

The Evolution of Deception Technologies as a Means for Network Defense

```
RP C Who  
THER Typ  
RP C Who  
468x60:s
```

```
s=<nop,nop,tst  
pe=0806 (ARP),  
o is 10.0.0.42  
pe=0806 (ARP),  
o is 10.0.0.42
```

```
{3249>
```



Abstract

Over the past several years networked systems have grown considerably in size, complexity and susceptibility to attack. At the same time, the knowledge, tools, and techniques available to attackers has also grown in proportion. Unfortunately defensive techniques have not evolved as quickly due to the reactive nature in which they are used. The current security technologies are reaching their limitations and innovative solutions are required to deal with current and future classes of threats.

This paper provides an examination of an emerging class of security mechanisms often referred to as *deception technology* or *honeypots*. It will provide an overview of the various technologies and techniques, examining the strengths and weaknesses of a number of approaches and explore their suitability within various network environments. It will also discuss deployment criteria and strategies and provide a brief overview of the advantages of a layered security approach using deception based intrusion detection.

The Use of Deception as a Defense

“All war is deception” - Sun Tzu

Deception techniques have been used throughout history. For thousands of years military leaders have used deception to win battles. The ancient Egyptian pharaoh, Rameses II, was defeated at the battle of Kadesh because a Hittite deception lured him into an ambushⁱ. During World War II the Germans were led to believe that the real invasion would occur at the Pas de Calais instead of at Normandy. Even after the landing at Normandy Hitler was convinced it was a feint and failed to respond in time.ⁱⁱ During operation Desert Storm the US used dummy soldiers, camps, and even tanks to distract the Iraqi army while real soldiers entered Iraq virtually unopposed.ⁱⁱⁱ

The same techniques used in warfare may also be applied to the defense of networked assets. Currently, the Internet provides today's attackers with a common knowledge base. Attackers can calmly research new vulnerabilities and take the time to find systems that exhibit these holes. By downloading an automated exploit even the most novice attacker can appear to have the skills of an expert. Information about circumventing firewalls and Intrusion Detection Systems (IDS) can be found with the click of a button. In addition, automation means that attackers can effectively spend months looking for holes in defenses without any interaction that might otherwise gain attention. Finally, the interconnected nature of the Internet means attackers from all over the world can attack any system they choose. On the other hand, the average corporate security professional is multitasked with daily system administration, resolving end-user problems and installing a myriad of security applications that do not provide interoperability. This discrepancy gives the attacker the advantage.

An attack need only succeed once. However, security professionals must defend against all current and future attacks and attackers. They must find and fix all vulnerabilities before an attacker does without affecting any operational network services and they must immediately detect and respond to any suspected compromise. Even a false alarm consumes large amounts of time. Responding to a successful attack is nearly impossible without first figuring out what the attacker was after and how far they penetrated into the network. Finding this information after the fact is a long and error prone effort.

“If all you ever do is defend you will be defeated.” – Sun Tzu

Traditional security techniques attempt to block attacks (firewalls) or detect them as they happen (IDS). Both of these techniques are critical but they have their limits. Given enough time and information an attacker can learn to circumvent a firewall. Once circumvented the firewall offers no further protection. An IDS will only provide information once an attack has begun. Often this does not give enough time to adequately secure all vulnerable systems. In addition, an IDS cannot determine if a new attack succeeded or if it would succeed against other systems. Using only firewalls and IDS is analogous to a medieval city defending against the barbarian hordes with only high walls and unarmed sentries. Eventually the city will fall.

A successful counter measure would cost the attacker more time than the defender and give the defender enough information about the attacker to prevent the attack from causing damage. Successful use of deception accomplishes these goals. Using deception costs the attacker time in fruitless attacks and feeds them false information; thus blunting future attacks. In addition, a good deception will give the defender information about the attacker’s means and motives without the large cost of a successful exploit. This information can then be used to enhance existing security measures such as firewall rules and IDS configurations. The first deployments of network deception were called honeypots.

