Know Your Enemy: Honeynets

What a Honeynet is, its value, how it works, and risk/issues involved.

Honeynet Project
http://project.honeynet.org
Last Modified: 14 January, 2002

Over the past several years the Honeynet Project has been dedicated to learning the tools, tactics, and motives of the blackhat community and sharing the lessons learned. The primary tool used to gather this information is the Honeynet. The purpose of this paper is to discuss what a Honeynet is, its value, how it works, and the risks/issues involved. It is hoped that the security community can use the techniques discussed here to learn for themselves about the blackhat community. It is also hoped that the security community can take the concepts and techniques discussed here and improve them, thereby improving the effectiveness of Honeynets and our ability to learn more about the enemy. However, we want to be sure that organizations are also aware of the many risks and issues involved with a Honeynet.

What is a Honeynet

A Honeynet is a type of honeypot designed specifically for research. A honeypot is a resource who's value is being probed, attacked, or compromised. Traditionally their value has been for deception or detecting attacks. They are usually single systems that emulate other systems, emulate known services or vulnerabilities, or create jailed environments. Some excellent examples of honeypots include Specter, Mantrap, or The Deception Toolkit. To learn more about honeypots, their values or specific honeypot solutions, refer to Honeypots: Definitions and Values.

A Honeynet is different from traditional honeypots, it is what we would categorize as a research honeypot. This does not make it a better solution then traditional honeypots, merely it has a different purpose. Instead of its value being detecting or deceiving attackers, its value is gaining information on threats. The two biggest design differences from a classic honeypot are:

- It is not a single system but a network of multiple systems. This network sits behind an access control device where all inbound and outbound data is controlled and captured. This captured information is then analyzed to learn the tools, tactics, and motives of the blackhat community. Honeynets can utilize multiple systems at the same time, such as Solaris, Linux, Windows NT, Cisco router, Alteon switch, etc. This creates a network environment that more realistically mirrors a production network. Also, by having different systems with different applications, such as a Linux DNS server, a Windows IIS webserver, and a Solaris Database server, we can learn about different tools and tactics. Perhaps certain blackhats target specific systems, applications, or vulnerabilities. By having a variety of operating systems and applications, we are able to accurately profile specific blackhat trends and signatures.

- All systems placed within the Honeynet are standard production systems. These are real systems and applications, the same you find on the Internet. Nothing is emulated nor is anything done to make the systems less secure. The risks and vulnerabilities discovered within a Honeynet are the same that exist in many organizations today. One can simply take a system from a production environment and place it within the Honeynet.

It is these two design differences that make a Honeynet primarily a tool for research. It can be used as a traditional honeypot, such as detecting unauthorized activity, however a Honeynet requires a great deal more work, risk and administration. Its simply not worth all the effort of building and maintaining a Honeynet just to detect attacks. You are far better off with the simpler honeypot solutions mentioned above.

Value of a Honeynet

Traditionally, information security has been purely defensive. Firewalls, Intrusion Detection Systems, encryption; all of these mechanisms are used defensively to protect one's resources. The strategy is to defend one's organization as best as possible, detect any failures in the defense, and then react to those failures. The problem with this approach is it purely defensive, the enemy is on the attack. Honeynets attempt to change that, they give organizations the ability to take the initiative. The primary purpose of a Honeynet is to gather information about threats that exist. New tools can be discovered, worms can be captured and analyzed, attack patterns can be determined, and attacker motives studied. Captured information can also be used as an early indications and warning system, alerting to attacks before they happen. The Honeynet Project demonstrated this in the paper Know Your Enemy: Statistics. The ultimate goal of Honeynets is to provide information that can be used to protect against threats. Honeynets can be compared to the Navy's use of SOSUS during the Cold War. During the 1950-1980's, enemy submarines posed a threat as they could silently approach and attack from anywhere in the world's oceans. To detect these threats, devices were placed throughout the ocean's floor to passively capture the activity of enemy submarines. Honeynets can be considered the SOSUS of cyber space, passively gathering information on threats. The only difference is, for a Honeynet to passively gather information, blackhats have to probe, attack, or exploit Honeynet systems.

