NetControl

Johanna Amann
johanna@icir.org
NetControl

Push rules to networking hard and software

Based on traffic observed by Bro

Simple to use but flexible API
Functional Example

```c
event connection_established(c: connection) {
    NetControl::drop_connection(c$id, 20 secs);
}
```
High level API

- `drop_connection (connection, timeout)`
- `drop_address (host, timeout)`
- `drop_address_catch_release (host)`
- `shunt flow (flow, timeout)`
- `quarantine (infected host, dns host, q. server, timeout)`
- `whitelist (prefix, timeout)`
Architecture

Bro

- Bro Event Engine
- NetControl Framework

NetControl Framework

- Backend 1
- Backend 2
- Backend 3
- Backend 4

Switch
- Switch
- Router
- Firewall

Network Traffic

High level calls or low-level primitives

Rules

Success, Failure, Timeout

Device communication
Adding a plugin

```cpp
event NetControl::init()
{
    local debug_plugin = NetControl::create_debug(T);
    NetControl::activate(debug_plugin, 0);
}

event connection_established(c: connection)
{
    NetControl::drop_connection(c$id, 20 secs);
}
```
What do Rules look like?

**Type**
- Drop
- Modify
- Redirect
- Whitelist

**Target**
- Forward
- Monitor

**Entity**
- Address
- Mac
- Connection
- Flow

**Additional Elements**
- Timeout
- Priority
- Location
Creating a drop rule

```cpp
event NetControl::init()
{
    local debug_plugin = NetControl::create_debug(T);
    NetControl::activate(debug_plugin, 0);
}

event connection_established(c: connection)
{
    NetControl::drop_connection(c$-id, 20 secs);
}
```
Creating a drop rule

event NetControl::init()
{
    local debug_plugin = NetControl::create_debug(T);
    NetControl::activate(debug_plugin, 0);
}

event connection_established(c: connection)
{
    local e = NetControl::Entity($ty=NetControl::CONNECTION, $conn=c$id);
    local r = NetControl::Rule($ty=NetControl::DROP,
                               $target=NetControl::FORWARD, $entity=e, $expire=20secs);

    NetControl::add_rule(r);
}
Choosing Backends

Network Traffic

Bro

Bro Event Engine

NetControl Framework

High level calls or low-level primitives

Rules

Success, Failure, Timeout

NetControl Framework

Backends

Backend 1

Backend 2

Backend 3

Backend 4

Device communication

Switch

Switch

Router

Firewall
Choosing Backends

NetControl Framework

OpenFlow Backend 1: 5
OpenFlow Backend 2: 2
OpenFlow Backend 3: 0

Network A

Network B

Tap switch
Choosing Backends

NetControl Framework

OpenFlow Backend 1

OpenFlow Backend 2

OpenFlow Backend 3

Network A

Network B

Tap switch
Choosing Backends

NetControl Framework →

OpenFlow Backend 1: 5
OpenFlow Backend 2: 2
OpenFlow Backend 3: 0

Network A → Tap switch → Network B
Choosing Backends

NetControl Framework

OpenFlow Backend 1

OpenFlow Backend 2

OpenFlow Backend 3

Network A

Network B

Tap switch
Choosing Backends

NetControl Framework

OpenFlow Backend 1: 5
OpenFlow Backend 2: 2
OpenFlow Backend 3: 0

Network A

Network B

Tap switch
Choosing Backends

NetControl Framework

OpenFlow Backend 1

OpenFlow Backend 2

OpenFlow Backend 3

Network A

Network B

Tap switch
Choosing Backends

NetControl Framework

OpenFlow Backend 1: 5
OpenFlow Backend 2: 2
OpenFlow Backend 3: 0

Network A
Tap switch

Network B
Choosing Backends

NetControl Framework

OpenFlow Backend 1

OpenFlow Backend 2

OpenFlow Backend 3

Network A

Network B

Tap switch
Choosing Backends

NetControl Framework

OpenFlow Backend 1
5

OpenFlow Backend 2
2

OpenFlow Backend 3
0

Network A

Network B

Tap switch
Choosing Backends

NetControl Framework

OpenFlow Backend 1
5

OpenFlow Backend 2
2

OpenFlow Backend 3
0

Network A

Network B

Tap switch
event NetControl::init()
{
  local debug_plugin = NetControl::create_debug(T);
  NetControl::activate(debug_plugin, 0);

