University of Colorado **Boulder**

CSCI 3403 INTRO TO CYBERSECURITY

Lecture: 14-2

Topic: IDS

Presenter: Matt Niemiec

Announcements

- Project 3 (final project) is posted
- Assignment 10 due Thursday at 1:30PM
- Upcoming guest lecturers
  - From Rule4 on 4/21
  - From Twitter on 4/23 (Andy Sayler)

Exam Extra Credit

- Research a topic that won't be discussed in class
- Get credit in one of the following ways
  - Record a 10-minute (updated) video explanation of your topic for up to 10%. To receive credit for this, see Moodle
  - Present your topic for 5 minutes in recitation for up to 15%. Depending on demand, this may be first-come-first-serve. UPDATE: No matter your grade, please feel free to sign up for this option!
  - If you're selected as an outstanding project from recitation, give a 10-minute presentation for up to 25% (total)
- Percentages apply to your higher exam score

Exam Extra Credit Criteria

- Will be graded on at least the following:
  - Interesting topic/information relevant to cybersecurity
  - Quality, professional preparation and presentation
  - Inspires the listener to want to learn more and provides resources to do so
  - Shows insight and depth in research presented in an appropriate manner for the given timeframe
  - Responds knowledgeably and accurately to any questions asked

Some Extra Credit Potential Topics

- Network security
  - Wireless security, honeypots, cloud security, SIEM, Tor
- Applied security
  - OWASP Top 10, reverse engineering, penetration testing
- Crypto
  - Common crypto libraries, homomorphic encryption
- Windows
  - Windows/AD security, Windows CLI
- Miscellaneous
  - Ethics in security, auditing, data provenance
- Or anything else! Just run it by Matt or your TA (grader)

See What's Being Presented

- Official signup slots are on Moodle
- You can also put your topic down here: https://docs.google.com/document/d/1NPXj46NKz1L5zpS1fr8EMYn0vbhS75XDDmhOAtdBG0I/edit?usp=sharing so that
  - We're better prepared to listen to your presentation
  - You can get and share topic ideas with others
  - You'll know if a topic already has lots of people presenting
  - Signing up in the Google Doc isn't mandatory, but encouraged

Technology Recap 3/17 (Old stuff)

- Piazza is used for content-related questions
- Feedback: https://forms.gle/WRUUbPkmFNsa6q3D6
- Instructor/TA email is used for individual circumstances
- cyber@Colorado.edu is used for accommodations/logistical questions
- Moodle is used for assignments, slides, and additional resources

Technology Recap 3/17 (New stuff)

- Calendar is used for holding all Zoom meetings, instructions, and meeting IDs
  - May contain due dates, but not guaranteed
- Lecture Zoom ID: https://cuboulder.zoom.us/j/633893668
  - This and others found in Google Calendar
- Lecture capture folder: https://drive.google.com/drive/folders/1VMrHEigP4AgDwRnRPTsgQS35EAozc19-?usp=sharing

Intrusion Detection System

We Built Firewalls... Now What?

- Firewalls block certain types of traffic
  - Don’t do Deep Packet Inspection (DPI) that we saw
    - Examine the contents of the messages
  - Operates on exact matches
  - Doesn’t detect if outsider is inside the network
  - Doesn’t support multi-layered rule sets
    - “If this rule fails, run this other rule”
  - What if our computer gets ransomware and calls back to C&C?
    - That “type of traffic” is permitted

Intrusion Detection System (IDS)

- Security intrusion: “Unauthorized act of bypassing the security mechanisms of a system.”
- Intrusion detection: “A hardware or software function that gathers and analyzes information from various areas within a computer or a network to identify possible security intrusions.”

Intrusion Detection System (IDS)

- IDSs can help prevent the following types of attacks
  - Root compromise of server, defacing a Web server, guessing and cracking passwords, exporting a compromised database, viewing sensitive data, unauthorized packet sniffing, distributing pirated information, breaking an unsecured modem, spoofing internal calls/emails, using an unattended workstation
- Note: it's more the traffic itself than the type of traffic
- Less effective against highly sophisticated attacks

Things Intruders Do

Target acquisition and information gathering

Initial access

Privilege escalation

Information gathering or system exploit

Maintaining access

Covering tracks

How Strict to Make Security?

