

MUNI
FI

Asset Management – Seminar

PA211 Advanced Topics of Cyber Security

September 20, 2022

Lukáš Sadlek, Pavel Čeleda, and Jan Vykopal

Goals of this tutorial

- Become acquainted with
 - **Asset discovery** using network scanners, IP flow, and system logs
 - Netbox **asset inventory**
 - Common Platform **Enumeration**
 - **ELK** stack and Kibana user interface

Prerequisites

Prerequisites – I

1. Run `pa211_setup` command on a school computer.
2. Change your working directory to the clone of repository from the previous week
<https://gitlab.fi.muni.cz/cybersec/pa211/management.git>
3. Run `git pull`.
4. Change directory to `dist`. This directory should contain **Vagrantfile**.
5. Run `vagrant up`.
6. You can **log to the Kali host student** using credentials `kali:kali`. You may need to **login twice**.

Prerequisites – II

- Verify that all network services are accessible:
 1. <http://localhost:8000> for **Netbox asset inventory** (credentials are *admin:admin*),
 2. <http://elk:5601> for **Kibana user interface** from **student**.
- Use **port forwarding** command to access services from your host:
 1. `vagrant ssh elk -- -L 5601:localhost:5601`
 2. `vagrant ssh student -- -L 8000:localhost:8000`
- Access from your host **is faster and more comfortable**
 - Access the services using `http://localhost:<port>`
- In a case of issues, look into **docker logs** for containers

Troubleshooting – I

– **Destroy and create** a virtual machine:

- `vagrant destroy <machine_name> -f`
- `Vagrant up <machine_name>`

– **Rerun ansible tasks, if ansible script failed:**

- `vagrant provision <machine_name>`

– **Start all** containers:

- `sudo docker start $(sudo docker ps -aq)`

– **List all** (not only running) containers:

- `sudo docker container ls -a`

Troubleshooting – II

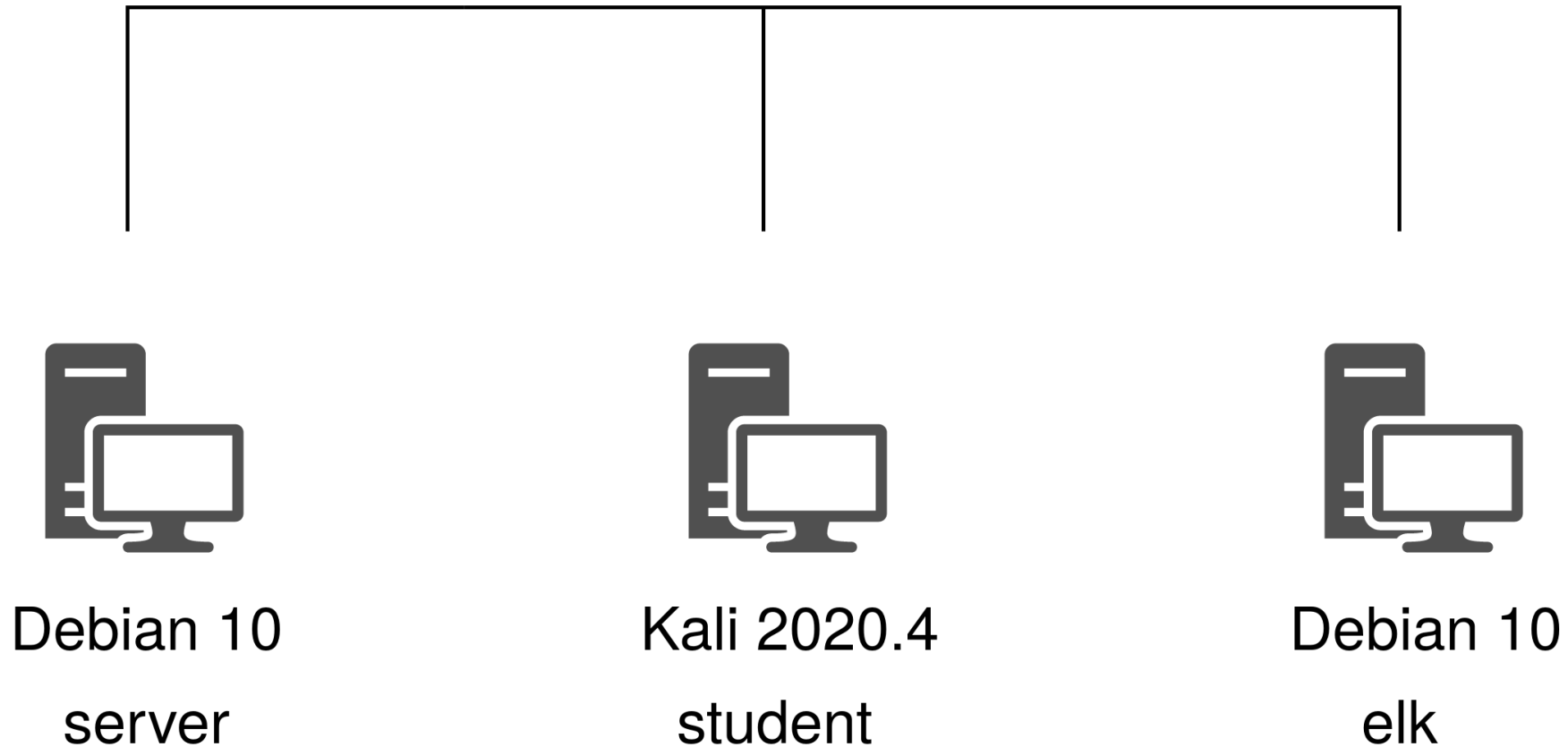
– List **open ports** on device:

