and research. These essentially depict the two views seen earlier and implement their strategies.

Introduction to honeypots:

Honeypots as the name implies, is not necessarily a system full of 'honey' or sensitive information as is commonly thought. Although it would be true to

say that it is better to catch more flies with honey (sensitive information) than with vinegar (false information). However, the concept is often a misconception for many people within security environment. A Honeypot is [35]:

*An information system resource whose value lies in unauthorised or illicit use of that resource.*

This however means that a honeypot can be anything - a program sitting on a computer logging all the users who log into the system and by means they log into, just a dummy account on the system which when logged into generates an alarm, and to some very extent it could even not be a computer system but just a mouse trap inside the computer cabinet which when touched traps a intruder's hands. In this broad sense any resource can be a honeypot, which unravels it's existence just by unauthorised penetration or access to that resource.

However, in security circles it is often thought to be as a bait-and-capture system, which has limitless legal liabilities and is thought as a research subject only. Within this thesis it will become clear sooner or later how is this a misconception and will try to perform a thorough analysis of various issues related to this new technology.

Also, since there are various configurations of honeypots it is hard to define what a particular honeypot does and how far it can meet its objective. Once a decision is made to deploy a honeypot on the system, it's purpose and goals has to be clearly stated in the security policy and its scope tightened, since then only can all the requirements be met - whether it be legal, technical or privacy concerns.

**Types of honeypots:**

Having defined a basic definition of honeypot, it is time to see the various ways honeypots are used commercially and academically. Although there are various configurations of honeypots they can be broadly classified into two main types [41]:

- Production
- Research

**Production honeypots:**

Production honeypots are low-interaction honeypots which have little or no interaction with the attacker or intruder in context. Also, they have less value to security of production resources. They try to create as less a realistic environment as possible i.e. when they are deployed they not necessarily emulate the whole system as a whole but try to emulate as much as possible within certain time and value. Once deployed they serve very little purpose – they capture data. In essence they just act as a basic event log, with a potential difference that they are not meant to be interacted with. For example, if you want to monitor web-based attacks, you just emulate a basic web server like Apache and listen to port 80(usually HTTP) connections. Once this is done, all the connections that scan the honeypots for HTTP vulnerabilities will be logged.

Production honeypots are made for mainly this reason, they capture data and send it to administrators. How they utilise this data and what precautions they take is left on to them. This has so many advantages as compared to competing technologies like Intrusion detection systems and firewalls. A honeypot has no production value i.e. they do not act as servers and so are not meant to be interacted with. If any probe or access comes on it, it is most likely a malicious activity, unless there has been a misconfiguration by the administrator or someone has mistakenly accessed the wrong system. Nevertheless, *noise reduction* for malicious activity is the

best advantage of these types of honeypots. They indicate the administrator succinctly of what the attack or the probe is and how the intruder got access in. Thus, instead of browsing through 10,000 alerts from firewalls and IDS it makes the work of an administrator much easier. He can pinpoint the exact log of the attack without getting into the hassle of going through each and every record of the activity. Thus, the idea of less false positives giving efficient use of manpower gets practical here.

Also, there occur no advanced algorithms or databases of attack signatures to be kept for validating packets entering your network. This is not just for detection. In fact, production honeypots help widely in prevention. They can prevent worms from entering into the system, which work by scanning a complete network range and targeting specific vulnerabilities. An example of this technique will be presented later in one of the deception techniques in later chapters. Also, they might slow down the inbound connections directed towards the network via them. An example of this sort of honeypot is the LaBrea tarpit [17], which detects connection to non-existent IP addresses or overwhelming ARP requests to particular IP address (thus predicting it to be a DDoS attack). It then acknowledges the connection but keeps it incomplete and thus the intruder is 'stuck' by the honeypot. However, bandwidth requirements can decide the time for which the connection can be left hanging.

Also, production honeypots often are used to deceive people as legitimate servers. An intruder might think he/she is interacting with the real system while they are just attacking a honeypot. A recent example of this was cited in one of the large security firms – Internet Security Systems (ISS) [56]. According to the firm, one of the web-server that suffered a breach and got defaced was just a honeypot and was meant to get hacked. However, the article said the “X-force Internet Watch” was then properly monitored and the malware was removed.

But this all does not mean that production honeypot are flawless. Firstly, they can only monitor anything that is coming their way. They can not prevent anyone from opening confidential files from ports they are not listening, nor can they stop Trojan installs from ports they are not monitoring. Secondly, as stated earlier, it requires a high skill set for maintaining a honeypot and it is a risky business if it is used as a “launch pad” for attacking other systems connected within the same network as the honeypot.

#### Research honeypots:

Research honeypots are more complex than production honeypots and are kept in more secure environment since they do not comparatively have much valuable assets to protect in the backbone. However, they simulate the whole operating system and thus present the intruder with a known set of vulnerabilities within the system. For example, for web attacks a default installation of Linux 3.1 with Apache 1.1 can be installed and the results observed.

Since, research honeypots are a step ahead than production ones, they naturally get the backward compatibility and advantages. Thus, all the advantages of production honeypots are present in research ones. Also, they are more stringent in their deployment and can serve response tasks like trace-back. Security firms might be interested in finding new attack tools and trends and thus keep their eye on research honeypots. However, law enforcement agencies and government look more for early warnings and prediction from the analysis of research honeypots. These are just a few examples to cite for the significance of research honeypots to security circles. For example, a recent result from the HoneyNet Project [45,46] revealed a vast increase in organised credit card fraud. According to it a vast majority of stolen credit cards are used across relay channels thus increasing the illicit

use of credit cards, performing identity thefts, compromising merchant sites and exchanging of these numbers.

Also, commercially research honeypots firms are now including the Microsoft vision – providing honeypots as service rather than just providing support and installation at commercial sites. In a common configuration of honeypots called bait-and-switch [53] all traffic to the main server is routed to various honeypots worldwide and they depict the main server completely. Once honeypot-providing firms install the ‘switches’ or ‘re-routers’ at the commercial site, they just can log all the activity passing through their honeypots, depicting the main server. The customers wouldn’t know the difference as they think they are interacting with the main server. So commercially as well honeypot service could mount to a great profit margin in the near future.

However, research honeypots are still thought to be a subject of the nerds. But even the results displayed by these honeypots have a substantial value and can be used for tightening the security of your network.

Having seen the two broad types of honeypots, we can see the various common configurations honeypots can be used in. The following types are most common in recent times.

1) **LaBrea**[14]:

As the name, so the function. Rancho La Brea is an ancient site [47] located in Los Angeles and homes a large variety of fossils of large mammoths, fierce sabertoothed cats etc. These creatures were trapped there because of the presence of large ‘tar pits’. This exactly is the function of LaBrea the honeypot. It is also called ‘sticky honeypot’. The main purpose of LaBrea is to hold attackers for a pre-established amount of time, which could be infinite

as well. In this way, it is a low-interaction production honeypot with not many bells and whistles.

Working [48]:

LaBrea takes its idea from virtual machines, where software based machines are 'created' on demand and they serve themselves exactly like real machines. You can ping, traceroute; anything to them and they will respond as a normal machine would [49].

LaBrea works at the ARP (address resolution protocol) level – layer 1 and 2. When a machine wants to communicate with another machine, for a TCP communication, it needs a 48-bit MAC (media access control) address of the destination machine if it is on the same local area network. If the machine is to be searched by spanning a series of networks then it needs a 32-bit IP (Internet Protocol) address (128 bit for IPv6) in addition to the MAC address. The IP address is used to route the packet to the destination machine while the MAC address completes the transfer of the packet once it has reached the final router. Thus, a router is tasked with locating the MAC address of a particular machine on receiving a packet. A router has its own ARP cache to do this job. It contains the IP Address and the corresponding MAC address of the machines under the serving networks. If however, the router cache does not have the MAC address it asks the parent router, which owns the IP Address through an ARP request. These requests are of the form:

ARP Request:           Who has 132.219.145.23 tell 132.219.145.1

If the destination machine recognises its IP address it will respond by providing the corresponding MAC address of the LAN card. These are of the form:

```
ARP Reply:          reply 132.219.145.23 is at 0:0:0:ff:ff:ff
```

Also, the router is persistent in locating the MAC address corresponding to the IP Address and will continuously query the network for the same. It is this persistence that LaBrea uses to operate. Once LaBrea sees a lot of request for a particular IP address with no corresponding reply it will create a virtual machine and send a dummy MAC address displaying that it owns the IP address. Also, while responding it will insert the fake MAC address within the packet so that the transmitting device (router) can know how to get the message delivered to that MAC address. The fake MAC is inserted within the source address of the packet also, so that the transmitting device (router) will register this MAC address to the corresponding IP Address.

One subtle question arises here – how can LaBrea see the replies in a switched network as the replies will be unicast as compared to requests which are broadcast and thus can be seen-in-clear? To overcome this, LaBrea has a command line option `-s` which ensures the application of the transmitting device that it sends mirrored ARP replies for any ARP requests it sees. In the mirrored ARP reply, the LaBrea machine will send an identical ARP request of each ARP request it sees, with itself as the destination. Thus, if the machine exists, LaBrea will receive an ARP reply as well.

As aside, it is important to know the three-way handshake of a TCP connection. In the first step, the source machine will send a frame with the SYN flag set to the destination machine. It in turn will send a new frame with both the SYN and ACK flag set. In acknowledgement to this, the source machine will send a frame with ACK flag set. In steps 1 and 2 the Maximum segment size (MSS) is also specified to the destination machine. Also, a window size, telling the destination machine of how many bytes the source machine will receive before an acknowledgement must be sent is specified.

Assuming that the router sending the ARP request must have a frame with SYN flag set, LaBrea replies with a SYN/ACK flag set in the replying frame. Having completed its part for the 3-way handshake the source machine is now 'tarptitted'. This is done by specifying a very low value of MSS, so that only small segments of data can be exchanged between the virtual machine LaBrea created and the source machine. Also, a window size of 0(zero) is specified by LaBrea and this makes the source machine to a 'wait' state. The source machine constantly probes LaBrea for window sizes and thus is tarptitted to an infinite amount of time.

However, this tarpit state does result in traffic overhead but it is assumed to be quiet minimal (1215 bytes/hour). Also, this can be seen as a good compromise to various worm attacks to the network. Tarptitting does ensure you relief from worms and DoS attacks as it slows the attacking machines to a great level. Thus, it proves as a thorough defence to Code Red, SoBig etc.

To narrate and concise some of the features of LaBrea here are some bullet points:

- Tarptits a malicious connection and thus stops other machines to get infected.
- A tool for deception, obfuscation and deviation for the white-hat community.
- Easy to set up and configure
- Open source

Demerits:

- A small cost in the form of traffic overhead (1215 bytes /hour)
- Consult your lawyer before deploying. The author of LaBrea, Tom Liston, came to know about Illinois state law after deploying it and had to get the server shifted for deploying LaBrea.
- Only runs on Unix based systems and understands TCP and ICMP only.

- As its open source, support is not provided.

## 2) **Honeyd** [13]:

The second quite common configuration of honeypots is Honeyd also known as honeypot daemon. This is again an open source honeypot primarily designed for Unix systems but now has Windows compatibility too. It is developed and maintained by Neils Provos at the University of Michigan [16]. However, it has a lot of advantages over LaBrea, the first and foremost being the ease of configuration. The other features are summarised later.

Honeyd works similar to LaBrea in the sense that it monitors all the unused IP addresses and whenever there is a request for connection to these addresses it interacts with the source machine. However, certain features are vital to note. Firstly, you don't have to create any port listener or utility for ports you need to monitor. Honeyd has built in capabilities for this. It can listen on all TCP and UDP ports and can detect some ICMP activity as well. Thus, Honeyd is a low-interaction virtual honeypot that simulates TCP and UDP services.

### Working:

As said earlier, honeyd again uses the concept of virtual machines, just like LaBrea. The difference is that LaBrea always creates a new machine while honeyd is smart enough even to recognise services already started i.e. if a TCP connection is already established it can proxy the service offered by the real application. The working of honeyd is expressed in figure 1.1. When a packet request for one of the virtual honeypots arrives, the router has to be configured to direct it to the specific virtual honeypots. Virtual honeypots have the advantage that they don't require additional computer systems, but the adversaries have to be convinced that they are not visiting virtual network of honeypots [49]. For this, the whole TCP stack is to be formed within the virtual environment and this is effectively done by honeyd – another point where honeyd outweighs LaBrea. This creation of the whole

stack can easily fool scanning tools like *Xprobe* and *nmap*, which rely on fingerprinting techniques for recognising operating systems. In fact, honeyd uses the database derived from *nmap* to thwart the fingerprinting techniques.

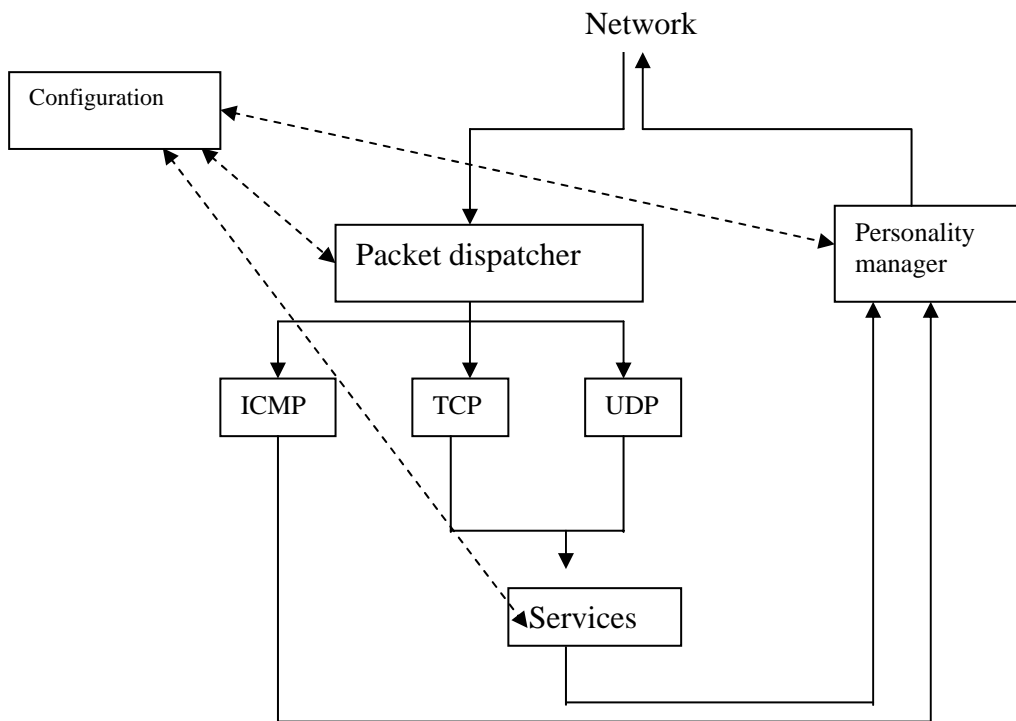


Fig. 1.2 Working of honeyd (Source: University of Michigan)

As can be seen, when the network sends a packet destined for one of the virtual honeypots of honeyd it is first processed by packet dispatcher. It checks the length and checksum of the packet and hands it to the corresponding protocol handlers. It can only recognise 3 protocols TCP, UDP and a fair bit of ICMP. The ICMP requests only get an ICMP\_ECHO reply message. For TCP and UDP, honeyd establishes connections to arbitrary services. It also maintains the 3-way handshakes of TCP connection but the congestion window is not supported, something LaBrea does. After this, the packets are sent to the personality engine, which adjusts the packet contents so that it looks like it has originated from the network in context.

