Practical Network Defense

Master's degree in Cybersecurity 2020-21

Intrusion Detection Systems: Snort/Suricata, fail2ban

Angelo Spognardi
spognardi@di.uniroma1.it
Dipartimento di Informatica
Sapienza Università di Roma

Host IDS/IPS

- Many host security products have integrated HIPS, anti-malware and firewall
  - Protects mobile hosts from attack when attached outside the protected network
  - Protects against local attacks from a user, and codes/scripts from removable devices
  - Protects against attacks from the same subnet/VLAN
  - Protects against encrypted attacks where the encrypted data stream terminates at the host being protected
  - Inspect packet content after decrypting received VPN or SSL packets
  - Inspect packet together with anti-malware software by decrypting or emulating malware
- Con: if an attacker takes over a host, then one can tamper with IDS/agent binaries and modify audit logs
- Con: only local view of the attack
- Con: Host-based anomaly detection has high false alarm rate

Network IDS/IPS

- Deploying sensors at strategic locations with a central monitor
  - Inspecting network traffic
    - Watch for violations of protocols and unusual connection patterns
  - Protect network equipment, such as printers that do not have HIDS
  - Protect against network-oriented attacks
    - DDoS, bandwidth consumption
    - Independent of host OS
  - Monitoring user activities
    - Look into the data portions of the packets for malicious command sequences
- Con: may not detect encrypted traffic
  - Data portions and some header information can be encrypted
- Con: can not detect some attacks in the host
- Con: high requirement for computation capability of IDS/IPS

