Fun things to do with a Honeypot

Alberto Gonzalez and Jason Larsen

Introduction:

Honeypots are a hot topic in the security research community right now. Everyone is starting up their own honeypot system. While most of current literature available on them deals with the potential gains a honeypot can give you, and how to monitor them, not very many of them deal with the mechanics of honeypots themselves.

Most honeypots as deployed from spare parts. Many start as just an extra box someone has lying around. A security savvy technician has slapped an OS on it, checksummed all the files, installed an IDS, and set about waiting for the hackers to arrive. These haphazard kinds of honeypots ignore some of the most interesting capabilities of honeypots. Honeypots can be used to ensnare and beguile potential hackers, entice them to give you more research information, and actively defend a production network.

In this paper, you'll find some cool and fun things to do with honeypots. We'll discuss techniques that can be used to create an environment that keeps a hackers interested piqued in your honeypot, will encourage them to upload new toys, and show you how to extract the maximum amount of data from them.

Simulated Traffic:

One of the reasons most people don't see interesting hacker tactics on their honeypot is because they've neglected to make sure that there is something interesting for the attacker to play with. If you're going to discover intimate details about the attacker's modus operandi, you need to make your honeypot interesting. One of the easiest ways to accomplish this is to create simulated traffic coming to and from the honeypot.

Replaying interesting traffic on the network can prompt the attacker to investigate other portions of your honeypot. Simulated traffic replayed over the wire can include e-mails, passwords, hostnames, etc… You want juicy, intriguing traffic to entice the attacker to further investigate your machine and or network (honeynet).

Simulated traffic can be used in conjunction with simulated targets. This allows you to replay traffic from those simulated hosts to lure the attacker to further investigate those targets. Such traffic could be pop3, samba, ftp, and http traffic coming from the simulated targets. Traffic from services known to have a infamous vulnerability history will definitely prompt the attacker to further investigate.

If you want to really see the hacker in action, simulate traffic that looks like someone trading MP3s and traffic that looks like someone transferring business documents. If the attacker spends most of his time looking at the MP3 traffic, he's probably pretty harmless. If he spends his time looking at the documents, he's much more likely to be dangerous.

Simulated traffic can be used as a kind of referral service among honeypots. Drop some packets on the wire that contain usernames and passwords or contain hints that the choice morsels are stored at a different location. Different breeds of attackers will chase down different leads and attack your other honeypots.

Simulated Targets

Once an attacker has taken all the trouble to set up shop on your honeypot, he'll probably want to explore the system further. If your honeypot is typical, there's not much for an attacker to do once he gains root access. He needs instead to be confronted with a challenge that will encourage him to transfer all the other toys in his arsenal so you can have a copy as well.

Providing an attacker with additional targets such as various operating systems and services can encourage him to use other programs and scripts. the targets can be real, but it is safer and you'll get almost as much mileage if they're simulated. A good place to start is to put a phantom private network up hung off the back of the honeypot.

Most corporate networks are divided into a private internal network and a public DMZ. It's a poor security practice to find direct links from DMZ machines into the private network, but they are often present. If the attacker takes over the box and finds such a link, He's probably going to want to explore it. You can create whatever environment you want for him to explore. It should probably include a number of different operating systems running different services.

Hopefully, the attacker will spot a service he has an exploit for and try to take it over. When the attacker transfers down the exploit, you'll get a copy to add to your library. The more compelling you make the simulated backend network; the more likely you'll get additional toys.

Switching to a vulnerable OS/Service:

"You have an exploit for a Wu-FTP server? I have one of those. Here you go."

Keeping your production servers patched is a must, but keeping your honeypot patched just limits the amount of fun you can have with it. New exploits are generated for old vulnerabilities all the time. If you just ignore those exploits, you'll miss what's going on behind the scenes in the root kit development, distributed hacking tools, and anything else that requires them to actually get on your box.

Nearly everyone that attempts an exploit has useful data to give. The trick is getting it from them. The best way to extract all the data is to let the exploit succeed and watch to see what they do. Even if they use an old exploit, they may use a new root kit or start up an IRC session that will lead you to the more fertile ground of zero day exploits. If someone has an exploit and takes time out of their busy day to send it at your network, the least you can seemingly do is to oblige them with a root shell.