Honeynets can also provide an organization information on their own security risks and vulnerabilities. Honeynets can consist of the same systems and applications that an organization is using for its production environment. Risks and vulnerabilities that exist in a Honeynet (which is far more closely monitored and analyzed) identify risks and vulnerabilities in an organization's production environment. For example, a company may want to implement a new webserver interface for credit card use. Both the system and application can be first tested in a Honeynet environment to identify any unknown risks or vulnerabilities. We ourselves have learned a great deal testing IDS, Firewall, and logging systems within the Honeynet environment. However, such activity is extremely time intensive, most organizations are better off focusing on securing their existing resources.

Additionally, a Honeynet can help an organization develop it's Incident Response capabilities. In the past two years we have vastly improved our abilities to detect, react to, recover, and analyze systems that have been compromised. We have released two works based on these experiences, specifically Know Your Enemy: Forensic Analysis and The Forensic Challenge. The advantage one has in analyzing these compromised systems is you already have most of the answers. You can then treat a compromised system as a 'challenge', where you test your abilities to determine what happened using various forensic techniques. You can then compare these results to the data captured from within the Honeynet. This information can also be used to determine if any other systems within your production network have been compromised. Once you have identified the signatures of the blackhat and his attacks, you can then review your production environment for the same signatures, identifying compromised systems you did not know about.

These examples of risk identification and incident response demonstrate two such possibilities of Honeynet functionality. Keep in mind, Honeynets are merely a tool, they provide value in however you choose to use them. However, their primary purpose and design is based on researching threats.

How it Works

Conceptually, Honeynets are a simple mechanism. We create a network similar to a fishbowl, where we can see everything that happens inside it. Similar to fish in a fishbowl, we can watch and monitor attackers in our network. Also just like a fishbowl, we can put almost anything in there we want. This controlled network, becomes our Honeynet. The captured activity teaches us the tools, tactics, and motives of the blackhat community.

Traditionally, the greatest problem security professionals face in detecting and capturing blackhat activity is information overload. The challenge for most organizations is determining from vast amounts of information what is production traffic and what is malicious activity. Tools and techniques such as Intrusion Detection Systems, host based forensics, or system log analysis attempt to solve this by using a database of known signatures or algorithms to determine what is production traffic and what is malicious activity. However, information overload, data pollution, unknown activity, false positives and false negatives can make analyzing and determining activity extremely difficult.

Like all honeypots, the Honeynet solves this problem of data overload through simplicity. A Honeynet is a network designed to be compromised, not to be used for production traffic. Any traffic entering or leaving the network is suspicious by definition. Any connection initiated from outside the Honeynet into the network is most likely some type of probe, attack, or other malicious activity. Any connection initiated from the Honeynet to an outside network indicates that a system was compromised. An attacker has initiated a connection from his newly hacked computer and is now going out to the Internet. This concept of no production traffic greatly simplifies the data capture and analysis.

There are two critical requirements that define every Honeynet, they are Data Control and Data Capture. If there is a failure in either requirement, then there is a failure within the Honeynet. Honeynets are extremely flexible tools, they can built and deployed a variety of different ways, as such almost no two Honeynets look the same. But they must all meet the requirements of Data Control and Data Capture. Data Control is what mitigates risk. It controls the attacker's activity by limiting what can happen inbound and outbound. The risk is that once an attacker compromises a system within the Honeynet, they can use that system to attack other non-Honeynet systems, such as organizations on the Internet. The attacker has to be controlled so they cannot do that. They can attack other systems within the Honeynet, but we have to protect non-Honeynet systems. Data Capture is what collecting all the activity that happens inbound, outbound, or within the Honeynet. This is how we learn, by capturing the attackers's activities. The trick to these requirements is meeting them without the attacker knowing. Our goal is to both control and capture all of the attacker's activity, without them realizing they are within a Honeynet.