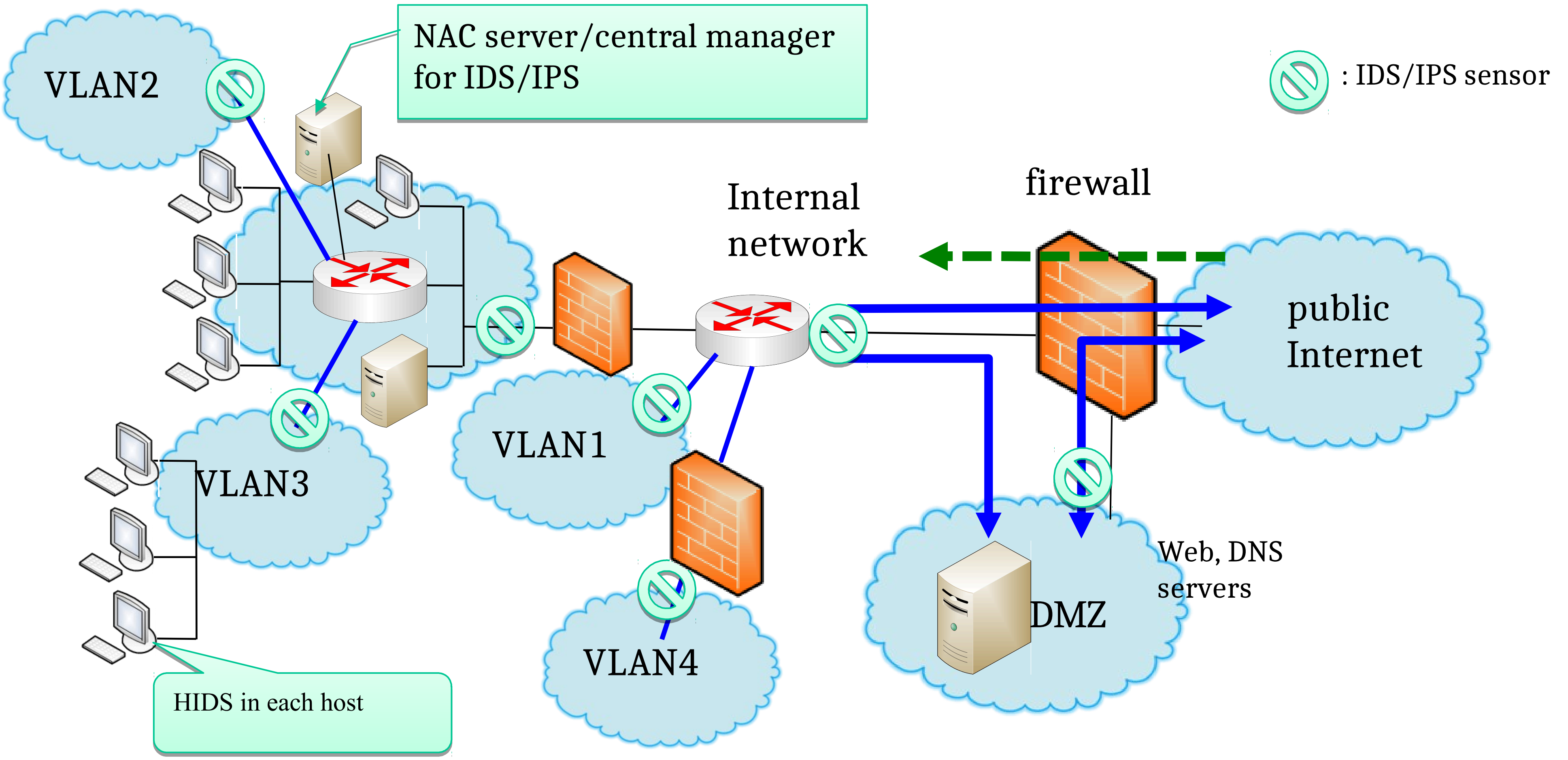
Then: combine host and network IDS/IPS

- Both HIDS and NIDS technologies are not equally adept at detecting and blocking certain attacks
- There are attacks that can only be detected by HIDS
  - E.g., local privilege escalation, metamorphic malware
- Attacks that can only be detected by NIDS
  - E.g.. Routing advertisement injection
- Integrating the strengths of both architectures provides a solution whose sum is greater than its parts
- More accurate result for quarantining a host or block/filter traffic
- The basis for NAC (Network Access Control) products

