Zeek

Introduction to Zeek (formerly Bro)

Zeek is an open-source network monitoring and analysis tool designed to provide deep insight into network traffic. Originally developed at the Lawrence Berkeley National Laboratory, Zeek is widely used in cybersecurity for network security monitoring (NSM), threat detection, and incident response.

zeek®
An Open Source Network Security Monitoring Tool
Flexible, open source, and powered by defenders.

Key Features of Zeek

1. **Network Traffic Analysis** – Zeek captures and analyzes network packets, providing detailed logs of network activity.
2. **Protocol Parsing** – It understands and logs various protocols like HTTP, DNS, FTP, and SSH.
3. **Signature and Anomaly-Based Detection** – Unlike traditional intrusion detection systems (IDS) like Snort and Suricata, Zeek combines signature-based and behavior-based detection techniques.
4. **Logging & Data Collection** – Zeek generates structured logs (conn.log, dns.log, http.log, etc.), which can be used for forensic investigations.
5. **Extensibility with Scripting** – Zeek uses its own scripting language, allowing security professionals to define custom detection logic.
6. **Integration with SIEM & Threat Intelligence** – Zeek logs can be fed into tools like Splunk, ELK Stack, or Security Onion for further analysis.

Zeek vs. Other Tools

Feature	Zeek	Snort/Suricata (IDS)	Wireshark
Detection Type	Behavioral & Signature	Signature-based	Packet Analysis
Primary Use	Network Security Monitoring	Intrusion Detection	Packet Inspection
Logging	Extensive and detailed	Limited logs	No automatic logging
Scripting Support	Yes	No	No
Performance	Processes traffic at scale	High-speed packet matching	Manual deep packet inspection

Use Cases of Zeek

- **Threat Detection:** Identifying network anomalies, malware infections, and brute-force attacks.
- **Incident Response:** Analyzing logs to trace attack sources and impact.
- **Network Forensics:** Investigating suspicious network behavior over time.
- **Compliance Monitoring:** Ensuring adherence to security policies by logging network activity.

Merits and Demerits of Zeek

Merits (Advantages)

1. **Deep Network Visibility** – Provides detailed logs for various network protocols, enabling comprehensive monitoring.
2. **Behavior-Based Detection** – Detects anomalies and suspicious activities beyond signature-based methods.
3. **Custom Scripting** – Zeek's scripting language allows customization for specific security needs.
4. **Scalability** – Works well in large-scale environments for continuous network monitoring.
5. **Integration with SIEM & Threat Intelligence** – Supports tools like Splunk, ELK Stack, and Security Onion for enhanced security analytics.
6. **Passive Monitoring** – Operates without interfering with network traffic, reducing the risk of performance degradation.

Demerits (Disadvantages)

1. **Steep Learning Curve** – Requires knowledge of Zeek scripting and log analysis for effective use.
2. **Resource-Intensive** – High network traffic environments may require significant CPU and storage resources.
3. **Lack of Real-Time Blocking** – Unlike traditional IDS/IPS, Zeek does not actively block threats; it only logs and analyzes traffic.
4. **Complex Deployment** – Setting up and configuring Zeek properly requires expertise.
5. **Limited Windows Support** – Primarily designed for Linux-based systems, limiting its deployment options.

Installing Zeek on Linux (Ubuntu/Kali/Debian-Based Systems)

Many people find it difficult to install Zeek on linux, I prefer building from source. Easy and Convenient.

Step 1: Update the System

```
sudo apt update && sudo apt upgrade -y
```

Step 2: Installing required dependencies

```
sudo apt-get install cmake make gcc g++ flex libfl-dev bison libpcap-dev libssl-dev python3 python3-dev swig zlib1g-dev
```

```
sudo apt-get install python3-git python3-semantic-version
```

Step 3: Cloning the Repo

```
git clone --recurse-submodules https://github.com/zeek/zeek
```

Step 4: Configuring and building

```
./configure
```

```
Make
```

```
sudo make install
```

Minimal ZeekControl Configuration (Standalone Setup)

1. Set the Network Interface to Monitor

- Open the configuration file:

```
nano $PREFIX/etc/node.cfg
```

- Modify it as follows:

```
[zeek]
type=standalone
host=localhost
interface=eth0  # Replace 'eth0' with your actual interface from `ifconfig`
```