There is a third requirement, Data Collection, but this is only for organizations that have multiple Honeynets in distributed environments. Many organizations will have only one single Honeynet, so all they need to do is both Control and Capture data. However, organizations that have multiple Honeynets logically or physically distributed around the world have to collect all of the captured data and store it in a central location. This way the captured data can be combined, exponentially increasing its value. The Data Collection requirement provides the secure means of centrally collecting all of the captured information from distributed Honeynets.

The Honeynet Project has created a document that defines these three requirement in extensive detail. The purpose of the document is to give organizations the flexibility to build a Honeynet tailored to their environment and goals. However, the document ensures that the Honeynets are effectively and securely deployed, allowing different Honeynets to inter operate. Any organization considering deploying their own Honeynets are HIGHLY encouraged to follow these requirements.

Honeynet Definitions, Requirements, and Standards

Data Control

As stated above, data control is the containment of activity. When we are dealing with blackhats there is always risk, we must mitigate that risk. We want to ensure that once compromised, a honeypot cannot be used to harm any system outside the Honeynet (anything inside the Honeynet is fair game). However, the challenge is to control the data flow without the blackhat's getting suspicious. Once a system is compromised, blackhats will often require Internet connectivity, such as retrieving toolkits, setting up IRC connections, etc. We have to give them the flexibility to execute these actions, as these are the very steps we want to learn and analyze. Also, blackhats may become highly suspicious if they cannot initiate any outbound connections. We made that very same mistake with our first honeypot. We did not allow any outbound Internet connections. It took the blackhat only fifteen minutes to figure out something was wrong, wipe the system drive, and leave the network. So, the trick is to give the blackhat flexibility to execute whatever they need, but without allowing them to use the compromised system to attacks others, such as Denial of Service attacks, system scans, and exploits.

We have designed a Honeynet that tracks what connections go into and out of a Honeynet. This is done by placing a firewall in front of the Honeynet, all traffic must go through the firewall. The firewall keeps track of how many connections are initiated from a honeypot out to the Internet. Once a honeypot has reached a certain limit of outbound connections, the firewall will then block any more attempts. This gives the blackhat the flexibility to execute whatever they need, while giving us automated protection against abuse. There is no right or wrong number of connections to allow. It all depends on what functionality you are looking for. If you want to capture automated attacks, such as auto-rooters or Worms, then you most likely do not need any outbound connections. The automated

attacks simply compromise one of the honeypots and then the firewall blocks all outbound connections, stopping the automated attacks from replicating. This way you can grab the tools used for the attacks without exposing other systems to risk. However, if you want to discover manual blackhat activity, or what happens after a system is compromised, then you will most likely have to allow some outbound type of activity. Once again, the more outbound activity you allow, the more you can learn, however the greater the risk.

We have found five to ten outbound connections a good number to keep blackhat's happy, while protecting others from attacks. This protects the Honeynet from being used as a platform to scan, probe, or attack most other systems. Some organizations may not require this functionality. If you can afford to have someone monitor the Honeynet 24 hours a day, then you can afford to allow unlimited outbound connections. If a Denial of Service attack is identified, the person monitoring the Honeynet can simply disable the attack. However, we have developed a automated means to do this, as we cannot afford 24 hour a day surveillance.

Additionally, a router is placed between the firewall and the Honeynet. This is done for two reasons. One, the router hides the firewall. When a honeypot is compromised, blackhats will find a production router between them and outside networks. This creates a more realistic environment and obscures the firewall from being discovered. The second purpose of the router is to act as second access control device. The router can supplement the firewall, ensuring compromised honeypots are not used to attack systems outside the Honeynet.

In Diagram A we have a detailed network map of a Honeynet. This is the very Honeynet used for research in the Know Your Enemy series of papers. In the diagram, you see the firewall separating the Honeynet into three networks. Specifically, the Honeynet, Internet, and Administrative Network. We will discuss the Administrative Network when we cover Data Capture. Any packet entering or leaving the Honeynet has to go through both the Firewall and the router. The Firewall is our primary tool for controlling inbound and outbound connections. The router is used to supplement this filtering. Our firewall is designed to allow any inbound or outbound connection. However, it limits honeypots within the Honeynet to initiate 5 connections to the Internet. After the 5th connection, any more connections are blocked by the firewall.

