Countermeasure Analysis

This step chooses how to change the target system

-statically

-at run time (under attack)

to avoid or at least minimize the risk

Countermeasures

A first classification

- Proactive
  - They are applied before an intrusion

    eg a vulnerability is removed
- Dynamic
  - They are applied as soon as an attack is detected

    eg a vulnerability is removed

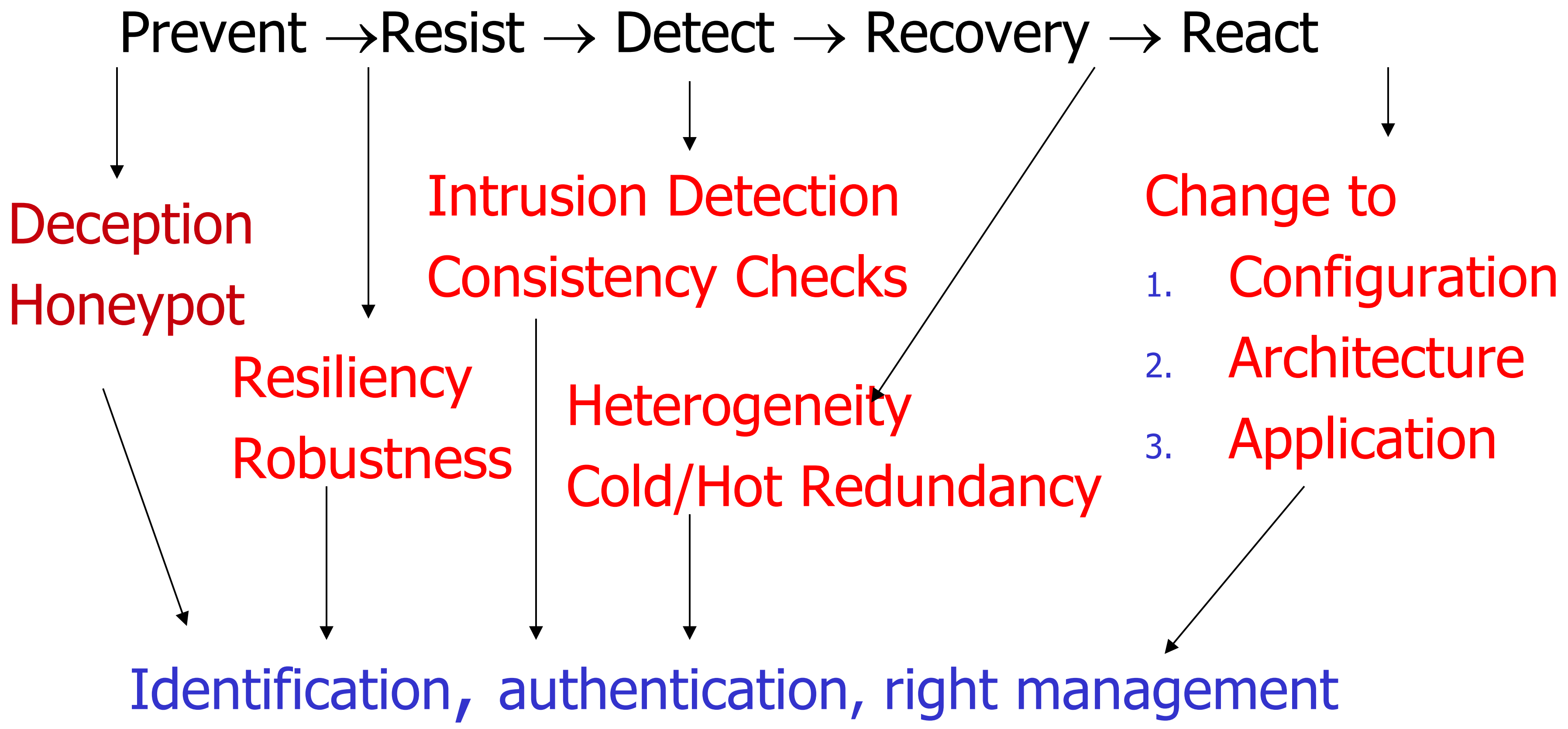
    eg a connection is killed
- Reactive
  - They are applied after a successful attack

    eg a vulnerability is removed

    eg a password is changed

Detection?

A more detailed taxonomy



Implementation mechanisms

- Countermeasures are implemented through a set of common mechanisms
- A set of shared mechanisms
  - It can increase the cost effectiveness of countermeaures
  - It should be highly robust because a vuln may affect several countermeasures

Base mechanisms

- The mechanisms are defined on top of a security kernel (= TCB) that manages
  - The user identities
  - User authentication (identity checks)
  - User rights
- The TCB should not be confused with the minimal system that is discussed in the following

Countermeasures Glossary- I

- Deception = no information about the system design is available = S&S, no open design + honeypot
- Honeypot = fake systems to
  - increase the complexity of discovering target nodes
  - detect attack
- Resiliency/Robustness = prevent a single vulnerability from enabling a successful intrusion (S&S, least privilege etc)
- Intrusion Detection/ Consistency Check = checks to discover the current or previous attacks