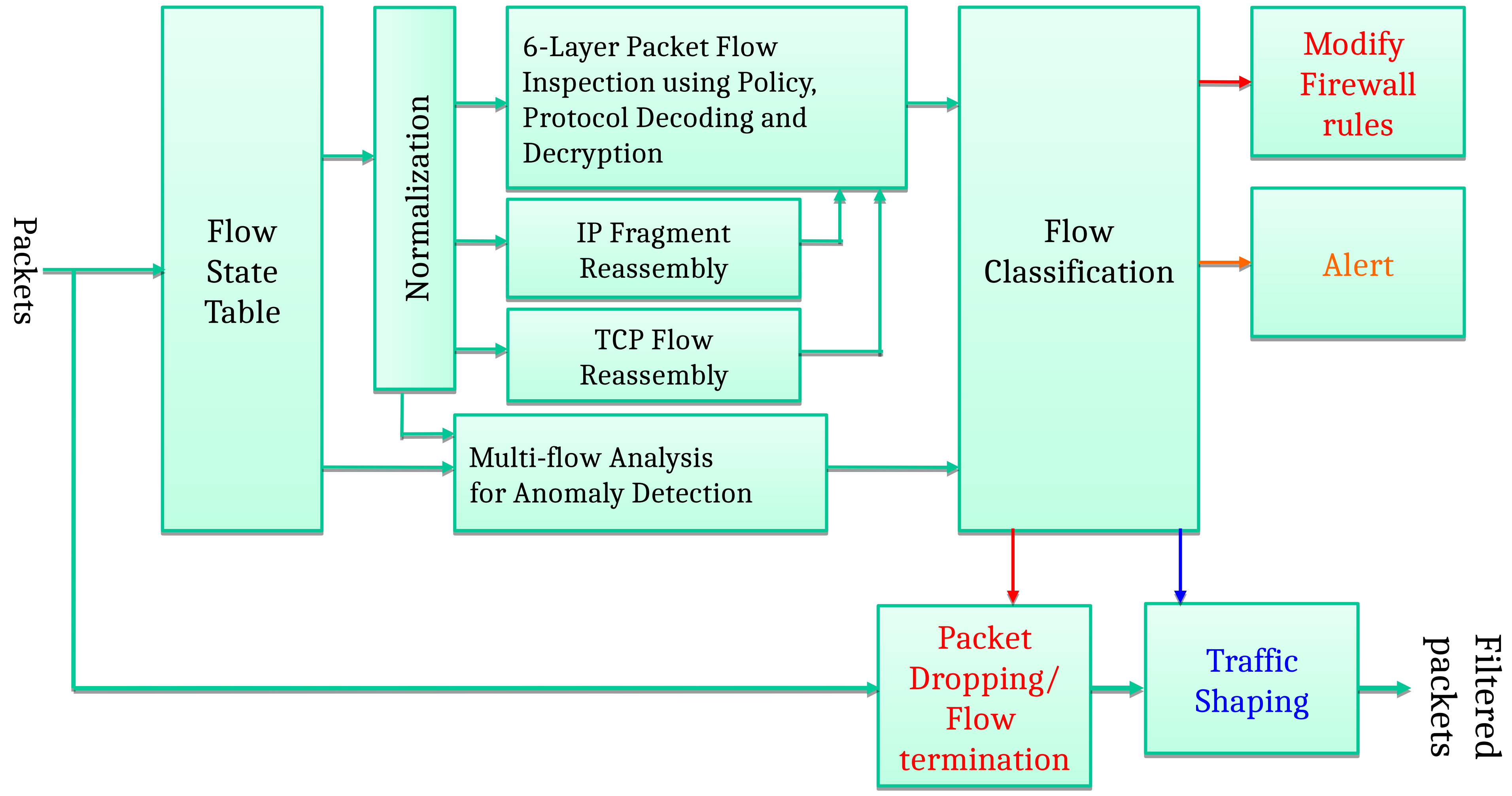
Distributed IDS

- Extend focus from single systems to information infrastructure
  - More effective defense has these working together to detect intrusions
  - Agent-based coordination between host and NAC server
- Monitoring and correlating public, internal VLANs, and DMZ segments of the IDS/IPS sensors and firewalls
- Correlation among these segments to yield an accurate picture of network attacks that were either blocked or made it into the internal network
- Correlate HIDS and NIDS for constant monitoring and blocking in NAC (central control)
- Exchange Format Working Group (IDWG) of the IETF
- Intrusion Detection Exchange Protocol (IDXP): RFC 4767
  - An application-level protocol for exchanging data between IDS's
  - IDXP supports mutual-authentication, integrity, and confidentiality over a connection-oriented protocol
  - The protocol provides for the exchange of the Intrusion Detection Message Exchange Format (IDMEF) messages in implementations of the data model in the Extensible Markup Language (XML)
  - The IDMEF message elements are described in RFC 4765, and developed by the Intrusion Detection

Distributed intrusion detection



Network-based IPS block diagram



State information and analysis

- State information for a session (flow):
  - Maintaining state information enables sensors to gain context for attack detection
  - Inspecting the entire content of the data packet
- State information is captured and updated in real time
- State information is the basis for Layer 2-7 detection
  - Utilize multiple token matches to capture attack signatures/behaviors that span packet boundaries or are out-of-order in a packet stream
  - Detect/block malware, Trojans, key loggers, P2P, botnets, worms
- Appropriate use of the state information is the key to detection
  - Accuracy depends on the selection of parameters and their transitions

Normalization

TCP normalization

- Inspect invalid or suspect conditions
  - E.g., a SYN sent to the client from the server or a SYNACK sent to the server from the client
- Block certain types of network attacks
  - E.g., insertion attacks and evasion attacks
    - Insertion attacks occur when the inspection module accepts a packet that the end system rejects
    - Evasion attacks occur when the inspection module rejects a packet while the end system accepts it
- Discards segments containing
  - Bad segment checksum
  - Bad TCP header or payload length
  - Suspect TCP flags (for example, NULL, SYN/FIN, or FIN/URG)
- To configure TCP normalization
  - Assemble various TCP commands into a parameter map for filtering as policy
  - E.g., parameter map contains ranges for MSS, # of SYN retries, # of out of order segments, control of timeout, random sequence number, Window scale factor, urgent flag, etc

IP normalization

- Inspect IP packets using configured parameter map for:
  - General security checks
  - ICMP security checks
  - Fragmentation security checks
  - IP fragment reassembly
  - IP fragmentation if a packet exceeds the outbound maximum transmission unit (MTU)
- Configure IP normalization parameter map
  - ToS
  - TTL
  - Unicast reverse path
  - Fragment reassembly
  - Maxiumum # of fragments
  - MTU

Actions from IPS

- Packet dropping
- Session termination
- Firewall rules modification for blocking suspicious hosts
- Traffic shaping for slowing down less critical traffic such as P2P, video
- Alerts generation
- Log generation

