virtual zeek week
13-15 October 2020
Towards a New Management Framework for Zeek Clusters

Robin Sommer
Corelight, Inc.
robin@corelight.com
Back in 2007 ...

The NIDS Cluster: Scalable, Stateful Network Intrusion Detection on Commodity Hardware

Matthias Vallentin\textsuperscript{3}, Robin Sommer\textsuperscript{2,1}, Jason Lee\textsuperscript{2}, Craig Leres\textsuperscript{2}, Vern Paxson\textsuperscript{1,2}, and Brian Tierney\textsuperscript{2}

\textsuperscript{1} International Computer Science Institute
\textsuperscript{2} Lawrence Berkeley National Laboratory
\textsuperscript{3} TU München

\textbf{Abstract.} In this work we present a \textit{NIDS cluster} as a scalable solution for realizing high-performance, stateful network intrusion detection on commodity hardware. The design addresses three challenges: (i) distributing traffic evenly across an extensible set of analysis nodes in a fashion that minimizes the communication required for coordination, (ii) adapting the NIDS’s operation to support coordinating its \textit{low-level} analysis rather than just aggregating alerts; and (iii) validating that the cluster produces sound results. Prototypes of our NIDS cluster now operate at the Lawrence Berkeley National Laboratory and the University of California at Berkeley. In both environments the clusters greatly enhance the power of the network security monitoring.
Back in 2007 …
Back in 2007 …

# cluster --help

BroCluster Shell Version 0.1

check <nodes>  - check configuration before installing them
cleanup <nodes> - delete old working directories on nodes (flushes state)
config        - print cluster configuration
cron          - perform jobs intended to run regularly from cron
diag <nodes>  - output Bro's stderr on node and other diagnostics
exit          - exit shell
install       - update cluster installation/configuration
nodes         - print node configuration
quit          - exit shell
restart <nodes> - stop and then restart processing
start <nodes>  - start processing
status <nodes> - summarize node status
stop <nodes>   - stop processing
top <nodes>    - show Bro processes ala top

See the cluster's Wiki page for more information:

http://www.bro-ids.org/wiki/BroCluster

Send questions to the Bro mailing list at bro@bro-ids.org.
Back in 2007 ...

Date: Fri, 22 Jun 2007 20:36:52 -0700 (PDT)
From: robin@icir.org
To: bro-dev@george.lbl.gov
Subject: [Bro-dev] [svn] r4480 - in branches/robin/work/aux/cluster: BroCluster policy
Message-Id: <200706230336.15N3aq55003539@portnoy.lbl.gov>

Log:
Checkpoint; starting documentation

Modified: branches/robin/work/aux/cluster/README

===============================================
--- branches/robin/work/aux/cluster/README 2007-06-23 02:34:41 UTC (rev 4479)
+++ branches/robin/work/aux/cluster/README 2007-06-23 03:36:52 UTC (rev 4480)
@@ -1,4 +1,127 @@
-Documentation to come once it works. :-)
+The Bro Cluster
+===============================================
+
+This document explains how to setup a Bro Cluster, i.e., a set of
+PCs jointly analyzing a network link. See
+http://www.icir.org/robin/papers/raid0.pdf for more information
+about the general architecture.
+
+A cluster consists of four kinds of components:
+
+  1. One or more frontends.
+     Frontends load-balance the traffic across a set of machines.
+  
+  2. One manager.
+     The manager provides the cluster's user-interface for
+     control and log aggregation.
+     
+  3. A number of workers.
+     Workers are doing the actual analysis, each seeing a slice
+     of the overall traffic as splitted up by the frontends.
+     
+  4. One or more proxies
+     Proxies relay the communication between worker nodes.
Back in 2007 ...

CHANGES

1.5 Wed Dec 16 21:28:47 PST 2009

- Bro now comes with a new framework, BroControl, for managing an operational Bro setup, including support for installation, configuration, and maintenance tasks such as log archival and mail notification. The framework transparently supports both traditional standalone setups as well as cluster installations in which multiple Bro boxes coordinate to analyze a high-volume network link.

See aux/broctl/README for more information about BroControl.