To build an OS/Service switch, you'll need a public box, a switching box, and a number of boxes with various vulnerable services loaded on them. To cut down on the amount of real hardware, the vulnerable boxes can be replaced with Vmware instances.

The switching box is an inline box with multiple interfaces. All new traffic is routed to the public box by default. Whenever and exploit is attempted on the public box, the IDS on the switching box looks up the OS and revision of attack and switches it over to the appropriate target.

The operating system and services of the public box are what the attacker is going to see when he scans the box. You can use a few tricks to get people to try more exploits. One is to obfuscate the banners. Instead of having your web server identify itself as Apache, Identify it as "Foobar.com front end proxy for Apache 1.3.19 and IIS 5.0". Be creative! The objective is to get attackers to throw exploits at the honeypot. Remember, though, what you are really interested in is what he does after the exploit.

After you are sure that you have extracted all useful data from a particular set of attackers, you can use utilities such as Hogwash or Snort-Inline to filter out that particular exploit or that particular root kit. The attacker may respond by changing their root kit or modifying their exploit in some way. The fact that he does and the methods by which this are accomplished are very interesting data indeed!

A major difficulty in running such an open honeypot is the recovery time. After each break-in, you need to flush evidence from the previous attacker so that everything is reset for the next one. The two most popular methods are using ghost or vmware. If you opt for ghost, you can simply ghost the drive before you put it up on the network and then restore the image as needed. With Vmware, you can keep a copy of the hacked image in an archive and the restart with a clean Vmware image.

I've seen a few honeypots where the administrators used a file system mounted on a loop back interface. I believe they met with limited success. There are also some people experimenting with user-space Linux. It looks promising.

Traffic Mangling:

Once you've got the Wiley hacker attacking your honeypot, it is imperative that you defend against the possibility that he might launch an attack on the rest of your network from the honeypot, or worse yet, attack a third party network. A good line of defense in this instance is traffic mangling.

Traffic mangling requires an inline box running software such as Hogwash. The inline box can replace parts of an exploit with a broken equivalent. An example of a common mangler is to replace all instances of /bin/sh coming from the honeypot with /bin/hs. The attacker's attempt to execute a shell on the remote box will fail.

This particular mangler has provided me with hours of entertainment while I watched the attacker download his debugging tools, source code, and favorite traffic analyzers in vain attept to discover why his exploits weren't working.

A good policy is to set up manglers for all the exploits you can get your hands on and then some general rules such as replacing all "sam._" with "mas._". It's impossible to stop all outgoing exploits with manglers, but it can give you peace of mind that the outside world is relatively protected from your compromised honeypot along with hours of fun watching attackers failed attempts to continue their attacks elsewhere.

This implementation can be considered a form of data control that every honeypot/net should employ. Data control is a defense mechanism to stop attackers from attacking other machines or networks on the internet from your honeypot.

Connection and Byte limiting:

Connection limiting can be used for both ingress and egress traffic. Connection limiting like traffic mangling can provide you many hours of enjoyment watching intruders not understand why they can't have multiple outbound/inbound connections. If you only allow certain number of outbound connections and vice versa, these method can be somewhat easier to fingerprint, thus hinting to the attacker that he is currently on a honeynet or a system with traffic control.

You can limit n number of connections inbound per x time frame. This would allow you control over your honeypot system in an attempt to control inbound recon and exploitation attempts. I

have seen multiple compromises happen simultaneously. Egress connection limiting is a must for most honeypots. There are a number of ways you can go about it. You can restrict the honeypot to n simultaneous outbound connections. This will stop a number of DDOS agents and port scanning tools. This will also limit the damage an attacker can do by attempted port-scans or exploits of external hosts.

Network folks and your ISP will take an active interest in your honeypot is if you're infected with a DDOS agent. Most of the time the network administrator has his pager set to go off when the external link hits 100%. Unfortunately for him, this usually happens at 3 o'clock in the morning.

