CE 817 - Advanced Network Security
Honeypots

Lecture 12

Mehdi Kharrazi
Department of Computer Engineering
Sharif University of Technology

Acknowledgments: Some of the slides are fully or partially obtained from other sources. Reference is noted on the bottom of each slide, when the content is fully obtained from another source. Otherwise a full list of references is provided on the last slide.

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.

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

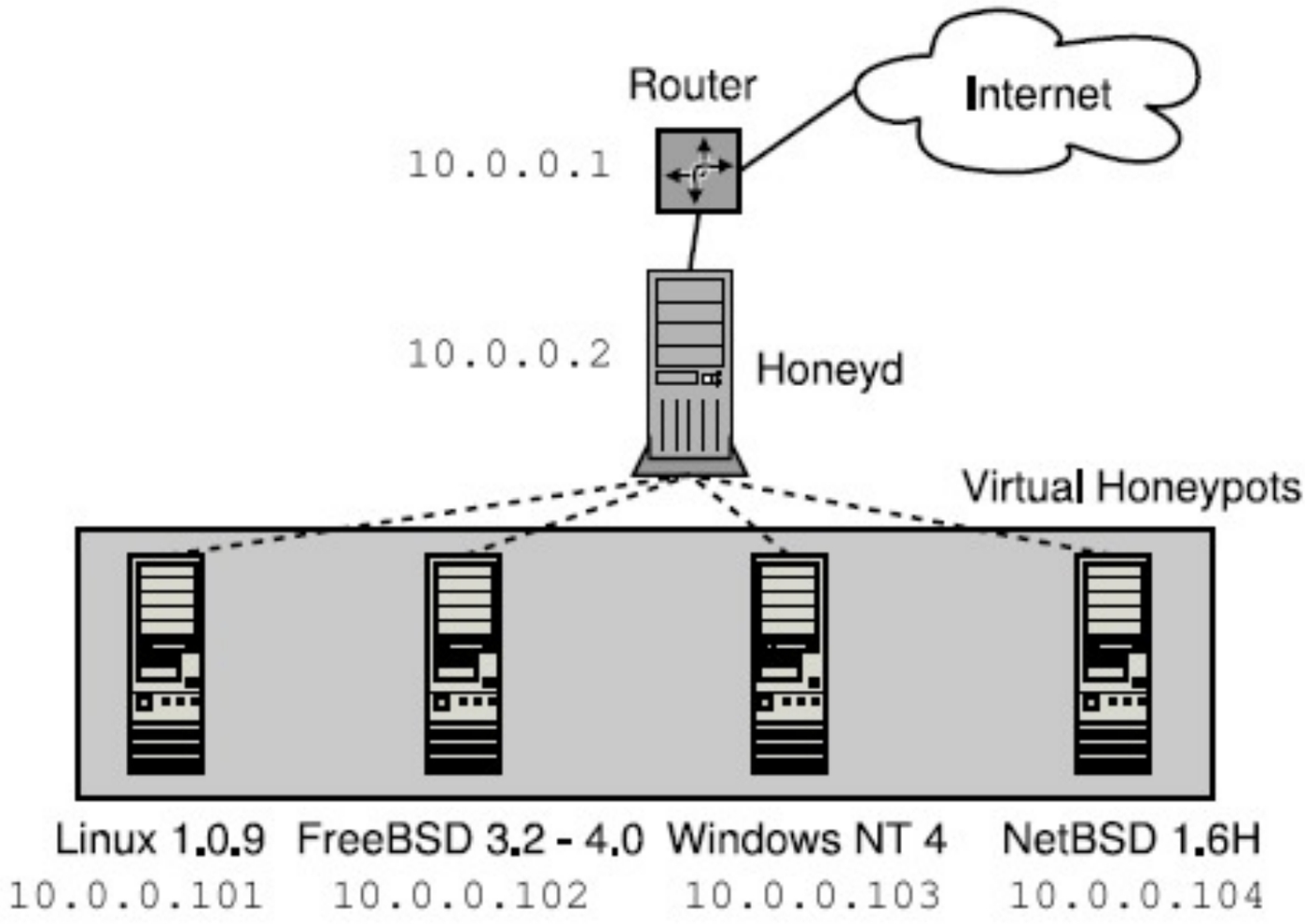
- Considerations
  - Network Architecture
  - Simulating honeypots
    - Simulate only network stack behavior Instead of simulating every aspect of an operating system
    - Simulate arbitrary network topologies
  - Security of the honeyd host
    - Limit adversaries to interacting with honeypots only 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