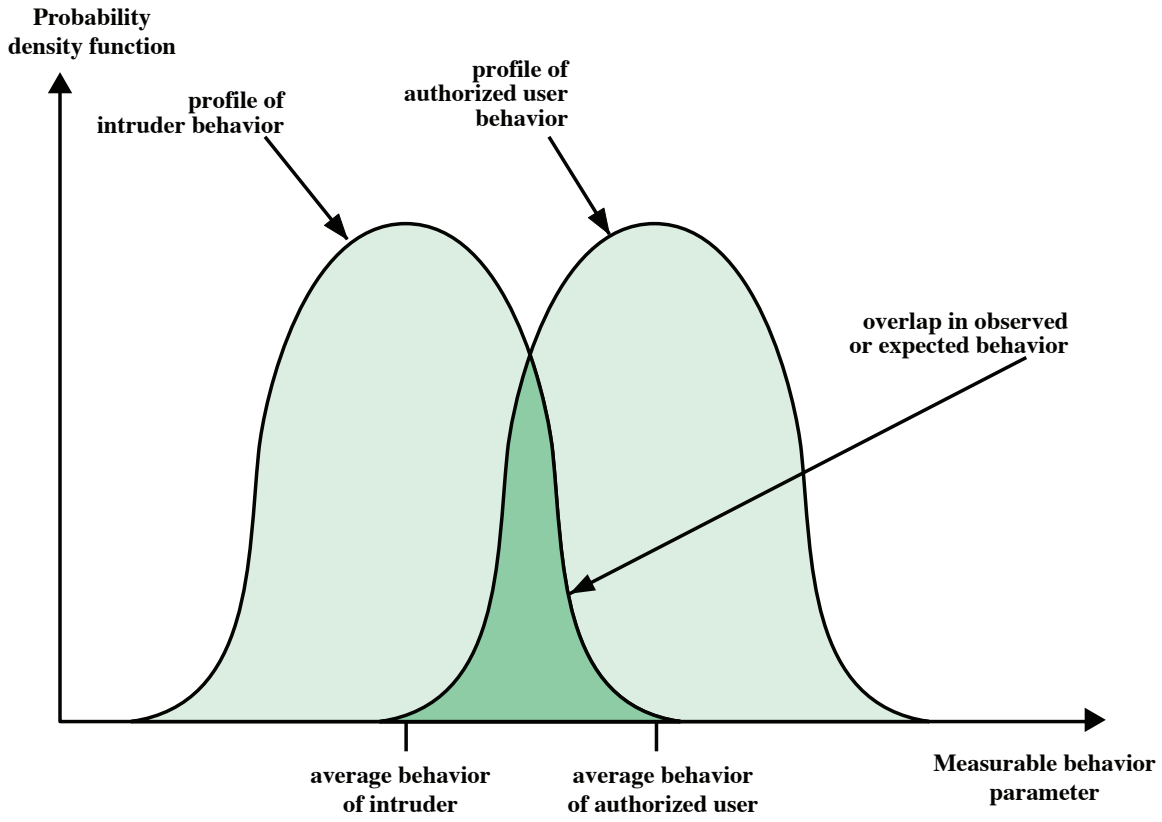

Figure 8.1 Profiles of Behavior of Intruders and Authorized Users

Anomaly Detection

- Anomaly detection
  - First establish baseline of what's "normal" on your network, then use comparison to see if traffic matches that
  - Typically uses machine learning
  - Problem: VERY high alert fatigue – math doesn't work well
  - Problem: Need baseline of what is "good" traffic

Signature/Heuristic Detection

- Signature detection
  - Similar to idea we saw with anti-virus
  - Compare with large collection of known bad traffic
- **Heuristic Detection**
  - Create rules for pattern matching
  - What we'll look at today

Network-based IDS

- As opposed to host-based IDS (not covered in this class
- Typically passive
  - Firewalls are active
- Less critical to infrastructure
  - Easy to replace/manage

