Intrusion Detection

Machine Learning
in Real World Applications

Intrusion Detection Datasets

- The intrusion detector learning task is to build a predictive model (i.e. a classifier) capable of distinguishing between “bad" connections, called intrusions or attacks, and “good" normal connections.
- Intrusion Detection Datasets need to **tag patterns of activity**, **not individual instances** (like malware /spam/phishing/uploads)
- Intrusion Detection Systems are designed to monitor these patterns using the limited information available in the network packet

IDS: Intrusion Detection System

- Gathers and analyzes information from within a computer or a network to identify the possible violations of security policy, including unauthorized access and misuse
  - The ultimate aim is to catch perpetrators before they do real damage to your resources
  - Based on your security policy and administrator intuition and experience
- Uses:
  - Rogue system detection.
  - Reconnaissance identification.
  - Attack pattern identification.

Intrusion Lifecycle

Phase	Technique	Description
1	**Reconnaissance**	• Gather as much info about targets as possible. • Required to craft an attack.
2	**Initial exploitation**	• Gain access to network or hosts, obtain credentials, etc.
3	**Privilege escalation**	• Gain greater control over systems. • Can do more damage with higher privileges.
4	**Pivoting**	• Compromise a central host. • Spread to other hosts and network segments.
5	**Persistence**	• Maintaining access is an important goal. • Avoiding discovery, erasing traces of activity

But Wait! …

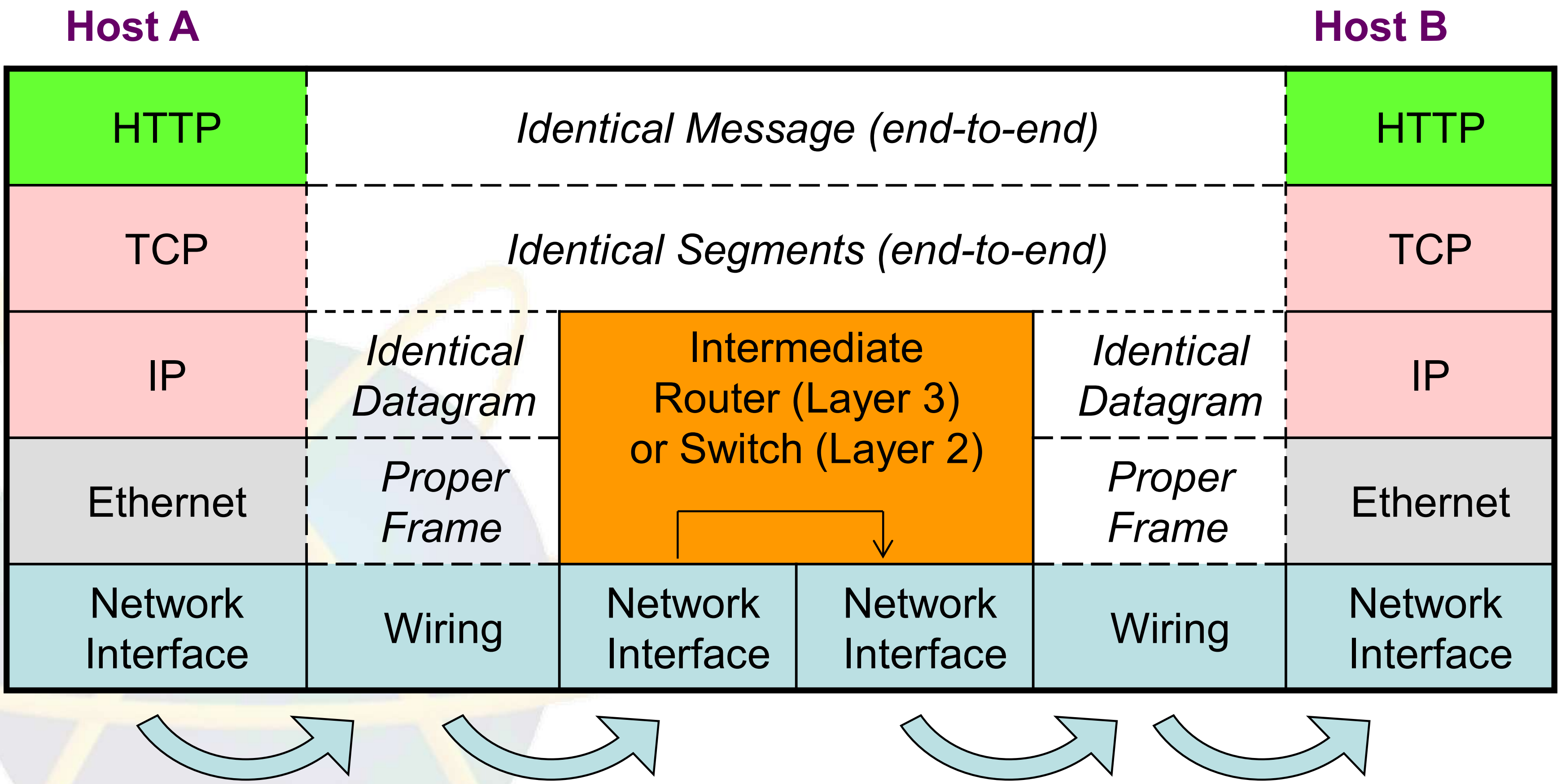
First, some basics of Internet Data Communications

