

## CHAPTER 7

# Network Security Monitoring

# Chapter Overview

01

## Security Onion

NSM platform — architecture and integrated toolset

02

## Zeek (formerly Bro)

Protocol analysis and structured network logging

03

## Snort, Sguil & Squert

Signature IDS and analyst workflow

04

## Elastic Stack

Log aggregation, search, and visualization

05

## Text-Based Log Analysis

Command-line tools for rapid log parsing

# Security Onion

The NSM platform for defenders

# Security Onion Architecture

**Security Onion:** free, open-source Linux distro for NSM and log management

**Three node types:** Manager, Search, Sensor

**Sensors:** tap or span port, capture raw traffic and generate logs

**Manager:** centralized configuration, alert management, user access

**Search nodes:** scalable Elasticsearch for large environments

**Integrates:** Zeek, Suricata, Elastic Stack, Kibana, Fleet, CyberChef

Suitable for SOC, hunt teams, and incident response teams

# Security Onion Deployment Modes

**Standalone:** all components on one system — lab/small deployment

**Distributed:** dedicated sensors, search nodes, and manager — enterprise deployment

**Import:** offline analysis of PCAP files — no live traffic required

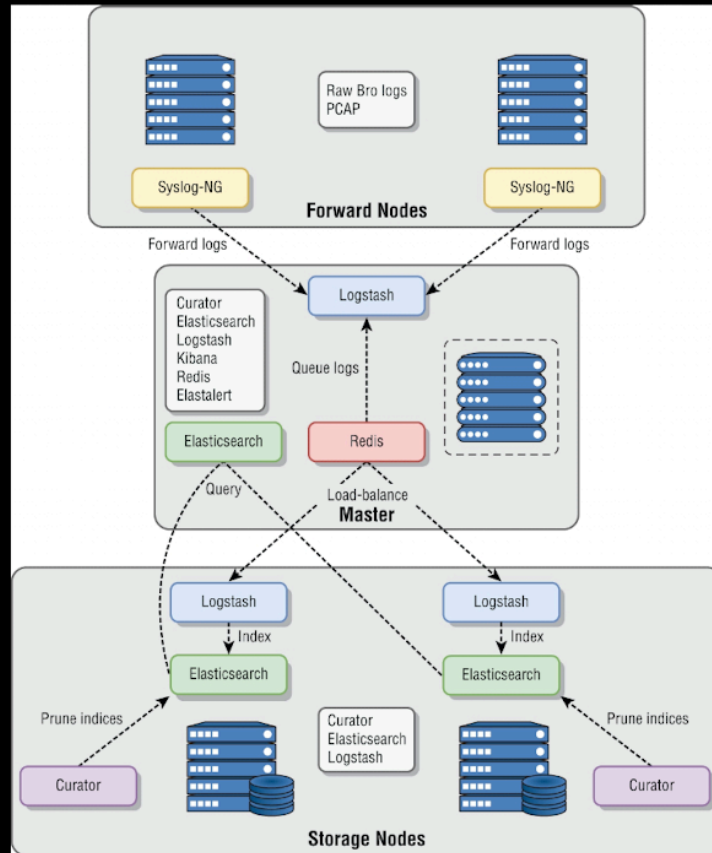
**Eval mode:** simplified setup for evaluation — not for production

**Sensor deployment:** span/mirror port from managed switch or inline tap

**TAP (Test Access Point):** passive optical or copper tap — no traffic impact

**Sizing:** 1 Gbps sustained traffic requires ~8 CPU cores and fast SSD storage

**Tuning:** whitelist internal scanners, backup tools to reduce noise



**Figure 7.1:** The recommended Security Onion architecture

# Snort, Sguil & Squert

Signature detection and alert triage

# Snort / Suricata + Alert Management

**Snort / Suricata:** signature-based IDS — match packets against rule sets

**Rule sets:** Emerging Threats, Snort Community, custom organizational rules

**Sguil:** analyst console — correlates IDS alerts with full packet capture

**Click-to-pivot:** Sguil alert → full packet reconstruction → Wireshark

**Squert:** web-based interface for Sguil database — summary views, search

**Alert fatigue is real:** tune rules to reduce false positives

Combine signature detection (Snort) with behavioral (Zeek) for best coverage

SGUIL-0.9.0 - Connected To localhost

File Query Reports Sound: Off ServerName: localhost UserName: user UserID: 2 2019-06-13 11:14:35 GMT

