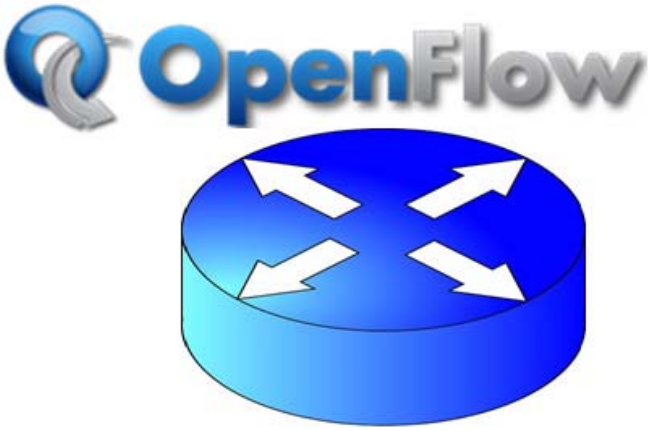
Software Defined Networking

SDN Security

David Hausheer

Department of Electrical Engineering
and Information Technology
Technische Universität Darmstadt

E-Mail: hausheer@ps.tu-darmstadt.de
http://www.ps.tu-darmstadt.de/teaching/sdn



*Based on original slides by Stephan Neuhaus (ETH Zürich) combined with slides from P. Porras, V. Yegneswaran, and S. Shin (Texas A&M and SRI International)

Killer Application of SDN?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Reducing energy in data center networks
- Dynamic virtual machine migration in cloud networks
- … Diverse network applications

- What about security
  - Can SDN enable new capabilities to improve network security?

Classic Network Perimeter Defense

TECHNISCHE UNIVERSITÄT DARMSTADT

- Security Policy Enforcement Methodology
  - Well-defined static security policy instantiated for a target topology
  - Deployed consistently across the network
  - Policy can only be altered by a small set of trusted elements
  - Policy modification events are audited and monitored for compliance

SDN / OpenFlow Network Model

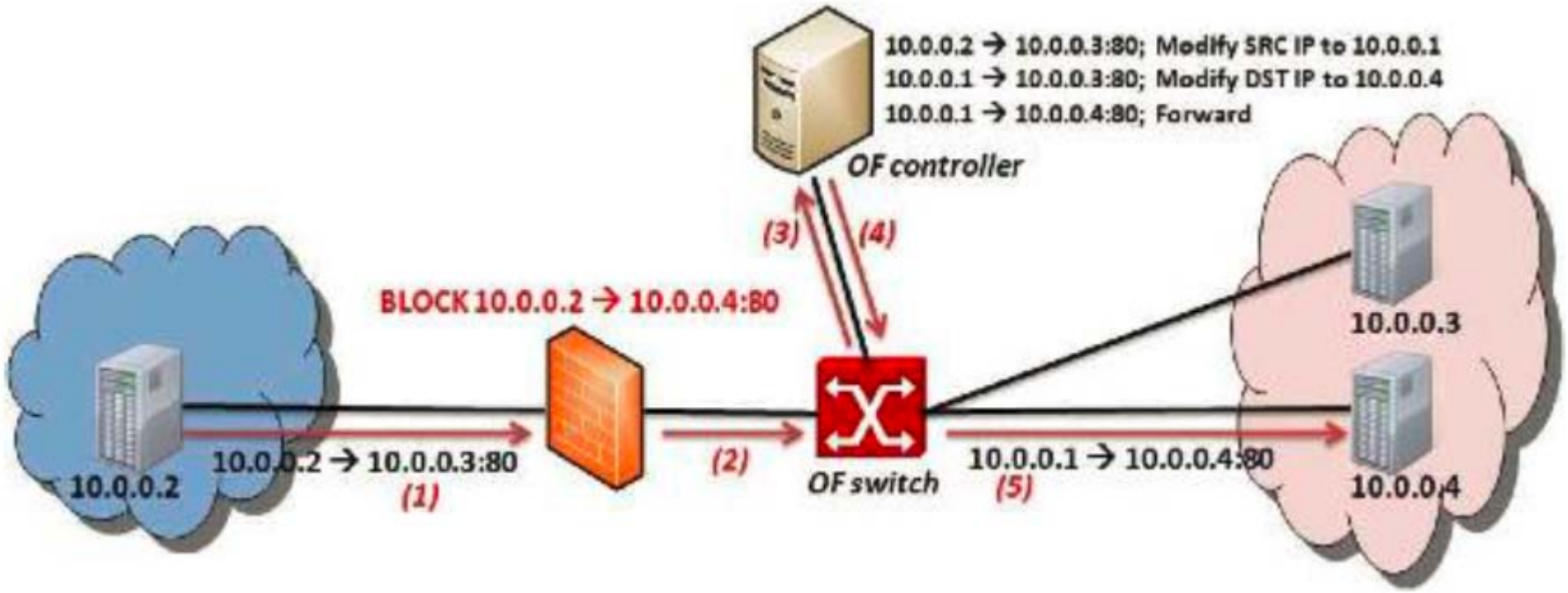
TECHNISCHE UNIVERSITÄT DARMSTADT

- Provides a set of continually and dynamically defined flow policies
- Flow policies are embodied in the current set of *flow rules* instantiated into the switch
- Flow rules are produced from OpenFlow applications that monitor and react to in and outbound packet flows
  - OF apps can compete, contradict, override one another, incorporate vulnerabilities
  - Worst case: an adversary can use the deterministic OF app to control the state of all OF switches in the network

OpenFlow Evasion Scenario

TECHNISCHE UNIVERSITÄT DARMSTADT

- Dynamic Flow Tunneling



OpenFlow Security Policy Enforcement

TECHNISCHE UNIVERSITÄT DARMSTADT

- Dynamic control plane (policies) and data plane (flows) introduces new enforcement challenges
- OpenFlow could benefit from better mechanisms for
  - specifying and authenticating policies
  - dealing with rewrite rules
  - detecting and auditing policy violations

- Objective:
  - Provide mechanisms that support the development and integration of *traditional* and *new* security applications into Software-Defined Networks

Motivating Security Applications

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Tarpits:** A Tarpit is an advanced anti-attack countermeasure designed to hold (reverse-DoS) inbound TCP connections from attackers
- **Reflector Nets:** A security app that reprograms the OF network to forward an external entity into a remote honeynet
- **Phantom Nets**: A technique in which a scanner is mislead into producing a false topology map for the network being scanned
- **Emergency Broadcast:** When a switch-wide exceptional state is detected, this security app auto-inserts a high-priority forward rule for all connections originating from network operator owned addresses, while inserting drop filters to reject detected flooding sources/ports
- **White holes**: A strategy for defeating sophisticated density-aware IP scanning techniques used by scan-and-infect malware to increase the rate at which viable infection targets are discovered
- **BotHunter**: A method for diagnosing infections in internal network assets using dialog correlation to discover flow sequences that match coordination centric malware infections

