

Snort Intrusion Detection System (IDS)

An Overleaf-ready Guide

Generated for Overleaf

October 14, 2025

Abstract

This document provides a practical guide to Snort, the open-source network intrusion detection and prevention system (NIDS/NIPS). It covers architecture, installation, configuration, rule syntax, common deployment scenarios, examples, testing, and troubleshooting. Code snippets and example Snort rules are included and formatted for easy copy/paste into Overleaf or a lab environment.

Contents

1	Introduction	3
1.1	Key features	3
2	Architecture and Components	3
2.1	Flow diagram	3
3	Installation (Linux example)	4
3.1	Prerequisites	4
3.2	Typical build-from-source steps	4
4	Configuration: <code>snort.conf</code>	4
4.1	Structure of <code>snort.conf</code>	4
4.2	Example snippets	5
5	Snort Rule Syntax	5
5.1	Rule header	5
5.2	Rule options	5
5.3	Example rule	5
5.4	Common options explained	5
6	Preprocessors	6
7	Output and Logging	6
8	Snort in Inline (IPS) Mode	6

9 Rule Writing Best Practices	6
10 Examples: Practical Rules	7
10.1 Detecting a simple web shell signature	7
10.2 Detecting SMB (EternalBlue-like) probe (example signature)	7
11 Testing and Validation	7
11.1 Running Snort in test mode	7
11.2 Generating test traffic	7
12 Performance Tuning	7
13 Deployment Scenarios	7
13.1 Tap/monitoring mode	7
13.2 Inline/IPS mode	8
13.3 Distributed sensor deployment	8
14 Integrations	8
15 Troubleshooting	8
16 Appendix: Useful Commands	8
17 Appendix: Sample local.rules	8
18 References and Further Reading	9

1 Introduction

Snort is a widely used open-source network intrusion detection system (NIDS) and intrusion prevention system (IPS). It performs real-time traffic analysis and packet logging on IP networks and can detect a wide range of attacks and probes.

1.1 Key features

- Real-time traffic analysis and packet logging
- Rule-based detection engine
- Flexible logging options (unified2, syslog, pcap)
- Preprocessors to normalize/inspect traffic
- Inline mode for intrusion prevention

2 Architecture and Components

Snort's architecture consists of several main components:

1. Packet acquisition (libpcap)
2. Packet decoder
3. Preprocessors (HTTP, Frag3, Stream5, DNS, etc.)
4. Detection engine (rule matching)
5. Output modules (logging, alerting)

2.1 Flow diagram

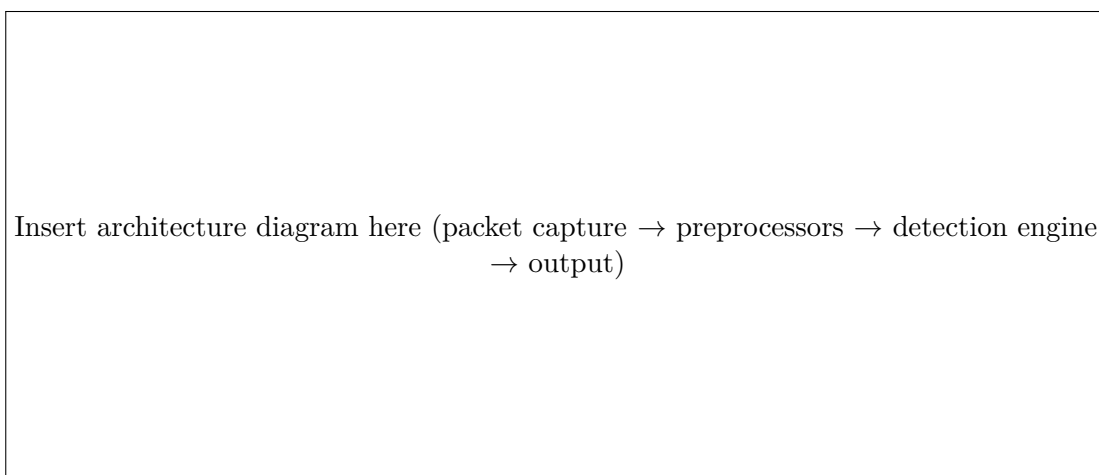


Figure 1: High-level Snort architecture

3 Installation (Linux example)

This section gives an example installation flow for a typical Linux environment (Ubuntu/Debian). Adjust package manager commands for CentOS/RHEL (yum/dnf).

3.1 Prerequisites

- libpcap development headers
- libpcre, libdnet (or libdnet-dev), liblzma (for certain features)
- libhwloc optional for performance

3.2 Typical build-from-source steps

Listing 1: Build and install Snort (example)

```
1
2 # update and install prerequisites (Debian/Ubuntu example)
3
4 sudo apt update
5 sudo apt install -y build-essential libpcap-dev libpcre3-dev libdumbnet
   -dev bison flex zlib1g-dev liblzma-dev openssl libssl-dev
6
7 # download Snort (use latest stable tarball from snort.org)
8
9 wget [https://www.snort.org/downloads/snort/snort-](https://www.snort.
   org/downloads/snort/snort-)<version>.tar.gz
10 tar xzf snort-<version>.tar.gz
11 cd snort-<version>
12
13 ./configure --enable-sourcefire
14 make
15 sudo make install
16
17 # create snort user and directories
18
19 sudo mkdir -p /etc/snort/rules /var/log/snort /usr/local/lib/
   snort_dynamicrules
20 sudo useradd --no-create-home --shell /sbin/nologin snort
21 sudo chown -R snort:snort /var/log/snort
22
23 enable Snort as a service or run manually
24
25 d# Example run (promiscuous mode)
26 sudo snort -c /etc/snort/snort.conf -i eth0 -A console
```

4 Configuration: snort.conf

4.1 Structure of snort.conf

- Network variables ($HOME_{NET}$, $EXTERNAL_{NET}$)

- Preprocessor configuration
- Rule path and include statements
- Output modules and logging
- Performance tuning options

4.2 Example snippets

Network variables

```
1 var HOME_NET 192.168.1.0/24
2 var EXTERNAL_NET any
3 var DNS_SERVERS 192.168.1.10
4
5 auth_servers: 10.0.0.5
```

Including rules

```
1 include $RULE_PATH/local.rules
2 include $RULE_PATH/community.rules
```

5 Snort Rule Syntax

A Snort rule has two main parts: the rule header and the rule options.