Note, BroControl supersedes the older BroLite system, which is no longer supported and has been deprecated for a while now.
In 2020, BreZeekControl is still there ...
In 2020, BreZeekControl is still there …

broctl looks like an APT #259

mpurzynski opened this issue on Jan 30, 2019 · 8 comments

mpurzynski commented on Jan 30, 2019 · edited

While doing Threat Hunting I saw an interesting string in our Audit Execve logs, only on my NSM fleet

```
python /usr/bin/python -c import zlib,base64;
exec(zlib.decompress(base64.b64decode("bEjxL8Fu2zAMPUdfIehSGdOMJtgpgi459DCsh+7kGYFr060KwIke05Q9fHsvaANAF1w+Tj1/LIMdMPzo/cBRXO+J+eB+9qCPhpXmzVq0Ad1KN6C86yp4efh1+/n/R2d89YaYz2fC0j2b1ORegbN4357M01sPq0cW/cfHIuwK0XZTeJVZIoAT1Me6b4KkB3EUnij2pW6G4U2iwH0/Xgk+RdQZVn+m1oWL129H5oxvAEkyIbFe+hBcDmdh7r2aeRbK86jC8kdIO8mZkhVMNw8P8Wnc5VXgEHPPhJNbdXeHrF5Chj+28TB03vKBSvKvKD1vXzv5r1/eVbLO1lnTNq67yVwUrzR3I3kL1C8qi0o1xYzNiYrcSiT0AmUJ6VfgtW5aT965CpUKLfbCqSTgLxReQrcCjvn21aDfP1h78RV8gE8nk8SnKo0mBGp9+FaU5ibxQXZBR7euIHSS2RZk0KPBymCszGvs9rsPnG1MVQDQguldUsXd4AkGFECMclrq/UNzaUyQRhwaGtceRGjLnAK1PQzoeAix1690qqjTkufp5dEj8J2ihLHS9q0cSs9LQGyDbhLLsQpITJvcs6mant/e4HLZBpeULFpVL6bJyynZyLzriAtbMo1wQMD91TxvCNfQoGw+9+wuYh/DoLv5LKHZ7X25Tt2hyzXhLKhXMcYjEHMZqnlKOsIaG7KsTB8Lkb85E2/U56TtcbLmd0ONyIvXo10o30pUz8CoNVLjW013Gb8W78A22ieEg=))))
```
In 2020, BreZeekControl is still there …
In 2020, BreZeekControl is still there …

The loose coupling between ZeekControl and Zeek itself limits both functionality & integration with the OS.
In 2020, BreZeekControl is still there …

The loose coupling between ZeekControl and Zeek itself limits both functionality & integration with the OS.

- Distributed installation & management through rsync+ssh
- Awkward fit with our package manager
- Process supervision through cron prevents fast recovery
- Limited health monitoring capabilities
- No dynamic adjustments to configuration
- No parallel offline processing of traces
- Multi-system focus doesn’t match *most* settings (but still *some*)!
- Difficult adapt to new use cases & deployment models
Towards a new cluster management framework
Towards a new cluster management framework

Observation: We have a data plane, but no real control plane.
Towards a new cluster management framework

Observation: We have a data plane, but no real control plane.
Idea: Implement cluster management inside Zeek itself.
Towards a new cluster management framework

Observation: We have a data plane, but no real control plane.
Idea: Implement cluster management inside Zeek itself.

Objectives:
- Support single-/multi-system setups
- Build on top of the current script-level cluster framework
- Keep offering a CLI user interface
- Be a good citizen with the operating system
- Accommodate new deployment models (containers)
Proposed architecture
Proposed architecture

User interface (zeekc)

Command line client
Proposed architecture

Cluster Controller

Maintains & persists desired state of cluster

API

Command line client

User interface (zeekc)
Proposed architecture

Cluster Agent
- Carries out management operations

Cluster Controller
- Maintains & persists desired state of cluster

Command line client
- User interface (zeekc)
Proposed architecture

- **Process Supervisor**: Watches & controls processes
- **Cluster Agent**: Carries out management operations
- **Cluster Controller**: Maintains & persists desired state of cluster
- **Command line client**: User interface (\texttt{zeekc})
Proposed architecture

- **Workers**
  - Watches & controls processes
- **Logger**
  - Carries out management operations
- **Manager**
  - Maintains & persists desired state of cluster
- **Process Supervisor**
- **Cluster Agent**
- **Cluster Controller**
- **Command line client**
- **User interface (zeekc)**
Proposed architecture

zeek

- Workers
- Logger
- Manager

Process Supervisor
- API

Cluster Agent
- API

Cluster Controller
- API

Command line client

Standard cluster node processes

Watches & controls processes

Carries out management operations

Maintains & persists desired state of cluster

User interface (zeekc)
Proposed architecture

systemd

zeek

Workers, Logger, Manager

Process Supervisor

Cluster Agent

Cluster Controller

Command line client

Standard cluster node processes

Watches & controls processes

Carries out management operations

Maintains & persists desired state of cluster

User interface (zeekc)
**Proposed architecture**

- **systemd**
- **zeek**
- **Watches & controls processes**
- **Process Supervisor**
  - API
- **Cluster Agent**
  - API
- **Cluster Controller**
  - API
- **Workers**
- **Logger**
- **Manager**
- **Data plane**
- **Control plane**
- **Standard cluster node processes**
- **Carries out management operations**
- **Maintains & persists desired state of cluster**
- **User interface (zeekc)**

