Advanced Network Security

Honeypots

Amir Mahdi Sadeghzadeh, Ph.D.

Most slides have been adapted from Mehdi Kharrazi, CE40-817, Sharif University of Technology, 2015

Honeypots

- A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource.
- Anything going to or from a honeypot is likely a probe, attack or compromise.
- Primary value to most organizations is information.

Why HoneyPots

- A great deal of the security profession and the IT world depend on honeypots.
- Honeypots
  - Build anti-virus signatures.
  - Build SPAM signatures and filters.
  - ISP's identify compromised systems.
  - Assist law-enforcement to track criminals.
  - Hunt and shutdown botnets.
  - Malware collection and analysis.

Advantages

- Collect small data sets of high value.
- Reduce false positives
- Catch new attacks, false negatives
- Work in encrypted environments
- Simple concept requiring minimal resources.

Disadvantages

- Limited field of view (microscope)
- Risk (mainly high-interaction honeypots)

Types

- Low-interaction
  - Emulates services, applications, and OS's.
  - Low risk and easy to deploy/maintain, but capture limited information.
- High-interaction
  - Real services, applications, and OS's
  - Capture extensive information, but high risk and time intensive to maintain.

Honeypots

- The more you allow the attacker to do
  - → the more you can learn.
  - → the more harm they can potentially cause.

Low to High Interaction

- Minimal servers
  - Provide an open service port
  - SMTP service provided by the BOF honyepot
  - simply disconnects with message
    - “503 Service Unavailable”
- Restricted servers
  - basic interactions
  - service seems fully functional
  - BOF telnet service
  - prompts for user/pass
    - no valid user/pass exists

Low to High Interaction (con't)

- Simulated servers
  - complex interactions
  - appears as a full working server
  - but logs actions instead of performing external operations
  - accept logins/requests, generate well known responses
- Full servers
  - full functional support
  - Lets the attacker fully interact and even compromise the simulated system
  - Allows limited external connections
    - Make it look fully functional
    - While preventing taking part in a DOS

Honeyd

Honeyd Overview

- Honeyd is a low-interaction virtual honeypot
  - Simulate arbitrary TCP/UDP service
    - IIS, Telnet, pop3…
  - Supports multiple IP addresses
    - Test up to 65536 addresses simultaneously
  - Supports ICMP
    - Virtual machines answer to pings and traceroutes
  - Supports integration of real system
    - Service can be proxied and redirected

Honeyd Overview

- Logging support
  - Simple connection log
  - Complete packet log
- Configuration via simple configuration file
  - Template
  - Route topology
- Limitations
  - Available services are small
  - Does not simulate the whole operating system

Honeyd Design

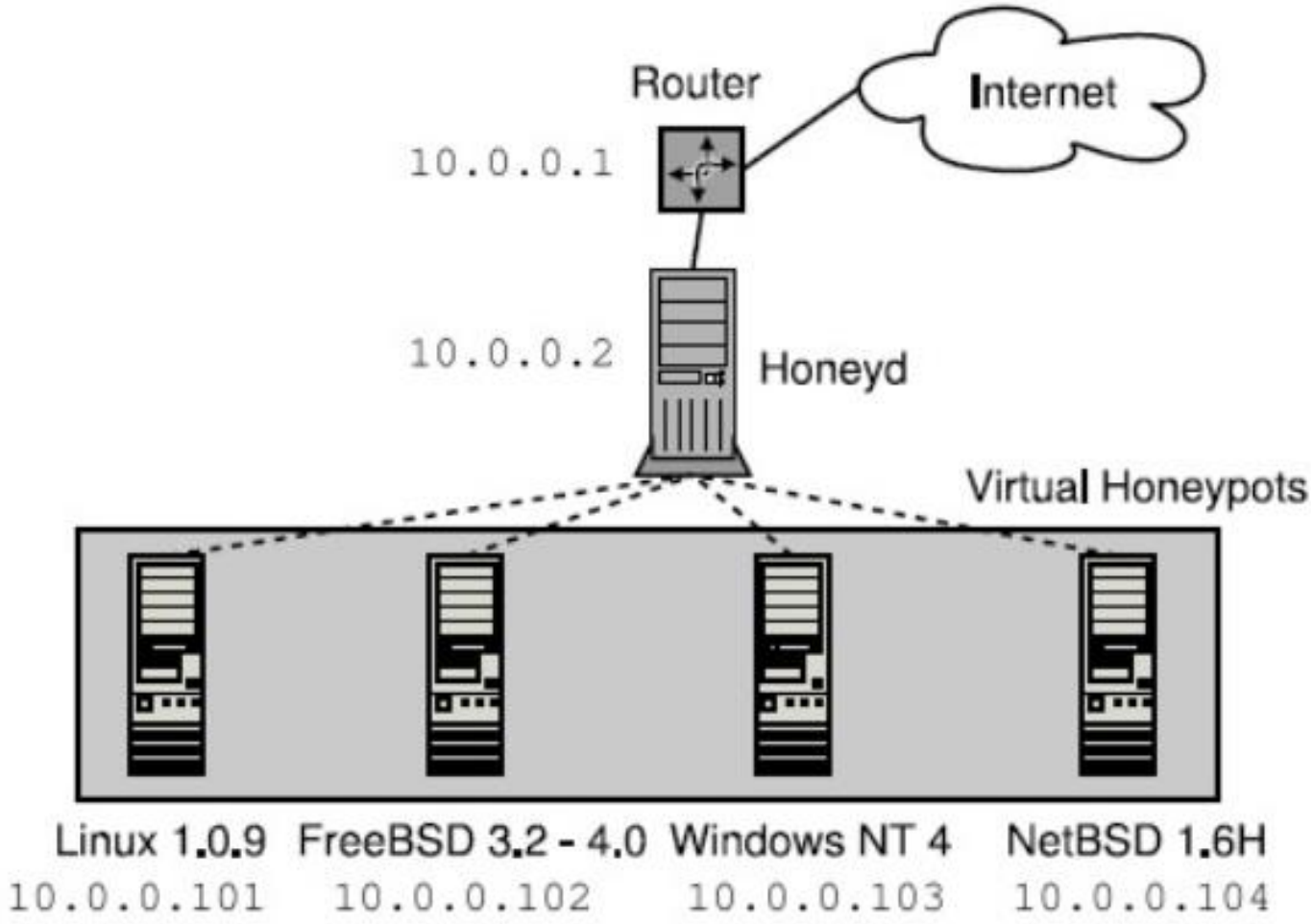
- Considerations
  - Network Architecture
  - Simulating honeypots
    - Simulate only network stack behavior Instead of simulating every aspect of an operating system
    - Simulate arbitrary network topologies
  - Security of the honeyd host
    - Limit adversaries to interacting with honeypots only at the network level. An adversary never gains access to a complete system
  - Connection and compromise attempts capturing
    - LOGS

