Lecture 22 – Intrusion Detection

November 8, 2018

Dr. Dan Massey

Motivating Example: SNORT Rule

- alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-ATTACKS /bin/ps command attempt"; flow:to_server,established; uricontent:"/bin/ps"; nocase; classtype:web-application-attack; sid:1328; rev:6;)

Network Layer Basics: IP Format and Addressing
Transport Layer Basics: UDP/TCP Header and connections
Application Layer: vast numbers of applications

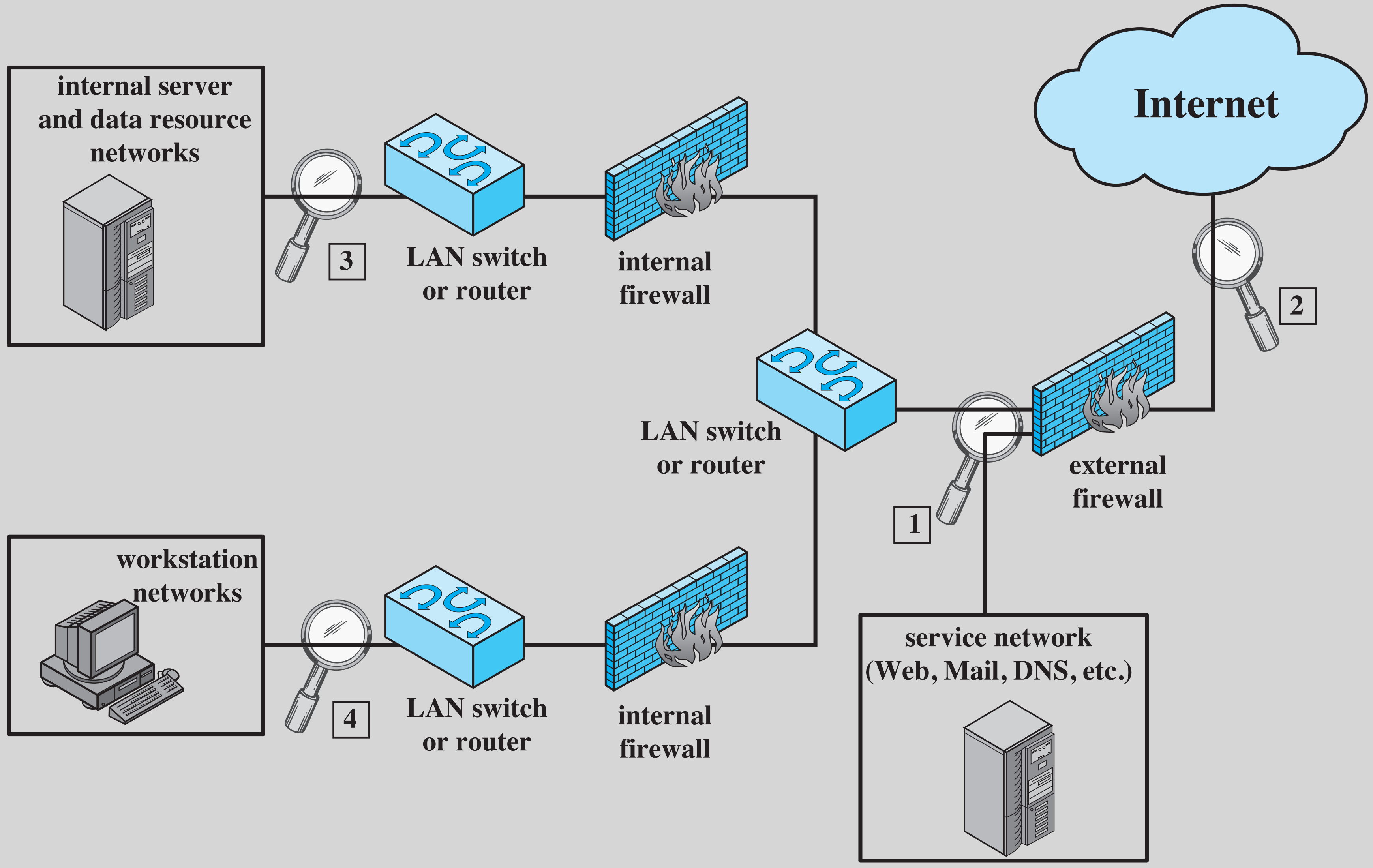


Figure 8.5 Example of NIDS Sensor Deployment

Intrusion Detection Techniques

Attacks suitable for

Signature detection

- Application layer reconnaissance and attacks
- Transport layer reconnaissance and attacks
- Network layer reconnaissance and attacks
- Unexpected application services
- Policy violations

Attacks suitable for

Anomaly detection

- Denial-of-service (DoS) attacks
- Scanning
- Worms

SNORT

- lightweight IDS
  - real-time packet capture and rule analysis
  - easily deployed on nodes
  - uses small amount of memory and processor time
  - easily configured

