
INTRUSION DETECTION SYSTEM

Zeek Cluster

Author

Dana Abushwesh

Leen Falah

1 Abstract

This report provides a concise overview of the steps involved in setting up a Zeek cluster on Ubuntu, emphasizing secure communication and proper configuration for efficient network security monitoring in diverse computing environments.

2 Introduction

A Zeek cluster is a distributed deployment of the Zeek Network Security Monitor, designed to enhance network security monitoring capabilities by leveraging multiple nodes for data collection and analysis. Zeek (formerly known as Bro) is an open-source network security monitoring tool that passively monitors network traffic, extracting meaningful information about network behavior, security events, and potential threats.

In a Zeek cluster, different nodes play specific roles, each contributing to the overall functionality of the monitoring system. The common node types in a Zeek cluster include:

Manager Node: Acts as the central coordinator of the cluster, distributing tasks to worker nodes, managing configurations, and overseeing the overall operation of the Zeek deployment. log files generated by Zeek and provide a centralized repository for analysis.

Worker Nodes: Perform the actual packet analysis and generate logs based on network activity. Worker nodes handle specific tasks assigned by the manager node.

Installation and configuration process of a Zeek cluster on Ubuntu, providing a scalable and distributed network security monitoring solution. Zeek, formerly known as Bro, excels in traffic analysis, protocol detection, and anomaly detection. The cluster setup involves a master node orchestrating multiple worker nodes for parallel processing.

The installation process begins with ensuring system updates and installing essential prerequisites, including compilers and development libraries. Zeek's official repositories are added to the system, and the software is installed using package management tools. The deployment involves configuring various node types, such as logger, manager, proxy, and worker, in the **node.cfg** file.

Secure communication between nodes is established using SSH key pairs. The SSH keys are generated on the master, and the public keys are distributed to worker nodes to enable passwordless authentication. The nodes are configured with appropriate interfaces and IP addresses for network monitoring.

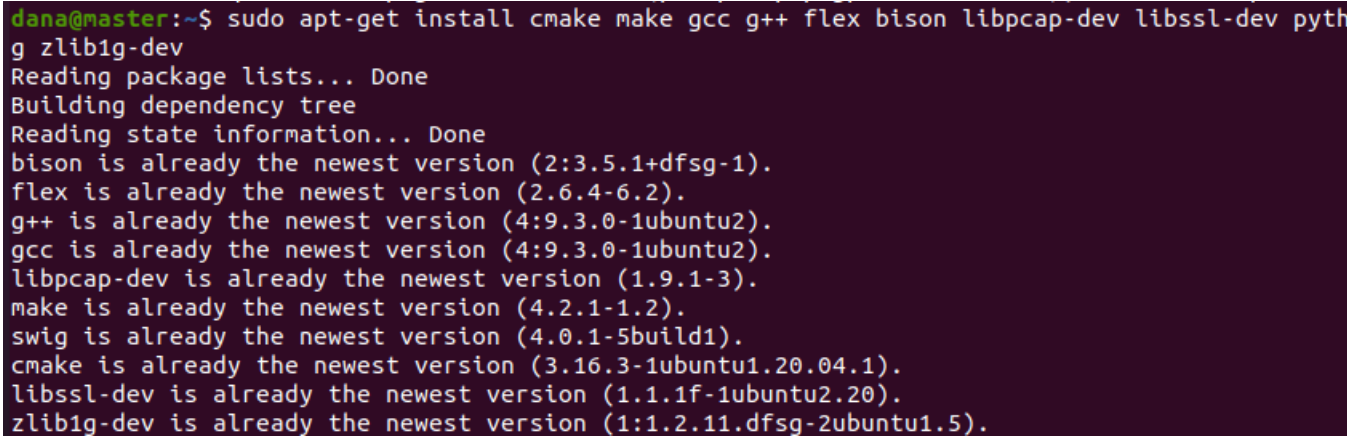
Upon successful SSH setup, the Zeek cluster is deployed using the **zeekctl deploy** command, which installs configurations, policies, and scripts across the cluster. Monitoring logs and outputs are centralized on the master node, facilitating streamlined analysis and management.

3 Procedures

3.1 Install Zeek

The command `sudo apt-get install cmake make gcc g++ flex bison libpcap-dev libssl-dev python2-dev swig zlib1g-dev` is used to install the necessary development tools and libraries required for compiling and building software on a Ubuntu system.