Thus, honeyd is a low-interaction honeypot but displays a fair bit of smartness within its structure. This and the ease in configuration make it easy to suit low to medium level organisations.

Some of the features of honeyd:

- Open source software so completely free and is distributed under BSD license.
- Full maintenance and support is provided by Neils Provos and various other mailing lists at a nominal charge.
- Ease in configuration.
- Even fools active fingerprinting techniques as used by nmap and Xprobe by emulating services at stack level.
- Can monitor any TCP and UDP ports and entire networks.

However, the dark side reveals some important demerits as well:

- As it is a low-interaction honeypot, it cannot provide real operating solutions for adversaries to play with.
- No built-in support for alerts, nor mechanism for capturing extensive sessions.

### 3) **Specter for windows:**

Specter, as a honeypot, is just a league apart. It is one of those bells and whistled honeypot with all the rich features and ease of use but not much core capabilities. It is one of the first commercial products sold, developed and maintained by a Swiss company NetSec [27]. Also, another distinguishing feature of Specter is it is one of the few honeypots meant for Windows platforms. However, although it emulates various operating systems it is still a low-interaction production honeypot. So the basic goal that it serves is to protect your organisation from malicious activities and not gathering information about them. However, in comparison it suffers from

many weaknesses but also displays it's stand by counteracting weaknesses of other honeypots like honeyd and LaBrea.

As with most of the production honeypots, Specter is really easy to configure. It doesn't need any huge appliance and is a piece of software sitting on a machine emulating OSes. It just monitors the IP assigned to the computer it sits on, thus it's capabilities to monitor unused IP spaces do not materialise. This limits it's use of course, but it provides with all the necessary support and maintenance required over a span of time.

**Working [27]:**

The greatest asset for Specter is it's simplicity. It works on the common principle that any activity interacting with a honeypot is malicious. This makes it extremely valuable for use in internal LANs. Specter works on TCP services, whenever an attacker interacts with one of these services it logs all the activity and generates corresponding alerts. However, it cannot detect ICMP, UDP or any other non-standard IP protocols.

Consequently, it monitors only specific ports – 14 to be precise in Specter 6.0. Out of this, seven are traps and seven are services. Traps are nothing but port listeners, that log the activities interacting with them and generate alert. Also, they tear down the connection once logging is done or a certain threshold is crossed. Services interact with the attackers in the way that they emulate the application(s) on those ports.

The 7 traps and 7 services are tabulated below:

<b>Traps</b>	<b>Services</b>
DNS	FTP
IMAP	TELNET
SUN-RPC	SMTP
SSH	FINGER
SUB-7	HTTP
BOK2	NETBUS
Generic	POP3

Table 1.1 Table of services and traps provided by Specter (Source: NetSec Inc.)

Another feature of Specter is it provides you with number of options to emulate the services it offers. This is done by changing the service based on operating system you choose for e.g. HTTP service will be IIS web server for a windows platform. Also, Specter can emulate upto 14 different operating systems which are – Windows 98, Window NT, Windows 2000, Windows XP, Linux, Solaris, Tru64, NeXTStep, Irix, Unisys Unix, AIX, MacOS, MacOS X, FreeBSD. However a major drawback with this emulation is that it doesn't emulate it at the stack level unlike Honeyd. Thus, active fingerprinting tools like Xprobe and nmap can rattle the deception Specter creates.

Also, the behaviour of the services can be changed, for e.g. making HTTP 'strange' will leave the intruder wondering of what is happening. Without any doubt, Specter is also the most easily configured and deployed honeypot. It comes with a standard windows installer which does all the work for you and you are ready to go. Also, there are a myriad of features working their way in the main window and you and click on whatever services you want and which alerts to generate.

Features:

- Ease of use and configuration simplicity.

- Full support provided.
- Emulates 14 different operating systems.
- Incident management facility with ability to pinpoint on specific incident.
- Services can be configured to frighten, bewilder or lure the attacker.
- Supports major services.

Demerits:

- Only supports TCP connections.
- Though it emulates all the major operating systems, can be installed only on windows platforms.
- Monitors only IP assigned to host machine it sits on, thus no support for unused IP addresses.
- Does not emulate OSES at stack level and thus gives away its presence on scanning by active fingerprinting tools.
- Costs larger as compared to open source honeypots like honeyd, even extension of upgrade and support period is charged.

Having seen the major low-interaction honeypots, let us take a peek into one of the high-interaction honeypots.

#### 4) **Symantec Decoy Server** (formerly called **ManTrap**):

ManTrap[22] is a high-interaction honeypot with various features. But first we need to know how are high-interaction honeypots any different. The foremost difference for high interaction honeypots is they are real systems, nothing is emulated. The adversaries are provided with real operating systems and services and the act is observed. By giving this out, you can learn and gather huge information. You can find out about new rootkits and IRC channels as well as mechanisms by which malware is introduced to the system. Next, high interaction honeypots also make no assumption about hacker behaviours. This gives for zero-day detection of newer exploits and viruses and worms. But this comes at the cost of increased risk to compromise these honeypots. Obviously, high-interaction honeypots

becomes handy only to experts who have time and money to spent on research areas related to these activities like law enforcement, research etc. Considering ManTrap [22], it is a decoy system to divert the attention of adversaries to lesser value machines as compared to main servers. It has stealth mode monitoring and thus detects each and every keystroke given out by the attacker. Since it's a commercial product the inner working is not described but some key features are as follows:

- Since a honeypot is a decoy system interacting traffic has to be seen with suspicion. This is the basic principle of ManTrap and it detects unauthorised use and access by means of this.
- Similar to Specter, ManTrap also contains incident management feature and thus can report and log activities and enhance prioritisation efforts.
- Provides response mechanisms based on frequency analysis and shuts down machines by monitoring increased hacker activity.
- Provides stealth monitoring and thus live attack analysis.
- Detects both host and network based intrusions.
- Zero-day recognition of unknown exploits and attacks.
- Reduces false positives to a very large extent.

However, these silver linings don't come without the dark cloud. Some of the demerits are:

- Need highly skilled expertise to maintain and deploy these kinds of honeypots.
- Even with that, the risk involved for getting compromised remains and if these are connected to the production servers a thorough risk analysis has to be done.
- Although a commercial product, the sole aim of high-interaction honeypots is to gather information and not secure the organisation. ManTrap combines both these contradicting goals.

Having seen all these common types of honeypots and techniques used within we can concentrate on other topics related to honeypots in the next chapter. The first and foremost being legal issues associated with these systems, but before going on a thorough discussion on legal issues, it has to be kept in mind that honeypots are new stars on the horizon. It is a maturing technology. Interestingly, people, industries and businesses are going to hesitate before deploying these systems on their own networks but as with everything in security, it depends on what are you trying to achieve and what advantages you get by deploying competitive technologies. If you can afford to install firewalls, IDSes and can manage to go through 10,000 alerts per day; honeypots are not for you. Every technology is built to ease out some aspect of manual labour and honeypots do just that. It is not a magic solution but just a very important tool.

## Chapter 2

### Legal issues in Honeypot usage

Having discussed in detail about the definition and concept of honeypots in the previous chapter, its time we move on to the issues relating to honeypots and how they are addressed in both commercial as well as research environment.

Honeypots are a new technology and its so true to say that even when researchers and academicians are learning skills to operate them, its easy to believe that legal community can not cope up with the legal issues related to honeypots. However, as said, the concept is old, only the technique to apply them and the place where these are applied has changed. The use of baits to catch animals has historic to pre-historic applications, so it is obvious to think that there are laws that address this issue in variety of ways. But the domain they apply to is different in the case of honeypots. Also, honeypots themselves have a varied field of application and usually this field is defined by either a pre-defined security policy or the application in context, even both simultaneously sometimes. With these many implications it is hard to define legal boundaries for the 'free and open' usage of honeypots. Some of the reasons that can be classified here are [37]:

- *New technology*: As said, when even the people coining this term are in learning curve, the legal framework and its adjudicators are obviously going to take the case in as-and-when circumstances i.e. take it according to the context defined and explained to them.
- *Varied applications*: Honeypots have not only varied and debatable definitions but their application too range from a simple port scanner

to a virtual machine which is created on demand [48]. A common law, which could then be internationalised, is thus hard to achieve.

- *No legal cases:* As of yet, there hasn't been a legal case pertaining to honeypots and its usage, so there isn't any pre-established laws directly addressing this concept.
- *Concepts already legalised still debatable:* some issues relating to honeypots like entrapment, enticement etc. themselves have debatable rulings in different scenarios. For example, while in the case of *Sorrells v. United States* [34] the court ruled out the possibility of entrapment but in case of *Sherman v. United States* [33] it made the government responsible for entrapment.
- *Thin line between honeypot technique and unauthorised usage:* As this thesis further illustrates, there will be applications either by governmental organisations or obsessive aficionados of spy-work, to track the very nature of hacker activity and their source. This technique, though precious if used by authorised and administrative faculty, could have severe legal obligations. The so-called 'patriotic hacker' term applies to this scenario.

Through all these points it is hard to define a definitive legal framework that can address the soul purpose of honeypots. As with other maturing technology, legal issues for honeypots can only see daylight once cases pertaining specifically to this issue are tackled and ruled. In this way, the first honeypot legal cases have to think of themselves as trendsetters.

However, I would like to show the present scenario and the issues relating to honeypots which are relevant enough to grant some thought provoking discussion and debates.

The basic legal themes related to honeypots are [37]:

1. Entrapment (including enticement),
2. Privacy and
3. Downstream liability.

Following is the discussion on each of them. Also, since no court case has been judged pertaining to honeypots, we generally consider United States Law here, however, there is mention of corresponding international law within the discussion.

### **Entrapment:**

The issue of entrapment, as is commonly known, came to limelight quiet early in US courts and followed in UK as well as rest of the world.

#### *In the United States:*

In 1932, the *Sorrells vs United States* became the first federal court case that defined ‘entrapment’ in clear legal terms. According to it [34]:

*“Entrapment is the conception of planning of an offense by an officer, and his procurement of its commission by one who would not have perpetrated it except for trickery, persuasion or fraud of officers.”*

This is a landmark definition that stated the very significance of entrapment and became a major defence for culprits finding a legal loophole to escape and/or prosecute the law-abiding officers. The key concept in this definition that became significant later is ‘predisposition’. The very fact that the defendant ‘would not have perpetrated it except for the trickery, persuasion or fraud of officers’ encompasses a broad variety of concepts and terms. Would the attacker have committed the crime in the absence of encouragement activity by the officers? This concept of predisposition played major part in later cases of *Jacobson vs United States* [19] etc.

One important point to note here again is the fact that the prosecuting official should be law enforcement official or an agent of law enforcement. If you are not law enforcement official and do not wish to prosecute, entrapment is not a problem for you. Also, entrapment is a defence for the defendant, a honeypot operator does not need to think about entrapment. If he prosecutes someone, he just has to keep in mind that the defendant can take entrapment as a defence. To make his case stronger he will have to prove entrapment wasn't an issue.

Formerly however, there has come out two distinct tests to test the presence or absence of entrapment in criminal cases in the US. They are [30]:

*The subjective test:* was the defendant predisposed to commit the crime when the government official approached him?

*The objective test:* Did the government's encouragement of crime exceed acceptable limits?

The objective test gave rise to a new term not seen prior in legal history – enticement. This is discussed later.

Still further, there are exceptions in Federal Wiretap Act [44], which can be applied to some honeypot configurations. One exemption permits monitoring or interception of communication if one of the parties consents to it. The honeypots may display banner messages warning that use of the particular system is monitored. But most hackers don't penetrate the system through the front door, so if they have not seen the banner, they did not consent and we are back to the same dilemma.

Also, this exemption might apply without a banner if a court determines that the honeypot itself is one of the 'parties' of the communication. But if it is

used as a 'launch pad' to connect to other machines or set up as a chat system on the system, then this exemption doesn't work. These are again kinds of situations where we need an example case to sort what is legal and what is not.

Also, there are relevant exemptions in USA-PATRIOT Act, 2001 [59] but it only applies to cases where the government steps in to do the spying. The so-called 'computer trespasser exemption' allows the government to intercept the communications of a computer intruder at the invitation of the victim. If we consider that everyone coming into that honeypot is a trespasser, which is normally true, then this exemption may work when government is coming in to do the monitoring. But then it has to be relevant to the ongoing investigation.

Then there is one more exemption called the 'provider exemption' in which you may monitor your system for the purpose of protecting your property or services from attack. But even this would not apply to a system that's designed to be hacked. According to Richard Salgado [29], senior counsel for the Department of Justice's Computer Crime Unit "the very purpose of the honeypot is to be attacked, so its little odd to say that we are doing our monitoring of this computer to prevent it from being attacked."

#### In the United Kingdom and the English Law:

This was not the general scenario only in the United States. Based on rulings of Sorrells vs United States there were cases in UK as well. The best one that raised major discussion was Regina vs Loosely [31] case in House of Lords. Also, the case of Nottingham City Council vs Amin, the taxidriver, [31] has references to entrapment. However, in English law entrapment is not a substantive legal defence. Lord Steyn [31] paves a clear basis in English law in the R vs Latif case. According to it:

*“The court has the discretion: it has to perform a balancing exercise..... the judge must weigh in the balance the public interest in insuring that those that are charged of grave crimes should be tried and the competing public interest in not conveying the impression that court will adopt the approach that the end justifies the means”*

However, this is a heavy legal language and its implications can only be on case by case basis.

*In Canadian and Australian laws:* In Canada, a stay is ordered on the proceedings while in Australia in cases on entrapment evidence obtained by improper and unlawful conduct on the part of law enforcement officers are excluded on the grounds of public policy.

In all of the above discussion we have observed that there is not a clear distinction of how legal framework understands the term entrapment itself. If this is the case, complicating it with Computer misuse and hacking – as is the cases with honeypots - gives rise to an exponential set of problems on the part of prosecutors as well as the judges, majority of whom don't have a varied computing know-how.

Another term that circles in legal matters in honeypots is *enticement*. Though, lawyers and legal practitioners do not accept this as a legal issue, it certainly needs discussion. *Enticement is a process by which an intruder is lured to a sensitive area.* This may or may not contain authentic material. If he steals the material, he can be tracked. However, if prosecution is held on this basis, this tilts to the definition of entrapment and then there is no definite yes or no. In general sense, enticement is considered legal while entrapment is dealt with case to case basis. In other words, enticement is *considered* legal (with a pinch of salt) while entrapment is illegal.