**Zeek scripts**

**Events/Broker**
Proposed architecture

- systemd
- zeek
- Workers
- Logger
- Manager
- Process Supervisor
- Cluster Agent
- Cluster Controller
- Command line client
Proposed architecture

systemd

zeek

Workers | Logger | Manager

Process Supervisor

Cluster Agent

Cluster Controller

Command line client
Proposed architecture

- systemd
- zeek
- Process Supervisor
- API
- Cluster Agent
- Workers
- Manager
- API
- zeek
- systemd
- zeek
- Process Supervisor
- API
- Cluster Agent
- Workers
- Logger
- API
Proposed architecture

- **systemd**
  - **Process Supervisor**
    - **Cluster Agent**
  - **Cluster Controller**
  - **Command line client**

- **zeek**
  - **Workers**
  - **Manager**
  - **Logger**
Usage example 1: single system, live cluster
Usage example 1: single system, live cluster

1. Install Zeek (e.g., RPM)
Usage example 1: single system, live cluster

1. Install Zeek (e.g., RPM)

2. Install desired Zeek packages (`zkg`) and adjust `local.zeek`
Usage example 1: single system, live cluster

1. Install Zeek (e.g., RPM)
2. Install desired Zeek packages (`zkg`) and adjust `local.zeek`
3. Configure `systemd` to spawn (*)
   
   ```
   zeek -j framework/cluster/controller framework/cluster/agent
   ```

(*) In practice, the RPM will probably do that for you.
Usage example 1: single system, live cluster

1. Install Zeek (e.g., RPM)
2. Install desired Zeek packages (zkg) and adjust local.zeek
3. Configure systemd to spawn (*)
   ```
   zeek -j framework/cluster/controller framework/cluster/agent
   ```
4. Create `zeek.cfg` (number of workers, interfaces, etc.)

* In practice, the RPM will probably do that for you.
Usage example 1: single system, live cluster

1. Install Zeek (e.g., RPM)
2. Install desired Zeek packages (zkg) and adjust local.zeek
3. Configure systemd to spawn (*)
   
   zeek -j framework/cluster/controller framework/cluster/agent

4. Create zeek.cfg (number of workers, interfaces, etc.)
5. Run zeekc –config zeek.cfg update

* In practice, the RPM will probably do that for you.
Usage example 1: single system, live cluster

1. Install Zeek (e.g., RPM)
2. Install desired Zeek packages (zkg) and adjust local.zeek
3. Configure systemd to spawn (*)
   
   ```
   zeek -j framework/cluster/controller framework/cluster/agent
   ```
4. Create `zeek.cfg` (number of workers, interfaces, etc.)
5. Run `zeekc -config zeek.cfg update`
6. `zeekc start`

* In practice, the RPM will probably do that for you.
Usage example 2: multiple systems, live cluster

* In practice, the RPM will probably do that for you.
Usage example 2: multiple systems, live cluster

1. Install Zeek on all systems (e.g., RPM)

* In practice, the RPM will probably do that for you.
Usage example 2: multiple systems, live cluster

1. Install Zeek on all systems (e.g., RPM)
2. Install desired Zeek packages on all systems (zkg) and adjust local.zeek

* In practice, the RPM will probably do that for you.
Usage example 2: multiple systems, live cluster

1. Install Zeek on all systems (e.g., RPM)
2. Install desired Zeek packages on all systems (zkg) and adjust local.zeek
3. On one system, configure systemd to spawn a Zeek running the central controller: (*)
   ```
   zeek -j framework/cluster/controller
   ```

* In practice, the RPM will probably do that for you.
Usage example 2: multiple systems, live cluster

1. Install Zeek on all systems (e.g., RPM)
2. Install desired Zeek packages on all systems (zkg) and adjust local.zeek
3. On one system, configure systemd to spawn a Zeek running the central controller: (*)
   \[
   \text{zeek} \ -j \ \text{framework/cluster/controller}
   \]
4. On all systems to run workers/logger/manager, configure systemd to spawn an agent: (*)
   \[
   \text{zeek} \ -j \ \text{framework/cluster/agent}
   \]

* In practice, the RPM will probably do that for you.
Usage example 2: multiple systems, live cluster

