Intrusion Detection Systems:
The Evolution of Deception Technologies as a Means for Network Defense

Inside

› The Use of Deception as a Defense
› The Evolution of Network Deception
› Deployment Strategies
› Symantec™ ManTrap™
Contents

Abstract ................................................................. 3
The Use of Deception as a Defense ............................... 3
The Evolution of Network Deception ............................. 4
  Classification of Honeypots ...................................... 5
  Sacrificial Lambs .................................................. 5
  Facades ............................................................. 6
  Instrumented Systems ............................................ 6
  Additional Considerations ....................................... 7
Deployment Strategies ............................................... 8
  Minefield .......................................................... 8
  Shield .............................................................. 8
  Honeynet ......................................................... 9
Symantec ManTrap .................................................. 10
  Mantrap Logs ..................................................... 10
  Responses ........................................................ 11
  Analysis .......................................................... 11
Summary .............................................................. 11
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 have also grown in proportion. Unfortunately, defensive techniques have not evolved as quickly. Current security technologies are reaching their limitations, and more innovative solutions are required to deal with current and future classes of threats.

This paper provides an examination of an emerging class of security technologies often referred to as “deception mechanisms” or “honeypots.” It will provide an overview of the various technologies and techniques, examining the strengths and weaknesses of a number of approaches while exploring their suitability within various network environments.

Furthermore, the paper 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 warfare is based on deception” - Sun Tzu

For thousands of years, military leaders have deceived their opponents in order to win battles. The ancient Egyptian pharaoh, Rameses II, lost the battle of Kadesh when a Hittite deception lured him into an ambush. 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. During operation Desert Storm, the United States used dummy soldiers, camps and even tanks to distract the Iraqi army while real soldiers entered Iraq virtually unopposed.

The same techniques used in warfare can also be applied to defend networked assets from today’s savvy attackers. The Internet now provides attackers with a very powerful and widely accessible resource which they can use to wage a new kind of war on the enterprise. For example, attackers can use the Internet to easily research new vulnerabilities. Or, by downloading an automated exploit, a novice attacker can launch the type of devastating attack that used to require significant expertise. Even 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. And finally, the interconnected nature of the Internet means attackers from all over the world can strike any system they choose.

An attack need only succeed once. Security professionals, however, must defend against all current and future attacks and attackers. They must find and fix all vulnerabilities before an attacker acts—without affecting any operational network services. And they must immediately detect and respond to any suspected compromise. Even a single false alarm consumes large amounts of time. What’s more, responding to a successful attack is nearly impossible without first determining what the attacker was after, how deeply he penetrated the network and by what means he accomplished the attack. Finding this information after the fact is a time consuming and error-prone effort, especially considering that the average corporate security professional is already over loaded with daily system administration, end-user problem resolution and the maintenance of myriad security applications that do not provide interoperability. All in all, these discrepancies give the attacker a serious advantage.
“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 leave 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 countermeasure would substantially delay the attacker while giving the defender enough information about his enemy to prevent the attack from causing damage. Successful use of deception accomplishes these goals. By deceiving the attacker, the defender feeds him false information and forces him to waste time in fruitless assaults, thereby blunting future attacks. In addition, a good deception will give the defender information about the attacker’s means and motives without the risk or penalty of a successful exploit. This information can then be used to enhance existing security measures such as firewall rules and IDS configurations.

The Evolution of Network Deception

Honeypots, are not a new idea. Researchers and security professionals have been using different forms of honeypots for many years. Much like an actual pot of honey used to attract and trap insects, a computer honeypot can be deployed to present an attractive target to an attacker.

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—and any time spent attacking a honeypot is time not spent attacking a real machine. Secondly, it gives the attacker a false impression of the existing security measures. He or she may spend 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. Finally, it provides extremely detailed information about what the attacker does and how they do it.

Many attackers scan large blocks of computers looking for victims. Even attackers targeting a specific organization will scan the publicly accessible machines operated 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. The honeypot also provides a very effective means to record this reconnaissance effort as well as any successful attacks.

Unlike other intrusion detection measures, there are no false positives with a honeypot. All traditional 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. Any communication with a honeypot is suspect simply because the device is not used for any purpose other than detecting attacks. In other words, there is no invalid traffic to produce false positives.
Honeypots are also capable of detecting attacks that other forms of IDS are not. 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 new or unknown attacks that exhibit no signature or anomalous characteristics. 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.

CLASSIFICATION OF HONEYPOTS

Honeypots can be classified into three primary categories: sacrificial lambs, facades and instrumented systems.

- The first honeypots to be used, sacrificial lambs are simply computers deployed with the sole purpose of being attacked. A sacrificial lamb usually consists of an “off the shelf” or “stock” system placed in a vulnerable location and left as a victim.

- To address some of the limitations of sacrificial lambs, a new class of honeypots were created: facades. 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 even newer class of honeypots, instrumented systems, build upon the strengths of both sacrificial lambs and facades. An instrumented system honeypot is a stock system with additional modification to provide more information, containment, or control.

The sections below explore each class with respect to implementation, strengths and weaknesses, and typical uses.

SACRIFICIAL LAMBS

A sacrificial lamb is an “off the shelf” system left vulnerable to attack. It can be built from virtually any device (a Linux server, a Cisco router, etc.). A 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 other 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 administrators 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. In addition, sacrificial lambs do not provide integrated containment or control facilities. They will also require additional network considerations (as mentioned above) to deploy in most environments, and will require dedicated expert security resources to manage and support, with advanced expertise to analyze the data in the instance of attack. Most commercial organizations would consider a sacrificial lamb too risky and resource intensive to deploy.

