ELK Stack for Security Monitoring Complete Setup Guide with Docker

Table of Contents

1. Overview

This guide describes a reproducible, Docker-based deployment of the ELK Stack (Elasticsearch, Logstash, Kibana) for collecting, indexing, and visualizing security-related telemetry in near real time. The guide is written for a proof-of-concept or small-scale deployment and includes configuration examples, automation scripts, dashboard instructions, and operational guidance.

2. Architecture

Data flows from instrumentation and collectors into Logstash for parsing and enrichment. Logstash forwards structured events to Elasticsearch for indexing and storage. Kibana reads indices from Elasticsearch to power dashboards and visualizations.
Logical data flow:
Data Sources → Logstash → Elasticsearch → Kibana

3. Prerequisites

System requirements (minimum recommended for a small POC):
- Ubuntu 20.04 or 22.04 (or compatible Linux distribution)
- 4 vCPU cores
- 8 GB RAM or more
- 20 GB available disk

Required software:
- Docker and docker-compose (installed and running as a user with appropriate permissions)

Network ports:
- 5601: Kibana (web UI)
- 9200: Elasticsearch HTTP API
- 5000: Logstash TCP input

4. Quick Start

1. Place the provided scripts (elk-security-setup.sh, logstash.conf, collect-metrics.sh, inject-security-data.sh) in a single directory on the target host.
2. Make the setup script executable and run it:
chmod +x elk-security-setup.sh
./elk-security-setup.sh

3. Verify containers are running:
docker ps

4. Access Kibana at http://:5601 and create an index pattern for 'security-metrics-*' (time field: timestamp).

5. Script Breakdown

5.1 elk-security-setup.sh (Docker Compose)

```
#!/bin/bash
echo "ELK Security Monitor setup started..."

# Create docker-compose file
```

```
cat > docker-compose-elk.yml << 'EOF'
version: '3.7'
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.17.0
    container_name: es-security
    environment:
      - discovery.type=single-node
      - xpack.security.enabled=false
    ports:
      - "9200:9200"
    networks:
      - elk-net

  logstash:
    image: docker.elastic.co/logstash/logstash:7.17.0
    container_name: ls-security
    ports:
      - "5000:5000"
    volumes:
      - ./logstash.conf:/usr/share/logstash/pipeline/logstash.conf:ro
    networks:
      - elk-net

  kibana:
    image: docker.elastic.co/kibana/kibana:7.17.0
    container_name: kb-security
    ports:
      - "5601:5601"
    environment:
      - ELASTICSEARCH_HOSTS=http://elasticsearch:9200
    networks:
      - elk-net

networks:
  elk-net:
    driver: bridge
EOF

# Start the stack
docker-compose -f docker-compose-elk.yml up -d
```

5.2 logstash.conf

```
input {
  tcp {
    port => 5000
    codec => json
  }
}

filter {
  mutate {
    convert => {
      "connections" => "integer"
      "open_ports" => "integer"
      "containers" => "integer"
      "arp_entries" => "integer"
    }
  }
}
output {
```

```
  elasticsearch {
    hosts => ["http://elasticsearch:9200"]
    index => "security-metrics-%{+YYYY.MM.dd}"
    # Optional: user and password when security is enabled
    # user => "elastic"
    # password => "${ELASTIC_PASSWORD}"
  }
  stdout { codec => rubydebug }
}
```