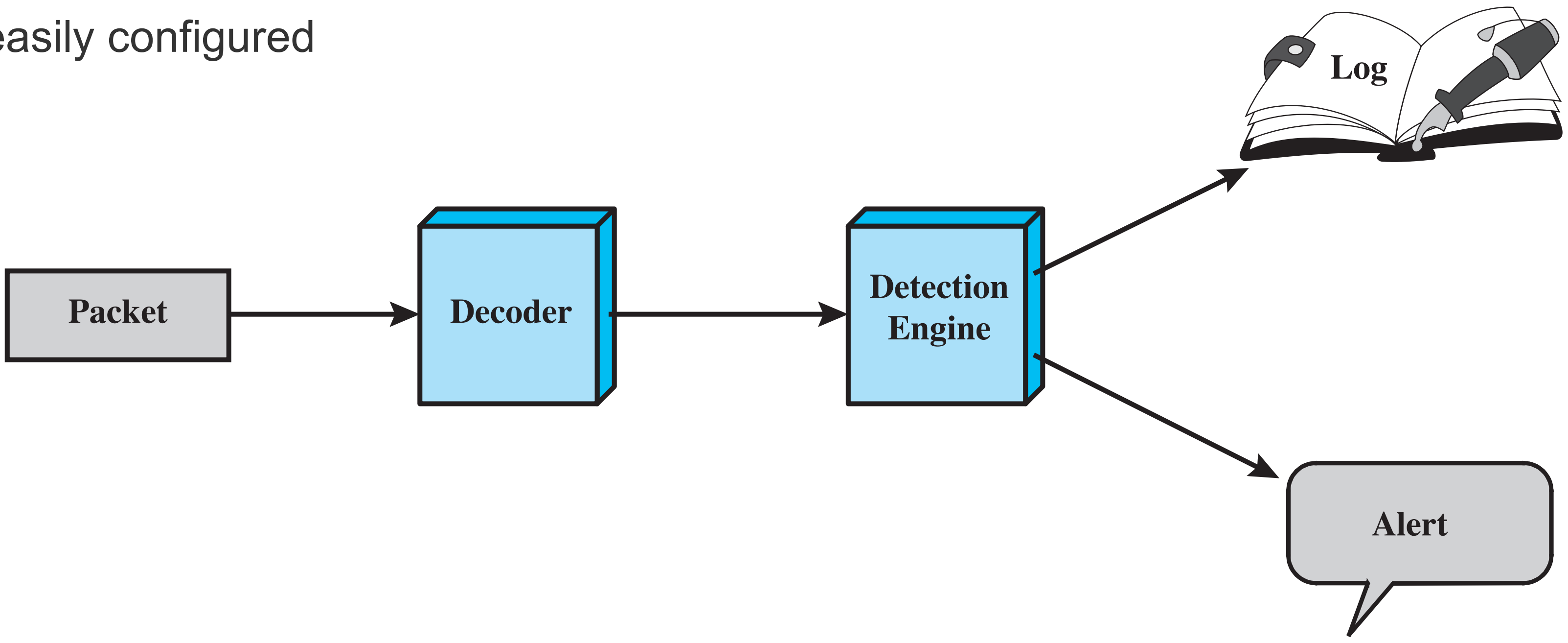


Figure 8.9 Snort Architecture

Action	**Protocol**	**Source IP address**	**Source Port**	**Direction**	**Dest IP address**	**Dest Port**

(a) Rule Header

Option Keyword	**Option Arguments**	• • •

(b) Options

Figure 8.10 Snort Rule Formats

SNORT Example

- alert tcp $HOME_NET any <> $EXTERNAL_NET 6666:7000 (msg:"CHAT IRC message"; flow:established; content:"PRIVMSG "; nocase; classtype:policy-violation; sid:1463; rev:6;)

- alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-ATTACKS /bin/ps command attempt"; flow:to_server,established; uricontent:"/bin/ps"; nocase; classtype:web-application-attack; sid:1328; rev:6;)

