

# Know Your Enemy: Honeynets

*What a honeynet is, its value, and risk/issues involved.*

[honeynet Project](#)

<http://www.honeynet.org>

Last Modified: 12 May, 2005

One of the primary tools used by the HoneyNet Project to capture information is the honeynet. The purpose of this paper is to discuss what a honeynet is, its value, an overview of how it works, and the risks/issues involved. This paper is not a technical blueprint on how a honeynet works. For technical deployment details, refer to the paper [Know Your Enemy: Gen2 Honeynets](#). However, it is highly recommended you read this paper first. It is critical you understand the concepts, risks and issues of a honeynet before deploying such a technology.

## HoneyNet Overview

Traditionally, information security has been primarily 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 has the initiative. Honeynets attempt to change that. The primary purpose of a honeynet is to gather information about threats. This information is used to better understand threats, how they are evolving and changing, and how to best counter those threats. New tools can be discovered, attack patterns can be determined, and attacker motives studied. Examples include the papers [KYE: Tracking Botnets](#), [KYE: Motives](#), and [KYE: Automated Credit Card Fraud](#).

A honeynet is a type of [honeypot](#). Specifically, it is a high-interaction honeypot designed to capture extensive information on threats. High-interaction means a honeynet provides real systems, applications, and services for attackers to interact with (as opposed to low-interaction honeypots such as [Honeyd](#) which provide emulated services and operating systems. It is through this extensive interaction we gain information on threats, both external and internal to an organization. What makes a honeynet different from most honeypots is that it is an entire network of systems. Instead of a single computer, a honeynet is a network of systems designed for attackers to interact with. These victim systems (honeypots within the honeynet) can be any type of system, service, or information you want to provide. If you want to create Oracle databases on Solaris servers, not a problem. If you want to create a e-commerce site using IIS webserver on Windows 2003, not a problem. You can run everything from VAX systems to Cisco routers. It is this flexibility that gives honeynets their true power.

Conceptually honeynets are very simple. They are simply a network that contains one or more

honeypots. Since honeypots are not production systems, the honeynet itself has no production activity, no authorized services. As a result, any interaction with a honeynet implies malicious or unauthorized activity. Any connections initiated inbound to your honeynet are most likely a probe, scan, or attack. Almost any outbound connections from your honeynet imply someone has compromised a system and has initiated outbound activity. This makes analyzing activity within your honeynet very simple. With traditional security technologies, such as firewall logs or IDS sensors, you have to sift through gigabytes of data, or thousands of alerts. A great deal of time and effort is spent looking through this information, attempting to identify attacks or unauthorized activity. In many ways, it's the classic needle in the haystack problem, as you attempt to find the critical incident amongst volumes of information. Since a honeynet is nothing more than a network of honeypots, all captured activity is assumed to be unauthorized or malicious. All you are doing is capturing needles. It's up to you to prioritize which of those needles has the greatest value to you, then analyze them in great detail.

Honeynets are not a product, you don't simply install software on a computer. Instead, they are an architecture. This architecture creates a highly controlled network, one that you can control and monitor all activity that happens within it. You then place your target systems within that architecture. In many ways a honeynet is like a fishbowl. You create an environment where you can watch everything happening inside it. However, instead of putting rocks, coral, and sea weed in your fish bowl, you put Linux DNS servers, HP printers, and Juniper routers in your honeynet architecture. Just as a fish interacts with the elements in your fishbowl, intruders interact with the systems within your honeynet. You can see a diagram of a [GenII HoneyNet](#).

One final challenge with honeynets is deployment. Once you have created the architecture, how do you deploy the honeynet to attract hostile activity? Traditionally, honeynets have simply been nothing more than default installations of commonly used operating systems, deployed on external networks. These systems have no perceived value, no one aware of their existence. As a result, these type of deployment attract only common script kiddies, or automated attacks such as worms, botnets or auto-rooters, attacks that depend on highly active and random scanning. The vast majority of the honeynets deployed by the honeynet Project have followed this pattern. To date, very few honeynets have captured advanced attacker activity. How you deploy your honeynet will determine the type of attacker (or activity) you capture. The more perceived value your honeynet has, the more likely you will capture advanced activity. This is extremely challenging to do, a problem that has yet to be easily solved.

## Architecture Requirements

As we stated earlier, honeynets are nothing more than an architecture. To successfully deploy a honeynet, you must correctly deploy the honeynet architecture. There is no single rule on how you must deploy this architecture. The details and technology are left up to you. However, the problem with honeynets is that they are complex. If not properly deployed, you most likely will fail to capture the attacker's activities, or even worse, expose yourself or others to great risk. To help you deploy a honeynet, there are two critical requirements for a success; Data Control and Data Capture. All honeynet deployments should satisfy these two requirements. Data Control defines how activity is contained within the honeynet, without an attacker knowing it. Its purpose is to minimize risk. Data Capture is capturing all of the attacker's activity, without the attacker knowing it. Of the two, Data Control is the more important. Data Control always takes priority over Data Capture.