RealTime Events Escalated Events

ST	CNT	Sensor	Alert ID	Date/Time	Src IP	SPort	Dst IP	DPort	Pr	Event Message
RT	349	so-ossec	1.8	2019-06-06 09:39:46	0.0.0.0		0.0.0.0		0	[OSSEC] File added to the system.
RT	1	so-ossec	1.99	2019-06-08 09:40:07	0.0.0.0		0.0.0.0		0	[OSSEC] Host-based anomaly detection event (pro
RT	7	so-ossec	1.100	2019-06-08 09:40:09	0.0.0.0		0.0.0.0		0	[OSSEC] New group added to the system
RT	7	so-ossec	1.101	2019-06-08 09:40:09	0.0.0.0		0.0.0.0		0	[OSSEC] New user added to the system
RT	2	so-ossec	1.117	2019-06-06 10:10:51	0.0.0.0		0.0.0.0			[OSSEC] System user successfully logged to the
RT	4	so-ens34-1	3.1	2019-06-11 16:03:41	192.168.3.35	1032	195.2.253.92	80	6	ET TROJAN Tibs/Harnig Downloader Activity
RT	5	so-ens34-1	3.2	2019-06-11 16:03:41	192.168.3.35	1032	195.2.253.92	80	6	ET USER_AGENTS Suspicious User-Agent + Pos
RT	24	so-ens34-1	3.5	2019-06-11 16:03:41	195.2.253.92	80	192.168.3.35	1032	6	ET POLICY PE EXE or DLL Windows file downlo
RT	24	so-ens34-1	3.17	2019-06-11 16:03:41	195.2.253.92	80	192.168.3.35	1032	6	ET TROJAN Possible Windows executable sent v
RT	1	so-ens34-1	3.55	2019-06-11 16:03:41	192.168.3.35	1035	66.96.224.213	80	6	ET TROJAN Generic .bin download from Dotted C
RT	1	so-ens34-1	3.56	2019-06-11 16:03:41	192.168.3.35	1036	195.2.253.92	80	6	ET TROJAN TrojanDownloader Win32/Harnig gen

IP Resolution Agent Status Snort Statistics System Msc

Reverse DNS  Enable External DNS

Src IP:

Src Name:

Dst IP:

Dst Name:

Whois Query:  None  Src IP  Dst IP

Show Packet Data  Show Rule

alert tcp \$HOME\_NET any -> \$EXTERNAL\_NET \$HTTP\_PORTS (msg:"ET TROJAN Tibs/Harnig Downloader Activity"; flow:to\_server,established; content:".php?adv=adv"; http\_uri; content:"User-Agent[3a] "; http\_header; nocase; content:".jver"; distance:0; http\_header; fast\_pattern; pcre:"/^User-Agent[3a]\*[^\r\n]+[^\r\n]+/"; reference:url,www.microsoft.com/security/portal/Threat/Encyclopedia/Entry.aspx?Name=TrojanDownloader%3aWin32%2fHarnig; reference:url,www.threatexpert.com/report.aspx?md5=2ce9c871a8a217cafc0ce15c6c1e8dfc;

IP	Source IP	Dest IP	Ver	HL	TOS	len	ID	Flags	Offset	TTL	chkSum
	192.168.3.35	195.2.253.92	4	5	0	180	58	2	0	128	3017

U A P R S F

S o u r c e D e s t R R R C S S Y I

TCP	Source Port	Dest Port	1 0 G K H T N N	Seq #	Ack #	Offset	Res Window	Urp	ChkSum	
	1032	80	. . . . X X . . . .	4251506915	2398391695	5	0	64860	0	16997

D A T A

47	45	54	28	2F	74	64	66	70	6D	6D	6E	2F	68	6E	68	GET /tdfpmen/hnk
70	79	7A	2E	70	68	70	3F	61	64	76	3D	61	64	76	35	ppz.php?adv=adv5
31	36	20	48	54	54	50	2F	31	2E	31	9D	0A	55	73	65	16 HTTP/1.1..Use
72	2D	41	07	65	6E	74	3A	29	4D	6F	7A	69	6C	6C	61	r-Agent: Mozilla
2F	34	2E	39	20	28	63	6F	6D	70	61	74	69	62	6C	65	/4.0 (compatible