- Three ways for Honeyd to receives traffic for its virtual honeypots
  - Special route lead data to honeyd host
  - Proxy ARP for honeypots
  - Support Network Tunnels(GRE)

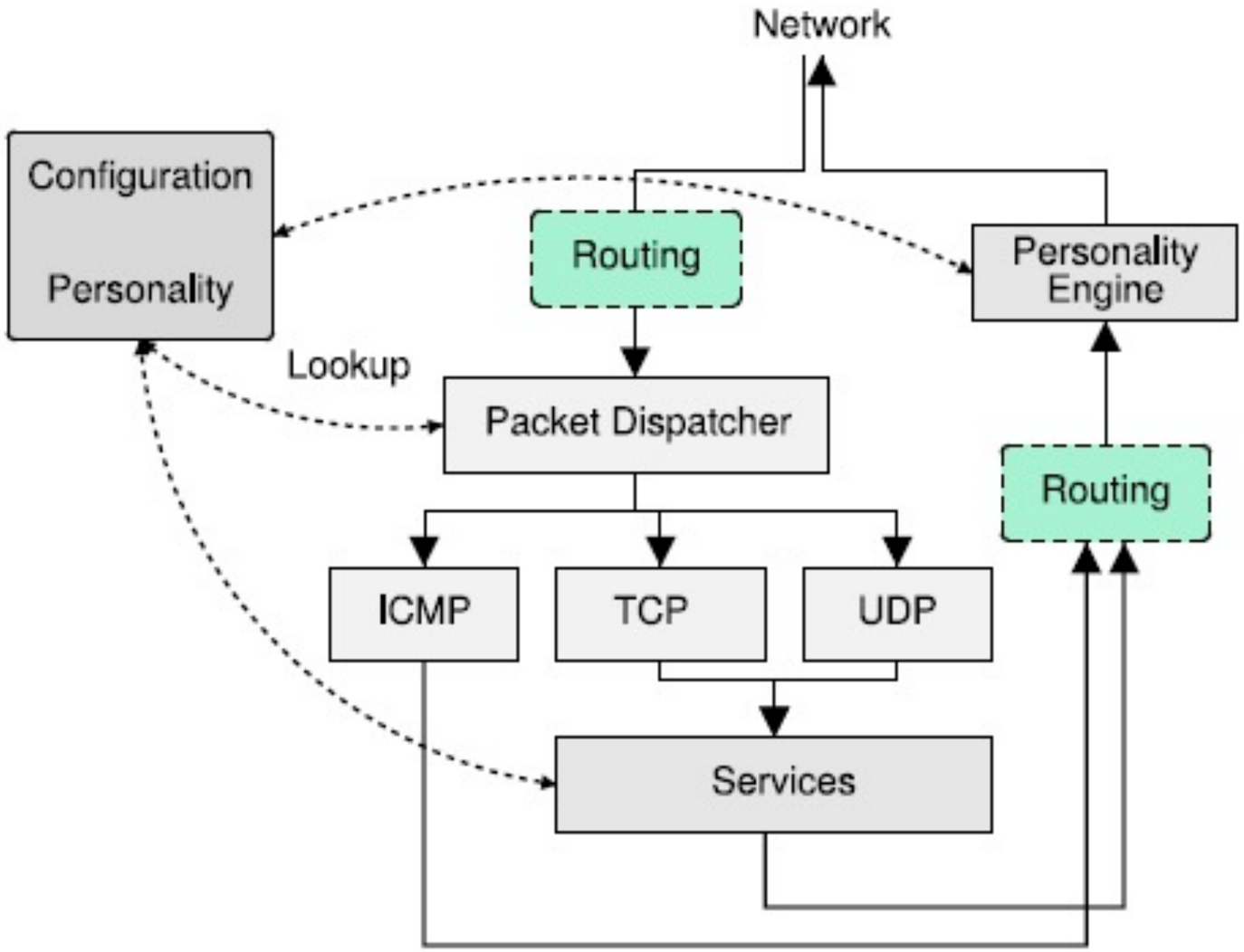


Ex : Arpd

- 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.