Data Control is the containment of activity. It is what mitigates risk. By risk, we mean there is always the potential of an attacker using a honeynet to attack or harm non-honeynet systems. We want to make every effort possible to ensure that once an attacker is within our honeynet, they cannot accidentally or purposefully harm other non-honeynet systems. This is more challenging than it seems. First, we have to allow the attackers some degree of freedom to act. The more activity we allow the attackers to perform, the more we can potentially learn about them. However, the more freedom you allow an attacker, the more risk there is they will circumvent Data Control and harm other non-honeynet systems. The balance of how much freedom to give the attacker vs. how much you restrict their activity is a decision every organization has to make themselves. Each organization will have different requirements and risk thresholds. Second, we have to control the attacker's activity without them knowing their actions are being controlled. One of the best ways to approach Data Control is not to rely on a single mechanism with which to implement it. Instead, implement Data Control using layers, such as counting outbound connections, intrusion prevention gateways, or bandwidth restrictions. The combination of several different mechanisms help protect against a single point of failure, especially when dealing with new or unknown attacks. Also, Data Control should operate in a fail closed manner. This means if there is a failure in your mechanisms (a process dying, hard drive full, misconfigured rules) the honeynet architecture should block all outbound activity, as opposed to allowing it. One thing to consider with Data Control, it can only minimize risk. We can never entirely eliminate the potential of an attacker using a honeynet to harm non-honeynet systems. Different technologies and approaches to Data Control have different levels of risk, but none eliminate risk entirely. We discuss risk in greater detail later in the paper.

Data Capture is the monitoring and logging of all of the blackhat's activities within the honeynet. It is this captured data that is then analyzed to learn the tools, tactics, and motives of members of the blackhat community. The challenge is to capture as much data as possible, without the blackhat detecting the process. As with Data Control, one of the primary lessons learned for Data Capture has been the use of layers. It is critical to use multiple mechanisms for capturing activity. Not only does the combination of layers help piece together all of the attacker's actions, but it prevents having a single point of failure. The more layers of information that are captured, at both the network and host level, the more that can be learned. One of the challenges with Data Capture is that a large portion of attacker activity happens over encrypted channels (such as IPSec, SSH, SSL, etc). Data Capture mechanisms must take encryption into consideration. Also, just as with Data Control, we have to minimize the ability of attackers to detect our capture mechanisms. This is done several ways. First, make as few modifications to the honeypots as possible. The more modifications you make, the greater the chance of detection. Second it is best that captured data not be stored locally on the honeypots themselves. Not only could this data be detected by attackers, but it could also be modified or deleted. As such, captured data must be logged and stored on a separate, secured system. Just as with Data Control, there are no guarantees with Data Capture. Attackers may identify ways to detect Data Capture mechanisms, and develop methods to bypass or disable them. We discuss these issues in greater detail later in this paper.

There is a third requirement, Data Collection, but this only applies to 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, such as the [Honeynet Research Alliance](#) 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. Refer to [Figure 1](#) to see an example of how the Honeynet Research Alliance has achieved this. 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 greater 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 interoperate. Any organization considering deploying their own honeynets are encouraged to follow these requirements. We recommend you take a moment and review them.

## [honeynet Definitions, Requirements, and Standards](#)

## Risk

Honeynets can be a powerful tool. They allow you to collect extensive information on a variety of threats. To obtain this information, you have to allow attackers access -- potentially privileged access -- to your systems and applications. As a result, the price you pay for this capability is risk. Any technology developed by man (or woman) can also be defeated by man (or woman.) Risk means different things to different organizations. You will have to identify what risks are important to you. Also, organizations have different thresholds for risk. We cannot determine what is right and wrong for you. Your organization will have to make those decisions for itself. All we can do is help make you aware of the risks. Also, we will not address legal issues of honeypots, or specifically honeynets. That is beyond the scope of this paper, and is specific to your country and organization. If you are interested in legal issues of honeypots, a good place to start is the legal chapter (which is freely available to the public) from our book [Know Your Enemy: 2nd Edition](#). We also recommend you seek your organization's legal counsel for more information, especially in reference to privacy or liability issues. In reference to risk, there are four general areas we will cover; harm, detection, disabling, and violation.