Honeyd Design

- Design and Implementation
  - Receiving Network Data
  - Architecture
  - Personality Engine
  - Routing Topology
  - Logging

Receiving Network Data

- Two ways for Honeyd to receives traffic for its virtual honeypots
  - Special route lead data to honeyd host
  - Proxy ARP for honeypots

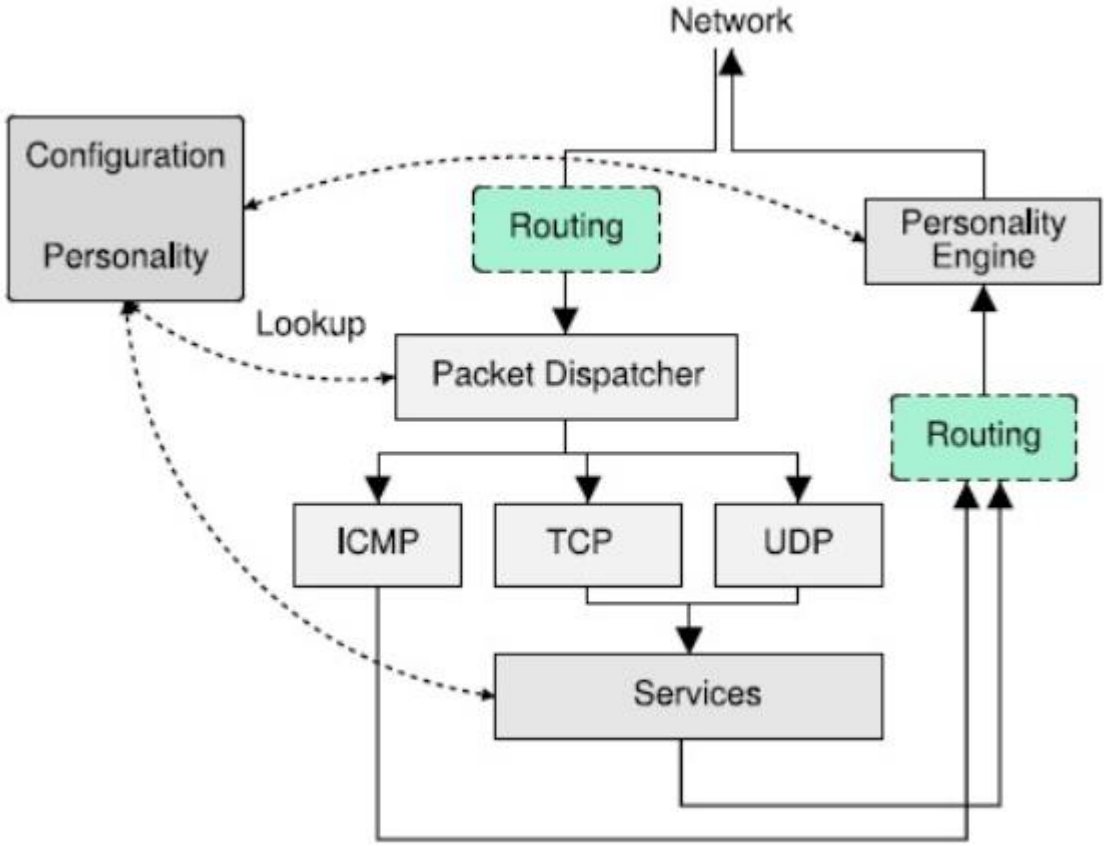


Arpd

- Proxy ARP tool: Arpd
  - Arpd is a daemon that listens to ARP requests and answers for IP addresses that are unallocated.
  - Using Arpd in conjunction with Honeyd, it is possible to populate the unallocated address space in a production network with virtual honeypots.