Encapsulation

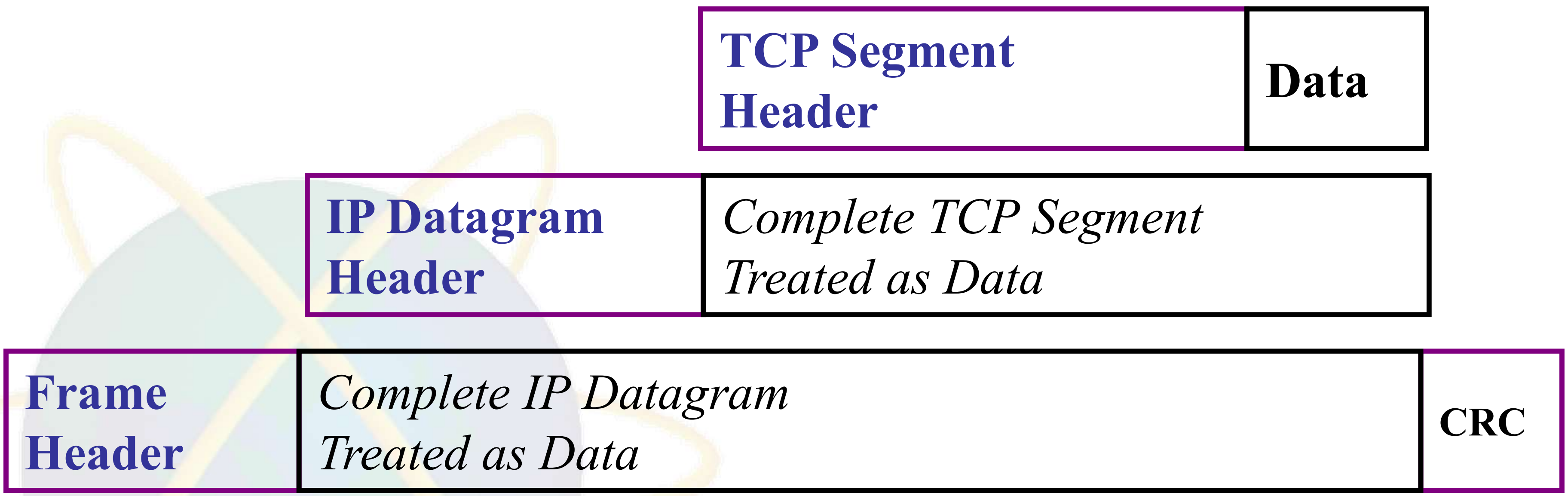
- A packet is a structured message.
- The control information of a given protocol must be treated strictly as data by the next "lower" protocol.
- As a packet moves *down* the protocol stack, it gets bigger as information relevant to the layer is added to the beginning and the end.
- Any given layer is allowed to work only with the data relevant to that layer, and nobody else's.
- As a packet moves up the stack it gets smaller, as the information from the current level is removed.

