Generating representative, live network traffic out of millions of code repositories

Tobias Bühler, Roland Schmid, Sandro Lutz, Laurent Vanbever
ETH Zürich nsg.ee.ethz.ch

ACM HotNets
Nov 14 2022
Today, we only have a few gold nuggets of network data available
We believe there exists an entire **gold mine/pile** of network data

Picture: https://labs.openai.com/s/zD0NTe1h8FPjPCsijSkZvKMy
We believe there exists an entire gold mine/pile of network data
In order to tap into this gold mine, we have to **bridge the gap** from static text/code to actual network data.
In order to tap into this gold mine, we have to bridge the gap from static text/code to actual network data.

**Static code analysis**
- Analyze usage of network functions
- Extracts high-level traffic insights

**Running the code**
- Compile and run each open-source project
- Generates live traffic which reacts to network events

???
- The next crazy idea
In order to tap into this gold mine, we have to bridge the gap from static text/code to actual network data.

<table>
<thead>
<tr>
<th>Static code analysis</th>
<th>Analyze usage of network functions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Extracts high-level traffic insights</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Running the code</th>
<th>Compile and run each open-source project</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Generates live traffic which reacts to network events</td>
</tr>
</tbody>
</table>

???

The next crazy idea
However, executing arbitrary open-source projects is challenging.

<table>
<thead>
<tr>
<th>Arbitrary code</th>
<th>How do we build the projects?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Arbitrary code, language and APIs</td>
</tr>
<tr>
<td>Missing documentation</td>
<td>How do we run the projects?</td>
</tr>
<tr>
<td></td>
<td>Missing commands, dependencies and support</td>
</tr>
<tr>
<td>Unexpected errors</td>
<td>How do we handle bugs and errors?</td>
</tr>
<tr>
<td></td>
<td>Unexpected crashes, inputs and runtime</td>
</tr>
</tbody>
</table>
We leverage the rise of automation frameworks which allow to compile and run arbitrary code.
We leverage the rise of automation frameworks which allow to compile and run arbitrary code.

Docker containers

Are a standalone, executable package

Contain all the code and its dependencies
We leverage the rise of automation frameworks which allow to compile and run arbitrary code

<table>
<thead>
<tr>
<th>Docker containers</th>
<th>Are a standalone, executable package</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Contain all the code and its dependencies</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Orchestration files</th>
<th>Define how multiple containers are configured</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A single command builds and starts all of them</td>
</tr>
</tbody>
</table>
Our vision is to combine big data and container solutions to generate representative, live network traffic.
Our vision is to combine big data and container solutions to generate representative, live network traffic with respect to a given user specification.
Our vision is to combine **big data** and **container** solutions to generate **representative, live network traffic** with respect to a given user specification traffic/applications that react to network events.
Input

high-level traffic specification

DYNAMO

orchestration

Output

representative, live network traffic
The input to DYNAMO's live traffic generation process is a high-level traffic specification. This specification leverages the abundance of open-source projects to build the Big Traffic database. Based on a user's traffic demands, DYNAMO finds and orchestrates adequate open-source projects for live traffic generation. After parsing the input, DYNAMO executes each project individually in an isolated virtual machine and captures all generated traffic. We process the recorded traffic traces in two ways: (i) we extract several virtual interfaces associated with hosts, and (ii) we analyze the behavior of the open-source project to record traffic features (e.g., application type, number of flows, etc.).

Finally, we use the Big Traffic database to find matching projects. As illustrated in Figure 2, a set of statements in Declarative Traffic Specification Language (DTSL) allows easy, yet flexible traffic specification. The DTSL parser infers the syntax definition and an example. The DTSL input is then parsed, and DYNAMO queries the Big Traffic database, or explicitly specifies a desired destination port. DYNAMO then finds and orchestrates adequate open-source projects to match the user's traffic demands as closely as possible. To that end, DYNAMO uses a generic solver, e.g., the Gurobi Optimizer. The challenge herein constitutes in matching the high-level traffic specification to features captured in the Big Traffic database. After solving the optimization, DYNAMO generates the traffic using the configured 7 flows. The running projects are #5, #7, and #18. We observe a database traffic from `h3` to `external` with 50 Mbps, using 5 flows.

generate database traffic
from h3 to external
with 50 Mbps
using ≥ 5 flows;
The input to DYNAMO, containing traffic statistics and meta information, is a high-level traffic specification. The DTSL parser infers the syntax definition and an example. The DTSL parser infers the syntax definition and an example. The DTSL parser infers the syntax definition and an example. The DTSL parser infers the syntax definition and an example.