4. Start ZeekControl

- Open the ZeekControl shell:

```
zeekctl
```

- Perform an initial installation:

```
[ZeekControl] > install
```

- Start Zeek:

```
[ZeekControl] > start
```

5. Deploy Changes (Alternative to Install & Start)

- Run after modifying configurations or scripts:

```
[ZeekControl] > deploy
```

6. Troubleshooting

- Check errors if Zeek fails to start:

```
[ZeekControl] > diag
```

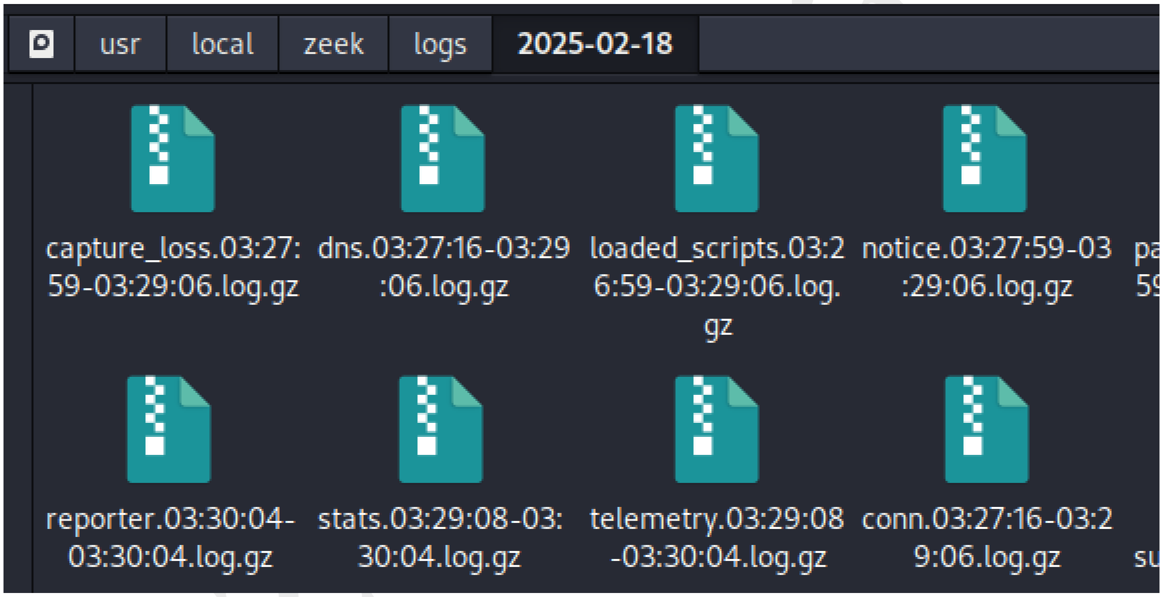
7. Stopping Zeek

- To stop Zeek:

  ```
  [ZeekControl] > stop
  ```

  Note: $PREFIX is basically where you have cloned the repo.

  Once Zeek is stopped, the log files in the `/usr/local/zeek/logs/current` directory are compressed and moved into the current day named folder inside the `/usr/local/zeek/logs/specific date` directory.



usr / local / zeek / logs / 2025-02-18

capture_loss.03:27:59-03:29:06.log.gz
dns.03:27:16-03:29:06.log.gz
loaded_scripts.03:26:59-03:29:06.log.gz
notice.03:27:59-03:29:06.log.gz
pa
59

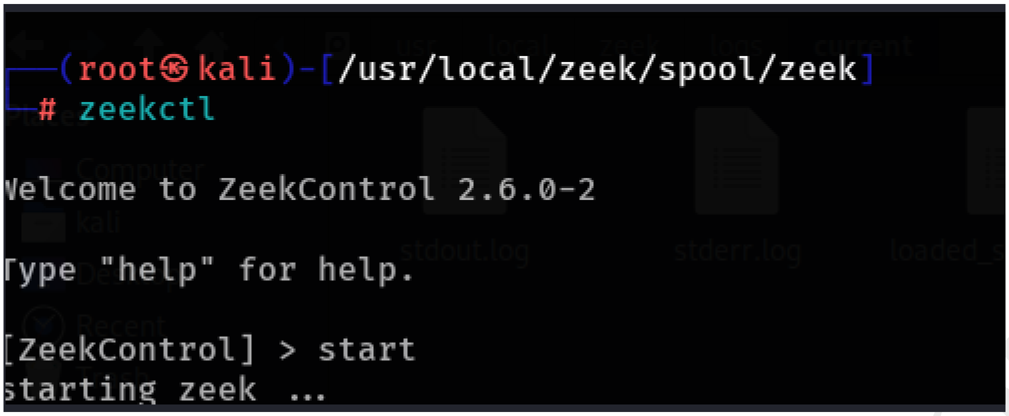
reporter.03:30:04-03:30:04.log.gz
stats.03:29:08-03:30:04.log.gz
telemetry.03:29:08-03:30:04.log.gz
conn.03:27:16-03:29:06.log.gz
su