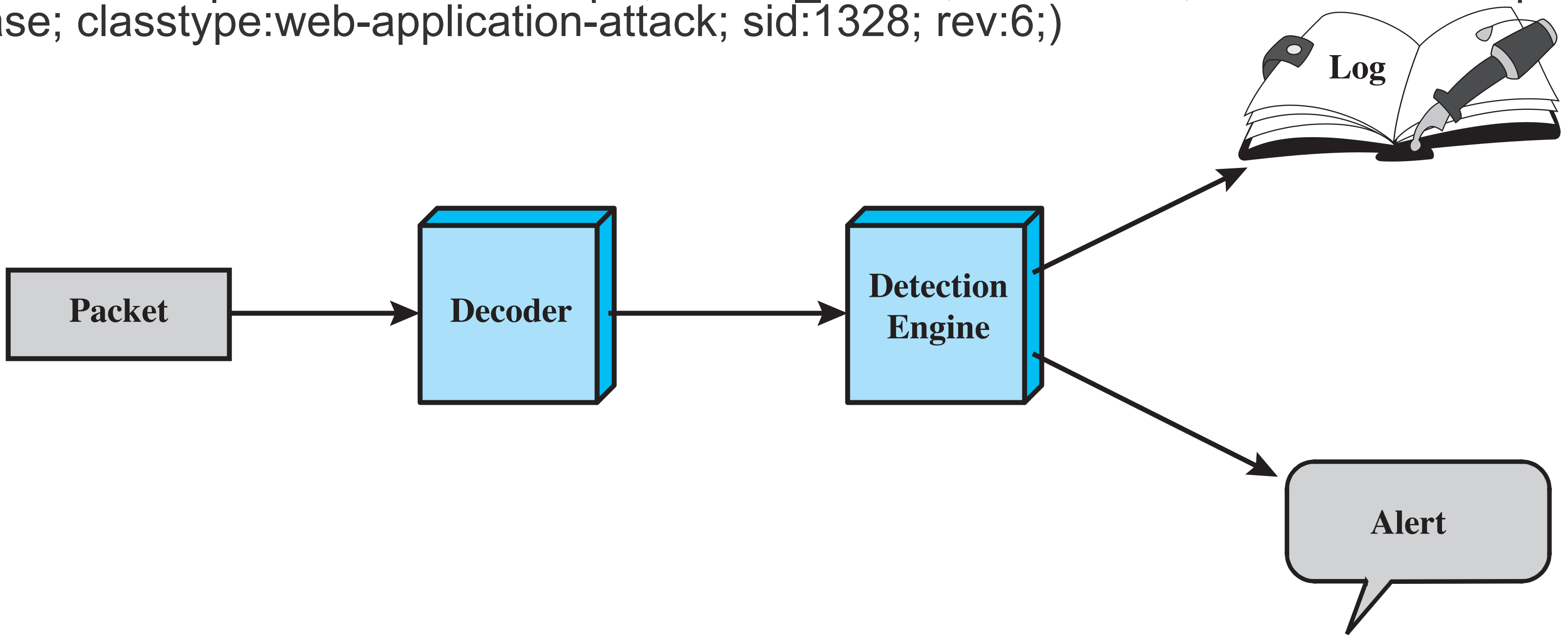
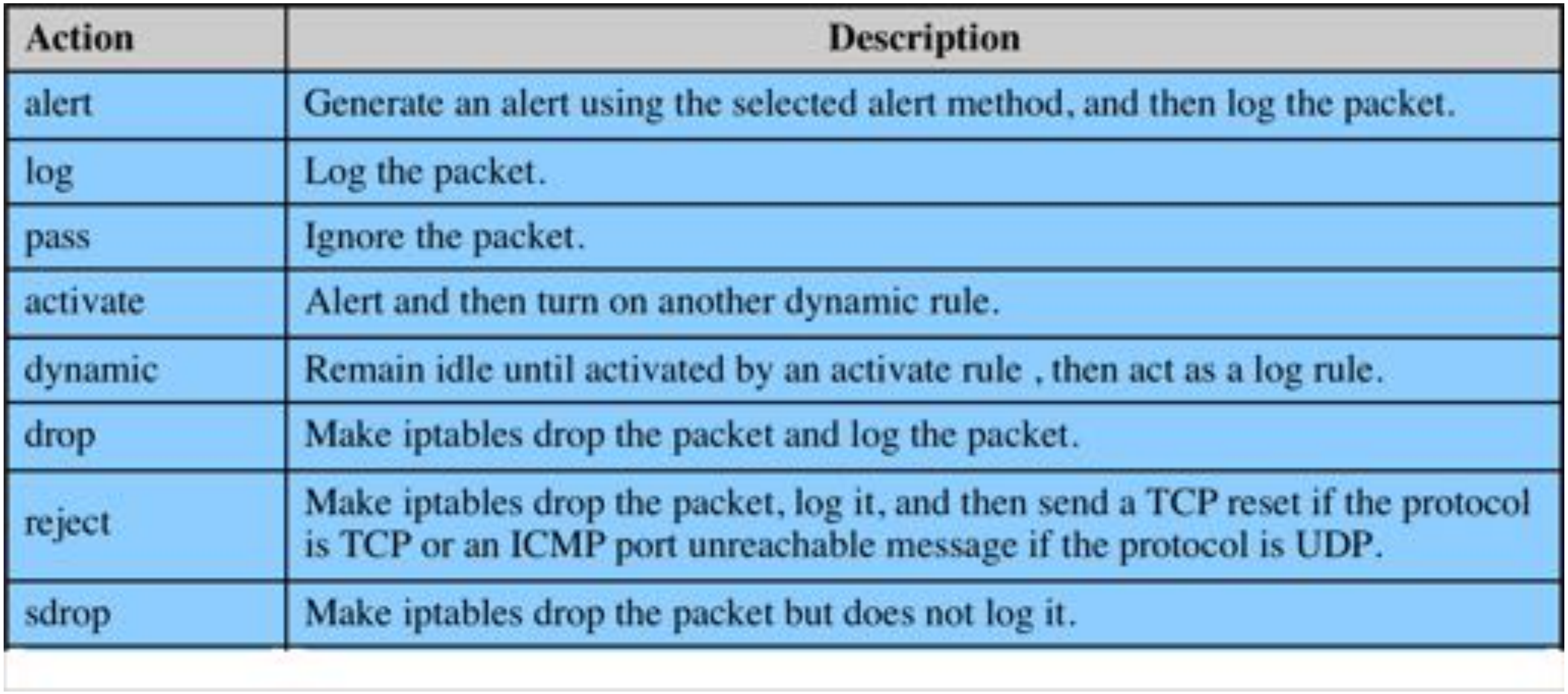


