Catching Flies: A Guide to the Various Flavors of Honeypots

GIAC (GCIA) Gold Certification

Author: Scott D Smith, Smith24197@gmail.com

Advisor: Mohammed Haron

Accepted: April 12th, 2016

Abstract

While the concept of baiting adversaries in order to monitor their activities is nothing new, honeypotting has evolved into a critical tool in information security analysis. Recent years have given rise to advances in the detection of network intrusions such as honeynets, honeytokens and adaptive honeypots. This paper will explore modern applications, as well as the legal and technical considerations behind emerging honeypot solutions in the dynamic blockage of emerging attack vectors and the potential exploitation of advanced persistent threats.

1. Introduction

In the broadest and time-honoured sense, the metaphorical honeypot is a trap that targets potentially malicious actors. The set-up typically involves an object of illicit value, an opportunity for one of more and a mechanism or agent to gather evidence. Common examples prior to the late 20th century include police forces' use of bait cars to lure out potential car thieves, or entrapment of criminals by undercover agents.

The concept has enjoyed resurgence since the advent of the Information Age as a valuable investigative tool in computer security. Nowadays, any vulnerable computer systems or networks designed to be attractive to hackers as a target for intrusion is considered a honeypot [16].

2. Honeypots in Information Security

2.1. Honeypot

SANS institute defines honeypots as: "*an information system resource whose value lies in unauthorized or illicit use of that resource*" [2]. To many in the IT security community, that definition fails to capture much of what sets honeypots apart from other security tools. Several subcategories of classifications, designs and foci will be analysed in depth in the next section. What is important to note, however, is that an information system resource need not be limited to computing when discussing honeypots. In fact, the concept has recently had great success at fingerprinting telemarketers and robo-call scam operations in the telephony industry [14].

2.2. Honeynet

The odds of a single honeypot picking up intrusion activities decreases as the number of systems connected in a network are increased. Larger and more diverse networks consequently typically require the deployment of multiple honeypot instances. When configured to operate in unison, the result is a network of high interaction honeypots that simulates a production network and configured to monitor, to record and discreetly regulate all activities.

Multiple iterations of honeypots have the added benefit of strategic placement in key subsystems or zones throughout the network. Correlation of logs between honeypots can map out a timeline of an intruder's activities, and provides further useful data for forensic investigation and clean up. Centralizing the collected activity logs and analysis tools in an administrative management client can further sped up the analysis. These management clients are labelled honeyfarms.

It should be noted that honeynets as described in this paper are unrelated to the international Honeynet Project effort, which is a non-profit organization dedicated to collaborative research on emerging threats to internet security.

2.3. Honeytoken

One of the most widespread misconceptions about honeypots is the idea they must be a computer or interactive resource. It is important to remember that an information system resource need not be limited to physical resources or computers, but includes data as well. Whether that data is a phony database entry or bogus email, the value lies in its unauthorized use. Augusto Paes de Barros is credited with coining the term honeytoken in 2003 in an attempt to distinguish the concept for the IT Security community [16].

A successful honeytoken has two key characteristics: uniqueness, and an improbable chance of appearing in legitimate traffic. These features will go beyond merely ensuring the integrity of the system resource by enabling the use of reactive security measures, such as dropping packets containing the honeytoken at the boundary router in order to contain a potential data spill. Simple alert rules triggered by the detection of distinct honeytokens may also provide system administrators advanced notice for incident handling. The value quickly diminishes, however, if the honeytoken appears in legitimate traffic and causes false positives.

2.3.1. HoneyCreds

Honey credentials (or HoneyCreds, Honey Hashes, and/or Canary Credentials, depending on whom you ask) are a relatively new application of the Honeytoken concept that has gained traction in web-facing login security. As the frequency and sophistication of brute-force attacks has increased, network administrators have taken to including faked logins and passwords within lists of legitimate credentials. They are mixed with legitimate credentials, salted and hashed, and made readable by root only. In this way, even if the hash file is stolen and cracked, the decoy credentials can provide early warning of unauthorized access [22].

HoneyCreds may also be used to prohibit remote access from super/elevated privilege users. Since root or administrator accounts should never be able to login via web services, an attempt of their logging in is a likely an indicator of attack. A DenyAll ruleset can be written in these cases, with logging of all attempted passwords for later review. This is an excellent source for future decoy passwords, as many brute-force dictionary attack tools share common password lists [22].

2.4. Honeytrap

While many in the IT security community use the term synonymously with honeypots, a honeytrap is more practically used to describe a planned cyber warfare operation towards a specific target with the goal of extorting vital information. A recent example is the exploitation of the Free Syrian Army and Islamic supporters' sensitive data, as reported by FireEye in 2015 [5]. An undisclosed hacker group targeted rebels via social media channels and fake matchmaking websites, and operators posed as attractive female sympathizers. Rebels were tricked into disclosing information over common chat and VoIP applications regarding what smartphones they had and which operating systems they used. The info gathered was subsequently used to develop exploits embedded into in pictures, which the fighters downloaded. Strategic documents, battle plans, inventories and personal data were stolen in this fashion over a two-month period.