The Evolution of Network Deception

Honeypots are not a new idea. Much like a pot of honey used to attract and trap insects, a honeypot can be deployed to present an attractive target to an attacker. Researchers and security professionals have been using different forms of honeypots since computers were interconnected. The first honeypots were simply computers used for no other purpose than to be attacked. These sacrificial lambs provided excellent targets for attackers. Unfortunately, extracting the attack data is time consuming and the *sacrificial lamb* can be used by the attacker to attack other machines.

Using a honeypot has numerous advantages. First, it wastes the attacker’s time. Depending on the depth of the deception, an attacker can spend large amounts of time attempting to exploit and then exploring the honeypot. Any time spent attacking a honeypot is time not spent attacking a real machine. Second, it gives the attacker a false impression of the existing security measures. Thus the attacker spends time finding tools to exploit the honeypot that may not work on a real system. And thirdly, the existence of a honeypot decreases the likelihood that a random attack or probe will hit a real machine.

Many attackers scan large blocks of computers looking for victims. Even attackers targeting a specific organization will scan the publicly accessible machines owned by the organization looking for a machine to compromise as a starting point. Using honeypots decreases the chance an attacker will choose a valuable machine as a target and they will detect and record the initial scan as well as any subsequent attack.

Unlike other intrusion detection measures, there are no false positives with a honeypot. IDS products produce false positives to varying degrees. This is because there is always a chance that valid traffic will match the characteristics the IDS uses to detect attacks. This is not the case with a honeypot as any communication with a honeypot is suspect because it is not used for any purpose other than detecting attacks. In other words, there is no invalid traffic to produce false positives.

In this way a honeypot can detect more attacks than any other IDS solution. New vulnerabilities can be found and analyzed because all actions an attacker takes are recorded. Since all communication with a honeypot is suspect, new attack tools can be detected based on their interaction, even so-called layer eight attacks or attacks against the information flow rather than the programs or protocols. These can include feeding false information into a service or database or using compromised credentials to gain unauthorized access. Finally, a honeypot can detect and record incidents that may last for months. These so-called ‘slow

scans' are difficult to detect using an IDS as the time involved makes them appear to be normal traffic.

The next advance in honeypot technology removed the security threat posed by a compromised honeypot by only emulating network services instead allowing the real machine to be attacked. These facades generally have the vulnerabilities of sacrificial lambs but they do not provide such a rich a set of data. Facades provide easier access to the recorded attack data and are therefore harder for attackers to avoid detection.

Instrumented systems build on the strengths of both sacrificial lambs and facades. Like the sacrificial lambs they provide a highly believable system for attackers to compromise. Like facades they are easily accessible and difficult to evade due to their logging of attack information. Finally, an advanced instrumented system provides a means to prevent the attacker from using the system as a base for further attacks.

CLASSIFICATION OF HONEYPOTS

Honeypots can be classified into three primary categories: sacrificial lambs, facades and instrumented systems. A *sacrificial lamb* usually consists of an “off the shelf” or “stock” system placed in a vulnerable location and left as a victim. A *facade* is the most lightweight form of a honeypot and usually consists of some type of simulation of an application of service in order to provide the illusion of a victim system. An *instrumented system* honeypot is a stock system with additional modification to provide more information, containment, or control.

The technology used in sacrificial lambs and network facades is somewhat restrictive and can be a limiting factor for detection, while an instrumented system addresses many of the issues faced by both of these tools to provide an integrated intrusion detection solution. The sections below explore each class with respect to implementation, each strengths and weaknesses and typical uses.