Architecture

- Configuration database
  - Store the personalities of the configured network stack.
- Central packet dispatcher
  - Dispatch Incoming packets to the correct protocol handler.
- Protocol handles
- Personality engine



Personality Engine

- Why do we need Personality Engine?
  - Different operating system have different network stack behaviors.
  - Adversaries commonly run fingerprinting tools like Xprobe or Nmap to gather information about a target system.
  - Personality Engine make honeypots appear like real target to a probe.

- Every packet generated by honeyd passes through the personality engine
  - Introduces operating system specific quirks into packets for Nmap/Xprobe identification.
  - Nmap fingerprint database reference for TCP/UDP connection.
  - Xprobe fingerprint database reference for ICMP request.

Personality Engine

- Ex: Personalities defined via Nmap fingerprint file
  - Create windows
  - Set windows personality “Microsoft windows NT 4.0 SP5-SP6”
  - add windows tcp port 80 "perl scripts/iis-0.95/iisemul8.pl"
  - add windows tcp port 139 open
  - add windows udp port 137 open
  - set windows default tcp action reset
  - set windows default udp action reset

  - bind 10.0.0.51 windows
  - bind 10.0.0.52 windows

Routing Topology

- Honeyd supports the creation of a complete network topology including routing
  - Simulation of route tree
    - Configure a router entry point
    - Configurable latency and packet loss
    - Simulation of arbitrary route
- Extension
  - Integrate physical machines into topology
  - Distributed Honeyd via GRE tunneling

Routing Topology Define

- route entry 10.0.0.1
- route 10.0.0.1 add net 10.1.0.0/16 latency 55ms loss 0.1
- route 10.0.0.1 add net 10.2.0.0/16 latency 55ms loss 0.1
- route 10.1.0.1 link 10.1.0.0/16
- route 10.2.0.1 link 10.2.0.0/16

- create routerone
- set routeone personality “Cisco 7206 router (IOS 11.1(17)”
- set routerone default tcp action reset
- set routerone default udp action reset

- bind 10.0.0.1 routerone
- bind 10.1.0.1 routerone
- bind 10.2.0.1 routerone

Logging

- The Honeyd framework supports several ways of logging network activity.
  - Honeyd creates connection logs to report attempted and completed connections for all protocols.
  - Information also can be gathered from the services themselves and be reported to Honeyd via stderr.
  - Honeyd can be runs in conjunction with a NIDS.

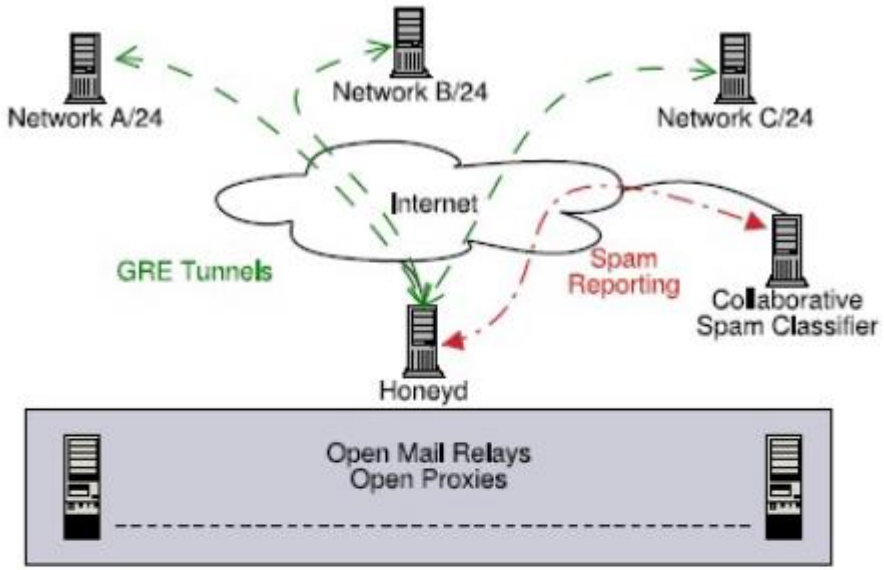
Applications

- Network Decoys
  - Instrument the unallocated addresses of a production network, confuse and deter adversaries scanning the production network
  - Conjunction with a NIDS, the resulting network traffic may help in getting early warning of attacks.

- Detecting and Countering new Worms
  - Deploy a large number of virtual honeypots as gateways in front of a smaller number of high-interaction honeypots.
  - Use Honeyd's subsystem support to expose regular UNIX applications like OpenSSH to worms.

Applications

- Spam prevention
  - Spammers abuse two Internet services: proxy servers and mail relays.
  - To understand how spammers operate we use the Honeyd framework to instrument networks with open proxy servers and open mail relays.
  - Use of Honeyd's GRE tunneling capabilities and tunnel several C-class networks to a central Honeyd host



Using the Honeyd framework, it is possible to instrument networks to automatically capture spam and submit it to collaborative filtering systems.

Honeynets

Honeynets

- High-interaction honeypot designed to capture in-depth information.
- Its an architecture you populate with live systems, not a product or software.
- Any traffic entering or leaving is suspect.