Search Packet Payload  Hex  Text  NoCase

**Figure 7.2:** The RealTime Events queue of the Sguil interface

\*eth0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.200.13	10.0.200.10	TLSv1.2	4184	Application Data
2	0.000272418	10.0.200.13	10.0.200.10	TLSv1.2	5157	Application Data
3	0.000471676	10.0.200.10	10.0.200.13	TCP	66	9200 → 34102 [ACK] Seq=1 Ack=
4	0.001037597	10.0.200.10	10.0.200.13	TCP	66	9200 → 34102 [ACK] Seq=1 Ack=
5	0.022532803	10.0.200.10	10.0.200.13	TLSv1.2	1838	Application Data
6	0.022574536	10.0.200.13	10.0.200.10	TCP	66	34102 → 9200 [ACK] Seq=9210 A
7	0.606091383	10.0.200.13	10.0.200.10	TLSv1.2	4184	Application Data
8	0.606631444	10.0.200.13	10.0.200.10	TCP	7306	37732 → 9200 [PSH, ACK] Seq=4
9	0.606656467	10.0.200.10	10.0.200.13	TCP	66	9200 → 37732 [ACK] Seq=1 Ack=
10	0.606704097	10.0.200.13	10.0.200.10	TLSv1.2	5342	Application Data
11	0.607294428	10.0.200.10	10.0.200.13	TCP	66	9200 → 37732 [ACK] Seq=1 Ack=
12	0.621930926	10.0.200.13	10.0.200.10	TLSv1.2	4184	Application Data
13	0.622156864	10.0.200.13	10.0.200.10	TLSv1.2	2645	Application Data
14	0.622300132	10.0.200.10	10.0.200.13	TCP	66	9200 → 55932 [ACK] Seq=1 Ack=
15	0.622300382	10.0.200.10	10.0.200.13	TCP	66	9200 → 55932 [ACK] Seq=1 Ack=
16	0.622964776	10.0.200.13	10.0.200.10	TLSv1.2	4184	Application Data
17	0.623108029	10.0.200.13	10.0.200.10	TLSv1.2	6622	Application Data
18	0.623284513	10.0.200.10	10.0.200.13	TCP	66	9200 → 55936 [ACK] Seq=1 Ack=
19	0.623321576	10.0.200.10	10.0.200.13	TCP	66	9200 → 55936 [ACK] Seq=1 Ack=
20	0.637851063	10.0.200.10	10.0.200.13	TLSv1.2	3301	Application Data
21	0.637941810	10.0.200.13	10.0.200.10	TCP	66	37732 → 9200 [ACK] Seq=16635
22	0.642515006	10.0.200.10	10.0.200.13	TLSv1.2	1246	Application Data
23	0.642537266	10.0.200.13	10.0.200.10	TCP	66	55932 → 9200 [ACK] Seq=6698 A

Wireshark network capture interface — inspecting raw packets during incident investigation

# Snort / Suricata Rule Syntax

**Rule format:** action protocol src src\_port -> dst dst\_port (options)

Example: alert tcp any any -> \$HOME\_NET 4444 (msg:'Metasploit default port'; sid:9000001; rev:1;)