FACADES

A network facade is a system that provides a false image of a target host. It is most often implemented as a 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 tries to open it. The depth of the simulation varies depending on the implementation. Some will provide only partial application-level behavior (e.g. banner presentation), while 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 efforts and equipment, 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 because they are not complete systems and cannot be used as a jumping-off point. Their one significant limitation is that they provide only basic information about a potential threat and are therefore typically used by small- to medium-sized 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. Created by professional developers focused on security, commercially available instrumented systems are now easy to both deploy and manage at the end-user level. By performing extensive O/S- 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.

These deep-deception honeypots are an evolutionary step up from the earlier forms of deception. 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 determine an attacker’s motives, and in implementing the countermeasures to ensure no 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 with regular 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 that should be considered by an organization evaluating honeypots. First, 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 examined.

Next, 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 much more scalable solutions 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 that have been attacked may also need to be periodically restored to a “clean” state. In both of these cases, solutions that 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 difficulty of monitoring normal user activity, and in general, do not provide containment or good administration functionality (for a honeypot approach). They can be used to create honeypots, but often produce very large signatures because they are not designed for stealth.

Integrity monitoring software has many of the same deficiencies as HIDS for honeypot use. It is designed for monitoring a production system for change, not user activity or security. It provides none of the additional functionality needed for a honeypot. And, as with a HIDS, it also creates very large signatures that 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 states 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 should dictate what the strategy of honeypot deployment will 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. 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, whereas 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 (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, theoretically, 20%. In reality, however, 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 (e.g., 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, but they can also be prevented by having the honeypot respond in place of the actual target of the attack. It should be added that a honeypot shield cannot 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, because 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 lightweight 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, instead of simply 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.
Symantec™ ManTrap™

Symantec ManTrap is an "instrumented system" honeypot. It provides an enterprise class real-time attack detection and analysis 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 intruders, safely monitor their actions, or terminate their sessions immediately. All intrusion information is captured by ManTrap so both the actual attack and the attacker's apparent motivations and methods can be analyzed, and organizations can then take the appropriate actions to prevent future attacks. Because of its sophisticated 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 decoys—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 into 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. At the same time, the cage contains the attacker and stops 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 created instead 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. Because all activity in a cage is suspicious (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 altered.
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 they will be. ManTrap 3.0 supports the following types of responses:

- SMTP (email) alerts
- SNMP traps (alerts to network management software)
- Integrates with other network security solutions from Symantec
- Custom responses: administrator-specified scripts or binaries to be run on a particular event

A simple example of an effective 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 cannot be used as a launch pad for 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 they can 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 represent an important, emerging security technology. They provide the defender with both the time and information needed to effectively respond to a wide variety of threats. Commercially available solutions, such as Symantec ManTrap, have evolved the honeypot concept of intrusion detection to a third generation. Combining early detection, advanced reporting and analysis, and an easy-to-deploy security solution, Symantec ManTrap provides a cost-effective defense mechanism powerful enough to prevent internal and external intrusions—and one that should be a component of any successful security solution.


About the Authors

Brian Hernacki is an architect in the Symantec Research Labs where he works with a dedicated team to develop future technologies. Hernacki has more than 10 years of experience with computer security and enterprise software development. Additionally, Hernacki has conducted research and commercial product development in a number of security areas including intrusion detection and analysis techniques.

Hernacki previously led the design and architecture of products and the investigation and research of new technologies at Recourse Technologies. Previously working at Recourse Technologies, Hernacki served as a Senior Software Developer, Group Manager and Product Architect at Netscape Communications Corporate, where he played a pivotal role in the development of a number of high-end enterprise and service provider server products.

Prior to Netscape, Hernacki’s experience included engineering and management positions at Computer Aided Engineering Network (CAEN) where he developed a network wide intrusion detection system and maintenance and system reliability tools.

Hernacki earned a bachelor of science degree in Computer Engineering, with honors, from the University of Michigan.

Jeremy Bennett is a development manager at Symantec and leads the design and architecture of the ManTrap product. Bennett has over 10 years of experience with computer security and enterprise software development. He also assists with the design of Symantec’s other IDS products and Jeremy promotes the use of deception techniques within the security community.

Prior to Symantec, Bennett was an architect and manager for Recourse Technologies where he played a key role in the design and development of Recourse’s products. In addition to developing software at Recourse, Jeremy performed extensive research in the areas of deception, intrusion detection, and computer forensics. Jeremy has also held management, architecture, and development positions at Hewlett-Packard and Saga.com.

Bennett earned a degree, in computer science, with honors from the University of Michigan.

Thomas Lofgren is a senior software engineer at Symantec where he contributes to the design and development of the intrusion detection products. Lofgren has 8 years of experience in computer security and software development.

Prior to Symantec, Lofgren was an engineer at Recourse Technologies, where he worked on development of deception hosts and other intrusion detection software. Thomas has also had a development position in Hewlett-Packard, as well as a research position at SICS (Swedish Institute of Computer Science).

Symantec, the world leader in Internet security technology, provides a broad range of content and network security software and appliance solutions to individuals, enterprises and service providers. The company is a leading provider of virus protection, firewall and virtual private network, vulnerability assessment, intrusion prevention, Internet content and email filtering, and remote management technologies and security services to enterprises and service providers around the world. Symantec’s Norton brand of consumer security products is the leader in worldwide retail sales and industry awards. Headquartered in Cupertino, Calif., Symantec has worldwide operations in 38 countries.

For more information, please visit www.symantec.com