Each layer looks only at its part of the packet

Only the Ethernet and IP layers are changed by the network



Ethernet, IP, and TCP



Remember, this is really just a stream of bits
0011110101010101110000101010101010001010110101001001010100101110010100

Ethernet Frame Format

Preamble (64 bits)
Destination Address (48 bits)
Source Address (48 bits)
Packet type (16 bits)
Data (368-12,000 bits)
CRC (32 bits)

Key Fields

•*Preamble:* Alternating *1's* and *0's* to help receiving nodes synchronise

•*Address:* Unique identifier assigned by the hardware manufacturer **(MAC Address)**

•*Packet Type:* identifies this as an Ethernet frame (allows mutiple protocols and versions)

•*CRC:* Error detection (Cyclic Redundancy Check)

Remember, this is really just a stream of bits
00111101010101011100001010101010100010101101010010010101001011100101000

Datagram Format

Each row represents 4 octets (32 bits)

Version - Length - QOS - Total Length
Unique ID - Flags - Fragment Offset
Time to Live - Protocol - Checksum
Source IP Address
Destination IP Address
Options - Padding
Data *(up to 4416 bits)*

Key Fields

- IP is version 4 or 6
- QOS requests priority
- Second Row controls Fragmentation (e.g., "2 of 4")
- Gateways decrement **TTL** and discard the datagram if zero
- Protocol is analogous to Ethernet Type, Header Checksum to CRC
- Options are included for network testing (not required)

Remember, this is really just a stream of bits

0011110101010101110000101010101010001010110101001001010100101110010100

TCP Segment Format

Each row represents 4 octets (32 bits)

Source Port - Destination Port
Sequence Number
Acknowledgement Number
Offset - Code - Window
Checksum - Urgent
Options - Padding
Data *(up to 4224 bits)*

Key Fields

- Port number specifies service
- Sequence is position in sender's byte stream
- Acknowledgement of position in sender's byte stream
- Some segments carry only ACK, others carry data, and others a request to establish or close a connection (Code)
- Window and Options negotiate maximum segment size

Remember, this is really just a stream of bits
0011110101010101110000101010101010001010110101001001010100101110010100

Ethernet, IP, and TCP

Is there a service on this port? **Yes: Pass up the DATA part** No: discard it

TCP Segment Header	Data

Is this my IP address? **Yes: Pass up the DATA part** No: discard it

Am I a Router? **Yes: Pass the whole packet to all interfaces** No: ~

IP Datagram Header	*Complete TCP Segment Treated as Data*

Is this my MAC address? **Yes: Pass up the DATA part** No: discard it

Frame Header	*Complete IP Datagram Treated as Data*	CRC

Is this an ethernet frame? **Yes: Pass it up** No: discard it

Data Link Layer: stream of bits
0011110101010101110000101010101010001010110101001001010100101110010100

Ports and Port Ranges

Port: The endpoint of a logical network connection.

- Client computers connect to server programs through a designated port.
- Port is a "Layer 4" concept – TCP header
- All ports assigned are between the numbers 0 and 65535.

Port	Service	Secure	Port
20	Telnet	SSH	**22**
25	SMTP	SMTPS	**465**
80	HTTP	HTTPS	**443**
143	IMAP	IMAPS	**993**
389	LDAP	LDAPS	**636**

Port	Service
53	DNS
67	DHCP (server)
68	DHCP (client)
587	Submission

Network IDS Techniques

Monitoring System	Description
Signature-based	• Uses predefined set of rules to identify unacceptable events. • Events have specific and known characteristics.
Anomaly-based	• Uses a a preconfigured baseline of acceptable events • Identifies events that don't follow these patterns.
Behavior-based	• Records patterns of actions by an entity being monitored. • Detects if future behavior deviates from the norm.

https://www.snort.org/

- Snort is an open-source network Intrusion Detection System, capable of performing real-time traffic analysis and packet logging on IP networks.
- Snort is flexible, lightweight, and popular. It can perform protocol analysis, content searching/matching, and take actin on the result.
- Snort uses a rule-based language that detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, etc.

https://www.snort.org/faq/

- **Exploits** are the methodologies or techniques that are utilized to take advantage of vulnerabilities.
- A **signature** is defined as a set of distinctive marks or characteristics present in a exploit.
  - Includes ego strings, fixed offsets, debugging information, or any other unique marking that may or may not be related to actually exploiting a vulnerability.
- This type of detection is typically classified as *day after detection*, since actual public exploits are necessary for this to work.
  - Anti-Virus companies commonly use this technique for protecting their customers from virus outbreaks.

https://www.snort.org/faq/what-are-community-rules

- Unlike signatures, **Rules** are based on detecting the actual vulnerability, not an exploit or a unique piece of data.
  – Rules can be based on signatures, but developing a rule requires an understanding of how the vulnerability actually works.
- Community rules have been submitted by members of the open source community or Snort Integrators. These rules are freely available to all Snort users.
  – To contribute, just send your rules along with and packet captures of the data to the Snort-sigs mailing list.
- The Community Ruleset is updated daily and is a subset of the subscriber ruleset.