We originally built this functionality using a CheckPoint FireWall-1 firewall and a shell script that counts how many connections have been initiated outbound. When the limit is met (in our case 5 connections) the script configures the firewall to block any more connections from the compromised honeypot. We have our firewall configured to send an alert when this happens, notifying that a compromised honeypot has been blocked. The specifics of how we configure this functionality, including source for the script, can be found at Intrusion Detection for FireWall-1. You are not limited to FireWall-1, nor are you limited to use the script we developed. We only present this as one possible method to Data Control. For example, the same functionality can be duplicated using IPFilter and Swatch. IPFilter is a Open Source firewall found on most BSD types of Unix. Swatch is an automated log watcher that can monitor and react to your firewall logs in near real time. Another option is IPTables. IPTables is the a firewalling software that comes with most distributions of Linux running kernel version 2.4.x. Several Honeynet members have successfully used IPTables and it's limit feature as a Data Control mechanism. This capability monitors the number of connections a system makes and can automatically block connections once a certain limit has been met. We have even created a automated script for IPTables that implements the Data Control mechanisms. What is critical is not *how* you implement Data Control, but as long as you meet the *requirements* of Data Control.

The router acts as a second layer of access controls. We primarily use this to protect against spoofed or ICMP based attacks. The router allows only packets with the source IP address of the Honeynet to leave the router. In our example, this means packets only with source IP of the 172.16.1.0/24 network can leave the Honeynet. This protects against most spoofed based attacks, such as SYN flooding or SMURF attacks. We also block ICMP outbound traffic. This was done due to our limited abilities in automated, stateful ICMP tracking, we hope to change this in the future. Other organizations do not have to limit ICMP in such a manner. Limiting ICMP protects against attacks such as SMURF, network mapping, and Ping of Death (yes, we still see that in the wild). Below is the ACL we use on our router, notice the large amounts of ICMP the router has dropped:

```
router#show ip access-list 100
Extended IP access list 100
  deny icmp any any (5446314 matches)
  permit ip 172.16.1.0 0.0.0.255 any (66372 matches)
  deny ip any any log (59990 matches)
```

We have found the combination of both the firewall and the router to make an extremely effective technique of controlling outbound traffic, in that it gives blackhats the flexibility to execute most of what they need to accomplish, while limiting attacks that can be launched against other systems. To the best of our knowledge, these security measures have not been defeated. A variety of Denial of Service attacks have been attempted from the Honeynet, including SYN Flooding, SMURF, and Ping of Death, all have been detected and blocked. Also, a variety of scanners or 'auto rooters' have been launched from the Honeynet, once again these have been detected and blocked. However, it is most likely only a matter of time before someone defeats these measures. Keep in mind, there is always risk when dealing with the blackhat community.

Data Capture

Data capture is the capturing of all of the blackhat's activities. It is these activities that are then analyzed to learn the tools, tactics, and motives of the blackhat community. The challenge is to capture as much data as possible, without the blackhat knowing their every action is captured. This is done with as few modifications as possible, if any, to the honeypots. Also, data captured cannot be stored on locally on the honeypot. Information stored locally can potentially be detected by the blackhat, alerting them the system is a Honeynet.

The stored data can also be lost or destroyed. Not only do we have to capture the blackhats every move without them knowing, but we have to store the information remotely. The key to this is capturing data in layers. You cannot depend on a single layer for information. You gather data from a variety of resources. Combined, these layers then allow you to paint the big picture. We will now discuss these layers and there uses.

The first layer of logging activity are is the firewall. Previously, we discussed how we can use the firewall to control data. This same firewall can be used to capture data. The firewall logs all connections initiated to and from the Honeynet. This information is critical, as all connections are suspicious. We have designed our firewall not only to log all connections, but to alert us whenever an connection is attempted. For example, if someone were to attempt a telnet connection to a system in the Honeynet, the firewall would log and alert us to the event. This is extremely useful for tracking scanning patters. Another use is of backdoors or proprietary ports. Most exploits create a shell or backdoor on a system. These backdoors are easy to detect when the firewall alerts you to a connection on a system on some random high port. The firewall also alerts us when a honeypot on the Honeynet initiates an outbound connection. The firewall once again logs and alerts us to this activity. However, this alert is a higher priority, as it indicates a system was compromised. Such an alert would be sent by both email and pager.

