Virtual Labs on Cybersecurity and P4 Programmable Switches

Jorge Crichigno, Elie Kfoury, Jose Gomez, Ali AlSabeh
University of South Carolina

2023 Winter ICT Educators Conference
January 5-6, 2022
Online

Agenda

- Virtual labs on Cybersecurity Fundamentals and Security+
- Virtual labs on P4 Programmable Data Plane Applications

Cybersecurity Fundamentals Lab Series

Cybersecurity Fundamentals Lab Series

The labs are available on NDG's NETLAB+ and provides hands-on experiences on:

- Reconnaissance and vulnerability assessment
- Infiltrating a victim's device with malware (trojan, spyware, keylogger, etc.)
- Social engineering attacks (phishing emails, credential harvesting)
- Attacks on web applications (SQL injection, cross-site scripting)
- Network attacks (Denial of Service (DoS))
- Cryptography fundamentals (symmetric encryption, asymmetric encryption, digital certificates)
- Packet filtering and access control lists
- Brute force attacks on passwords
- Intrusion detection and prevention system

Cybersecurity Fundamentals Lab Series

The labs provide learning experiences on cybersecurity topics

Lab 1: Reconnaissance: Scanning with NMAP, Vulnerability Assessment with OpenVAS

Lab 2: Remote Access Trojan (RAT) using Reverse TCP Meterpreter

Lab 3: Escalating Privileges and Installing a Backdoor

Lab 4: Collecting Information with Spyware: Screen Captures and Keyloggers

Lab 5: Social Engineering Attack: Credentials Harvesting and Remote Access through Phishing Emails

Lab 6: SQL Injection Attack on a Web Application

Lab 7: Cross-site Scripting (XSS) Attack on a Web Application

Lab 8: Denial of Service (DoS) Attacks: SYN/FIN/RST Flood, Smurf attack, and SlowLoris

Lab 9: Cryptographic Hashing and Symmetric Encryption

Lab 10: Asymmetric Encryption: RSA, Digital Signatures, Diffie-Hellman

Lab 11: Public Key Infrastructure: Certificate Authority, Digital Certificate

Lab 12: Configuring a Stateful Packet Filter using iptables

Lab 13: Online Dictionary Attack against a Login Webpage

Lab 14: Intrusion Detection and Prevention using Suricata

Organization of Lab Manuals

Each lab starts with a section *Overview*

- Objectives
- Lab settings: passwords, device names
- Roadmap: organization of the lab

Section 1

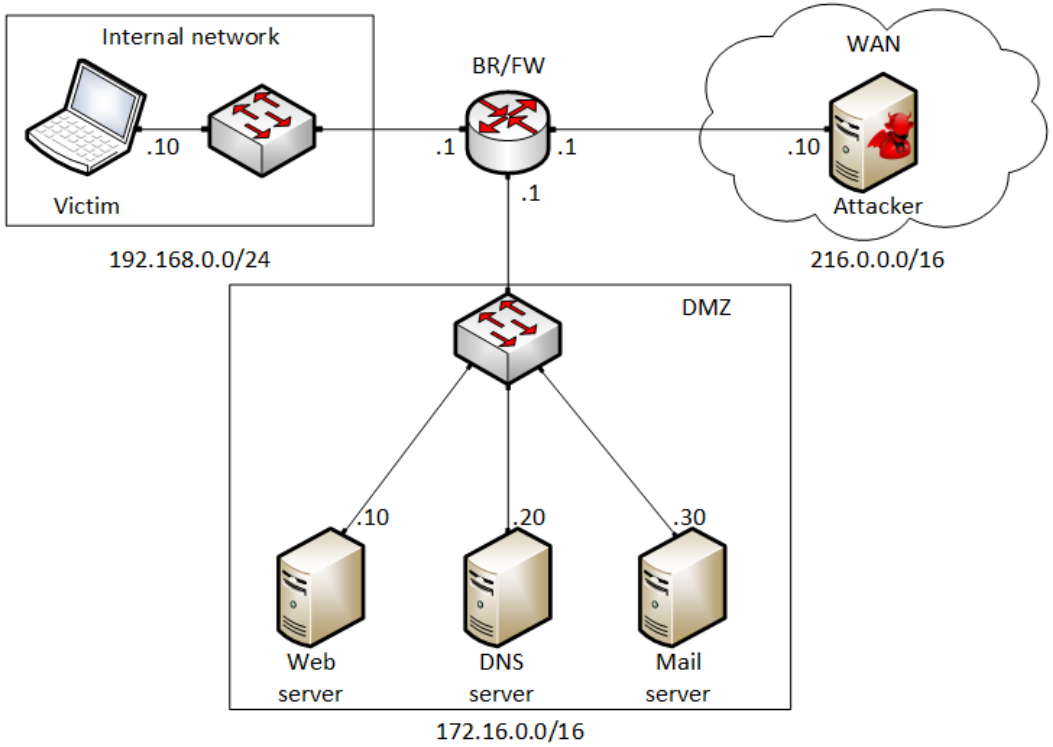
- Background information (theory) of the topic being covered (e.g., malware fundamentals)
- Section 1 is optional (i.e., the reader can skip this section and move to lab directions)