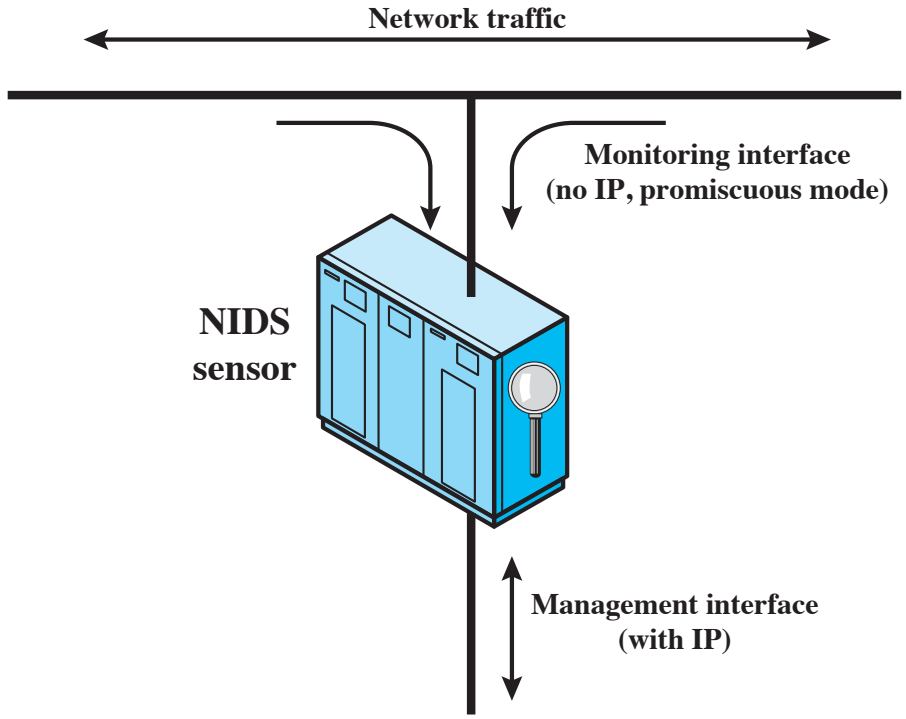

Figure 8.4 Passive NIDS Sensor

Where Do We Put IDS?

- **Question:** In front of or behind the firewall?