Another critical layer is the IDS system, it has two purposes. The first, and by far most important, is to capture all network activity. Its primary job is to capture and record every packet that hits the wire. If you refer to DiagramA, you will see that our IDS system is on a switch physically shared by all other systems on the Honeynet. The IDS system resides on a 'port monitoring' port, so it can record all network activity. These records are then used to analyze the blackhat's activities. The second function of the IDS system is to alert us to any suspicious activity. Most IDS systems have a database of signatures, when a packet on the network matches a signature, an alert is generated. This function is not as critical for a Honeynet, as any activity is considered suspicious by nature. However, IDS systems can give detailed information about a specific connection.

The Honeynet Project has had phenomenal success with the Open Source IDS snort. We have configured snort to capture all network activity to a binary log file. These binary logs are critical, as they capture every packet that enters and leaves the Honeynet. In addition, snort logs all ASCII communication (such as keystrokes from an FTP session) to session breakout files. Both binary and ASCII logs are logged to their own directory for each day. This makes it much easier to review and analyze logs on a daily basis. To accomplish this daily logging, we have configured a startup script which is executed by cron every day, restarts snort and logs to a new directory. Third, all snort alerts are forwarded to the syslog server. These alerts are then logged to /var/log/messages, which is monitored by Swatch.. Swatch monitors this log file in real time and does two things. First, it identifies any snort alerts and forwards them via email to a system administrator for real time warning. Second, Swatch archives all snort alerts to a simple text file for archiving. We use the following swatchrc configuration file for this functionality. Last, we use a backend database to collect alerts from distributed Honeynets.

A third layer are the systems themselves, we want to capture all system and user activity that occurs on a system. The first method for this is to have all system logs not only log locally, but to a remote log server. For Unix systems and most network devices, this is simply done by adding an entry for a remote syslog server in the configuration file. For Window based systems there are third party applications that will remotely log information. Also, system logs can be written to a NFS or SMB share on the remote log server. NT requires you use third party software to have the capability to write system information to syslog, but it can write it to a network file system. This way, critical system information such as process activity, system connections, and attempted exploits are safely copied to a remote system. We do not want to make any attempt to hide the use of a remote syslog server. If the blackhat detects this, the worse they can do is disable syslogd (which is standard behavior for most blackhats). This means we will no longer have continued logs, however we will at least have information on how they gained access and from where.

More advanced blackhats will attempt to compromise the remote syslog server in an attempt to cover their tracks. This is exactly what we want to occur. The syslog server is normally a far more secured system. This means for a blackhat to successfully take control of such a system they will have to use more advanced techniques, which we will capture and learn from. If the syslog server is compromised, we have lost nothing. Yes, the blackhat can gain control of the system and wipe the logs. However, do not forget, our IDS system that is on the network passively captured and recorded all of the logging activity that happened on the network. In reality, the IDS system acts as a second remote log system, as it passively captured all the network data.

A second method to capturing system data is to modify the system to capture keystrokes and screen shots and remotely forward that data. The Honeynet Project is currently testing several tools that have this functionality. The first is a modified version of bash. This shell, developed by Antonomasia, can be used to replace the system binary /bin/bash. The trojaned shell forwards the user's keystrokes to syslogd, which is then forwarded to a remote syslog server. A second option is a modified version of TTY Watcher. This kernel module captures both user keystrokes and screen captures and forward this information over a non-standard TCP connection. There are a variety of other options to this functionality.

Data Collection

Data Control and Data Capture are two requirements for Honeynet technologies. Any time an organization deploys a Honeynet, they have to ensure these standards are met. Data Collection is different, it is optional. Data Collection is the aggregation of data from multiple Honeynets to a centralized point. Its purpose is to exponentially increase the value of information collected. Most organizations deploy only a single Honeynet, so Data Collection does not apply. However, some organizations, such as the Honeynet Research Alliance, deploy multiple Honeynets. For organizations such as these, there needs to be a standard to Data Collection.