Section 2… n

- Step-by-step directions

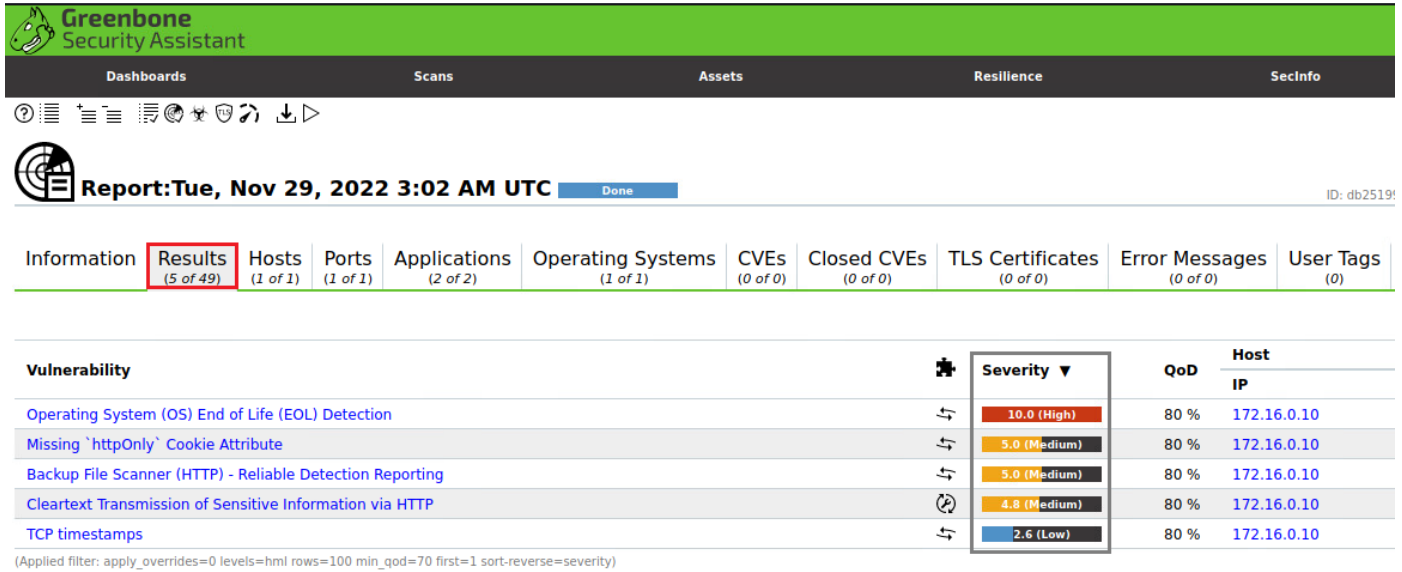
Pod Design

- Attacker in the WAN running Kali
- Victim in the internal network running Windows 10
- Web, DNS, and Mail servers in the DMZ zone
- Border router interconnect the networks
- Border router implements basic security policy:
  - Attacker cannot initiate connections to devices in the internal network



