**Politecnico di Milano**
**Dip. Elettronica e Informazione**
**Milano, Italy**


# Intrusion Detection Systems, Honeynets

A basic introduction to the concepts of detection and reaction

# A troublesome world

- Recap of unique challenges in security:
  - Vulnerabilities are discovered on a daily basis
  - A firewall cannot control everything
  - Access control and privilege management does not scale to large networks
  - Unauthenticated users access unsafe services
  - Trusted but untrustworthy machines are exposed to the network perils
  - TCP/IP packets are unauthenticated and unencrypted, allowing all sorts of manipulation
  - ...

# Crouching tiger, hidden dragon

- Crackers **do not** use brutal attacks against our defenses, they rely on skill, subtlety, and treachery
- The classic paradigm of security of authentication and authorization, “Who are you? What can you do?”, makes it necessary to build layers upon layers of complex defense systems
- Remember an historical lesson: King Dario at Gaugamela plains against Alexander Magnus
- For computer scientists it’s the KISS principle: Keep It Simple, Stupid
- We need a complementary paradigm

# A paradigm of resistance, detection and reaction

- Murphy's law is unescapable
  - Trusted components and people are untrustworthy
  - Secure programs are unsecure
  - Reliable systems do fail
- The only difference between systems that can fail and systems that cannot is that, when a system which cannot fail, fails, it fails in a totally devastating way
- We must prepare for the fact that security incidents will – not can – happen
  - Build systems that fail gracefully
  - Build systems that rely on others as little as possible
  - Build systems that are tamper-evident
  - Design system for recovery

# Detecting intrusions: why are you doing this ?

- We want to be able to *detect* security incidents, in order to *react* as soon as possible. How can we do this?
- Back to basics: the goal of information security is to ensure that a system is used for a specific set of *goals* following a proper *procedure*, and that it cannot be used otherwise
- A violation of the security paradigm is evident, because the system does something else than what it's intended to do
- From the "Who are you ? What can you do ?" mindset to the "Why are you doing this?" mindset
- INTRUSION DETECTION SYSTEMS

# The IDS mantras

- An IDS is a system, not a software
  - A skilled human looking at logs is an IDS
  - A skilled network admin looking at TCPdump is an IDS
  - A company maintaining and monitoring your firewall is an IDS
  - A box bought by a vendor and plugged into the network is **not** an IDS by itself
- An IDS is not a panacea, it's a component
  - Does not substitute a firewall (despite what Gartner thinks)
  - It's the last component to add to a security architecture, not the first
- Detection without reaction is a no-no
  - Like burglar alarms with no guards!
- Reaction without human supervision is a dream
  - "Network, defend thyself !"

# Terminology and taxonomies

- Different types of software involved in IDS
  - Logging and auditing systems
  - Correlation systems
  - So-called “IDS” software
  - Honeypots / honeytokens
- The logic behind an IDS is always the same: those who access a system for illegal purposes act differently than normal users
- Two main detection methods:
  - Anomaly Detection: we try to describe what is normal, and flag as anomalous anything else
  - Misuse Detection: we try to describe the attacks, and flag them directly