3. Design Criteria

3.1. Classification

3.1.1. Production

Production honeypots typically used as a litmus test to signal unauthorized access of a company or corporation's production systems. They are most commonly deployed alongside essential servers and emulate only the services a hacker is likely to call. They are inexpensive to maintain and restore, and require few network resources to operate.

3.1.2. Research

Research honeypots are fully interactive designed to be compromised systems. Rather than simply signal that an intrusion has occurred, these honeypots provide information on the motives and tactics of hacker communities. Researchers are then able to study specific exploits and learn how to better protect systems against those threats. The complexity of research honeypots makes them extremely expensive to deploy, maintain and study. They are used primarily by large groups with extensive security research and development interests such as government, corporate leaders in IT security solutions, military, and academic organizations. The Honeynet Project is an example of a collaborative research honeypot community.

3.2. Design

3.2.1. Full

Also called pure honeypots, these are full-fledged production systems. Taps on the honeypots' links to the network used to capture attacker interactions. These systems are identical to other production systems, with the exception that they are unused. Any activity on the honeypot is an indicator of unauthorized use.

3.2.2. High Interactive

High-interaction honeypots are designed to mimic the services of the production systems. In so doing, they provide better security in obscuration, and deceive malicious actors into wasting their time and resources. These honeypots are best set up as virtual machines, which are easily restored once compromised.

3.2.3. Low Interactive

Low-interaction honeypots simulate only the services frequently requested by attackers. They are well suited for virtualization since fewer resources are required. The virtual systems also have a short response time and require less code, thereby reducing the complexity of the virtual system's security.

3.3. Focus

3.3.1. Malware

These are honeypots designed to detect malware by exploiting known replication and attack vectors such as removable media. These vectors are checked for evidence of modifications, through either manual scans or the use of special-purpose honeypots that emulate drives in a proxy environment. Any unusual activity will trigger an alert, and may quarantine the media for investigation.

3.3.2. Email/Spam

Also called spamtraps, the earliest email honeypots were simply unused email addresses. Any unsolicited electronic messages sent to these email addresses have a high likelihood of being spam. While most spam is relatively easily to identify, an email honeypot can better distinguish elaborate phishing attempts from legitimate emails.

Recent developments in email honeypots have featured specific capabilities to combat spam. By posing as a known 'dropbox' (i.e. an email address know to be targeted by spammers for testing purposes), the honeypot deceives the attacker into thinking it is an open relay on the SMTP server. When the bulk relay messages are sent, the honeypot drops the request. They are also able to support blacklisting, whereby the honeypot records malicious IP address and URLs, and shares them with other web and email servers in order to block repeated attacks upstream.

3.3.3. Database

Databases are an ideal environment for utilizing the full potential of honeytokens. Linking a few false entries in an authentic table to intrusion detection system (IDS) alerts permits the triggering of firewall rules in dropping egress packets and dynamic blocking of unknown IPs used by attackers.

Several web application firewalls have been specifically designed for web database protection. Some feature honeypot architecture support, which employs trap databases when the firewall detects unauthorized use through signature comparison. In this way, the system discreetly quarantines the attacker while the web application remains functional.

4. Critical Considerations

4.1. Legal Responsibility

It should be noted the scope of this paper is limited to bridging the gap between the technical and legal aspects regarding honeypots, and by no means should be considered a replacement for legal advice from informed legal counsel.

4.1.1. Entrapment

A popular misnomer with honeypots is that they are a form of entrapment, and employing one may be grounds for legal prosecution. There are two mistakes in this concept. First, entrapment is purely a defence argument against criminal conviction. This means that the owner of the honeypot would need to first press charges against the attacker [13].

Secondly, entrapment is defined as "The inducement, by law officers or their agent, of another person to commit a crime for the purpose of bringing charges for the commission of that artificially-provoked crime" [23]. This effectively excludes entrapment from applying to the non-law enforcement community. Furthermore, even if a honeytrap was deployed by a government or police agent in order to generate evidence for prosecution, the defendant would need to prove they were coerced into breaching said honeypot. In an overwhelming number of cases, it is clear legal entrapment is a non-issue [15].

4.1.2. Privacy

Privacy is a multi-faceted challenge. First and foremost, no single statute presently exists in the global protection of information privacy. There may in fact be variable and contradictory laws within a single country. The United States of America is a well-known case wherein even laws at the federal level may conflict, such as the Federal Wiretap Act and the Electronic Communication Privacy Act [13]. To compound the confusion, state laws concerning privacy can supplement Federal law, as it is in the state of California. Therefore, if a honeypot is deployed in Colorado Springs and the attacker breaches it from San Francisco, which privacy laws apply; Colorado, California, or Federal? Do Section 184 of the Canada Criminal Code and Article 185 of the

Convention on Cybercrime apply if attack originates from Ontario? This problem becomes exponentially greater as attackers hide behind multiple proxy servers from separate countries.