5.1 Rule header

Header format: `action proto srcipsrcport -> dstipdstport`

`action:` alert, log, pass, activate, dynamic, drop, reject

`proto:` tcp, udp, icmp, ip

5.2 Rule options

Options are enclosed in parentheses and include metadata, payload checks, flow, content, pcre, etc.

5.3 Example rule

Listing 2: Simple Snort rule

```
1 alert tcp $EXTERNAL_NET any -> $HOME_NET 22 (msg:"SSH connection
  attempt"; flow:to_server,established; sid:1000001; rev:1;)
```

5.4 Common options explained

msg message to include with alert

sid unique rule identifier

rev	rule revision
content	payload string to search for
flow	TCP flow direction/state
pcre	regex matching
classtype	classification (e.g., attempted-admin)

6 Preprocessors

Preprocessors normalize traffic and extract data to be used by the detection engine. Common preprocessors:

- stream5 (TCP stream reassembly)
- frag3 (IP defragmentation)
- *httpinspect(HTTP normalization and detection)*
- dns (DNS preprocessor)
- sfportscan (portscan detection)

7 Output and Logging

Snort supports multiple output modules: unified2 (binary), syslog, alert action, pcap, and database outputs (deprecated direct DB writes). Modern deployments write to unified2 and use Barnyard2 or other processors to move alerts to SIEM tools.

8 Snort in Inline (IPS) Mode

When deployed inline, Snort can drop or reject malicious packets using NFQUEUE on Linux or similar mechanisms. This requires careful tuning to avoid dropping legitimate traffic.

9 Rule Writing Best Practices

- Use precise *HOME_NET/EXTERNAL_NET* definitions to reduce false positives
- Prefer content and pcre that uniquely identify malicious payloads
- Use flow and flowbits to ensure correct packet direction/state
- Keep SID numbering consistent (private SIDs typically > 1000000)
- Test rules in logging (alert) mode before switching to drop

10 Examples: Practical Rules

10.1 Detecting a simple web shell signature

```
1 alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (msg:"Possible PHP web
  shell upload"; flow:to_server,established; content:"eval("; nocase;
  http_client_body; sid:1000002; rev:1;)
```

10.2 Detecting SMB (EternalBlue-like) probe (example signature)

```
1 alert tcp $EXTERNAL_NET any -> $HOME_NET 445 (msg:"SMB suspicious NT
  Trans request"; flow:to_server,established; content:"\x00\x00\x00";
  offset:4; depth:3; sid:1000003; rev:1;)
```

11 Testing and Validation

11.1 Running Snort in test mode

```
1
2 # check configuration and rules
3
4 snort -c /etc/snort/snort.conf -T
5
6 # run in console alert mode on interface eth0
7
8 snort -A console -q -c /etc/snort/snort.conf -i eth0
```

11.2 Generating test traffic

Use tools such as `nmap`, `hping3`, `netcat`, or crafted pcap files to trigger rules.

12 Performance Tuning

- Use BPF filters to limit captured traffic
- Disable unnecessary preprocessors
- Use multiple Snort instances and load balancing for high traffic
- Consider hardware offload and tuning kernel parameters

13 Deployment Scenarios

13.1 Tap/monitoring mode

Snort listens on a mirrored port or tap and logs alerts without affecting traffic.

13.2 Inline/IPS mode

Snort sits inline (e.g., between network segments) and can drop malicious packets.

13.3 Distributed sensor deployment

Multiple Snort sensors report to a centralized collector or SIEM via syslog/unified2 or message bus.

14 Integrations

Common integrations include Barnyard2 (unified2 processing), ELK/Elastic Stack, SecurityOnion, Splunk, and OSSIM.

15 Troubleshooting

- "No rules loaded" — check include paths and rule syntax
- "Too many false positives" — tune `HOME_NET`, `addexceptions(passrules)`, and `refinecontent`
- "Performance issues" — enable packet filters, reduce preprocessors, increase resources

16 Appendix: Useful Commands

```
1
2 # test configuration
3
4 snort -T -c /etc/snort/snort.conf
5
6 # run snort with pcap logging
7
8 snort -c /etc/snort/snort.conf -i eth0 -l /var/log/snort -k none
9
10 # check active rules loaded (example parsing)
11
12 grep -n "alert" /etc/snort/rules/*.rules | wc -l
```

17 Appendix: Sample local.rules

```
1
2 # Local rules
3
4 alert icmp any any -> $HOME_NET any (msg:"ICMP ping to HOME_NET"; sid
   :1000004; rev:1;)
5
6 # Detect HTTP GET for suspicious path
7
```

```
s | alert tcp any any -> $HOME_NET 80 (msg:"HTTP GET suspicious /shell.php"  
    ; flow:to_server,established; content:"GET /shell.php"; http_uri;  
    sid:1000005; rev:1;)
```

18 References and Further Reading

- Snort official documentation and rule distribution
- Community rules and Emerging Threats (ET) rules
- Books and training resources on IDS/IPS

This document was generated to serve as a starting point for Snort configuration, rule writing, and deployment. Customize network addresses, interfaces, and rules to match your environment.