Figure 8.9 Snort Architecture

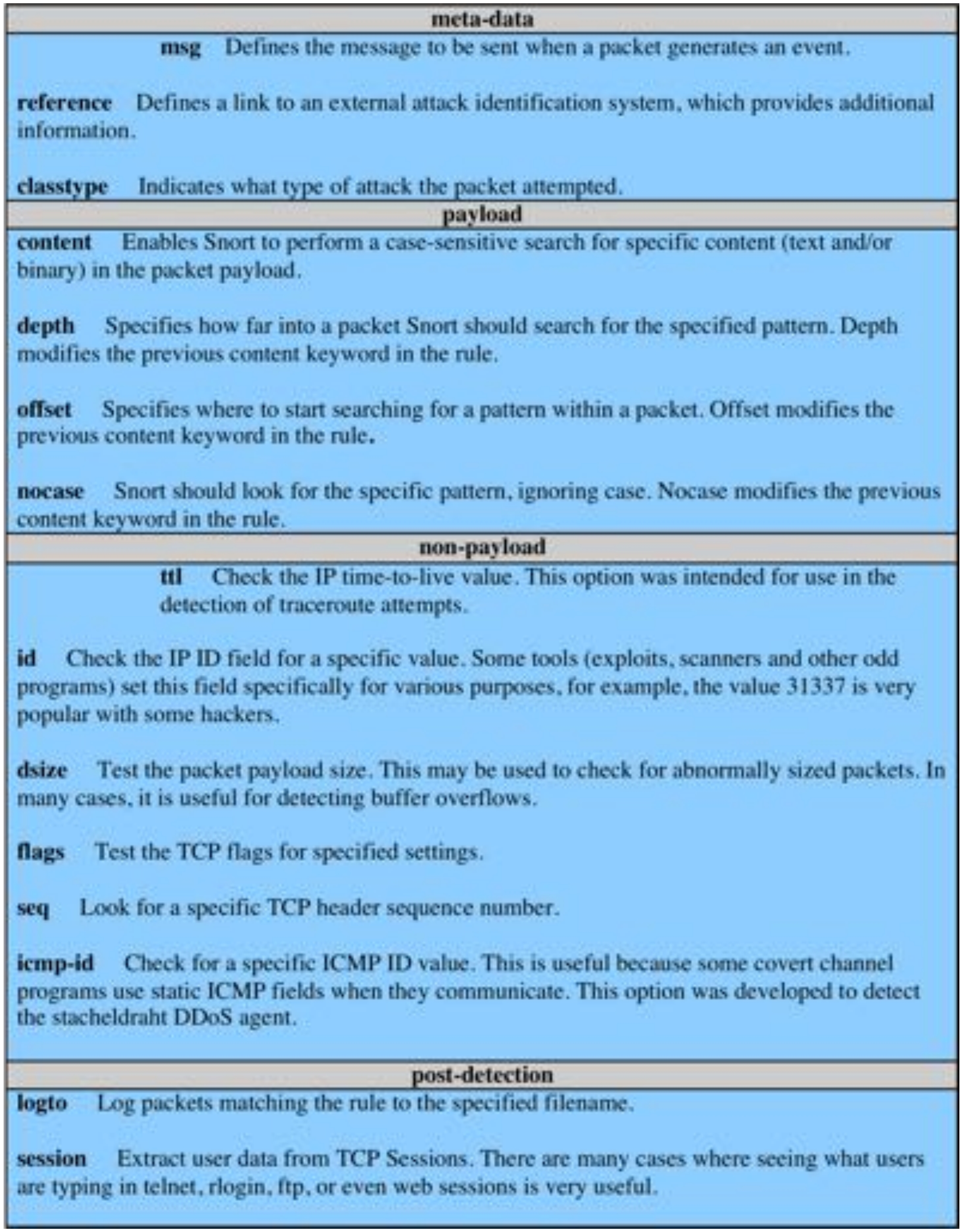
SNORT Rules

- use a simple, flexible rule definition language
- each rule consists of a fixed header and zero or more options



Action	Description
alert	Generate an alert using the selected alert method, and then log the packet.
log	Log the packet.
pass	Ignore the packet.
activate	Alert and then turn on another dynamic rule.
dynamic	Remain idle until activated by an activate rule , then act as a log rule.
drop	Make iptables drop the packet and log the packet.
reject	Make iptables drop the packet, log it, and then send a TCP reset if the protocol is TCP or an ICMP port unreachable message if the protocol is UDP.
sdrop	Make iptables drop the packet but does not log it.