Also, a major distinction between enticement and entrapment comes from the fact that enticement can be performed by non-government or non-law enforcement official as well. In fact, many practitioners do state and believe that activities in Clifford Stoll's book [42], Cheswick's report in Evening in Berferd [8] are enticement rather than entrapment.

I would like to cite example of a Canadian case [12] in this regard, the Wallace vs United Grain Grower's Ltd.(UGG). Wallace was a salesperson with his former company for more than 14 years. The Supreme Court of Canada ruled that Wallace was *enticed* to join UGG and told that he would have a secure job until retirement. As is the case with entrapment that it became prominent with regards to honeypots just by discussions and legal understanding by experts, enticement too can become a defence on the part of prosecutors in justifying the practice of honeypots on their networks. Thus, honeypot operators should keep enticement too in mind while pursuing a court case.

Observing all the above topics and their implications, it is clear that honeypot usage on your network is not without risks. It is better to deploy a legal -limitations-proof system once you have sought the necessary legal advice regarding laws related to your domain, country or network. Below is a checklist of points to be considered while considering entrapment issues:

- 1) Keep your honeypots as near to the production systems as possible. Making them embedded in same box can be the best solution to entrapment issue, since you can display banners on both the systems simultaneously. Also, its said that the more near the honeypot to the system, the less legal obligations it has to establish.
- 2) If you do not want to prosecute intruders, entrapment is not an issue you should think of, since it is one of the defence the acquitted will seek in a court trial.

The following table enlists the major differences between Entrapment and Enticement:

Sr. #	Entrapment	Enticement
1	It is a protection mechanism by a law-enforcement agent, practising which the victim does a fraud, but he/she would not have performed it if he wasn't predisposed by the official.	It is a process by which an intruder is lured to a pseudo or true sensitive area.
2	Considered a major legal issue while discussing honeypots	Has not been able to claim its stand as a major legal issue.
3	It's a defence that can be sought out by defendants while being acquitted of honeypot related fraud.	It's a tool for the prosecutors to justify their monitoring of communication by the defendant.
4	Numerous and prominent non-computer legal cases.	Various cases but haven't been prominent enough to grant discussions.
5	Cases defined the basic definition of entrapment and context it has to be used in.	Still not a legal definition or the context it has to be understood in.

Table 2.1 Differences between Entrapment and Entrapment

- 3) Keep in mind enticement is an issue in your favour if you want to prosecute your intruders. It gives you the right to lure them, in order to protect your systems. Once they cross the boundary by stealing or modifying or deleting any data, you have hearsay evidence.
- 4) If possible, try to make a law-enforcement officer do the monitoring for you. In this way, you will have lesser liability and more protection from

legal issues pertaining to this area. Some exemptions, as stated above, are more favourable to a law enforcement agent than to an over-zealous administrator.

- 5) Keep everything documented, from the time you touched your computer to the time you had a power outage in your locality.

## **Privacy**

Another major concern and the best legal issue related with usage of honeypots is privacy. But this is not only relevant to honeypots but to all intrusion detection systems, firewall logging etc. There are various situations and debates related to this issue. Following is analysis of these issues according to region they are concerned with:

*In United States of America [37]:*

The issue rises because in US law it is illegal to log or record data about an attacker, even if he is breaking into your honeypot. The attacker is then just considered an ordinary customer visiting your website or your system and satisfying the very purpose for which you installed it on the Internet. If you consider your system to be valuable the responsibility and risk lies on you and you need to secure it with suitable mechanisms. Also, another issue is logging of conversations. If an attacker uses your honeypot as a platform to chat, and discuss his ideas with his fellow-attackers logging their conversation can have severe liabilities on the part of honeypot operators.

The major chunks of legal debates related to privacy in USA have their roots from

- Electronic Communications Privacy Act [10] and
- The Federal Wiretap Act [44]

There is also the Pen Register and Trap and Trace Statute [28] but it hasn't seen much light in legal discussions related to honeypots. But the basis of all the disputes lies with the basic interpretation of the Fourth Amendment addressing individual privacy. According to Fourth Amendment [11]:

*The right of the people to be secure in their persons, houses, papers and effects, against unreasonable searches and seizures, shall not be violated, and no warrants shall issue, but upon probable cause, supported by Oath or affirmation, and particularly describing the place to be searched, and the persons or things to be seized.*

However, as can be interpreted this naturally protects individual privacy and it becomes a complex issue when electronic communications are ruled based on this. It is well known that email is protected by fourth amendment, as the basic technology driving email is similar to telephony, which is covered under Fourth Amendment [50]

An important issue that has come up with discussing the Fourth Amendment is the fact that in certain rulings it has been stated that the more 'open' the communication is the less privacy protection is provided under fourth amendment. This interpretation has great value for the honeypots because with this context and several others chatrooms, online bulletins etc. are not covered under Fourth amendment. Also, the monitoring is relevant if the users have no "reasonable expectation of privacy". Since, attackers can not enjoy any reasoned privacy; they are not protected under privacy rights by Fourth amendment. If this is the case then there is no harm for individual companies to log activities running on their honeypots. However, Fourth amendment is not the only legal liability a honeypot operator has to think about.

The Federal wiretap act is by far the most relevant and the most challenging legal issue while considering honeypots. The understated are some issues addressed under it for privacy:

***Logging:*** According to it, it is unlawful to intentionally intercept any wire, oral or electronic communication without prior court order etc. This necessarily includes email, chats everything that is considered electronic communication. Although, this prevents the intruder from getting logged, there are certain exemptions within this as discussed earlier. Under one called the Service provider protection it is legal to collect information on people, visitors (including website visitors) as long as that technology is used to protect your network or systems. Thus, these are exempted from privacy violations etc. If your honeypot's sole purpose is to protect your networks and is stated in a regulatory document aka security policy then it is exempted from privacy restrictions. However, it might not be enough to state to a court law whether this was the sole purpose of the honeypot, also since there hasn't been any court cases on honeypots whether this would work is still debatable.

For research honeypots, this is a major issue, as they can not necessarily state that their sole purpose is securing, as they are used to understand threats, attackers etc.

***Information gathered:*** Another issue is what type of data is being collected by the honeypot. According to federal wiretap act, the data providers have to be notified that their data is being collected. As discussed, banners can be solution to this, but no attacker would intrude from the front door. But not providing these banners then again may include the neglect of "due diligence" and thus another legal trap is set for you. Also, the data being collected should be reasonable enough. This means you can collect transactional data like destination or source IP address, destination or source phone numbers etc. However, the more content data like chat conversations, private

information like National Insurance numbers, SSN is collected more privacy restricted the issue becomes. Thus, while employing honeypots the users have to be notified that all the conversations they perform on that particular system is recorded. This is comparatively trivial if employees are considered, but non-trivial when intruders and attackers are brought into the picture.

**Consent:** Another exemption under this act protects privacy if one of the parties agrees to monitoring or logging of the content as discussed. Since, this is never so easy, this exemption comes to little help.

**Investigation relevancy:** under the computer trespasser exemption, an owner of the system under attack can call a law enforcement agent to monitor on his behalf. However, for this to be true, the monitoring has to be relevant to the investigation and it has to be proved.

Another matter intriguing with the concept of privacy is the Electronic Communications privacy act [10]. The Title I a.k.a. 18 USC 2510 – 2521, which amends the federal wiretap act, deals with intentional interception of communications while Title II a.k.a. 18 USC 2701-2711 deals with intentional access without authorisation to stored communications. We have discussed a lot about Title I and so we move our focus to Title II or unauthorised interception of stored communications.

Much of the discussions of interception of communications apply here as well. A person needs to be properly authorised for accessing that stored communication. However, exceptions apply if there is consent by the users of the system, or if the provider of the system allows access to the stored communication. Also, exceptions occur for government agents and the service provider may keep a back up copy for maintaining his current business position. Thus, to abide by ECPA operators just have to either

observe strict consent from its users or have authorised access to the stored communication.

*In EU and UK:*

In European union, privacy issues are advocated based on:

- 1) **Directive 97/66/EC** – Article 5(1 and II) [9]
- 2) **Regulation of Investigatory powers act, 2000** [32] (only for UK)

According to Directive 97/66/EC article 5(I) member states pertaining to EU shall make sure that they preserve the confidentiality of communication both network and public telecommunications. Thus, this relates to preservation of public communications. On the contrary, article 5(II) states that the above shall not affect any legally authorised recording of communications whether it be private or public. Thus is a duel of ECPA for EU states and gives that monitoring powers if authorised.

Under RIPA, chapter 23, Section 1 a thorough legal description of unlawful interception is provided. It is a bit similar to Federal wiretap statute and states it an offence to intercept transmissions over ‘private telecommunication system’, unless with consent of system controller. This is same as the Service provider exemption of Federal wiretap statute. Also, it encompasses unauthorised access to stored communications and thus reflects excerpts from ECPA Title II.

For UK however, there is also the Lawful business practice regulations (2000/2699)(under RIPA) under which the authorised purposes of monitoring communications and records is enlisted. But there are restrictions in the form that monitoring has to be for the sole purpose as described and not for any other functions, and to perform all reasonable efforts to inform all the entities that use the system under consideration.

Thus, for privacy the following points can be noted worldwide for honeypot operators to abide by the most important privacy regulations:

- 1) If possible get the consent of all the users within the system, or who are using the system. Steps may include use of banners and establishing a clear security policy stating monitoring of communications. This also serves the purpose of “due diligence”.
- 2) The information being gathered has to be protected and should be taken care of being not exposed to unauthorised parties, thus serving “due care”. Also, whenever exposing the materials to law enforcement officers make sure they have proper court orders.
- 3) Make sure the honeypot is taken care of, an unattended honeypot may become a privacy issue with all sorts of matter – pirated software installations, illegal files, pornography, logging of private conversations etc.
- 4) Also, discuss the privacy issues with your local solicitor before deploying a honeypot and tell him precisely the purpose(s) of your honeypot so that he can decide what laws are applicable in your province.

### **Liability:**

The next major issue in deployment of honeypot is potential liability for the owner. Commercially, this is the most sought out legal issue to sell the idea of honeypot as a technology. Once this is digested by commercial market, there may not be any end to the development honeypots can bring to information security.

The concept is called downstream liability, which is defined as [58]:

*The requirement of the actor to confirm to certain standard of conduct, for the protection of others, against reasonable risks.*

According to CERT it includes duty, breach, causation and damages. For the sake of this discussion, we will just adopt the above definition and carry on the discussing basic nature of liability.

There are a certain amount of cases wherein downstream liability has played a major role. Although, this issue hasn't come out yet on the security sides, there are bells on whistles for it and in no time the first case being ruled that a queue of arguments and discussions are likely to follow. It is so very logical to look that a company facing a large denial-of-service attack will focus on prosecuting the zombied terminals of a multi-national company for negligence rather than a poor 15-year-old boy sitting at 3 am in his bedroom.

The same issues arise with usage of honeypots. If you developing and deploying honeypots on your network it is your duty to take "due care" that they don't expose inadvertent loopholes by which other systems can be thrown at risk. There has been heating debate on these issues even in HoneyNet project and they have taken this largely into consideration. The usual solution to this problem has been to lessen as much outbound connections from honeypots as possible. For example, in a typical setup of a honeypot the firewall prior to the honeypot is configured so that it allows maximum of 10 connections outbound. According to Lance Spitzner "increasing your outbound connections will give you greater chance of learning more about the black hats but it increases your liability exponentially".

If this is the case with research honeypots, production honeypots should not attempt to increase their outbound connections and be negligent. But still as a defence, there is considerable foresight in this issue. Although it has become a matter of fear to deploy honeypots due to liability issues, there hasn't been a case by compromised system owners suing other companies for negligence even on vast scale denial of service attacks. For example, in February 2000 a 15-year-old teenager pseudo named MafiaBoy brought down various well-known sites like Yahoo!, e-bay, Amazon etc. but these companies never held cases against owners of zombied terminals which he used to launch his attacks.

As such, every new technology brings with it risks and honeypots are no different. Even with systems like IDS and firewalls there are liability issues but they still sell the concept of security because they are thought to be in defence side of the line. But being on the offensive side it is hyped that honeypots are bringing unprecedented liability to system owners – a dictum, which can be tested only when cases based on it are ruled.

However as a matter of care following points need to be kept in mind and practised:

- 1) Keep at par with peer organisations in security practices. This can either be done by assuring an independent audit or more formerly an accreditation process for security, or by following some standard code of practice like BS 7799.
- 2) As with most other things - patch your system as often as you can or as often as is stated in the security policy. Read your logs and keep the updates on them documented. In a legal trial that will serve as a major defence.
- 3) Perform audits for your practices and policies. This may be either independent or internal but it will prove as a legal document for overall implementation of security. This also includes what security measures

have you taken to protect your honeypots from corrupting other production networks.

- 4) Also, keep record of improvements you did for earlier breaches and if possible what improvements and patches you adopted. This has perfect relevance to honeypots as they often get breached and so have to be taken care of.
- 5) Most importantly, keep a security policy and revise it time-to-time to keep it at par with varied regulations and practices.

As can be seen in all of the above legal issues related to honeypots that there is no definite answer. The reasons for this blurry scenario are just inexperience in handling honeypot-related cases and related contexts. However, as time progresses new ideas and technologies might bring with them solutions to this myriad of problems and intriguing legislations. Till then the best thing to do while adopting this technology is to keep as much less space between getting into legal troubles and avoiding them, as possible, by practicing best practices and industry standards. As they say - Contact your lawyer, after all that's what they are paid for.

## Chapter 3

### Risk Mitigation in Honeypot deployment

In this chapter, we understand the basic risk mitigation techniques to be kept in mind while deploying honeypots on networks and systems.

#### Risk mitigation:

As is said at the conclusion of the last chapter that honeypots are a new technology and there are and will be risks involved in adapting it to any network. However, it is important to know that for what and how you are going to use the honeypots in the environment under context. Is it a law enforcement network then what are assets being protected or whether they need to be protected? Is it a banking environment where monetary losses are critical for business continuity? Once this goal is decided risks involved with honeypots can be properly addressed.

Having seen the working and introduction to honeypots we can categorise the risks involved with using and maintaining a honeypot in a network. Let us consider a very simple network as in Fig 3.1

In fig. 3.1, the honeypot is just assumed to emulate the corporate web server. Usually, in a commercial setup the honeypots will either be kept outside the corporate firewall or a separate firewall (also called a Honeywall gateway) might be placed between the honeypot and the production LANs. It is also possible that the honeypot or honeynet is a completely separate network away from production LAN. With this in hand, let us perform a small risk analysis.