The application of some laws further depends on the type of information being collected. The two general categories are transactional and content. Transactional is not the data itself, but information about the data. Common examples include OSI layer header data, metadata and date/time stamps. Content data refers to packet payloads, such as chat and email messages, and even keystrokes. Transactional data understandably has less privacy sensitivities associated with it when compared to content data.

Finally, privacy laws may also be subject to issues of consent. How does one get an attacker to demonstrably consent to monitoring and waive their rights to privacy? A simple solution is the configuration of text banners into the TCP handshake of the honeypots ports. A simple example of such a banner could look like [7]:

```
############################################################
#              !READ BEFORE CONTINUING!
#  This system is for the use of authorized users only.
#  By using this computer you are hereby consenting to
#  all of your activity on this system being monitored
#  and disclosed to others.
#############################################################
```

Should an attacker continue to compromise the honeypot, they will have officially consented to information collection on their activities.

Advocates will rightly point out the impossibility to banner all the ports that are possible for an attacker to break into, not to mention all the different languages that an attacker could possibly speak. The challenge then becomes which ports should be bannered, and in what language? The common practice is to banner the ports/services that are normally bannered in the official language of the country the honeypot is deployed. This is usually sufficient to demonstrate an acceptable measure of due diligence [13].

4.1.3. Liability

Liability indicates an operator could be sued in civil court if others came to harm because their honeypot was compromised and used to in subsequent attacks. In the event those attacks caused damage to third party other systems or resources, those third parties could seek legal damages. This argument is dependent on claiming that had the honeypot operator taken proper precautions to keep secure, the attacker could not have been able to harm the honeypot. Ergo, the operator is at fault for a share of any damage that occurred to the plaintiff because of the attack.

The problem with these legal accusations is that anytime even the most secure technology is deployed, there is some inherent risk. New vulnerabilities are discovered regularly in firewalls, IDS, and network sniffers. Obviously, honeypots are no exception. There are, however, different levels of risk dependent upon what kind of honeypot is deployed. Low-interaction variants using emulated services cannot be broken and abused, and are therefore the least risky. High-interaction honeypots, the other hand, provide actual operating systems for attackers to interact. More safeguards and configuration testing may be required to lower the risk to acceptable levels [3].

It is worth noting that presently there has been no legal precedent established regarding whether an insecure system operator can be held liable for the misuse of their system by a hacker [15]. However, a measure of due diligence can be gained by placing a firewall in front of the honeypot that allows any and all incoming traffic while restricting egress traffic that could be used in upstream attacks. FTP, ICMP and DNS UDP are relatively safe outbound protocols [17].

4.2. Identifying Honeypots

Aside from the aforementioned banners and restricted egress protocols potentially tipping off attackers, conventional honeypots may expose red flags with which an intruder successfully deduces they are in a duplicitous environment. This is rarely a concern for conventional deployments, as the attacker has revealed himself through interacting with the honeypot, but what about research honeypots? Their value lies in monitoring the activities of unsuspecting attackers.

Scott D Smith, Smith24197@gmail.com

There exists a few reports that have observed an attacker, once alerted their activities are being monitored, switch from data theft tactics to destructive attacks in an effort to break the honeypot [3]. Additionally, hackers may feel pressured to update their techniques and tools if honeypots are frequently encountered, making it even harder to detect intrusion activities. Honeypots are therefore most beneficial when they alert defenders they have caught something, yet do not tip off the attacker in the process [17].

4.2.1. VMWare

VMWare has increasingly gained popularity as a honeypot tool. The desktop virtualization solution creates and hosts virtual machines (VMs), which emulate hardware drivers. A VMware virtual machine is also given a separate IP address from the host for semi-autonomous network communications. From outside the host system, a VM is perceived as an independent system, and is well suited for low-interaction honeypot operations.

The major failure of a VMWare honeypot, however, is they are easily identified through inspection of the MAC address. Recall that MAC addresses are bound to the network interface card and cannot be changed, and that the first three octets represent the vendor identification number. In the case of VMWare, the virtual MAC address will fall under one of three subsets [7]:

1. 00-05-69-xx-xx-xx
2. 00-0C-29-xx-xx-xx
3. 00-50-56-xx-xx-xx

The attacker has several means to check the MAC address [7]:

1. In a UNIX environment, run 'ifconfig –a' .
2. In a Windows environment, run 'ipconfig/all'.
3. 'arp –a' is also an option from the router.

4.2.2. Chroot

The use of the change root command (chroot) in a Unix environment is another common method for creating a honeypot. A chroot jail isolates processes from outside the directory tree, thereby limiting an attacker's access to the system. While similar to a

VM solution, the jail is runs on the same kernel as the main operating system. This gives an attacker the impression they are at the top level of the file system.

