

Know Your Enemy: Honeynets

Honeynet Project 2001-04-26

Know Your Enemy: Honeynets

by "*Honeynet Project*"

last updated April 26, 2001

Over the past several years the [Honeynet Project](#) has been dedicated to learning and 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 to the security community, 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 methods 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 tool for learning. It is a network of production systems designed to be compromised. Once compromised, this information is captured and analyzed to learn about the blackhat community. This idea is similar to honeypots, but there are several differences. A honeypot is a system designed to be attacked, usually for the purpose of deception or alerting of blackhat activity. Generally, honeypots are systems that emulate known vulnerabilities, emulate other systems, or are modified production systems that create caged environments. Examples of such honeypots are [The Deception Toolkit](#), [CyberCop Sting](#), and [Mantrap](#). Deception Toolkit is a collection of scripts that emulate known vulnerabilities. CyberCop Sting is a NT box that emulates the IP stack and inetd of various systems. Mantrap modifies a Solaris system to create several caged environments. These are all excellent solutions, however they are limited, focusing primarily on alerting and deception. (Note, of the three, we feel that Mantrap has the most potential to also be used as a research tool, however it is has certain limitations).

A Honeynet is different. Its two biggest design differences from a classic honeypot are that:

- It is not a single system but a network of multiple systems. This network sits behind a firewall where all inbound and outbound data is contained, captured and controlled. This captured information is then analyzed to learn the tools, tactics, and motives of the blackhat community. The Honeynet 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 services, such as a Linux DNS server, a Windows NT webserver, and a Solaris FTP 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 the fact that production systems are used that make a Honeynet unique. Traditionally the Honeynet Project has used the most commonly found systems on the Internet, default installations of Linux, Solaris, Windows98, and Windows NT. Because we use the most common operating systems, the risks and vulnerabilities discovered in our research apply to the largest percentage of the Internet community.

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 intelligence about the enemy, to learn the tools, tactics, and motives of the blackhat community. By gathering this intelligence, organizations can better understand their threat, and how to protect against this threat. Information security has often been compared to the military, such as the defense of a castle or guerilla warfare. Regardless of the analogy you choose, organizations can take the initiative by learning about the enemy, before they strike.

For example, one of the primary sources of information a Honeynet can gather is communication amongst blackhats, such as IRC (Internet Relay Chat). Blackhats often talk freely amongst each other, revealing their motives, goals, and actions. We have captured these conversations through the use of Honeynets, monitoring their every word between each other. We have even captured real time video shots of blackhats involved in the attacks on our systems. This gives us insight on how blackhats target and attack systems. A specific example is the paper [Know Your Enemy: Motives](#). In this paper we tracked a group of blackhats that had targeted a specific country. Over a three week period, not only did we learn how they target and attack systems, but more importantly why. Based on this detailed information, we can now better understand and protect against this common threat.

Honeynets also provide an organization intelligence 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.

Last, 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 methods have raised some issues.

The issue of entrapment:

Entrapment is a legal term applied to law enforcement agents who entice a criminal into engaging in an illegal behavior in which they would not otherwise engage. We 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, we don't consider setting up a Honeynet to be entrapment. Others would argue that we are providing an "attractive target", which means we are sticking unsecured systems on the Internet to entice people to break into them and in turn use them as stepping stones to attack others. This is also false, as we do not advertise these systems in any way. If someone finds one of our systems, compromises it, and uses it for things they shouldn't be doing, that is because they are actively and knowingly engaging in this unauthorized activity.