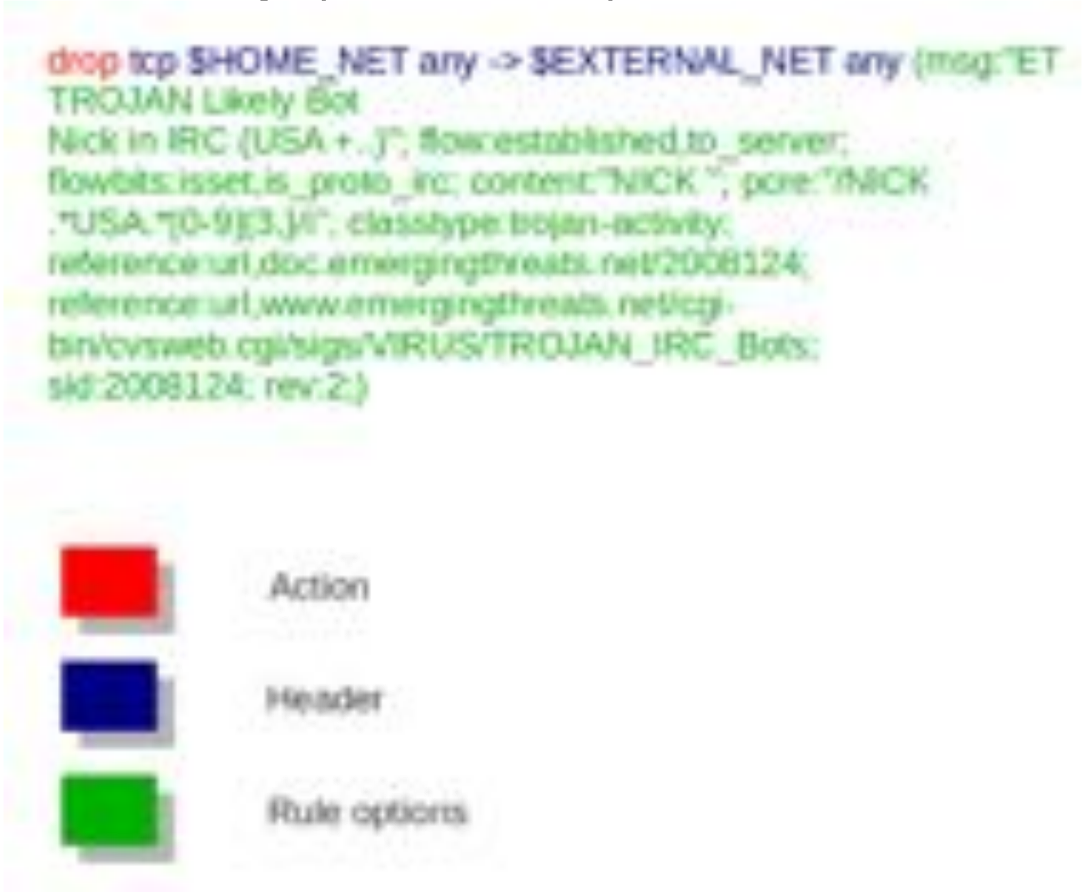
Examples of SNORT Rule Options



meta-data	
msg	Defines the message to be sent when a packet generates an event.
reference	Defines a link to an external attack identification system, which provides additional information.
classtype	Indicates what type of attack the packet attempted.
payload	
content	Enables Snort to perform a case-sensitive search for specific content (text and/or binary) in the packet payload.
depth	Specifies how far into a packet Snort should search for the specified pattern. Depth modifies the previous content keyword in the rule.
offset	Specifies where to start searching for a pattern within a packet. Offset modifies the previous content keyword in the rule.
nocase	Snort should look for the specific pattern, ignoring case. Nocase modifies the previous content keyword in the rule.
non-payload	
ttl	Check the IP time-to-live value. This option was intended for use in the detection of traceroute attempts.
id	Check the IP ID field for a specific value. Some tools (exploits, scanners and other odd programs) set this field specifically for various purposes, for example, the value 31337 is very popular with some hackers.
dsize	Test the packet payload size. This may be used to check for abnormally sized packets. In many cases, it is useful for detecting buffer overflows.
flags	Test the TCP flags for specified settings.
seq	Look for a specific TCP header sequence number.
icmp-id	Check for a specific ICMP ID value. This is useful because some covert channel programs use static ICMP fields when they communicate. This option was developed to detect the stacheldraht DDoS agent.
post-detection	
logto	Log packets matching the rule to the specified filename.
session	Extract user data from TCP Sessions. There are many cases where seeing what users are typing in telnet, rlogin, ftp, or even web sessions is very useful.

Suricata – Snort Competitor

- SourceFire acquired by Cisco
  - Continues to be available open source
  - Concerns over future development in multi-threaded code, code updates, etc
  - Suricata introduced as fully open source implemenation.



```
drop tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"ET
TROJAN Likely Bot
Nick in IRC (USA +..)"; flow:established,to_server;
flowbits:isset,is_proto_irc; content:"NICK "; pcre:"/NICK
.*USA.*[0-9]{3,}/i"; classtype:trojan-activity;
reference:url,doc.emergingthreats.net/2008124;
reference:url,www.emergingthreats.net/cgi-
bin/cvsweb.cgi/sigs/VIRUS/TROJAN_IRC_Bots;
sid:2008124; rev:2;)
```