  local of_controller = OpenFlow::log_new(42);
  local netcontrol_of = NetControl::create_openflow(of_controller);
  NetControl::activate(netcontrol_of, 0);
}

event connection_established(c: connection)
{
  NetControl::drop_connection(c$id, 1secs);
}
What can you customize?

Hooks into rule-creation

Plugins have hooks into rule-conversion

Plugins are easy to write

Python-API for Netcontrol rules
event NetControl::init()
{
    local debug_plugin = NetControl::create_debug(T);
    NetControl::activate(debug_plugin, 0);
}

hook NetControl::rule_policy(r: NetControl::Rule)
{
    if ( r$entity$ty == NetControl::CONNECTION &&
         r$entity$conn$resp_p == 443/tcp)
        break;
}

event connection_established(c: connection)
{
    NetControl::drop_connection(c$id, 20 secs);
}
module NetControl;

export {
    ## Instantiates the plugin.
    global create_skeleton: function(argument: string): PluginState;
}

function skeleton_name(p: PluginState): string {
    return "NetControl skeleton plugin";
}

function skeleton_add_rule_fun(p: PluginState, r: Rule): bool {
    print "add", r;
    event NetControl::rule_added(r, p);
    return T;
}
function skeleton_remove_rule_fun(p: PluginState, r: Rule): bool {
    print "remove", r;
    event NetControl::rule_removed(r, p);
    return T;
}

global skeleton_plugin = Plugin(
    $name = skeleton_name,
    $can_expire = F,
    $add_rule = skeleton_add_rule_fun,
    $remove_rule = skeleton_remove_rule_fun
);

function create_skeleton(argument: string): PluginState {
    local p = PluginState($plugin=skeleton_plugin);

    return p;
}
NetControl & Broker

Block, Shunt, ... Decisions

Network Control Framework

NC Broker Backend

Hardware/software switch/firewall

Broker-capable Receiver

Broker Protocol
#!/usr/bin/env python

import logging, netcontrol, pprint

logging.basicConfig(level=logging.DEBUG)

ep = netcontrol.Endpoint("bro/event/netcontrol-example", "127.0.0.1", 9977);
pp = pprint.PrettyPrinter(indent=4)

while 1==1:
    response = ep.getNextCommand()
    if response.type == netcontrol.ResponseType.AddRule:
        ep.sendRuleAdded(response, """)
    elif response.type == netcontrol.ResponseType.RemoveRule:
        ep.sendRuleRemoved(response, """)
    else:
        continue

pp.pprint(response.rule)
function our_predicate(p: NetControl::PluginState, r: NetControl::Rule): bool
{
    if ( r$entity$conn && r$entity$conn$orig_h in 10.0.0.0/8 )
        return F;

    return T;
}

event NetControl::init()
{
    local netcontrol_broker = NetControl::create_broker(
        NetControl::BrokerConfig($host=127.0.0.1, $bport=9977/tcp,
            $topic="bro/event/netcontrol-example", $check_pred=our_predicate), T);
    NetControl::activate(netcontrol_broker, 0);
}
NetControl & OpenFlow

Network Control Framework

NC OpenFlow Backend

OpenFlow Module

Ryu OpenFlow Controller

Block, Shunt, … Decisions

OpenFlow Protocol

Broker Protocol

OpenFlow Switch
NetControl Summary

NetControl is now in master

API is mostly stable

Please test and report problems
Get NetControl

github.com/bro/bro-netcontrol