5.3 collect-metrics.sh

```
#!/bin/bash
# Collect basic security-related metrics and forward to Logstash over TCP (JSON)
while true; do
  CONNECTIONS=$(netstat -an 2>/dev/null | wc -l)
  OPEN_PORTS=$(ss -tuln 2>/dev/null | wc -l)
  CONTAINERS=$(docker ps -q 2>/dev/null | wc -l)
  ARP_ENTRIES=$(arp -n 2>/dev/null | wc -l)

  TIMESTAMP=$(date -Iseconds)
  HOSTNAME=$(hostname -f 2>/dev/null || hostname)

  METRICS_JSON=$(cat <<METRICS
{
  "timestamp": "${TIMESTAMP}",
  "connections": ${CONNECTIONS},
  "open_ports": ${OPEN_PORTS},
  "containers": ${CONTAINERS},
  "arp_entries": ${ARP_ENTRIES},
  "host": "${HOSTNAME}"
}
METRICS
)

  # Send to Logstash
  echo "${METRICS_JSON}" | nc localhost 5000 || echo "Failed to send metrics to Logstash"
  sleep 30
done
```

5.4 inject-security-data.sh

```
#!/bin/bash
# Inject a sample document into Elasticsearch (for testing)
curl -s -X POST "http://localhost:9200/security-logs-2024.01.01/_doc"   -H "Content-Type: applicati
    "timestamp": "'"$(date -Iseconds)"'",
    "connections": 131,
    "open_ports": 22,
    "containers": 11,
    "arp_entries": 68,
    "host": "security-server"
  }'

echo "Sample document indexed."
```

6. Kibana Dashboard

Initial setup:
- Open Kibana at http://:5601
- Go to Stack Management → Index Patterns and create a new index pattern: security-metrics-*
- Choose 'timestamp' as the time field.

Suggested visualizations:
- Metric: Average of 'connections'
- Line chart: Date histogram of 'timestamp' vs average 'connections'
- Data table: Count grouped by 'host' (terms aggregation)

Use Dashboards to combine visualizations and save them for operations or reporting.

7. Troubleshooting

Common issues and remediation steps:

Elasticsearch container fails to start
- Confirm available memory and increase vm.max_map_count on the host:
sudo sysctl -w vm.max_map_count=262144

No data in Kibana
- Check Elasticsearch indices:
curl http://localhost:9200/_cat/indices?v
- Check Logstash logs and pipeline status:
docker logs ls-security
echo '{"test":"data"}' | nc localhost 5000

Port conflicts
- Find processes using ports (replace port as needed):
sudo ss -tulpn | grep 5601

8. Maintenance

Recommended regular operations:
- Update images and restart the stack:
docker-compose -f docker-compose-elk.yml pull && docker-compose -f docker-compose-elk.yml up -d

- Delete older documents (example: delete older than 30 days):
curl -X POST "http://localhost:9200/security-metrics-*/_delete_by_query" -H "Content-Type: application/json" -d '{"query":{"range":{"timestamp":{"lt":"now-30d"}}}}'

- Backup example:
curl -X GET "http://localhost:9200/security-metrics-*/_search" > backup_data.json

- Monitor resources:
docker stats es-security ls-security kb-security

9. Security Considerations

For production deployments, follow these recommendations:
- Enable Elasticsearch security features (authentication and TLS):
xpack.security.enabled=true
- Do not disable security in production; use strong passwords and rotate credentials.
- Configure TLS/SSL for Elasticsearch transport and HTTP layers.
- Front Kibana with a reverse proxy and require authentication (e.g., OAuth, SSO, or basic auth behind TLS).
- Restrict access to management ports using firewall rules and VPN.
- Store credentials in environment variables or a secrets manager, not in plaintext files.

10. Conclusion and Next Steps

This document provides a complete, production-minded starting point for deploying an ELK Stack-based security telemetry pipeline using Docker. Next steps:
- Build and save Kibana dashboards for operational use.
- Implement alerting (e.g., ElastAlert or built-in alerting) and integrate with an incident management tool.
- Harden the deployment by enabling security, configuring TLS, and applying least-privilege access controls.
- Implement index lifecycle management and retention policies to control storage usage.

Appendix: Useful Commands

```
docker ps -a
docker-compose -f docker-compose-elk.yml ps
docker-compose -f docker-compose-elk.yml logs -f
curl http://localhost:9200/_cluster/health?pretty
echo '{"test":"data"}' | nc localhost 5000
```