Prerequisites for a Secure OpenFlow Platform

TECHNISCHE UNIVERSITÄT DARMSTADT

- Must be resilient to
  - Vulnerabilities in OF applications
  - Malicious code in 3rd party OF apps
  - Complex interaction that arise between OF app interactions
  - State inconsistencies due to switch garbage collection or policy coordination across distributed switches
  - Sophisticated OF applications that employ packet modification actions
  - Adversaries who might directly target our security services to harm the network

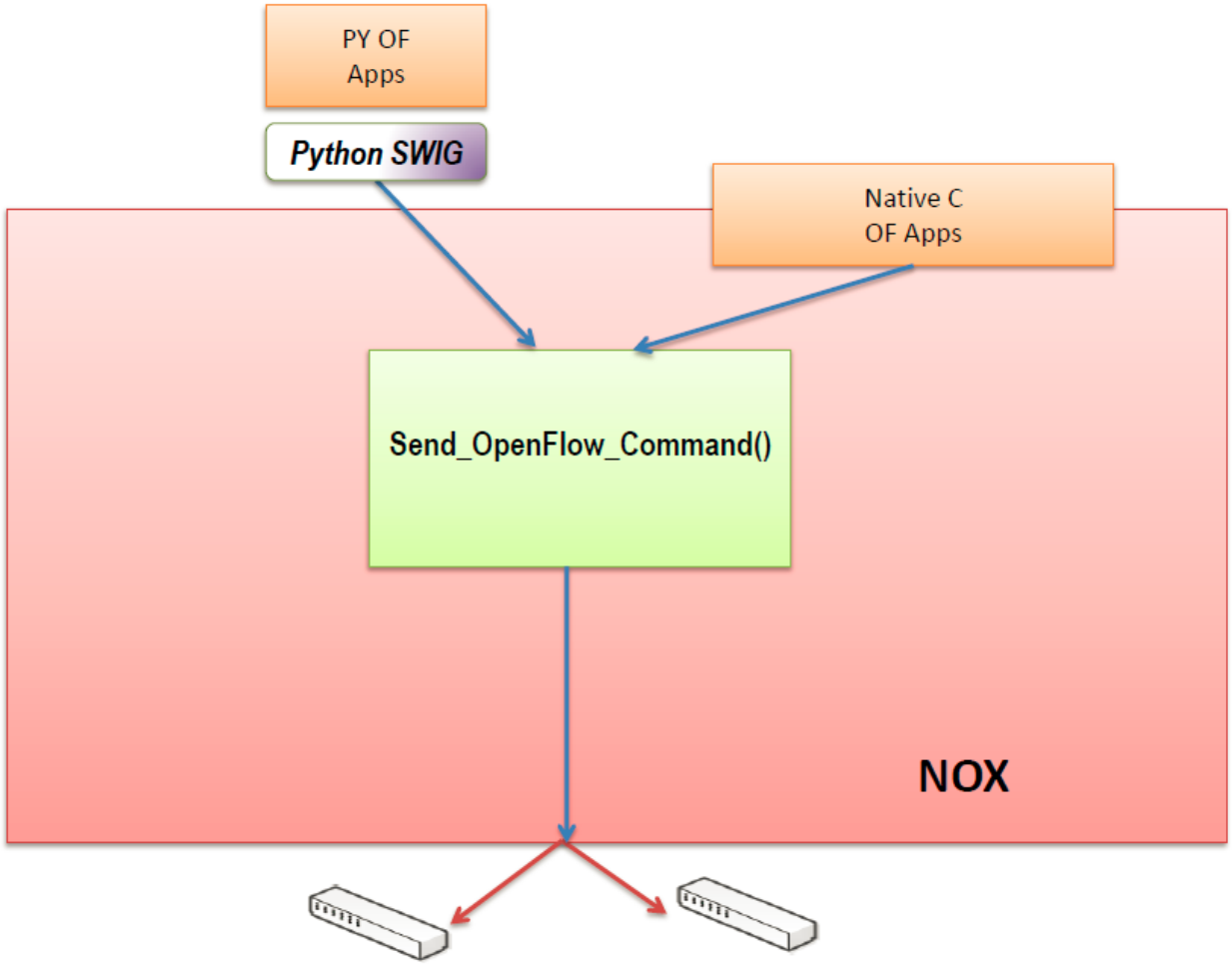
Lecture Overview

TECHNISCHE UNIVERSITÄT DARMSTADT

- FlowVisor: Security as Network Slicing
- FortNOX: Security as Conflict Detection/Resolution
- FRESCO: Security as Service Composition

FortNOX: Security as Conflict Detection / Resolution

Classic NOX Architecture



The FortNOX Security Enforcement Kernel

TECHNISCHE UNIVERSITÄT DARMSTADT

- FortNOX: A non-bypassable mediation service
  - All interactions between the controller and the switch must be mediated by the controller
  - Checks OpenFlow Application flow rules
    - Against the current set of network flow constraints
    - Defined by administrators or OpenFlow Security applications
  - Executes independently (in a separate process space) from that of the OpenFlow applications it services
    - And ideally from a separate user account

Authenticating Rule Producers

TECHNISCHE UNIVERSITÄT DARMSTADT

- FortNOX implements source authentication through the use of digital signatures
  - Rule producers export a public key, which administrators may choose to install into FortNOX, assigning this key to an authorization role
  - FortNOX accepts FLOW_MOD commands with an extra digital signature
  - Legacy OF application rules assigned default roles and lowest priorities

Role-Based Authorization

TECHNISCHE UNIVERSITÄT DARMSTADT

- FortNOX extends the controller to recognize 3 standard authorization roles among flow rule producers
  - **OF Operator Role** – define authoritative security policy
  - **OF Security Role** - add flow constraints to combat live threat activity
  - **OF Application Role** – legacy OF Apps, may remain security unaware

- Authorization roles inform
  - Rule priority assignments
  - Conflict resolution when conflicts are detected

Rule Conflict Analysis

TECHNISCHE UNIVERSITÄT DARMSTADT

- FortNOX incorporates a live rule conflict detection engine
  - **Rule Conflict**: arises when a new candidate rule enables or disables a network flow that is otherwise inversely prohibited (or allowed) by existing rules
  - **Alias set rule reduction** – a method detecting flow rule conflicts, even when OF set operations are used

FortNOX: Detecting Rule Conflicts

TECHNISCHE UNIVERSITÄT DARMSTADT

- Flow rules installed in a switch are an expression of some policy
  - Example: “Allow HTTP traffic to web server”
- Allows flow rules of the form
  - <anywhere> → webserver port 80 FORWARD
  - <anywhere> → webserver DROP
- But the actual rules installed in the switch change over time
- Someone wants to install new rule
  - 123.21.23.43 → webserver port 22 FORWARD

Alias Set Reduction Algorithm

TECHNISCHE UNIVERSITÄT DARMSTADT

- Definitions:
  - **established rule set**: set of rules already installed in the switch: *fRules*.
  - **candidate rule**: rule about to be installed: *cRule*.