Despite these details, there remain some telltale signs an attacker may determine they are in a chroot jail. The first and simplest is to run 'ls–lia' on the root directory and review the inode of the '.' and '..' directories. For example, a standard system will display the two directories as [7]:

```
1.  2 drwxrxrx   21 root    root       2096 Feb 16 08:37 .
2.  2 drwxrxrx   21 root    root       2096 Feb 16 08:37 ..
```

When the same command is run in a chroot jail, the system will display a much higher value for the inodes of the chroot directory [7]:

```
1.  1553552 drwxrxrx   6 1000    100        4096 Mar 11 17:22 .
2.  1553552 drwxrxrx   6 1000    100        4096 Mar 11 17:22 ..
```

4.2.3. Honeyd

Honeyd is an open source daemon designed to emulate a variety hosts through packet manipulation. It works by listening for packets targeting specific IP addresses and ports and craft responses in accordance to a configuration file. A single host running Honeyd can emulate 65536 IP addresses. It operates on both Unix and Windows hosts, and may be outfitted with Perl scripts to emulate common services for greater usability. These capabilities make Honeyd a relatively economic and effective honeypot against potential hackers by creating a needle-in-a-haystack approach to finding the real host.

Emulation has several limitations when compared to genuine systems. Honeyd relies solely on IP and port addresses to formulate responses, and may respond to malformed packets. A response to a packet crafted with an invalid checksum, for example, may tip off an attacker that something is amiss. Script emulated services, such as troute telnet which is included in the standard Honeyd installation, have also been observed malfunction on occasion.

Semantic errors in Honeyd's configuration file are also a known issue. Pairing a Linux service to a Windows machine is an obvious mistake, but a cautious hacker may notice service version incompatibilities when compared to the emulated host. Spelling mistakes are also common giveaways in configuration files containing thousands of entries [7].

4.3. Honeypot Vulnerabilities

Some commercial software developers have been known to leave I/O backdoors open for the configuration of their product during runtime, specifically in the case of solutions not implicitly designed for security. VMWare is a notable example, and may be compromised or 'broken' through a few assembly commands [7]:

```
mov eax, VMWARE_MAGIC ; 0x564D5868

mov ebx, b; <parameter of command>

mov ecx, c ; <number of command>

mov edx, VMWARE_PORT ; 0x5658

in eax, dx
```

Some examples of the commands that can be used via this are as follows [7]:

1. 04h - Get current mouse cursor position
2. 05h - Set current mouse cursor position
3. 06h - Get data length in host's clipboard
4. 07h - Read data from host's clipboard
5. 08h - Set data length to send to host's clipboard
6. 09h - Send data to host's clipboard
7. 0Ah - Get VMware version
8. 0Bh - Get device information

By running the code above, a hacker could determine that they are inside a VMWare virtual machine. The backdoor could further be used for disruptions such as affecting mouse movement across the screen, or even creating a buffer overflow to break the VM. Despite the availability of patches that change the value of VMWARE_MAGIC in order to hide the backdoor, there is no documentation on how successful this method may be.

5. Trapping Techniques

Lance Spitzner wrote one of the earliest papers on the utility of honeypots as a compliment to traditional intrusion detection systems (IDS) in 1999 [17]. He summarizes his strategy in three simple steps:

1. Build the box. Recreate the system that you are interested in learning about vulnerabilities.
2. Connect it to the internet. Wait for an attacker to gain root access and observe.
3. Eject the attacker. It is an important, yet often overlooked, step that the operator must kick the attacker out of the system before they realize it is a honeypot. As previously discussed, the activities may otherwise change from covert information probing to full-out attacks, and vital information and resources may become destroyed [3].

5.1. Building a Better ~~Mousetrap~~ Honeypot

As it turns out, the principles that produce success in small game trapping are comparable to catching information system intruders. Retired SAS Sergeant-Major John Wiseman, a leading authority on wilderness survival, developed four fundamental points in wildlife trapping that are equally applicable to successful honeypot deployment. These points provide a framework for modern tactics recognized for luring in desired attackers, concealing the observer's methods for monitoring, and improving the honeypot's survivability and reusability [21].

5.1.1. Make the Trap Strong

It may be obvious that trap has little value if the intended prey destroys it attempting to escape. What is less apparent is the chance a sixty-pound coyote may trigger and wreck a snare rated for a forty-pound hare. While the trap could be reinforced to accommodate either target, it would require significantly more expensive materials, end up less suited for either type of prey, and still be vulnerable to damage from natural disasters or even larger game such as moose or bears. The bottom line is to be prepared that a trap may end up triggered and fruitless, or even irrecoverable, in use. This is why the best trappers set up several cheap and easy-to-reset snares at a time. Likewise, in

IDS, this is where the value of a virtual environment truly shines for honeypotting; simply reset, tweak and redeploy.

To prevent chances of a wasted catch, Spitzner recommends tracking a hacker by using multiple, or layered sources of information. This makes it more difficult for a panicked attacker to destroy evidence, and has the added benefit to cross-reference logs from separate sources in order to create a more detailed picture of the compromising activities. Traditionally, the best layers to rely upon are outside the system, since attackers will attempt replace the system log (syslog) daemon and erase or alter the log files after gaining root access [17]. Configuring the honeypot to send syslog data to a dedicated network server will ensure the integrity of the logs after the honeypot is compromised.

Boundary firewalls running a sniffer provide provides two more layers of information gathering. While an attacker may ignore the firewall logging, an open source sniffer such as snort or sniffit can be configured to pick up all screen captures and keystrokes, and the firewall logs will capture nmap scans that would not appear in syslog. Finally, a binary integrity tool such as Tripwire or OSSEC, stored on removable media will add further depth and redundancy to tracking activities.