# Anomaly vs. misuse

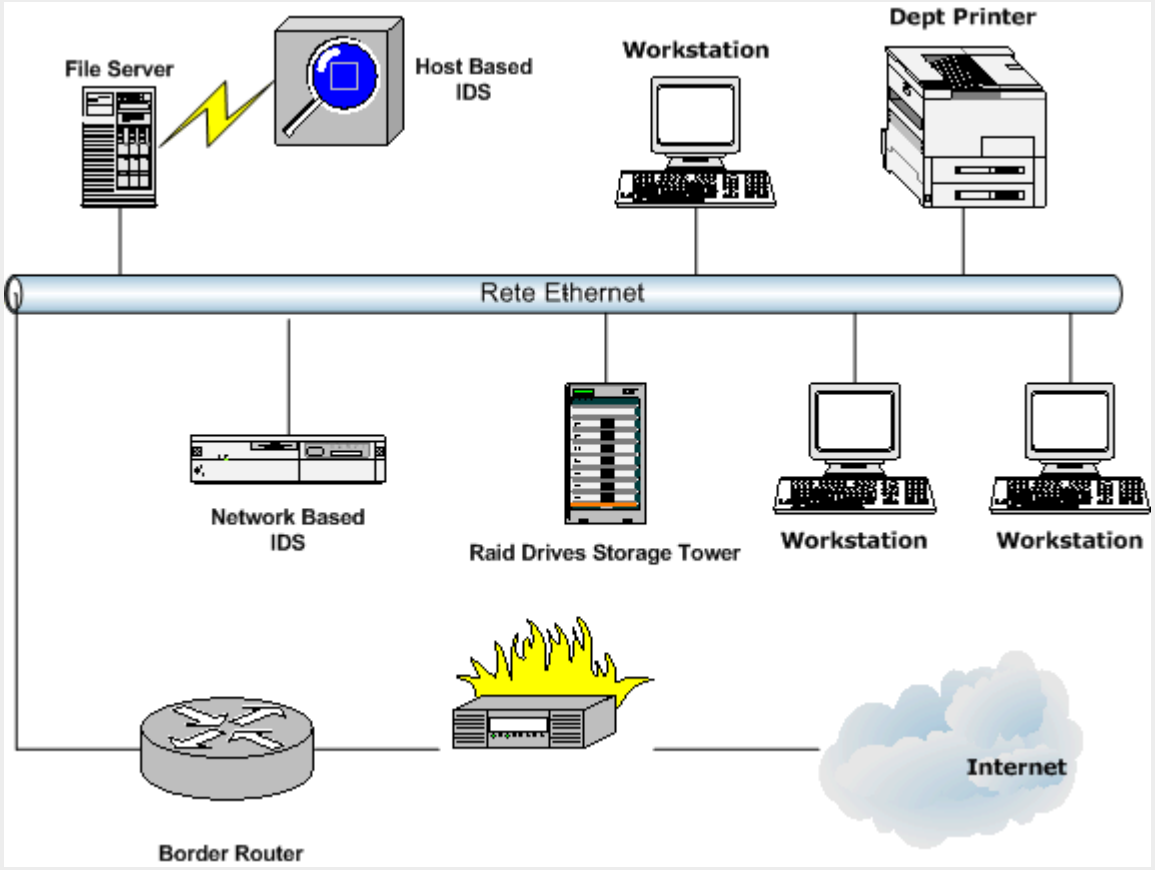
## Anomaly Detection Model

- Describes normal behaviour, and flags deviations
- Uses statistical or machine learning models of behaviour
- Theoretically able to recognize any attack, also 0-days
- Strongly dependent on the model, the metrics and the thresholds
- Generates statistical alerts: “Something’s wrong”

## Misuse Detection Model

- Uses a knowledge base to recognize the attacks
- Can recognize only attacks for which a “signature” exists in the KB
- When new types of attacks are created, the language used to express the rules may not be expressive enough
- Problems for polymorphism
- The alerts are precise: they recognize a specific attack, giving out many useful informations

# Host based vs. Network based



# Misuse detection alone is an awful idea

- Misuse detection systems rely on a knowledge base (think of the anti-virus example, if it's easier to grasp)
- Updates continuously needed, and not all the attacks become known (as opposed to viruses)
- Signature engineering problems (a NAV update, a couple of years ago, recognized *itself* as a virus…)
- Attacks are polymorphs, more than computer viruses: ADMutate, UTF encoding...
- Attacks are not atomic, and interleaving helps in avoiding detection!

# Anomaly Detection, perhaps not better

- We must describe the behaviour of a system
  - Which features/variables/metrics do we use?
  - Which model do we choose to fit them?
- Thresholds must be chosen to minimize false positive vs. detection rate: a difficult process
- The base model is fundamental: if the attack shows up only in variables we discarded, the system is blind on it!
- Any type of alert is more or less equivalent to “hey, something’s wrong here”. What? Your guess!