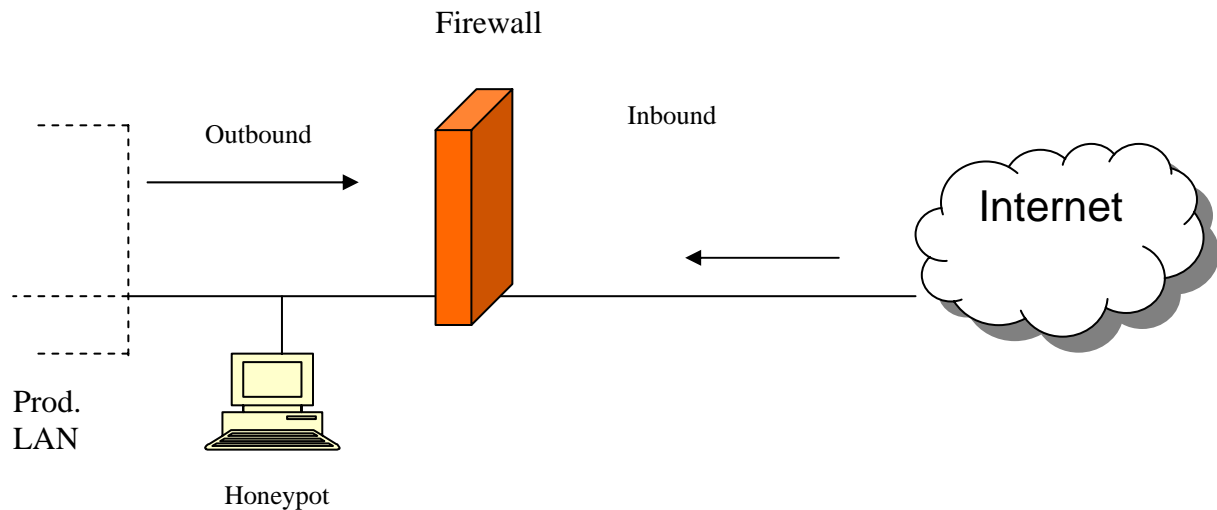


Fig 3.1 Example network for risk analysis

Asset(s): Production LAN, router

Threats: malicious worms or malware, Viruses, disclosure of corporate secrets from production LANs, Trojans, DoS attacks, system failures, physical attacks.

Vulnerabilities: Firewall misconfiguration, honeypot compromise, disgruntled insiders etc.

Risks: honeypot compromise and corresponding attack on production LANs, honeypot used as a launch pad for attacking other vulnerable systems on the internet (downstream liability) viruses, active content in inbound traffic, spyware in inbound traffic etc.

Once we have categorised these risks it is on the shoulders of either the security administrator (if the policy is formed) or on the owner of the system to decide which risks are acceptable and which need mitigation. However, just considering the honeypot issue, we have to make sure the introduction of honeypots does not issue more risks to the bundle of risks we already have. Even for such a trivial network, honeypots are supposed to issue more risks, so it has to be kept in mind that honeypots need management. Essentially, honeypots are meant to be 'used' in order to function, as their definitions say

in chapter 1. Thus, the moral we get just from this small example is proper management is critical for honeypot deployment and functioning. In fact, providing a honeypot and not managing it might be thought as negligence of 'due care' and might lead to legal hassles.

Once we have seen this, we can progress to see how can risk mitigation be done. Below are some points that narrate risk mitigation for the use of honeypots [60].

- Firstly, security policies should address honeypots if they are deployed on the network. The basic aim of policy towards honeypots should be to tighten the scope of it's use. It should be treated as another security logging device.
- Consider the level of interaction with the honeypot. If it is a low-interaction honeypot then the risks involved are low since little functionality is offered. However, if it's a high interaction honeypot then it needs more maintenance care and expertise.
- There is also downstream liability to be taken care of. Make sure your honeypot is not used as a launch pad for mounting attacks on other systems outside your own network. For this firewalls have to be made stringent or IDS sensors have to be deployed for outbound traffic from the honeypot.
- Other security devices need to complement the role of honeypots as well. Like firewall should be properly configured for inbound and outbound traffic.
- Secure the operating system the honeypot resides on. Apply up-to-date patches for it if its vulnerabilities are not completely known.
- For high interaction honeypots deploy data control strategies as the attacker can go out of bounds.
- Decide on whether you want your honeypot to be fingerprinted. More then often, you will want your honeypot to be exposed and used.

- Include honeypot deployment even in your business continuity and disaster recovery planning.
- At the very end, also consider the legal aspects. The next chapter gives a thorough insight of this.

## Chapter 4

### Deception Techniques

In this chapter we go into some great techniques and configurations honeypots can be used in order to complement their usage. These differ widely from the aspects we have seen in earlier chapters, especially first chapter. Also, these are related both to the commercial as well as research areas and can be improvised to serve for both the needs. As again, it depends on how and where you use these techniques to give out their true colour. Also, the examples of honeypots presented in Chapter 1 are just adoption of one of these configurations and implementing that novel idea. However, several non-deployed ideas still exist and give out a growth potential for businesses.

In various discussions and papers several deployment strategies of honeypots have been presented and studied. Though an analysis has been made in many of the papers, these have not been deployed in any domains. This again may be because of grey areas surrounding honeypots legal issues. But once these get settled, there will only be some not to implement these technologies in near future, especially in sensitive sectors like defense, law enforcement, nuclear strategies, banking etc.

#### **Deployment strategies**

Some of the common deployment strategies are [57]:

1) 'Sacrificial Lamb':

As the name says, these systems are just placed on the network so that they can be compromised. They have no connections to the production network and just act as perfect dummy services. The idea behind this strategy is to

quench the thirst of the attackers. In simple terms, give the attackers what they want, and let them play with it. These techniques necessarily developed from Clifford Stoll's publication of his encounter with a German hacker [42]. Although his idea was just to stall the hacker so that he can track him to his root. These systems just sit there on entry points and serve with no production value. Even data gathered within it may not be used by administrators to prevent future attacks. They just give a level of deterrence to the attackers and might buy additional time for administrators to act on. Some of the examples of these types of strategies are deception tool kit (DTK) by Fred Cohen [15] and Specter for Windows [27].

### 2) Deception ports on production systems:

Examples of these are already cited in earlier chapters. These are basically low-interaction honeypots that mimic various services on different ports. For example, HTTP is mimicked on port 80, SMTP on 25 etc. Basically these honeypots first 'observe' the operating system they reside on and then portray these services according to that. Honeyd is a common example of these sorts of honeypot. Also, specter is a feature-rich edition to this kind of honeypots strategy. The basic idea is deception so that the adversaries are just 'stuck-up' in solving the deception while they can either be knocked down from the network or suitable measures like trace-back, forensics can be taken. Also, various home made honeypots use this technique, as this seems to be the most common and less-liability-shared strategy to adopt.

### 3) 'Proximity Decoys'

For legal reasons this strategy is supposed to be the most effective and less troublesome of all. According to Richard Salgado, Chief solicitor for Department of Justice, USA:

"The closer the honeypot is to the production server, the less likely that it's going to have some of the legal issues that we're talking about, because the

monitoring becomes part of the normal process of protecting the production machine”

And this is true logically as well. Once the honeypot is part of the same subnet the main servers are included in it becomes part of your own network and you are allowed to *monitor* activities pertaining to your network. Also, once they are in proximity to other production systems you have ease in either re-routing traffic once some malicious attack is detected on the production systems, or trapping that attack. This helps in non-proliferation of worms, viruses as well. Examples of these types of honeypot include recent study and deployment of virtual honeypots using VMware [52] and User mode linux [51].

#### 4) Redirection shield:

These acts as means of deterrence, but in the near future the most developing aspect of honeypots attracting commercial use is this strategy. This is because it can be extended to provide commercial service to major networks. In this deployment by using port redirection or re-routing the traffic, honeypots can be said as acting in place of production systems. More precisely, it can be said that honeypots are just on the network to protect the production servers in case of attack. Thus, it can be legally argued that honeypots are just a layer of defence in order to protect the production systems. Also commercially, if rerouting switches are installed on client sites, honeypots while sitting at any remote system across the world can serve as services instead of just a device. Once this is done the client can be charged either based on attacks – which in any open huge corporate network would be enormous, or based on time length – contract basis. This will give out a tremendous profit margin as well because today we don't need to invite attackers, there are enough bees searching for honey.

5) 'Minefield':

Even this technique is not new in security. Here, honeypots are placed just at the perimeter so that any scans or vulnerability detectors can just exploit the contents of honeypots, sparing the production servers. Also, once attacks or scans are recognized suitable alerts can be raised in order to mitigate them. Thus, honeypots just act as third layer of defence in these types of deployment. Also, this does not mean singular honeypots but even multiple honeypots if deployed can serve as means to trap, deceive, trace, tear down, or tar pit the attackers. Commercially, these kinds of strategies may prove quite valuable. Examples of these strategies are LaBrea and Honeyd which when used in stealth mode can just provide basic services and contain the attackers within themselves. Also, Mantrap is deployed mostly in this strategy.

The table below presents the various commercial honeypots in market today and strategies they employ. The examples presented in chapter 1 also come under either one or the other strategy just discussed, but commercial honeypots are feature-rich and stable in configurations [57].

Sr. #	Honeypot name	Vendor	Strategy used	Description
1	Backofficer friendly	NFR security	Deception ports	Simulates windows OS and Back Orifice server. Responds and logs accordingly.
2	CyberCop Sting	Network Associates / PGP	Deception ports / services deception	Simulates entire network segment, fools fingerprinting tools, simulates lots of OSes. Logs and responds.
3	Deception toolkit	Fred Cohen and associates	Deception ports	Listens to requests on ports normally blocked and respond to them. Logging is extensive.
4	NetFacade	GTE federal network systems	Deception ports / virtual machines	Simulates CISCO IOS, Unix and windows services to mimic the real services

5	VMware	VMware Inc.	Virtual machines / proximity decoys / shield	Honeypot OS executing within the hostOS. Kernel level security provided especially for Linux.
6	ManTrap / Symantec Decoy server	Symantec	Virtual machines / shield / minefield	Runs a complete Unix Solaris OS in a 'jail' configuration with no emulation. High-interaction honeypot. Virtual hosts provided in DoS attack.

Table 4.1 Commercial honeypots and their deployment strategies (Source: [57])

From the above table it can be observed that most of the commercial honeypots are based on either deception ports or virtual machines. The logic behind this is clear that since they are production honeypots, they have to be low-interaction honeypots. Although examples like ManTrap are a bit of exception, their acceptance in market needs to be seen. Interestingly, it inevitably becomes clear that there is a need for more techniques both in commercial as well as research side. Even research honeypots only address the basic concern of gathering more information about attackers. As this does not include trace back, court admissible evidences, financial loss mitigation we are in need of constant in flow of ideas for improving honeypot technology so that it's acceptance and need become distinct.

In lieu of the above discussion, here are some more techniques to mitigate and compensate various other areas that can be addressed by use of honeypots. Although the techniques presented here have their roots in above strategies, the usage they are put to is different and innovative.

Long before computers were invented deception served a primary means to protect information. For example, in World War 2, the series of upcoming landings was protected by a number of deceptions ranging from convincing Hitler then the invasion was taking place elsewhere to attack on Pearl harbour by coded information. In fact, it will be hard to find areas in security

where deception is not used in some form to protect information [15]. The entire area of steganography is devoted to concealing information by deception. Cryptography as a whole is transforming information and protecting by converting to unusable form. This is in fact considering that in effect when we conceal presence of information we are deceiving. But then its important to note that while hardly 14 out of 140 defensive techniques are deceiving in nature, almost half of the attack techniques are deceiving in nature. This makes it intriguing to think that while the attackers have a whole arsenal of deception techniques in their quiver, the defenders are hardly rendered a few to protect their assets. However, an attempt has been made here just to attract the defensive side of deception and attempt further honeypot strategies. We will make use of common deception approaches namely, camouflage, concealment, false and planted information, ruses, displays and deterrence.

### **Deception technique 1: Simple Port Listener**

#### Description:

In this low interaction configuration, the honeypot is just being set up to listen to various port activities and raise appropriate alerts once certain threshold is exceeded. For example, if from an open port an attacker is able to gain access to the root shell a suitable alarm/email is delivered to the system administrator. This seems similar a lot to various logging capabilities built-in in many operating systems, but this basic functionality adds to the fact that honeypots are passive devices and are not supposed to be interacted with. If however, there is access to it, as their definition suggests they have to blow their bells and whistles. Also, another thing that has to be kept in mind is the logs regarding these port activities have to be preserved in systems other than honeypot because 90% of the attacks are directed to gaining root access and once that is done an apt attacker can easily erase his records.

Thus, the overall idea can be described just in three words listen, log and alert.

### Working model:

In every deception technique we might have to consider a network and analyse our technique within it. As a start thus, we set up an example network so that all the future techniques can be analysed and visualised on it.

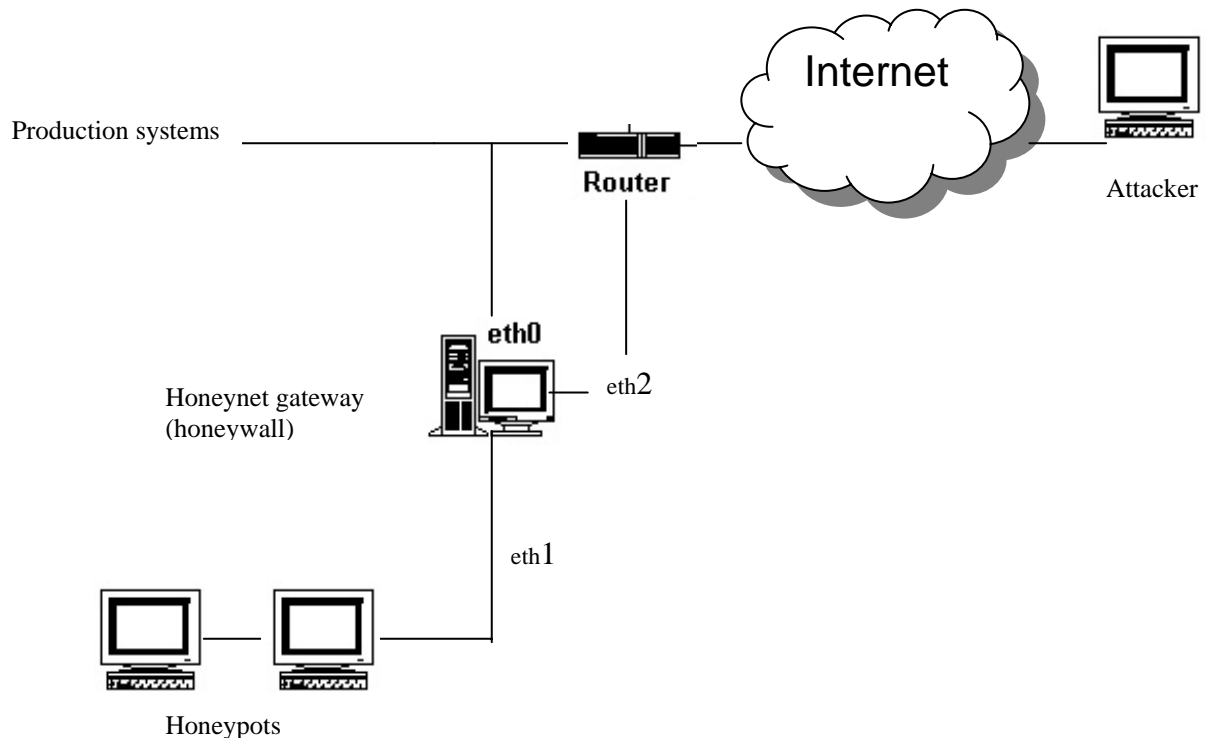


Fig 4.1 Example network for deception techniques (Source: Honeywall.org)