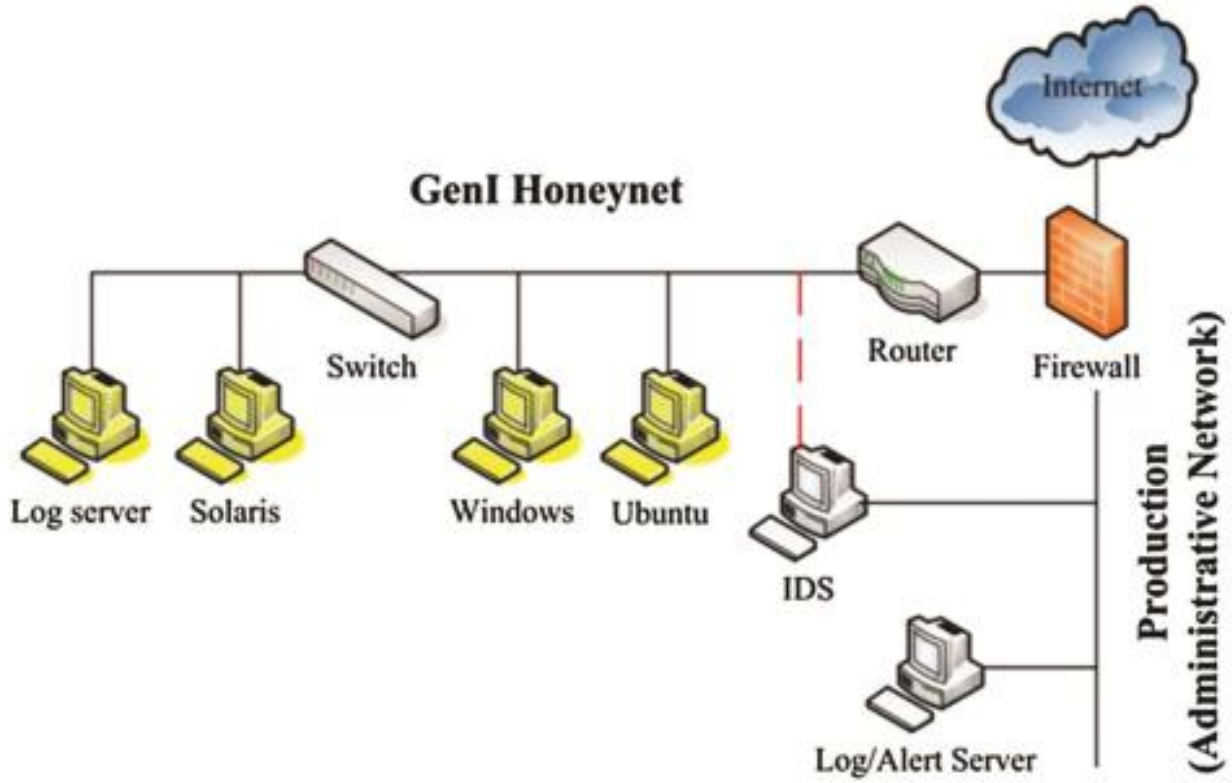
How it works

- A highly controlled network where every packet entering or leaving is monitored, captured, and analyzed.
  - Data Control
    - It is aimed to mitigate the risk that the adversary uses the compromised honeypot to attack other non-Honeynet systems
  - Data Capture
    - Log all of the attacker's activity for later investigation.
  - Data Collection
    - Data Collection requirement proposes the secure means of centrally collecting all of the captured data from distributed honeynets.

- the Data Control, Data Capture and Data Collection should be hidden or camouflaged to avoid the adversary to be aware of these honeynet activities.

Honeynet Architecture, Generation I

- As can be seen, a firewall separates the network into three different parts: Internet, Honeynet and Administrative Network.



Honeynet Architecture, Generation I

- Firewall
  - The firewall keeps the track of any connection that has been made between the honeynet and the Internet.
  - The firewall can block the outbound connections once a defined limit has been reached for the goal of Data Control.
  - On the other hand, the firewall also logs all connections to and from the honeynet for the goal of Data Capture.
- Router
  - It hides the firewall from the attacker and provides the attacker a more realistic network with a production router in order to keep the attacker from becoming suspicious

Honeynet Architecture, Generation I

- IDS
  - The IDS connects to the honeynet via a physical switch. Thus, the IDS can use a “port monitoring” port enabling it to record all network activity for the goal of Data Capture.