When part of a distributed environment, each Honeynet is assigned a unique identifier. All data sent by each Honeynet to a central location is tagged with the unique identifier. There are currently six types of data that are collected by Honeynets. Four types of data are captured by Snort, the other two types are captured by the Honeynet firewall. This data is then forwarded by each Honeynet to the single data collection point. The six data types are shown below. For specifics on the definitions and standards of the data format, refer to the Honeynet Definitions, Requirements, and Standards documentation. Each data type example below is based on the data collected from a Honeynet on 02 December, 2001.

```
Snort Alerts,     Full     -> MySQL database - real time
Snort Alerts,     Full     -> ASCII text     - daily
Snort Alerts,     Fast     -> ASCII text     - daily
Snort binary log, Full     -> snort.log      - daily
```

```
Firewall logs,    Full     -> ASCII text     - daily
Firewall logs,    Unique   -> ASCII text     - daily
```

GenII Honeynets

The mechanisms we have just described for Data Control and Data Capture are effective, however we feel that they can be improved upon. The Honeynet Project is in the process of designing and deploying a far more effective Honeynet, one that is simpler to deploy, but harder to detect. The Honeynet we described above was successfully used during PhaseI of the Project, from 1999-2001. We consider this design GenI, or First Generation technology. The Honeynet Project has entered Phase II of the Honeynet Project, our goal is to capture the activity of more advance blackhats. As such, we need a more advance Honeynets, the GenII, or Second Generation Honeynet. The goal of GenII Honeynets are to create a solution that is easier to deploy, more difficult to detect, and yet is highly flexible, able to adapt to a variety of environments. To meet these goals, and the requirements set forth in the Definitions, Requirements, and Standards doc, the Honeynet Project is researching and developing the following.

GenII Honeynets will have all requirement combined onto a single device. This means all Data Control, Capture, and Collection will happen from a single resource. This will make much easier to both deploy and manage a Honeynet. This single device will be a layer2 gateway. This gives us several advantages. The fact the device is layer2 will make it much more difficult to detect, as it has no IP Stack. It is acting as a bridge, there is no routing of traffic nor any TTL decrement. The second advantage is, as a gateway, all inbound and outbound traffic must go through the device. This means we can both control and capture all inbound and outbound traffic from the single device. Refer to Diagram B for an example of what such a deployment could look like. The Project intends to implement more advance, automated data control mechanisms. We will give attackers greater ability to interact with compromised systems, yet have greater control over their activities, with this control being more difficult to detect. We hope by giving the attackers more flexibility in their actions, we can gather greater information on them. This is done by creating a more intelligent and flexible response to the blackhat's actions.

The first advancement in data control will be in our ability to detect unauthorized activity. Instead of tracking the attacker's activity by counting the number of outbound connections, we will add more intelligence by tracking what their activity is. We will identify unauthorized activity by their actual actions and intent. If they attempt ten outbound FTP connections, that will be fine. However, if they attempt a single outbound FTP exploit against a non-Honeynet system, then that activity must be controlled. We intend on creating greater intelligence by analyzing the attackers activity, not by merely counting the number of connections. Rob McMillan has built this capability into IPTables and Snort, creating an IDS Gateway.