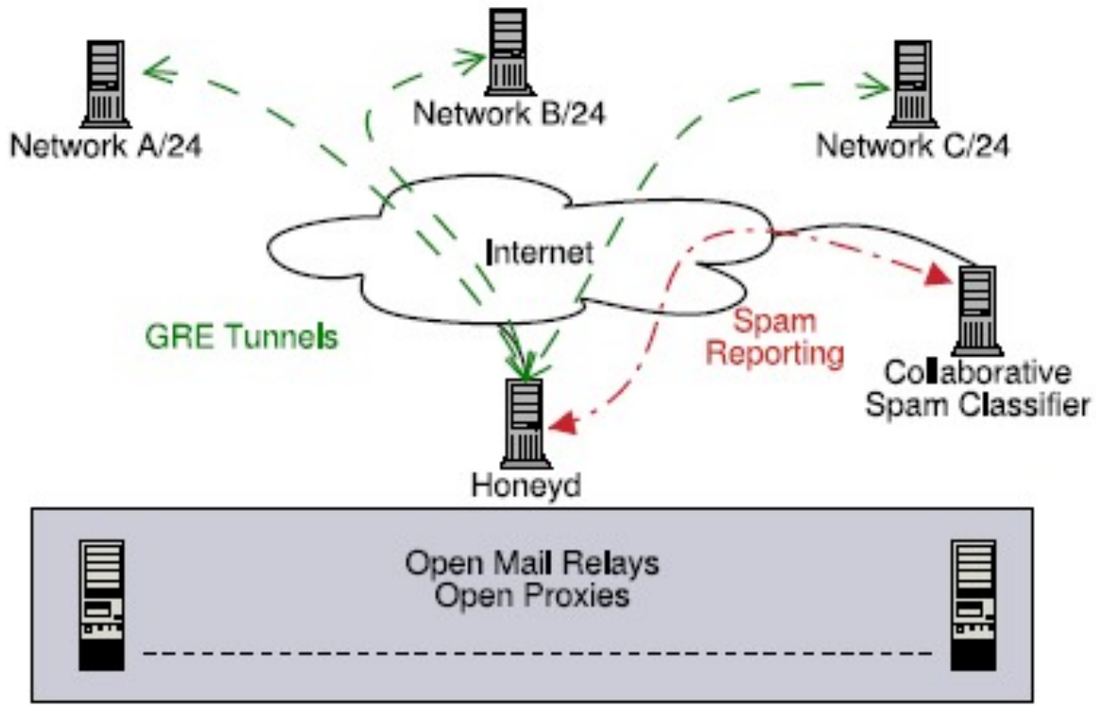
Applications

- 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 open 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

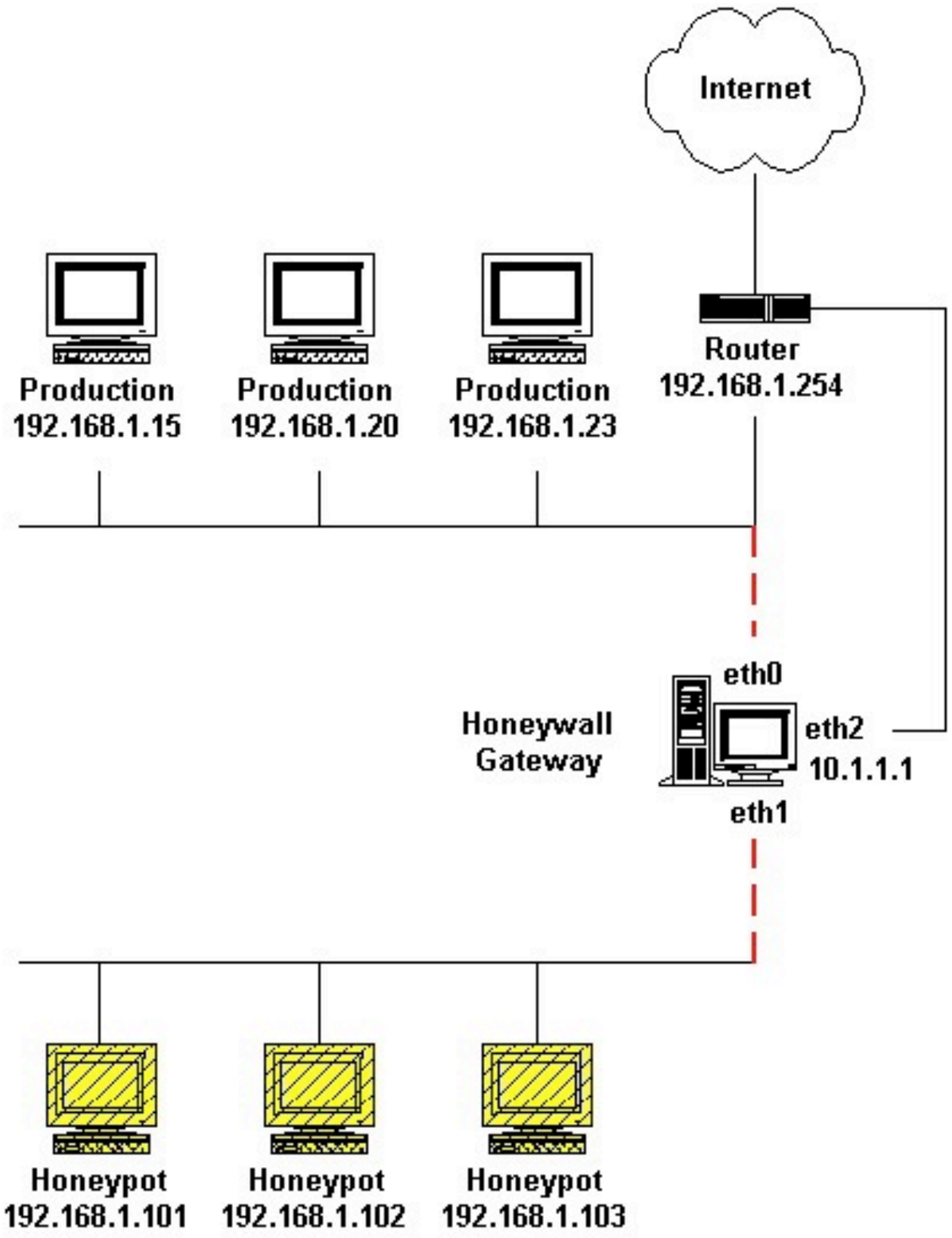
- High-interaction honeypot designed to capture in-depth information.
- Information has different value to different organizations.
- Its an architecture you populate with live systems, not a product or software.
- Any traffic entering or leaving is suspect.