- Assume for now only FORWARD and DROP actions

- If an OpenFlow rule would name only a single source and a single destination, this would do it:
  - For all members *f* of *fRules* do:
  - if cRule.src == f.src && cRule.dst == f.dst \
  - && cRule.action != f.action:
  - throw OpenFlowRuleConflict

Problem: The SET Action

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ignore port numbers, netmasks for the moment
- Assume that fRule contains:
  - a → b DROP
- Assume the new cRules are:
  - a → c SET a = a'
  - a' → c SET c = b
  - a' → b FORWARD
- This won't be caught by the previous algorithm
- We have to represent source and destination of rules as *sets*

Alias Set

TECHNISCHE UNIVERSITÄT DARMSTADT

- We call the addresses that a rule address can refer to the *alias set* for that address
- We write alias sets for rules like this:
  - a → b ACTION ({alias set for a}, {alias set for b})
- Original formulation: Conflict occurs if fRule's source (destination) is equal to some cRule's source (destination) and actions are unequal
- We simply replace "source" with "source alias set" and "equal" with "non-empty intersection"

Alias Set Reduction Algorithm

TECHNISCHE UNIVERSITÄT DARMSTADT

- Compute alias sets for fRules and cRule(s)

- For all members *f* of *fRules* do:
  - if cRule.src_alias ∩ f.src_alias != Ø \
  - && cRule.dst_alias ∩ f.dst_alias != Ø \
  - && cRule.action != f.action:
  - throw OpenFlowRuleConflict

Alias Set Computation

TECHNISCHE UNIVERSITÄT DARMSTADT

- With alias sets, we have in fRules:
  - a → b DROP ({a}, {b})
- Now for the cRules:
  - a → c SET a = a' ({a, a'}, {c})
    - a and a' refer to the same address
  - a' → c SET c = b ({a, a'}, {c, b})
    - b and c refer to the same address, as do a and a'
  - a' → b FORWARD ({a, a'}, {c, b})
- Source alias sets and destination alias sets intersect, actions differ, so there is a CONFLICT!

Conflict Resolution

TECHNISCHE UNIVERSITÄT DARMSTADT

- So far we've been doing conflict *detection*
- Possible conflict *resolution* based on role-based *priority*:
  - Human admins (high)
  - Security apps (medium)
  - Others (low)
- On conflict detection, priority decides:
  - cRule.prio >= f.prio: install rule
  - cRule.prio < f.prio: discard rule

Policy Enforcement

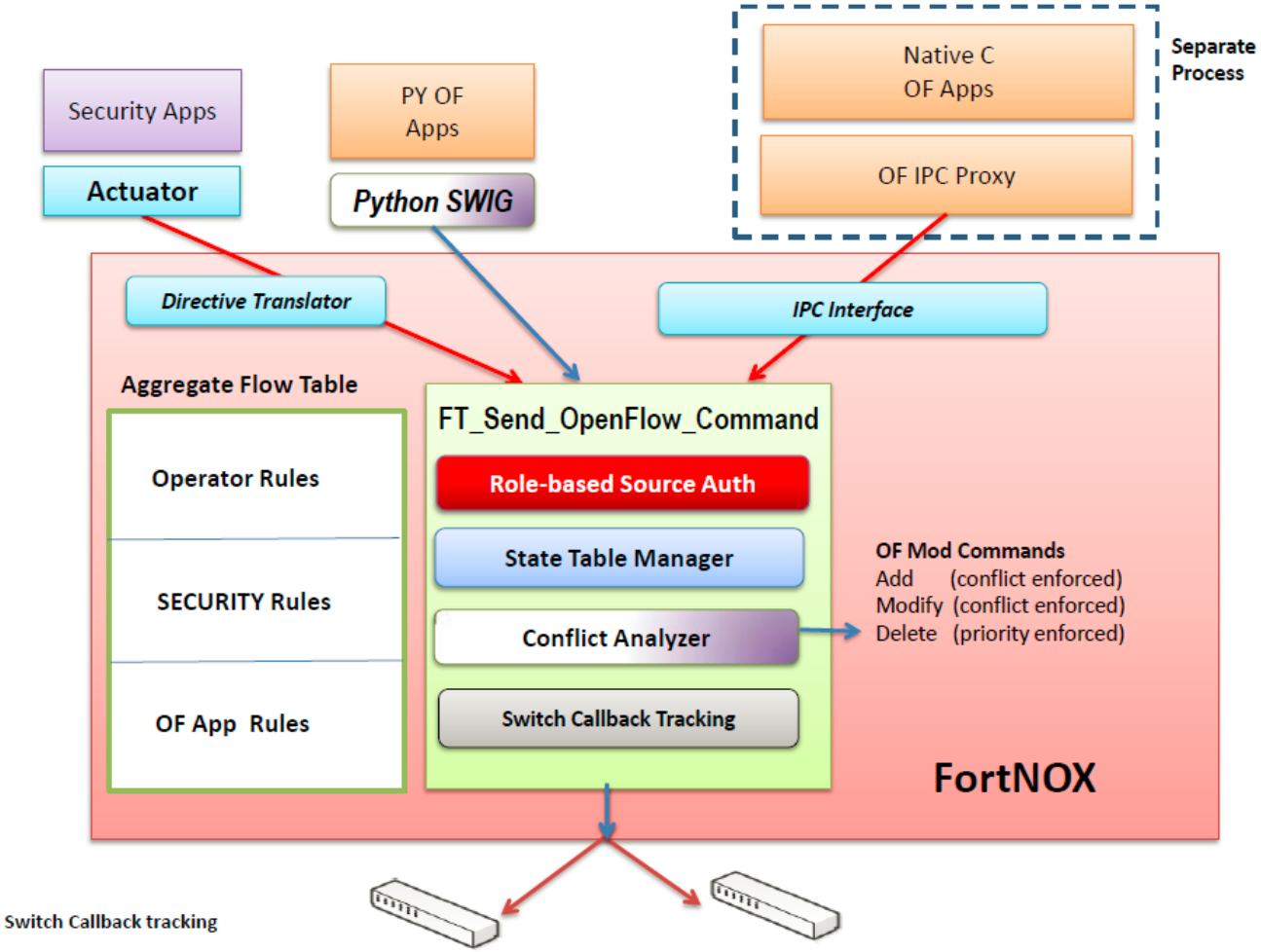
TECHNISCHE UNIVERSITÄT DARMSTADT

- FortNOX contains a Security Directive Translator
  - Python interface for translating high level mitigation directives into flow rules
  - Directive *block*: blacklist CIDR block
  - Directives *deny*, *allow*: as in firewall
  - Directive *redirect*: tunneling (honeynet)
  - Directive *constrain*: inactivate all rules not of specific priority (used in emergencies during DDoS)
  - Directives *undo*, *info*: ??? (no info available)

