

The Value of Honeypots, Part Two: Honeypot Solutions and Legal Issues

Lance Spitzner 2001-10-25

The Value of Honeypots, Part Two: Honeypot Solutions and Legal Issues

by *Lance Spitzner* with extensive help from *Marty Roesch*

last updated October 23, 2001

This is the second article in a two-part series that will offer an overview of honeypots: what they are, how they can add value to an organization, and several honeypot solutions. The [first article](#) offered a brief overview of honeypots, as well as the discussion of some their inherent strengths and weaknesses. This installment will take a look at some examples of different types of honeypots. We will also briefly discuss some important legal issues associated with honeypots and their use.

Honeypot Solutions

Now that we have been discussing the different types of honeypots and their value, let's discuss some examples. The more I work with honeypots, the more I realize that no two honeypots are alike. Because of this, I have identified what I call 'level of involvement'. Simply put, the more involved a honeypot is, the more value it can have. At the same time, the more involved a honeypot is, the more risk it is likely to have. The more a honeypot can do and the more an attacker can do to a honeypot, the more value can be derived from it. By the same token, the more an attacker can do to the honeypot, the more potential damage that attacker can inflict on the host system.

For example, a low involvement honeypot would be one that is easy to install that simply emulates a few services. Attackers can merely scan and potentially connect to several ports. Here the information generated by the honeypot is limited (mainly who connected to which ports and when); however, there is little that the attacker can exploit. At the other extreme would be highly-involved honeypots. These would be actual systems. We can learn far more from these honeypots, as there is an actual operating system for the attacker to compromise and interact with. Unfortunately, there is also a far greater level of risk, as the attacker has an actual operating system to work with. Neither solution is necessarily a better honeypot, it all depends on what you are attempting to achieve. Remember, honeypots are not a solution. Instead, they are tools, it all depends on what you are attempting to achieve.

Based on this concept of 'level of involvement', let's compare some different honeypot solutions. As stated before, none of these are necessarily better than any others, it just depends on what you are attempting to achieve.

We will discuss six honeypots: BackOfficer Friendly, Specter, Deception Toolkit, homemade honeypots, Mantrap, and Honeynets. This paper is not meant to be a comprehensive review of these products. Instead, I will cover the different types of honeypots, how they work, and demonstrate the value they add and the risks involved. If you wish to learn more about the capabilities of these solutions, I highly recommend you try them out on your own in a controlled, lab environment.

BackOfficer Friendly

BOF (as it is commonly called) is a very simple but highly useful honeypot developed by Marcus Ranum and crew at [NFR](#). It is an excellent example of a low involvement honeypot. Released several years ago, you can no longer find this product online at their site. However, I am such a big fan of this goody that I have maintained a copy which you can [download here](#) for free, (Marcus has given me permission to distribute this).

The reason I am such a big fan of BOF is due to it's simplicity. It is a great way to introduce a beginner to the concepts and value of honeypots. BOF is a program that runs on most window-based operating systems. All it can do is emulate some basic services, such as http, ftp, telnet, mail, or Back Orifice. Whenever someone attempts to connect to one of these ports, BOF is listening and will then log the attempt. BOF also has the option of "faking replies", which gives the attacker something to connect to. This way you can log http attacks, telnet brute force logins, or a variety of other activities. I like to run BOF on my laptop, as it gives me a feel for what type of activity may be taking place. The value in BOF is in detection. It can monitor only a limited number of ports, but these ports often represent the most commonly scanned and targeted services.

Specter

Specter is a commercial product and what I would call another 'low involvement' production honeypot. It is similar to BOF in that it emulates services, but it can emulate a far greater range of services and functionality. In addition, not only can it emulate services, but emulate a variety of operating systems. Similar to BOF, it is easy to implement and is low risk. Specter

works by installing on a Windows system. The risk is reduced, as there is no real operating system for the attacker to interact with. For example, Specter can emulate a Web server or telnet server of the operating system of your choice. When an attacker connects, he or she is then prompted with a http header or log-in banner. The attacker can then attempt to gather Web pages or log in to the system. This activity is captured and recorded by Specter; however, there is little else the attacker can do. There is no real application for the attacker to interact with, instead just some limited, emulated functionality.

Specter's value lies in detection. It can quickly and easily determine who is looking for what. As a honeypot, it reduces both false positives and false negatives, simplifying the detection process. Specter also support a variety of alerting and logging mechanisms.