How it works

- A highly controlled network where every packet entering or leaving is monitored, captured, and analyzed.

  - Data Control
  - Data Capture
  - Data Analysis

Honeynet Architecture



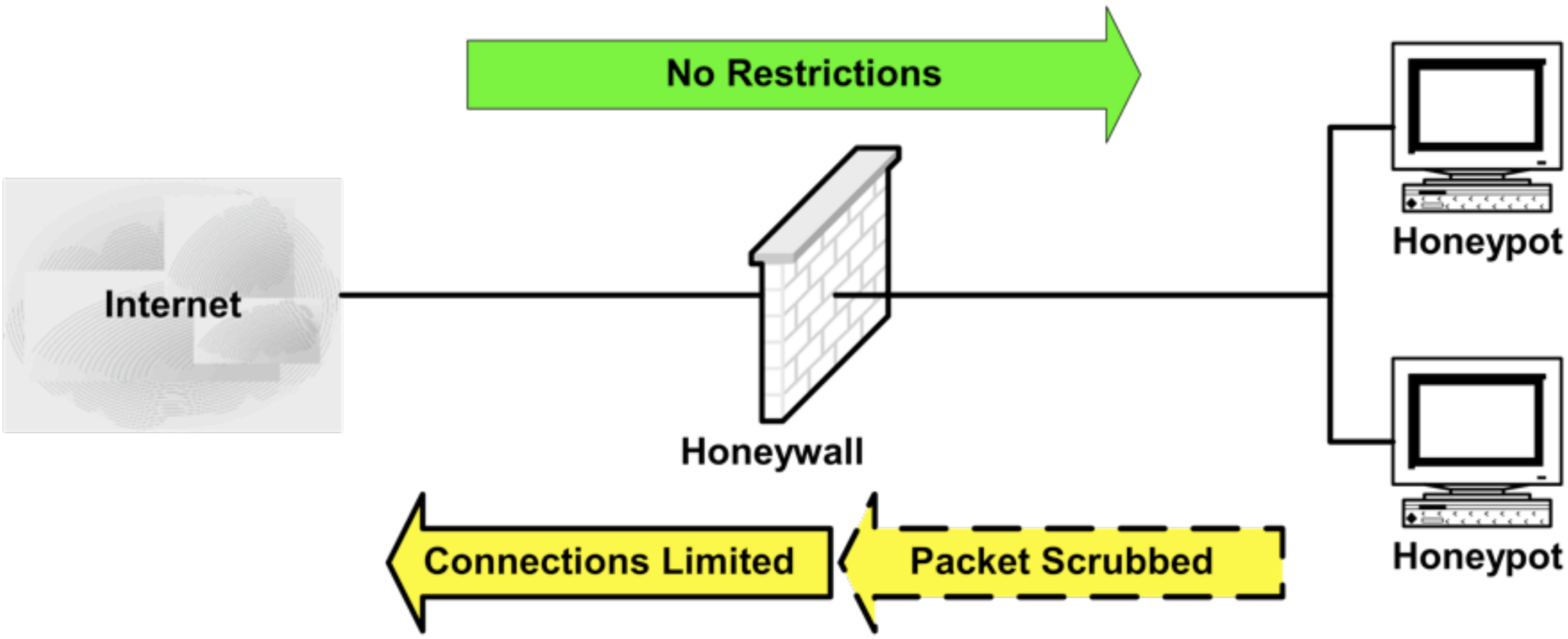
Data Control