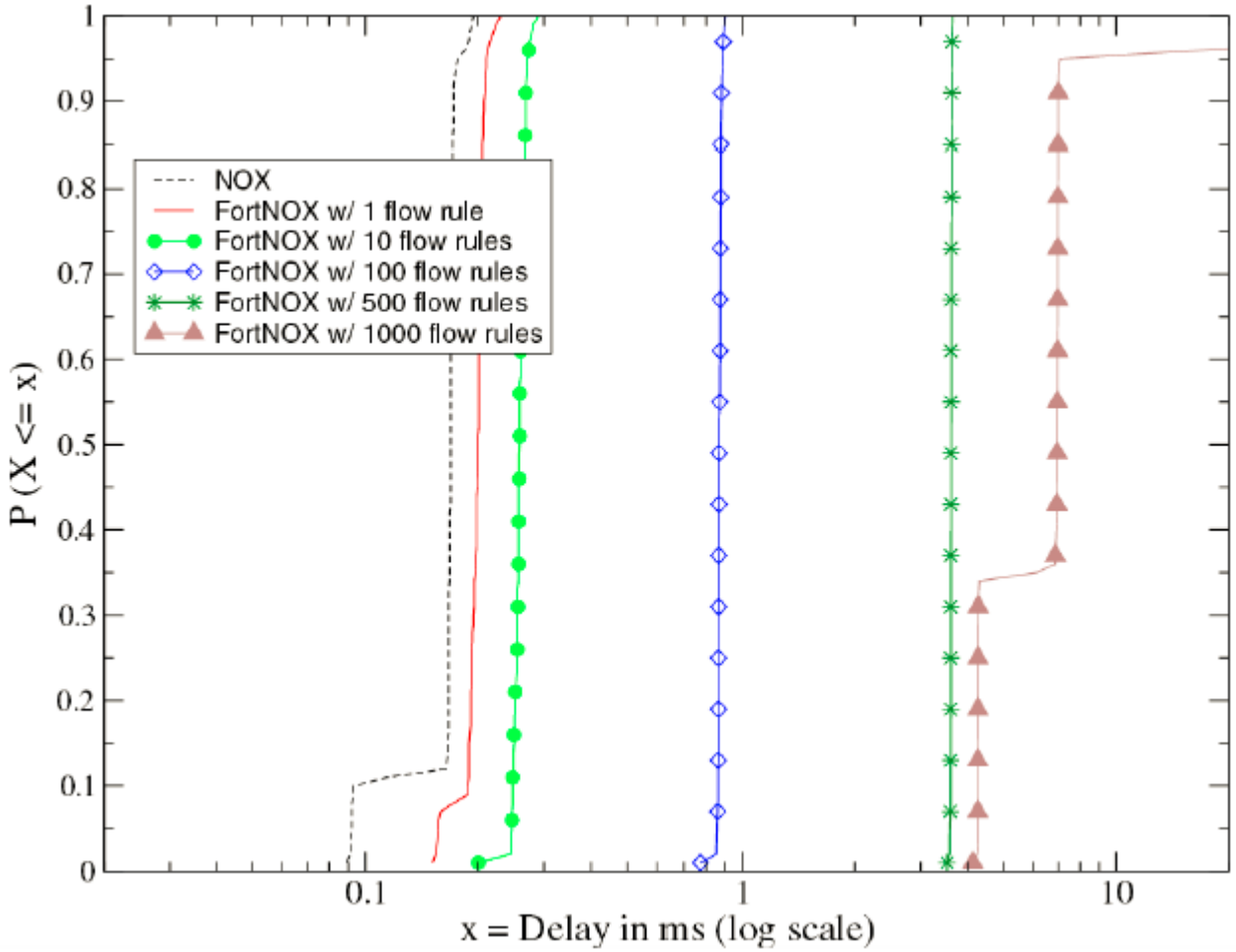
CIDR: Classless Inter-Domain Routing

FortNOX Architecture

TECHNISCHE UNIVERSITÄT DARMSTADT



Performance



FRESCO: Security as Service Composition

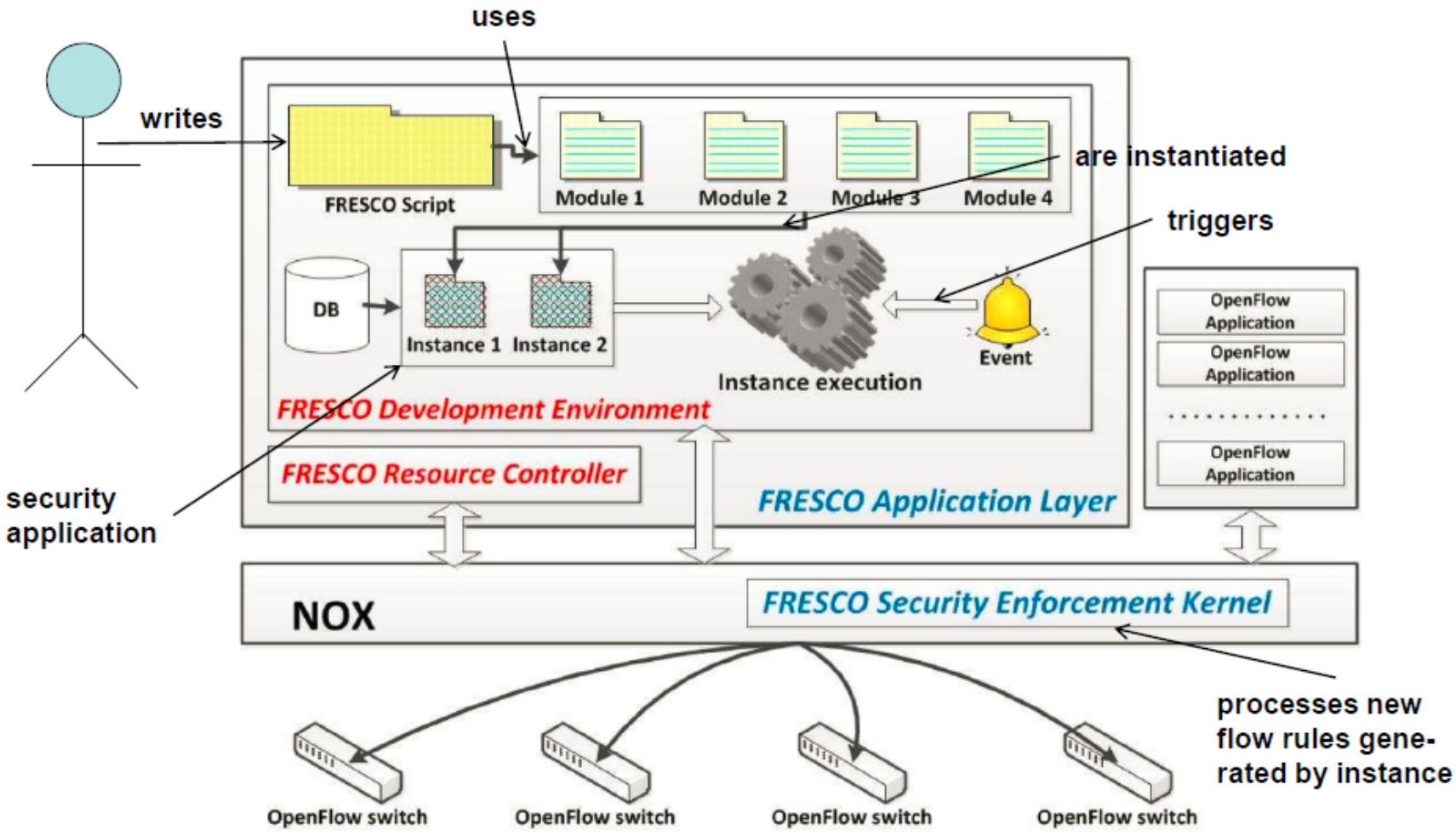
FRESCO: Composing Security Services

TECHNISCHE UNIVERSITÄT DARMSTADT

- It is not easy to create security apps in SDN
  - Security service creation and composition challenge
    - How do we simplify development of security applications?
  - Information deficiency challenge
    - E.g., TCP session, network status
  - Threat response translation challenge
    - How do we enforce security policies to the network devices?

- FRESCO: an application framework that
  - enables modular design
  - of complex OpenFlow-aware network security services
  - built from smaller [...] libraries of security functions