On A Network

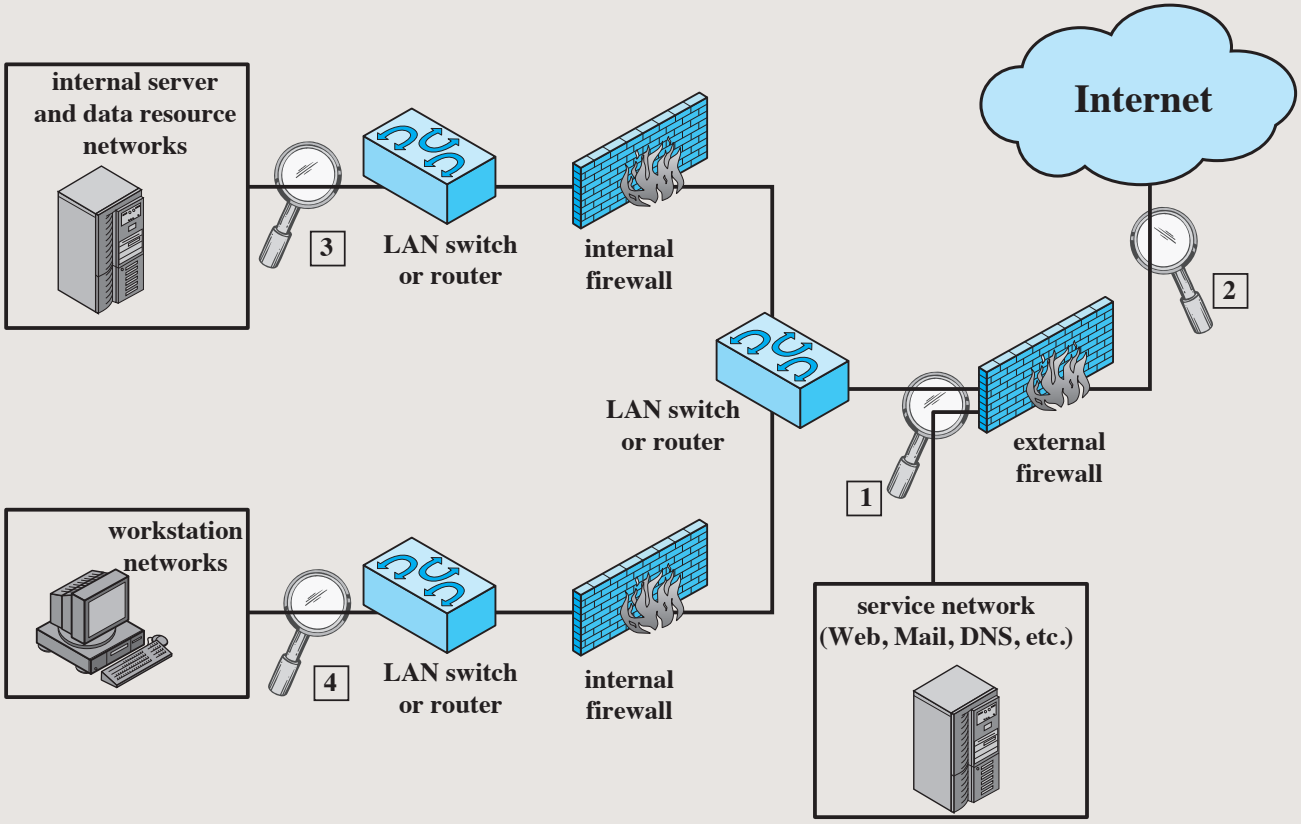

Figure 8.5 Example of NIDS Sensor Deployment

Where Do We Put IDS?

- **Question:** In front of or behind the firewall?
- **Answer:** Both, if possible
  - In front: Give idea of attacks that firewalls are blocking
  - Behind: Give idea of attacks that firewalls AREN'T blocking
- May depend on type of traffic you're trying to find

Problem

- Your NIDS captures all your network traffic
  - Now what?
- You can scroll through by hand, or use one of the previous techniques
- Each of the previous techniques takes your traffic, analyzes it, then makes some action

Snort

- Snort is an open-source*, highly-configurable and portable host-based or network-based IDS. It is referred to as a lightweight IDS, which has the following characteristics:
  - Easily deployed on most nodes of a network
  - Efficient operation that uses small amount of memory and processor time
  - Easily configured by system administrators who need to implement a specific security solution in a short amount of time

*The program is open-source, but you have to pay for the rules if you don't want to make your own