# SNORT: a Lightweight NIDS

# What's Snort ?

- Snort is a "multi-mode packet analysis tool", fundamentally a network based IDS sensor, which operates fundamentally on misuse detection (signatures)
- Developed by Martin Roesch
- Snort is available at www.snort.org
- Snort-based commercial appliances and support: www.sourcefire.com

# Characteristics of snort

- It's a lightweight, high-performance packet sniffer
- Based on Libpcap libraries
- Can work in 4 basic modes
  - Sniffer Mode (similar to tcpdump)
  - Packet Logger Mode (as above, but more output options)
  - NIDS Mode (activates attack rulesets)
  - Forensic Data Analysis Mode (as above, but takes a network dump as input)
- A plug-in system adds flexibility

# Snort plug-ins

- Preprocessor: examines and mangles packets before detection stage
- Detection plugin: executes tests on a field or aspect of packet
- Output plugin: transform analysis data for representation, visualization and logging

# A complete NIDS sensor

- Snort rules are widely available, a true standard: they are even inserted in advisories sometimes. Even ISS added snort rules format support to RealSecure
- Detection Plug-ins extend rules expressivity, allowing to write rules not only matching packet content and header, but also adding statistical informations and protocol anomaly detection
- Snort preprocessors: portscan detection, IP defragmentation, TCP stream reassembly, application layer analysis, etc.

# Snort outputs

- Database
  - MySQL
  - PostgreSQL
  - Oracle
- XML (SNML DTD by CMU/CERT)
- tcpdump/libpcap format
- Unified format (Snort's own format)
- ASCII
- Syslog
- WinPopup (SMB)