SACRIFICIAL LAMBS

A sacrificial lamb is an “off the shelf” system left vulnerable to attack. They can be built from virtually any device (a Linux server, a Cisco router, etc). The typical implementation involves loading the operating system, configuring some applications and then leaving it on the network to see what happens. The administrator will examine the system periodically to determine if it has been compromised and if so what was done to it. In many cases, the only form of data collection used is a network sniffer deployed near the honeypot. While this provides a detailed trace of commands sent to the honeypot, it does not provide any data in terms of host effects. In many cases, additional examination is done either by hand or using various third-party forensic tools. Also the systems themselves are “live” and thus present a possible jumping off point for an attacker. Additional deployment considerations must be made to isolate and control the sacrificial lamb by means of firewalls or other network control devices.

Sacrificial lambs provide real targets. All the results are exactly as they would be on a real system and there is no “profiling” possible since there is nothing that differentiates this system from any other. These types of honeypots are also fairly simple to build locally since they only use off-the-shelf components. Sacrificial lambs provide a means to analyze a compromised system down to the last byte with no possible variation. However, this type of honeypot requires considerable administrative overhead. The installation and setup requires the administrator to load the operating system themselves and manually perform any application configuration or system hardening. The analysis is manual and often requires numerous third-party tools. They also do not provide integrated containment or control facilities This will also require additional network considerations (as mentioned above) to deploy in most environments, and would require dedicated expert security resources to manage and support, with advanced expertise to analyze the data in the instance of attack, which most organizations may consider as an undesirable trait.

FACADES

A network facade is a system that provides a false image of a target host. It is most often implemented as software emulation of a target service or application. When the facade is probed or attacked, it gathers information about the attacker. This is similar to having a locked door with nothing behind it and watching to see who attempts to open it. The depth of the simulation varies depending on the success of the implementation. Some will provide only partial application level behavior (e.g. banner presentation) and others will actually simulate the target service down to the network stack behavior. This is done in order to prevent remote signaturing by some form of O/S fingerprinting. The value of a facade is defined primarily by what systems and applications it can simulate and how easy it is to deploy and administer.

Facades offer simple, easy deployment as they often require minimal installation or equipment requirements and can provide a large number of targets of considerable variety. Since they are not real systems, they do not have the vulnerabilities of real systems. They also present no real additional risk to your environment since they are not complete systems and cannot be used as a jumping off point. Their only significant limitation is that they provide only basic information about a potential threat and are typically used by small to medium enterprises or by large enterprises in conjunction with other technology.

INSTRUMENTED SYSTEMS

Instrumented systems provide an ideal compromise between the low cost of a facade and the depth of detail provided by a sacrificial lamb. Engineered by a skilled team of developers focused on security, commercially available Instrumented systems are now both easy to deploy and manage at the end-user level. By performing extensive OS and kernel level modifications and application development to a stock system, commercial companies have evolved the concept of the honeypot as an effective means of network defense to include advanced data collection, attack containment, policy-based alerting and enterprise administration functionality.

Deep deception honeypots are an evolutionary step from the earlier forms of deception as they are able to provide an exceptional level of attack detail while providing a very plausible, highly interactive environment that keeps an attacker “interested” for a greater length of time. Time that is critical in helping an administrator find out attacker’s motives, and the time to implement the counter-measures to ensure future compromise of networked assets.

Security professionals interested in instrumented systems should consider one designed by security professionals with significant honeypot experience, and a vendor that provides a software product that is supported with product updates and technical support. Enterprise deployments that require more attack information than a facade provides, but that cannot afford the large administrative overhead of a sacrificial lamb system, should consider an instrumented system honeypot. These are typically used by medium to large enterprises.

ADDITIONAL CONSIDERATIONS

While not specific to a particular class or form of honeypot, there are a number of additional features or functions which should be considered by an organization evaluating honeypots.

It is important to consider the nature and the cost of containment and control. Any system deployed in a network presents possible risk. Measures should be taken to mitigate that risk. If the product does not support any native containment and control, the cost and complexity of implementing it should be seriously considered.

While honeypots can provide an excellent source of data, it's important to remember that the data by itself does nothing. In order to be useful, the data must be analyzed. Some products provide integrated analysis, reporting and alerting. Others require the administrator to provide the data review and security expertise. How much analysis is offered and how the administration is done is an important consideration and has significant impact on the cost of using such a system.