Now, if you want to live monitor you can just start zeekctl and then execute the command:

(Remember to be in the "current" folder)

```
tail -f  <log file name>
```

Here is the pictorial steps:

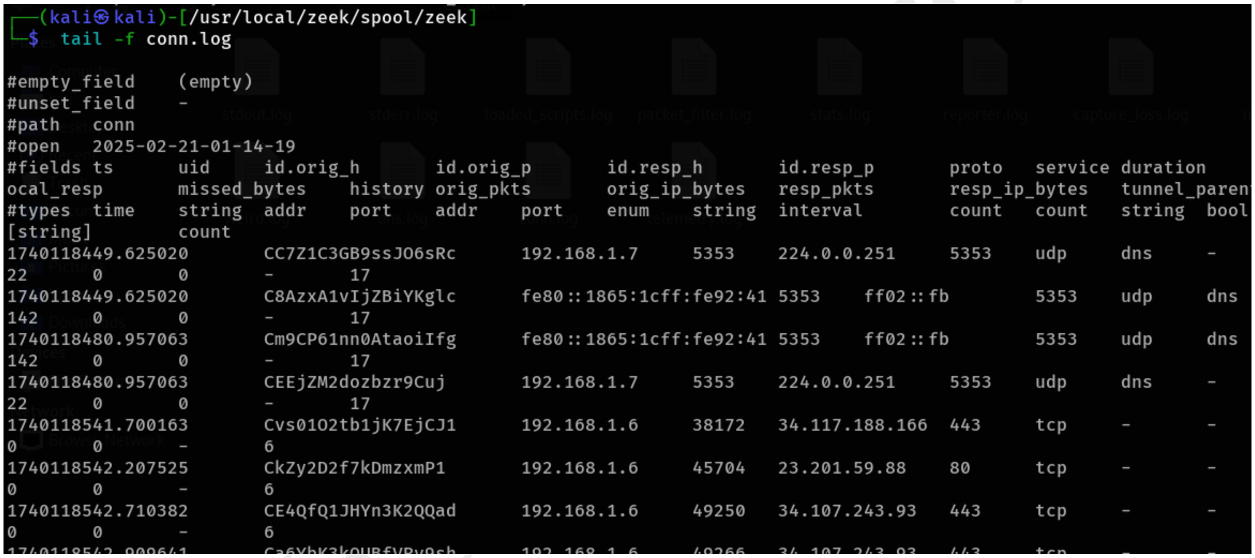


```
┌──(root㉿kali)-[/usr/local/zeek/spool/zeek]
└─# zeekctl

Welcome to ZeekControl 2.6.0-2

Type "help" for help.

[ZeekControl] > start
starting zeek ...
```



```
┌──(kali㉿kali)-[/usr/local/zeek/spool/zeek]
└─$ tail -f conn.log

#empty_field    (empty)
#unset_field    -
#path   conn
#open   2025-02-21-01-14-19
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p       proto   service duration
ocal_resp       missed_bytes    history orig_pkts       orig_ip_bytes   resp_pkts       resp_ip_bytes   tunnel_paren
#types  time    string  addr    port    addr    port    enum    string  interval        count   count   string  bool
[string]        count
1740118449.625020       CC7Z1C3GB9ssJO6sRc      192.168.1.7     5353    224.0.0.251     5353    udp     dns     -
22      0       0       -       17
1740118449.625020       C8AzxA1vIjZBiYKglc      fe80::1865:1cff:fe92:41 5353    ff02::fb        5353    udp     dns
142     0       0       -       17
1740118480.957063       Cm9CP61nn0AtaoiIfg      fe80::1865:1cff:fe92:41 5353    ff02::fb        5353    udp     dns
142     0       0       -       17
1740118480.957063       CEEjZM2dozbzr9Cuj       192.168.1.7     5353    224.0.0.251     5353    udp     dns     -
22      0       0       -       17
1740118541.700163       Cvs01O2tb1jK7EjCJ1      192.168.1.6     38172   34.117.188.166  443     tcp     -       -
0       0       -       6
1740118542.207525       CkZy2D2f7kDmzxmP1       192.168.1.6     45704   23.201.59.88    80      tcp     -       -
0       0       -       6
1740118542.710382       CE4QfQ1JHYn3K2QQad      192.168.1.6     49250   34.107.243.93   443     tcp     -       -
0       0       -       6
1740118542.909641       Ca6YbK3kOUBfVRv9sh      192.168.1.6     49266   34.107.243.93   443     tcp     -       -
```

Now for a basic example lets analyse dns.log .