**Action types:** alert (log + alert), drop (IPS mode, block), pass (ignore)

**Content matching:** content:'cmd.exe'; nocase; — search for strings in payload

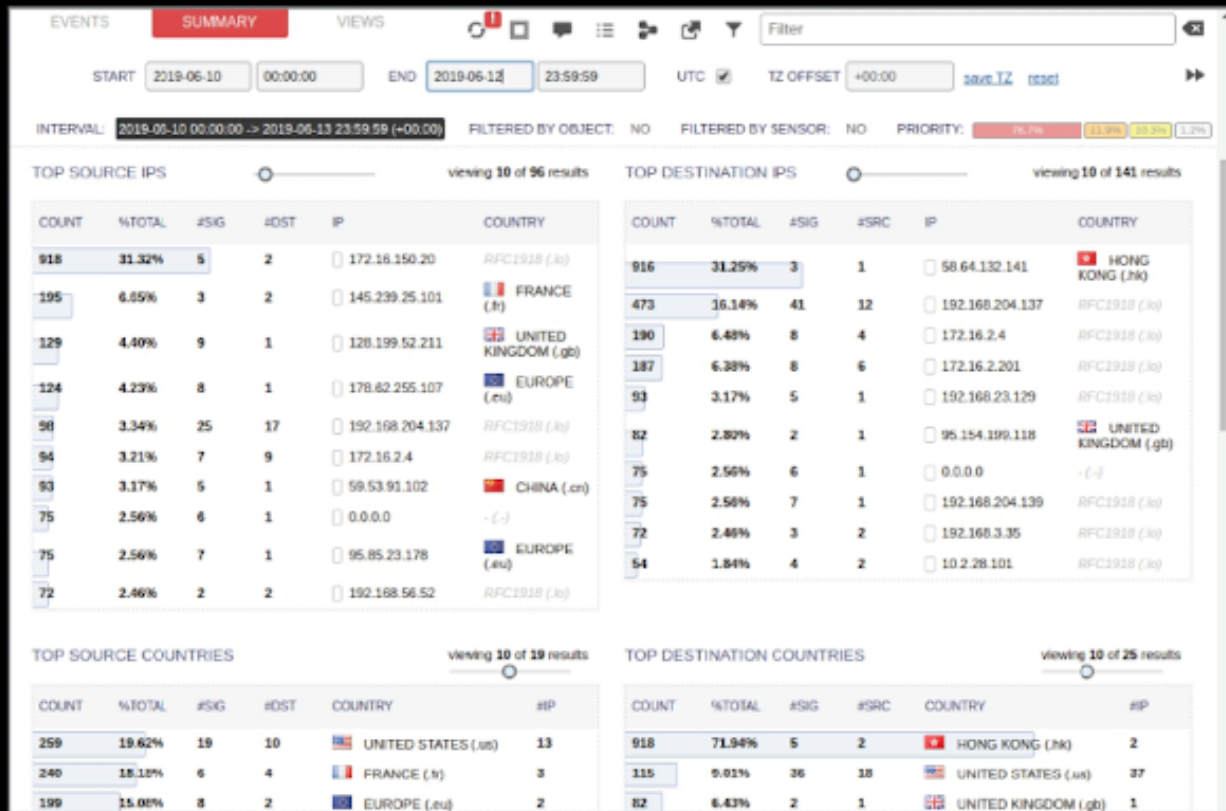
**PCRE (Perl Compatible Regular Expression):**

pcre:'/\bin/sh/i'; — regex for shell commands in payload

**Flow:** flow:established,to\_server; — match only established connections

**Threshold:** threshold:type limit, track by\_src, count 5, seconds 60; — rate limiting

**Emerging Threats:** free community rules updated daily at [rules.emergingthreats.net](http://rules.emergingthreats.net)



**Figure 7.7:** The Squert Summary tab dashboards

# Zeek (formerly Bro)

Structured protocol-aware network logging

# Zeek Log Types

## Core Log Files

**conn.log:** every TCP/UDP/ICMP connection (src, dst, duration, bytes)

**dns.log:** all DNS queries and responses

**http.log:** HTTP requests — method, URI, user-agent, response code

**ssl.log:** TLS/SSL sessions — cert details, SNI, cipher suite

**files.log:** all file transfers with MIME type and hash

## Threat Hunting with Zeek

**Long-duration connections:** beacons to C2

**High-volume DNS:** DNS tunneling or DGA

**Unusual user agents:** malware C2 callbacks

**Self-signed certs on non-standard ports**

**zeek-cut:** command-line tool to extract specific fields from logs

# Zeek Log Field Reference — Key Fields

**conn.log key fields:** ts, id.orig\_h, id.orig\_p, id.resp\_h, id.resp\_p, proto, duration, orig\_bytes, resp\_bytes, conn\_state

