Zeek Fundamentals

Zeek, as defined by its developers, is an open-source network traffic analyzer. It is typically employed to scrutinize every bit of traffic on a network, digging deep for any signs of suspicious or malicious activity. But, Zeek isn't limited to just that. It can also be a handy tool for troubleshooting network issues and conducting a variety of measurements within a network. Once we deploy Zeek, our defensive cybersecurity teams (or blue teams) can immediately tap into a wealth of log files, which offer an elevated view of all types of network activity. Specifically, these logs contain detailed records of every connection made over the network, along with transcripts of application-layer activities such as DNS queries and responses, HTTP sessions, and more. But Zeek's capabilities extend beyond mere logging. It comes bundled with a suite of functions for analyzing and detecting network activities.

What sets Zeek apart is its highly capable scripting language, which allows users to craft Zeek scripts that are functionally similar to Suricata rules. This powerful feature enables Zeek to be fully customizable and extendable, letting blue team members develop their own logic and strategies for network analysis and intrusion detection.

Considering Zeek's functionality, combined with its ability to run on standard hardware and the powerful scripting language, it becomes clear that we're not looking at another signature-based IDS. Instead, Zeek is a platform that can facilitate semantic misuse detection, anomaly detection, and behavioral analysis..

Zeek's Operation Modes

Zeek operates in the following modes:

- `Fully passive traffic analysis`
- `libpcap interface for packet capture`
- `Real-time and offline (e.g., PCAP-based) analysis`
- `Cluster support for large-scale deployments`

Zeek's Architecture

Zeek's architecture comprises two primary components: the `event engine` (or core) and the `script interpreter`.

The `event engine` takes an incoming packet stream and transforms it into a series of high-level `events`. In Zeek's context, these `events` describe network activity in policy-neutral terms, meaning they inform us of what's happening, but they don't offer an interpretation or evaluation of it. For instance, an HTTP request will be transformed into an `http_request` event. While this event provides all the details of the request, it doesn't offer any judgement or analysis, such as whether a port corresponds to a port known to be used by malware.

Such interpretation or analysis is provided by Zeek's `script interpreter`, which executes a set of event handlers written in Zeek's scripting language (Zeek scripts). These scripts express the site's security policy, defining actions to be taken upon the detection of certain events.

Events generated by Zeek's core are queued in an orderly manner, awaiting their turn to be processed on a first-come, first-served basis. Most of Zeek's events are defined in `.bif` files located in the `/scripts/base/bif/plugins/` directory. For a more comprehensive list of events, refer to the following resource:
https://docs.zeek.org/en/stable/scripts/base/bif/

Zeek Logs

As for Zeek's logging, when we use Zeek for offline analysis of a PCAP file, the logs will be stored in the current directory.

Among the diverse array of logs Zeek produces, some familiar ones include:

- `conn.log`: This log provides details about IP, TCP, UDP, and ICMP connections.
- `dns.log`: Here, you'll find the details of DNS queries and responses.
- `http.log`: This log captures the details of HTTP requests and responses.
- `ftp.log`: Details of FTP requests and responses are logged here.
- `smtp.log`: This log covers SMTP transactions, such as sender and recipient details.

Let's consider the `http.log` as an example. It is chock-full of valuable data like:

- `host`: The HTTP domain/IP.
- `uri`: The HTTP URI.
- `referrer`: The referrer of the HTTP request.
- `user_agent`: The client's user agent.
- `status_code`: The HTTP status code.

For a more exhaustive list of common Zeek logs and their respective fields, refer to the following resource: https://docs.zeek.org/en/master/logs/index.html

It's noteworthy to mention that Zeek, in its standard configuration, applies gzip compression to log files every hour. The older logs are then transferred into a directory named in the `YYYY-MM-DD` format. When dealing with these compressed logs, alternative tools like `gzcat` for printing logs or `zgrep` for searching within logs can come in handy.

You can find examples on how to work with gzip-compressed Zeek logs at this link: https://blog.rapid7.com/2016/06/02/working-with-bro-logs-queries-by-example/

As stated earlier, when interacting with Zeek log files, we can utilize native Unix commands such as cat or grep. However, Zeek also provides a specialized tool known as `zeek-cut` for handling log files. This utility accepts Zeek log files via standard input using pipelines or stream redirections and then delivers the specified columns to the standard output.

If you're interested in exploring Zeek examples, use cases, and learning the basics of writing Zeek scripts, take a look at the following link: https://docs.zeek.org/en/stable/examples/index.html

For a quick start guide to Zeek, refer to the following link: https://docs.zeek.org/en/stable/quickstart/index.html

Zeek Key Features

Key features that bolster Zeek's effectiveness include:

- Comprehensive logging of network activities
- Analysis of application-layer protocols (irrespective of the port, covering protocols like HTTP, DNS, FTP, SMTP, SSH, SSL, etc.)
- Ability to inspect file content exchanged over application-layer protocols
- IPv6 support
- Tunnel detection and analysis
- Capability to conduct sanity checks during protocol analysis
- IDS-like pattern matching
- Powerful, domain-aware scripting language that allows for expressing arbitrary analysis tasks and managing network state over time
- Interfacing that outputs to well-structured ASCII logs by default and offers alternative backends for ElasticSearch and DataSeries
- Real-time integration of external input into analyses
- External C library for sharing Zeek events with external programs
- Capability to trigger arbitrary external processes from within the scripting language