Cluster or group administration functionality should also be considered when deploying multiple deception devices. Systems that provide the ability to work in clusters and have single points of administration and reporting are a much more scalable solution than those that require manual operation of each node.

Maintenance of content and restoration of the honeypot should also be evaluated. These both contribute to the ongoing administrative cost of maintaining a deception system. Content on a deception device will need periodic updates, so to appear valid and "live". Deception systems which have been attacked may also need to be periodically restored to a "clean" state. In both of these cases, solutions which allow automated capabilities for this procedure can greatly reduce your administrative costs.

Finally, it's worth considering the relationship of honeypots to *host-based intrusion detection systems* (HIDS) and *integrity monitoring systems*. HIDS are usually deployed on a production system and designed more as a burglar alarm. Running these on a production system really doesn't provide the same value as a honeypot. They are much more prone to false positives, force the administrator to deal with the difficult of monitoring normal user activity, and in general, do not provide containment or good administration functionality (for a honeypot approach). These can be used to create honeypots, but often produce very large signatures since they are not designed for stealth.

Integrity monitoring software has many of the same deficiencies as HIDS for honeypot use. It designed for monitoring a production system for change, not user activity or security. It provides none of the additional functionality needed for a honeypot. As with a HIDS, these also create very large signatures which are not desirable for a honeypot.

Deployment Strategies

While many honeypot implementations may function well in single deployments with dedicated administrative efforts, larger deployments (a.k.a. "enterprise deployments") require additional functionality to be effective solutions. An organization that wishes to deploy honeypots should have an overall computer security policy that state what the threats are, what the main goals for an attacker might be, where the high-value systems are, and how potential targets will be protected. In essence, the security policy will dictate what the strategy of honeypot deployment should be.

This section describes a few different deployment strategies. These strategies, or combinations of them, can be used together with firewalls and IDS to form a cohesive security infrastructure to protect an organization.

MINEFIELD

In a minefield deployment, honeypots are installed among live servers, possibly mirroring some of the real server data on the honeypots. The honeypots are placed among external servers in the DMZ to capture attacks against the public servers and/or in the internal network, or internal attacks (which either originated from an internal or external source, penetrating the firewall and using internal machines as launching pads to attack other systems).

Attacks are rarely restricted to a single machine. Many manual and automated network attacks follow the same pattern: Assuming a successful attack has taken place on one machine in the network, that machine is then used to scan the network for other potential targets, which are subsequently attacked. For manual attacks, this takes some time, while worms will normally execute the scan just seconds after the first infection. Stealth scanning can be performed in a manner that specifically avoids setting off IDS systems (e.g., through "slow scans"), but honeypots in a minefield will be alerted.

For example, if a network has one honeypot for every four servers, then the chances of hitting a honeypot with a random, single-point attack is 20%. In reality, the chances are significantly better than that, because in most cases an entire block of network addresses will be scanned. When this happens, it is practically guaranteed that the honeypot will detect the intrusion shortly after any machine on the network has been compromised.

Even though the intrusion detection aspect alone is important, another feature of using honeypots is to see what the attack tools are, and what the purpose of the attack is. With good security practices on the production machines (i.e. good password policies, no plain text passwords over the network, machines running the latest vendor patches, etc), slightly decreasing the security on the honeypots themselves may increase the chance that they will be some of the first machines that are attacked. A well-designed honeypot will then have the information about the services attacked, how that service was attacked, and - if the attack was successful -- what the intruder did once inside. Having the honeypots configured exactly the same way as the regular servers, however, has other advantages. It increases their deception value slightly, and it also means that when a honeypot has detected a successful attack, that attack is likely to succeed also on the production hosts.

SHIELD

In a shield deployment, each honeypot is paired with a server it is protecting. While regular traffic to and from the server is not affected, any suspicious traffic destined for the server is instead handled by the honeypot shield. This strategy requires that a firewall/router filters the network traffic based on destination port numbers, and redirects the traffic according to the shielding policy.