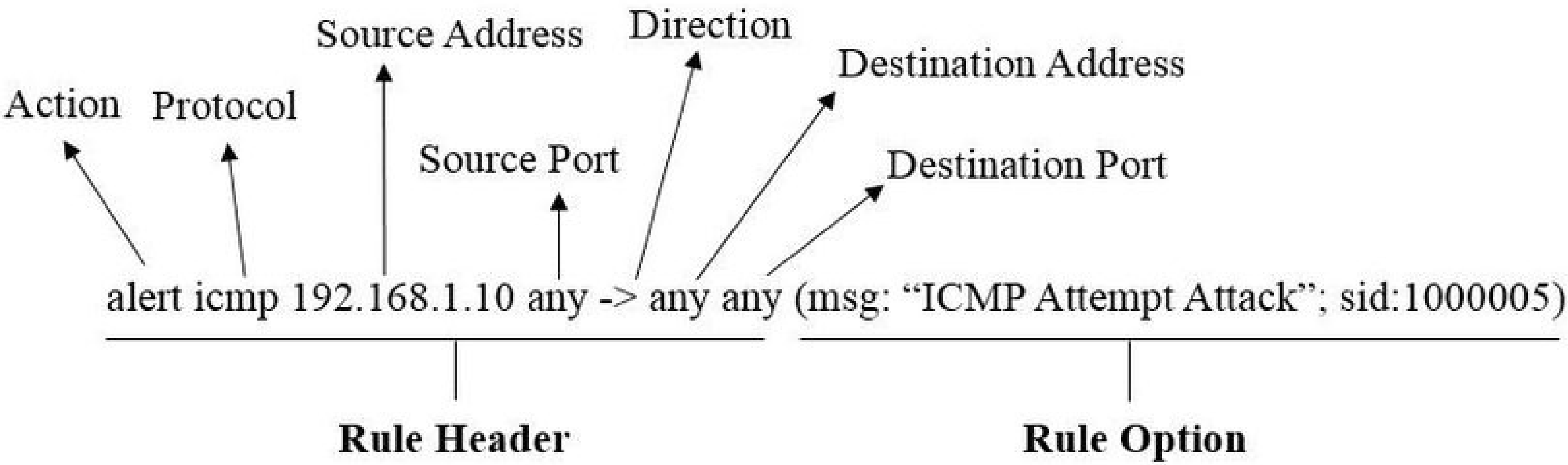
Snort Rules

The Snort rule description language is simple, flexible and quite powerful.

Snort rules are divided into two logical sections:

1. **The Rule Header** contains the action, protocol, source and destination IP addresses, and the source and destination ports.
2. **The Rule Options** contain alert messages and information on which parts of the packet should be inspected to determine if the rule action should be triggered.

Example Rule



alert tcp !192.168.1.0/24 any -> 192.168.1.0/24 111 (content: "|00 01 86 a5|"; msg: "external mountd access";)

This rule matches "any tcp packet with a source IP address not originating from the internal network and a destination address on the internal network".

The options indicate "if the payload contains this hexidecimal sequence, print this message".

Activate/Dynamic Rules

- **Activate/dynamic rule pairs** give Snort a powerful capability.
- Activate rules act just like alert rules, except they have a *required* option field: "activates".
- Dynamic rules act just like log rules, but they have a different option field: "activated_by". Dynamic rules also have a second required field, "count".