In the above diagram, the main part of the whole network is the honeywall gateway also called a honeywall. The attacker sits somewhere on the Internet and mounts his scans and attacks. The final router routes traffic to our administered network. The honeywall acts as a bridge and the basic feature it should have is it should not be detected, since then the existence of honeypots will be revealed. The gateway is configured as a bridge with three

interfaces. The first external interface eth0 is our connection to the production systems. The second internal interface eth1 is connected to a network of honeypot(s) and a third interface eth2 is just for administrative purposes. It should be noted that while the IP subnets of eth0 and eth1 could be same (in our instance we will take them as same) eth2 should have a complete new subnet i.e. it should be considered as a completely different network altogether. The honeywall thus will be a centralised management console for control of data to and from the honeypots. Also, as it is a bridge and externally connected to the production systems, the honeywall will serve to tear down the connections if they try to access production systems. eth2 will serve as administrative interface either to remotely configure the gateway or establish logging capabilities. Also, all the logs for the honeypots should be collected within the subnet of eth2 so that they can be protected from illegal erasures once the honeypots are compromised. Another purpose the eth2 interface will serve is to establish a pre-defined amount of outbound connections from the honeynet. This is because, once the honeypots are compromised every attempt will be done to gain more tools from other systems so that further attacks can be launched. Also, it can be used as a 'chat' system to communicate ideas between attackers and gain tools. For our purposes we will consider:

```
Eth0: 134.219.50.0/24
```

```
Eth1: 134.219.50.0/16
```

```
Eth2: 10.1.1.1/24
```

Thus, if we set up this network so that honeypots within it act like a port listener only, then the honeypots in eth1 just have some common port listening services like netcat [26], nmap [61] etc. They just log traffic coming to monitored ports to log files and enunciate alarms once a certain threshold like bandwidth usage, command shell capture is reached. Since all the traffic is already being logged by the honeypot gateway suitable measures to track

attackers can also then be established. The idea can be further novelised by use of other services like tar pitting the connection as used in LaBrea and thus can be used to prevent proliferation of worms.

#### Discussion:

As this is a low interaction honeypot, it doesn't have much production value. However, it can give excellent results as scans, reconnaissance are widespread across the Internet. Also, it is a commercially viable solution as there is no crossing-the-line activities. Functionally of course it is limited, but that has to be a compromise since most of the defensive deception techniques are not feature-rich, especially if we consider detection.

Similar examples: LaBrea, Honeyd, Specter etc.

#### Case study:

Below is a case study for a netcat honeypot [1]. As we know netcat is quite an interesting utility that utilises the basic TCP/IP services to log and write to ports within networks. However, netcat acts at network level and not at IP level, as a result it will not detect half scans and so cannot log active fingerprinting tools like nmap, xprobe etc. Since, most of the attack have a scan, gather information, find vulnerability, attack cycle using netcat at the scan stage helps in preventing the attack to a great extent. The alerting capability is not a part of netcat however and is discussed later.

The following is the code needed to get a honeypot working on a Linux system. In this instance, the author has made a centralised approach by making script files for each port and logging these files on a separate network or directory. We use the following netcat options:

```
-l          regular listen mode
-p          port(s) you want to monitor
-vv        double verbose mode
x.x.x.x    IP address of the host the honeypot resides on
```

After setting it up on various ports, we echo comments or remarks on the audit files and end it all up within a continuous 'do' loop. Below is the contents of the port25 - SMTP file:

```
1   while true; do
2   /usr/bin/nc -l -p 25 -vv xxx.xxx.xxx.xxx 2>> /var/audit/nc-port25
3   date >> /var/audit/nc-port25 echo "***** FAILED SENDMAIL
      ATTEMPT - PORT 25 *****\n" >> /var/audit/nc-port25
4   cat /var/audit/nc-port25 >> /var/audit/nc-log
5   cp /dev/null /var/audit/nc-port25 done
```

The first line begins the loop and the second line starts the netcat (nc) utility for port 25. The output is redirected to a 'nc-port25' file. The comments are echoed within the file and the original file is copied to another 'nc-log' file and deleted afterwards. This file just acts for port25 and you can write scripts like this for various other ports. Once that is done, a file 'portwrap' was scripted just to start all the ports at one place. This 'portwrap' can then be executed at each start up run and your honeypot is ready to serve at the ports specified.

Portwrap file:

```
/var/audit/nc-wrappers/port25 &
/var/audit/nc-wrappers/port21 &
/var/audit/nc-wrappers/port80 &
.....
```

However, the author didn't put any alerting capabilities within this system and thus, it just acts as a listener rather than providing any alerts. This can easily be done by using simple search utilities on the audit files. For example, if we know common exploits on ssh like sshnuke.c or Blaster.exe searching for them in the audit files can give you the message that the worm or exploits have been entered in your honeypot. For example the following will be a good way to start:

```
grep "MSBlast.exe" /var/audit/nc-wrappers/portxx
```

This might reveal the presence of the Blaster worm that hit the Internet recently. Also, knowing signatures of common attacks might yield unprecedented results on this simple honeypot.

### **Deception Technique 2: Honeypots as mobile code throttlers**

This technique is highly based on Matthew Williamson's work at HP labs [54], Bristol and behaviour blocking techniques prescribed by Messmer [23]. I would be glad to admit that in one of those introductory sessions at HP labs this idea was delivered to me and since I was thinking about my thesis related to honeypots, this relation crept up. I am not sure work like this is available in the security arena but this looks like a good prospect for controlling mobile code on-the-wild using honeypots. Interestingly, mobile code in here is pragmatically defined as programs, which transfer from system to system without little or no human intervention.

#### Description:

The foundation of this technique is based on the fact that an infected machine makes a lot more connections to other machines as compared to a normal machine. Using this approach to throttle the spread of mobile code like viruses, worms etc. is a comparative innovation. The paradigm shift comes from the fact that current approaches reside on virus signatures to quarantine infected machines. This doesn't yield many benefits, as this is more case-to-case basis to identify and vaccinate infected systems. But throttling has distinct advantages, namely it is more benign in approach and doesn't harm or misconfigure the system in any way. Also, it is based on network behaviour of mobile code rather than signature based approaches.

Lastly, it is unique, as it depends not on mobile code entering the system but more on *leaving* the system – something that is not recognised previously. Thinking honeypots, this is a great technique to adopt. Also, in one of the honeypot deployments called ‘minefield’ there are different honeypots serving different services on the same network. They are randomised to provide services to incoming requests and only the probability factor determines whether you get a real machine or a honeypot while requesting a service. An example of this is ManTrap configuration [43]. Once there are several servers serving as honeypots within a network the virus throttler can be installed in each of them. This will prevent the overall spreading of mobile code and agents within the network to a comparable level. Also, the network traffic jam due to the proliferation of such mobile codes will be prevented to an alarming level.

#### Working model:

Only for this technique we will have to go out of bounds from our network considered in Fig 4.1 This is because deploying this technique needs to install virtual honeypots on most of production systems in order to be more effective. The reason behind this will become clear as we move on. The basic need is to design a filter so that all the traffic passing out of a system is monitored. Thus, we need to device a honeypot within the network layer so that all traffic is monitored. Since we have seen TCP handshake protocol, we can state that whenever a system tries to make connection to another it is bound to send SYN package to the destination. So if we are able to count these SYN packets and limit the rate during ‘infection’ we have accomplished our task. However, in TCP at the application level a socket is opened when a connection needs to be sent out. Once this is done the Transport layer forms SYN packets and sends it. If a corresponding SYN/ACK packet is not received within certain time it resends the SYN packet again. These retrials are done until the socket doesn’t time out, in which case the application is notified. In our model, we count these retried SYN packets as separate connections as

well. This is not going to affect the results as during a worm or virus infection the SYN packet sent rate would be much more even than the addition of true SYN packets and retrials. Fig 3.2 shows the overall configuration.

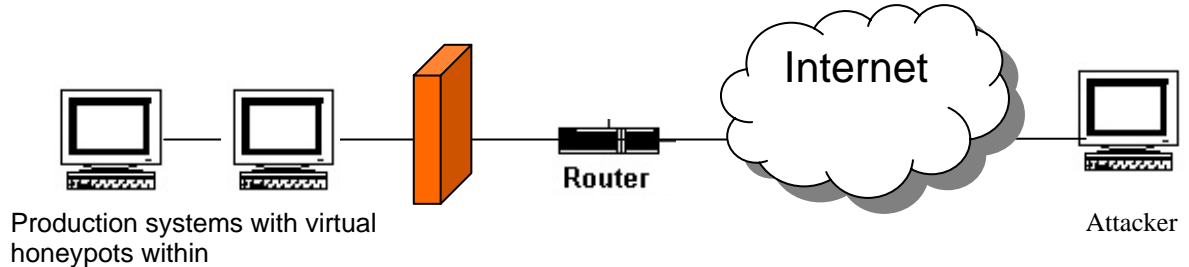


Fig 4.2 An example network for deploying virus throttling

Now that the connection count is sorted we need to figure how are we going to limit the rate. It is common observation that a machine makes a lot more connection to systems visited recently, we take this into consideration. In fact, this is also called local redirection and the normal rate of this operation is *one* connection per second. Thus, newness is defined by comparing the request with a list of recently visited hosts. The flowchart for the method would thus be as follows [54]:

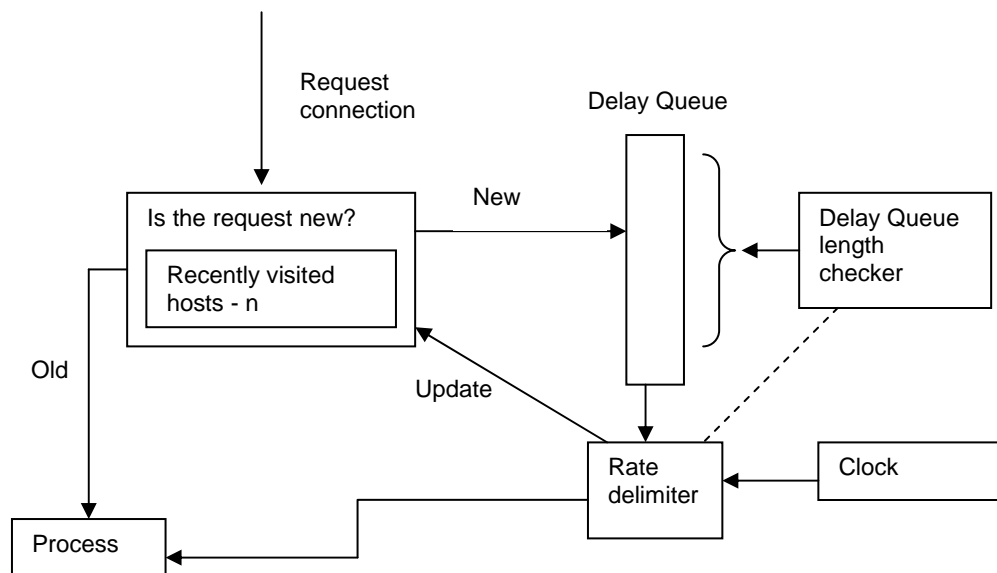


Fig 4.3 Flowchart for honeypots as mobile code throttlers (Source: HP Labs)

When a request for a connection goes out, it is checked for newness by comparing it with a set of recently visited hosts. If it is new it is put in a delay queue to be processed later, otherwise it is processed straight away. The control is established by the presence of a delay queue length checker. The delay queue will increase dramatically in case of a viral or worm infection and thus it will alarm the rate delimiter to throttle the rate of new connections. It in turn updates the incoming requests either not to be processed or alert system administrators to take suitable actions.

Having seen the working of the overall configuration we need to see another vital criteria - the placement of honeypot within the system. As we need to observe every packet going out of the system we need to keep the honeypot to monitor within the network stack of the system. Virtual honeypots like VMware are of great value here since they serve as just another system within a system and spare us the extra cost of installing one hardware honeypot for each system. The virtual honeypots would act as network scanner for the system and the system should be allowed to listen only on particular port and should be meant only for the virtual honeypot. This means that for example, a system establishes port 2000 to make the virtual honeypot scan the traffic then utilities like netcat has to be configured only to allow IP address of the virtual honeypot to listen on port 2000. The following script will do the trick:

```
/user/bin/nc -l -p 2000 x.x.x.x << c:/windows/cmd.exe
```

Thus, the incoming service at port 2000 of the virtual honeypot will get a command shell on connection. An important point is to make sure x.x.x.x is the IP address of the virtual honeypot only. Once the honeypot gets the command shell it can monitor traffic going out from the system through several packet grabber utilities like ethereal, tcpdump etc. On extracting the destination IP address on the packet it can then perform the procedure

outlined above to see the newness of the connection and limit the rate thereafter.

#### Discussion:

However, this approach has limitations in comparison to all its merits. Firstly the merits:

- A unique approach in comparison to recent studies on mobile code which are too signature dependent.
- Depends on network behaviour of mobile code rather than application behaviour. Thus, is able to limit network traffic congestion during infection.
- Based on traffic entering the network and is thus benign as compared to competitive designs based on traffic entering the system.
- Easy to deploy as needs no extra hardware and consumes less memory.
- Highly effective. The results in Williamson's case showed the throttling of Nimda virus by 80%.
- No legal hassles for this kind of honeypots as they are near to the production systems and are protecting them rather than logging.

#### Demerits:

- Is based on the assumption that systems connect to recently visited hosts. This might not be true for say email servers, which send emails at different rates and to different hosts. If that's the case worms and viruses are spread more through emails than any other medium. However, work has been done to deliver an email throttler as well [55].
- The length of delay queue is another concern. There has to be a certain threshold to the length of the queue because during virus propagation it will increase dramatically. However, if the threshold is specified, once it is crossed all the connections have to be dropped. This results in a denial of service for any system. Thus, the throttler just acts as a temporary

solution to a bigger problem. However, it is claimed that even during massive propagation the delay queue did not exceed length of 100 new connections.

However, we can see that the merits far outweigh the demerits associated with this strategy. Furthermore, it can be so visualised that once this technique has made its mark by proving and setting several results, providing this as a service to corporate users will bring out a thorough business model and huge profit margins in comparison to anti-virus software.