The second advancement will be in the way we respond to unauthorized activity. Instead of simply blocking connections, we intend to modify or throttle the attacker's activity. It is hoped these responses will be far more difficult for the attacker to detect. We do this by modifying packets as they travel through the layer2 gateway. For example, once an attacker has taken over a system within the Honeynet, they may attempt to launch an FTP exploit against a non-Honeynet system. With GenI technology, the data control is limited, after the fifth attempt outbound, all further activity, including any exploits, would be blocked. However, with GenII technology, the exploit attempt would be identified and then modified to make the attack ineffective. The layer2 gateway would modify several bytes within the exploit code, disabling its functionality, then allow the crippled attack to proceed. The attacker would see the attack launched and packets return, but would not understand why their exploit never worked. We gain better control of the attacker's actions, without their knowledge. Another example would be throttling unauthorized activity. An attacker may attempt to scan or launch Denial of Service attacks against non-Honeynet systems, these attacks would pass through the layer2 gateway, allowing us to drop certain packets or entire connections. We also have the ability to fake responses, such as blocking entire connections, but return RST packets to the attacker, forging a dropped connection. Once again, GenII technologies give us more flexible responses that are harder to detect. One of the Project's primary tools for implementing GenII Data Control is Hogwash, developed by team member Jed Hail, with extensive help from Jason Larsen. Hogwash is a layer 2 IDS gateway, based on Snort, that will have true GenII functionality.

We also are researching more advance data capture methods. Traditionally a great deal of information could be captured passively from the network, such as sniffers. This is no longer possible with the dramatic increase in use of encryption. Almost all attacks today use some type of encryption to gain or maintain control of victims. As such, most of the attacker's activity has to be captured from the honeypots themselves. GenII technologies include a variety of system and kernel modules designed to capture all of the attacker's activities, without the attacker's knowledge. Team members K2 and Dragos Ruiu are leading the development of these technologies.

Virtual Honeynets

Another area the Honeynet Project is researching is the use of virtual Honeynets. Virtual Honeynets combine all the elements of a Honeynet onto one physical system. Not only are all three requirements of Data Control, Data Capture, and Data Collection met, but the actual honeypots themselves run on the single system. Virtual Honeynets can support either GenI or GenII technologies. The honeypots are actual operating systems. Nothing is emulated. The advantage here is one of cost and efficiency. It is much cheaper to use a single system to run all the elements of a Honeynet, and it is much easier to deploy and maintain. One such example is the use of VMware. VMware allows varios operating systems to run at the same time on the same system. Team member Michael Clark is leading this research and has published the paper Virtual Honeynets. Kurt Siefried has written a whitepaper on the Data Capture issues with VMware virtual Honeynets.

The Honeynet Project is also actively working with other organizations in the research of User Mode Linux, developed by Jeff Dike. UML has the capability of running multiple instances of Linux on the same systems (over twenty guest OS's have been achieved on a single box). UML is Open Source and free, however it is currently limited to Linux systems. It is hoped to port UML to other operating systems in the future.

Care, Feeding and Risk

Honeynets are not a "fire and forget" solution. They are a complex type of honeypot that requires constant maintenance, administration and vigilance. For maximum effectiveness, you need to detect and react to incidents as soon as possible. By watching the blackhat activities in real time, you can maximize your data capture and analysis capabilities. Also, to detect the unknown, you are required to constantly review suspicious activity. This requires extensive time and analysis capabilities. For example, in just 30 minutes a blackhat do enough damage to a compromised honeypot to require 30-40 hours to fully understand what happened. Constant maintenance is also required to ensure operability of your Honeynet. If something goes wrong (and something always does) this can cause a failure within the Honeynet Your alert processes may die, disks can fill, IDS signatures become out of date, configuration files become corrupted, system logs need to be reviewed, firewalls need to be updated and patched. These represent just some of the constant care and feeding that is required for a successful Honeynet. Your work has only begun when you implement a Honeynet.

Also, there are risks involved with building and implementing a Honeynet. We have blackhats attacking and compromising our systems. By setting up a network to be compromised, we expose ourselves, and others, to risk. You assume a responsibility to ensure that the Honeynet, once compromised, cannot be used to attack or harm other systems. However, with an environment like this, there is always the potential for something to go wrong. We have implemented a variety of measures to mitigate this risk. However, it is quite possible for a blackhat to develop a method or tool that allows them to bypass our access control methods. Also, one needs to be constantly testing and updating the environment to ensure control measures are working effectively. Never underestimate the creative power of the blackhat community. The use of a firewall, routers, and other techniques helps mitigate the risk of the Honeynet being used to damage other systems. However, there is still risk.