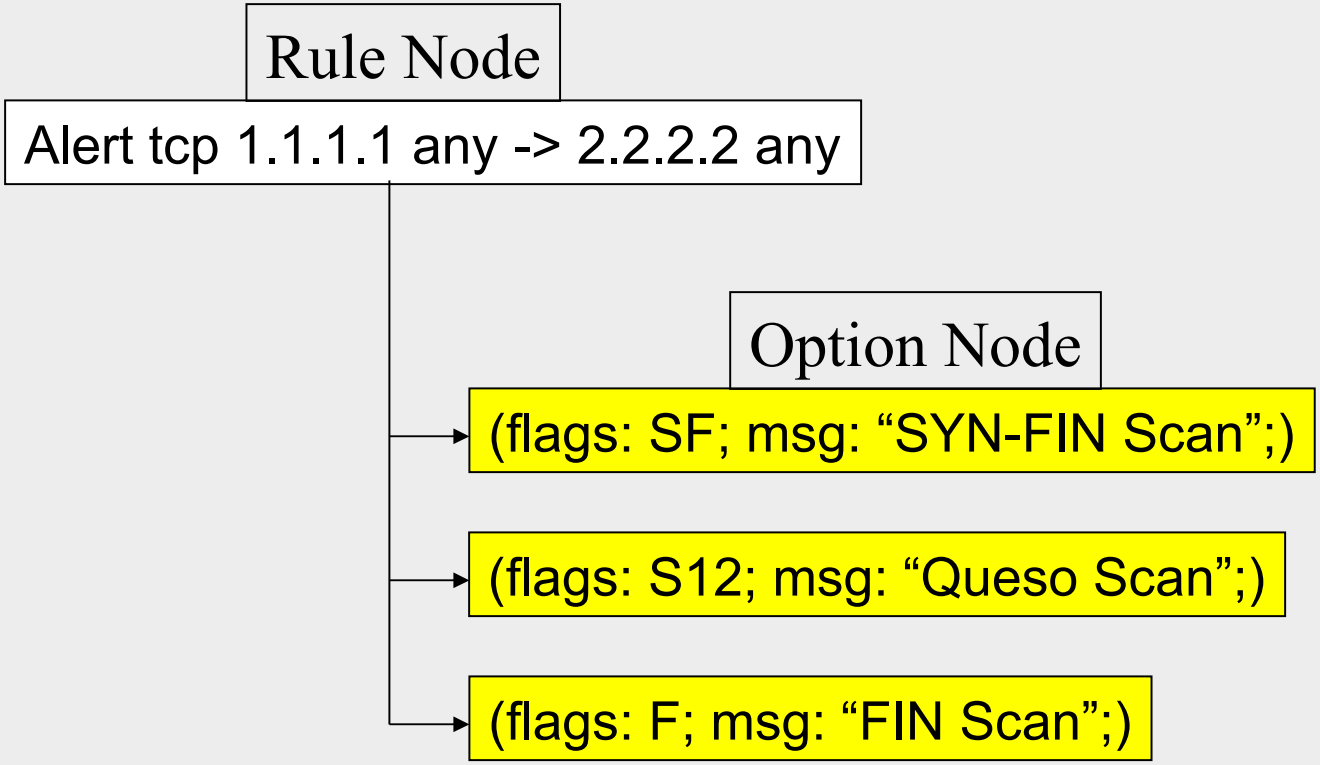
# Snort 1.x Detection Engine

- Tridimensional linked list of rules
  - The first and the second dimension contain the parameters to test on the packets
  - Third dimension contains pointers to functions that must be used on packets
  - The engine is recursively applied on packets
  - First match detection: as soon as one rule matches, packet is discarded and next packet is processed
- Snort reaches with few problems 100Mb/sec of throughput on really common hardware