1. Install Zeek on all systems (e.g., RPM)

2. Install desired Zeek packages on all systems (zkg) and adjust local.zeek

3. On one system, configure systemd to spawn a Zeek running the central controller: (*)
   
   ```
   zeek -j framework/cluster/controller
   ```

4. On all systems to run workers/logger/manager, configure systemd to spawn an agent: (*)
   
   ```
   zeek -j framework/cluster/agent
   ```

5. Create `zeek.cfg` (cluster systems, layout of nodes, interfaces, etc.)

   (*) In practice, the RPM will probably do that for you.
Usage example 2: multiple systems, live cluster

1. Install Zeek on *all* systems (e.g., RPM)
2. Install desired Zeek packages on *all* systems (zkg) and adjust `local.zeek`
3. On one system, configure `systemd` to spawn a Zeek running the central controller: (*)
   ```
   zeek -j framework/cluster/controller
   ```
4. On all systems to run workers/logger/manager, configure `systemd` to spawn an agent: (*)
   ```
   zeek -j framework/cluster/agent
   ```
5. Create `zeek.cfg` (cluster systems, layout of nodes, interfaces, etc.)
6. Run `zeekc -config zeek.cfg update`

* In practice, the RPM will probably do that for you.
Usage example 2: multiple systems, live cluster

1. Install Zeek on all systems (e.g., RPM)
2. Install desired Zeek packages on all systems (zkg) and adjust local.zeek
3. On one system, configure systemd to spawn a Zeek running the central controller: (*)
   
   ```
   zeek -j framework/cluster/controller
   ```
4. On all systems to run workers/logger/manager, configure systemd to spawn an agent: (*)
   
   ```
   zeek -j framework/cluster/agent
   ```
5. Create zeek.cfg (cluster systems, layout of nodes, interfaces, etc.)
6. Run zeekc -config zeek.cfg update
7. zeekc start

* In practice, the RPM will probably do that for you.
Usage example 3: ad-hoc processing ("easy mode")
Usage example 3: ad-hoc processing ("easy mode")

1. Install Zeek as usual (e.g., RPM)
Usage example 3: ad-hoc processing ("easy mode")

1. Install Zeek as usual (e.g., RPM)
2. Install desired Zeek packages (zkg) and adjust local.zeek
Usage example 3: ad-hoc processing (“easy mode”)

1. Install Zeek as usual (e.g., RPM)

2. Install desired Zeek packages (zkg) and adjust `local.zeek`

3. For live analysis with $N$ workers, run:

   ```bash
   # zeek -j N -i eth0
   ```
Usage example 3: ad-hoc processing ("easy mode")

1. Install Zeek as usual (e.g., RPM)
2. Install desired Zeek packages (zkg) and adjust `local.zeek`
3. For live analysis with $N$ workers, run:
   
   ```bash
   # zeek -j N -i eth0
   ```
4. For offline (trace) analysis with $N$ workers, run:
   
   ```bash
   # zeek -j N -r trace.pcap
   ```
Usage example 3: ad-hoc processing (“easy mode”)

1. Install Zeek as usual (e.g., RPM)
2. Install desired Zeek packages (zkg) and adjust `local.zeek`
3. For live analysis with $N$ workers, run:
   ```sh
   # zeek -j N -i eth0
   ```
4. For offline (trace) analysis with $N$ workers, run:
   ```sh
   # zeek -j N -r trace.pcap
   ```

Implementation: These modes will skip the Controller & instead configure Agent directly.
Current state

First version of the Supervisor landed in Zeek 3.1

```zeek
event zeek_init()
{
    if ( Supervisor::is_supervisor() )
    {
        local sn = Supervisor::NodeConfig($name="foo", $interface="en0");
        local res = Supervisor::create(sn);

        if ( res == "" )
            print "supervisor created a new node";
        else
            print "supervisor failed to create node", res;
    }
    else
        print fmt("supervised node '%s' zeek_init()", Supervisor::node()->$name);
}
```

From https://docs.zeek.org/en/current/frameworks/supervisor.html
Current state

Factored out log rotation into standalone service

zeek-archiver

A Zeek log archival service.

This tool is derived from bro-atomic-rotate and intends to solve the same failings of Zeek's historical log-archival process: robustness and atomicity. It's rewritten with two further requirements in mind:

- Independence from ZeekControl. It's meant for use in conjunction with the upcoming Zeek Supervisor Framework.
- Independence from Python. In retrospect, it's dubious whether that's a benefit: implementing in C++ has little advantage over a solution done in a simpler Bash/Python script, so it may get completely rewritten later if any concrete maintenance burden/problems are encountered.

https://github.com/zeek/zeek-archiver
Current state

Initial design of Cluster Agent/Controller ready for feedback.