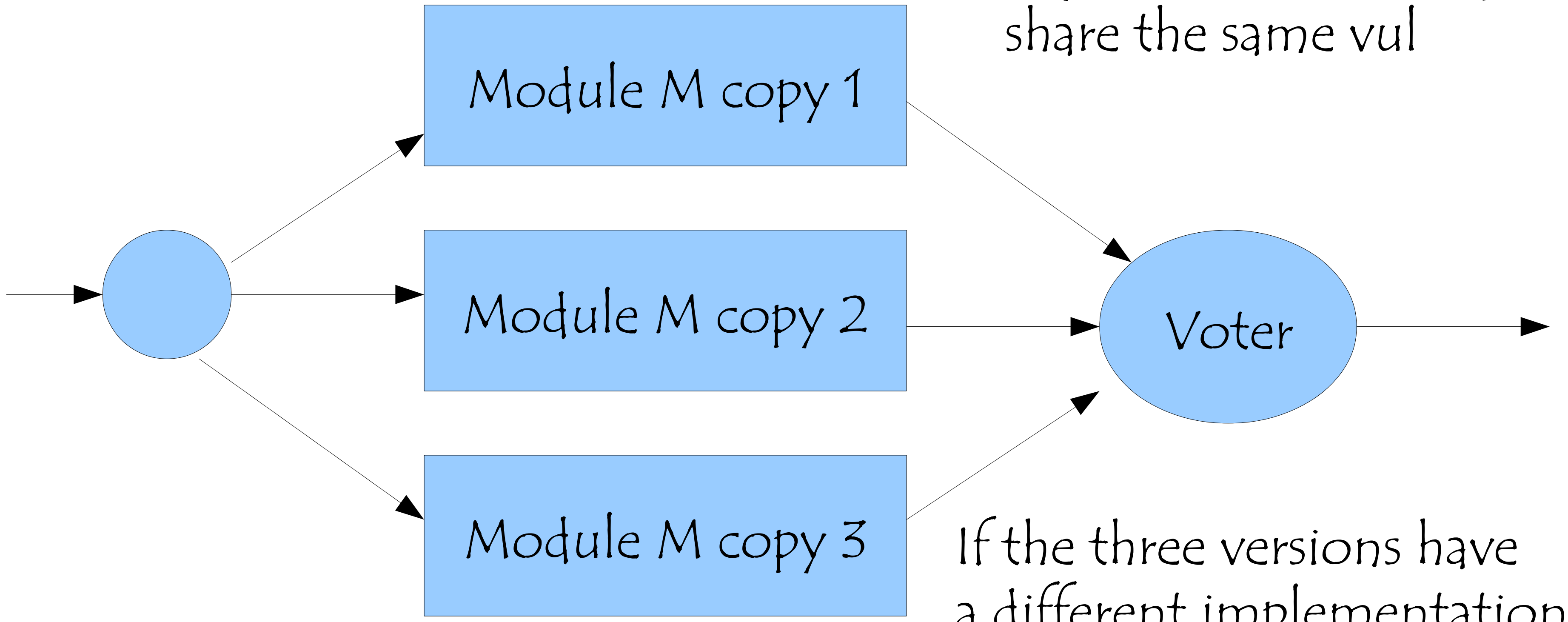
Countermeasures Glossary - II

- Redundancy = spare components to replace the attacked ones. The impact is reduced and control on the system is not lost
  - Cold = Stand by spare components
  - Hot = Spare components are in use (oversize system)

  The underlying problem is a proper evaluation of expected performance
- Heterogeneous components = genetic diversity = the vulns of spare components differs from those of standard components
- A generalization of triple modular redundancy

Triple Modular Redundancy

Safety, not security
anytime the three copies
share the same vul



If the three versions have
a different implementation
some security is achieved

Countermeasures Glossary- III

- Minimal system
  - A subset of components
  - More robust
  - Large number of severe checks
- Control of the minimal system should never be lost
- It is a starting point to gain back control on the whole system
- Strongly related to normal vs power law impact we have discussed

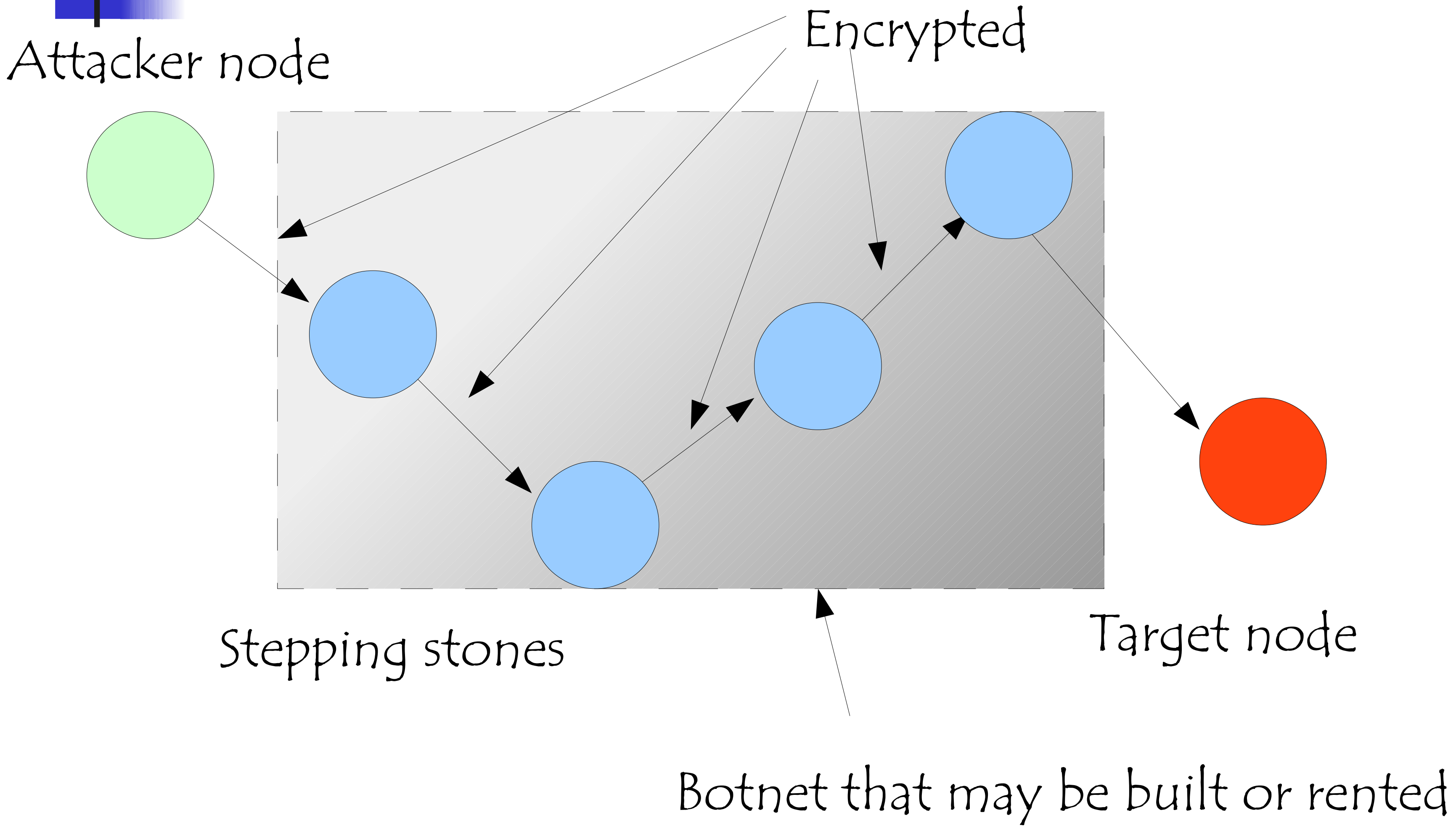
Countermeasures Glossary- IV

- Reaction = Updates to
  - The configuration of the OS and applications
  - System architecture
  - Enabled application
  - Patch
- The reaction usually updates the target system and it not involves the attacking one
- Offensive security = react by attacking the attacking system = Huge set of problems

Attacking the attacking sys?

- Attack attribution, remember the difference between a missile and a worm?
- Stepping stone = a chain of hosts that starts at the one of the attacker and that are, illegally, controlled by the attacker =botnet + com&contr
- The attacker uses the chain to hide his/her node
- The last node in the chain implements the attack to hide the first one
- Any node connected to the internet has a value as it can be used as a stepping stone =hygene
- How can we discover a stepping stone?