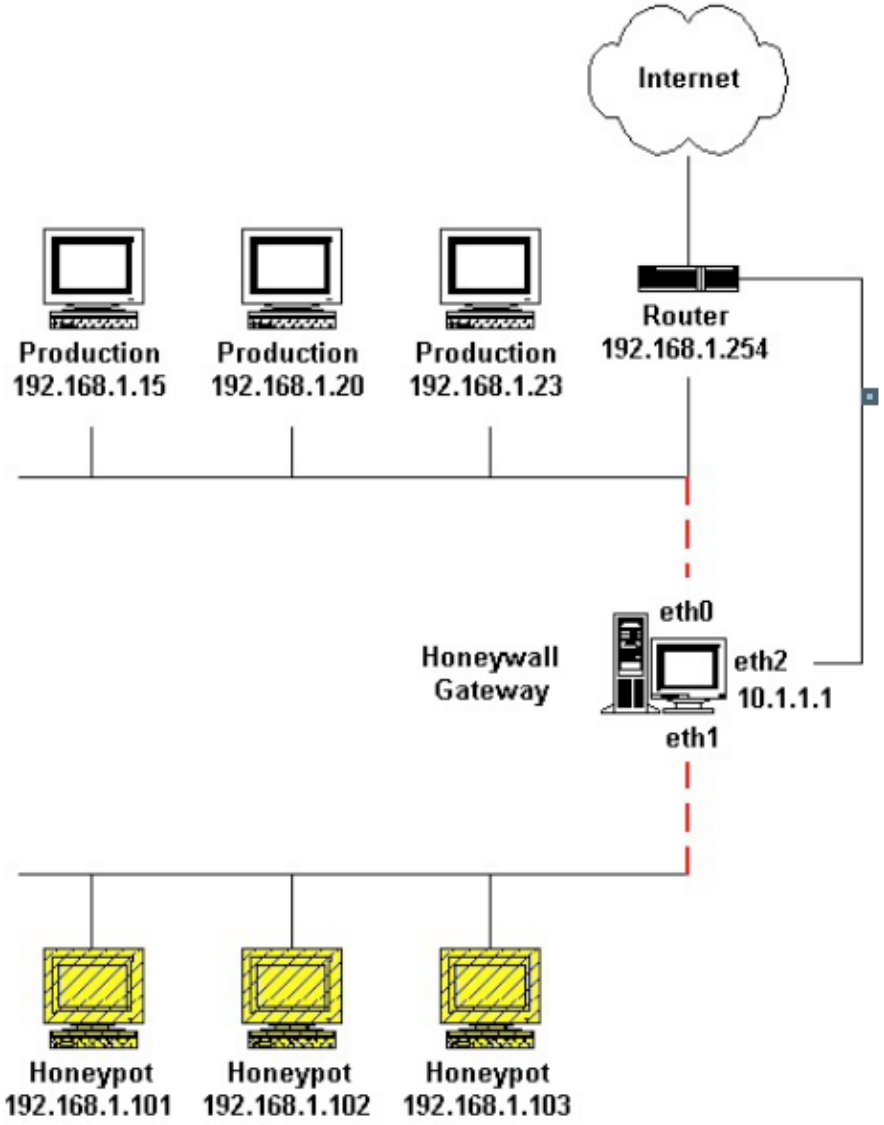
- If the attacker reaches the honeynet
  - The honeypots capture all system activity and the remote log server can provide the centralized Data Collection. If an advanced attacker detects the syslog and even compromises the remote log server, we still have the IDS that act as a backup remote log system.

Honeynet Architecture, Generation I

- Disadvantages
  - First of all, if the limit of outbound connections is reached and all of the attacker's outbound activity is blocked, the attacker will suspect he is in a honeynet environment.
  - Secondly, the firewall is easy to be identified since all traffic that passes through the firewall has TTL (time to live) decrement.
  - The attacker can use encryption technology such as protocol SSH to transfer the data. Thus, IDS will fail to monitor the attacker's connection.

Honeynet Architecture, Generation II

- Honeynet Project released a honeywall, which allowed for Gen II Honeynet architecture and improved both Data Capture and Data Control capabilities over Gen I Honeynet architecture



Honeynet Architecture, Generation II

- Honeywall
  - It is traditionally a layer 2 bridging device with three interfaces.
    - Two of them (eth0 and eth1) are used to segregate the honeypots traffic from the production network. They are bridged interfaces with no IP stack.
    - The third interface (eth2, which is optional) has an IP stack and is used for remote administration.
    - The honeywall combines the functionality of both the IDS and the firewall in a single system, which can perform attack control and network activity logging.