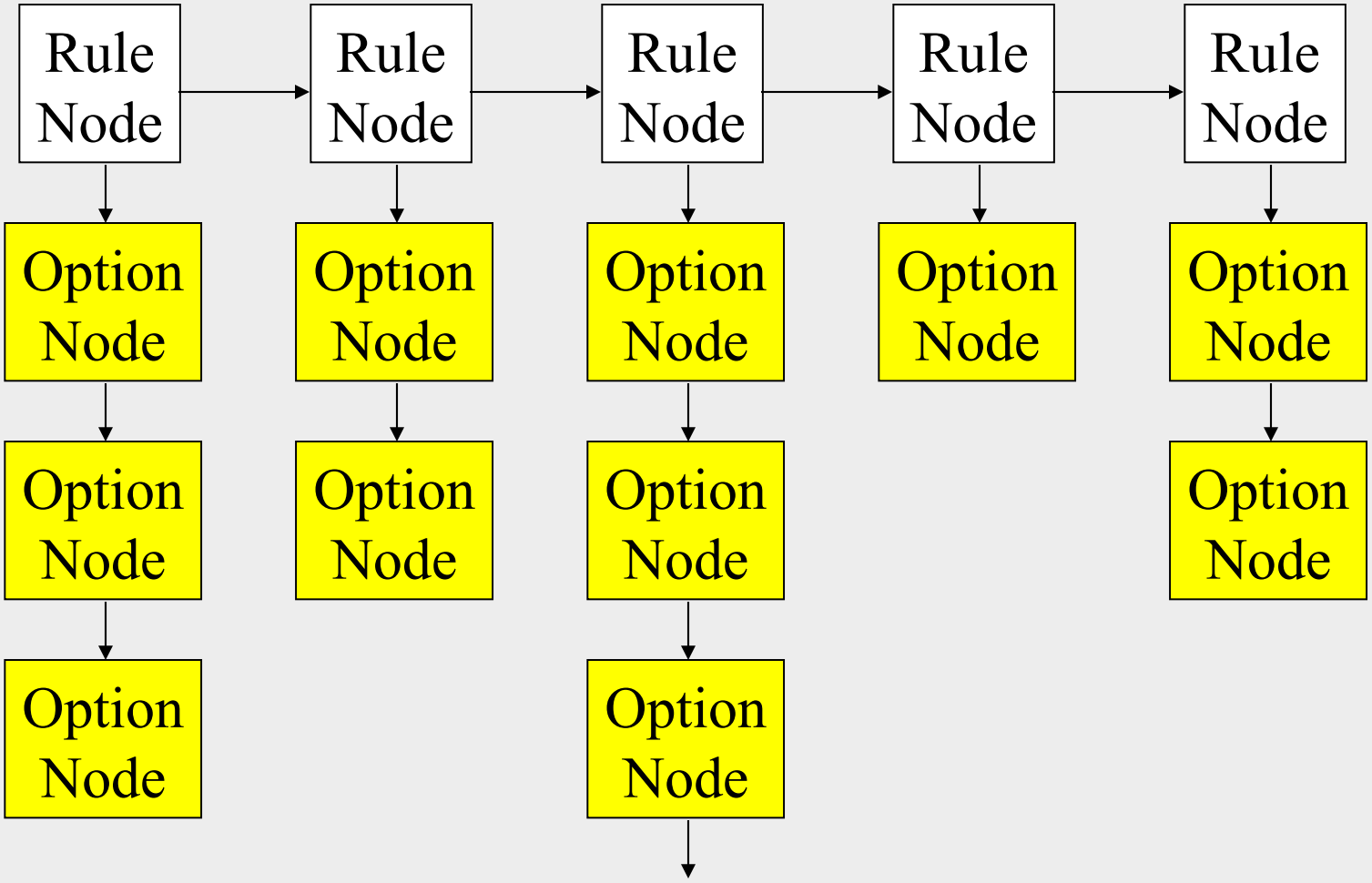
# Three sample rules

| Rule Header | Rule Options |
| --- | --- |
| Alert tcp 1.1.1.1 any -> 2.2.2.2 any | (flags: SF; msg: “SYN-FIN Scan”;) |
| Alert tcp 1.1.1.1 any -> 2.2.2.2 any | (flags: S12; msg: “Queso Scan”;) |
| Alert tcp 1.1.1.1 any -> 2.2.2.2 any | (flags: F; msg: “FIN Scan”;) |

# Their representation…



# You got the idea ?



# Snort 1.x architectural limits

- Focused on the single IP packet
- IP defragmentation and TCP stream reassembly as preprocessors… often becoming bottleneck
- Output plugins, too, were wasting resources
- New protocol support was not modular and not easy to add
- Application layer not managed by the rule engine, relayed on the shoulders of the rule writer. So, simple to write rules for IP/TCP/UDP/ICMP/IGMP… but complext to describe HTTP, RPC, SMTP…

# Snort 2.0: a new architecture

- Lots of new plug-in types
  - Acquires various data formats
  - An extensible traffic protocol decoder (protocol verification, stream analysis on multi-path routing…)
  - Multiple formats for rules (from a DBMS, XML format…)
  - Detection engine is a plug-in: Standard NIDS, Target-based IDS, Statistical IDS, Host-based IDS…
- Improved pattern matching:
  - Aho-Corasick; Boyer-Moore; set-wise Boyer-Moore-Horspool algorithms
    - in human terms, a speedup of five times!
- Output plugins are now attached to a secondary process (“barnyard”) which acts as a buffer, avoiding Snort the output bottleneck

# Regarding snort configuration

- Configuration files with explanations and comments, for 1.9 and 2.0: http://www.0xdeadbeef.info/
  - Lots of other goodies, there, too :)
- The user manual explains everything about rule writing: www.snort.org/docs/SnortUsersManual.pdf
- "Intrusion Signatures and Analysis" (Northcutt, Cooper, Fearnow, Frederick): a great book, generic, but using snort as a specific example!

# HIDS

- An host based IDS sensor uses system auditing auditing resources to gather information
  - System logs
  - System events (e.g. login attempts)
  - System calls (file access, devices…)
- File integrity checking is a form of HIDS: see Tripwire, www.tripwire.com
- Anomaly detection is often prominent here!
- A compromised system is NOT a great source of information, as you may guess…

# What is an “honeypot” ?

- *An honeypot is a security resource whose value lies in being probed, attacked, or compromised*
- Simple concept: a resource that expects no data, so any traffic to or from it is most likely unauthorized activity
- A form of “anomaly detection” !
- Honeypots *do not solve a specific problem*, but can be a tool that contributes to your overall security architecture
- Their value, and the problems they help solve, depends on how you build, deploy, and use them