Step 1: Start zeek to capture the network interface.

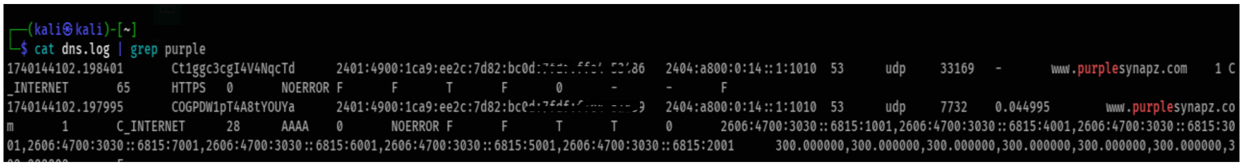
Step 2: Go to a browser on your VM or on your Host Machine (only in bridged network).

Step 3: Open any website, lets say purplesynapz.com

Step 4: Now go to current logs folder and open the terminal:

cat dns.log | grep purple

(grep will filter out the data for a selected keyword)



```
┌──(kali㉿kali)-[~]
└─$ cat dns.log | grep purple
1740144102.198401       Ct1ggc3cgI4V4NqcTd      2401:4900:1ca9:ee2c:7d82:bc0d:[redacted]86   2404:a800:0:14::1:1010  53      udp     33169   -       www.purplesynapz.com    1 C
_INTERNET       65      HTTPS   0       NOERROR F       F       T       F       0       -       -       F
1740144102.197995       COGPDW1pT4A8tYOUYa      2401:4900:1ca9:ee2c:7d82:bc0[redacted]9   2404:a800:0:14::1:1010  53      udp     7732    0.044995        www.purplesynapz.co
m       1       C_INTERNET      28      AAAA    0       NOERROR F       F       T       T       0       2606:4700:3030::6815:1001,2606:4700:3030::6815:4001,2606:4700:3030::6815:30
01,2606:4700:3030::6815:7001,2606:4700:3030::6815:6001,2606:4700:3030::6815:5001,2606:4700:3030::6815:2001      300.000000,300.000000,300.000000,300.000000,300.000000,300.000000,3
```

Key Observations:

1. **Client IP (2401:4900:1ca9:ee2c:7d82:…..)**
   - This is the source IP making the DNS query.
   - It appears to be an **IPv6 address**.
2. **DNS Server (2404:a800:0:14::1:1010)**
   - This is the **DNS resolver** handling the request.
3. **Queries & Responses:**
   - Three queries were made for www.purplesynapz.com:
     - **HTTPS Record Query (Type 65)** → Likely related to encrypted DNS.
     - **AAAA Record Query (Type 28)** → Request for an IPv6 address.
     - **A Record Query (Type 1)** → Request for an IPv4 address.
4. **Response Details:**
   - **AAAA Record Response:**
     - Returns multiple IPv6 addresses:
     - 2606:4700:3030::6815:1001, 2606:4700:3030::6815:4001, etc.
     - These are **Cloudflare IPs**, indicating www.purplesynapz.com is behind Cloudflare.
   - **A Record Response:**
     - Returns multiple IPv4 addresses:
     - 104.21.112.1, 104.21.96.1, etc.
     - Again, Cloudflare's IP range.
5. **Query Timing:**
   - The responses were received in **0.044995s** and **0.143526s**, indicating fast DNS resolution.
   - The TTL (Time-To-Live) for the records is **300 seconds** (5 minutes), meaning the response is valid for that duration.

How Zeek analyses the Traffic

Zeek captures network traffic from a live interface or a PCAP file, then processes it using event-driven scripting. It first parses packets at the network layer, extracts session details, and applies protocol analyzers (e.g., HTTP, DNS, SSL). The extracted metadata is fed into Zeek scripts, which generate structured logs based on predefined or custom policies. These logs (e.g., conn.log, http.log, dns.log) are then stored in /opt/zeek/logs/ or the configured directory for further analysis.

When you use zeekctl start, Zeek runs on a **live network interface** as defined in node.cfg. It does **not** process a PCAP file. If you want to analyze a PCAP, you must use zeek -r <pcap_file> instead of zeekctl.