Stepping stones = botnet



Stepping stone detection - 1

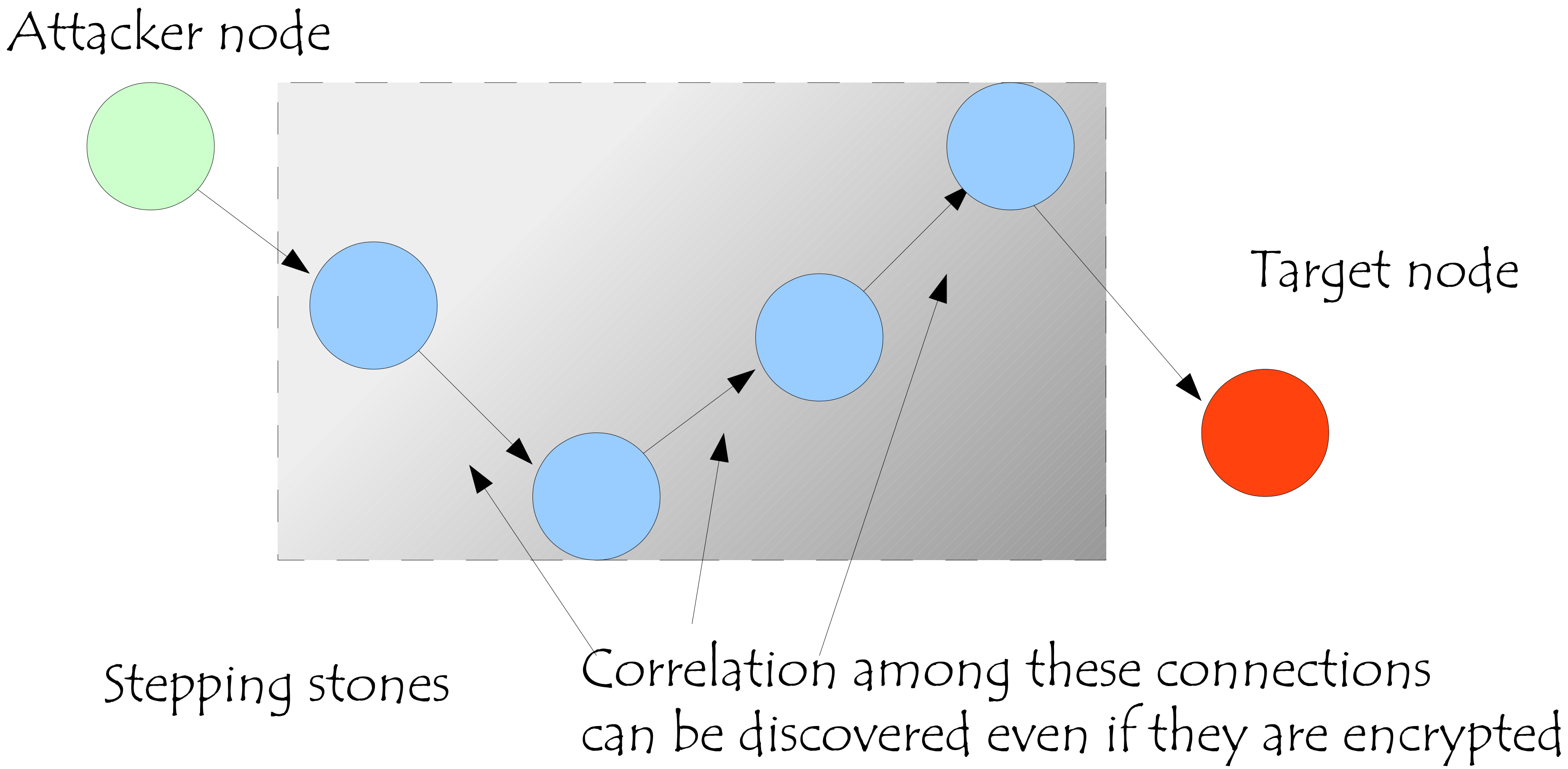
- An analysis of input/output node channel to evaluate their correlation
- If there are an input channel and an output one (i/o port) that are correlated as far as concerns
  - Time = when a communication occurs
  - Data = size of exchanged data

  then the node may act as a stepping stone
- By repeating the analysis for the sender/receiver of the two channeld, the whole chain of stepping stones (= the whole botnet) may be discovered

Stepping stone detection- 2

- The proposed analysis is a traffic analysis that can be applied even to encrypted flows because it does not consider the information content of the two flows
- No serious attacker uses stepping stone chains that communicate in clear

Stepping stone


Attacker node
Target node
Stepping stones
Correlation among these connections can be discovered even if they are encrypted

Deception = Honeypot

- Both diffusion and adoption has increased because of virtualization technologies that minimizes its cost
- It increases the complexity of attacks that use a vulnerability scanner to discover nodes in a network that can be attacked
- For each address the scanner generates, the defender creates a new fake virtual node the attacker has to analyze
- These virtual nodes are useless but as far as the scanning is concerned, they behave like real nodes
- The fake nodes
    - reply to the fingerprinting messages with frequency that becomes slower and slower to slow down the scanning
    - raise an alarm

Honeypot - Definition

An ICT resource whose value lies in unauthorized or illicit use of that resource.

- Has no production value; anything going to/from a honeypot is likely a probe, attack or compromise
- Used for monitoring, detecting and analyzing attacks
- Does not solve a specific problem. Instead, they are a highly flexible tool with different applications to security.

Classification

- By level of interaction
  - High
  - Low
  - Middle
- By Implementation
  - Virtual
  - Physical
- By purpose
  - Production
  - Research

Level of Interaction

- Low interaction—A simple port listener is considered extremely low interaction because, after the connection, the attacker cannot do anything else.
- Medium interaction—An emulated service that analyzes communications and returns simulated responses to replicate a real service
- High interaction—This involves the use of real, but deceptive services, fully operational hosts or complete deceptive networks.

Level of Interaction

- As the level of interaction increases, the attacker ability to "play" with the resources also goes up.
- Higher interaction gives the attacker a more realistic experience and also provides significantly opportunities to analyze attacker activity.
- A better understanding of attacker activity allow security teams
  - to respond more effectively,
  - to enhance their ability to design improved deception scenarios.

Physical vs Virtual Honeypots

- Two types
  - Physical
    - Real machines
    - Own IP Addresses
    - Often highly-interactive
  - Virtual
    - Simulated by other machines that:
      - respond to the traffic sent to the honeypots
      - may simulate distinct virtual honeypots at the same time