The issue of privacy:

While there are some moral and ethical issues with this activity (and we wrestle with them internally all the time), the recently published [Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations](#) by the Computer Crime and Intellectual Property Section, Criminal Division, United States Department of Justice, provides several case references where appellate judges have ruled against defenses of privacy violations in criminal computer trespass and fraud cases. Some of these issues include the following:

- The people breaking into these systems are NOT AUTHORIZED to use them, and if they place any files on them (when they have no legitimate accounts or use privileges), they have given up their privacy rights to that data by placing it on our systems.
- By using our systems for communication, they have given up their right to privacy in that communication.
- We do not provide public accounts, therefore we are not a service provider and are not bound by privacy requirements designed for service providers.
- We are not law enforcement, nor do we act under the control of law enforcement or even prosecute intruders into our systems anyway, so we are not bound by the evidence collection restrictions otherwise placed on law enforcement and their agents.
- And even if we did witness a serious computer crime and chose to report it, we gather logs and record network traffic as a regular course of "business" and are free to give them to law enforcement if and when we choose.

Until there are more clear judgments at the highest levels of the US Judicial system, we believe

we are following the current laws and staying within the lines of propriety. (Those outside the US should consult your own legal agencies for guidance before implementing a Honeynet.)

How it Works

The primary goal of a Honeynet is to learn about the blackhat community. This is done by tracking their every move. We create a fishbowl environment where we can monitor all of the activity that happens within the Honeynet. This captured activity teaches us the tools, tactics, and motives of the blackhat community. Traditionally, the greatest problem security professionals face in identifying and tracking blackhat activity is information overload. The challenge 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 known signatures based on previous attacks 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.

The Honeynet solves this 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. This greatly simplifies the data capture and analysis.

To successfully build a Honeynet, there are two critical elements, data control and data capture. Data control is the containment of activity, this means you control what packets can go where. The primary purpose of data control is to ensure that once a honeypot within the Honeynet is compromised, it cannot be used to attack other systems outside the Honeynet. The second element is data capture. Data capture is the capturing of all of the blackhat's activities, from system keystrokes to transmitted packets. Once captured, this data is then analyzed to determine the tools, tactics and motives of the blackhat community.

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 attack 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. 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 build this functionality by using a CheckPoint FW-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 have developed. We only present these as possible methods to Data Control. For example, the same functionality can be duplicated using [IPFilter](#) and [Swatch](#). This same functionality can also be built into IPTables using [Dynamic IPTable Scripts](#).

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. We have found this to be effective 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 routers' 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 their uses.

The first layer of logging activity 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 a 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 patterns. 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 [Diagram A](#), 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 currently uses 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. Last, 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 [snortsnarf](#) to create a simple, interactive database to query and review the archived snort alerts. This is excellent for trend analysis or in-depth research.

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 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. Often NT may not have the capability to write system information to syslog, but 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 advance 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 a 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 advance 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. The Honeynet Project encourages the security community to code and develop these options.

Care, Feeding and Risk

Honeynets are not a fire and forget solution, they require constant maintenance and vigilance. For maximum effectiveness, you will want 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 in depth suspicious activity. This requires extensive time and analysis capabilities. For example, for every 30 minutes a blackhat spends on a compromised honeypot, you can expect to spend 30-40 hours analyzing the data. 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.

There is also the potential that data capture can be bypassed. Blackhats are continually developing methods to avoid detection, such as anti-IDS techniques or encryption. For example, Dug Song has developed a toolkit called [fragrouter](#) that is designed to bypass Intrusion Detection Systems. It fragments packets in unique patterns, making it difficult for IDS systems to detect attack signatures. Rain Forest Puppy has developed a scanning tool called [whisker](#), this tool bypasses data capture by segmenting or modifying signatures. Most data capture systems can detect these techniques. However, there may be new ones that we do not know about and will bypass whatever methods we use. These are just several examples of how a Honeynet's security measures can be bypassed. Keep in mind that no matter what security measures we put in place, there is risk. Specifically the risk of someone smarter than us will come along.

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 can assist organization only after these best practices have been met, and are continued to be followed.

Conclusion

Honeynet is a tool designed to gather intelligence, specifically the tools, tactics and motives of

the blackhat community. They share the advantages of honeypots, specifically tools of deception or alerting, however their primary function is for learning. There are two design differences between 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. 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.

[Privacy Statement](#)

Copyright 2006, SecurityFocus