For instance, consider a web server deployed behind a firewall. Web server traffic will be directed to the web server IP address on TCP port 80. Any other traffic to the web server is considered suspicious, and can be directed towards a honeypot.

The honeypot should be deployed in a DMZ, and to maximize the deception value, it may replicate some or all of the non-confidential content of the server it is shielding. In the example of the web server, this is merely a matter of mirroring some or all of the web content to the honeypot.

In conjunction with the firewall or router, honeypots deployed in this fashion provide actual intrusion prevention in addition to intrusion detection. Not only can potential attacks be detected, they can be prevented by having the honeypot respond in place of the actual target of the attack. It should be added that a honeypot shield can not protect a mail server from SMTP exploits, nor a web server from HTTP exploits, since "regular" traffic must be able to reach its target. However, since live servers generally need very few open ports, it is reasonably easy to find the point of an attack -- both for prevention and forensic purposes -- and all other ports lead straight to the honeypot, where the attack can be analyzed in detail.

A shield deployment is an example of how honeypots can protect a high-value system where attacks can be expected.

HONEYNET

In a honeynet deployment, a network of honeypots imitates an actual or fictitious network. From an attacker's point of view, the honeynet appears to have both servers and desktop machines running many different types of applications on several different platforms. Another term for this deployment is "zoo", as it displays a variety of honeypot species.

A honeynet is an extension of the honeypot concept in that it takes multiple deception hosts (single honeypots), and turns it into an entire deception network. A typical honeynet may consist of a mix of facades (because they are light-weight and reasonably easy to deploy), some instrumented systems for deep deception and possibly some sacrificial lambs. In order to provide a reasonably realistic network environment, some sort of content generation is necessary. On a host basis this involves simulating activity on each deep honeypot as well as generating network traffic to and from the clients and servers, so that the network itself looks realistic from the outside.

As an example, a DMZ that contains a web server and a mail server could deploy two honeypots that act as shields to the servers. Any traffic to the web server that is not HTTP traffic will be directed to the web server's shield. Any traffic to the mail server that is not SMTP will be directed to the mail server's shield. By adding a few more honeypots, another dimension can be added to this deception; all traffic to unknown IP addresses can be directed to honeypots, not only traffic to known hosts. The strength of the honeynet shield is that it shields an entire network instead of a single host. Similarly, honeynet minefields represent the scenario where each mine is an entire network, as opposed to just a single honeypot.

Honeynets can be useful in a large enterprise environment and offer a good early warning system for attacks. A honeynet may also provide an excellent way to figure out an intruder's intention, by looking at what kind of machines and services are attacked, and what is done to them. The Honeynet Project (<http://project.honeynet.org>) is an excellent example of a honeynet used as a research tool to gather information about attacks on computer infrastructure.

Recourse ManTrap

ManTrap is an enterprise-scale early warning system that can be implemented as a stand-alone solution or as an extension to other intrusion detection solutions. ManTrap enables organizations to monitor and track intrusion activities in real-time. By creating a realistic mock network, IT administrators can easily identify an intruder, safely monitor their actions or terminate the session immediately. All intrusion information is captured by ManTrap so both the actual attack and the attacker's apparent motivations and methods can be analyzed. ManTrap uncovers the methods and motives of the attacker, allowing organizations to take the appropriate actions to prevent against future attacks. Because of its sophisticated early warning system, ManTrap can successfully detect and prevent against "Zero Day" threats, threats that have not been publicly identified or named, without requiring prior knowledge of attacks.

ManTrap runs on a Solaris 7 and 8, on both SPARC and Intel machines. Each physical machine can provide up to four different honeypots -- or "cages" -- with each cage being completely isolated from the other cages as well as the real host system. A user logged into a cage will not be able to see the processes, network traffic, and other system resources of the other cages, nor of the host system itself. To the attacker, each cage appears to be a separate machine. If a system file is deleted in one cage, the activity logs used to capture evidence of misuse are protected in the underlying system and cannot be compromised.