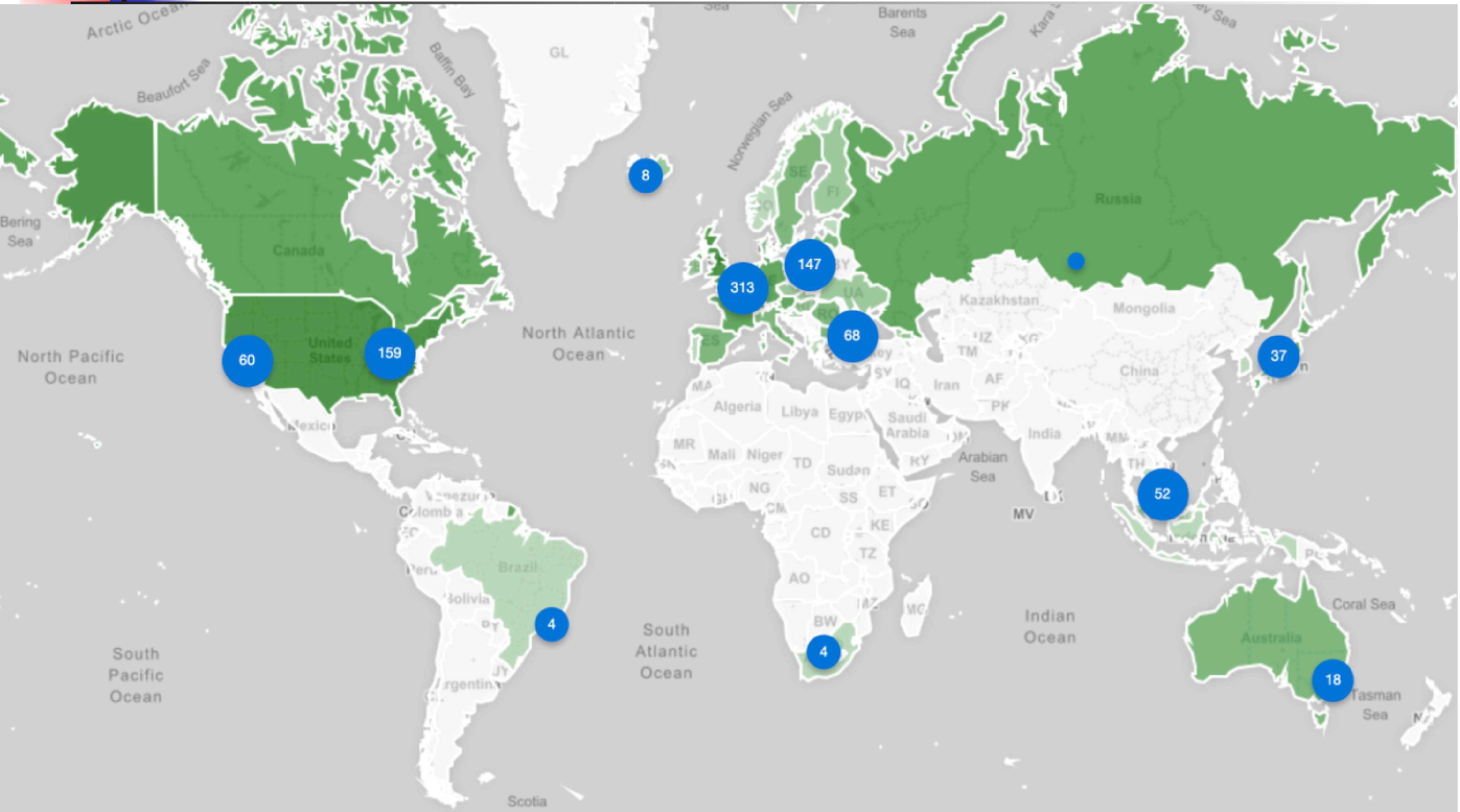
Production HPs: Protect the systems

- Prevention
  - Keeping the bad guys out
  - not effective prevention mechanisms.
  - Deception, Deterence, Decoys do NOT work against untargeted attacks: worms, auto-rooters, mass-rooters
- Detection
  - Detecting the burglar when he breaks in.
  - Great work
- Response
  - Can easily be pulled offline
  - Little or no data pollution

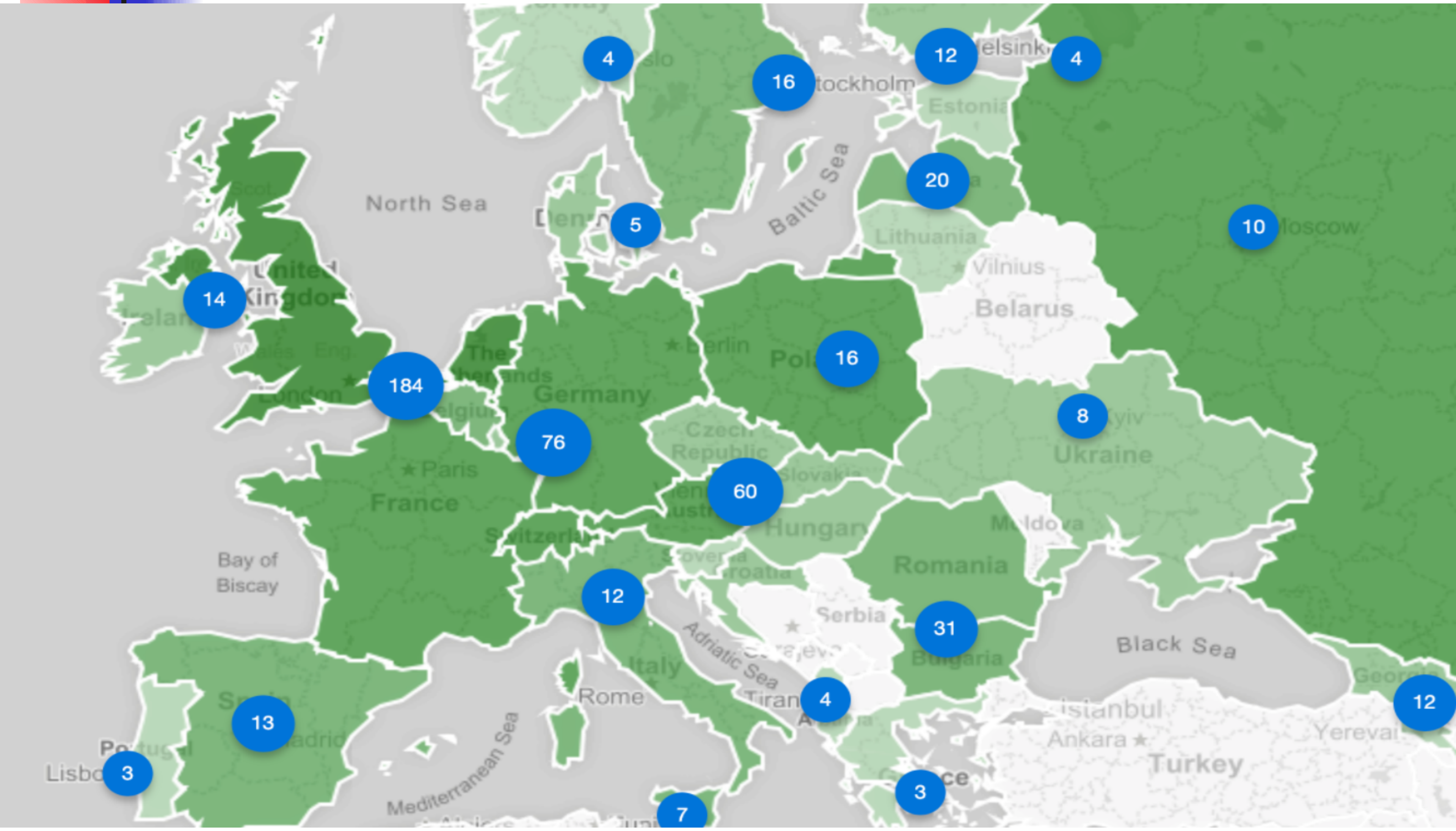
Research HPs: gathering information

- Threat Intelligence
- Collect compact amounts of high value information
- Discover new ttps and tools
- Understand Motives, Behavior, and Organization
- Develop Analysis and Forensic Skills
- Discover new worms/viruses and signatures