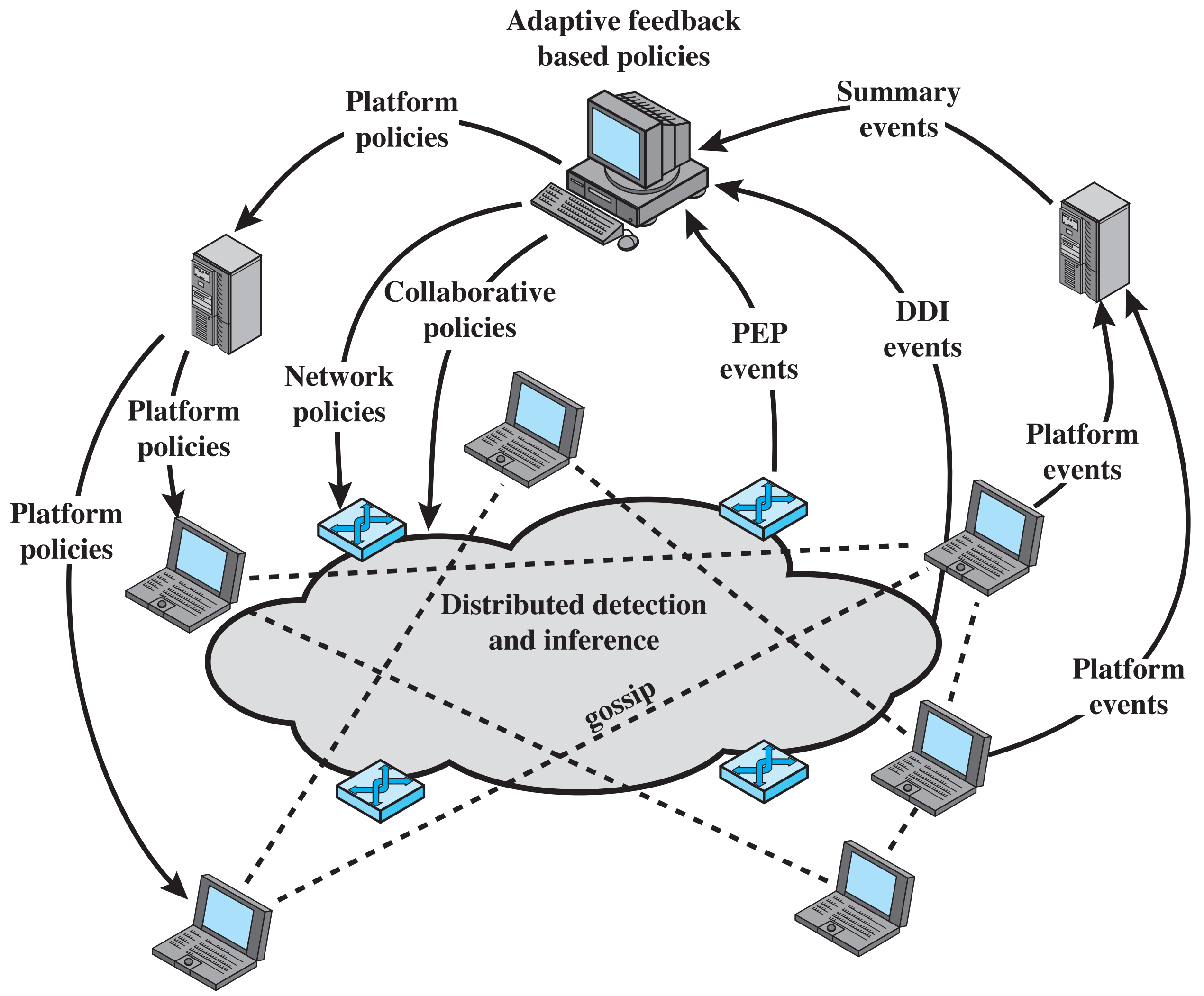
🟥	Action
🟦	Header
🟩	Rule options

Stateful Protocol Analysis (SPA)

- Subset of anomaly detection that compares observed network traffic against predetermined universal vendor supplied profiles of benign protocol traffic
  - This distinguishes it from anomaly techniques trained with organization specific traffic protocols
- Understands and tracks network, transport, and application protocol states to ensure they progress as expected
- A key disadvantage is the high resource use it requires

Logging of Alerts

- Typical information logged by a NIDS sensor includes:
  - Timestamp
  - Connection or session ID
  - Event or alert type
  - Rating
  - Network, transport, and application layer protocols
  - Source and destination IP addresses
  - Source and destination TCP or UDP ports, or ICMP types and codes
  - Number of bytes transmitted over the connection
  - Decoded payload data, such as application requests and responses
  - State-related information



PEP = policy enforcement point
DDI = distributed detection and inference

Figure 8.6 Overall Architecture of an Autonomic Enterprise Security System

IETF Intrusion Detection Working Group

- Purpose is to define data formats and exchange procedures for sharing information of interest to intrusion detection and response systems and to management systems that may need to interact with them
- The working group issued the following RFCs in 2007:

Intrusion Detection Message Exchange Requirements (RFC 4766)

- Document defines requirements for the Intrusion Detection Message Exchange Format (IDMEF)
- Also specifies requirements for a communication protocol for communicating IDMEF

The Intrusion Detection Message Exchange Format (RFC 4765)

- Document describes a data model to represent information exported by intrusion detection systems and explains the rationale for using this model
- An implementation of the data model in the Extensible Markup Language (XML) is presented, and XML Document Type Definition is developed, and examples are provided