Snort

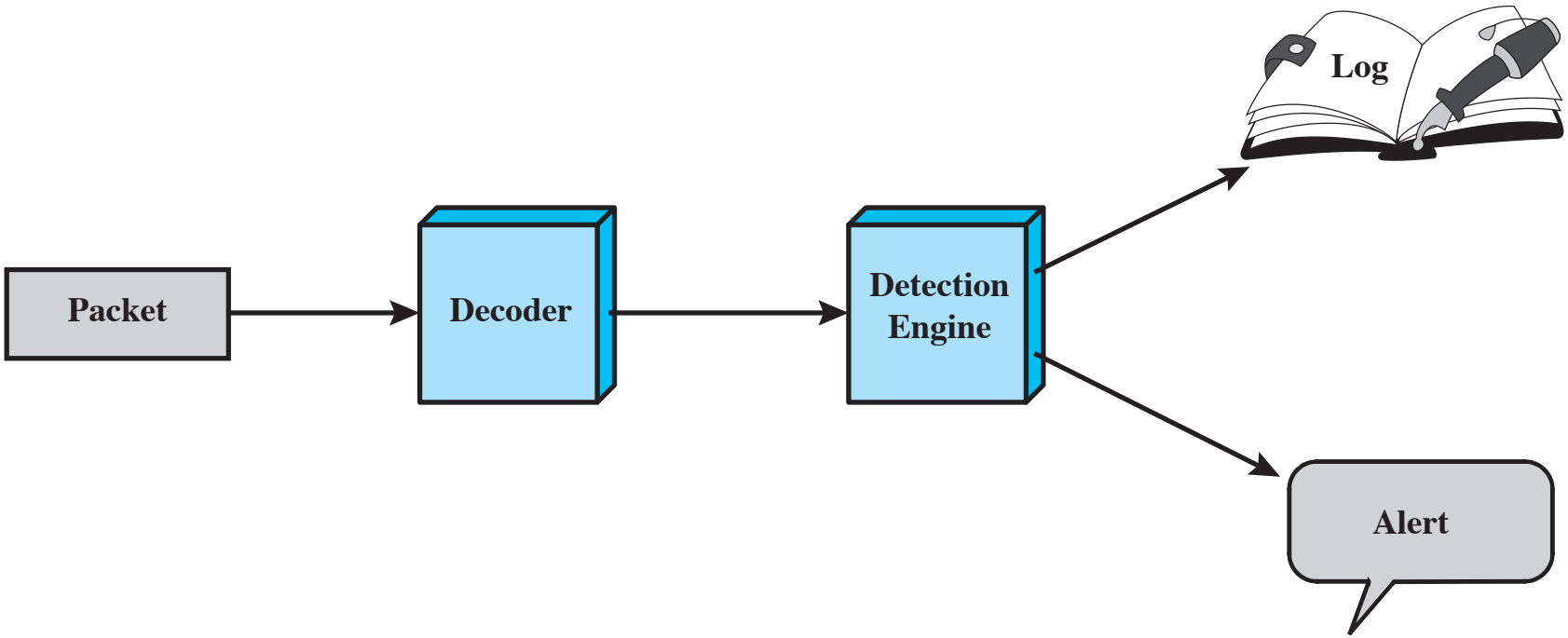

Figure 8.9 Snort Architecture

Snort

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 Rule Example

alert tcp $EXTERNAL_NET any -> $SQL_SERVERS $ORACLE_PORTS (msg: “ORACLE create database attempt”; flow: to_server, established; content: “create database”; nocase; classtype: protocol-command-decode; sid:100001, rev: 2)

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 Rule Example

alert tcp $EXTERNAL_NET any -> $SQL_SERVERS $ORACLE_PORTS (msg: “ORACLE create database attempt”; flow: to_server, established; content: “create database”; nocase; classtype: protocol-command-decode; sid:100001, rev: 2)

- alert is the action
  - What do we want to do if it matches this rule?
  - List of actions shown on next slide

Snort Actions*

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.

*Some actions are only possible if IDS is inline

Snort Rule Example

alert tcp $EXTERNAL_NET any -> $SQL_SERVERS $ORACLE_PORTS (msg: “ORACLE create database attempt”; flow: to_server, established; content: “create database”; nocase; classtype: protocol-command-decode; sid:100001, rev: 2)

- tcp is the protocol

Snort Rule Example

alert tcp $EXTERNAL_NET any -> $SQL_SERVERS $ORACLE_PORTS (msg: “ORACLE create database attempt”; flow: to_server, established; content: “create database”; nocase; classtype: protocol-command-decode; sid:100001, rev: 2)

- $EXTERNAL_NET is a variable, defined elsewhere
  - In Snort, you define variables just like this
  - This is likely all possible non-internal IP addresses

Snort Rule Example

alert tcp $EXTERNAL_NET any -> $SQL_SERVERS $ORACLE_PORTS (msg: “ORACLE create database attempt”; flow: to_server, established; content: “create database”; nocase; classtype: protocol-command-decode; sid:100001, rev: 2)

- any is a Snort keyword: activates on any source port

Snort Rule Example

alert tcp $EXTERNAL_NET any -> $SQL_SERVERS $ORACLE_PORTS (msg: “ORACLE create database attempt”; flow: to_server, established; content: “create database”; nocase; classtype: protocol-command-decode; sid:100001, rev: 2)

- -> means that the traffic we’re looking for is unidirectional
  - We don’t care if this traffic exists going the other way
  - Seems obvious here, but in other cases it matters more

Snort Rule Example

alert tcp $EXTERNAL_NET any -> $SQL_SERVERS $ORACLE_PORTS (msg: “ORACLE create database attempt”; flow: to_server, established; content: “create database”; nocase; classtype: protocol-command-decode; sid:100001, rev: 2)

- (...) defines the options for the Snort rule
  - These are all... Optional...
  - List of options shown on next page

Snort 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.