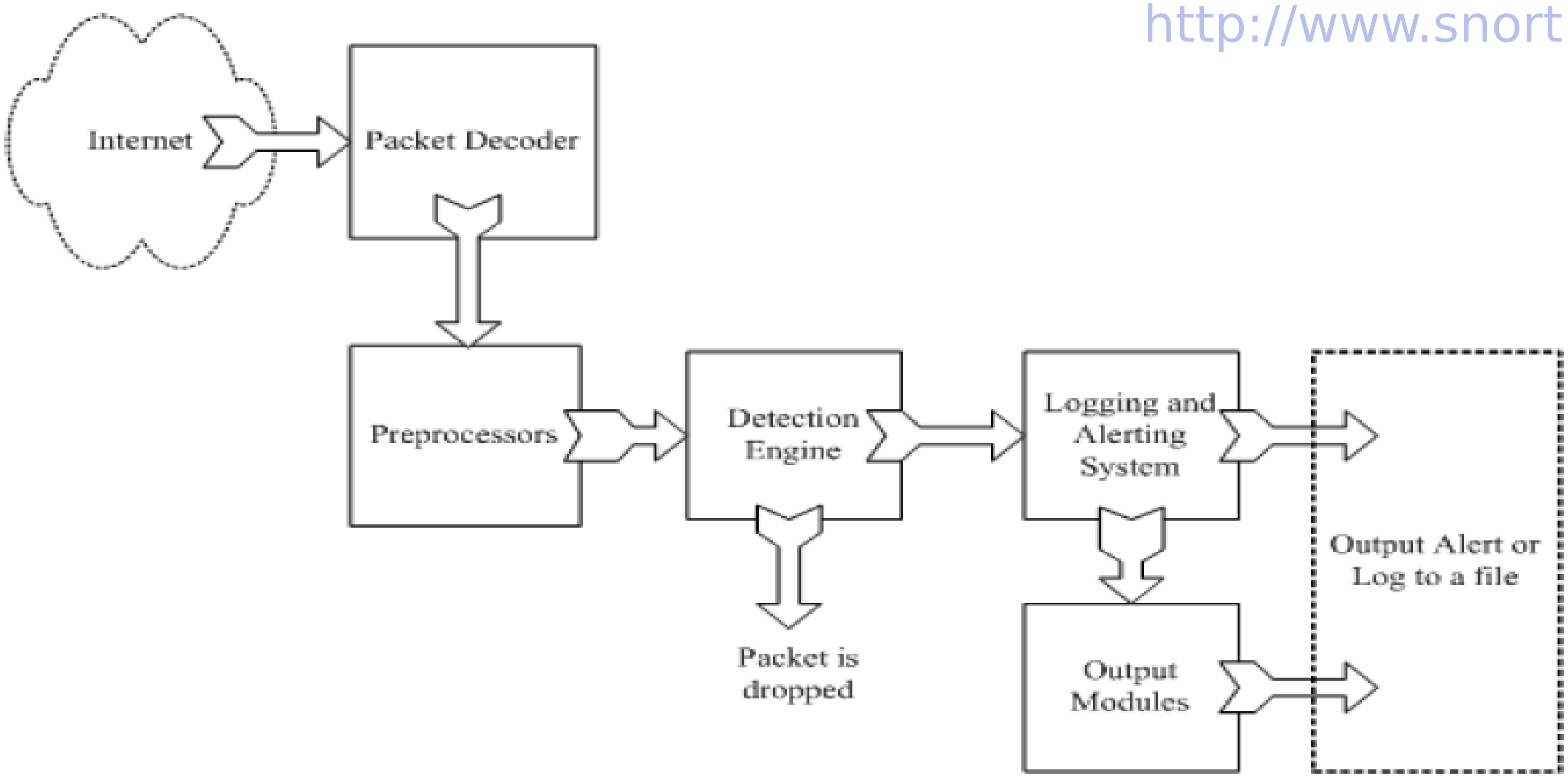
Snort and Suricata

Snort: open source NIDS/NIPS

- Written by Martin Roesch
  - Now developed by Sourcefire, of which Roesch is the founder and CTO
- The most widely deployed intrusion detection and prevention technology worldwide
- Using a rule-driven language
- Combining the benefits of signature, protocol and anomaly based inspection methods.
  - Snort can be combined with other software such as SnortSnarf, sguil, OSSIM, and the Basic Analysis and Security Engine (BASE) to provide a visual console
  - Emerging Threats: community maintained Snort rule sets are evolving
- Large rule sets for known vulnerabilities

