Anti-Honeypot Technology

Neal Krawetz
Hacker Factor Solutions

Spammers continually scan the Internet for open proxy relays: by using these open relays, they can obscure their originating IP address and remain anonymous. However, when a spammer comes across a service on a honeypot, that honeypot can collect valuable information about the spammer's true identity and help unmask it.

In response to the threat that honeypots pose to spammers, the first commercial anti-honeypot technology has surfaced: Send-Safe's Honeypot Hunter (www.send-safe.com) attempts to detect "safe" proxies for use with bulk-mailing tools. This honeypot-detection system's appearance, in association with other emerging spam tools, suggests three important trends:

- honeypots are affecting spammers,
- current honeypot technology is detectable, and
- more honeypot-identification systems are likely.

The ability to detect a honeypot is unlikely to remain limited to spammers; other hostile or malicious groups could benefit from similar identification systems. In an effort to create undetectable honeypot systems, we need a significant improvement in today's honeypot technologies.

Basic honeypot services

Honeypots are designed to resemble valid systems. As previous columns and books have discussed, they use this cloak to collect information about attackers and their methods.[1] To appear as a tempting target, honeypots offer a variety of seemingly vulnerable services. Although the complexity of honeypot services varies dramatically, they generally fall into one of four types: minimal, restricted, simulated, and full. From low complexity to high,

- Minimal servers provide an open service port.
- Restricted servers provide basic interactions.
- Simulated servers provide complex interactions.
- Full servers provide full functional support.

Some minimal servers will reply with a basic connection header, but they usually don't perform anything more detailed. An example of a minimal service is the SMTP server that the BackOfficer Friendly (BOF) honeypot provides: it simply disconnects with the message, "503 Service Unavailable."

By adding a minor amount of interaction to a minimal server, a restricted service can appear fully functional, even though no authorization is available. The BOF telnet server, for example, prompts for a username and password, but no valid log in mechanism exists. Niels Provos and his colleagues have a Web page devoted to their honeyd project (www.citi.umich.edu/u/provos/honeyd/) that provides a series of restricted service scripts, including SMTP and a simple Web proxy.

A simulated service appears as a full working server, but in reality, it logs actions instead of performing external operations. Simulated servers accept log ins and requests, and generate well-known replies and error messages. Examples of simulated servers include scripts that emulate full SMTP and Microsoft IIS Web servers.

In contrast to these pseudo-services, full honeypot services are rare. They not only manage requests, but they also let malicious entities fully interact and even compromise the simulated system. Many full honeypots also permit limited external connections, which makes the service appear fully functional while preventing it from taking part in denial-of-service (DoS) attacks.

Although people involved with illegally trading credit-card information (carders) or black-hat hacking commonly relay through multiple proxies, most spam tools only support relaying through a single open proxy. If the honeypot acts as that proxy, then the spammer's actual IP address is disclosed to the honeypot. We then can use this information to identify the spammer.

Spammers strike back

Spam developers are generally reactive, not proactive: they only change their tools when those tools become ineffective. For example, one of the first technologies to prevent spam used hash-based filters that summa-

rized each email message's content into a hash table. Repeated hash-table entries denoted identical message content—that is, a bulk mailing. To counter hash systems, spam developers created "hash busters"—unique strings that generate different hash values. Similarly, today's bulk-mailing tools use anti-Bayesian encoding methods—such as random words, sentences, or paragraphs—to pass Bayesian filters.

The Send-Safe tool suite makes an extensive collection of bulk-advertising tools commercially available. Its bulk mailer is popular for generating spam email, and its proxy scanner can find multiple open proxy servers for obscuring a spammer's identity; its other tools include an email verifier and a tool for generating bulk instant messages.

Send-Safe's latest tool, Honeypot Hunter, suggests that spammers are aware they need to identify honeypots. Honeypot Hunter's developers imply a negative effect on their activities from honeypots in its product description (www.send-safe.com/honeypot-hunter.php):

> "Send-Safe Honeypot Hunter is a tool designed for checking lists of HTTPS and SOCKS proxies for so called "honey pots". "Honey pots" are fake proxies run by the people who are attempting to frame bulkers by using those fake proxies for logging traffic through them and then send complaints to ones' ISPs."

We can safely assume that Send-Safe's users are not the only people negatively affected by honeypots. The appearance of this honeypot-detection application implies a reactive technological escalation.

We must remember that spam tools—particularly, commercial spam tools—rarely employ unique technologies. Using carbon copies and blind-carbon copies to increase distribution is an old IRC DoS attack method. We can trace the use of mail-server "VRFY" queries and return receipts, used for verifying email addresses, to common black-hat information-gathering approaches.

Honeypot Hunter's detection methods are likely widely known in the underground community. In fact, the community probably has more sophisticated detection methods that those Honeypot Hunter uses. The free honeyd project has a default installation with fixed-response messages; administrators who don't change the default messages might unknowingly provide an attacker with a unique method for identifying the honeypot. Other detection methods—such as known application error handling, operating system fingerprinting, TCP sequence analysis, and ARP addresses—could also identify a honeypot.