As shown in the Figure(1).



```
dana@master:~$ sudo apt-get install cmake make gcc g++ flex bison libpcap-dev libssl-dev python2-dev swig zlib1g-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
bison is already the newest version (2:3.5.1+dfsg-1).
flex is already the newest version (2.6.4-6.2).
g++ is already the newest version (4:9.3.0-1ubuntu2).
gcc is already the newest version (4:9.3.0-1ubuntu2).
libpcap-dev is already the newest version (1.9.1-3).
make is already the newest version (4.2.1-1.2).
swig is already the newest version (4.0.1-5build1).
cmake is already the newest version (3.16.3-1ubuntu1.20.04.1).
libssl-dev is already the newest version (1.1.1f-1ubuntu2.20).
zlib1g-dev is already the newest version (1:1.2.11.dfsg-2ubuntu1.5).
```

Figure 1: Install Dependencies

cmake: A cross-platform build system that helps manage the build process of software projects.

make: A utility for building and compiling projects based on a Makefile.

gcc and g++: The GNU Compiler Collection for compiling C and C++ programs, respectively.

flex and bison: Tools for generating lexical analyzers and parsers, commonly used in language processing.

libpcap-dev: Development files for the Packet Capture library, used for network traffic capture and analysis.

libssl-dev: Development files for the OpenSSL library, providing cryptographic functions.

python2-dev: Development files for the Python 2 interpreter, often needed for building software with Python 2 support.

swig: A tool that connects programs written in C and C++ with scripting languages like Python, facilitating language integration.

zlib1g-dev: Development files for the zlib compression library, used for data compression.

These commands shown in Figures (2) & (3) are related to adding the GPG (GNU Privacy Guard) key of the Zeek package repository to your system. This is a common method to verify the authenticity of the packages you download from a repository.

```
wget -nv https://download.opensuse.org/repositories/security:/zeek/xUbuntu_22.04/Release.key
```

wget is a command-line utility for downloading files from the web.

-nv stands for "no verbose," which means it won't display unnecessary information during the download.

https://download.opensuse.org/repositories/security:/zeek/xUbuntu_22.04/Release.key is the URL of the GPG key file.

-O Release.key specifies the output file name for the downloaded key, which is named "Release.key" in this case.



```
2023-12-16 21:29:07 URL:https://download.opensuse.org/repositories/security:/zeek/xUbuntu_20.04/Release.key [1084/1084] -> "Release.key" [1]  
dana@master:~$ apt-key add -> Release.key
```

Figure 2: Add Zeek repositories key

```
sudo apt-key add - < Release.key:
```

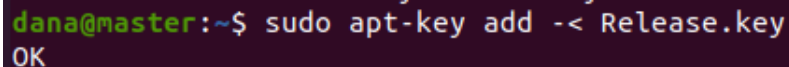
sudo is used to execute the following command with superuser privileges.

apt-key is a command-line tool used to manage authentication keys for repositories.

add instructs apt-key to add a new key to the keyring.

- is used to read the key from the standard input.

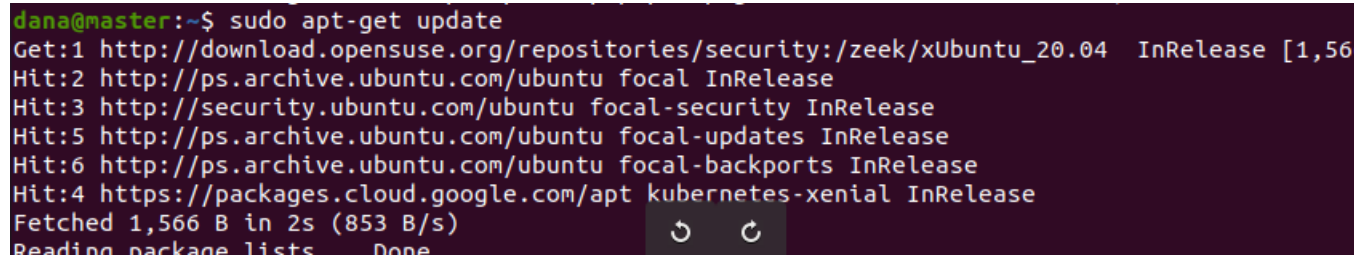
< Release.key redirects the contents of the "Release.key" file to the standard input of the apt-key add command.



```
dana@master:~$ sudo apt-key add -< Release.key  
OK  
dana@master:~$ apt-key update
```

Figure 3: Add Zeek repositories key

Updating the system post adding the **GPG key** ensures that you have the most recent information about available packages, their dependencies, and any security updates. This helps in maintaining a secure, stable, and well-functioning system. Using **sudo apt update** command. As shown in Figure(4).