Examples

Vulnerability assessment using OpenVAS

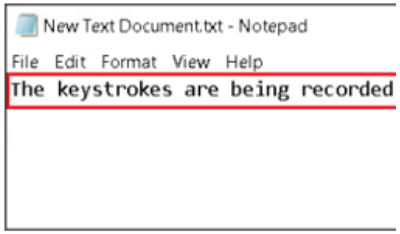


Examples

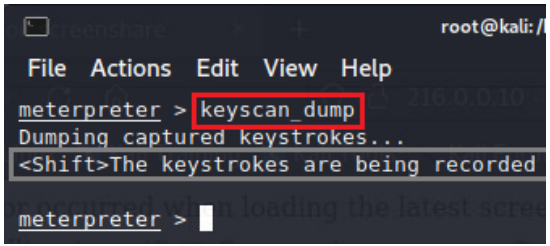
Deploying a Spyware

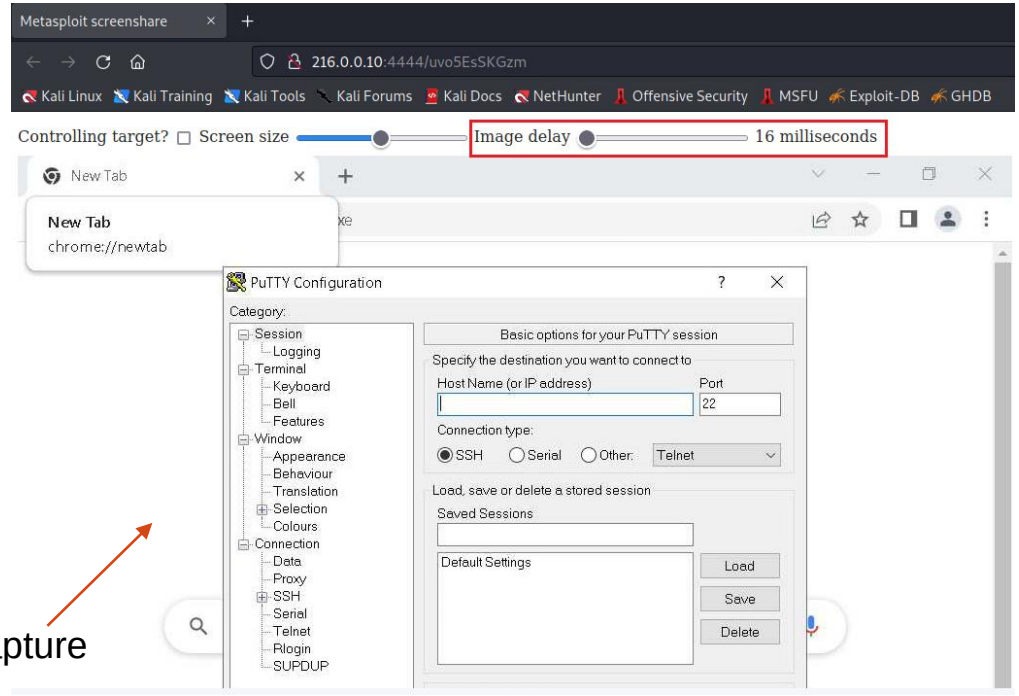
Keylogger

Victim



Attacker





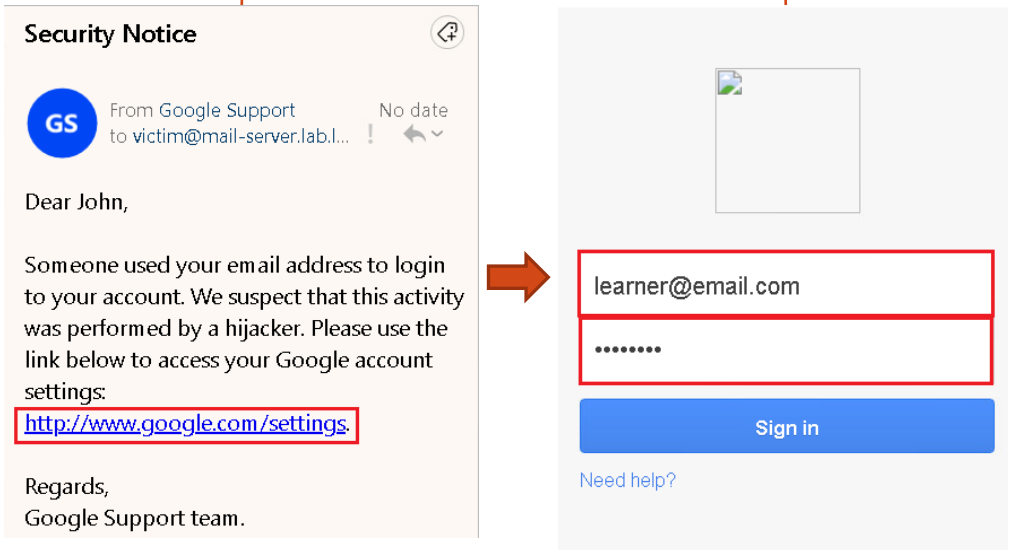
Screen capture

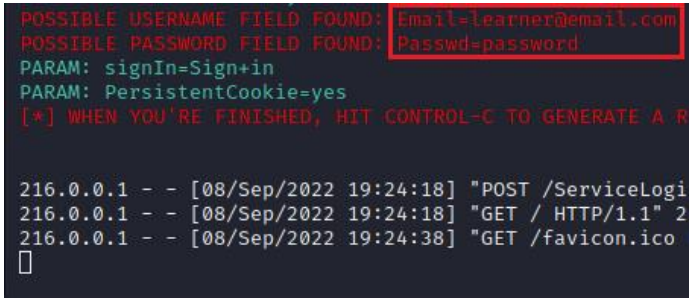
Examples

Social engineering and phishing emails

Victim

Attacker



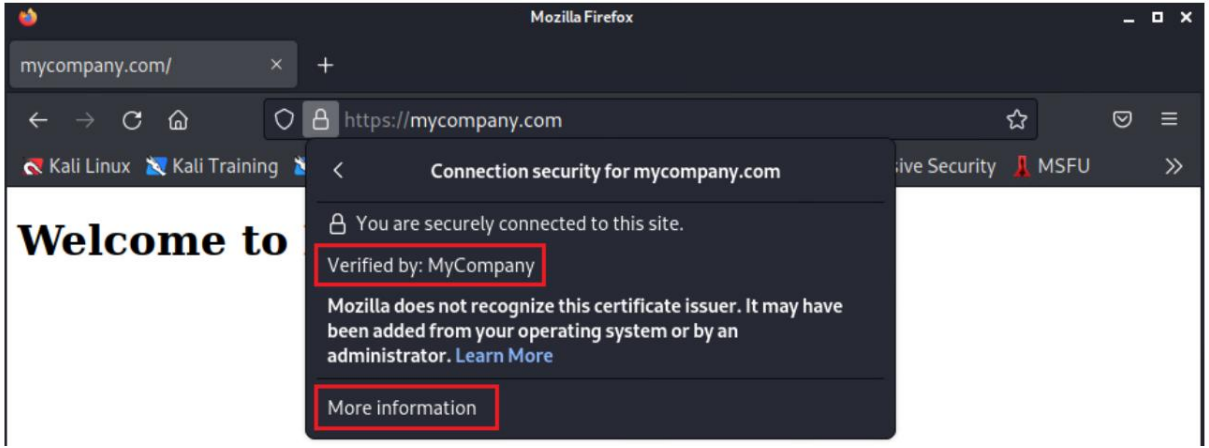


Examples

Creating a digital certificate and deploying it on an Apache web server

```
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:SC
Locality Name (eg, city) []:Columbia
Organization Name (eg, company) [Internet Widgits Pty Ltd]:MyCompany
Organizational Unit Name (eg, section) []:IT
Common Name (e.g. server FQDN or YOUR name) []:mycompany.com
Email Address []:admin@mycompany.com
```