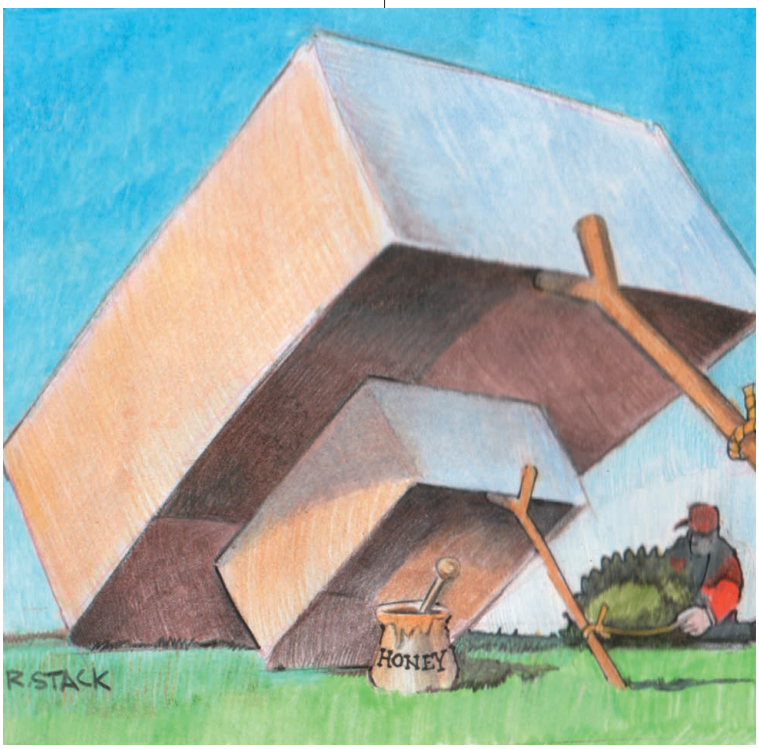
What Honeypot Hunter does

Honeypot Hunter is designed to test open proxy connectivity. Depending on the type of connection response, it classifies the proxy as safe (good), bad (failed), or a trap (honeypot). Honeypot Hunter currently tests port 1080 for Socks4 and Socks5 proxy support and all other ports for HTTP "CONNECT" proxy support.

Honeypot Hunter essentially performs a series of simple tests. First, it opens a false mail server on the local system (port 25) to test the proxy connection and then connects to the server's proxy port. After connecting, Honeypot Hunter attempts to proxy back to its own false mail server. The basic approach of connecting back to itself is enough to identify most invalid proxies and honeypots. In particular, if the remote server claims to have successfully connected, but Honeypot Hunter's false mail server didn't receive a connection, then the proxy is likely a honeypot.

The effect on honeypots

Obviously, the appearance of mainstream honeypot-detection systems has significant ramifications for honeypots. If malicious users can detect honeypots, then they can bypass detection. At minimum, this ability



```
220 kbssj.org (IMail 8.00 153-1) NT-
    ESMTP Server X1
HELO hfuhksspy.net
250 hello kbssj.org
MAIL FROM:<vhjentsqgbf@hfuhksspy.net>
250 ok
RCPT TO:<jfrqk@kbssj.org>
250 ok its for <jfrqk@kbssj.org>

DATA
354 ok, send it; end with <CRLF>.<CRLF>
From: <vhjentsqgbf@hfuhksspy.net>

Message-Id:
    <6f5501c3bb4d$7a3b2c00$f898e605@
    vhjentsqgbf
Date: Fri, 05 Dec 2003 08:33:13 -0800
Subject: fn mqecrscy
To: <jfrqk@kbssj.org>

Content-Type: text/plain;
     charset="iso-8859-1"
Content-Transfer-Encoding: 7bit

gblh psp p x ft ujtky cjkif nir
    ie etqxuce
jnyy dulx lavf pm fkggj yecer fy

tumtlejy joiv jpkf wqg gv ecs ii
.
250 message queued
    767968bda10041f999554685039d37d8]
```

Figure 1. Sample email transaction. Honeypot Hunter version 1.0.1 generated this transaction as it relays through a Socks server.

lowers the value of the information gathered because bypassed honeypots would not detect any new attacks.

More importantly, if people can detect a honeypot, they can attack it. Three basic approaches exist for attacking a honeypot: compromising, poisoning, and studying. Researchers generally place honeypots in isolated LANs adjacent to critical network junctions. By compromising this honeypot, a hostile entity could use it to stage internal attacks. Alternatively, the entity could use the honeypot to stage attacks on other systems throughout the Internet.

Instead of compromising the honeypot, a malicious user also could opt to flood the honeypot with false information. This poisoning effectively buries any valuable information under a mound of noise. By poisoning the honeypot, other hostile activities could go unnoticed.