An equally vexing concern in small game trapping is whenever wildlife turns a trap to their advantage; that annoying forty-pound coyote may make a dinner out of the snared hare before the hunter returns to retrieve it. In the case of honeypots, we do not want an attacker launching attacks from our own system. A firewall configured to filter egress traffic between the system and the internet can prevent such misuse, although an operator must exhibit care when building the rulebase. The attacker will become suspicious should they be unable to download their tool set after gaining access. If all outbound traffic denied, they will cover their tracks and leave. The balance between allowing enough outbound traffic for the hacker to operate while limiting their malicious capabilities upstream is imperative to a honeypot's success. Spitzner recommends allowing FTP, ICMP, and DNS (UDP) outbound as a successful starting rulebase [17].

5.1.2. Hide your Scent

Trappers have noticed that prey will quickly learn to avoid snares that have been touched barehanded. Animals have a sense of smell several times sharper than human is and are accustomed to the scents of their environment. So, too, will experienced hackers be tipped off should they detect their activities are being monitored.

Obscuring your syslog channels is effective in keeping honeypot transgressors in the dark. In the previous section, syslog configuration file (syslog.conf) was recompiled to send local log data to a remote server; however, this traffic may be sniffed out, provided syslogd is not modified or removed. Using a lesser-known protocol, such as IPX, will reduce the chances of syslog traffic being picked up. Many remote auditing solutions use encryption to mask network logging traffic, although this is not recommended for honeypots since it may look suspicious to the attacker. As a further method of concealing local logging configurations, syslogd can be recompiled to read from a renamed syslog.conf, such as /var/tmp/.conf. This will prevent the hacker from determining the actual destination of the remote log data.

5.1.3. Camouflage

Most mature prey will evade traps that look out-of-place. Choosing a name for your system that sounds like a common, yet valued, resource to hackers will increase the honeypot's attractiveness. File, mail and intranet servers are typical examples [17].

Once access is gained, it is important the honeypot resemble an authentic system to avoid tipping the attacker off. Critical directories and files may be probed in early reconnaissance to ascertain the value and authenticity of the system. A few dummy directories like the /dev and /proc, as well as dummy versions of standard OS files found within /usr and, will make the honeypot appear genuine. It is important to note that although the previous sections recommend changing syslogd to read from a hidden syslog configuration file, it is important to keep a standard version of syslog.conf pointing to all local logging on the system [17]. Periodic system and application updates will further sell the deception [7].

5.1.4. Avoid Disturbing the Environment

A trapper risks alerting game to his presence the more he frequently and carelessly checks his traps. This can decrease not only the effectiveness of a particular trap instance, but prey may associate the trapper with future iterations of that trap, thereby reducing reusability. Similar to attack signatures, hackers learn to associate normally innocuous signs with known honeypot deployments and share this information among the black hat community. This limits repeat business from specific hackers that may have new techniques to teach honeypot operators.

Given sufficient time, an attacker will realize he is in a honeypot. Remote syslog is a valuable information layer to observe compromise activities in near-real time without alerting the attacker directly. This allows the operator to assess when enough has been learned, and decide to kick the attacker off the system.

When booting the attacker off, Spitzner recommends sending a message to all logged-on users that the system is going down for routine maintenance before taking the honeypot offline. Such messages added to the system updates described in the section above will enhance the illusion of normalcy. Once the backdoors removed and vulnerabilities are fixed, the honeypot reconnected to the internet [17]. This increases the chances a hacker will revisit the honeypot, and perhaps demonstrate a new exploit.

6. Emerging Honeypot Technics

6.1.1. Cloud-based Honeypots

Cloud computing is a business and technology distribution model for on-demand access to a pool of shared of configurable computer resources through the internet. Software- (SaaS), platform- (PaaS) and infrastructure-as-a-service (IaaS) are the primary service categories that can be rapidly provisioned with minimal oversight and service provider interaction [12]. The scalability and transference of technical overhead costs and risk has made cloud computing an attractive solution for online applications and services start-up companies.

As more consumers subscribe to online data storage, more resources that are valuable become located in a single domain, making cloud service providers a juicy target for hackers. The technology model remains immature and several notable vulnerabilities have been reported. The Apple iCloud breach of several hundred prominent celebrity accounts is one notorious example. The breach was attributed to a weak password model that did not limit the rate of password entries or lock out access after a set number of failed attempts. This opened the door for brute force attacks [1], and severely weakened public confidence in cloud storage as a secure medium.

A unique solution proposed by Alert Logic recommends unused cloud resources are allocated as potential honeypots. By creating obvious vulnerabilities in IP space allocated to honeypots, a cloud provider could steer attacker attention away from clients. Furthermore, since any communications with the honeypots from the public domain is an indicator of illicit activity, those IP address can then be added to a blacklist for the edge firewalls to block incoming traffic [20].