Snort

http://www.snort.org/

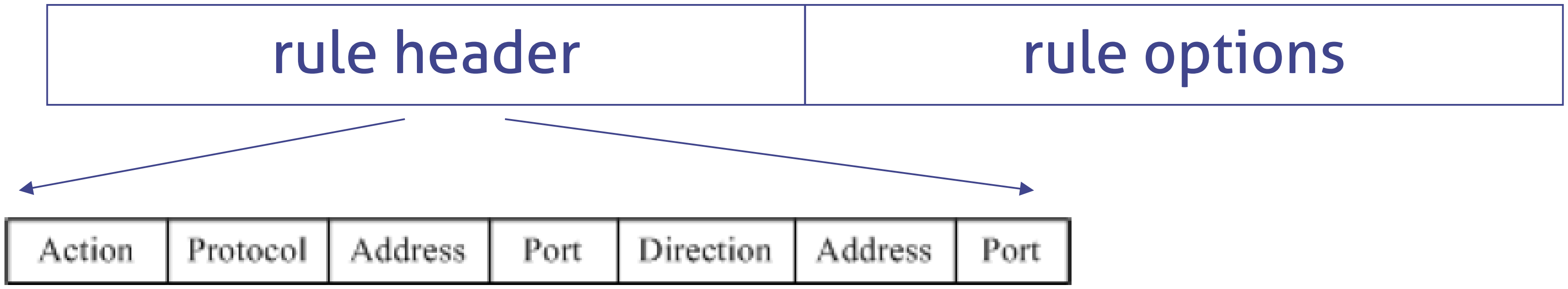


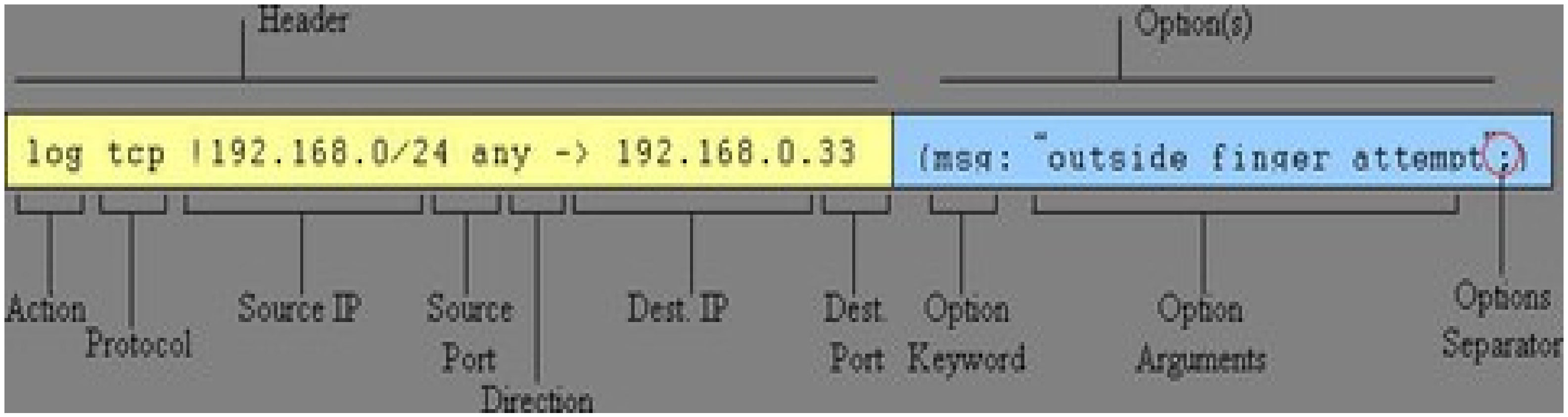
From: Rafeeq Ur Rehman, *Intrusion Detection Systems with Snort: Advanced IDS Techniques with Snort, Apache, MySQL, PHP, and ACID.*

Snort components

- Packet Decoder
  - input from Ethernet, SLIP, PPP...
- Preprocessors
  - detect anomalies in packet headers
  - packet defragmentation
  - decode HTTP URI
  - reassemble TCP streams
- Detection Engine: applies rules to packets
- Logging and Alerting System
- Output Modules: alerts, log, other output