Sensor network for threat intelligence



Sensor network for threat intelligence



A sensor

- 9 different honeypot types
- Each focused on observing distinct attacks against SSH/telnet services, web services, remote management protocols, databases, mail relays, ICS devices, including exploits, scans, brute force attacks.
- Each sensor is a VM with at least:
  - 1 core
  - 512MB RAM
  - 5 GB hard drive
  - 2 or more static IPv4 addresses

Building your HoneyPots

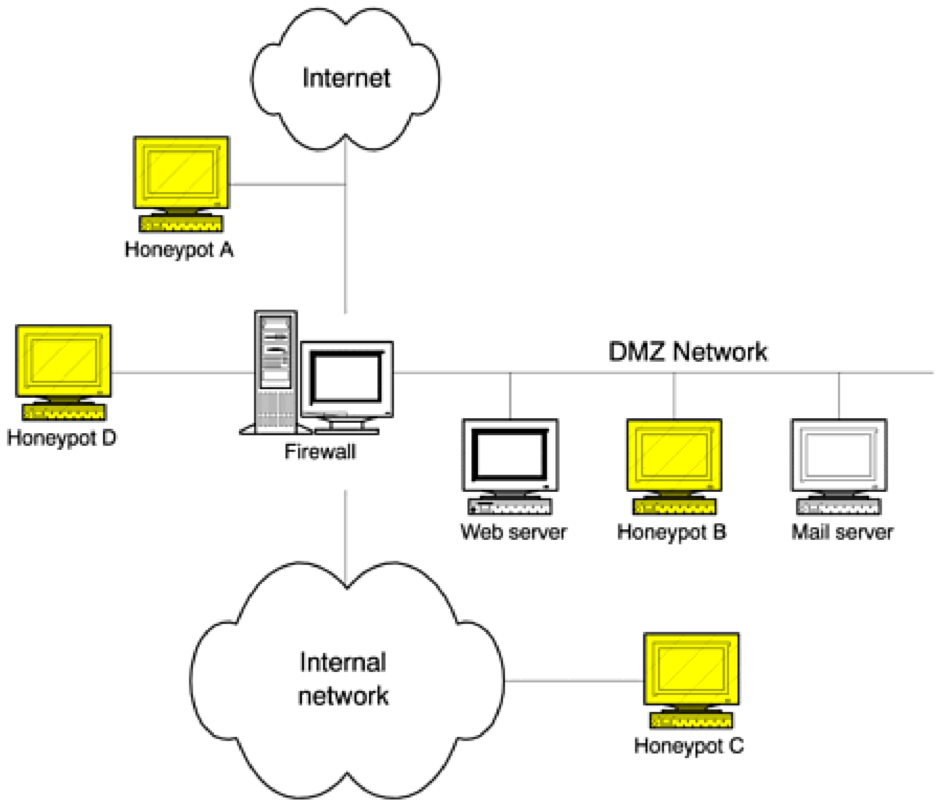
- Specifying Goals
- Selecting the implementation strategies
    - Types, Number, Locations and Deployment
- Implementing data capture
- Logging and managing data
- Mitigating risk
- Mitigating fingerprint

Just an anticipation …

- Firewall
  - A system that connects two networks with distinct security requirements
  - It filters the information flowing across the two networks and the services each network can access in the other one
  - It hides some components in the most critical networks so that they cannot be accessed from the less critical network
  - It defends the most critical network from attacks originating in the less critical and less protected one at the expence of the bandwidth between the two networks

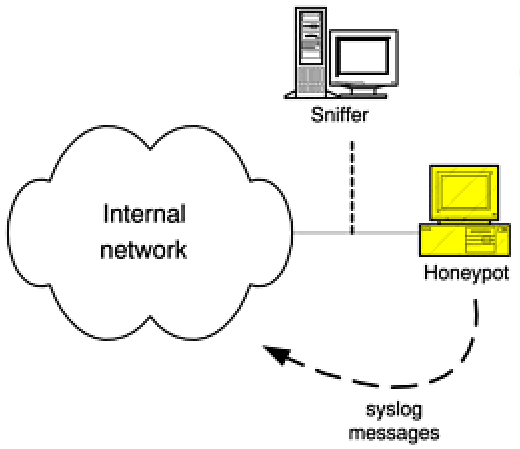
Location of Honeypots

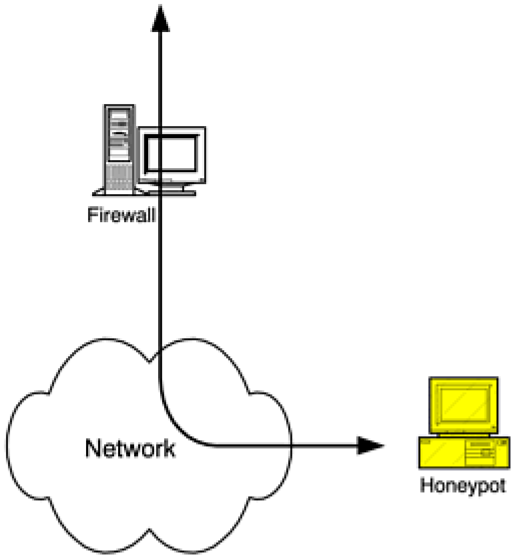
- In front of the fire
- Demilitarized Zone
- Behind the firewall (Intranet)
- Understand the de an attacker can reach