6.1.2. Adaptive Honeypots

Adaptive, or self-configuring, honeypots are a solution in development for creating a reactive environment using the elements of game theory [19]. Dr. Gérard Wagener has noted a correlation between the permissiveness of a honeypot and the speed at which a hacker completes their goal and compared it to the time it takes the same attacker to give up on an overly restrictive honeypot. The academic objective is to keep hackers engaged longer with incremental challenges in order to reveal as much information as possible about themselves, such as motives, techniques and tactics. Machine learning is also leveraged so the honeypot 'learns', or statistically determines the most accurate and effective approach, through successive interactions with hackers that include blocking commands, returning erroneous messages, and even insulting the intruder in a quasi-reverse Turing test [4]. While it has yet to be applied to commercially available honeypots, adaptive versions may be encountered in the near future of internet security.

7. Conclusion

Honeypots are undoubtable a flexible tool that are becoming more prominent in information security. They can be customized in countless configurations to detect, deceive or counteract the activities of attackers. Despite this versatility, it is vital to remember that honeypots are complimentary to conventional security appliances, and is no substitute for firewalls, IDS/IPS and defense-in-depth techniques.

It is important to understand that implementation of a honeypot will always introduces some risk to the network as a whole. Critical design, testing and development are necessary to ensure production honeypots do not give an attacker an advantage, interfere with existing IDS/IPS activities or attract undue malicious attention from the Blackhat community [3]. Organisations must therefore take care to outline the goals they wish to achieve as well as the risks they are willing to accept, as with all defense in depth techniques, as in many cases honeypots may not in turn out to be the most effective or economic solution to improving the overall network security posture [2].

8. References

[1] Aurther, C. (2014). *Naked celebrity hack: experts focus on iCloud backup theory*. The Guardian. Retrieved April 4th, 2016: http://www.theguardian.com/technology/2014/sep/01/naked-celebrity-hack-icloud-backup-jennifer-lawrence

[2] Cole, E., & Northcutt, S. (2008). *Honeypots: a security manager's guide to honeypots*. SANS Institute. Retrieved January 26, 2016: http://www.sans.edu/research/security-laboratory/article/honeypots-guide

[3] Chismon, D. (2016). *Hunting with honeypots*. MWR Infosecurity. Retreived March 15, 2016: https://www.mwrinfosecurity.com/our-thinking/hunting-with-honeypots/

[4] Even, L. R. (2000). *Intrusion detection FAQ: what is a honeypot?* SANS Institute. Retrieved January 26, 2016: https://www.sans.org/security-resources/idfaq/honeypot3.php

[5] Farmer, B. (2015). *Honey-trap cyber spies stole Syrian rebel plans*. The Telegraph. Retrieved 25 Jan 2016: http://www.telegraph.co.uk/news/worldnews/middleeast/syria/11385317/Honey-trap-cyber-spies-stole-Syrian-rebel-plans.html

[6] Graves, R. (2015). *Honeypots and honeytokens for Webmail ID/IR*. SANS Institute. Retrieved January 25, 2016: https://www.sans.org/reading-room/whitepapers/incident/honeytokens-honeypots-web-id-ih-35962

[7] Innes, S. & Valli, C. (2006). *Honeypots: how do you know you're in one?* Edith Cowan University. Retrieved January 26, 2016: http://ro.ecu.edu.au/cgi/viewcontent.cgi?article=1027&context=adf

[8] Kabay, M.E. (2003). *Liability and ethics of honeypots*. NetworkWorld. Retrieved January 25, 2016: http://www.networkworld.com/article/2342377/network-security/honeypots--part-4.html

[9] Martin, W. W. (2001). *Honey pots and honey nets – security through deception*. SANS Institute. Retrieved January 26, 2016: https://www.sans.org/reading-room/whitepapers/attacking/honey-pots-honey-nets-security-deception-41

[10] Merkow, M. (2001). *Playing with fire: not so sweet honeypots*. CISSP. Retrieved January 25, 2016: http://www.ecommerce.internet.com/news/insights/outlook/article/0,,7761_559431,00.html

[11] Niem, J. (2006). *Enhancing IDS using Tiny Honeypot*. SANS Institute. Retrieved January 25, 2016: https://www.sans.org/reading-room/whitepapers/detection/enhancing-ids-using-tiny-honeypot-1665

[12] Nithin Chandra, S., & Madhuri, T. (2012). *Cloud security using honeypot systems*. International Journal of Scientific & Engineering Research [Vol 3, No 3]. Retrieved April 4th, 2016: http://www.ijser.org/researchpaper%5CCloud-Security-using-Honeypot-Systems.pdf

[13] Radcliffe, J. (2007). *Cyberlaw 101: a primer on US laws related to honeypot deployments*. SANS Institute. Retrieved January 25, 2016: https://www.sans.org/reading-room/whitepapers/legal/cyberlaw-101-primer-laws-related-honeypot-deployments-1746

[14] Saltzman, A. (2015). *Endless telemarketing calls, scams and how the big telcos could stop them*. Canadian Broadcast Corporation News. Retrieved January 26, 2016: http://www.cbc.ca/news/business/telcos-telemarketing-scams-spam-1.3334194