Honeynet Architecture, Generation II

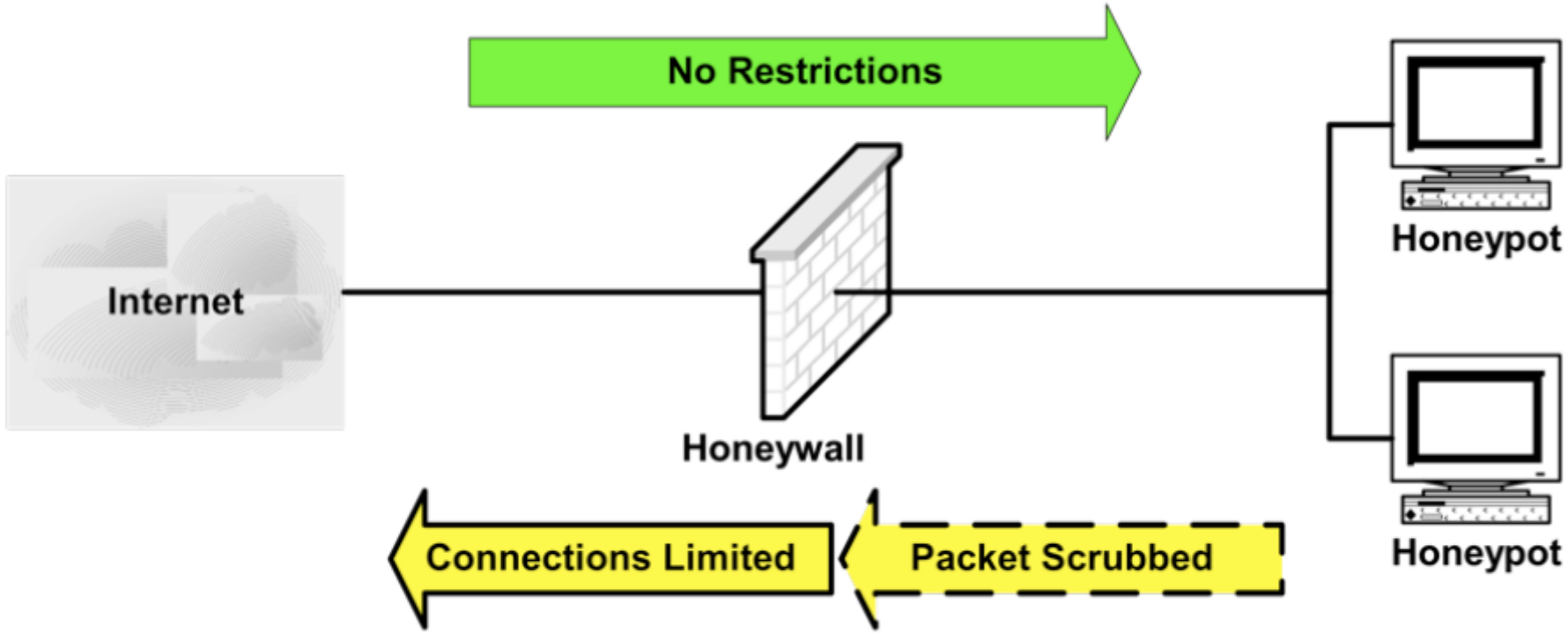
- Advantages of using the honeywall
  - The honeywall is implemented as a transparent bridge to the attacker, meaning the device should be invisible to anyone interacting with the honeypots.
  - Any traffic going to or from the honeypots must go through the honeywall but there is no routing of packets, no TTL decrement of system hops, thus the honeywall is hard to detect.
  - The honeywall divides the honeynet from the production network at layer two, as opposed to layer three.
    - The honeynet deployment can be part of a production network instead of being on an isolated network.
    - Proxy ARP

Data Control

- Mitigate risk of honeynet being used to harm non-honeynet systems.
  - Count outbound connections.
  - IPS (Snort-Inline)
  - Bandwidth Throttling