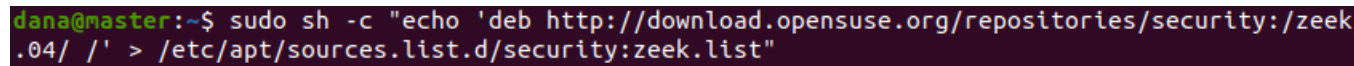
```
dana@master:~$ sudo apt-get update
Get:1 http://download.opensuse.org/repositories/security:/zeek/xUbuntu_20.04 InRelease [1,56
Hit:2 http://ps.archive.ubuntu.com/ubuntu focal InRelease
Hit:3 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:5 http://ps.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:6 http://ps.archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:4 https://packages.cloud.google.com/apt kubernetes-xenial InRelease
Fetched 1,566 B in 2s (853 B/s)
Reading package lists... Done
```

Figure 4: Update package lists

This command

```
sudo sh -c "echo 'deb http://download.opensuse.org/repositories
/security:/zeek/xUbuntu_20.04/ /' > /etc/apt/sources.list.d
/security:zeek.list"
```

Used to add a new software repository configuration file for Zeek to the system. creates a new repository configuration file (**security:zeek.list**) in the **/etc/apt/sources.list.d/** directory. The file includes the repository information for Zeek, specifying the Zeek repository URL and the Ubuntu version (in this case, 20.04). This configuration allows the system package manager (**apt**) to retrieve Zeek packages from the specified repository during the update and installation processes.



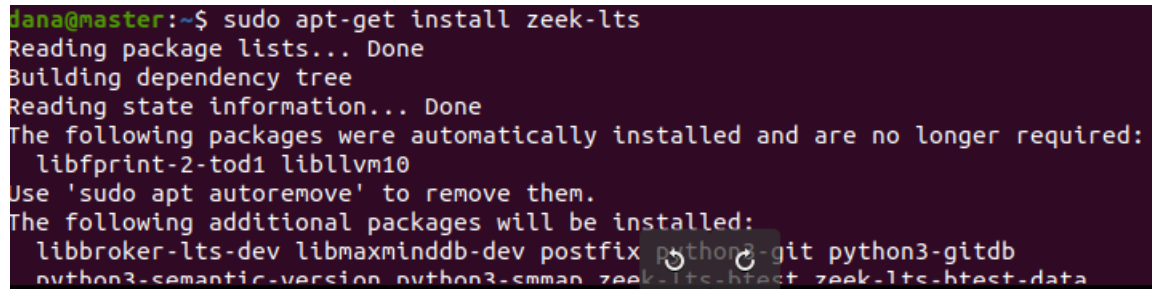
```
dana@master:~$ sudo sh -c "echo 'deb http://download.opensuse.org/repositories/security:/zeek
.04/ /' > /etc/apt/sources.list.d/security:zeek.list"
```

Figure 5: Add Zeek repository

The command `sudo apt-get install zeek-lts` shown in Figure (6), is used to install Zeek on a Debian-based Linux system.

zeek-lts: This is the name of the Zeek package to be installed. The `zeek-lts` package typically represents the long-term support (LTS) version of Zeek, which is a stable and well-supported release.

The package manager will download the Zeek software and its dependencies from the configured repositories that were added before.

