Extending KLEE's Environment Model For Networking

Jeroen Robben, Mathy Vanhoef
imec-Distrinet, KU Leuven

Overview

This work extends KLEE to allow a sound analysis of real-world network software without requiring altering application code. The POSIX environment model of KLEE was improved to include most common networking-related APIs provided by the Linux kernel. The behaviour of the network environment can be customized by standalone modules.

Network application → **Environment model** → **User-provided module**

Modelled APIs

Network I/O

- POSIX sockets and extensions
  - socket(), bind(), recvmsg(), …

Network environment discovery and configuration

- Rtnetlink
- Selected ioctls
- Selected /proc and /sys entries

Module interface

A *module* is a LLVM runtime library implementing a header file. Modules are provided by the tester and encapsulate the *behaviour* of the network environment.

1. Incoming network traffic
Define the structure and metadata of incoming network data at runtime

2. Available network interfaces
Define present network devices and their configuration

3. Handle outgoing data (Optional)
Add a handler for outgoing network traffic, e.g., to verify some assumption on the packet's contents

4. Liveness heuristic (Optional)
Instrument KLEE to stop exploring a path if some requirement is met, e.g., all incoming data is processed

Reference module

The reference module implementation allows to declare a network environment using a configuration file. The tester can statically define the available network interfaces, the structure of incoming network traffic and corresponding metadata.

Network interface

```
device klee-eth0 {
    flags: 4163
    mac: 50:eb:71:56:ab:9e
    mac-broadcast:  ff:ff:ff:ff:ff:ff
    mtu: 1500
    address-ipv4: {
        address: 192.168.1.2
        netmask: 255.255.255.0
    }
    address-ipv6: {
        address: 2001:41d0::4bc1
        prefix: 64
    }
}
```

Packet structure

```
packet-format packet1 {
    uint8: 0x12
    char[3]:
    uint32: 0
}

packet-format packet2 {
    uint8:
    uint16: 0x0102
    uint8:
    char[4]: 0x00abcdef
}
```

Packet metadata

```
incoming-config queue1 {
    domain: AF_INET
    type: SOCK_DGRAM
    protocol: IPPROTO_UDP
    device-name: klee-eth0
    receiver-port: 5000
    receiver-ip: 192.168.1.2
    sender-port: 8000
    sender-ip: 192.168.1.4
    packet-queue: packet1 packet2
}
```