The Intrusion Detection Exchange Protocol (RFC 4767)

- Document describes the Intrusion Detection Exchange Protocol (IDXP), an application level protocol for exchanging data between intrusion detection entities
- IDXP supports mutual authentication, integrity, and confidentiality over a connection oriented protocol

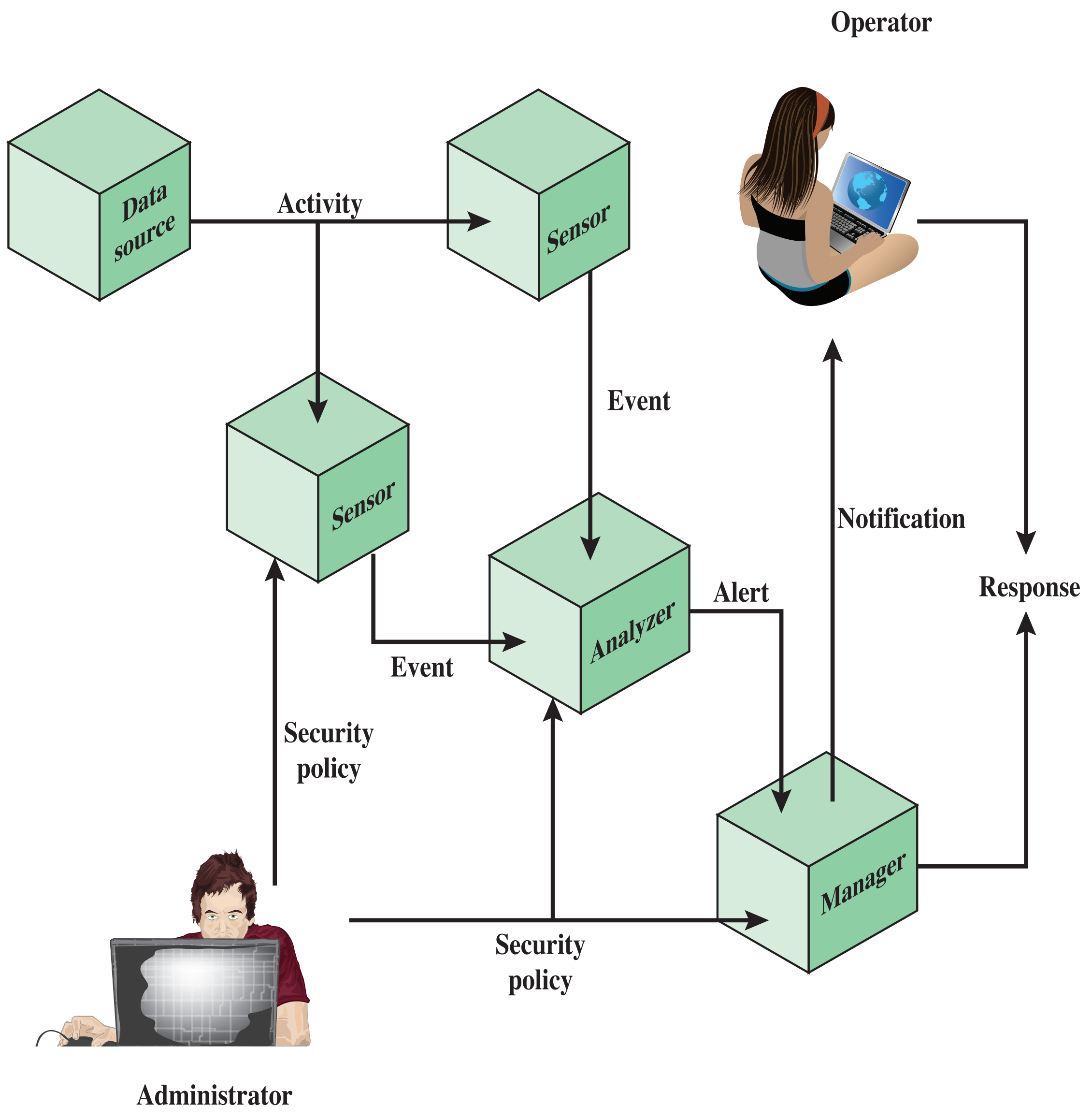


Figure 8.7 Model For Intrusion Detection Message Exchange

Honeypots

- Decoy systems designed to:
  - Lure a potential attacker away from critical systems
  - Collect information about the attacker’s activity
  - Encourage the attacker to stay on the system long enough for administrators to respond
- Systems are filled with fabricated information that a legitimate user of the system wouldn’t access
- Resources that have no production value
  - Therefore incoming communication is most likely a probe, scan, or attack
  - Initiated outbound communication suggests that the system has probably been compromised

Honeypot Classifications

- Low interaction honeypot
  - Consists of a software package that emulates particular IT services or systems well enough to provide a realistic initial interaction, but does not execute a full version of those services or systems
  - Provides a less realistic target
  - Often sufficient for use as a component of a distributed IDS to warn of imminent attack
- High interaction honeypot
  - A real system, with a full operating system, services and applications, which are instrumented and deployed where they can be accessed by attackers
  - Is a more realistic target that may occupy an attacker for an extended period
  - However, it requires significantly more resources
  - If compromised could be used to initiate attacks on other systems