A terminal window with a dark background and light-colored text. The prompt is 'dana@master:~\$'. The command entered is 'sudo apt-get install zeek-lts'. The output shows the package manager reading lists, building a dependency tree, and reading state information. It then lists packages to be removed and additional packages to be installed. The additional packages listed are libbroker-lts-dev, libmaxminddb-dev, postfix, python3-git, python3-gitdb, python3-semantic-version, python3-smmap, zeek-lts-btest, zeek-lts-btest-data, and zeek-lts.

```
dana@master:~$ sudo apt-get install zeek-lts
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libfprint-2-tod1 libllvm10
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libbroker-lts-dev libmaxminddb-dev postfix python3-git python3-gitdb
  python3-semantic-version python3-smmap zeek-lts-btest zeek-lts-btest-data
  zeek-lts
```

Figure 6: Install zeek

3.2 Install zeek on Worker node

Updating the system before installing packages. Using **sudo apt-get update** As shown in Figure(7), is a good practice to ensure that the package manager has the latest information about available packages and their versions. It helps prevent compatibility issues and ensures that you're installing the most recent and secure versions of the required software components. This step is especially important when setting up a new system or installing software for the first time.

Install Dependencies for Zeek on the Worker node. As shown in Figure(7)

```
dana@worker:~$ sudo apt-get update
[sudo] password for dana:
daSorry, try again.
[sudo] password for dana:
Sorry, try again.
[sudo] password for dana:
Hit:1 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:2 http://ps.archive.ubuntu.com/ubuntu focal InRelease
Get:3 http://ps.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Hit:4 http://ps.archive.ubuntu.com/ubuntu focal-backports InRelease
Fetched 114 kB in 2s (71.8 kB/s)
Reading package lists... Done
dana@worker:~$ sudo apt-get install cmake make gcc g++ flex bison libpcap-dev libssl-dev python3-zlib-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
g++ is already the newest version (4:9.3.0-1ubuntu2).
g++ set to manually installed.
gcc is already the newest version (4:9.3.0-1ubuntu2).
gcc set to manually installed.
make is already the newest version (4.2.1-1.2).
make set to manually installed.
The following packages were automatically installed and are no longer required:
```

Figure 7: Install Dependencies for zeek on worker node

These commands shown in Figure (8), download the GPG key from the specified URL and then add it to the system's keyring, allowing the package manager (apt) to verify the authenticity of packages from the Zeek repository. This helps ensure that the packages have not been tampered with and are from a trusted source.

And add a new software repository configuration file for Zeek to the worker system.

```
dana@worker:~$ wget -nv https://download.opensuse.org/repositories/security:/zeek/xUbuntu_20.04/Release.key -O Release.key
2023-12-19 19:42:45 URL:https://download.opensuse.org/repositories/security:/zeek/xUbuntu_20.04/Release.key [1084/1084] -> "Release.key" [1]
dana@worker:~$ sudo apt-key add - < Release.key
OK
dana@worker:~$ sudo sh -c "echo 'deb http://download.opensuse.org/repositories/security:/zeek/xUbuntu_20.04/ /' > /etc/apt/sources.list.d/security:zeek.list"
dana@worker:~$ sudo apt-get update
Get:1 http://download.opensuse.org/repositories/security:/zeek/xUbuntu_20.04 InRelease [1,56 kB]
Get:2 http://download.opensuse.org/repositories/security:/zeek/xUbuntu_20.04 Packages [16.0 kB]
```