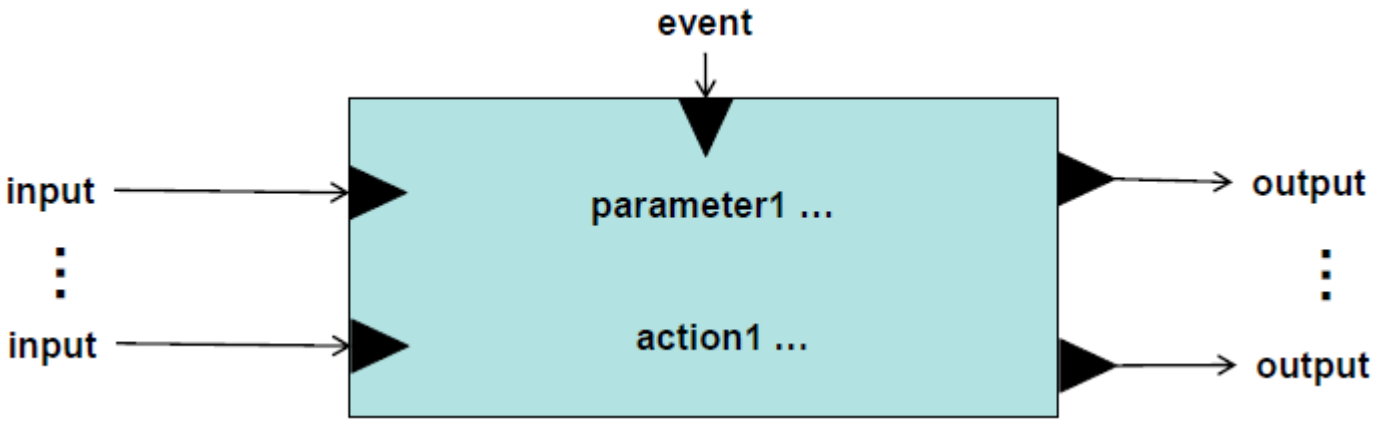
Architecture



Basic Unit of Execution: Module

TECHNISCHE UNIVERSITÄT DARMSTADT

- Modules are Python objects with interfaces:
  - Input: receive values
  - Output: transmit values
  - Parameter: configure/initialise module
  - Action: operate on network packet
  - Event: trigger module execution



Possible FRESCO Actions

TECHNISCHE UNIVERSITÄT DARMSTADT

- drop: drop a packet
- output (a.k.a. forward): output packet to port
- group: process packet through specified group
- set: rewrite packet header
  - redirect: redirect packet to host (no state management or address translation by application)
  - mirror: copy packet to mirror port
  - quarantine: tag packet as malicious; tagged packets can only be forwarded to allowed hosts, so the flow is isolated