**conn\_state values:** S1=established, SF=normal close, REJ=rejected, RSTO=aborted, S0=attempt no reply

**dns.log:** ts, query, qtype\_name, rcode\_name, answers — hunt for NXDomain storms (DGA activity)

**http.log:** method, host, uri, user\_agent, status\_code — spot malware beaconing by user-agent

**ssl.log:** server\_name (SNI), subject, issuer, cipher, validation\_status — find self-signed certs

**files.log:** fuid, mime\_type, filename, md5, sha1 — hash every file transferred over HTTP/SMB

**weird.log:** protocol anomalies and parser errors — often reveals malformed C2 traffic

**zeek-cut usage:** zeek-cut id.orig\_h id.resp\_p conn\_state < conn.log — extract specific fields

# Threat Hunting with Zeek Logs

**C2 beaconing:** group conn.log by dest IP, compute interval standard deviation — low  $\sigma$  = beacon

**DNS tunneling:** cat dns.log | zeek-cut query | awk '{print length(\$1)}' | sort -n | tail -20

**High-query domains:** cat dns.log | zeek-cut query | sort | uniq -c | sort -rn | head -20

**User-agent analysis:** cat http.log | zeek-cut user\_agent | sort | uniq -c | sort -rn

**Large outbound:** cat conn.log | zeek-cut id.resp\_h orig\_bytes | sort -k2 -rn | head -20

**Self-signed TLS:** cat ssl.log | zeek-cut server\_name issuer | grep 'CN=.\*CN=' | head -20

**New external IPs:** baseline known-good IPs, then filter: grep -vFf known\_ips.txt

**East-west SMB:** connections on port 445 between internal workstations — never normal

# Elastic Stack & Text Analysis

Search and visualize at scale

# Elastic Stack & Command-Line Analysis

## Elastic Stack (ELK)

A suite of tools to aggregate, index, search, and display large amounts of information

**Elasticsearch:** distributed search and storage

**Logstash:** ingest, parse, and enrich logs

**Kibana:** dashboards, search, and visualizations

**Beats:** lightweight shippers (Filebeat, Packetbeat)

Enables hunting across millions of events in seconds

## Text-Based Log Analysis

**grep -E 'pattern' zeek/conn.log:** quick searches

```
awk -F'\t' '{print $5}' conn.log | sort | uniq -c |  
sort -rn
```

```
zeek-cut id.orig_h id.resp_h id.resp_p conn_state  
< conn.log
```

```
cat http.log | zeek-cut -d ts host uri | grep -v '200'
```

**Command-line fu:** fast, no SIEM required, works on any log

# Kibana Hunting Queries

**KQL syntax:** agent.type:zeek AND network.transport:tcp AND destination.port:4444

**High-volume DNS:** filter on event.dataset:zeek.dns, aggregate by dns.question.name

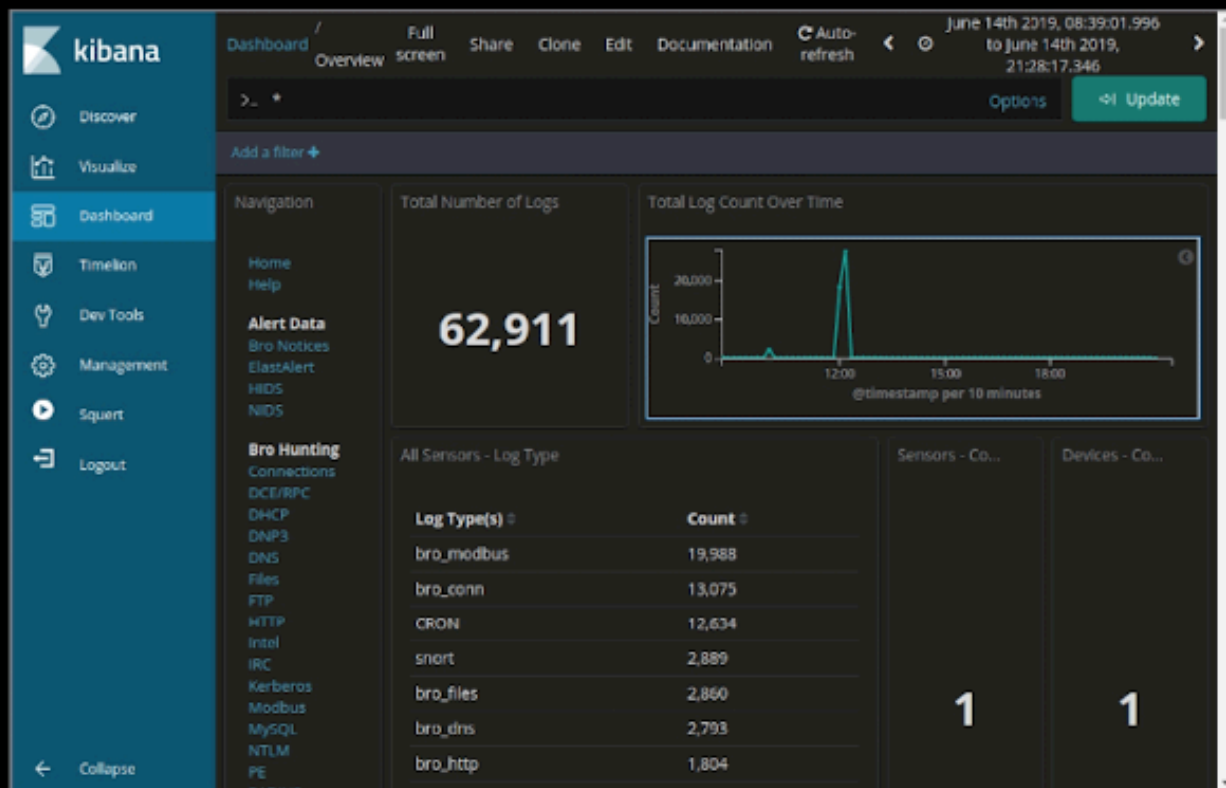
**Unusual user agents:** http.request.user\_agent.original: \* AND NOT (\*Mozilla\* OR \*Chrome\*)