Figure 8: Add zeek repository on worker node

The command **sudo apt-get install zeek-lts** as shown in Figure(9) is used to install Zeek on a Debian-based Linux system using the package manager **apt**.

```
dana@worker:~$ sudo apt-get install zeek-lts
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libfprint-2-tod1 libllvm10
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libbroker-lts-dev libmaxminddb-dev postfix python3-git python3-gitdb python3-semantic-version
  python3-smmmap zeek-lts-btest zeek-lts-btest-data zeek-lts-client zeek-lts-core zeek-lts-core-dev
  zeek-lts-spicy-dev zeek-lts-zkg zeekctl-lts
Suggested packages:
  procmail postfix-mysql postfix-pgsql postfix-ldap postfix-pcre postfix-lmdb postfix-sqlite3
  | dovecot-common resolvconf postfix-cdb postfix-doc python-git-doc python-semantic-version-doc
  python3-nose
The following NEW packages will be installed:
  libbroker-lts-dev libmaxminddb-dev postfix python3-git python3-gitdb python3-semantic-version
  python3-smmmap zeek-lts zeek-lts-btest zeek-lts-btest-data zeek-lts-client zeek-lts-core
  zeek-lts-core-dev zeek-lts-spicy-dev zeek-lts-zkg zeekctl-lts
0 upgraded, 16 newly installed, 0 to remove and 533 not upgraded.
Need to get 74.6 MB of archives.
After this operation, 572 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Figure 9: Install zeek on worker node

3.3 Establish SSH Connection between master&worker

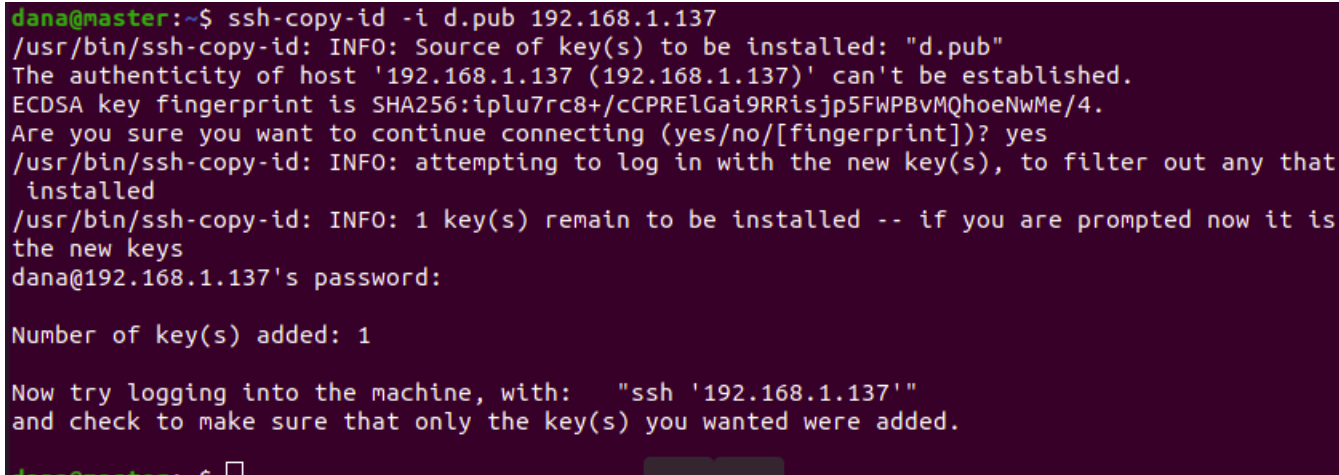
Generate public and private key for ssh using **ssh keygen** command. As shown in Figure(10). SSH keys are used for secure communication between the nodes in the Zeek cluster, allowing the master and worker nodes to authenticate and communicate with each other without the need for password authentication. This is important for the automated deployment and coordination of tasks within the Zeek cluster.

```
dana@master:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/dana/.ssh/id_rsa): d
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in d
Your public key has been saved in d.pub
The key fingerprint is:
SHA256:MbLgq2IVRBPlzPUHlNFtCrgsffevS0JFV0zKbINo dana@master
The key's randomart image is:
+---[RSA 3072]---+
| .+o. ..++ .o. |
| . = . .Bo...o o|
| . .+. B+=oo. = |
| o . o.*E+ o |
| o . S o . |
| . o o o o |
| . . o o + . |
| .. . o . o |
| ... o. |
+-----[SHA256]-----+
```

Figure 10: Generate pair of keys

This command `ssh-copy-id -i d.pub 192.168.1.137` shown in Figure(11). Copies the public key from the master node to the **authorized_keys** file on the worker node.

The **authorized_keys** file is crucial for enabling secure, passwordless communication between nodes. When setting up a Zeek cluster, the public key of the master node is typically added to the **authorized_keys** file on each worker node. This allows the master node to establish SSH connections to the workers without requiring a password, facilitating automated deployment and coordination within the cluster.



```
dana@master:~$ ssh-copy-id -i d.pub 192.168.1.137
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "d.pub"
The authenticity of host '192.168.1.137 (192.168.1.137)' can't be established.
ECDSA key fingerprint is SHA256:iplu7rc8+/cCPRElGai9RRisjp5FWPBvMQhoeNwMe/4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that
installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is
the new keys
dana@192.168.1.137's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh '192.168.1.137'"
and check to make sure that only the key(s) you wanted were added.
```

Figure 11: Copy the public key to worker node

After copying the public key, you should be able to establish an SSH connection from the master node to the worker node without entering a password.

3.4 Setup configuration

Move to `/opt` then to **zeek** directory using `cd` command.

Installing Zeek in `/opt` is a common practice in Unix-like systems. The `/opt` directory is designed to contain software packages that are not part of the operating system's default installation. The name "**opt**" stands for "*optional*," indicating that the contents of this directory are optional add-on software.