### **Deception technique 3: “Honeypot farms”**

#### Description:

In this strategy the honeypots serve primarily as a service. This idea might find huge acceptance by security sales firms and if implemented correctly can reap high amount of profits, since the infrastructure required for this is nothing but a couple of re-routing switches and virtual honeypots. Thus, even mid-size to small companies can extend their hands in this pool of profit. The basic principle is simple; re-route all the traffic coming to production system to pass through honeypots, which can be either in proximity or remotely located. That much done, the honeypots need to emulate the production systems and fool the incoming traffic that it is the real production system. The end users would not notice any difference, but if there is any malicious activity it can be logged, trapped, traced back or suitable action taken as per the security policy. In this way the host companies are saved from legal hassles and maintaining, deploying and monitoring honeypot configurations. Honeypot farms present a great deal of potential for the near future [38]. Also, another important point worth noting is the fact that how beneficial this would be to large enterprises, as they have thousands if not hundreds of nodes across the world. Deploying multiple

honeypots at various nodes require time, effort and a lot of manpower. And sometimes if the security policy within a certain network segment is loose, it might become a launch pad for attacking the whole firm. This is excluding the risk and liability to be considered. Honeypot farms can prove as a major rescue on these networks. One of the commercial products like this is NetBait [25]. NetBait provides off-site as well as in-house services for emulating and analysing on-demand your network traffic by use of virtual honeypots.

#### Working model:

There are two basic working models for this strategy. This is done because there are many approaches to this re-routing. As it's a new deployment of honeypots, the current approach is simple – to re-route only the traffic destined for non-used IP addresses. However, categorising threats only based on this may not be sound logic. Thus, novel ideas for categorising attacks and malicious intent has to be developed to make this strategy work.

#### Model 1: “Re-route only those bulls (traffic) to *farms*, who see red”

This deployment gets its working principle from Honeyd or specter. Only unused IP addresses are monitored and when request for interaction with these IP addresses comes in, they are re-routed. The following figure makes it clear.

The attacker and legitimate user both connect to the network via Internet and the final router routes packet to our network. The firewall allows traffic according to its configuration and the traffic heads to the re-routing switch. The switch, which in this case is a system, compares the destination address with unused IPs and if there is a match re-routes it to the honeypot farms.

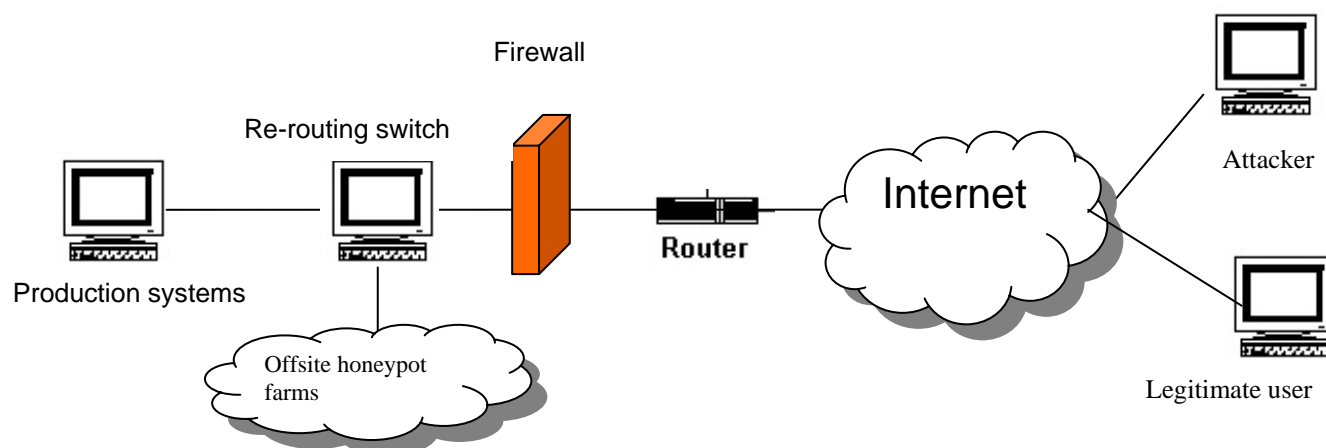


Fig. 4.4 An Example network for deploying honeypot farms

Otherwise the traffic is allowed to pass to the production systems. In this case, the switch will again act like a bridge and will have the same interfaces we had in our earlier network. Also, the farm can be localised within the corporate network if an in-house service is needed. As can be seen, the infrastructure used is simple and cheaper to install as compared to other competitive technologies. Also, we have combined two different honeypot technologies within this system – the re-routing nature of honeyd, which is a low-interaction honeypot and high interaction honeypots can be used in the honeypot farm like honeynet. This gives immense information about the purpose, tools and motives of the attackers and has the potential of securing malicious activity within lesser response time than singular honeypots. Another argument in the favour of this deployment is even if the attacker knows that he/she has been switched, he/she would just tear down the connection rather than just fiddling with a dummy server. This again is advantageous to the defenders, since they have succeeded in backing him/her off.

#### Model 2: “Re-route only labelled bulls (traffic) to the farms”

This strategy depends on its functioning on other technologies, but this dependence is just for recognising the malicious nature of the traffic. This

means that just to know the malicious intent of the incoming request we use other technology like intrusion detection systems (IDS) and once that is known, that particular request is re-routed to the honeypot farms. In fact, a research project bait-n-switch [53] is based on this principle. For deploying this technique we need:

- 1) A packet recogniser or Intrusion detection system like Snort, Nessus etc.
- 2) Re-routing switch.
- 3) Honeypot farm.

The following figure gives the setup:

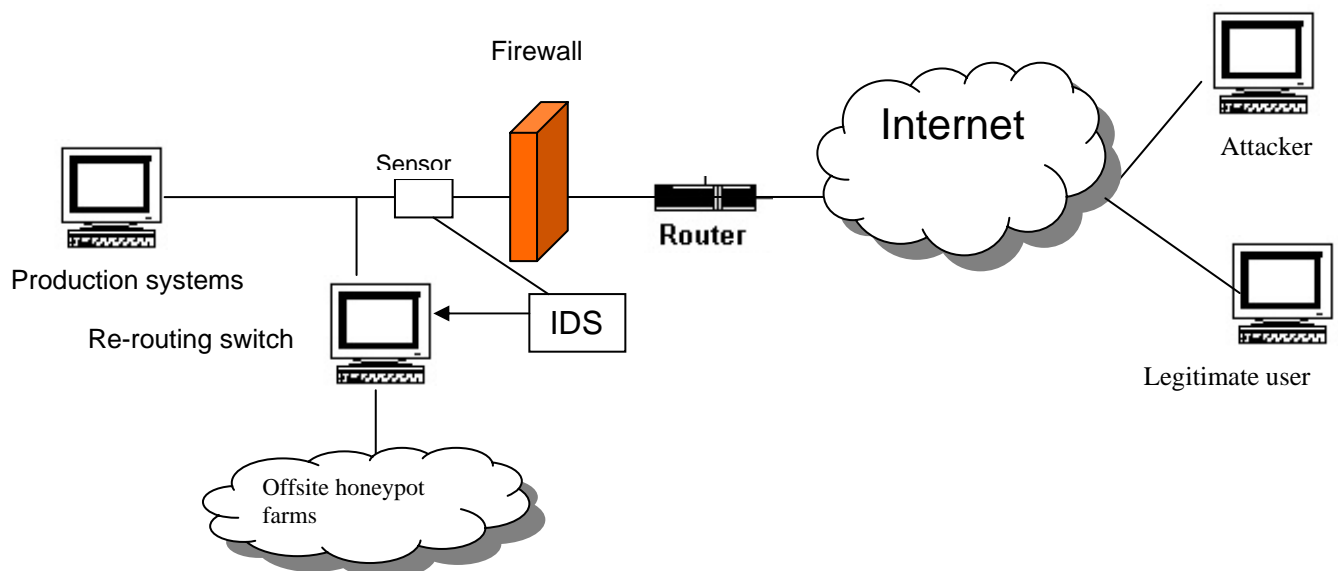


Fig 4.5 Another example for deploying honeypot farms

The major difference as compared to model 1 is the placement of the switch away from the production systems. But again, the switch will have the same three interfaces, but one of them will now serve as trigger from technologies like IDS, which detect malicious packets. Thus, when the IDS detects such packets the switch will re-route that traffic to honeypot farms where it will be logged, analysed and suitable action taken. Also, under normal behaviour the traffic will reach to its destined production systems and no harm would be caused.

Similar commercial products: NetBait, Bait-n-switch honeypot project

Discussion:

Both the above models surprise the commercial world with their potential. With no or negligible risks involved these will readily be adopted in the corporate world. But there are some answers that need to be answered for this deployment to succeed. If the honeypot farms are to be served as services then it means transferring attackers from one network to another, not considering there may be several routers in between. If this is so, how can this be achieved without the attacker knowing it? What activity do the re-routers transport and to which honeypots within the farm? Also, if there is dynamic transfer how do we ensure that the honeypot that the attacker is transferred to emulates the main server in context? Nonetheless, the potential and the uniqueness that this strategy has developed within the context of honeypot is worth a praise and will become a distinct phenomenon once some questions like above are answered. For a start, it is better to enlist the advantages for this strategy [38]:

- Removes one of the disadvantages of honeypots that they have a narrow field of vision. Honeypots can only monitor traffic passing through it and so have a smaller area to produce results. However, this honeypot deployment proves so useful that whatever traffic is deemed to be malicious is passed through honeypot and analysed. This makes it's vision more broad and gives more probability of giving results.
- Can combine the joys of both the world. This deployment can be configured to take advantages of both low-interaction honeypots such as Honeyd and high interaction honeypots as HoneyNet. As pointed, the re-routing feature is common in Honeyd and honeynets can be placed in the farms to monitor thoroughly the traffic in consideration.

- Puts all eggs in one basket and gives a centralised management for honeypots. More than often in a large enterprise managing different honeypots at different sites could be just infeasible, this approach negates that dictum and provides a day of relief to the already burdened system administrator(s).
- Low infrastructure and none to low initial cost for deployment.
- Can be provided as a service as compared to current approaches of providing only in-house honeypots.
- Since the honeypot farms can be excluded from the main production lines, the risk involved is either just a bit to none. Interestingly, considering the legal hassles a honeypot deployment has to go through, this proves so much of an easy deployment strategy.
- A novel and business oriented approach to deploy honeypots. Once this becomes mature enough to answer all the querying minds, this technology will reap enormous profits for security firms offering such services. In fact, the best buyers will be defense and governmental organisations.

However, we do need to study the demerits and are enlisted below:

- The second model depends on competitive technologies like intrusion detection systems, which have inherent weaknesses. If they fail, this technology might not be able to survive.
- Privacy issues may be of concern since false positives might yield legitimate traffic being passed through honeypot farms. If that traffic contains confidential information, exposing it to some outsourced honeypot service provider is intriguing to mind.
- Downstream liability may prove fatal if the honeypots are used to attack other sites.

However, seeing this demerits we can state that the overall strategy is not completely bleak but has extensive growth potential.

#### **Deception technique 4: Random servers – ‘You never know what you get’**

##### Description:

This deployment comes under the notion of security through obscurity. However, there is nothing confidential, but the network itself is just so probabilistic that you never know which node you are touching. The basic principle is to simulate the main servers within different honeypots and make a demilitarised zone of them [43]. Whenever the request for particular server(s) comes in, the request is assigned a server based on some pre-defined rule set or mathematical function. It is left only to probability that you get either the real server or a ‘honeypotted’ server. Although, this might prove to be less legal prone deployment, subtle attacks may be missed and the server still might get compromised. However, the success rate is at least better than having just main servers placed on the network. Also, the centralised management is another ease in this deployment.

##### Working model:

The model consists of virtual honeypots set on different servers. Virtual honeypots are chosen because they have such ease in use, and each of them can have different IP Address on a domain. Besides, putting this in context of a demilitarised zone (DMZ) gives enormous benefit because usually DMZs are smaller network (even a single class C network could be large for a DMZ). Observing a potential for greater number of unused IP addresses in a DMZ we can place virtual honeypots mirroring services offered on a server. For example, if five virtual honeypots (for example VMware) want to simulate the front page of the web server they reside on, a simple script would do:

##### On the real web server:

```
/usr/bin/nc -l -p 2000 x.x.x.x <c:/index.html
```

##### On the virtual honeypots:

```
/usr/bin/nc -p 2000 y.y.y.y >c:/index.html
```

where x.x.x.x = IP address of the virtual honeypot(s)

y.y.y.y = IP address of the real web server

Once this is done, most of our work is over. The following figure illustrates the idea:

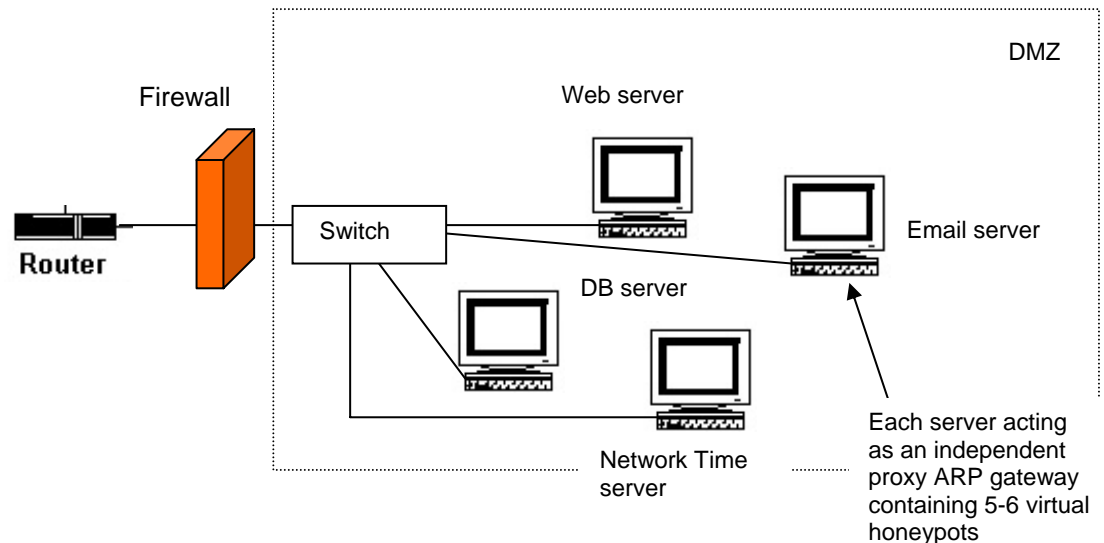


Fig 4.6 An example network for deploying random servers as honeypots

Also, while placing these deceptive services it has to be kept in mind that it will present the attackers with the first attack targets but not immediately attract the attackers. The probability that it will become a target of attack will depend on:

- Deployment,
- Quantity,
- Naming scheme and
- System policy.

Deploying it within the DMZ would mean the network offers several similar services. The attacker may not find any differences and might be confused about what to attack. It is this confusion that we take advantage of and log his activities. Also, this can be used against insider threats as well. Interestingly enough, if you want to increase the chances of the honeypots

being attacked you can increase the above factors in suitable ways like loose naming schemes like “Primary mail server” etc.