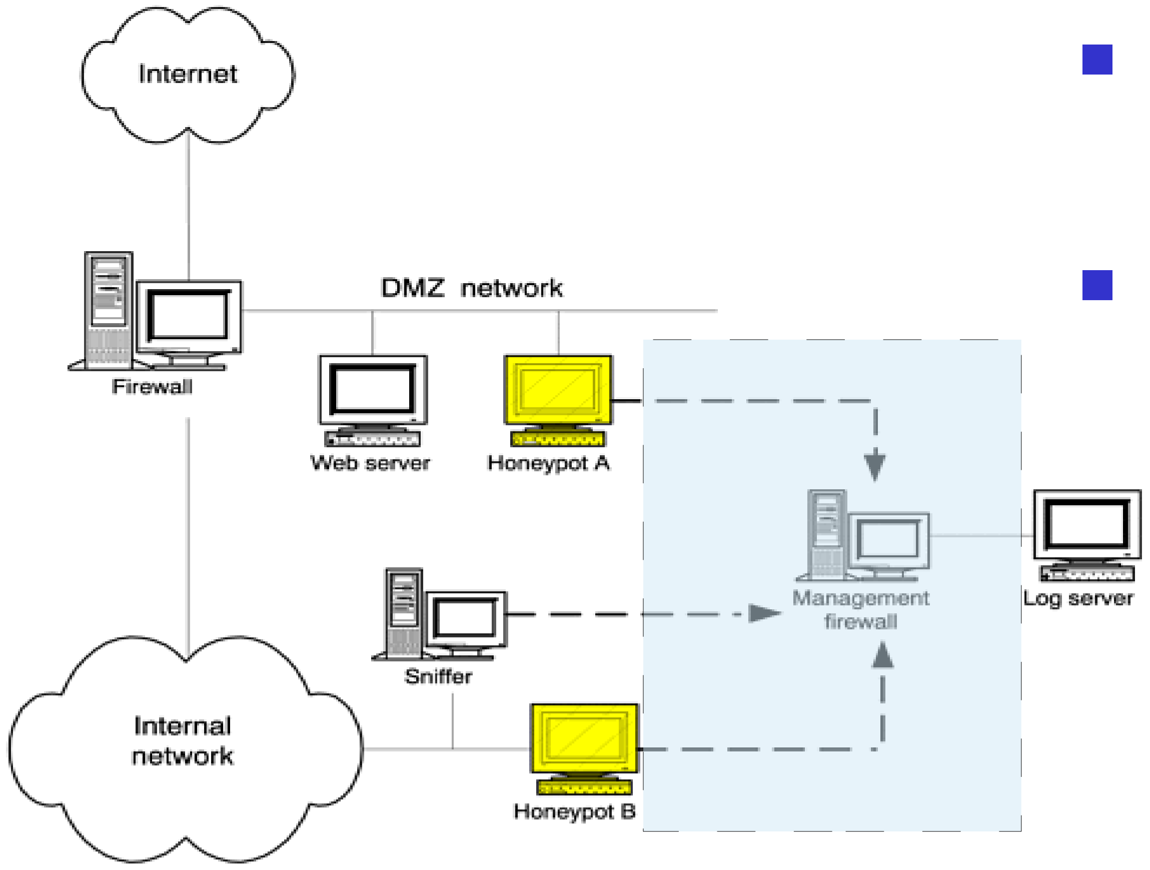
Capturing Information

- Host based:
  - Keystrokes
  - Syslog
- Network based:
  - Firewall
  - Sniffer
  - IP not resolved name





Logging and Managing Data



- Logging architecture
- Managing data

What is Honeyd?

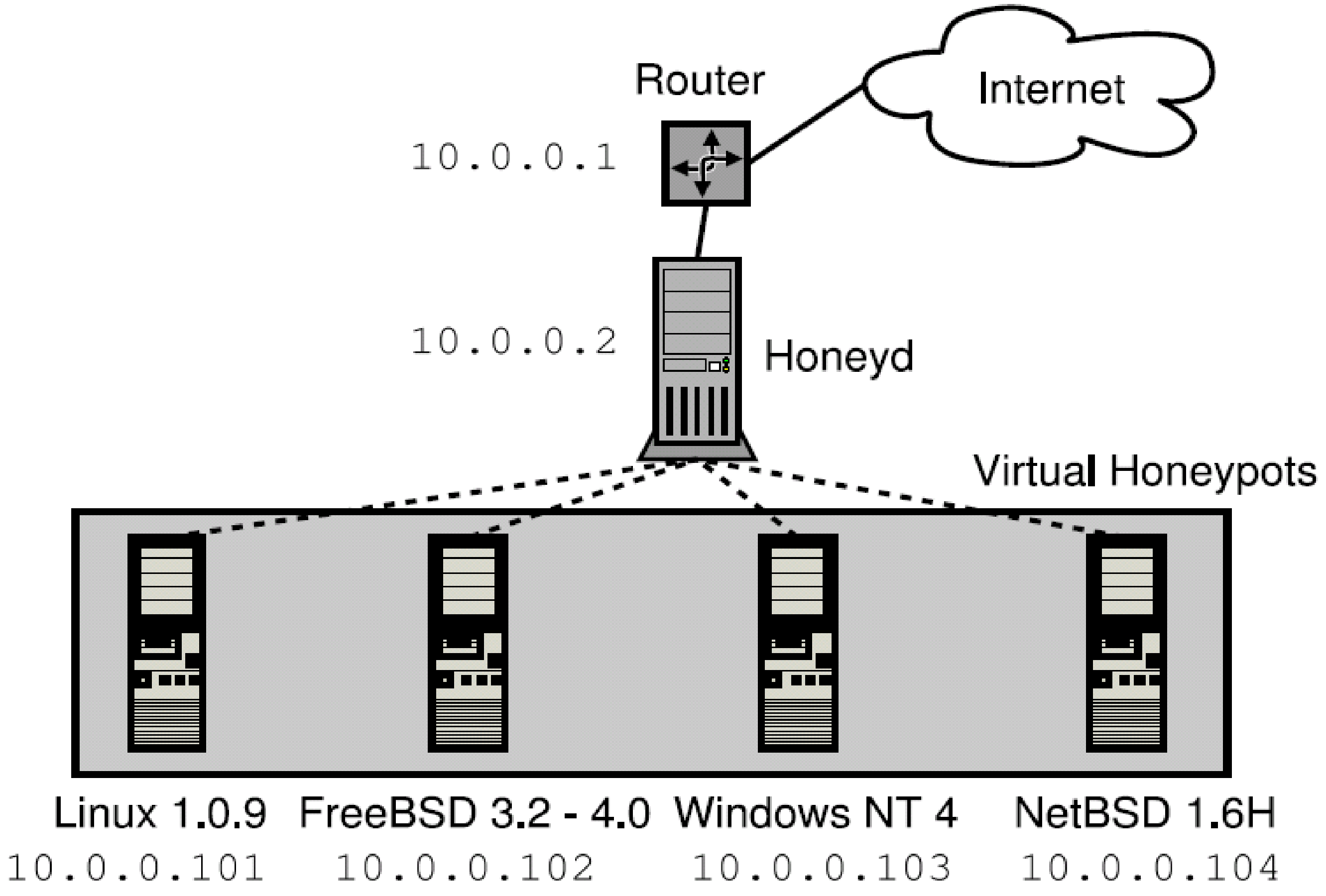
- **Honeyd**: A virtual honeypot application, which allows us to create thousands of IP addresses with virtual machines and corresponding network services.