```
dana@master:~$ ls /opt/
cni  containerd  VBoxGuestAdditions-6.1.30  zeek
dana@master:~$ cd /opt/
dana@master:/opt$ ls
cni  containerd  VBoxGuestAdditions-6.1.30  zeek
dana@master:/opt$ cd zeek
dana@master:/opt/zeek$ ls
bin  etc  include  lib  logs  share  spool  var
```

Figure 12: `cd /opt/zeek`

In Zeek, the `/bin` directory typically contains executable binaries, including essential command-line tools and scripts.

So we executed `./Zeek -h` command inside it. As shown in Figure(13). It shows all the options for Zeek.

```
dana@master:/opt/zeek$ cd bin/
dana@master:/opt/zeek/bin$ ./zeek -h
zeek version 6.0.2
usage: ./zeek [options] [file ...]
usage: ./zeek --test [doctest-options] -- [options] [file ...]
<file>                                | Zeek script file, or read stdin
-a|--parse-only                        | exit immediately after parsing scripts
-b|--bare-mode                        | don't load scripts from the base/ directory
-c|--capture-unprocessed <file>      | write unprocessed packets to a tcpdump file
-d|--debug-script                     | activate Zeek script debugging
-e|--exec <zeek code>                 | augment loaded scripts by given code
-f|--filter <filter>                  | tcpdump filter
-h|--help                             | command line help
-i|--iface <interface>                | read from given interface (only one allowed)
-p|--prefix <prefix>                  | add given prefix to Zeek script file resolution
-r|--readfile <readfile>              | read from given tcpdump file (only one allowed, pass '-' as the fi
lename to read from stdin)
-s|--rulefile <rulefile>               | read rules from given file
-t|--tracefile <tracefile>             | activate execution tracing
-u|--usage-issues                     | find variable usage issues and exit
    --no-unused-warnings               | suppress warnings of unused functions/hooks/events
-v|--version                          | print version and exit
-V|--build-info                       | print build information and exit
-w|--writefile <writefile>             | write to given tcpdump file
-C|--no-checksums                     | ignore checksums
```

Figure 13: Show help option for zeek

The **node.cfg** file contains the main configuration. It defines the configuration for different nodes in a Zeek cluster, specifying their roles, hosts, and other relevant parameters. The node.cfg file is crucial for orchestrating the deployment and management of a Zeek cluster.

As shown in the Figure(14). It's the master node configuration. It specifies three things:

1. Node Types

2. Ip address or name for the Node

3. Network Interface Manager, logger, and proxy don't need a Network interface, Because the *worker* node listens to the network traffic and redirects the traffic to them.

```
#[logger-1]
#type=logger
#host=localhost
#
[manager]
type=manager
host=localhost
#
[proxy-1]
type=proxy
host=localhost
#
```

Figure 14: Config node.cfg file on master node

As shown in Figure(15). It's a worker node configuration. Network.cfg file to add the

```
#
[worker-1]
type=worker
host=localhost
interface=enp0s3
#
```

Figure 15: Config node.cfg file on worker node

Networks CIDR notation that zeek cluster work inside it, or listen to their traffic.

```
GNU nano 4.8                                networks.cfg
# List of local networks in CIDR notation, optionally followed by a descriptive
# tag. Private address space defined by Zeek's Site::private_address_space set
# (see scripts/base/utils/site.zeek) is automatically considered local. You can
# disable this auto-inclusion by setting zeekctl's PrivateAddressSpaceIsLocal
# option to 0.
#
# Examples of valid prefixes:
#
# 1.2.3.0/24           Admin network
# 2607:f140::/32       Student network
192.168.0.0/16         Private IP space
```

Figure 16: Config network.cfg file o

3.5 Install and deploy zeekctl

Install zeekctl tool using **sudo ./zeekctl install** command. As shown in Figure(17).

zeekctl is a control framework for managing Zeek installations. It provides a set of scripts and commands that allow you to deploy, configure, and control Zeek instances. Some of the common tasks zeekctl helps with include starting and stopping Zeek, deploying configurations, managing clusters, and installing additional components.