If an attacker gets in to a cage, whether by a stolen password, remote network exploit, or other means, the cage will provide a controlled environment where information is gathered about the activity, while at the same time containing the attacker, and stopping him from discovering that he is being monitored.

ManTrap also provides a module that automatically generates email traffic to and from some of the users on the system. This provides an additional piece of deception, as an intruder may be fooled into thinking he is capturing actual email traffic. The generated email messages are instead created from templates provided by the ManTrap administrator.

The ManTrap administration console allows the user to administer several ManTrap machines at once.

MANTRAP LOGS

ManTrap keeps extensive logs of activities in its cages. Since all activity in a cage is suspicious (because no legitimate users belong there), as much information as possible is logged. Examples of the activities that a running ManTrap will log:

- All terminal input and output
- All files opened for writing
- All device accesses
- All processes that are started
- All network activity

The ManTrap logs are meant to provide an (almost) complete view of the activities inside the cage. ManTrap also allows the administrator to cryptographically verify that the logs have not been tampered with.

RESPONSES

When ManTrap notices any sort of activity, it is capable of alerting the ManTrap administrator of what is going on and/or responding to it. The administrator can configure what kind of activity will trigger responses, and what kind of responses will be triggered. ManTrap 3.0 supports the following types of responses:

- SMTP (E-mail) alerts
- SNMP traps (alerts to network management software)
- Integration with other threat management solutions from Recourse Technologies
- Custom responses: administrator-specified scripts or binaries to be run on a particular event

A simple example of effective use of response is to alert the security personnel by an urgent email (possibly in addition to calling a pager) as soon as some unusual activity is happening in the cage. Another example is to completely shut down a cage once a successful login has happened. In the latter example, little information may be gained about the purpose of the attack, but it guarantees that the cage can not be used as a launch pad of other attacks throughout the network.

ANALYSIS

The log data that is collected inside a cage is used to provide different types of activity reports. Reports can be generated on-demand or on a scheduled, regular basis, and cover cage activities such as:

- File modifications
- Successful logins to the cage
- Responses triggered by the cage
- Attempted connections
- Outgoing connections
- TCP and/or UDP port activity on the cage

In addition, the ManTrap administration console allows a user to be able to monitor interactive sessions in a terminal window, either while the session is going on, or after the fact. This gives the ManTrap administrator a unique and realistic view of what the intruder saw and did during the attack.

Summary

Deception technologies are an important, emerging security technology. Deception provides the defender with both the time and information needed to effectively respond to a wide variety of threats. Unlike the first generation technology concepts, a commercially available solution such as the Recourse Technologies ManTrap has evolved the honeypot concept of intrusion detection to a third generation. Combining early detection, simplified deployment and advanced reporting and analysis, Recourse ManTrap provides a powerful, cost-effective defense mechanism to prevent internal and external intrusions and should be a component of any enterprise security solution.

ⁱ Dunningan, James F. Nofi Albert A., *Victory and Deceit: Dirty Tricks at War* (New York, NY: William Morrow & Co., 1995)

ⁱⁱ Breuer, William B., *Hoodwinking Hitler: The Normandy Deception* (Westport, CT: Praeger, 1993)

ⁱⁱⁱ Dunningan, James F. Nofi Albert A., *Victory and Deceit: Dirty Tricks at War* (New York, NY: William Morrow & Co., 1995)



WE GIVE YOU RECOURSE AGAINST HACKING

1.877.786.9633

info@recourse.com

www.recourse.com

Recourse Technologies and ManTrap are registered trademarks and ManHunt is a trademark of Recourse Technologies, Inc. All other registered or unregistered trademarks are the sole property of their respective owners. © 2002 Recourse Technologies, Inc. All rights reserved.

51602-1100