One of the unique features of Specter is that it also allows for intelligence gathering, or the automated ability to gather more information about the attacker. Some of this information gathering is relatively passive, such as Whois or DNS lookups. However, some of this intelligence gathering is active, such as port scanning the attacker. While this intelligence functionality may be of value, many times you do not want the attacker to know he is being watched. Be careful when implementing any active, automated responses to the attacker.

Homemade Honey pots

Another common honeypot is homemade. These honeypots tend to be low involvement, as their purpose is usually to capture specific activity, such as Worms or scanning activity. These can be used as production or research honeypots, depending on their purpose. Once again, there is not much for the attacker to interact with; however, the risk is reduced because the attacker can inflict less damage. One common example of a homemade honeypot is to create a service that listens on port 80 (http), capturing all traffic to and from the port. This is commonly done to capture Worm attacks. One such implementation would be using netcat, as follows:

```
netcat -l -p 80 > c:\honeypot\worm
```

In the above command, a Worm could connect to netcat listening on port 80. The attacking Worm would make a successful TCP connection and potentially transfer its payload. This payload would then be saved locally on the honeypot, which can be further analyzed by the administrator, who can assess the threat of the Worm. Organizations such as [SANS](#) and [SecurityFocus.com](#) have had success using homemade honeypots to capture and analyze

Worms and automated activity.

Homemade honeypots can be modified to do (and emulate) much more, requiring a higher level of involvement, and risk. For example, FreeBSD has a [jail](#) functionality, allowing an administrator to create a controlled environment within the operating system. The attacker can then interact with this controlled environment. The value here is the more the attacker can do, the more can be potentially learned. However, care must be taken, as the more functionality the attacker can interact with, the more can go wrong, with the honeypot potentially compromised. If you have implemented your own homemade honeypot, I would love to [hear from you](#) on what you developed.

Deception Toolkit

Created by Fred Cohen, [Deception Toolkit](#) (DTK) is one of the original honeypots. I would characterize DTK as a low-to-mid involvement honeypot. It can do more than Specter and give us more information, but takes more work to install and has additional risk. However, this is still not a high involvement honeypot, as there is no true OS for the attacker to interact with. DTK is a collection of PERL scripts designed for Unix systems that emulate a variety of known vulnerabilities. The big advantage of DTK is that the toolkit is free and the user has the source. The disadvantage is that these scripts can potentially be exploited to give an attacker access to the system. Each script emulates a known vulnerability. For example, certain scripts emulate vulnerable SMTP servers. If successfully attacked and exploited, they can be configured to send bogus password files to attackers.

As stated at its home Web site, the goal of DTK is twofold: deception (prevention) and alerting (detection). As discussed earlier, I feel deception contributes little value to prevention. When DTK was first released, deceiving attackers had great value, as attacking systems were manually intensive. However, a large percentage of the threats today, such as autorooters and Worms, are highly automated and will probe and attack anything with an IP stack. There is no human to deceive, because no human interaction is required for the tool activity. However, DTK does have value in detection. Just like Specter, DTK can be used to detect attacks against a system. The advantage of DTK in detection is that users can modify the scripts to emulate any vulnerability they want. The disadvantage is that more work is involved on the user's part to deploy the product.

Now we begin to move into more involved honeypots. These solutions give us far greater

information, but potentially have far greater risk. We will be discussing two such honeybots: Mantrap and Honeybots. We will begin with Mantrap.

Mantrap

Produced by [Recourse](#), Mantrap is a commercial honeybot. Instead of emulating services, Mantrap creates up to four sub-systems, often called "jails". These jails are logically separated operating systems that are separated from a master operating system. Security administrators can modify these jails just as they normally would any operating system, to include installing applications of their choice, such as an Oracle database or Apache Web server. This makes the honeybot far more flexible, as the attacker has a full operating system to interact with, and a variety of applications to attack. All of this activity is then captured and recorded. Not only can we detect port scans and telnet logins, we can also capture rootkits, application level attacks, IRC chat sessions, and a variety of other threats.

However, just as far more can be learned, more can go wrong. Once compromised, the fully functional operating system can be used by the attacker to attack others. Care must be taken to mitigate this risk. As such, I would categorize this as a mid-high level of involvement. Also, these honeybots can be used as either a production honeybot (used both in detection and reaction) or a research honeybot to learn more about threats. There are limitations to this solution. The biggest one is that users are limited to what the vendor supplies them. Currently, Mantrap only exists on Solaris operating system.

Honeybots

[Honeybots](#) represent the extreme of research honeybots. They are high involvement honeybots. Users can learn a great deal from them; however, they also have the highest level of risk. Their primary value lies in research: gaining intelligence on threats that exist in the Internet community.