– `sudo netstat -tulpn`

– **Check logs** of a specific **container** for issues:

– `sudo docker logs <container_id>`

Topology



Nmap network scanner

Task 1 – asset discovery using Nmap

IP address of your Kali machine **student** is `10.1.26.23`. Use Nmap scanner to enumerate

- **hosts in the subnet** of your PC having the same first three parts of IP address,
- their **open ports, network services, and operating systems**.

In other words, accomplish horizontal and vertical scan. Documentation is available on [\[1\]](#).

Solution 1

– Several solutions

- `nmap 10.1.26.23/24 -sV`
- `nmap 10.1.26.23/24 -A`

– IP addresses

- 10.1.26.2 (server)
- 10.1.26.9 (elk)
- 10.1.26.23 (student)

– Services

– OS Linux

- For each IP address

```
└─$ nmap 10.1.26.23/24 -sV
Starting Nmap 7.92 ( https://nmap.org ) at 2022-09-16 13:38 CEST
Nmap scan report for server (10.1.26.2)
Host is up (0.00083s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for elk (10.1.26.9)
Host is up (0.00081s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
9200/tcp  open  http     Elasticsearch REST API 7.12.1 (name: es01; cluster-cluster; Lucene 8.8.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap scan report for student (10.1.26.23)
Host is up (0.00077s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.0p1 Debian 1 (protocol 2.0)
5900/tcp  open  vnc      VNC (protocol 3.8)
8000/tcp  open  http-alt Unit/1.27.0
```

Optional: Task 2 – asset identification

Use Nmap for **banner grabbing** and **OS fingerprinting**.

- a) SSH versions from the previous task were very long. **Determine service banner of SSH service** on server (10.1.26.2) using appropriate Nmap's script.
- b) Show results of **OS detection** using Nmap's **verbose** output option for **10.1.26.23**. Find the appropriate **TCP / IP stack fingerprint** in the output.