No Data Control

Data Control



Snort-Inline

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 53
(msg:"DNS EXPLOIT named";flags: A+; content:"|
CD80 E8D7 FFFFFF|/bin/sh";
```

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

Data Capture

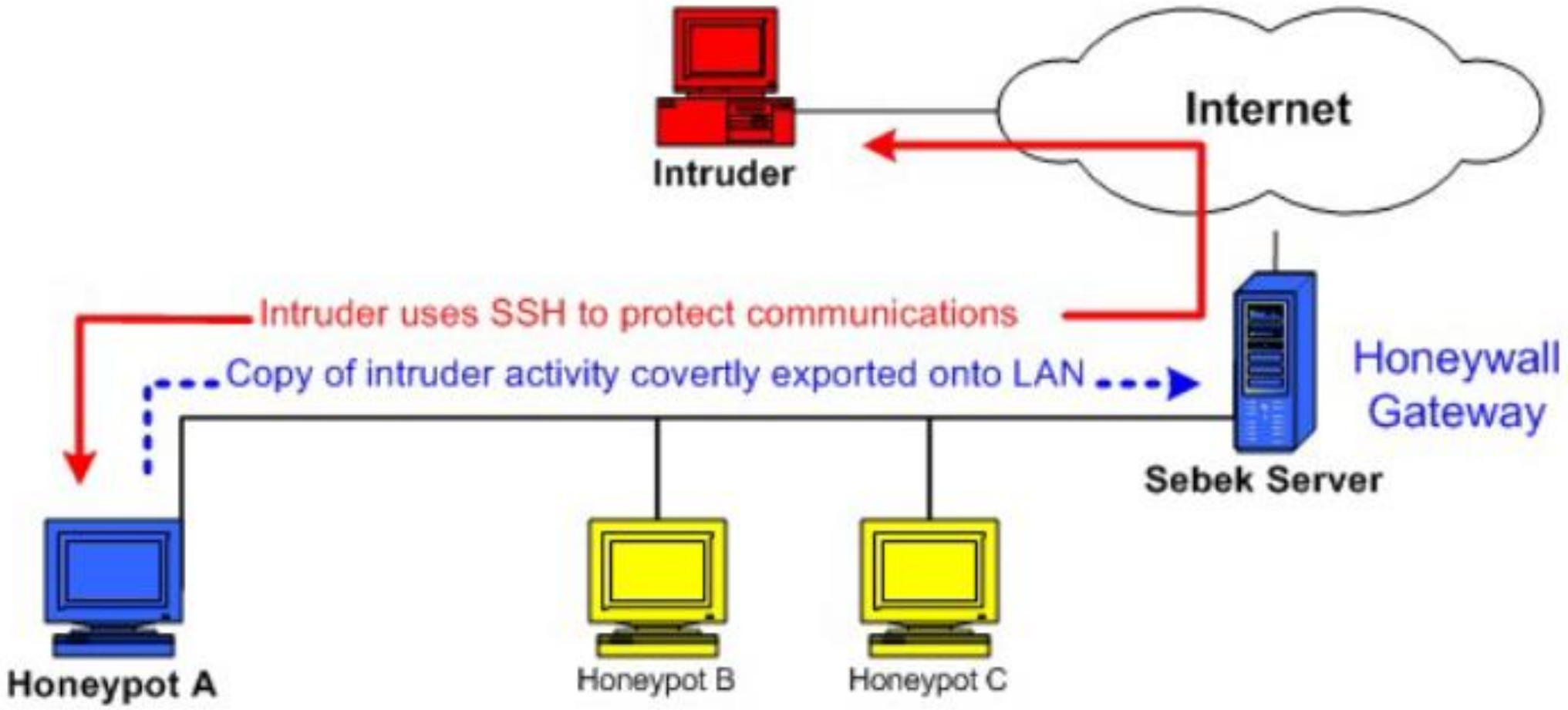
- Capture all activity at a variety of levels.

- Network activity.
- Application activity.
- System activity.

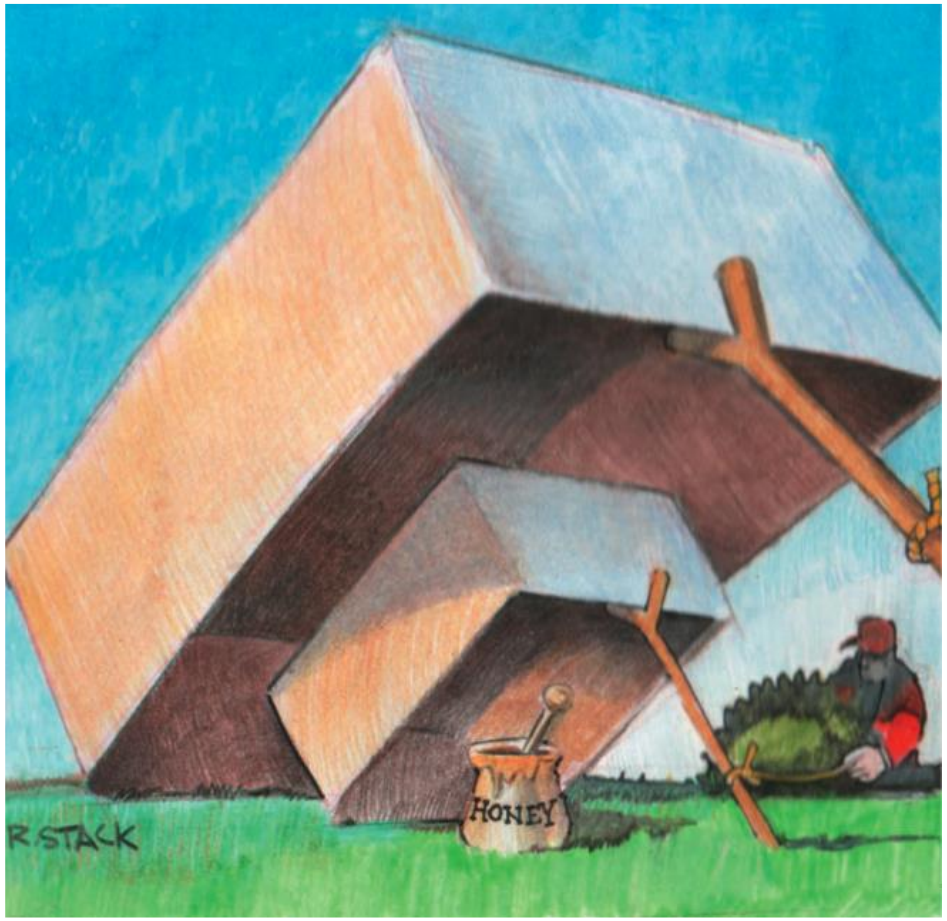
Sebek

- Hidden kernel module that captures all host activity
- Dumps activity to the network
- Attacker cannot sniff any of the Sebek originated packets

Sebek Architecture



Attacks on Honeypots



Attacking the Honeypot

- Compromising
  - Honeypots usually placed in isolated LANs adjacent to critical network infrastructure
  - Good stage for internal attacks
- Poisoning
  - Provide false information
  - Bury valuable info by producing noise
- Spying
  - Assume honeypot is compromised
  - Identify personal info (names, working hours, expertise level)
  - Learn about the internals of the organization
    - If honeypot only emulating windows ...
    - if honeypot emulating Oracle

Honeypot - Local Detection

- Technical properties of the honeypot
  - Respond times, banners, registry entries, inconsistent parameters
- "Social" properties of the system, user interaction
  - No typical usage (e.g. no new files created or accessed on a server for more than a week…)
- Network sniffing
  - Packets going to/from the system (sniffing may be done from a different system on the network if possible)
- Search for traces of Vmware
  - Vmware is a popular platform for honeypots, but it can be detected locally

Honeypot - Local Detection (con't)

- Search for traces of honeypot tools
  - Temp folders, kernel dumps, backdoors (sebek etc.)
- Search for the history files/logs and other configuration errors
  - Not only bad guys make mistakes :-)
- Vulnerabilities/exploits for the honeypot product itself (low- or medium interaction honeypots only)
- Just be creative :-)

Examples of honeypot detection (cont.)

- Inconsistencies in TCP/IP stack (remotely detectable):
  - Tools like hping can be used to detect incorrect TCP/IP stack emulations indicating the use of a low-interaction honeypot:

  1. Normal RH9: TTL=64, window=0, id=0, DF
  2. RH9 on vmware: TTL=64, window=0, id=0, DF
  3. RH9 on honeyd: TTL=64, window=1460, id=0, DF

- This method works even better on Unix systems emulating Windows and vice versa:

  1. Normal Win2k SP4: TTL=128, window=0, id=+, DF
  2. honeyd emulating Win2k SP4: TTL=64, window=1460, id=0, DF

- The interesting elements of a packet are: Time to live, window size, IPID and Don’t Fragmentation-Bit

Overview of different TCP/IP stacks

- A list of properties of different TCP/IP stacks could easily be build (e.g. with hping):