- Harm is when a honeynet is used to attack or or harm other, non-honeynet systems. For example, an attacker may break into a honeynet, then launch an outbound attack never seen before, successfully harming or compromising its intended victim. Data Control is the primary means of mitigating this risk. Multiple layers of Data Control are put in place to make it more difficult for the attacker to cause damage. However, there is no guaranteed method to ensure that a honeynet can not used to attack or harm someone else. No matter what mechanisms are put in place, an attacker can eventually bypass them. Your organization will have to decide how much risk it is willing to assume. For low risk organizations, you may want to minimize the activity allowed outbound (to zero, perhaps.) For organizations with greater risk thresholds, you may decided to to allow greater outbound activity.
- Second, there is the risk of detection. Once the true identity of a honeynet has been identified, its value is dramatically reduced. Attackers can ignore or bypass the honeynet, eliminating its capability for capturing information. Perhaps even more dangerous is the threat that once identified, an attacker can introduce false or bogus information into a honeynet, misleading your data analysis. For example, with local access to the honeynet, an advanced attacker, or an attacker armed with proper tools, can potentially identify that a honeynet is in place and may

even identify the honeynet Data Control and/or Data Capture mechanisms themselves. If you are simply blocking after 10 outbound connection attempts, the attacker simply needs to try 20 outbound connections and watch the 11th one consistently fail. If you are modifying packets as they pass through the honeynet, the attacker simply needs to send packets with known a payload to systems they control and see if they are modified in transit. If you are tunnelling traffic to a "honey farm", the added latency may give away the fact that a honeynet is in place. Or the attacker may simply use methods to detect the presence of local Data Capture capabilities on the honeypot itself.

- Third, there is the risk of disabling honeynet functionality. This could be an attack against either Data Control or Data Capture routines. Attackers may want to not only detect a honeynet's identity, but disable its Data Control or Data Capture capabilities, potentially without the honeynet administrator knowing that functionality has been disabled. For example, an attacker may gain access to a honeypot within the honeynet, then disable Data Capture functionality on the honeypot. The attacker could then feed the honeypot with bogus activity, making administrators think Data Capture is still functioning and recording activity, when it's not. Having multiple layers of Data Control and Data Capture helps mitigate this risk, as there is no single point of failure.
- The fourth and last risk, violation, is the catch all of remaining risk. Attackers may attempt criminal activity from your compromised honeynet without actually attacking anyone outside your honeynet. One example is an attacker using a Honeypot to upload, then distribute contraband or illegal material, such as illegal copies of movies, music, stolen credit cards, or child pornography. Remember, this individual broke into your system on their own initiative. You are not dealing with the most law abiding of cyber citizens. If detected, this illegal activity would be attributed (at least initially) to you by way of it being on your system. You may then have to prove to that it was in fact *not* you who was responsible for this activity.

In all four cases, there are two steps you can take to help mitigate these risks, human monitoring and customization. By human monitoring, we mean you have a trained professional monitoring and analyzing your honeynet in real time. Anytime you suspect an attacker has successfully gained (or attempting to gain) access to one of your honeypots (such as detection of outbound connections, frequent established inbound connections, increased inbound traffic, transfer of files, unusual system activity, etc) a security professional should be monitoring and analyzing all captured data. This helps prevent the risk of an attacker detecting or disabling a honeynet, and attempting to harm other non-honeynet systems. By having a human analyzing honeynet activity, instead of just depending on automated techniques, you help protect yourself against new or unknown attacks or honeynet countermeasures. In a worse case scenario, you can always shut down the honeynet down if the attacker has exceeded your organizations threshold for risk. Second, customization is critical. This paper and all honeynet technologies are OpenSource and publicly available. This means that anyone has access to this information, including the blackhat community, which we can assume are actively reading this and developing countermeasures. For you to successfully deploy a honeynet, you need to customize or modify your honeynet, so it is different as possible from what we discussed. Do not use default settings, modify current or add additional techniques (such as bandwidth limiting for Data Control) and customize your honeynet environment. A simple default installation that has no purpose or system activity is in itself a give away of a honeypot. Ideally, you will use customized and random

configuration, layering, some kind of dynamism, or other creative means to make detection and counter measures of the honeynet as difficult as possible to accomplish. Honeynet technologies help mitigate risk. However, its critical that you understand that risk is not eliminated, and that you understand these issues

## Conclusion

Honeynets are a form of a high-interaction honeypot. Their primary advantage is their ability to gather extensive information on threats. A honeynet is an architecture, similar to a fishbowl. Within this architecture you can deploy any type of system or application you desire. The two critical requirements for this architecture are Data Control and Data Capture, with Data Control taking the priority. While very powerful, honeynets present unique risks. Mechanisms can be put in place to mitigate these risks, but there is no way to eliminate all risk, it is critical you understand this. If you are interested in deploying a honeynet, you can learn more about the technical details at [Know Your Enemy: Gen2 Honeynets](#).

The Honeynet Project