Optional: Solution 2 a)

- Nmap **internally** uses banner grabbing
 - `nmap --script=banner 10.1.26.2`
- Outputs from Task 1 and Task 2a) are **almost identical**

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
```

```
PORT      STATE SERVICE
22/tcp    open  ssh
|_banner: SSH-2.0-OpenSSH_7.9p1 Debian-10+deb10u2
```

Optional: Solution 2 b)

– Nmap **internally** uses TCP/IP fingerprints for **OS detection**

– `sudo nmap -O 10.1.26.23 -vv`

– Parts of the **fingerprint** (e.g., SCAN, SEQ, WIN) are **explained** in [\[1\]](#)

```
OS CPE: cpe:/o:linux:linux_kernel:2.6.32
OS details: Linux 2.6.32
TCP/IP fingerprint:
OS:SCAN(V=7.91%E=4%D=9/7%OT=22%CT=1%CU=44371%PV=Y%DS=0%DC=L%G=Y%TM=63185A72
OS:%P=x86_64-pc-linux-gnu)SEQ(SP=108%GCD=1%ISR=109%TI=Z%CI=Z%II=I%TS=A)OPS(
OS:01=MFFD7ST11NW7%02=MFFD7ST11NW7%03=MFFD7NNT11NW7%04=MFFD7ST11NW7%05=MFFD
OS:7ST11NW7%06=MFFD7ST11)WIN(W1=FFCB%W2=FFCB%W3=FFCB%W4=FFCB%W5=FFCB%W6=FFC
OS:B)ECN(R=Y%DF=Y%T=40%W=FFD7%O=MFFD7NNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=0%A=
OS:S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=AA%Z=F=R%O=%RD=0%Q
OS:=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A
OS:%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y
OS:%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T
OS:=40%CD=S)
```

Common Platform Enumeration

- **Describes classes** of applications, operating systems, and hardware devices

cpe:2.3:<part>:<vendor>:<product>:<version>:<update>:<edition>
:<language>:<sw_edition>:<target_sw>:<target_hw>:<other>

Task 3 – CPE

Nmap can output CPE identifiers of operating systems.

- a) Find **CPE match strings** present in the output of previous Nmap commands. Was there anything interesting about the **length of CPE identifiers**?
- b) **What CPE match string** returned Nmap for Debian operating system?

Task 3 – solution

- a) Nmap outputs **fields up to version** (see Solution 1)
- b) String contains only ***linux_kernel* instead of *debian_linux***
 - ***Minor*** imprecisions can appear
 - We **cannot** reveal all information about OS from the outside

Examples from the NVD:

- **both** are specified as CPEs – [\[1\]](#), [\[2\]](#)
- **other** Linux operating systems – [\[3\]](#)
- Debian **without** Linux kernel – [\[4\]](#)

Netbox asset inventory

Netbox asset inventory

- **IP address management (IPAM)**
 - planning, tracking, and managing **IP address space**
 - **Netbox**: IP addresses, prefixes, ASNs (Autonomous System Number), and other
- **Data center infrastructure management (DCIM)**
 - Manages **hardware** inside of **data centers** – energy, equipment, floor space
 - **Netbox**: devices, facilities, power consumption, and other
- Can be accessed via **REST API** and using **Python client** [\[1\]](#)
- **Public demo** [\[2\]](#), **repository** [\[3\]](#), and its **documentation** [\[4\]](#)

Netbox asset inventory – main menu

The screenshot displays the Netbox main menu with a dark theme. The left sidebar contains navigation links for Organization, Devices, Connections, Wireless, IPAM, Overlay, Virtualization, Circuits, Power, and Other. The main content area is divided into several panels, each representing a different asset category with a list of sub-items and their respective counts.

Category	Sub-item	Count
Organization	Sites	0
	Tenants	0
Inventory	Racks	0
	Device Types	0
	Devices	0
Wireless	Wireless LANs	0
	Wireless Links	0
IPAM	VRFs	0
	Aggregates	0
	Prefixes	0
	IP Ranges	0
	IP Addresses	2
Power	Power Panels	0
	Power Feeds	0
Virtualization	Clusters	0
	Virtual Machines	0
Circuits	Providers	0
	Circuits	0
Connections	Cables	0
	Console	0
	Interfaces	0
	Power Connections	0

Task 4 – IPAM

Create essential **IPAM items** in the Netbox asset inventory.

- a) Fill information about hosts (**IP addresses, hostnames, prefixes**) to the asset inventory. For other fields, fill in only information that you are sure about. Otherwise, use **default options** or None value.