A Honeybot is a network of production systems. Unlike many of the honeybots we have discussed so far, nothing is emulated. Little or no modifications are made to the honeybots. This gives the attackers a full range of systems, applications, and functionality to attack. From this we can learn a great deal: not only their tools and tactics, but their methods of communication, group organization, and motives.

However, with this capability comes a great deal of risk. A variety of measures must be taken to ensure that once compromised, a Honeynet cannot be used to attack others. Honeynets are primarily research honeypots. They could be used as production honeypots, specifically for detection or reaction, but it is most likely not worth the time and effort. Most of the low involvement honeypots we have discussed so far give the same value for detection and reaction, but require less work and have less risk.

We have reviewed six different types of honeypots. No one honeypot is better than the other, each one has its advantages and disadvantages, it all depends on what you are trying to achieve. To more easily define the capabilities of honeypots, we have categorized them based on their level of involvement. The more involved a honeypot is, the more we can learn, but the greater the risk. For example, BOF and Specter represent low involvements honeypots. They are easy to deploy and have minimal risk. However, they are limited to emulating specific services and operating systems, used primarily for detection. Mantrap and Honeynets represent mid-to-high involvement honeypots. They can give far greater depth of information, however more work and greater risk is involved.

Legal Issues

No discussion about honeypots would be complete without exploring the legal issues involved. Honeypots are just too cool not to have some legal issues. I am not a lawyer, I have no real legal training or background. In fact, I was a History major at college, and not a very good one at that. So what I'm about to discuss is based on my own opinions and not on any legal precedent. When discussing honeypots, there are often two legal issues; entrapment and privacy. We will briefly review these issues. First, let's start with entrapment. I found the following legal definition of entrapment at the [Lectric Law Library](#) Web site:

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.

I feel honeypots are not entrapment for the following reasons.

1. Honeypots do not induce or persuade anyone, they are most often production systems, or emulate production systems.
2. Attackers find and attack honeypots based on their own initiative.
3. Most administrators are not law enforcement. They are not using honeypots to collect

evidence and prosecute. Normally they are used as a means to detect, and possibly learn about, attacks.

With privacy issues, things become a little more complicated. Does one have the legal right to collect information on an attacker? In general, I believe they do: however, there are some considerations to take into account. First, this depends on who you are and what country you are located in. For American organizations, the Department of Justice is a good resource to check, specifically [Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations](#). Some of the points include:

1. 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 the honeypot.
2. By using honeypots for communication, they have given up their right to privacy in that communication. Honeypots generally do not provide public accounts; therefore, they are not a service provider and are not bound by privacy requirements designed for service providers.
3. Most organizations are not law enforcement, nor do they act under the control of law enforcement, so they are not bound by the evidence collection restrictions otherwise placed on law enforcement and their agents. Think about it, a honeypot is collecting the same information, in the same technical manner, as many of your other security devices are (system logs, IDS sensors, etc.).

There are far too many variables involved in the legality of honeypots to adequately discuss them in this paper. It all depends on who you are, where you are located, and what you intend on doing with your honeypots. For this, I leave it to the lawyers to decide. Meanwhile, I intend on to continue playing with honeypots :)

Conclusion

Honeypots are a tool, how you use that tool is up to you. There are a variety of options honeypots, each having different value to organizations. We have categorized two types of honeypots, production and research. Production honeypots help reduce risk in an organization. While they do little for prevention, they can greatly contribute to detection or reaction. Research honeypots are different in that they are not used to protect a specific organization, instead they are used as a research tool to study and gain intelligence on threats in the Internet

community. Regardless of what type of honeypot you use, keep in mind the 'law of involvement'. This means that the more your honeypot can do, and the more you can learn from it, the more risk that potentially exists. You will have to determine what is the best relationship of risk to capabilities that exist for you. Honeypots will not solve an organizations security problems, only best practices can. However, honeypots may be a tool to help contribute to those best practices. Lance Spitzner is currently an active member of the HoneyNet Project. He enjoys learning by blowing up systems in his home lab. Before this, he was an [Tanker in the Rapid Deployment Force](#), where he blew up things of a different nature. You can reach him at lance@honeynet.org.

Relevant Links

[The Value of Honeypots, Part One](#)

Lance Spitzner

[Download BOF](#)

[Specter](#)

[Deception Toolkit](#)

[ManTrap](#)

[The HoneyNet Project](#)

[Privacy Statement](#)

Copyright 2006, SecurityFocus