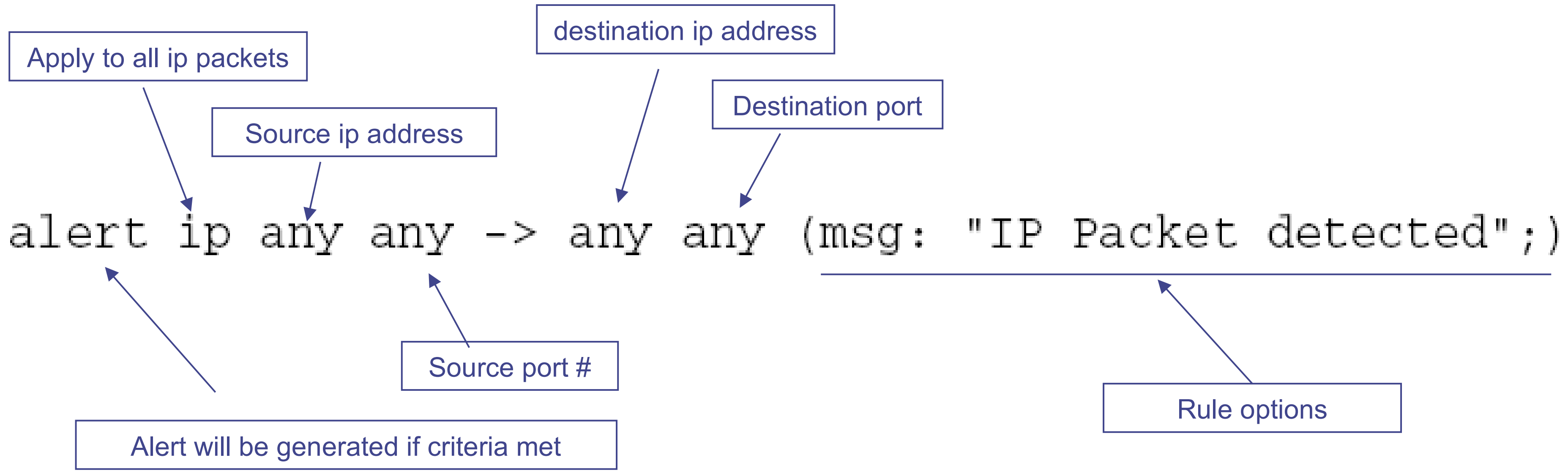
Snort detection rules





```
log tcp !192.168.0/24 any -> 192.168.0.33 (msg: "outside finger attempt";)
```

Example



```
alert ip any any -> any any (msg: "IP Packet detected";)
```

```
alert tcp $TELNET_SERVERS 23 -> $EXTERNAL_NET any (msg:"TELNET
   Attempted SU from wrong group"; flow:
from_server,established; content:"to su root"; nocase;
   classtype:attempted-admin; sid:715; rev:6;)
```

TCP/IP header rule options

`ipopts: opt`	Match specific IP option opt
`flags: fff`	Match settings of one or more TCP flags
`seq: nnn`	Match specific TCP sequence number
`ack: nnn`	Match specific TCP ack number
`window: nnn`	Match specific TCP window size
`itype: nnn`	Match ICMP type field value (or range)
`icode: nnn`	Match ICMP code field value (or range)
`fragbits: mmm`	Match IP fragmentation/reserved header bits
`ip_proto: proto`	Match IP Protocol field by number or name
`id: nnn`	Match value of IP ID header field
`ttl: nnn`	Match value of IP TTL header field
`dsize: nnn`	Match packet payload size (or size range)
`flow: flowstat`	Match flow direction/state
`rpc: app,ver,proc`	Match RPC application, version and procedure

Payload checking rule options

`content: "xxx"`	Match pattern "xxx" in packet payload
`offset: nnn`	Offset for start of search for content match
`depth: nnn`	Number of bytes to search for content match
`distance: nnn`	Offset for search relative to end of last match
`within: nnn`	No. of bytes to search rel. to end of last match
`nocase`	Ignore case when looking for matches
`isdataat: nnn`	Checks that data are present in byte onnn, possibly relative to previous match
`pcre: "/regex/mmm"`	Match pattern given by Perl regular expression regex with modifiers mmm
`uricontent: "sss"`	Match a onormalised URI, i.e. where hex codes, directory traversals etc. have been rationalised
`byte_jump: rules`	Gives rules for matching TLV-encoded protocols