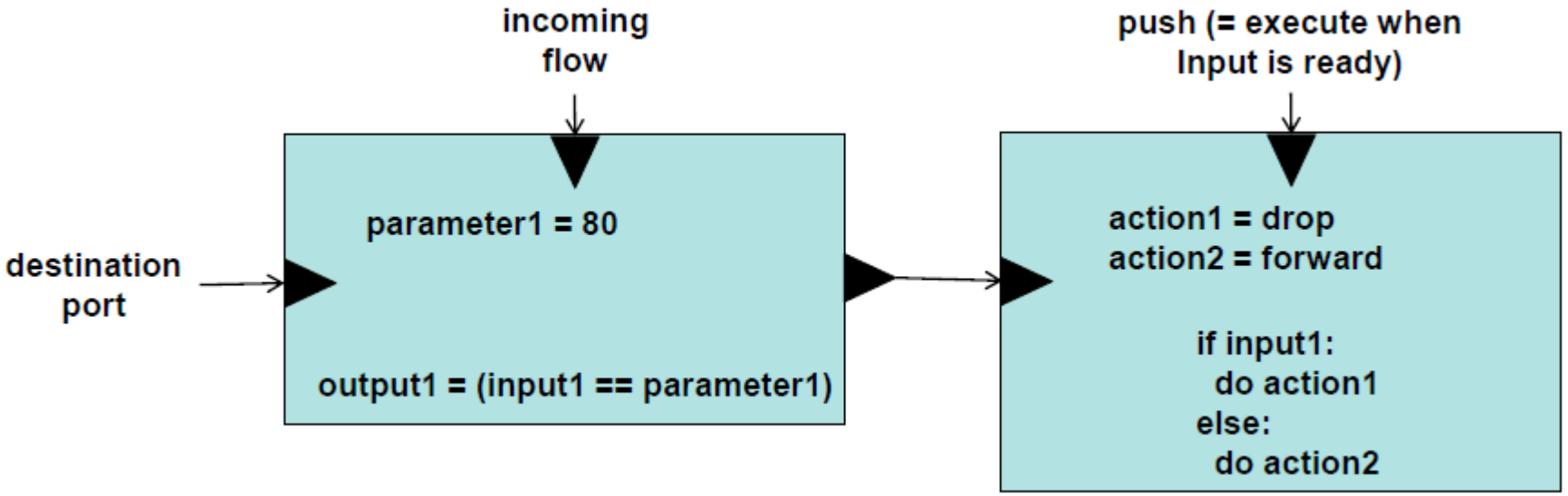
FRESCO Script

TECHNISCHE UNIVERSITÄT DARMSTADT

```
instance_name (#inputs) (#outputs) {
  type: name of existing module of which this is an instance
  event: triggering event
  input: name of input item, name of input item, ...
  output: name of output item, name of output item, ...
  parameter: value
  action: action to be performed
}
```

Inspired by Click Modular Router
Robert Morris, Eddie Kohler, John Jannotti, and M. Frans Kaashoek. Proceedings of SOSP '99

Example: Drop HTTP Packets

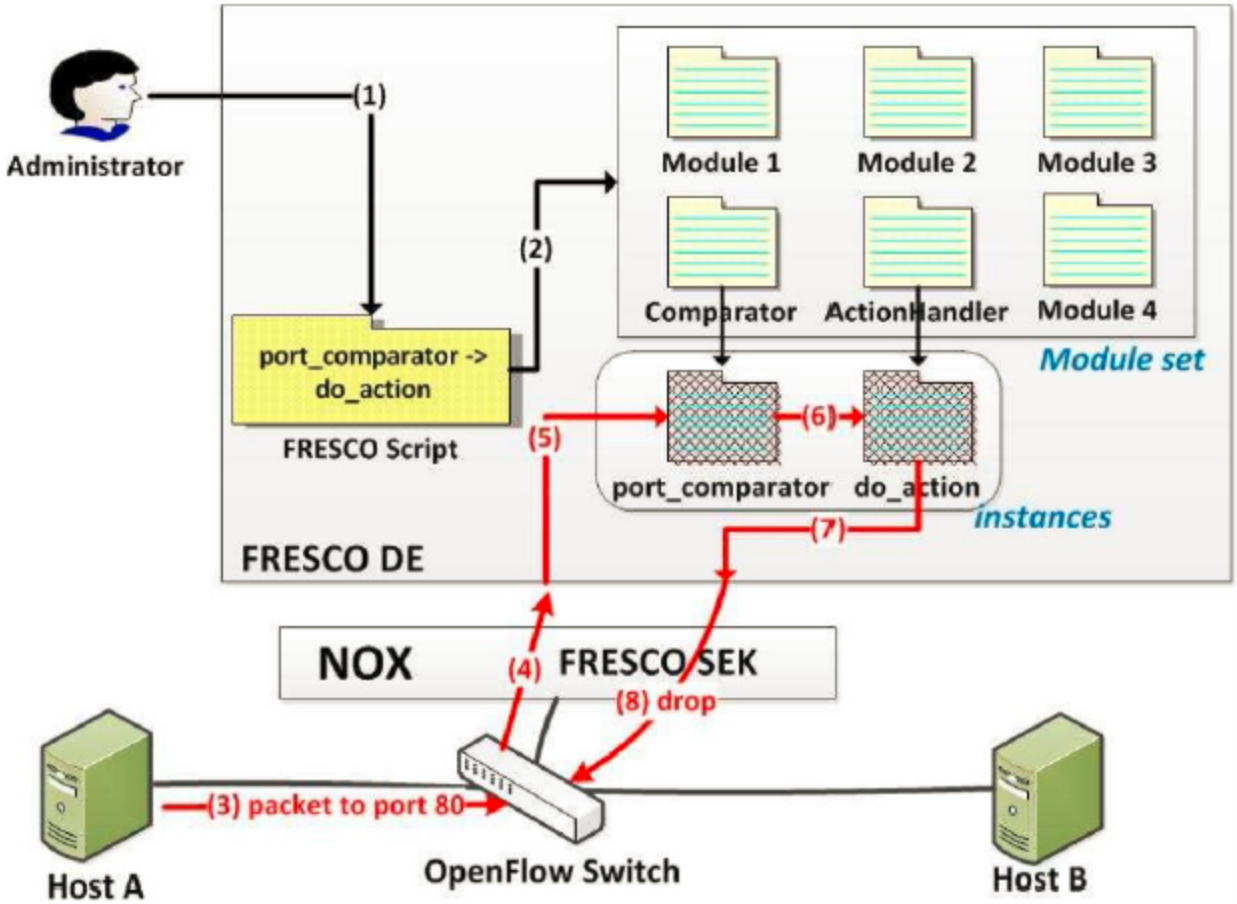



```
port_comparator (1) (1) {
  type: Comparator
  event: INCOMING_FLOW
  input: destination_port
  output: comparison_result
  parameter: 80
  action: -
}
```

```
do_action (1) (0) {
  type: ActionHandler
  event: PUSH
  input: comparison_result
  output: -
  parameter: -
  action: comparison_result ? DROP : FORWARD
}
```