Snort Rule Example

alert tcp $EXTERNAL_NET any -> $SQL_SERVERS $ORACLE_PORTS (msg: “ORACLE create database attempt”; flow: to_server, established; content: “create database”; nocase; classtype: protocol-command-decode; sid:100001, rev: 2)

- msg is what the sysadmin sees when the message alerts
  - This is the most important field of a Snort rule!!!
  - Always look at the msg field first when analyzing
  - I’ll never try and trick you with a bad message field

Snort Rule Example

alert tcp $EXTERNAL_NET any -> $SQL_SERVERS $ORACLE_PORTS (msg: “ORACLE create database attempt”; flow: to_server, established; content: “create database”; nocase; classtype: protocol-command-decode; sid:100001, rev: 2)

- flow: The message stream must be established and going to the server
  - Snort did a good job of making these rules sound like what they are!

Snort Rule Example

alert tcp $EXTERNAL_NET any -> $SQL_SERVERS $ORACLE_PORTS (msg: “ORACLE create database attempt”; flow: to_server, established; content: “create database”; nocase; classtype: protocol-command-decode; sid:100001, rev: 2)

- content: the packet must contain the text, “create database”
  - Not possible if the connection is encrypted!
  - This is one of the most helpful fields
  - Can use regex to create more helpful searches

Snort Rule Example

alert tcp $EXTERNAL_NET any -> $SQL_SERVERS $ORACLE_PORTS (msg: “ORACLE create database attempt”; flow: to_server, established; content: “create database”; nocase; classtype: protocol-command-decode; sid:100001, rev: 2)

- nocase: the case of the text isn’t important
  - Straight forward again

Snort Rule Example

alert tcp $EXTERNAL_NET any -> $SQL_SERVERS $ORACLE_PORTS (msg: “ORACLE create database attempt”; flow: to_server, established; content: “create database”; nocase; classtype: protocol-command-decode; sid:100001, rev: 2)

- Classtype, sid, and rev are just for organizational purposes. Don’t affect filtering at all
  - sid: Snort id: Unique id for Snort rule
  - rev: revision id: Shows what version of the rule you’re using

Snort Rule Example

alert tcp $EXTERNAL_NET any -> $SQL_SERVERS $ORACLE_PORTS (msg: “ORACLE create database attempt”; flow: to_server, established; content: “create database”; nocase; classtype: protocol-command-decode; sid:100001, rev: 2)

- Classtype, sid, and rev are just for organizational purposes. Don’t affect filtering at all
  - sid: Snort id: Unique id for Snort rule
  - rev: revision id: Shows what version of the rule you’re using

Intrusion Protection System (IPS)

- An extension to IDS: in addition to logging and alerting, you can drop packets
- Must be inline with the network
  - IPS must regulate traffic, not simply listen
  - Why?
- Considered a more advanced, flexible firewall
  - At least, it started out this way. However, firewalls have advanced over the years, that they also do DPI, and are considered to be very similar to IPS
  - Differences can be pedantic

Honeypots

- A honeypot is a system whose value lies in being attacked
- Another form of IDS
- Idea: Put a valuable-looking computer on your network
  - It isn't a real service, so all traffic is illegitimate
- Honeypots can be internal or external facing
  - **Internal**: Detect intrusions and attackers trying to move laterally through your network
  - External: Typically used for research. Wait for SSH scanners to try to connect, then let them and observe the attack