Bypassing prevents the honeypot from collecting information and flooding obscures collected information, but an attacker could choose to use the honeypot as it's intended: for gathering information. Just as a honeypot provides valuable insights about an attacker to the observer, an attacker that compromises a honeypot could learn a lot about the observer. He or she could identify personal information such as people's names, operating hours, or skill levels. A compromised host could identify the protected network's organization, items that the organization considers "valuable," and where this value is kept. A compromised honeyd system that emulates only Windows systems, for example, would suggest a company that only uses Windows; a honeypot database server that emulates Oracle would suggest a corporate Oracle database.

Detecting the hunter

At my company, we evaluated Honeypot Hunter version 1.0 (released on 11 November 2003) and version 1.0.1 (released on 4 December 2003). Unfortunately, version 1.0 appears unstable and crashes when the false mail server is activated. Moreover, it doesn't seem to generate valid Socks5 requests. Version 1.0.1 corrected the crashing problem but still has issues around email header formatting and honeypot detection. But, the updated release indicates that Honeypot Hunter is actively and rapidly being developed.

Honeypot Hunter provides a significant amount of insight into the honeypot-detection approach. By detecting it, a honeypot could impersonate a full proxy and remain undetected. Honeypot Hunter has many identifiable aspects, including network connection methods, server identification, and test email formats.

Honeypot Hunter generates self-via-proxy connections, from the Honeypot Hunter system to the proxy and back to itself. A honeypot configured to permit self-via-proxy connections could appear viable to the tool while remaining undetected.

Honeypot Hunter's false mail server identifies itself as "`220 %s (IMail 8.00 153-1) NT-ESMTP Server X1`", where "`%s`" is replaced with a random hostname (see Figure 1). Honeypot Hunter appears to check this string with the expected reply ("`(IMail 8.00 153-1) NT-ESMTP Server X1`"), using it to detect honeypot mail servers. A Honeypot Hunter-blocking honeypot could initiate the full proxy connection and determine the mail server type. A server with a different identification would denote a non-Honeypot Hunter system and not need to deliver email sent through the honeypot.

Different email programs generate different email headers. Honeypot Hunter's test email has a fixed number of headers in a specific order with specific capitalization. A honeypot mail server that only passes messages with these specific headers would be undetectable by Honeypot Hunter:

```
From: %s
Message-Id: %s%s
Date: %s
Subject: %s
To: %s
Content-Type:
text/plain;
   charset="iso-8859-1"
Content-Transfer-
    Encoding: 7bit
```

Although these anti-detection approaches might work with Honeypot Hunter's current version, they

are unlikely to be general enough for future honeypot-detection systems. Future honeypot-detection systems will likely use additional detection techniques, different test email formats, and even a variety of test server configurations.

End of the honeypot honeymoon

Honeypot systems that use anti-detection techniques will likely lead to anti-anti-detection systems. The next logical step for Honeypot Hunter, for example, would be to split the false mail server from the Honeypot Hunter client. This change would remove the self-via-proxy connections, thus permitting connections from the Honeypot Hunter client to the proxy to a different Honeypot Hunter server under the user's control. Removing the self-via-proxy connections would make detecting the anti-honeypot system more difficult. Moreover, Honeypot Hunter's client component could relay tests through known open proxies and hide the user's true IP address from the test system.

In addition to changing its connection approach, Honeypot Hunter could change its static strings. Its false mail server could generate a variety of responses, which would make detecting it more difficult. Furthermore, elaborate anti-detection attempts to determine the actual type of system at the end of the proxy connection could give the honeypot away. Finally, the variable header formats that Send-Safe bulk mailer provides could be adapted to Honeypot Hunter; future versions of the tool thus would generate nondistinct test emails.

Extending anti-honeypot and detection techniques to better suit non-spam groups is another logical next step. It's only a matter of time before such groups begin to use honeypot-detection systems more widely—if they aren't doing so already. Currently, many Web sites list open proxies; soon they'll start listing the IP addresses of known honeypots.

Honeypots appear to have an impact on preventing spam. Unfortunately, just as spam evolves around spam filters, spammers are evolving around honeypots.

With the appearance of commercial honeypot-detection systems, honeypot operators are about to learn that their ideal monitoring solutions are not so ideal; the honeymoon with the honeypot is over, and the technical battle is about to begin.

The emerging ability to detect a trap suggests that current honeypot technology might not be adequate for much longer; honeypot technology must evolve and soon. Current limitations and simple implementations make them detectable, and changing the system might not be as simple as we would wish—a simple change to a honeypot implies a simple change to the tools for detecting it. □

Reference

1. The Honeynet Project, *Know Your Enemy*, Addison-Wesley, 2002.

Neal Krawetz *is the CSO for Hacker Factor Solutions and a senior researcher for Secure Science Corporation's External Threat Assessment Team. His research interests include anti-anonymity technology and the use of digital forensics to track individuals online. He has a PhD in computer science from Texas A&M University. Contact him at nealk@securescience.net.*