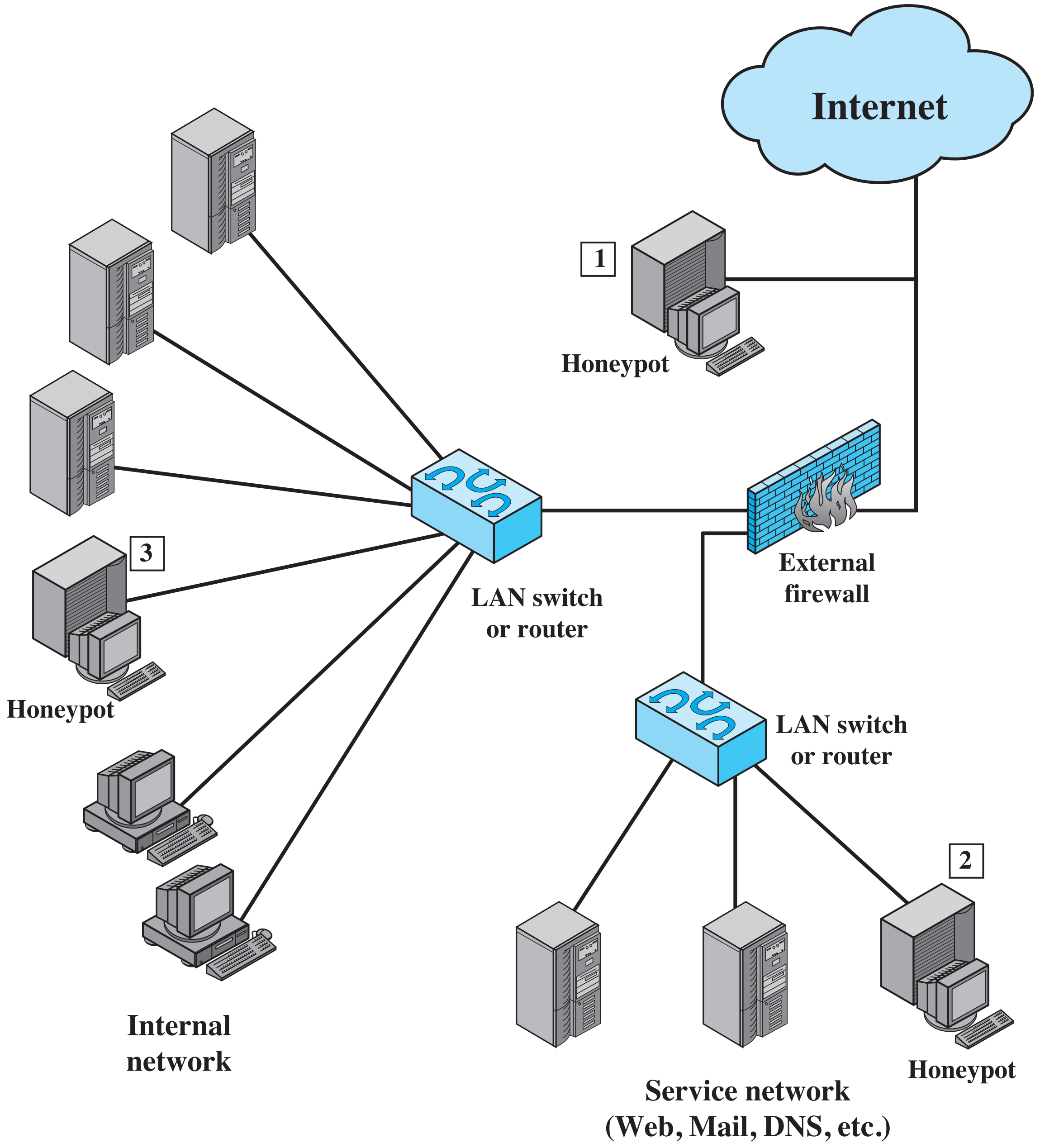


Figure 8.8 Example of Honeypot Deployment

Summary

- Intruders
  - Intruder behavior
- Intrusion detection
  - Basic principles
  - The base-rate fallacy
  - Requirements
- Analysis approaches
  - Anomaly detection
  - Signature or heuristic detection
- Distributed or hybrid intrusion detection
- Intrusion detection exchange format
- Honeypots
- Host-based intrusion detection
  - Data sources and sensors
  - Anomaly HIDS
  - Signature or heuristic HIDS
  - Distributed HIDS
- Network-based intrusion detection
  - Types of network sensors
  - NIDS sensor deployment
  - Intrusion detection techniques
  - Logging of alerts
- Example system: Snort
  - Snort architecture
  - Snort rules

Heilmeier Questions

- What are you trying to do? Articulate your objectives using absolutely no jargon.
- How is it done today, and what are the limits of current practice?
- What is new in your approach and why do you think it will be successful?
- Who cares? If you succeed, what difference will it make?
- What are the risks?
- How much will it cost?
- How long will it take?
- What are the mid-term and final “exams” to check for success?