Figure 2: A set of statements in Declarative Traffic Specification Language (DTSL). See Figures 1 and 2 for the specification, that is, a set of statements in the Declarative Traffic Specification Language (DTSL). See Figures 1 and 2 for the specification, that is, a set of statements in the Declarative Traffic Specification Language (DTSL). See Figures 1 and 2 for the specification, that is, a set of statements in the Declarative Traffic Specification Language (DTSL). See Figures 1 and 2 for the specification, that is, a set of statements in the Declarative Traffic Specification Language (DTSL).

2.1 Overview

The lower block outlines DYNAMO's main components divided into

- Project Search
- Traffic Specification
- Traffic Statistics & Behavior Analysis
- Project Execution
- Orchestration

After parsing the input, DYNAMO creates a set of virtual network interfaces associated with hosts and (ii) we analyze the behavior of the open-source project to identify which components send and receive which type of traffic features (e.g., application type, number of flows, etc.); recorded traffic traces in two ways: (i) we extract several traffic-generating projects which could potentially generate network traffic. We then execute each project individually in an isolated virtual environment. As illustrated, DYNAMO leverages the abundance of open-source projects to build the Big Traffic database. Based on a user's demands, Next, the same or different projects to meet each of the user's traffic demands as specified, the Big Traffic database to find matching projects. As illustrated, DYNAMO's main components divided into

- Project Search
- Traffic Specification
- Traffic Statistics & Behavior Analysis
- Project Execution
- Orchestration

generate database traffic from h3 to external with 50 Mbps using >= 5 flows;

2.2 System Details

...
**2.1 Overview**

**2.2 System Details**

**Figure 1:** The lower block outlines the offline phase, which prepares the applications for traffic generation. The upper block depicts the online phase, which orchestrates the execution of the selected open-source projects for live traffic generation.

**Figure 2:** A set of statements in the Declarative Traffic Specification Language (DTSL) allows to easily, yet flexibly specify what traffic needs to be generated by DYNAMO. The DTSL parser infers the traffic specification to features captured in the Big Traffic database.

**Input**

- high-level traffic specification

**DYNAMO**

- container orchestration
- traffic-generating projects

**Output**

- representative, live network traffic for physical or virtual networks
  - test security apps with realistic background traffic

---

*generate database traffic from h3 to external with 50 Mbps using ≥ 5 flows*;
DYNAMO first searches for *traffic-generating* projects
DYNAMO first searches for traffic-generating projects

338M repositories available
DYNAMO first searches for traffic-generating projects

338M repositories
found via GitHub API queries

> 2M orchestration files
available
orchestrable
DYNAMO first searches for traffic-generating projects

- 338M repositories found via GitHub API queries
  - > 2M orchestration files
    - so far, 6 VMs running for ~9 months
    - > 74k orchestration files

- available
- orchestrable
- traffic-generating
Users then specify traffic requirements in a
Declarative Traffic Specification Language

What kind of traffic?

Between which hosts?

How much traffic?

Example specification

generate web traffic
from h1 to h2
with 100 Mbps ;
generate database traffic
from h3 to external
with 50 Mbps
using $\geq 5$ flows ;
Users then specify traffic requirements in a Declarative Traffic Specification Language

**What kind of traffic?**

**Between which hosts?**

**How much traffic?**

**Example specification**

```plaintext
generate web traffic
  from h1 to h2
  with 100 Mbps ;
generate database traffic
  from h3 to external
  with 50 Mbps
  using ≥ 5 flows ;
```
Users then specify traffic requirements in a Declarative Traffic Specification Language (DTSL). See Figures 1 and 2 for the specification, that is, a set of statements in the Declarative Traffic Specification Language (DTSL).

### Example specification

```plaintext
generate web traffic
  from h1 to h2
  with 100 Mbps ;
generate database traffic
  from h3 to external
  with 50 Mbps
  using ≥ 5 flows ;
```

**What kind of traffic?**

**Between which hosts?**

**How much traffic?**
Users then specify traffic requirements in a Declarative Traffic Specification Language

What kind of traffic?

Between which hosts?

How much traffic?

Example specification

```plaintext
generate web traffic
  from h1 to h2
  with 100 Mbps;
generate database traffic
  from h3 to external
  with 50 Mbps
  using ≥ 5 flows;
```
Users then specify traffic requirements in a Declarative Traffic Specification Language.