Honeypot Internal And External Use

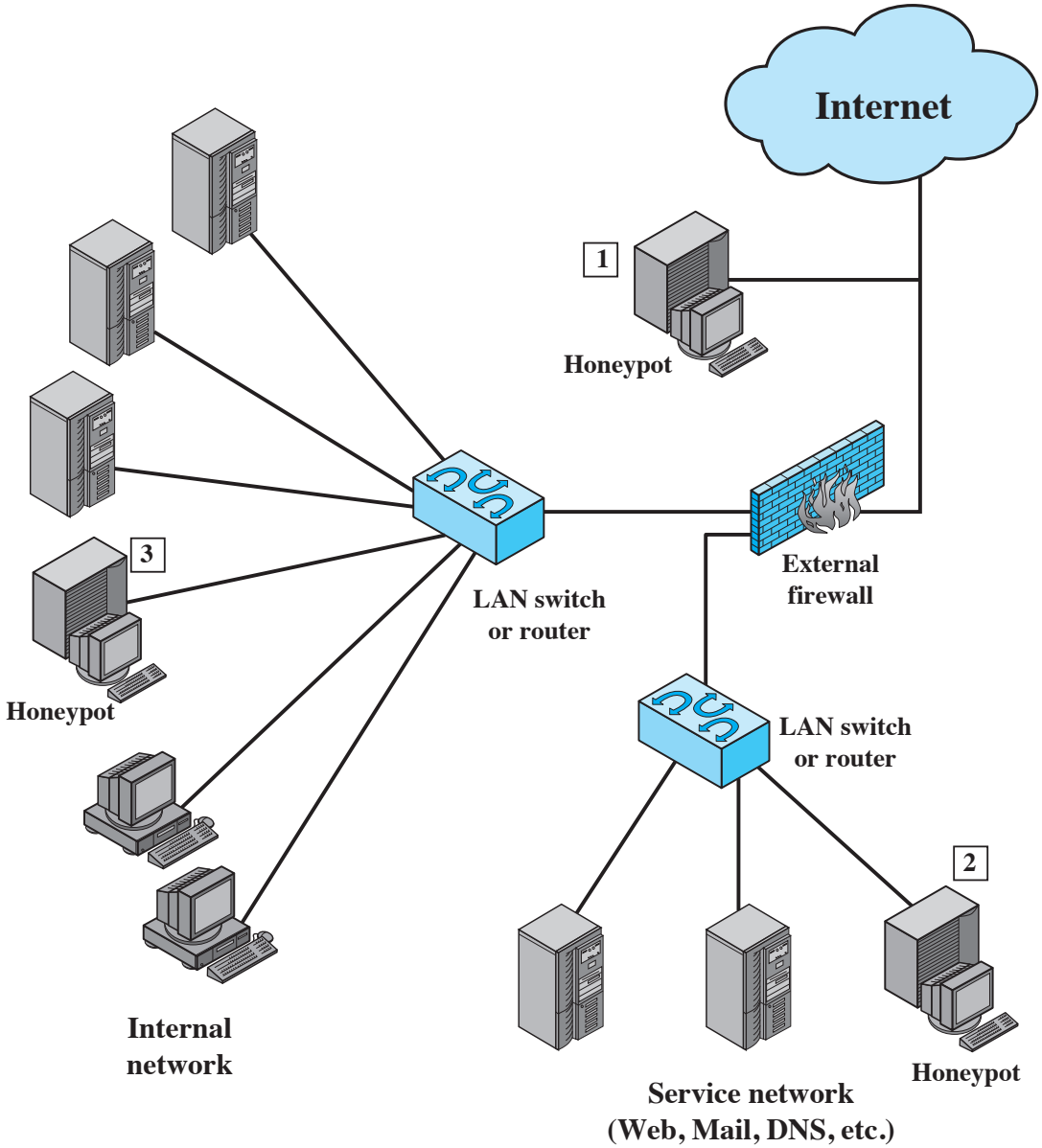

Figure 8.8 Example of Honeypot Deployment

Honeypot Classifications

- Low interaction honeypot
  - Consists of a software package that emulates particular IT services or systems decently well
  - 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
  - Takes longer to realize it isn't real
  - Requires significantly more resources
  - If compromised, could be used to initiate attacks on other systems