#### Discussion:

Understated are the relative merits and demerits of this strategy:

- Addresses outsider and insider threats equally and protects against both.
- Attackers are not lured into attacking targets, it is left to them.
- Central management of honeypots and ease in it because they are within the DMZ. So the central security office can control the whole operation.

However, there are quite a bit of demerits:

- Not much advantage, as the servers are still prone to attacks, only the probability has been decreased.
- Legal hassles and trouble maintaining honeypots up-to-date, besides skilled expertise.
- Required a good quantity to give results. The more the honeypots the less probability the attacker will hit the main server(s).

#### **Deception technique 5: Digital Breadcrumbs**

As there is an ascending order in sophistication of techniques as we move from 1 to 4, the present technique is a league apart in its own. The present technique too has a lot of potential in the ways and means it can be used. It is flexible in the sense that applications can include banking, law enforcement and other commercial sectors to home users who might just want to save their files. Without further adieu, the idea culminates from some detective works by Sherlock Holmes, and thus the name.

The basic idea behind this is that whenever an attackers accesses data from a system or network, he would either just access (read/write/delete) it or steal it. To prevent this, bogus data is embedded within actual data and

when this bogus data is accessed alerts are raised. Still further, if the data is stolen then the embedded 'bogus data' can call back the host system giving its location, however on suitable activation only. The just-access concept comes from a term coined, by Augusto Paes de Barros, 'honeytokens' in one of the mailing lists [2]. However, the technique under present context takes the concept of honeytokens a bit further and applies it to areas like steganography and trace-back.

Honeytokens are just bogus data, accounts, database entries, SSN or NI numbers which have no value in real world. However, they are embedded within real data for deceiving users. If they are accessed, alarms are raised to the system administrators cautioning them of malicious activity. In this way, they are even a synonym of honeypots by the definition we stated in Chapter 1, but a new term because they are not any information resource. They are just information in raw form, so it deviates from our definition of honeypot. Still, according to a recent article by Lance Spitzner [39] he does call it the other honeypot. These honeytokens or as I call it digital breadcrumbs have great value. There is no infrastructure needed, no signatures to update, no constant monitoring required nothing at all. Besides cost, they gain all the advantages of honeypots as they themselves are a part of honeypots.

#### Working procedure:

Since these are just dummy data within real information there can not be any models for this but just procedures on how to embed this data and its use. The honeytokens as is said can be anything, a bogus word document, a .tar file, SSN or NI numbers anything that can be thought of as significant. These are mixed with the real data and seen-to-be-touch. For example, a bogus medical record within the set of real records can be a honeytoken. Once it is accessed, we can have a probabilistic idea of some malicious activity. Significantly, Spitzner [39] gives a great example of honeytokens. If we are to find whether anyone is intercepting emails of higher management or human resources we can just plant a bogus email saying:

To: The CEO <CEO @ honeynet.com>  
From: Financial resources  
Subject: Access to financial database

Sir,

The security team has updated your access to the company's financial server (FSO1.honeynet.com). Your new login credentials are understated. Please do not hesitate to contact us for further assistance.

Login: honey

Password: h0n3yt0k3n

Now, whoever intercepts this message might try the futile effort to access FSO1 as well. But they just hit a honeytoken.

Traceback techniques like this are quite naïve but prove to be effective while dealing in commercial sector, especially banking and investments. However, there are advanced trace back techniques as well. One such approach would be to embed executables or scripts within sensitive files and once these scripts are stolen and activated, they would alert the host system of their presence. As one of my ideas under implementation, which was inspired by discussion from Rakan Al-Khalil the maker of Hydan [18], I am trying to embed .bat files within certain files like document files etc. These batch files get embedded to the source file in their raw binary form and once the document files are opened, the batch files get 'activated' and execute the script within them. The script could be as lame as:

```
Ifconfig > ip1.txt  
Mail me@mydomain.com  
~r ip1.txt  
.
```

,to something as a small code accessing the MAC address of the machine the file is on. However, the prolific question still remains and is yet to be answered within this research. How do we potentially activate these scripts within those transferred files? Some probable solutions are presented here. Firstly, if the real files are a web based file like .html or .asp there can be

mechanisms same as installing cookies, where in 'strings of text' (cookies) are stored within the user's computer depicting his preferences for certain items. In our case, it could be encrypted message of the above `ip1.txt` file stored as a cookie on the user's computer. If we are lucky, the careless attacker won't delete the cookie files and when he requests some other web page the cookie will be delivered to that server clandestinely. This approach however depends on mutual co-operation of Internet servers, but the *breadcrumb has been dropped* and tracing that can lead to the attacker. Another approach is of self-activating viruses, but we might be crossing lines here. However, if we see the application in law-enforcement etc, there may be ways to circumvent it. The idea is to transfer .hta files within the startup folders of windows so that whenever the system reboots there is an email back to the host system. However, this idea has to be carefully implemented knowing all the laws, liabilities etc. Other techniques include storing files within executables, as described by Hydan, but the idea that attackers will go after executables is a bit going-over-the-top [62]. Another idea is digital watermarking which itself has a great potential. Digitally watermarking our data or files will give them the authenticity we require in proving that the files belong to such and such host system. Once these files are found on the wild from any of the users, he/she may be prosecuted for stealing them.

Beyond comparison however, there will be ways to counter act on this matter and once a thorough analysis is made there could be good use of this technique within arena of Digital Rights Management (DRM), software piracy etc. The need of the hour is just to know where your file is on the Internet.

#### Discussion:

Having seen the various implications of honeytokens and uses they can be put to, it becomes clearer that they are those stars on the horizon which can outshine all the others soon. It left on the hands of implementers and

analysts how do we put them into practice. However, at this point a revision of their potential merits can be cited:

- Low-to-none infrastructure at all. As we have seen they are nothing but small software codes, entries, numbers, which are inserted just as another record within the real records.
- Eliminates false positives. Their importance comes into light because they are illicitly used. This eliminates the number of false positives largely.
- Protects equally from insider threats and attacks.
- Too flexible in use. They can be used in wide variety of applications, a common example presented here was trace back, but this flexibility leaves to your imagination about other applications.
- Highly effective in use, as it is common knowledge that the attackers will go through most records, entries within a database (especially the ones that seems to have more honey!)

However, there are a few understated demerits:

- The legal limits, similar to honeypots, are not yet known.
- Effectiveness depends on use. If they are not used they are of no value.

Having seen the various techniques and tools by which a healthy deception can be implemented using honeypots, it becomes clear that they are highly flexible in their applications and thus serve as a major security tool. The next chapter gives a complete conclusion to the research as a whole and brings to light some exciting points about honeypots.

## Chapter 5

### Conclusion

From the start to the end of this varied research it should have come to a logical mind that honeypot unravels itself as a worthy candidate for acceptance within security. Their flexibility, applications in various areas, value, cost involved, result they produce are some interesting topics we can consider as their potential advantage. These almost necessarily outwit others. In this chapter however, we give a chapter wise and final conclusion of how this varied tool exerts its importance in security.

In chapter 1, we saw a thorough analysis of the honeypot concept. The ways they can be moulded to research and production view presented us with an outlook to rate them highly from other competitions. Besides, there are very few tools that lend themselves usable to both these sides. In research, they gather outstanding information, reveals tactics used, unravels newer attacks and educates the defenders. Commercially, they neglect false positives, eases administration, logs successful and unsuccessful attacks with thorough details, acquaints with zero-day attacks and serve as a third line of defence. Having these varied applications, they prove to be a great concept and when particularly understood by higher management will get wider acceptance.

Chapter 2 described the legal issues concerning honeypots and thus served us with a basis for comparing our knowledge with laws within honeypots. As can be concluded, there is no distinct line for what is right and what is not; because there hasn't been any court cases regarding honeypots. But a thorough insight into these legal concepts revealed at least some points to the intriguing mind. Legal issues concerning honeypots are not new and they are adopted from similar criminal concepts like entrapment. Seeing that these concepts are dealt with case-by-case bases, honeypot cases would also

be dealt similarly. The best practice will be to avoid as much as hassle by keeping them nearer to production systems and developing case scenarios for privacy concerns. Studying and researching the local laws completely and consulting respective lawyers can decrease liabilities concerned within. However, the first cases will pave a general way for further deployment, but presently it does seem that there will be implications for using them widely.

Chapter 3 presented us with some points on risk mitigation for honeypots. As it is a new technology with blurred legal boundaries, it comes with its own risks. While deploying it has to be taken care that we undergo a thorough risk analysis and develop a tightened security policy for their maintenance. A separate policy might mean more work on administration, but it will also ease them of future burden of attacks and analysis.

Chapter 4, which happened to be the core of the research gave us in depth view of some innovative ways deception can be applied to honeypots. These techniques when used wisely and in proper context will give excellent results. Firstly, as a simple port listener, avoidance of false positives, noise reduction, efficient use of manpower, and ease in deployment gives their advantages. However, this being the most basic deployment of honeypots, does not raise their usage wider as there are many feature-rich competitive candidates in the field. As mobile code throttlers, they display their immense potential for being so flexible. Honeypots started as tools for reaction rather than detection and this technique negates that ice-age conception. With the help of in-depth analysis we could cite that they could mould themselves to have effect on major security problems like mobile code. The technique is worth implementing in mid-size to larger network and gives excellent results. As decoy servers, they can have business advantage as well over other candidates and here they display their commercial merits. Also, central management of these servers within 'farms' gives ample time for research and analysis even within a commercial R&D department. Providing honeypots as

services will reap greater profits to firms in this field in near future. Similarly, for high-sensitive information and agencies honeypots adorn the cloak of honeytokens and breadcrumbs and yield quality results. They can be used for traceback, recognising intent, prosecution and probably return of investment. Also, the low-cost in this deployment make them acceptable to a great extent.

Over the ages, newer technology always found resistance for acceptance and it is nothing new with honeypots. It is a completely new arena in the field of security. Currently there are quite a number of researches and discussions all around the world, several research groups and companies have deployed products already, but their usage and future needs to be seen. Also, there is a larger misconception of them being evolved from a military setup and that is a hindrance to its usage. Nevertheless, they have matured as a technology due to their flexible nature and wide applications. But this flexible nature also infers of them having no firm placement in security as IDSs and firewall. But they can be moulded to meet any objective. Having seen that security objectives defer between companies, research, law enforcement and financial institutions a common ground is established by honeypots. They do bring risks, harms and unexplored legal hassles with them, which have to be thoroughly analysed before deploying them. Especially, third-party complains like downstream liability and privacy may induce these analysis to be wider and more detailed. However, it is a varied tool and in dealing with discovering malicious intent and gathering information no technology can outwit them at present. The obvious advantages of reducing false positives and pinpointing the attacks make them far efficient than competitive technologies.

Honeypots are still in their infancy. Once tightened laws and thorough understanding of their varied concepts are explained, they will have a niche for themselves in security.

# Bibliography

**1. 'Brian'**

A simple NetCat honeypot

Link:

<http://www.securityhorizon.com/whitepapers/technical/honeypot.html>

**2. Augusto Paes de Barros, CISSP (2003)**

An active researcher in the field of honeypots and the first one to coin the term 'honeytokens'

Link: <http://lists.insecure.org/lists/focus-ids/2003/Feb/0095.html>

**3. Bakos George and Beale Jay (2001)**

*Honeypot advantages and disadvantages*

Honeypot best practices

Seminar at Dartmouth College, Hanover, New Hampshire

**4. Bellovin Steven (1992)**

*There be dragons* - in proceedings of the third USENIX Unix Security Symposium

Link: <http://www.research.att.com/~smb/papers/dragon.pdf>

**5. Benett Jeremy (2002)**

*Deploying Deception* – seminar on Cybersecurity, Feltham UK  
Recourse Technologies (now acquired by Symantec)

**6. Bruce Schneier (2000)**

Secret and Lies – Digital Security in networked World  
Wiley Computer Publishing

**7. CCITT, Recommendation X.800 (1991)**

*Security Architecture for Open Systems Interconnection for CCITT (The International Telegraph and Telephone Consultative Committee) Applications*

Recommendation X.800

\* Recommendation X.800 and ISO/ITU 7498-2 are technically aligned.

**8. Cheswick Bill (1991)**

*An evening with berferd in which a cracker is lured, endured and studied*

AT&T Bell Laboratories.

Link:

<http://www.eecs.umich.edu/~aprakash/security/reviews/jiayingz-IDS.pdf>

**9. Directive 97/66/EC (1997)**

Concerning the processing of personal data and the protection of privacy in the telecommunications sector.

Link:

<http://europa.eu.int/ISPO/infosoc/telecompolicy/en/9766en.pdf>

**10. Electronic Communications Privacy Act (1968)**

Privacy law in USA

18 USC 2510 – 2521 – *wire and electronic communications interception and interception of oral communications*

Link: <http://floridalawfirm.com/privacy.html>

**11. Fourth amendment of US constitution, USA (1791)**

Privacy protection in terms of US constitution

Link: <http://www.usconstitution.net/const.html#Am4>

**12. Growsman Norman**

*Courts willing to consider enticement in calculating notice period – as article on enticement used as a defense in Canadian court.*

Link:

<http://www.workopolis.com/servlet/Content/rprinter/20030528/1s20030528>

**13. Gubbels Kecia (2002)**

*Hands in the Honey pot*

SANS research paper on Honeyd – the virtual honeypot.

Link: <http://www.sans.org/rr/papers/30/365.pdf>

**14. Haig Leigh (2002)**

*LaBrea – a new approach top securing our networks*

SANS paper on LaBrea – the ‘sticky’ honeypot

Link: <http://www.sans.org/rr/paper.php?id=36>

**15. Homepage for Deception Toolkit (DTK)**

The first homemade honeypot in context by Fred Cohen

Link: <http://www.all.net/dtk/dtk.html>

### **16. Honeyd homepage**

Neils Provos the maker of honeyd - A virtual honeynet for gathering information by re-routing malicious traffic

Link: <http://www.citi.umich.edu/u/provos/honeyd/>

### **17. Honeynet Project**

Lance Spitzner

*Know Your Enemy series I, II and III – revealing the security tools, tactics and motives of the blackhat community*

Addison –Wesley 2000

### **18. Hydan – Rakan Al-Khalil (2003)**

A groundbreaking program that hides messages within executables.

The researcher is a student at Columbia university NY, USA

Link: <http://www.crazyboy.com/hydan/>

### **19. Jacobson vs United States (1992)**

503 US 540

Another prominent case on entrapment

Link:

<http://caselaw.lp.findlaw.com/scripts/getcase.pl?court=US&vol=503&invol=540>

### **20. Lipson Howard F. (2002)**

*Tracking and tracing cyber attacks: technical challenges and global policy issues* – CERT report on traceback

Link: <http://www.cert.org/archive/pdf/02sr009.pdf>

### **21. Liston Tom**

*LaBrea – The ‘sticky honeypot’ and IDS*

Honeypot that tar-pits hackers for indefinite time.