Last, Honeynets will not solve your security problems. We highly recommend that organizations focus on best practices first, such as strong authentication, use of encrypted protocols, reviewing system logs, and secure system builds. By prioritizing on proper policies and procedures, organizations can greatly reduce risk. Honeynets do not reduce risk, they most likely increase it. If your organization is interested in the detection or deception capabilities of honeypots, then we recommend you review the honeypot whitepaper and products discussed at the beginning of this article. Honeynets are a honeypot designed primarily for research, to gather information on the enemy. They will not fix your unsecured server, nor fix bad process or procedures.

Legal Issues

These methods of research have raised some issues, specifically entrapment, privacy, and upstream liability. The Honeynet Project has attempted to address those issues. However, we are not lawyers, nor do we pretend to be. The views expressed below are only our opinion and in no way represent legal advice or counsel.

Lets start first with the issue of entrapment. The legal definition of entrapment is

A person is 'entrapped' when he is induced or persuaded by law enforcement officers or their agents to commit a crime that he had no previous intent to commit.

The Honeynet Project feels that entrapment is not an issue. First, the members of the Project are not law enforcement. We are not acting under the control of law enforcement, and we don't even have prosecution as an intent. Therefore, the legal definition of entrapment does not apply. This is true for most organizations that are not law enforcement and do not intend on prosecuting. Even for law enforcement, Honeynets most likely do not represent entrapment, as they are not used to induce nor persuade attackers. Honeynets are designed to mirror production systems, they have the same configurations and functionality found in most systems today. Nothing is done to induce or persuade attackers to target Honeynets. Instead, attackers target and attack Honeynets are there own initiative. As such, entrapment is most likely not an issue with Honeynet technologies.

The next potential issue is privacy, either in the files placed on compromised systems by intruders and the interception of communication (usually IRC) relayed through Honeynets. While there is case law about the loss of the right of privacy in storing files on a stolen computer, or one that an intruder has compromised and is using without the owner's authorization, there is less case law surrounding interception of communication that is relayed through a compromised host. Privacy laws exist in the form of state statutes and federal statutes. State statutes may supersede, or may be superseded by, the federal ones.

At the federal level, the two main statutes concerning communications privacy are the Electronic Communication Privacy Act (18 USC 2701-11), and federal Wiretap Statute (Title III, 18 USC 2510-22). And don't forget that other countries may have similar privacy laws that must be considered if you are implementing Honeynets outside the U.S.

The Honeynet Project is attempting to determine what issues exists and how they apply to most organizations today. Until we can establish the legal issues involved, organizations are recommended to ***review all legal issues with their own legal counsel before proceeding***.

Last, there is the issue of upstream liability. You or your organization can potentially be held accountable if a Honeynet system is used to attack any non-Honeynet systems. This is why the Project is extremely focused on Data Control functionality. You must ensure that your Honeynet does not expose other organizations to risk, or you may be held liable for any damage occurred.

Conclusion

Honeynets are a type of honeypot designed to gather information, specifically the tools, tactics and motives of the blackhat community. This information is then used to protect organizations against various threats. There are two design differences between traditional honeypots and a Honeynet. The first difference is a Honeynet is not a single system, but a network of multiple systems and applications. The second difference is Honeynets are production systems, the same systems found on the Internet today, neither the systems nor vulnerabilities are emulated. This combination makes Honeynets an excellent tool to learn, specifically a honeypot designed for research. However, Honeynets require a tremendous amount of administrative overhead. The Honeynet administrator has the responsibility of ensuring that no other systems will be attacked from the compromised Honeynet. Without proper administration, risks of use may outweigh the reward. This tool is not the security panacea, and may not be suitable the solution for every organization. The Honeynet Project highly recommends that organizations first focus on securing their organization, such as patching systems or disabling services. Once secured, organizations may then be able to use Honeynets as a powerful tool to take the initiative and learn more about the enemy and themselves. However, organizations are highly recommended to review with their legal counsel what legal issues may exist, before deploying Honeynets. Individuals or organizations interested in learning more about Honeynet technologies are recommended to review the book *Know Your Enemy*, written by the Honeynet Project.