[15] Spitzner, L. (2003). *Honeypots: are they legal*? Symantic. Retrieved January 25, 2016: http://www.symantec.com/connect/articles/honeypots-are-they-illegal

[16] Spitzner, L (2003). *Honeytokens: the other honeypot*. Symantic. Retrieved March 7, 2016: http://www.symantec.com/connect/articles/honeytokens-other-honeypot

[17] Spitzner, L (1999). *To build a honeypot*. SCN Research. Retreived March 18. 2016: http://www.scn.rain.com/~neighorn/docs/security/misc/honeypot.html

[18] Walden, I. & Flanagan, A. (2003). *Honeypots: a sticky legal landscape?* Rutgers Computer & Technology Law Journal [Vol. 29, No. 2]. Retrieved January 27, 2016: https://www.questia.com/library/journal/1G1-106474528/honeypots-a-sticky-legal-landscape

[19] Wagener, G. & Dulaunoy, A. (2011). *Adaptive and self-configuring honeypots*. IEEE. Retrieved January 27, 2016: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5990710

Scott D Smith, Smith24197@gmail.com

[20] Winder, D. (2014). *How to use the cloud as a honeypot*. CloudPro. Retrieved April 4th, 2016: http://www.cloudpro.co.uk/cloud-essentials/cloud-security/4728/how-to-use-the-cloud-as-a-honeypot

[21] Wiseman, L (1986). *SAS survival handbook*. Harvill Press (UK).

[22] (2015). *Detecting mimikatz use on your network*. SANS ISC InfoSec Forums. Retrieved March 15, 2016: https://isc.sans.edu/forums/diary/Detecting+Mimikatz+Use+On+Your+Network/19311/

[23] *Entrapment*. Duhaime's Law Dictionary. Retrieved 11 Mar 2016: http://www.duhaime.org/LegalDictionary/E/Entrapment.aspx

9. Appendix – Selection of Honeypots

Name	Details / Source
CONPOT	An open source low interactive honeypot developed under the Honeynet Project. Specifically designed to emulate server side Industrial Control Systems (ICS), CONPOT includes a range of common industrial control protocols for emulating a vast and diverse honeypot environment. Also features the ability artificially delay service response times to mimic the behaviour of a system under constant load. Source: http://conpot.org/
Honeyd	Honeyd is an open source daemon designed to emulate a variety hosts through packet manipulation. It works by listening for packets targeting specific IP addresses and ports and crafts responses in accordance to a configuration file. A single host running Honeyd can emulate 65536 IP addresses. It operates on both Unix and Windows hosts, and may be outfitted with Perl scripts to emulate common services for greater usability. Source: http://www.honeyd.org/
KFSensor	Designed for Windows clients, KFSensor is pre-configured to monitor all TCP and UDP ports, along with ICMP. It is also configured with the emulation of common services. KFSensor also allows full packet dump available for additional analysis, using common tools such as Wireshark. Source: http://www.keyfocus.net/kfsensor/
Tiny Honeypot	Tiny Honeypot (thp) is a simple honeypot program based on iptables redirects and an xinetd listener. It listens on every TCP port not in use, logging all activity and providing some feedback to the attacker. The responders are entirely written in Perl, and provide enough interaction to fool most automated attack tools, as well as quite a few humans, at least for a little while. With appropriate limits (default), thp can reside on production hosts with negligible impact on performance. Source: http://www.securityfocus.com/tools/2771

Specter	SPECTER is a commercial smart honeypot-based intrusion detection system developed by Network Security. It offers decoy common internet services, such as SMTP, FTP, POP3, HTTP and TELNET, which falsifies response traffic over all TCP, UDP and ICMP ports, logs activity and notifies the system operator. SPECTER provides massive amounts of decoy content including images, MP3 files, email messages, password files, documents and various software. It dynamically generates decoy programs that will leave hidden marks on the attacker's computer. Source: http://www.specter.com/default50.htm
DTK	Deception toolkit is an open source honeypot. Written in Perl, DTK uses TCP wrappers to process incoming service requests on ports normally blocked ports. The operator may customise subroutines to log an attacker's activity and develop scripts to build flexible responses to input. Source: http://all.net/dtk/
GhostUSB	Developed by the Honeynet project, Ghost is a honeypot for malware that uses USB storage devices for propagation. It is able to capture such malware without signatures. Detection is achieved by emulating a USB flash drive on Windows systems and observing the emulated device. The assumption is that on an infected machine the malware will eventually copy itself to the removable device. Source: https://www.honeynet.org/node/871
LaBrea	An open source exertion of the FreeBSD project, LaBrea takes over unused IP addresses, and creates virtual servers that are attractive to worms, hackers, and other internet threats. The program answers connection attempts without opening sockets in such a way to prolong an indefinite authentication or resource request, wasting time and reducing the attacker's efficiency. SMTP and IP protocols are currently supported for Linux, Solaris and Windows environments. Source: http://labrea.sourceforge.net/labrea-info.html
Spamd	Spamd is a ISC-licensed lightweight spam-deferral daemon written under the umbrella of the OpenBSD project. Spamd works directly with SMTP connections, and supports features such as greylisting, minimizing false positives compared to a system that does full-body analysis. Source: http://www.openbsd.org/spamd/