# How do they fit into the IDS scenario ?

- Prevention
  - Honeypots, in general are not effective prevention mechanisms.
  - "Deception, Deterrence, Decoy": psychological weapons. They do NOT work against worms or auto-rooters
- Detection
  - They are a great anomaly IDS
  - Unique advantages
- Response
  - Honeypots can be used to help respond to an incident
  - They can be pulled offline and analyzed (unlike production systems)
  - Little to no data pollution (all the data on a honeypot is attack data)

# Also Research Honeypots

- Used by universities and research groups
- Early Warning and Prediction
- Discovery of new Tools and Tactics
- They help to understand motives, behavior, and organization of the criminals
- Used for training and development of analysis and forensic skills
- E.g. the Honeynet project (born in 1999)
- E.g. WOMBAT project (www.wombat-project.eu)

# Example: Motives and Behavior

- Captured IRC session between hacker and pal on a private channel

**J4ck:** why don't you start charging for packet attacks?
**J4ck:** "give me x amount and I'll take bla bla offline for this amount of time"
**J1LL:** it was illegal last I checked.
**J4ck:** heh, then everything you do is illegal. Why not make money off of it?
**J4ck:** **I know plenty of people that'd pay exorbatent amounts for packeting.**

# Level of Interaction

- Level of Interaction determines amount of functionality a honeypot provides: “how real and complex it looks”
- The greater the interaction, the more you can learn
- The greater the interaction, the more complexity and risk
- Chances are that an attacker can use your honeypot to harm, attack, or infiltrate other systems or organizations
- Which is best? It depends on your objectives

# Low Interaction vs. High Interaction

- Lowly interactive honeypots:
  - Provide Emulated Services
  - No operating system for attacker to access
  - Information limited to transactional information and attackers activities with emulated services
  - Many attackers will realize that services are fake after a while
- Highly interactive honeypots:
  - Provide Actual Operating Systems
  - Learn extensive amounts of information
  - High risk: a subverted honeypot can be turned against you

# Honeyd

```
create default

set default personality "FreeBSD 2.2.1-STABLE"

set default default action open

add default tcp port 80 "sh /usr/local/honeyd/scripts/web.sh"


add default tcp port 22 "sh /usr/local/honeyd/scripts/test.sh"


add default tcp port 113 reset

add default tcp port 1 reset



create windows
set windows personality "Windows NT 4.0 Server SP5-SP6"
set windows default action reset
add windows tcp port 80 "sh /usr/local/honeyd/scripts/web.sh"
```

# Honeytokens

- Honeytoken: a honeypot that is not a computer
  - A honeytoken can be a credit card number, Excel spreadsheet, PowerPoint presentation, a database entry, or even a bogus login.
  - Just as a honeypot computer has no authorized value, no honeytoken has any authorized use.
- Map-making companies inserting bogus cities or roads into maps to spot copies
- Example:
  - Add a bogus credit card number into an e-commerce DB
  - E.g., the credit card number 435697483758471O
  - Add a Snort rule: alert ip any any -> any any (msg:"Honeytoken Access"; content:"435697483758471O";)
- Minimal cost and complication!