```
dana@master:/opt/zeek/bin$ sudo ./zeekctl install
removing old policies in /opt/zeek/spool/installed-scripts-do-not-touch/site ...
removing old policies in /opt/zeek/spool/installed-scripts-do-not-touch/auto ...
creating policy directories ...
installing site policies ...
generating standalone-layout.zeek ...
generating local-networks.zeek ...
generating zeekctl-config.zeek ...
generating zeekctl-config.sh ...
```

Figure 17: Install zeekctl tool

Deploy the new configuration using **sudo ./zeekctl deploy** command. As shown in Figure(18).

```
dana@master:/opt/zeek/bin$ sudo ./zeekctl deploy
checking configurations ...
installing ...
removing old policies in /opt/zeek/spool/installed-scripts-do-not-touch/site ...
removing old policies in /opt/zeek/spool/installed-scripts-do-not-touch/auto ...
creating policy directories ...
installing site policies ...
generating standalone-layout.zeek ...
generating local-networks.zeek ...
generating zeekctl-config.zeek ...
generating zeekctl-config.sh ...
stopping ...
```

Figure 18: Deploy and activate the Zeek components

Start zeekctl using **sudo ./zeekctl start** command. As shown in Figure(19).

Check the status of zeek using **sudo ./zeekctl status** command. As shown in Figure(19).

All these commands should execute in **/opt/zeek/bin**. As shown in Figures(19).(18).(16)

```
root@master:/etc/postfix# sudo /opt/zeek/bin/zeekctl start
starting logger ...
starting manager ...
starting proxy ...
starting worker ...
root@master:/etc/postfix# sudo /opt/zeek/bin/zeekctl status
```

Name	Type	Host	Status	Pid	Started
logger-1	logger	192.168.1.13	running	4266	20 Dec 21:13:19
manager	manager	192.168.1.13	running	4474	20 Dec 21:13:21

Figure 19: Check zeekctl status

3.6 Verify zeek installation

Check the logs are located in `/opt/zeek/logs/current`. Notice that at the beginning it 4 logs, and after `curl google.com`. New logs appear according to the traffic in the network. Zeek might be configured to rotate logs based on certain criteria, such as size or time, If the logs are reaching a certain size or age, they might be automatically removed during log rotation. This is a common behavior to prevent log directories from filling up.

Some logs might only be created when specific events occur. For example, the `http.log` file is generated when Zeek detects HTTP traffic. If there's no HTTP traffic during a certain period, the log might not be created.

```
root@worker:/opt/zeek/logs/current# ls
conn.log dns.log stderr.log stdout.log
root@worker:/opt/zeek/logs/current# curl google.com
<HTML><HEAD><meta http-equiv="content-type" content="text/html; charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.com/">here</A>.
</BODY></HTML>
root@worker:/opt/zeek/logs/current# ls
conn.log files.log stats.log stdout.log weird.log
dns.log http.log stderr.log telemetry.log
root@worker:/opt/zeek/logs/current#
```

Figure 20: Test logs

To listen for the type of traffic according to logs type, use `tail -f /opt/zeek/logs/current/type.log`. As shown in Figure (21).

```
dana@master:/opt/zeek/logs$ sudo tail -f /opt/zeek/logs/current/http.log
#empty_field (empty)
#unset_field -
#path http
#open 2023-12-16-22-09-54
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p trans_depth m
ethod host uri referrer version user_agent origin request_body_len response
_body_len status_code status_msg info_code info_msg tags username p
assword proxied orig_fuids orig_filenames orig_mime_types resp_fuids resp_filenames resp_mime
e_types
#types time string addr port addr port count string string string string string s
tring string count count count string count string set[enum] string string set[stri
```

Figure 21: Start capturing packets 'http.log'

4 Conclusion

In conclusion, installing and configuring a Zeek cluster on Ubuntu involves several key steps. Beginning with dependency installation and system updates, the process includes repository setup, Zeek installation in `/opt`, and cluster deployment using `zeekctl`. Security considerations, such as SSH key management, are crucial. Configuration files like `node.cfg` and `network.cfg` defines the cluster layout. Regular log monitoring is essential for network activity and issue detection. Successful deployment requires attention to detail, understanding of network architecture, and adherence to security measures, enabling effective threat detection and response.

5 References

<https://docs.zeek.org/en/master/cluster-setup.html>

<https://docs.zeek.org/en/master/frameworks/cluster.html>