**Long-lived connections:** network.duration > 3600 AND destination.geo.country\_iso\_code != 'US'

**Internal scanning:** source.ip: 10.0.0.0/8 AND destination.port: 445 AND event.type: connection

**Dashboards:** pre-built Zeek, Suricata, and host dashboards in Security Onion

**Timeline pivot:** from alert to full packet capture with one click in Sguil/Security Onion



**Figure 7.8:** The Kibana web interface

# Network Baselineing & Full Packet Capture

**Baseline:** document normal traffic patterns before hunting for abnormal

**Top talkers:** which internal hosts generate most traffic? Any unusual spikes?

**Time-of-day patterns:** spikes at 3 AM when users are offline = automated activity

**New connections:** hosts or IPs not seen in the past 30 days — immediate hunt focus

**pcap:** raw binary capture of every packet — the ground truth of network forensics

**Wireshark display filter:** `http.request.method=='POST' && ip.dst==1.2.3.4`

**Follow TCP Stream:** reconstruct attacker's full session from individual packets

**Export HTTP objects:** File → Export Objects → HTTP — recover transferred files

# Network Forensics Tools

**NetworkMiner:** parses pcap and automatically extracts files, images, credentials, and host data

**Wireshark:** full pcap analysis — display filters, stream reassembly, protocol dissectors

**tcpdump:** command-line capture and display — `tcpdump -i eth0 -w capture.pcap`

**ngrep:** network grep — `ngrep 'password' -q -W byline port 80` — search packet payloads

**Moloch (Arkime):** full packet capture + indexed search — scales to enterprise traffic volumes

**RITA (Real Intelligence Threat Analytics):** automated analysis of Zeek logs for C2 beaconing

**CapTipper:** Python tool for HTTP traffic analysis from pcap — extracts and replays flows

**The right tool depends on scale:** Wireshark for single files; Arkime/Elastic for fleet-level pcap

# Conclusion

Network visibility is essential — you cannot detect what you cannot see

Zeek provides rich, structured logs that complement signature-based IDS

Elastic Stack scales log search to enterprise traffic volumes

Combine IDS alerts with full packet capture for complete incident context

Text-based analysis is a fast, lightweight alternative when tooling isn't available

Baselines are the foundation — without normal, you cannot spot abnormal

# Knowledge Check

The Kahoot! logo is displayed in a large, white, bold, sans-serif font. The text is centered horizontally and vertically within a rectangular area that has a purple-to-blue gradient background. The background image is a blurred photograph of a modern office interior, showing a ceiling with recessed lighting and several glass-walled doors or partitions.

**Kahoot!**