Activate/dynamic rule pairs and options make it possible to track a *series* of network packets that make up an exploit


Q&A

Intrusion Detection Datasets

- The intrusion detector learning task is to build a predictive model (i.e. a classifier) capable of distinguishing between “bad" connections, called intrusions or attacks, and “good" normal connections.
- Intrusion Detection Datasets need to **tag patterns of activity**, **not individual instances** (like malware /spam/phishing/uploads)
- Intrusion Detection Systems are designed to monitor these patterns using the limited information available in the network packet
- Tagged Intrusion Detection Datasets can be derived from IDS logfiles
- Snort is a good example - other tools are available ,,,

KDD99 Dataset

- A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol. Each connection record consists of about 100 bytes.
- Each connection is labeled as either **normal** or a specific **attack type**. The datasets contain a total of 24 training attack types, with an additional 14 types in the test data only.
- Attack types (exploits) fall into four **categories**:
  - **DOS**: denial-of-service, e.g. syn flood;
  - **Probe**: surveillance and other probing, e.g., port scanning;
  - **R2L**: unauthorized access from a remote machine, e.g. guessing password;
  - **U2R**: unauthorized access to local superuser (root) privileges, e.g., various “buffer overflow" attacks.

KDD99 Dataset

feature name	description	type
duration	length (number of seconds) of the connection	continuous
protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
service	network service on the destination, e.g., http, telnet, etc.	discrete
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0 otherwise	discrete
wrong_fragment	number of ``wrong'' fragments	continuous
urgent	number of urgent packets	continuous

Basic features of individual TCP connections.

KDD99 Dataset: Derived Features

- Stolfo et al. defined higher-level features that help to distinguish normal connections from attacks. There are several categories of derived features.
- The "same host" features examine only the connections in the past two seconds that have the same destination host as the current connection, and calculate statistics related to protocol behavior, service, etc.
- The "same service" features examine only the connections in the past two seconds that have the same service as the current connection
- ."Same host" and "same service" features are together called *time-based traffic features* of the connection records.

Cost-based Modeling and Evaluation for Data Mining With Application to Fraud and Intrusion Detection: Results from the JAM Project by Salvatore J. Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Philip K. Chan.

KDD99 Dataset

feature name	description	type
count	number of connections to the same host as the current connection in the past two seconds	continuous
	Note: The following features refer to these same-host connections.	
serror_rate	% of connections that have ``SYN'' errors	continuous
rerror_rate	% of connections that have ``REJ'' errors	continuous
same_srv_rate	% of connections to the same service	continuous
diff_srv_rate	% of connections to different services	continuous
srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
	Note: The following features refer to these same-service connections.	
srv_serror_rate	% of connections that have ``SYN'' errors	continuous
srv_rerror_rate	% of connections that have ``REJ'' errors	continuous
srv_diff_host_rate	% of connections to different hosts	continuous
Traffic features computed using a two-second time window.		

KDD99 Dataset: Derived Features

- Some probing attacks scan the hosts (or ports) using a much larger time interval than two seconds, for example once per minute. Therefore, connection records were also sorted by destination host, and features were constructed using a window of 100 connections to the same host instead of a time window.
- This yields a set of *host-based traffic features*.
  - *No separate table*

KDD99 Dataset: Derived Features

- Stolfo et al. used domain knowledge to add features that look for suspicious behavior in the data portions, such as the number of failed login attempts. These are called *content features*.
- DOS and probing attacks involve many connections to some host(s) in a very short period of time.
- R2L and U2R attacks are embedded in the data portions of packets, and normally involve only a single connection,
- So, there appear to be no sequential patterns that are frequent in records of R2L and U2R attacks.

KDD99 Dataset

feature name	description	type
hot	number of ``hot'' indicators	continuous
num_failed_logins	number of failed login attempts	continuous
logged_in	1 if successfully logged in; 0 otherwise	discrete
num_compromised	number of ``compromised'' conditions	continuous
root_shell	1 if root shell is obtained; 0 otherwise	discrete
su_attempted	1 if ``su root'' command attempted; 0 otherwise	discrete
num_root	number of ``root'' accesses	continuous
num_file_creations	number of file creation operations	continuous
num_shells	number of shell prompts	continuous
num_access_files	number of operations on access control files	continuous
num_outbound_cmds	number of outbound commands in an ftp session	continuous
is_hot_login	1 if the login belongs to the ``hot'' list; 0 otherwise	discrete
is_guest_login	1 if the login is a ``guest''login; 0 otherwise	discrete
Content features within a connection suggested by domain knowledge.