Ticket: https://github.com/zeek/zeek/issues/582

Draft design document: https://tiny.cc/zeekc (Google Doc)
## Timeline

<table>
<thead>
<tr>
<th>Feature</th>
<th>ZeekControl supports</th>
<th>New framework will support</th>
<th>3.1 (done)</th>
<th>3.2 (done)</th>
<th>4.1</th>
<th>4.2</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spawn and monitor cluster processes</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configuration through *.cfg files</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>Until we support this, configuration will work through Zeek scripts</td>
</tr>
<tr>
<td>Standalone mode</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cluster mode</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command-line client</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log rotation</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log archival</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log expiration</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capturing Zeek's output</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>Zeek can now send stdout output to log files; stderr is up to OS service</td>
</tr>
<tr>
<td>Crash reports</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>Need to develop a new mechanism for this</td>
</tr>
<tr>
<td>trace-summary report</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>Implement as an optional package</td>
</tr>
<tr>
<td>stats.log</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>Implement inside Zeek</td>
</tr>
<tr>
<td>OS health monitoring (disk space etc.)</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>Deployment model changes</td>
</tr>
<tr>
<td>Multi-system clusters</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sync Zeek installation across systems</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>We will no longer copy Zeek binary &amp; library over to all the systems as</td>
</tr>
<tr>
<td>Sync scripts/packages across systems</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>Not quite sure about this; we'd need to go through Broker; needs some</td>
</tr>
<tr>
<td>Plugins</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>Just Zeek packages through cluster framework hooks</td>
</tr>
<tr>
<td>Jailing of spawned processes</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>Zeek C++ plugin API extension</td>
</tr>
<tr>
<td>Parallel pcap mode (-j)</td>
<td></td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZeekControl supports</td>
<td>New framework will support</td>
<td>3.1 (done)</td>
<td>3.2 (done)</td>
<td>4.1</td>
<td>4.2</td>
<td>Notes</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------</td>
<td>------------</td>
<td>------------</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>Spawn and monitor cluster processes</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td>Until we support this, configuration will work through Zeek scripts</td>
<td></td>
</tr>
<tr>
<td>Configuration through *.cfg files</td>
<td>✔</td>
<td></td>
<td>☐</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standalone mode</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cluster mode</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command-line client</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log rotation</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log archival</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Log expiration</td>
<td>✔</td>
<td></td>
<td>☐</td>
<td></td>
<td></td>
<td>Use standard system log management instead</td>
<td></td>
</tr>
<tr>
<td>Capturing Zeek's output</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td>Zeek can now send stdout output to log files; stderr is up to OS service</td>
<td></td>
</tr>
<tr>
<td>Crash reports</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Need to develop a new mechanism for this</td>
<td></td>
</tr>
<tr>
<td>trace-summary report</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implement as an optional package</td>
<td></td>
</tr>
<tr>
<td>stats.log</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Implement inside Zeek</td>
<td></td>
</tr>
<tr>
<td>OS health monitoring (disk space etc.)</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-system clusters</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sync Zeek installation across systems</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sync scripts/packages across systems</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td>Not quite sure about this; we'd need to go through Broker; needs some</td>
<td></td>
</tr>
<tr>
<td>Plugins</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Just Zeek packages through cluster framework hooks</td>
<td></td>
</tr>
<tr>
<td>Jailing of spawned processes</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td>Zeek C++ plugin API extension</td>
<td></td>
</tr>
<tr>
<td>Parallel pcap mode ((\pm))</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Wrap-up
Wrap-up

This is a major re-architecture of Zeek’s management plane that makes a few trade-offs along the way.
Wrap-up

This is a major re-architecture of Zeek’s management plane that makes a few trade-offs along the way.

The goal is to build a new default way of installing Zeek. It will not be the only way of doing so.
Wrap-up

This is a major re-architecture of Zeek’s management plane that makes a few trade-offs along the way.

The goal is to build a new default way of installing Zeek. It will not be the only way of doing so.

We will explicitly provide hooks (APIs) at different levels to accommodate other deployment models as well.
Wrap-up

This is a major re-architecture of Zeek’s management plane that makes a few trade-offs along the way.

The goal is to build a new default way of installing Zeek. It will not be the only way of doing so.

We will explicitly provide hooks (APIs) at different levels to accommodate other deployment models as well.

That said, we are still in the process of understanding if this is the way to go, and we are looking for feedback.

https://github.com/zeek/zeek/issues/582
zeek-dev@lists.zeek.org
#development
Stay Connected

Website - Zeek.org

Mailing List - zeek@lists.zeek.org


Find out more ways to connect at: https://zeek.org/community/