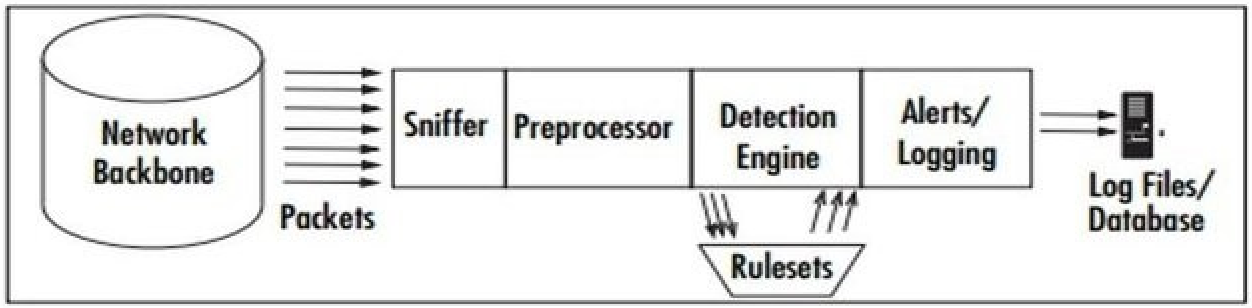
Snort challenges

- Misuse detection – avoid known intrusions
  - Database size continues to grow
    - Today Snort has > 3000 rules
  - Snort spends 80% of time doing string match
- Anomaly detection – identify new attacks
  - Probability of detection is low (very low…)

Suricata

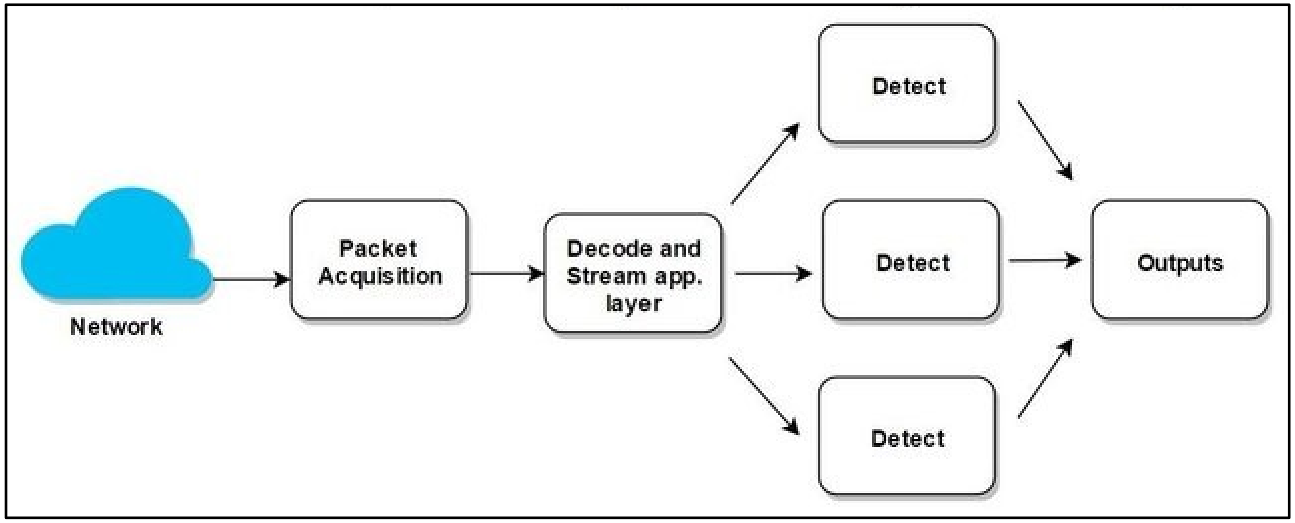
- High performance Network IDS, IPS and Network Security Monitoring engine
- Open source and owned by a community-run non-profit foundation, the Open Information Security Foundation (OISF)
- Real time intrusion detection (IDS), inline intrusion prevention (IPS), network security monitoring (NSM) and offline pcap processing
- Developed to overcome snort limitations
  - Financial help from the U.S. Department of Homeland Security

Suricata vs Snort architecture



Snort

Suricata



Suricata features

- Suricata's workload is distributed thanks to its multi-threading capabilities and GPU acceleration
- Support for packet decoding of: IPv4, IPv6, TCP, UDP, SCTP, ICMPv4, ICMPv6, GRE, Ethernet, PPP, PPPoE, Raw, SLL, VLAN, QINQ, MPLS, ERSPAN
- App layer decoding of: HTTP, SSL, TLS, SMB, DCERPC, SMTP, FTP, SSH, DNS, Modbus, ENIP/CIP, DNP3, NFS, NTP, DHCP, TFTP, KRB5, IKEv2
- IP reputation
- Integration with other solutions, such as SIEMs and databases using YAML and JSON files as inputs and outputs
- Incorporates the Lua scripting language to create rules that identify conditions that would be difficult or impossible with a legacy Snort Rule
- A lot more: https://suricata-ids.org/features/all-features/