Link: <http://labrea.sourceforge.net/labrea-info.html>

### **22. Mantrap or Symantec Decoy Server honeypage**

A high interaction commercial honeypot by Symantec

Link:

<http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=157>

### **23. Messmer Ellen (2002)**

*Behaviour blocking repels new viruses* - article in network world on mobile code

Link: <http://www.nwfusion.com/news/2002/0128antivirus.html>

**24. Neils Provos, University of Michigan**

*Honeyd- a virtual honeypot daemon*

Paper on working of honeyd – a virtual honeypot.

Link: <http://niels.xtdnet.nl/papers/honeyd-eabstract.pdf>

**25. NetBait**

A commercial off-site and in-house honeypot service provided by the same company

Link: <http://www.netbaitinc.com/>

**26. Netcat homepage**

One of the most varied tool in security that logs, listens, writes and exposes network traffic

Link: <http://netcat.sourceforge.net/>

**27. NetSec homepage**

Company offering commercial low-interaction windows-based honeypot Specter

Link: <http://www.specter.com/default50.htm>

**28. Pen Register and Trap and Trace Statute, USA**

18 USC 3121 – *exception to the general prohibition on use of Pen Register and trap and trace devices.*

Link:

[http://caselaw.lp.findlaw.com/cascode/uscodes/18/parts/ii/chapters/206/sections/section\\_3121.html](http://caselaw.lp.findlaw.com/cascode/uscodes/18/parts/ii/chapters/206/sections/section_3121.html)

**29. Poulsen Kevin**

*Use a honeypot, go to prison?*

An interesting article on Securityfocus

Link: <http://www.securityfocus.com/news/4004>

**30. Prof. Allen Ronald, Luttrell Melissa, Kreeger Anne**

*Clarifying Entrapment* - a great article on clarifying entrapment and concepts surrounding it.

Northwestern University school of Law

Link: <http://www.law.qub.ac.uk/ice/papers/entrap1.html>

**31. Regina vs Loosely**

The first famous UK case on entrapment

Link: <http://www.parliament.the-stationery-office.co.uk/pa/ld200102/ldjudgmt/jd011025/loose-4.htm>

**32. Regulation of Investigatory Powers Act (RIPA), 2000**

Investigatory law in UK describing various privacy and detective surveillance issues

Link: <http://www.hmso.gov.uk/acts/acts2000/20000023.htm>

**33. Sherman vs United States (1958)**

Important ruling on entrapment

356 US 369

Link:

<http://caselaw.lp.findlaw.com/scripts/getcase.pl?court=US&vol=356&invol=369>

**34. Sorrells vs United States, 1932**

First federal court case on entrapment

287 US 435

Link: [http://caselaw.lp.findlaw.com/cgi-](http://caselaw.lp.findlaw.com/cgi-bin/getcase.pl?court=us&vol=287&invol=435#442)

[bin/getcase.pl?court=us&vol=287&invol=435#442](http://caselaw.lp.findlaw.com/cgi-bin/getcase.pl?court=us&vol=287&invol=435#442)

**35. Spitzner Lance (2003)**

*Honeypots: Definitions and value of honeypots*

Link: <http://www.tracking-hackers.com/papers/honeypots.html>

**36. Spitzner Lance**

*Specter: a commercial honeypot solution for windows*

Article on Specter a low-interaction honeypot in Security Focus

Link: <http://www.securityfocus.com/infocus/1683>

**37. Spitzner, Lance (2003)**

*Honeypots: are they illegal*, Guest Article in Securityfocus

Link: <http://www.securityfocus.com/infocus/1703>

**38. Spitzner Lance (2003)**

*Honeypot Farms*- guest article in Security Focus

Link: <http://www.securityfocus.com/infocus/1720>

**39. Spitzner Lance (2003)**

*Honeytokens: the other honeypot* a guest article in Security Focus

Link: <http://www.securityfocus.com/infocus/1713>

**40. Spitzner Lance (2003)**

*Honeypots: Tracking Hackers*

Addison Wiley Publications

**41. Spitzner Lance and Roesch Marty**

*The value of honeypots Part I and II: Definitions and values of honeypots*

Security focus guest article

Link: <http://www.securityfocus.com/infocus/1492>

**42. Stoll Clifford (1989)**

The cuckoo's egg – tracking a spy through the maze of computer espionage  
Pocket book publications

**43. Symantec Inc.**

*ManTrap – a secure deception system* - a technical report on deployment and use of ManTrap, the high-interaction honeypot

Link:

<http://www.dlt.com/quest/pdf/application%20monitoring/symantec/mantrap.pdf>

**44. The Federal Wiretap Act, USA(1968)**

18 U.S.C. 2511 – *Interception and disclosure of wire, oral, or electronic communications prohibited*

Link: <http://www.cybercrime.gov/usc2511.htm>

**45. The HoneyNet Project**

Link: [www.honeynet.org](http://www.honeynet.org)

**46. The HoneyNet Project and HoneyNet Research Alliance**

*Profile – Automated Credit Card fraud 2003*

Link: <http://www.honeynet.org/papers/profiles/cc-fraud.pdf>

**47. The Page Museum at Rancho, Los Angeles**

Museum for huge tar-pits that caught large mammoths

Link: <http://www.tarpits.org/info/visit.html>

**48. Tom Liston talks about LaBrea**

The maker of sticky honeypot that tar-pits attacks on unused or non-existent IP addresses.

Link: <http://labrea.sourceforge.net/Intro-History.html>

**49. Magalhaes Ricky M. (2003)**

*Understanding Virtual honeynets* – an article in Windowsecurity.com

Link:

[http://www.windowsecurity.com/articles/Understanding\\_Virtual\\_Honeynets.html](http://www.windowsecurity.com/articles/Understanding_Virtual_Honeynets.html)

**50. United States vs. Maxwell (1996)**

45 M.J. 406 – case describing protection of individual privacy in basic telephone system.

Link: <http://www.ipwatchdog.com/maxwell.html>

**51. User Mode Linux**

A kernel of Linux that supports virtual machines and their creation at will.

Link: <http://user-mode-linux.sourceforge.net/>

**52. VMware honeypage**

The first commercial virtual machines software selling company

Link: <http://www.vmware.com/>

**53. Whitsitt Jack, 'joFny'**

Violating US Inc.

*The Bait-n-switch honeypot*

A research project for re-routing malicious traffic to remote honeypots

Link: <http://violating.us/projects/baitnswitch/>

**54. Williamson Matthew and Twycross Jamie (2003)**

*Implementing and testing a virus throttle* – mobile code research at Hewlett Packard (HP) labs in Bristol, UK

Link: <http://www.hpl.hp.com/techreports/2003/HPL-2003-103.pdf>

**55. Williamson Matthew (2003)**

*The design, implementation and testing of an email throttle* - submitted at Annual Computer Security applications conference, Las Vegas

Link: <http://www.hpl.hp.com/techreports/2003/HPL-2003-103.pdf>

**56. X-force Internet watch honeypot modified by USG**

Clarification by Internet Security systems (ISS) on May 2003 hack on their server.

Link: <http://xfiw.iss.net/>

**57. Yurcik William, Rosendale Jeff and Barlow James (2003)**

A research paper on: *Maintaining Perspective on Who Is The Enemy in the Security Systems Administration of Computer Networks*

National Center for Supercomputing Applications (NCSA)

University of Illinois at Urbana-Champaign

Link: [http://www.cs.berkeley.edu/~mikechen/chi2003-sysadmin/papers/WilliamYurcik\\_NCSA\\_WhoIsTheEnemy.pdf](http://www.cs.berkeley.edu/~mikechen/chi2003-sysadmin/papers/WilliamYurcik_NCSA_WhoIsTheEnemy.pdf)

**58. Zimmerman Scott, Plesco Ron, Rosenberg Tim**

Downstream liability for attack relay and amplification – *Citation from RSA conference 2002, San Jose, California.*

Link: [http://www.cert.org/archive/pdf/Downstream\\_Liability.pdf](http://www.cert.org/archive/pdf/Downstream_Liability.pdf)

**59. The USA – Patriot Act (2001)**

*An act to deter and punish terrorist activities within the USA and around the world, to enhance law enforcement investigatory tools and other purposes.*

Link: <http://www.epic.org/privacy/terrorism/hr3162.html>

**60. Edmead Mark and Kim Gene**

*Honeypot Best Practices: mitigating risks* – a seminar on mitigating risks involved in honeypot deployment

Information Technology Research Associates (ITRA)

Article in ComputerWorld September 2002

**61. Nmap homepage**

Nmap - A stealth port scanner and network tool for writing and accessing data across ports. Authored by Fyodor.

Link: <http://www.insecure.org/nmap/>

**62. Edward G. Amoroso**

*Intrusion detection: An introduction to Internet surveillance, correlation, trace-back, traps and response.*

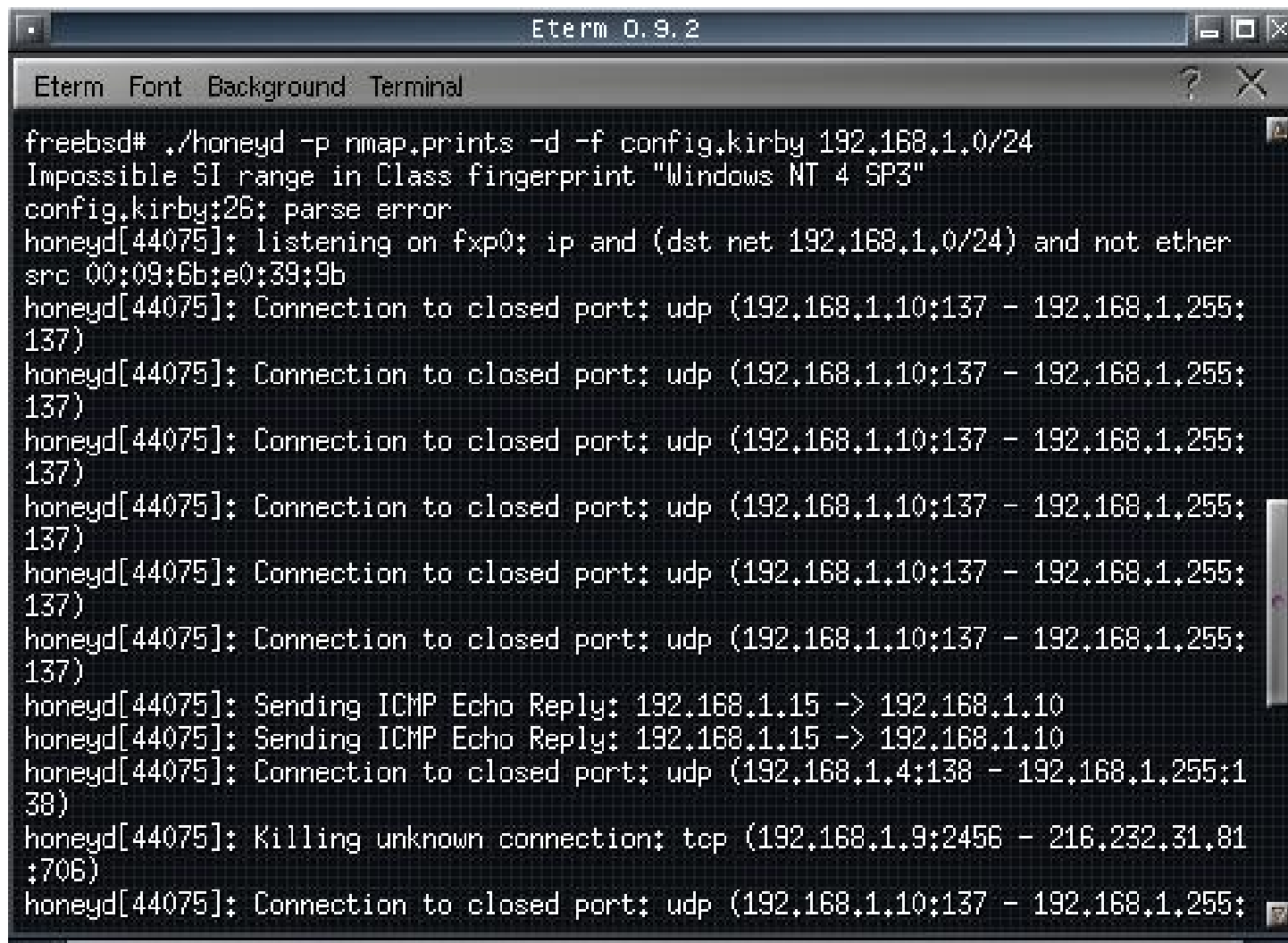
Addison – Wiley Publications

## List of tables

<b><u>Table number</u></b>	<b><u>Title</u></b>	<b><u>Page</u></b>
1.1	Services and traps in Specter	20
2.1	Differences between entrapment and enticement	31
4.1	Commercial honeypots and their deployment strategies	49

## List of figures

<b><u>Figure number</u></b>	<b><u>Title</u></b>	<b><u>Page</u></b>
1.1	LaBrea Screenshot	16i
1.2	Working of honeyd	17
1.3	Honeyd Screenshot	18i
3.1	Example network for risk mitigation	42
4.1	Network for port listener Technique	51
4.2	Network for deploying virus throttling	57
4.3	Flowchart for mobile code throttler	57
4.4	Network for deploying Honeypot farms	62
4.5	Another network for deploying honeypot farms	63
4.6	Network for deploying random servers	67



```
freebsd# ./honeyd -p nmap.prints -d -f config.kirby 192.168.1.0/24
Impossible SI range in Class fingerprint "Windows NT 4 SP3"
config.kirby:26: parse error
honeyd[44075]: listening on fxp0: ip and (dst net 192.168.1.0/24) and not ether
src 00:09:6b:e0:39:9b
honeyd[44075]: Connection to closed port: udp (192.168.1.10:137 - 192.168.1.255:
137)
honeyd[44075]: Connection to closed port: udp (192.168.1.10:137 - 192.168.1.255:
137)
honeyd[44075]: Connection to closed port: udp (192.168.1.10:137 - 192.168.1.255:
137)
honeyd[44075]: Connection to closed port: udp (192.168.1.10:137 - 192.168.1.255:
137)
honeyd[44075]: Connection to closed port: udp (192.168.1.10:137 - 192.168.1.255:
137)
honeyd[44075]: Connection to closed port: udp (192.168.1.10:137 - 192.168.1.255:
137)
honeyd[44075]: Connection to closed port: udp (192.168.1.10:137 - 192.168.1.255:
137)
honeyd[44075]: Sending ICMP Echo Reply: 192.168.1.15 -> 192.168.1.10
honeyd[44075]: Sending ICMP Echo Reply: 192.168.1.15 -> 192.168.1.10
honeyd[44075]: Connection to closed port: udp (192.168.1.4:138 - 192.168.1.255:1
38)
honeyd[44075]: Killing unknown connection: tcp (192.168.1.9:2456 - 216.232.31.81
:706)
honeyd[44075]: Connection to closed port: udp (192.168.1.10:137 - 192.168.1.255:
```

Fig. 1.3 Honeyd screenshot