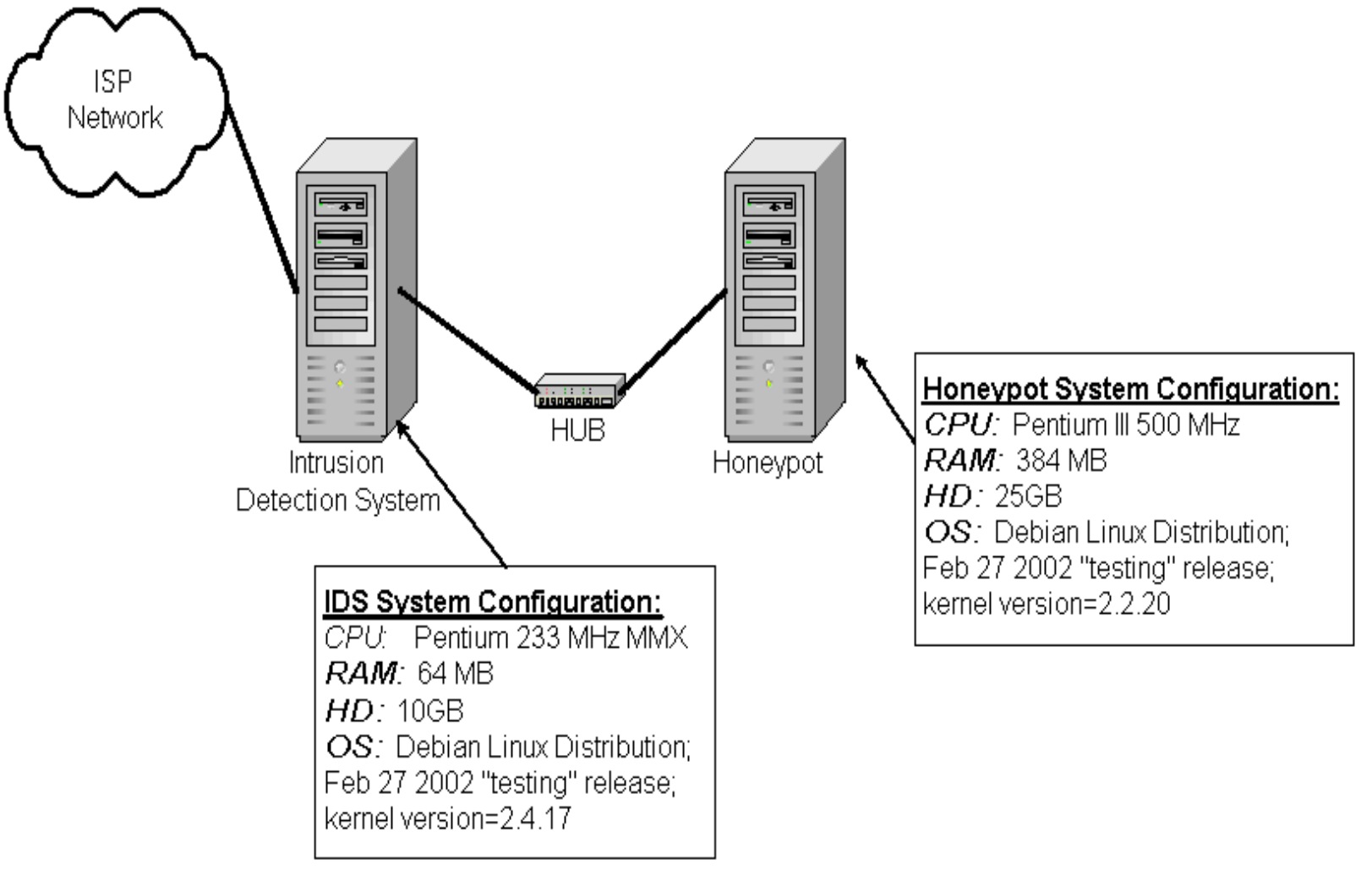
# Honeynets

- http://project.honeynet.org
- An extension of a Honeypot
- Network topology provides many advantages over standard honeypot
  - Covert logging
  - More points of attack for a blackhatter
  - Looks realistic from the outside
- In a typical honeynet, there may be one or more honeypot machines spanning several operating systems

# Basics of an honeynet

- Containment: once a honeypot within the Honeynet is compromised, we have to contain the activity and ensure the honeypots are not used to harm non-Honeynet systems. We must control how traffic can flow in and out of the Honeynet
- Data Capture: capture all activity within the Honeynet and the information that enters and leaves the Honeynet
- Data Collection: Once data is captured, it is securely forwarded to a centralized data collection point
- … All of these avoiding that the “bad guys” detect us!

# A sample Honeynet architecture



# Limitations of HIH and LIH

- LIH are limited to known services and specific types of attacks
- HIH are labour-intensive and prone to disasters, e.g. being taken over by an attacker and abused
- Many projects (e.g. sgnet) aim to blend the two types of systems, e.g. by automated learning