Security App in Action

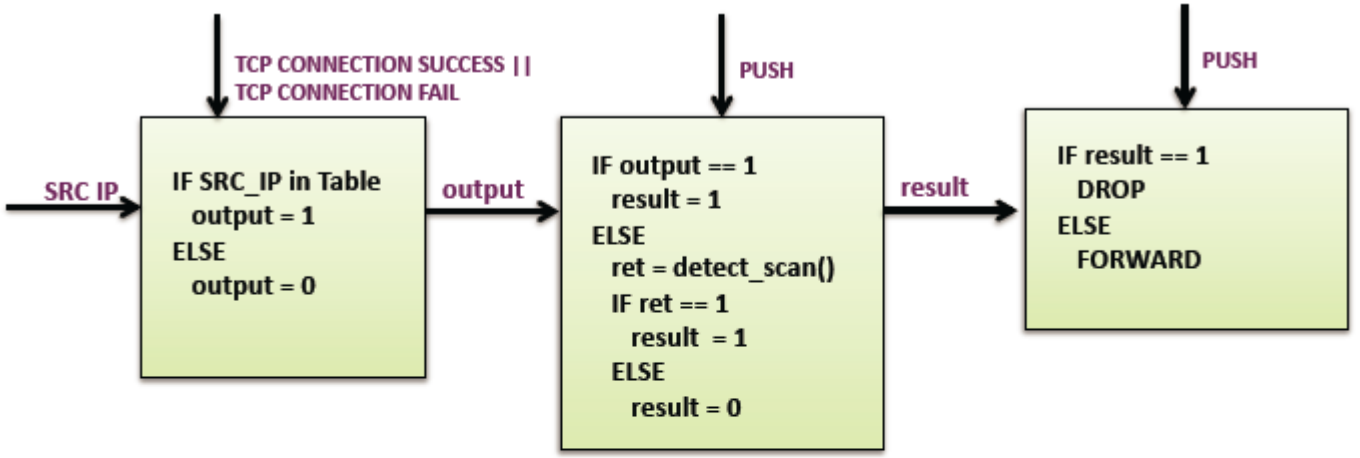
TECHNISCHE UNIVERSITÄT DARMSTADT



Example: Scan Detection

TECHNISCHE UNIVERSITÄT DARMSTADT

- Steps
  - Check blacklist => Threshold based scan detection => Drop or Forward



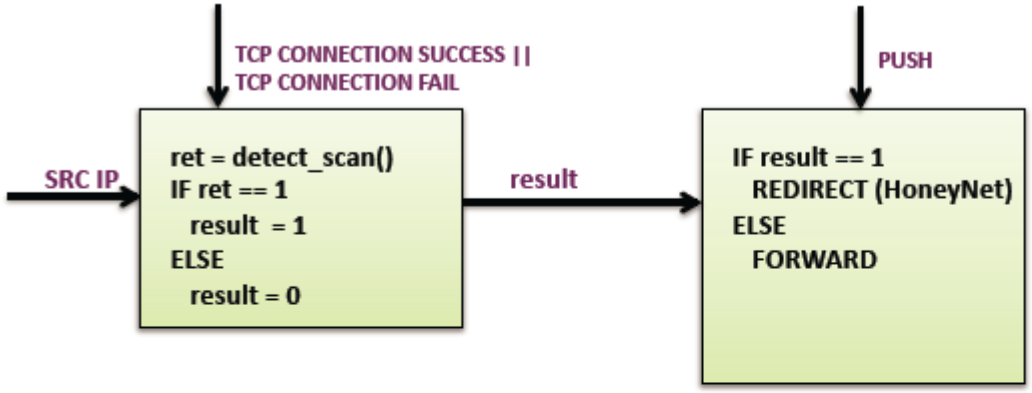

```
blacklist_check (1)(1) {
   type: TableLookup
   event: TCP_CONNECTION_FAIL,
         TCP_CONNECTION_SUCCESS
   input: SRC_IP
   output: blacklist_out
   parameter: NONE
   action: NONE
}
```

```
scan_detect (1)(1) {
   type: ScanDetect
   event: PUSH
   input: blacklist_out
   output: detect_result
   parameter: NONE
   action: NONE
}
```

```
final_action (1)(0) {
   type: ActionHandler
   event: PUSH
   input: detect_result
   output: NONE
   parameter: NONE
   action: detect_result == 1
         ?DROP:FORWARD
}
```

Example: Reflector Net

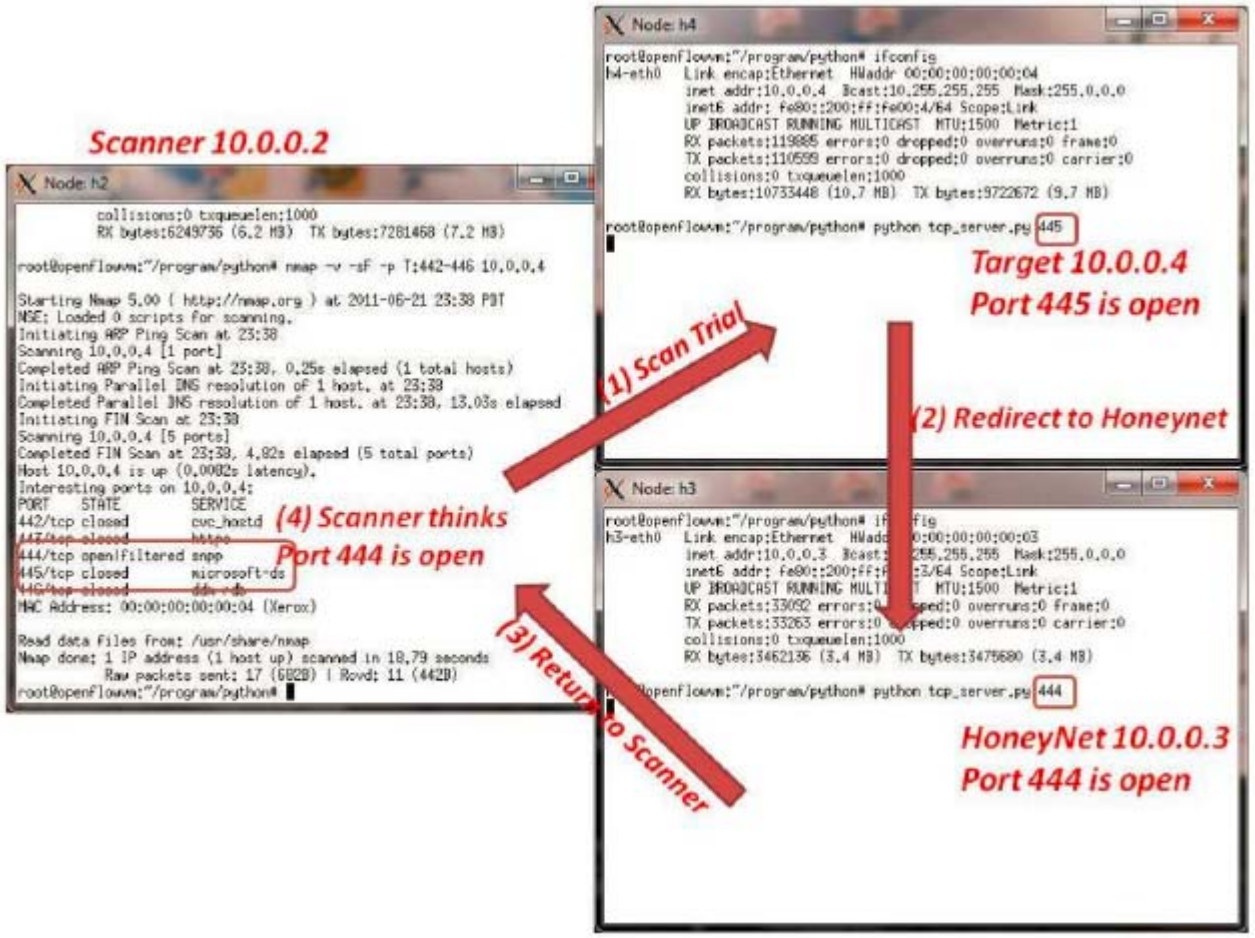
- Confuse network scan attackers
- Steps
  - Threshold-based scan detection => Reflect or Forward




```
scan_detect (1)(1) {
   type: ScanDetect
  event: TCP_CONNECTION_FAIL,
         TCP_CONNECTION_SUCCESS
input: blacklist_out
   output: detect_result
   parameter: NONE
   action: NONE
}
```