What kind of traffic?

Between which hosts?

How much traffic?

Example specification

```plaintext
generate web traffic
from h1 to h2
with 100 Mbps ;
generate database traffic
from h3 to external
with 50 Mbps
using > 5 flows ;
```
Given a specification, DYNAMO generates matching live traffic

Example specification

```plaintext
generate web traffic
   from h1 to h2
   with 100 Mbps ;
generate database traffic
   from h3 to external
   with 50 Mbps
   using ≥ 5 flows ;
```
Given a specification, DYNAMO generates matching live traffic

Example specification

```
generate web traffic
    from h1 to h2
    with 100 Mbps ;
generate database traffic
    from h3 to external
    with 50 Mbps
    using ≥ 5 flows ;
```

Traffic generation

Send live traffic through a given user network
To achieve that, DYNAMO needs to orchestrate matching containers

Example specification

```plaintext
generate web traffic
    from h1 to h2
    with 100 Mbps ;
generate database traffic
    from h3 to external
    with 50 Mbps
    using > 5 flows ;
```

Identified projects

- 2x Project #5: 100 Mbps web traffic
- Projects #7 and #18: 50 Mbps database traffic using 7 flows

Traffic generation

Send live traffic through a given user network
To achieve that, DYNAMO needs to orchestrate matching containers

Example specification

```plaintext
generate web traffic from h1 to h2 with 100 Mbps;
genenerate database traffic from h3 to external with 50 Mbps using >= 5 flows;
```

Identified projects

- 2x Project #5: 100 Mbps web traffic
- Projects #7 and #18: 50 Mbps database traffic using 7 flows

Setup

Traffic generation

Run the correct containers

Send live traffic through a given user network
To achieve that, DYNAMO needs to **orchestrate** matching containers

Example specification

```plaintext
generate web traffic from h1 to h2 with 100 Mbps;
generate database traffic from h3 to external with 50 Mbps using ≥ 5 flows;
```

Identified projects

- 2x Project #5: 100 Mbps *web* traffic
- Projects #7 and #18: 50 Mbps *database* traffic using 7 flows

### Setup

- Run the correct containers

### Orchestration

- Combine to virtual hosts

### Traffic generation

- Send live traffic through a given user network
DYNAMO enables many use cases
And we'd love to hear more from you!

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security testing</td>
<td>DYNAMO generates real background traffic</td>
</tr>
<tr>
<td></td>
<td>E.g., to combine with attack traffic</td>
</tr>
<tr>
<td>Network design</td>
<td>DYNAMO tests applications under different designs</td>
</tr>
<tr>
<td></td>
<td>E.g., impact of packet loss on Bitcoin traffic</td>
</tr>
<tr>
<td>Trace generation</td>
<td>DYNAMO creates data sets with specific properties</td>
</tr>
<tr>
<td></td>
<td>E.g., to complement skewed ML training data</td>
</tr>
</tbody>
</table>
Our preliminary trace analysis shows the potential of the idea
Our preliminary trace analysis shows the potential of the idea

We found a wide range of traffic-generating applications

- **web** (HTTP, HTTPS)
- **crypto** (Bitcoin, IPFS)
- **database** (MongoDB, MySQL)
- **message-broker** (RabbitMQ, Apache Kafka)
Our preliminary trace analysis shows the potential of the idea

We found a wide range of traffic-generating applications

- **web** (HTTP, HTTPS)
- **crypto** (Bitcoin, IPFS)
- **database** (MongoDB, MySQL)
- **message-broker** (RabbitMQ, Apache Kafka)

Some of the applications generate *a lot of traffic*

- > 13M pkts (~417 Mbps), a multi-paxos implementation: thibmeu/imperial-multi-paxos-in-elixir
- > 367k flows (~4 Mbps), a Telegram proxy: squizduos/docker-server
DYNAMO showcases one approach to **bridge the gap** from static text/code to actual **network data**

- Static code analysis
  - Running the code
- ML-based techniques
- Meta relationships

Diagram:
- Static text
- Network data
DYNAMO showcases one approach to bridge the gap from static text/code to actual network data.
DYNAMO showcases one approach to **bridge the gap** from static text/code to actual **network data**

---

**Traffic-generating projects**

- Static code analysis
- Running the code
- ML-based techniques
- Meta relationships

Static text

Network data

- Bug testing
- Code completion