- Written by Neil Provos available at http://www.honeyd.org/

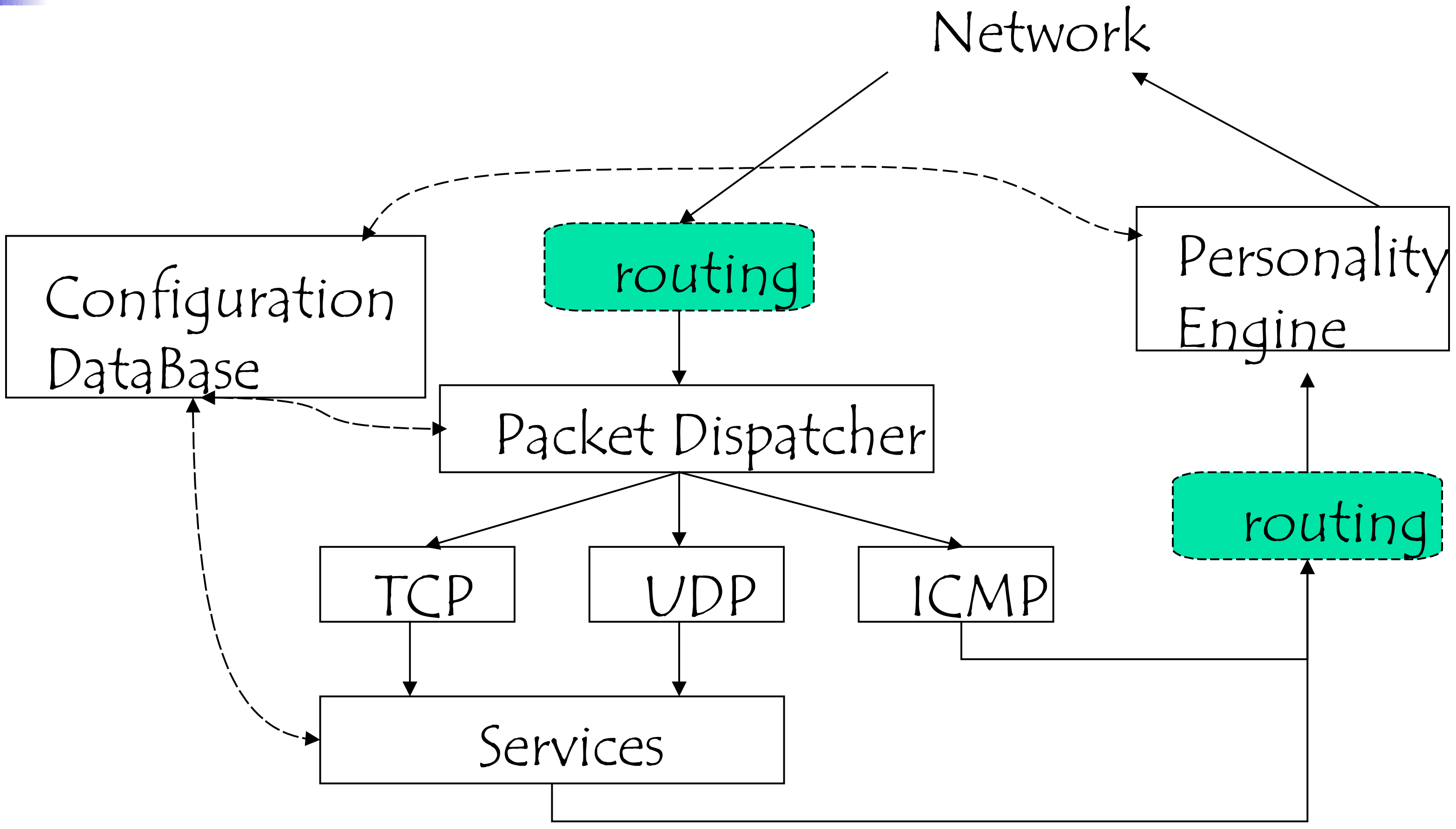
What can honeyd do?

- Simulates operating systems at TCP/IP stack level, supporting TCP/UDP/ICMP;
- Support arbitrary services;
- Simulate arbitrary network topologies;
- Support tunneling and redirecting net traffic;

Illustration Simple



How it works?


Network
Configuration DataBase
routing
Personality Engine
Packet Dispatcher
TCP
UDP
ICMP
routing
Services

Configuration

```
route entry 10.0.0.1
route 10.0.0.1 link 10.0.0.0/24
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
route 10.1.0.1 link 10.1.0.0/24
route 10.2.0.1 link 10.2.0.0/24

create routerone
set routerone personality "Cisco 7206 running IOS 11.1(24)"
set routerone default tcp action reset
add routerone tcp port 23 "scripts/router-telnet.pl"

create netbsd
set netbsd personality "NetBSD 1.5.2 running on a Commodore Amiga (68040 processor)"
set netbsd default tcp action reset
add netbsd tcp port 22 proxy $ipsrc:22
add netbsd tcp port 80 "sh scripts/web.sh"

bind 10.0.0.1 routerone
bind 10.1.0.2 netbsd
```

Why Personality Engine?

- To fool fingerprinting tools
- Uses fingerprint databases by
  - Nmap, for TCP, UDP
  - Xprobe, for ICMP
- It changes to the headers of every outgoing packet before it is sent to the network

Why Routing topology?

- Simulates virtual network topologies;
  - Some honeypots are also configured as routers
  - Latency and loss rate for each edge is configured;
- Support network tunneling and traffic redirection;

Current version

- Can implement passive fingerprinting to discover some features of the remote host that is attacking (the final stepping stone)
- Can run actual OS to better mimic their behavior
- To be run in a sandbox with ptrace

Passive fingerprinting

- This style of fingerprinting does not send any packets, but relies on sniffing to analyze the information sent in normal network traffic.
- If a target is running publicly available services, passive fingerprinting may be a good way to start off fingerprinting.
- It is less accurate than active fingerprinting and it relies on an existing traffic stream
- It can also take much longer depending on the activity level of the target system

p0f—a Passive Fingerprinting Tool

- p0f looks at the following IP and TCP fields:
  - Initial Time To Live – IP header
  - Don't Fragment – IP header
  - Overall SYN packet size – TCP header
  - TCP Options like windows scaling or maximum segment size – TCP header
  - TCP window size –TCP header
  - TCP session startups -the SYN segment.
- The program uses a fingerprint database to identify the hosts that opens a connection

p0f—a Passive Fingerprinting Tool

```
-----------------MacOS-------------------
S2:255:1:48:M*,W0,E:.:MacOS:8.6 classic
16616:255:1:48:M*,W0,E:.:MacOS:7.3-8.6 (OTTCP)
16616:255:1:48:M*,N,N,N,E:.:MacOS:8.1-8.6 (OTTCP)
32768:255:1:48:M*,W0,N:.:MacOS:9.0-9.2
32768:255:1:48:M1380,N,N,N,N:.:MacOS:9.1 (1) (OT 2.7.4)
65535:255:1:48:M*,N,N,N,N:.:MacOS:9.1 (2) (OT 2.7.4)
```

- 9.0-9.2 the initial window size is 32768 bytes, the initial time to live is 255, the don't fragment bit is on, the total length of the SYN packet is 48 bytes, the maximum segment size option is bolted on —as is the window scaling option, there is a no-operation (NOP) in the option list

What is a Honeynet <> Honeypot

- A network with nodes and honeypots to design a high-interaction honeypot able to:
  - capture in-depth *information*
  - learn who would like to use your system without your permission for their own ends
- Its an architecture, not a product or software.
- Populate with live systems.
- Can look like an actual production system

What is a Honeynet

- Once nodes are compromised, data is collected to learn the tools, tactics, and motives of the blackhat community.

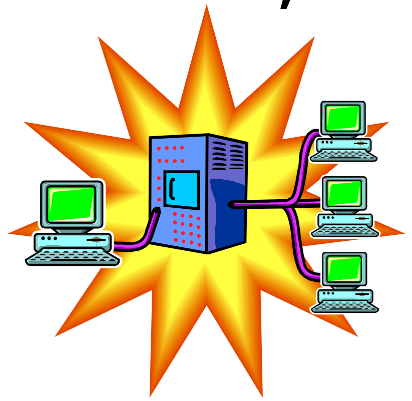
- Information has different value to different organizations.
  - Learn vulnerabilities
  - Develop response plans

What’s The Difference?