The number of bytes transferred per second in-bound or outbound can be limited; this method would be employed to stop the DDOS situation discussed above. This could halt some exploit attempts (e.g.: FreeBSD telnetd exploit). Unlike connection limiting, byte limiting is somewhat harder to fingerprint. A more elegant approach is to set the TCP window size in each packet to a small number. Although any of these methods will help, you should probably have a general purpose "kill the honeypot if you see this" process running somewhere.

Bait-n-Switch:

The most basic, but among the most useful concepts a honeypot can be used for is to divert hackers from attacking your production network. This is commonly known as the bait-and-switch method. Bait-and-Switch consists of a production machine, the bait-and-switch machine, and a honeypot.

A Bait-n-Switch honeypot is comprised of three machines: your real web server, which can be an exact mirror of your web server minus all the sensitive data, and a BNS box. Both the Honeypot and the Production web server are plugged directly into the BNS box. The BNS box runs an Intrusion Detection System. When the IDS alert's that someone is an attacker, it starts redirecting the attacker's packets to the Honeypot instead of the production machine.

On most networks having two machines with the same IP address is a bothersome, problematic occurrence, but that actually works in your favor with a BNS style honeypot. If the honeypot has the same IP and MAC address as the production server, the attacker may not notice that he's been switched. If he does not notice, you will get to see all the fun things he has planned for your production server. If he does notice that he no longer has access to the production server, he may instead quit and go away.

One current implementation of this approach is The Bait N Switch Project from Violating Networks. This method has defensive and research capabilities rolled into one system. Research comes into play when the attacker is switched and targeting the compromised honeypot (assuming the attack was successful). You have successfully defended your production machine and now have further research information on the attacker.

Honeypots and the law:

Whenever the topic of honeypots comes up, invariably there is someone who wants to debate the legalities of this or that. We're not lawyers, but here are some things you should think about when those inevitable discussions come up.

1) Entrapment is not a crime. It only applies to law enforcement and is only used as a defense to keep from going to jail. A normal citizen can't entrap anyone even if he really wants to.

2) Trials generally have a bunch of people just like you in a jury box. If you're just trying to protect your networks, they will understand that. The legal system isn't quite that messed up.

3) Most of the time, the lawyers only get involved when there's enough money to make it worth their time. The FBI and other law enforcement generally functions the same way.

4) Unless you're prosecuting them, the chances of an attacker bringing any sort of legal action against you is near zero.

5) I've port scanned someone I don't know at least once a day for the last five years. I haven't seen the inside of a court room yet.

Conclusions:

Honeypots are a serious research endeavor, but you can garner some entertainment from them. Your fun will translate into interesting stuff for the attacker to play with. The attacker is more likely to spend time and intellectual effort with an interesting site than with one that is boring and unchallenging. He probably already has all the credit card numbers and free porn he wants, but he may be willing to send you a few more exploits for the chance to read about the affair you're having, or better yet planted fake marketing plans, emails of problems with product development, and mock ups of product specifications.

There's no rule that says the network topology has to be anything conventional when you're setting up your honeypot. Once someone logs in, you can present fictitious new hosts, traffic, subnets and, if you want to get fancy use vLANS, they don't have to exist. After all, they are only packets; you can craft requests and replies as well as any hacker.

A honeypot is an illusion that you weave for the attacker. Your illusion can be as creative as you want it to be. A good illusion will net zero day exploits, root kits, and loads of information on how attackers work. When developing a honeypot, challenge yourself and have fun with it.

About the Authors:

Jason Larsen is the Network Security Architect for the Idaho National Engineering and Environmental Laboratories, a DOE nuclear research lab in central Idaho. He is also the main developer of the Hogwash inline packet scrubber. He's also been published on a number of online security journals as well as medical journals.

Alberto Gonzalez is a Traffic/Intrusion Analyst with EDS out in Northern Virginia. He is also Founder of Violating Networks. He contributes to various open source projects including The Bait N Switch Honeypot and Hogwash. He is currently in the process of getting his GCIA certification from SANS.