← X.509 certificate



← Certificate deployed on a production grade web server

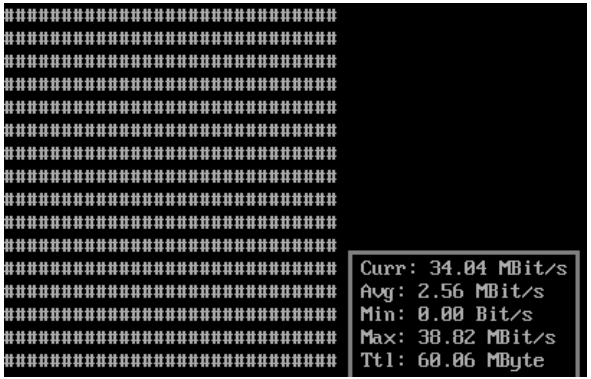
Examples

Detecting and blocking SYN Flood attack using Suricata IDS/IPS

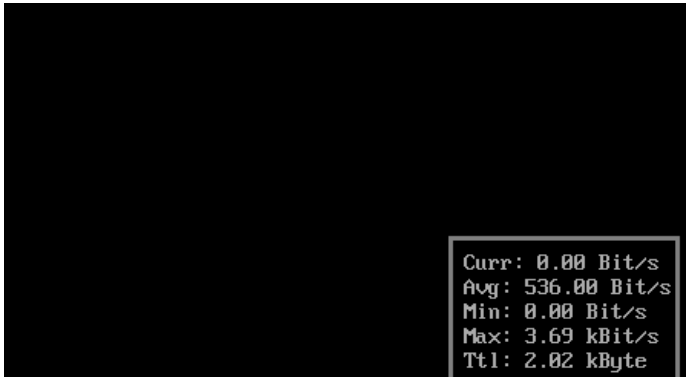
```
alert tcp any any -> 172.16.0.20 any (flags:S; sid:1234568; rev:1;)
~
~
```

```
rate_filter gen_id 1, sig_id 1234568, track by_dst, count 1000, seconds 1, new_action drop, timeout
30_
```

Incoming rate before mitigation



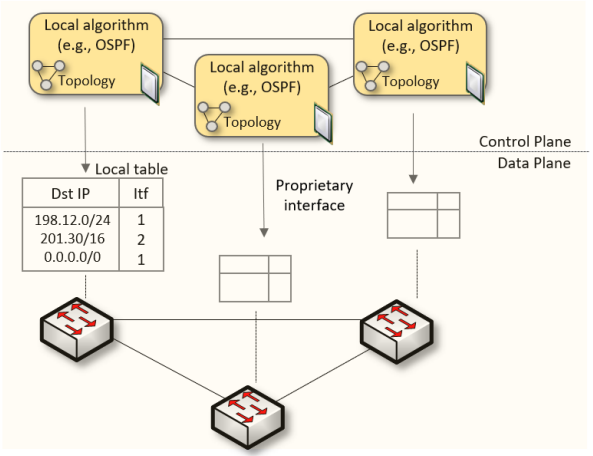
Incoming rate after mitigation



P4 Applications Lab Series

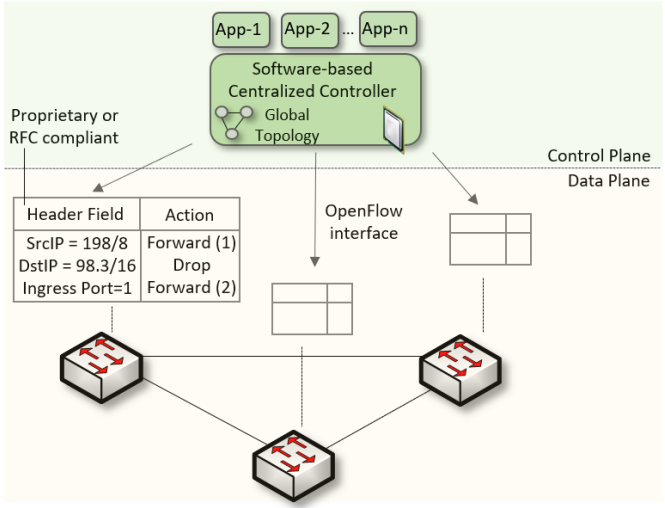
Traditional (Legacy) Networking

- The interface between the control plane and data plane has been historically proprietary
- A router is a monolithic unit built and internally accessed by the manufacturer only
- There is a vendor dependence: slow product cycles of vendor equipment, standardization, no room for innovation from network owners



SDN

- Protocol ossification has been challenged first by SDN
- SDN explicitly separates the control and data planes, and implements the control plane intelligence as a software outside the switches