Worth mentioning Zeek (old name: Bro)

- Older than snort (1998 vs. 1995)
- Support and attention following a grant from the National Science Foundation in 2010
- Zeek uses Bro Script (similar to C++) rather than a rules structure to define network traffic
- It can be used as IDS but also as a network monitor
  - However, the deep-packet inspection aspect of Zeek makes it far more resource intensive for basic alerts
  - Setup and maintenance of Zeek can be challenging at best for even experienced users

Bro script example: Matching URLs

- Report all Web requests for files called “passwd”

```
event http_request(c: connection,             # Connection.
                   method: string,            # HTTP method.
                   original_URI: string,      # Requested URL.
                   unescaped_URI: string,     # Decoded URL.
                   version: string)           # HTTP version.
{
    if ( method == "GET" && unescaped_URI == /.*passwd/ )
        NOTICE(...); # Alarm.
}
```

Bro script example: Scan Detection

- Count failed connection attempts per source address

```
global attempts: table[addr] of count &default=0;

event connection_rejected(c: connection)
{
    local source = c$id$orig_h;       # Get source address.

    local n = ++attempts[source];     # Increase counter.

    if ( n == SOME_THRESHOLD )        # Check for threshold.
        NOTICE(...);                  # Alarm.
}
```

fail2ban

A small break…

Student's Opinions Questionnaires (OPIS)

- Two options:
  - the infostud app (probably best option)
  - the infostud website
    - follow the following instructions https://www.uniroma1.it/sites/default/files/field_file_allegati/vadevecum_opis_eng_27_11_2018_002_modalita_compatibilita.pdf
    - use this course code **CITY115P**
- Be positive!

Fail2ban

- Intrusion prevention software framework that protects computer servers from brute-force attacks
  - https://www.fail2ban.org/wiki/index.php/Main_Page
- Fail2ban scans log files and bans IP addresses of hosts that have too many failures within a specified time window
- Think of it as a dynamic firewall: it detects incoming connection failures, and dynamically adds a firewall rule to block that host after too many failures

Fail2ban features

- client/server
- multi-threaded
- autodetection of datetime format
- lots of predefined support
  - services – sshd, apache, qmail, proftpd, sasl, asterisk, squid, vsftpd, assp, etc
  - actions – iptables, tcp-wrapper, shorewall, sendmail, ipfw, etc

Fail2ban limitations

- Reaction time – fail2ban is a **log parser**, so it cannot do anything before something is written to the log file.
- Syslog daemons normally buffer output, so you may want to disable buffering in your syslog daemon
- fail2ban waits 1 second before checking log files for changes, so it is possible to get more failures than specified by `maxretry`
- A local user could initiate a DoS attack by forging syslog entries with the logger(1) command

Terminology

- fail2ban
  - Software that bans & unbans IP addresses after a defined number of failures
- (un)ban
  - (Remove)/Add a firewall rule to (un)block an IP address
- jail
  - A jail is the definition of one fail2ban-server thread that watches one or more log file(s), using one filter and can perform one or more actions
- filter
  - Regular expression(s) applied to entries in the jail's log file(s) trying to find pattern matches identifying brute-force break-in attempts
- action
  - One or more commands executed when the outcome of the filter process is true AND the criteria in the fail2ban and jail configuration files are satisfied to perform a ban

Fail2ban components

- Content of /etc/fail2ban/ directory
- fail2ban-server
  - The core of the IPS
- fail2ban-client
  - The cli interface with the server
- fail2ban-regex
  - A cli utility to test regular expressions and filters

fail2ban activity

Activity 1

- In the topology of ACME, configure the webserver to ban/unban hosts bruteforcing SSH access using fail2ban

Activity 2

- Configure fail2ban in the syslog server to ban/unban hosts that are bruteforcing other hosts
- It has to interact with the two firewalls to add the banned hosts in an alias list (like fail2ban)
  - HINT: you can use shell scripts and SSH keys, with the pfctl command
    - https://www.openbsd.org/faq/pf/tables.html
    - Beware that pfctl needs administrator rights, then be sure to adequately protect your script!

That's all for today

- **Questions?**
- See you next lecture!
- References:
  - NIST Guide to Intrusion Detection and Prevention Systems (IDPS)
  - Chapter 19 textbook
- Other interesting readings:
  - A Framework for Constructing Features and Models for Intrusion Detection Systems
  - Specification-based anomaly detection: a new approach for detecting network intrusions