- 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

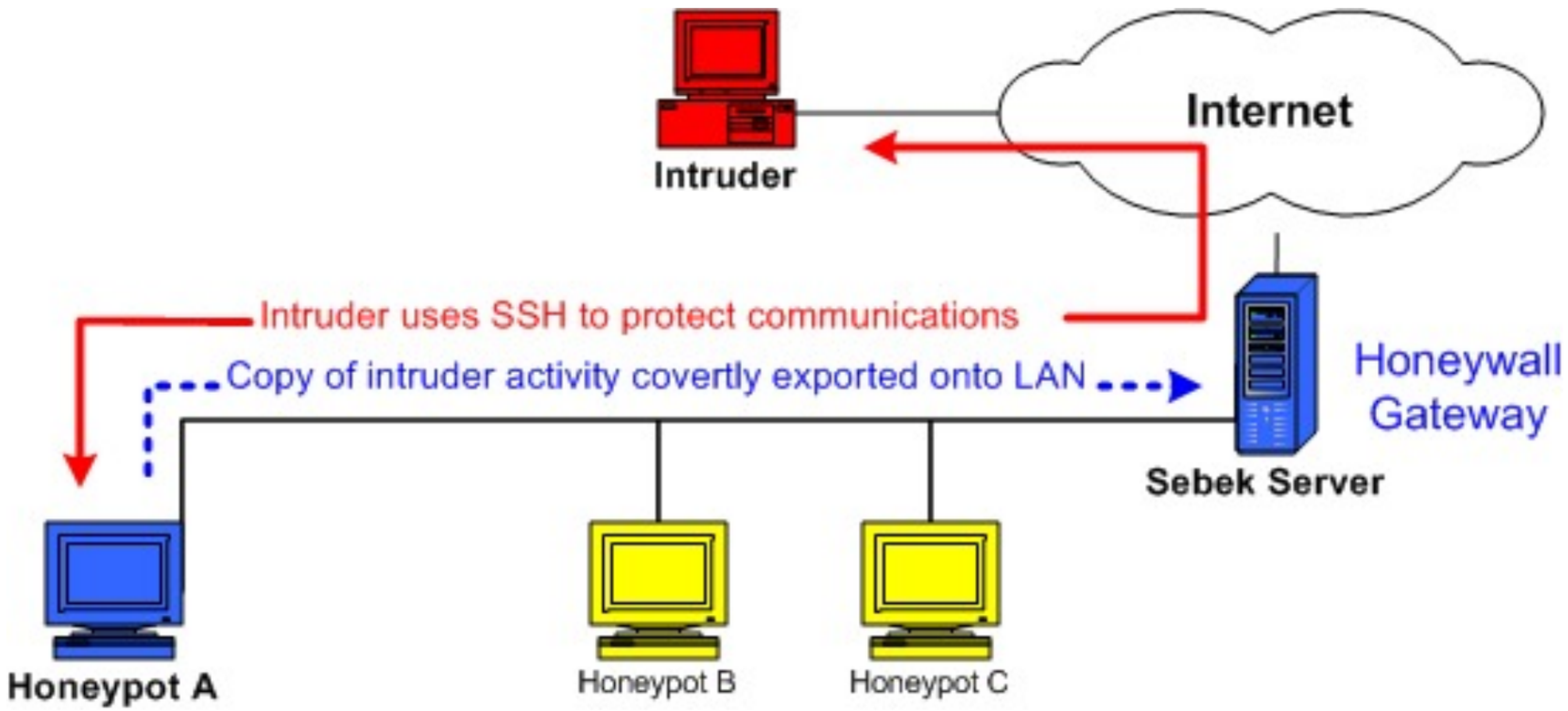
- Capture all activity at a variety of levels.

- 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



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.