OS	Platform	Vendor	Device/System	Default TTL	WINDOW SIZE	ID	DF bit
AIX 4.2.1	R6000	IBM	n/a	60	16384	+	Y
AIX 5.2	R6000	IBM	n/a	60	16384	+	N
FreeBSD 4.7	Intel	FreeBSD	n/a	64	57344	+	Y
Linux 2.4.20	Intel	Gentoo	n/a	64	32767	0	Y
Linux 2.4.20	Intel	Debian	n/a	64	5840	0	Y
Linux 2.4.21	Intel	SuSE	n/a	64	0	+	Y
Linux 2.4.21	Intel	RedHat	n/a	64	5840	+	Y
OS/400 5.1	Intel	?	n/a	64	8192	+	Y
Solaris 2.5.1	Sparc	Sun	n/a	255	9112	+	Y
Solaris 2.6	Sparc	Sun	n/a	255	9112	+	Y
Solaris 2.7	Sparc	Sun	n/a	255	9112	+	Y
Solaris 2.7	Sparc	Sun	n/a	255	9112	+	Y
Solaris 2.8	Sparc	Sun	n/a	255	24656	+	Y
Solaris 2.9	Intel	Sun	n/a	60	65392	+	Y
Windows 2000 Professional SP3	Intel	Microsoft	n/a	128	64512	+	Y
Windows 2000 Professional SP3	Intel	Microsoft	n/a	128	64240	+	Y
Windows 2000 Server SP4	Intel	Microsoft	n/a	128	65535	+	Y
Windows 2003 Server Standard	Intel	Microsoft	n/a	128	16616	+	Y
PIX 6.2.2	?	Cisco	n/a	257	4096	+	N
FreeBSD 4.9	Intel	FreeBSD	n/a	64	57344	+	Y
D-Link DWL-900+ Wireless AP	?	D-Link	Wireless AP	127	8192	+	N
Linux 2.4.24	Intel	Kernel.org	n/a	64	5840	0	Y
Solaris 2.8	Intel	Sun	n/a	60	65392	+	Y
Fiberline Broadband Router	?	Fiberline	Broadband Router	60	4096	+	N

VMware detection

- Detect installed VMware-tools
- Detect VMware magic value (0x564D5868)
  - This is a special I/O Port used by the VMware-tools to communicate between the Host system and the virtual system. Can be used for funny tricks, too (move mouse, set clipboard, pop-up dialogs, …).
  - read more at: http://www.codegurus.be/codegurus/Programming/virtualpc&vmware_en.htm
- VMware fingerprinting checks for standard virtual VMware devices (e.g. processor, ioport, scsi, …)
- Anomalies in VMware configuration (Intel Pentium4 2,6GH with only 128M RAM??? or an unusual amount of system memory such as 96MB or 224MB)

Honeypot Hunter

- Send-Safe's Honeypot Hunter
  - First commercial anti-honeypot technology
- Attempts to detect "safe proxies for use with bulk mailing tools"
  - Honeypots are effecting spammers
  - current honeypot technology is detectable
  - more honeypot identification systems are likely
- depending on the response classifies the proxy as:
  - safe (good)
  - bad (failed)
  - trap (honeypot)

Simple Test

- Hunter opens a false mail server on the local host (PORT 25)
- Connects to the server proxy port
- Hunter then attempts to proxy back to it's own false proxy server
  - Most honeypots will be detected like this
  - If the server states that it did connect and it did not, then ...

Acknowledgments/References

- [honeypot] The honeypot project, http://www.honeynet.org/speaking/index.html
- [Masud] Digital Forensics, Bhavani Thuraisingham, Department of Computer Science, The University of Texas at Dallas, Fall 2007
- [Liang] Chinese Honeypot Project, Zhiyin Liang, October 2004
- [Krawetz] Anti-Honeypot Technology, Neal Krawetz of Hacker Factor Solutions, IEEE SECURITY & PRIVACY , 2004.