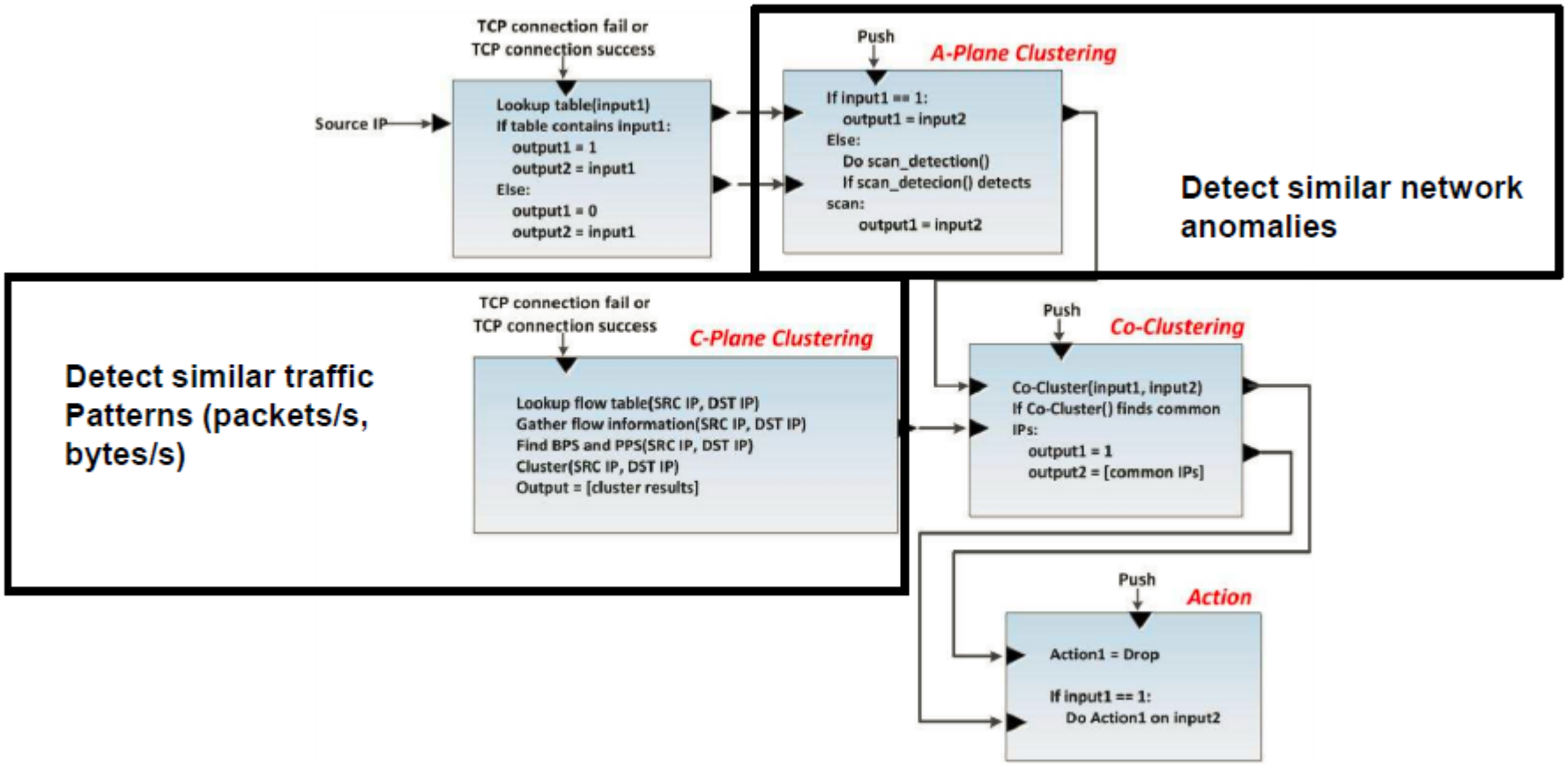
```
final_action (1)(0) {
   type: ActionHandler
   event: PUSH
   input: detect_result
   output: NONE
   parameter: NONE
   action: detect_result ==  ?REDIRECT(10.0.0.3):FORWARD
}
```

Reflector Net in Action



BotMiner

TECHNISCHE UNIVERSITÄT DARMSTADT



Evaluation

TECHNISCHE UNIVERSITÄT DARMSTADT

Source code length comparison

Algorithm	Implementation		
	Standard	OpenFlow	FRESCO
TRW-CB	1,060	741	66 (58 + 8)
Rate Limit	991	814	69 (61 + 8)

Results for Standard and OpenFlow are obtained in the following paper,
S. A. Mehdi, J. Khalid, and S. A. Khayam.
Revisiting Traffic Anomaly Detection Using Software Defined Networking, In Proceedings of Recent Advances in Intrusion Detection, 2011.

Flow rule setup time

	NOX	Simple Flow Tracker	Simple Scan Detector	Threshold Scan Detector	BotMiner	P2P Plotter Detector
Time (ms)	0.823	1.374	2.461	7.196	15.461	11.775

Refer to FRESCO paper for the explanation of each test case

Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- FortNOX
  - A new security enforcement kernel for OF networks
  - Role-based Authorization
  - Rule-Authentication
  - Conflict Detection and Resolution
  - Security Directive Translation
- FRESCO
  - Create security applications easily
  - Deploy security applications easily
  - Focus on creating security applications

References

- P. Porras, S. Shin, V. Yegneswaran et al.: A Security Enforcement Kernel for OpenFlow Networks, Workshop on Hot Topics in Software Defined Networks, Aug 2012. (FortNOX Paper)

- S. Shin, P. Porras, V. Yegneswaran, et al.: FRESCO: Modular Composable Security Services for Software-Defined Networks. Internet Society NDSS, Feb 2013.

- Open Networking Foundation: SDN Security Considerations in the Data Center, ONF Solution Brief, Oct 2013. Available from https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-security-data-center.pdf