- Honeypots use known vulnerabilities to lure attack.
  - Configure a single system with special software or system emulations
  - Want to find out actively who is attacking the system
- Honeynets are networks open to attack
  - often use default installations of system software
  - behind a firewall
  - hope attackers mess up the honeynet instead than your production system

How a honeynet works

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



- Any traffic entering or leaving the Honeynet is suspect by nature.

Countermeasures - Deception

- Cryptography algorithms
- Information is coded so that only who knows a further info, the key, can access it
- Already known

Just a reminder …

- Cryptography does not solve the problems, it only simplify the solution
- It is very difficult to safely store a 2 gb file
- It is simpler to encrypt the file through a 256 bit key and safely store the key
- The same problem has to be solved (safely store an info) but now the solution is simpler because the problem size has been reduced

Just a reminder …

- Hiding and protecting
  - Information at rest
  - Exchanged information
- Integrity (hash function)
- Authentication (digital signature)
  - Hash + Encrypt with private key
- Coprocessor (smartcard)
  - Hardware root of trust
- Symmetric and Asymmetric

Resist – Robust (proactive) programming

- Validate program inputs aka input is evil
- Prevent buffer overflow
- Robust implementation
- Check the invocations to other resources
- Check returned results

Robust programming – Input validation

Input validation + default deny (S&S)

- Define the input legal structure
- Check that any input satisfy the defined structure

Example: Strings

- A grammar that defines the structure
- Longest input string
- Define which special characters are legal
- Check that any input satisfies 1-2-3

Robust programming – Input validation

- The checks to validate the input should be specified together with the program rather than after an attack
- In the correct approach, the specification may simplify the design and the implementation of the checks through a simple grammar, eg LR grammar, ie controls implemented by finite state automaton
- A complex control may be useless if we are not confident that it has been correctly implemented
- Several languages offer built in functions to check a string against a regular expression or to filter out dangerous characters

Robust programming – Input validation

- Parameters to be validated
  - Environment variables
  - File names (blanks , .., /,)
  - Email addresses
  - URL
  - Html
  - data
- Use built in function to match a string against a predefined pattern, remove dangerous characters, extract substring with the desidered length

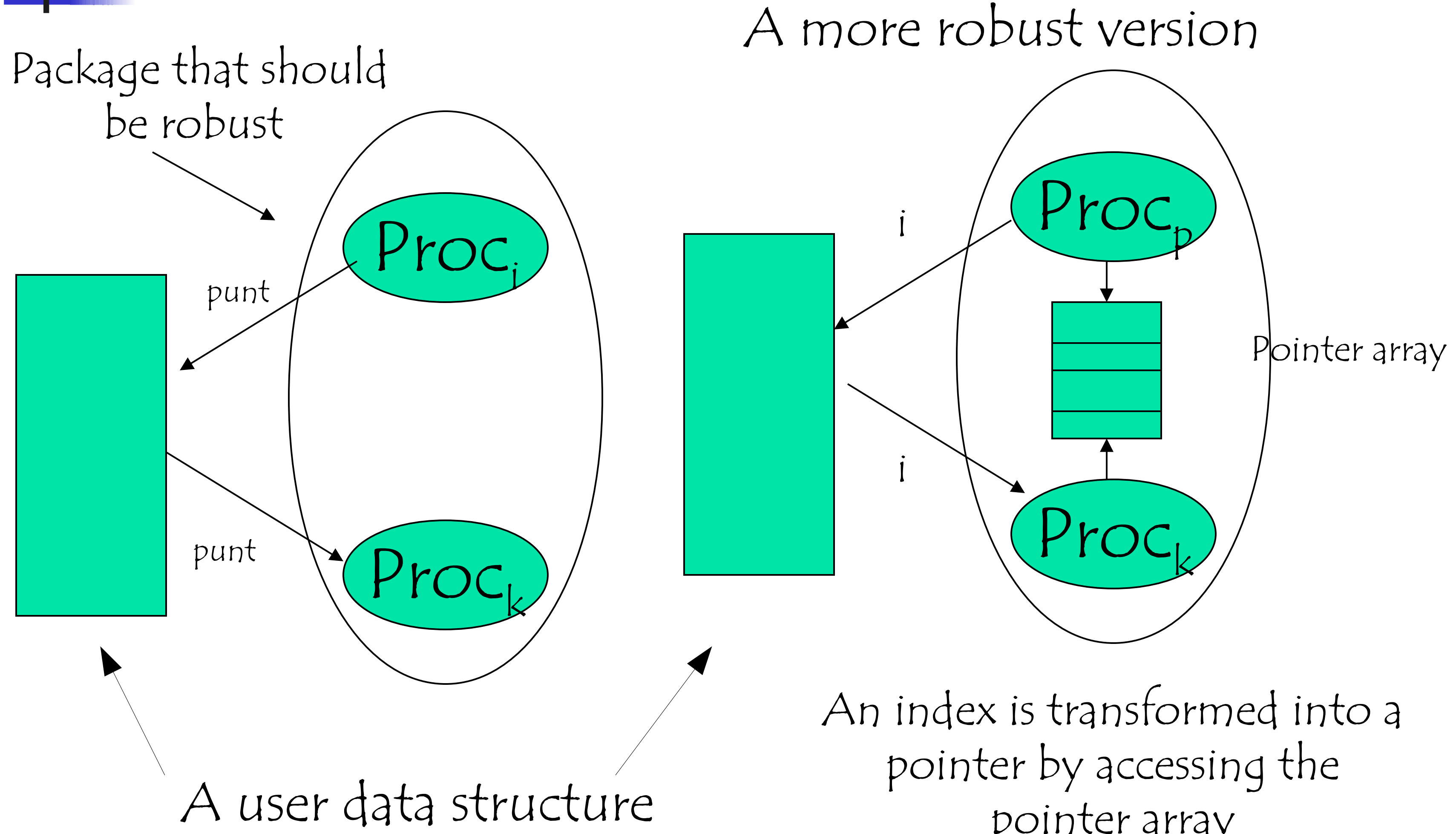
Robust programming – no buffer overflow

- Do not use any library function that does not check it input parameters
- Use only those functions that check the length of their input strings
- Dynamic memory allocation of a data structure according to its size rather than static allocation of the largest amount of memory the structure may require in some execution

Robust programming – robust implementation - I

- Satisfy S&S
- Rigorous definition of the program interface
- Do not assume that input/output values are related
  - If a function of a library returns a pointer and another function of the same library has a pointer parameter, there is no reason to assume that the second will receive only the pointers the first function returns
  - If an input parameter of a function should be the one another function returns, the parameter type has to be defined so that this relation can be checked
- Data and instruction should be different
- The data that each function needs to access should be minimized

Pointer - I



An index is transformed into a pointer by accessing the pointer array

Pointers - II

- By replacing an array of pointers with an array of records where one field is a pointer we can
  - Introduce fields in the records to discover whether each element is properly initialized
  - Check access to the array
  - Define proper checks on the input output relation of a pointer
- This is a simplified, redundant version of an access control matrix for the pointers
- Built in in some programming languages

Pointers - III

- We can also return an encrypted index to one position of the array of pointers rather than the real one

  realpositioin= m*returnedpos+cost

- It simplifies the detection of pointer manipulation
- Access control does not change