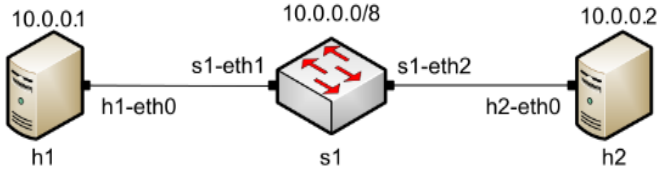
Environment: Mininet

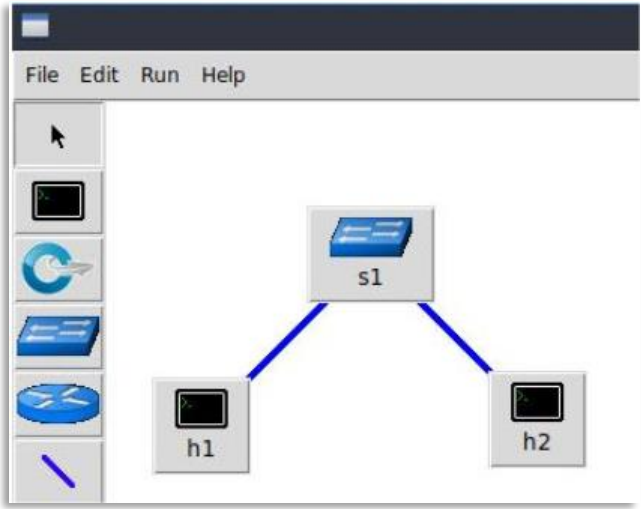
Mininet

- Mininet is a virtual testbed for developing and testing network tools and protocols
- Nodes are sometimes called containers, or more accurately, *network namespaces*
- Features
  - Fast prototyping for new protocols
  - Simplified testing for complex topologies
  - It runs real code on Linux (realistic emulation)
  - Open source
  - Containers consume few resources (100s or 1,000s of nodes)

MiniEdit

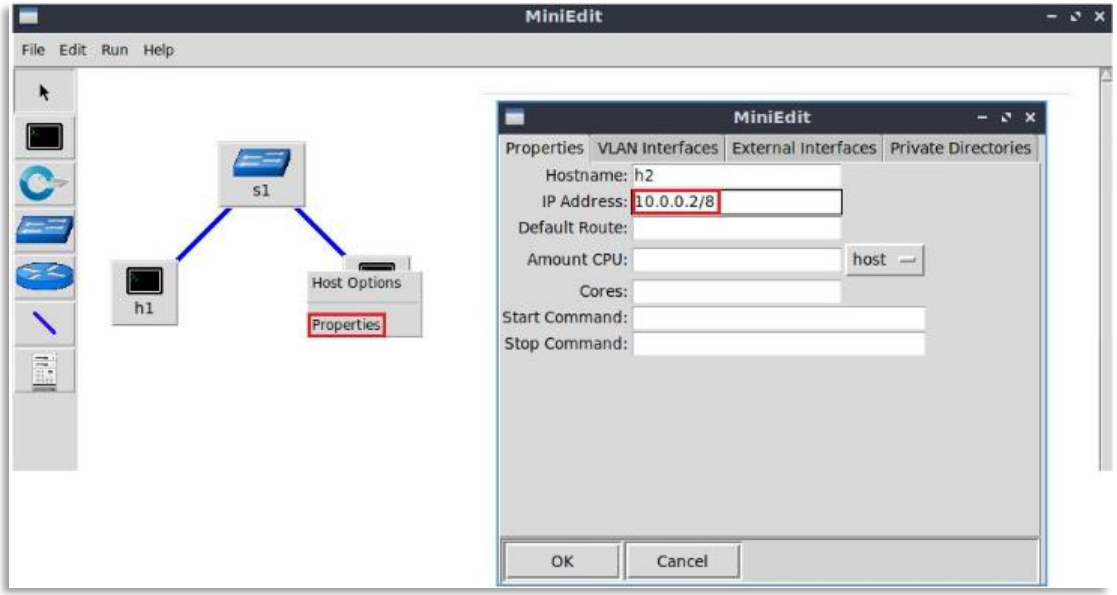
- To build a topology, we use MiniEdit
- MiniEdit is a simple GUI editor for Mininet
- Example:





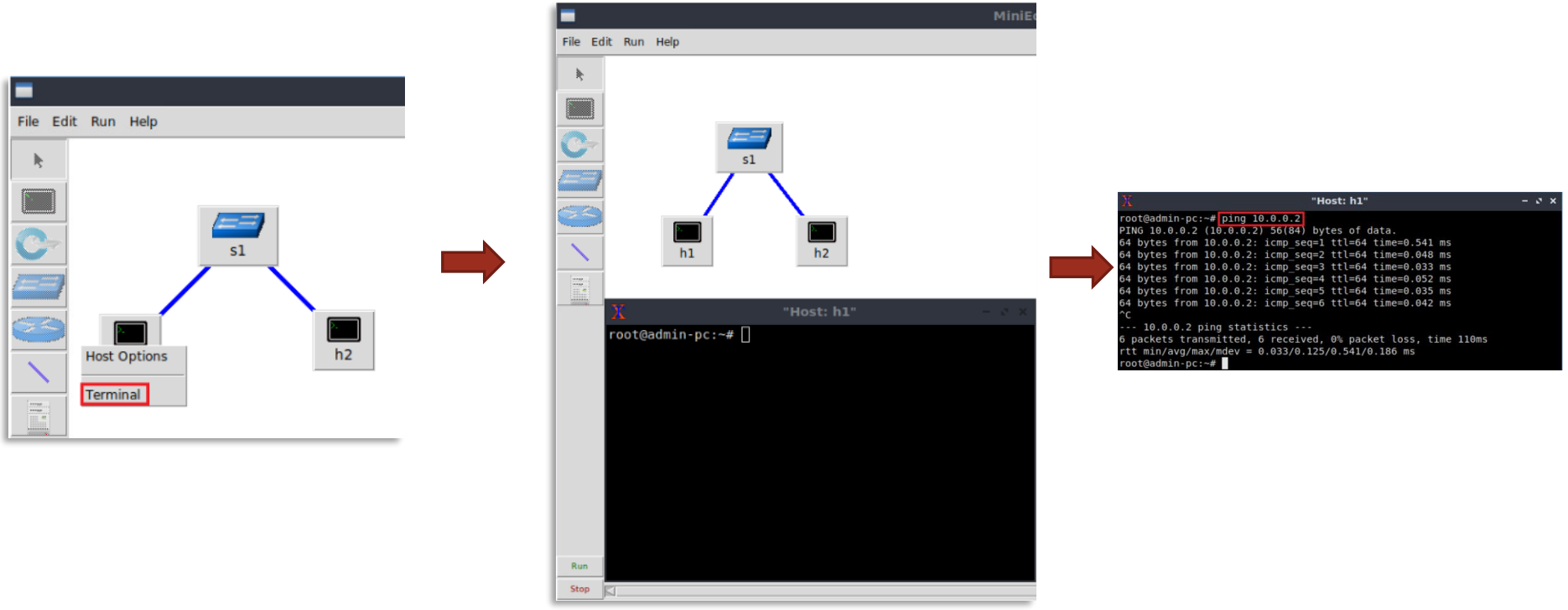
Host Configuration

- Configure the IP addresses at host h1 and host h2
- A host can be configured by holding the right click and selecting properties on the device



Executing Commands on Hosts

- Open a terminal on host by holding the right click and selecting *Terminal*



Overview P4 Applications Labs

P4 Programmable Switches

- The programmable forwarding can be viewed as a natural evolution of SDN
- P4 programmable switches permit a programmer to program the data plane
  - Defining and parsing new protocols
  - Customizing packet processing functions
  - Measuring events occurring in the data plane at nanosecond resolution
  - Inspecting and analyzing each packet (per-packet analysis)
- P4 stands for stands for Programming Protocol-independent Packet Processors

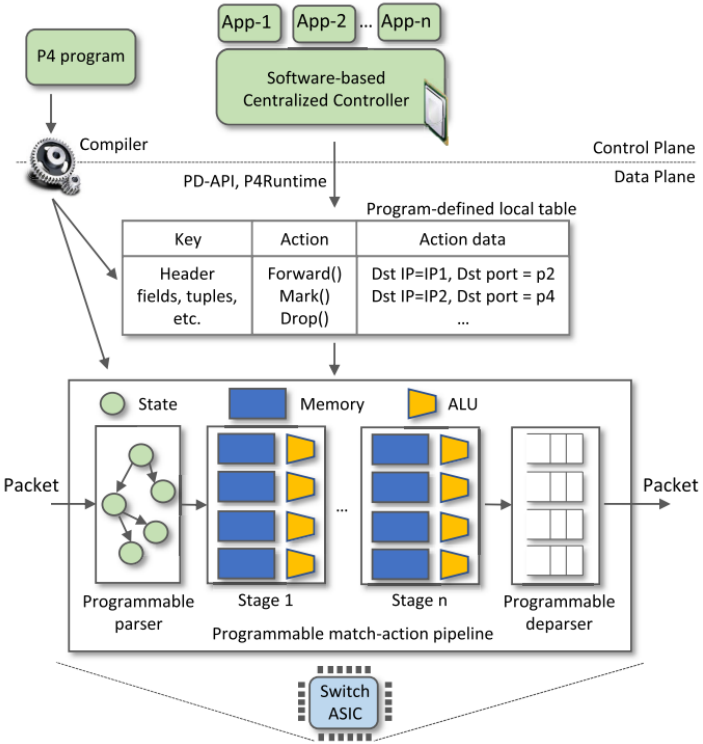
P4 Programmable Switches

- Analogy between networks and other computing domains

Domain	Year	Processing Unit	Main Language/s
General computing	1971	Central Processing Unit (CPU)	C, Java, Phyton, etc.
Signal processing	1979	Digital Signal Processor (DSP)	Matlab
Graphics	1994	Graphics Processing Unit (GPU)	Open Computing Language
Machine learning	2015	Tensor Processing Unit (TPU)	Tensor Flow
Computer networks	2016	Protocol Independent Switch Architecture (PISA)	P4

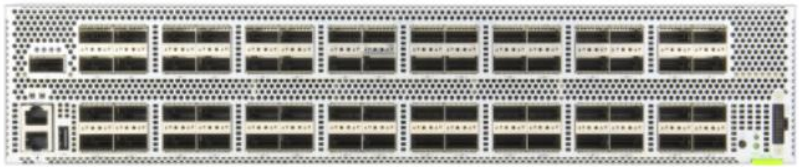
P4 Programmable Switches

- Programmable chip
  - Parser parses header fields, written by the programmer
  - Stages contain memory and Arithmetic Logic Units (ALUs)
  - Memory are used for tables, match bits
  - ALUs are simple, suitable for header field operations, actions
  - Stages are sequentially arranged (1, 2, …, n), for sequential computation
  - Deparser assembles packet headers back