Kippo	Kippo is a medium-interaction SSH honeypot written in Python. It is used to log brute force attacks and the entire shell interaction performed by an attacker Source: https://www.digitalocean.com/community/tutorials/how-to-install-kippo-an-ssh-honeypot-on-an-ubuntu-cloud-server

10. Appendix - List of Honeynet Project Chapters

Chapter	Description
Argentina Chapter	Members from Instituto Universitario Aeronáutico, Córdoba, Argentina.
Dutch Honeypots	Dutch Honeynet Chapter.
EG-CERT Chapter	Egyptian CERT.
Swizterland Chapter	Provide analytics and demonstrate threat intelligence relating to global and local threats in/around and towards Switzerland.
Croatian Chapter	Work for the Croatian government, helping them secure their country's information systems.
THE Chapter	Loosely organized alliance of Italian, German, and Dutch hackers devoted to the freedom of computing machinery worldwide.
Rochester Institute of Technology Chapter	We wish to create new tools that will create visual representations of network data obtained as well as improve existing tools.
Ukraine CERT chapter	Ukraine CERT chapter.
Malaysian Chapter	Malaysian Chapter.
HoneyNED Chapter	Netherlands Chapter.
Swedish Chapter	Swedish Chapter.
Irish Chapter	Irish Chapter.
Japan Chapter	Japan Chapter.
Kenya Chapter	Kenya Chapter.
California Chapter	California Chapter.
Turkish Chapter	Turkey Honeynet Project .
Indonesian Chapter	Indonesian Honeynet Project.
Tunisian Chapter	Tunisian chapter.
Polish Chapter	Polish Chapter.
Saudi CERT Chapter	Saudi CERT Chapter.
honeyTARG	honeyTARG chapter in Brazil.
South African Chapter	South African Chapter.
RoT-1 Chapter	RoT-1 Chapter .
OCERT Honeynet Project	OCERT Honeynet Project.
Sysenter Chapter	Sysenter Chapter.
Saudi Honeynet Chapter	Saudi Honeynet Chapter.

Group	Description
Vietnam Honeynet Chapter	Vietnam Honeynet Chapter.
Iranian Chapter	Iranian Honeynet Chapter; Research focus: Malware Analysis, Botnet Detection, Client-side attacks, and Web Honeypots.
Pacific Northwest Chapter	High interaction client honeypots for research of drive by downloads attacks. Application of deception theory in the research of drive by downloads.
United Arab Emirates Chapter	U.A.E. Chapter .
Indian Chapter	Indian Chapter .
Southern California ("SoCal") Chapter	SoCal Chapter; Research Focus: malware forensics,malware profiling.
Italian Chapter	Italian Chapter; Research Focus: Botnets (Tracking,Visualisations,Monitoring), Distributed Honeypots, Threat Intelligence.
UNAM Chapter	UNAM Chapter; Research focus: Data malware collection, Darknets, Honeynet infrastructure, Automated traffic log processing.
Spartan Devils Chapter	Spartan Devils Chapter; Research focus: motives, beliefs and social dynamics of the hacker community around the world.
Taiwan Chapter	Taiwan Chapter; Research Focus: Malware analysis.
French Chapter	French Chapter.
CyberSecurity Malaysia Chapter	Our efforts are currently focused on malware collection, web application honeypots and visualizing trends.
Hong Kong Chapter	Hong Kong Chapter; Research Focus: Low-Interaction Honeypots.
Canadian Chapter	Canadian Chapter; Research focus: Botnet analysis, malware analysis.
Hawaii Chapter	Hawaiin Chapter .
Brazilian Chapter	Brazilian Chapter .
Australian Chapter	Australian Chapter; Research Focus: tracking Malicious Activity in AU; data visualization.
Global Chapter	Global Chapter - we connect sparse researchers!
Alaskan Chapter	Alaskan Chapter;Research focus: visualization, HI honeypots, automated config and deployment of honeynets, virtualization, and VMI.

Group	Description
Mexican Chapter	Mexican Chapter; Research Focus: Malware Analysis & Bots/Botnets.
Singapore Chapter	Singapore Chapter .
New Zealand Chapter	New Zealand Chapter; Research Focus: Client Honeypots.
Chinese Chapter	Chinese Chapter; Research focus: client honeypot, high-interaction honeypot, malware col and analysis.
Czech Chapter	Czech Chapter .
Norwegian Chapter	Norwegian Chapter .
Pakistan Chapter	Pakistan Chapter; Research Focus: virtual honeynet, low-interaction honeynet, data analysis tools/UIs development.
UK Chapter	UK Chapter; Research focus: Distributed honeynets, Improving honeynet data analysis tools.
Giraffe Chapter	We are a development oriented honeynet chapter. Our main research interests are: low interaction honeypots, emulation, reverse engineering.

Scott D Smith, Smith24197@gmail.com