Examples of P4 Programmable Switches

- Behavioral Model Version 2 (BMv2)
  - Open source
  - Software switch used for teaching, researching ideas
  - Good to validate ideas
- Commercial physical devices
  - E.g., Edgecore Wedge 100BF-65X (based on Intel's Tofino chip)
  - 65x100G switch ports
  - Used in production networks and research



P4 Applications and Custom Processing Lab Series

Lab experiments

Lab 1: Introduction to Mininet

Lab 2: Introduction to P4 and BMv2

Lab 3: P4 Program Building Blocks

Lab 4: Defining and processing custom headers

Lab 5: Monitoring the Switch's Queue using Standard Metadata

Lab 6: Collecting Queueing Statistics using a Header Stack

Lab 7: Measuring Flow Statistics using Direct and Indirect Counters

Lab 8: Rerouting Traffic using Meters

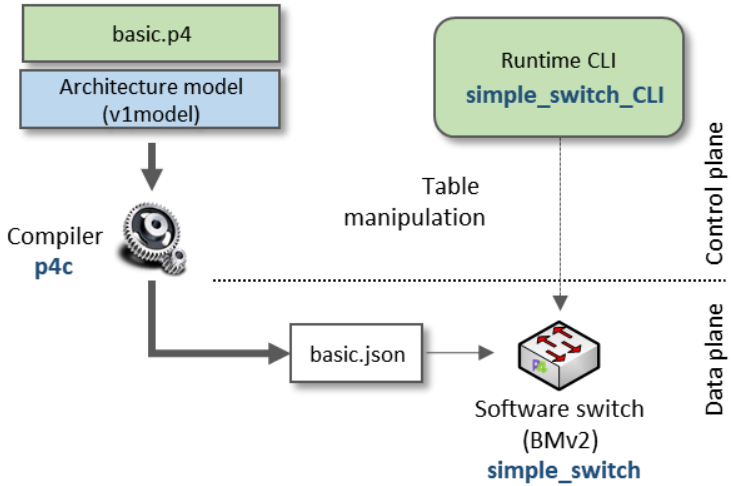
Lab 9: Storing Arbitrary Data using Registers

Lab 10: Calculating Packets Interarrival Times using Hashes and Registers

Lab 11: Generating Notification Messages from the Data Plane using Digests

Workflow of a P4 Program

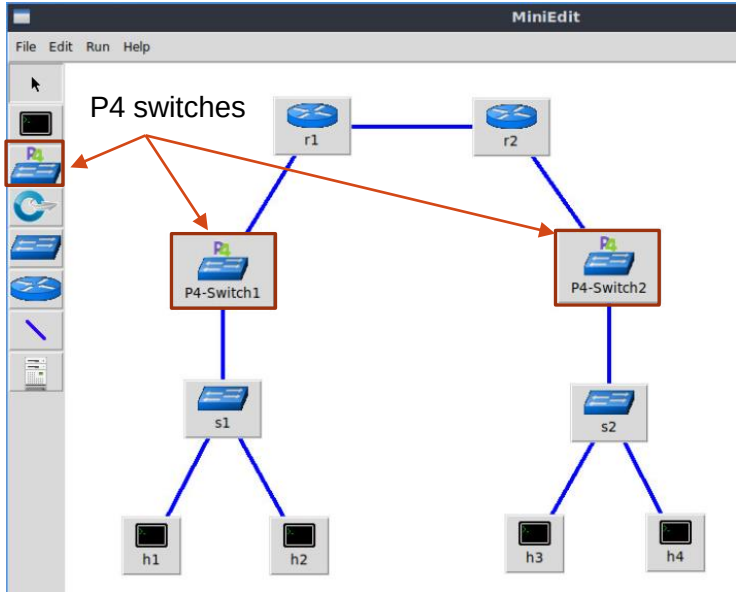
- Workflow used to program the BMv2 switch

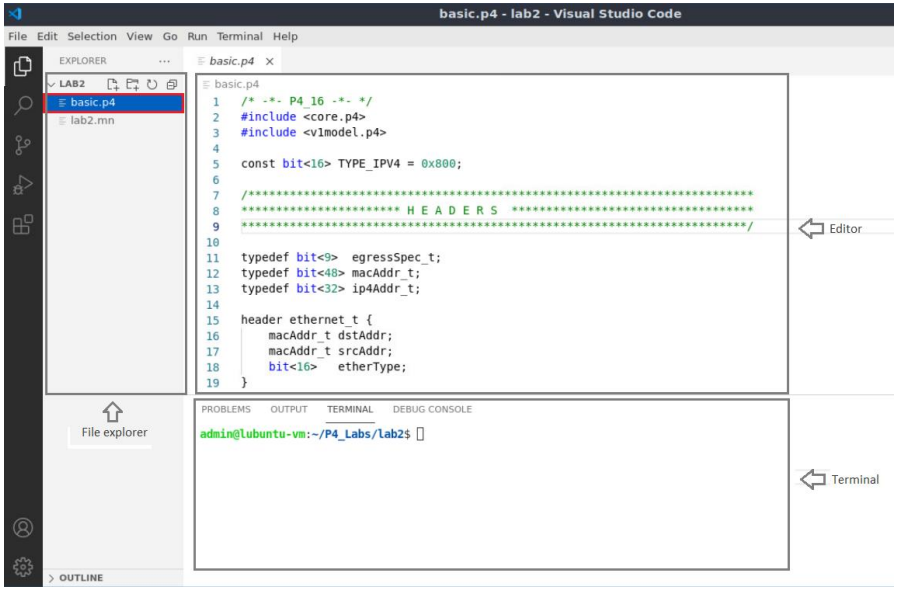


Workflow used in the lab series

Development Environment

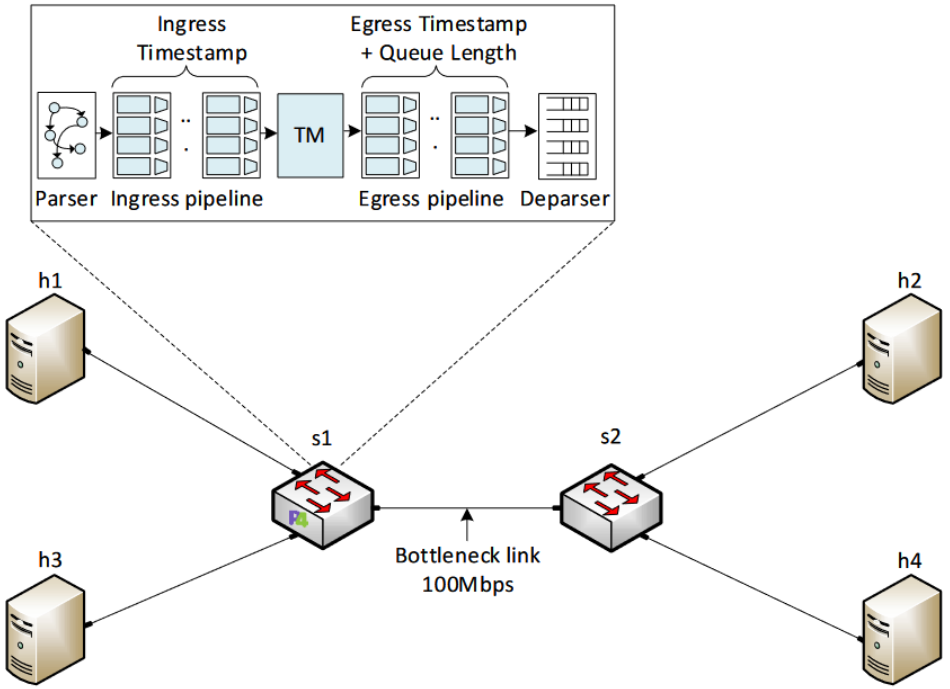
- Topology constructed with a modified version of the MiniEdit editor
- P4 software switches (BMv2) running inside Docker containers (through Containernet)
- Code written in Visual Studio Code with P4 syntax highlighting and a built-in terminal





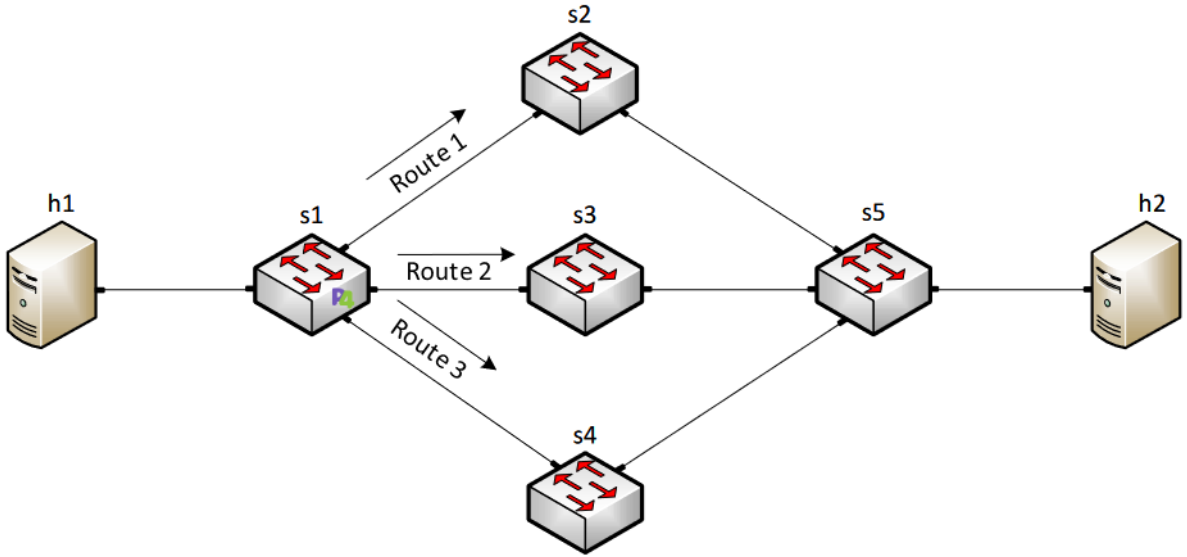
Examples

- Switch standard metadata contains the enqueueing and dequeuing timestamps
- Using these timestamps, we can compute the queueing delay



Examples

- Using meters to determine the sending rate of host h1 and reroute the traffic
  - Route 1: if the sending rate is less than 100Mbps.
  - Route 2: if the sending rate is between 100Mbps and 500Mbps
  - Route 3: if the sending rate is greater than 500Mbps



Additional Information

- Jorge Crichigno:
  - jcrichigno@cec.sc.edu
- Cyberinfrastructure lab at the University of South Carolina:
  - http://ce.sc.edu/cyberinfra/