- b) What **novel functionality** compared to an ad-hoc list of assets (e.g., Excel sheet) does the **asset inventory** provide?

Solution 4 a)

- Should be similar to **screenshots**
- **Detailed view** of 10.1.26.2/24:

The screenshot displays a network management interface with three main sections:

- IP Address:** A metadata table for the IP address 10.1.26.2/24.
- Parent Prefixes:** A table showing the parent prefix 10.1.26.0/24 with an 'Active' status and 0 children.
- Related IP Addresses:** A table listing related IP addresses: 10.1.26.9/24 (ELK stack server) and 10.1.26.23/24 (Student device), both with 'Active' status.

IP Address	
Family	IPv4
VRF	Global
Tenant	—
Status	Active
Role	—
DNS Name	server
Description	Debian server
Assignment	—
NAT (inside)	—
NAT (Outside)	—

Parent Prefixes								
Prefix	Status	Children	Tenant	Site	VLAN	Role	Description	
10.1.26.0/24	Active	0	—	—	—	—	PA211 sandbox subnet	

Related IP Addresses								
IP Address	VRF	Status	Role	Tenant	Assigned	DNS Name	Description	
10.1.26.9/24	Global	Active	—	—	—	elk	ELK stack server	
10.1.26.23/24	Global	Active	—	—	—	student	Student device	

Solution 4 a), b)

- **List of IP addresses** from Task a) is in Figure
- Asset inventory **automates manual tasks**
 - **Example:** assignment of IP addresses to subnet

<input type="checkbox"/> IP Address	VRF	Status	Role	Tenant	Assigned	DNS Name	Description
<input type="checkbox"/> 10.1.26.2/24	Global	Active	—	—	—	server	Debian server
<input type="checkbox"/> 10.1.26.9/24	Global	Active	—	—	—	elk	ELK stack server
<input type="checkbox"/> 10.1.26.23/24	Global	Active	—	—	—	student	Student device

Task 5 – DCIM

Netbox provides support for **virtual machines** and **devices**.

a) Fill in necessary information about **virtual machine**

`student` (**10.1.26.23**) and its **network services**.

b) **Optional:** How would you fill in **necessary information** about your **device** `nymfeXY`?

Solution 5 a) – virtual machine

Virtual Machine

Name	student
Status	Active
Role	—
Platform	Kali 2020.4
Tenant	—
Primary IPv4	10.1.26.23
Primary IPv6	—

Cluster

Site	Faculty of Informatics
Cluster	Local cluster
Cluster Type	Virtualbox
Device	nymfeXY

Resources

Virtual CPUs	1.00
Memory	3 GB
Disk Space	16 GB







Tags

No tags assigned

Comments

None

Services

SSH	TCP	22	10.1.26.23	  
VNC	TCP	5900	10.1.26.23	  

Solution 5 a) – interface



The screenshot shows a web-based configuration interface for a virtual machine. At the top, there are navigation tabs: "Virtual Machine", "Interfaces 1", "Config Context", "Journal", and "Change Log". Below the tabs is a "Quick search" input field. The main content is a table with the following columns: Name, Enabled, MAC Address, MTU, Mode, Description, and IP Addresses. A single row is visible for the interface "eth1", which is enabled and has the IP address "10.1.26.23/24".

<input type="checkbox"/> Name	Enabled	MAC Address	MTU	Mode	Description	IP Addresses
<input type="checkbox"/> eth1	✓	—	—	—	—	10.1.26.23/24

Solution 5 a) – interface and services

- Assigning **IP address** to a **virtual machine**
 - **Create an interface** of a virtual machine
 - The **ifconfig** command reveals the network **interface** name (**eth1**) for 10.1.26.23
 - An IP address is **assigned** to the interface in **IPAM** part of menu during **creation** or **editing**
 - **The form** for editing the IP address contains the section **Interface Assignment**
- Use the **custom option** for creating services

Optional: Solution b)

Netbox is much **network-centric** and we have **non-racked** device. Device will have **name** `nymfeXY`, **platform** Ubuntu 22.04, some **manufacturer** and **model**. **Site** could be Faculty of Informatics, **location** PA211's room. A lot of fields will be **empty** because we do not have **knowledge about hardware**. See **public demo** for inspiration [\[1\]](#).

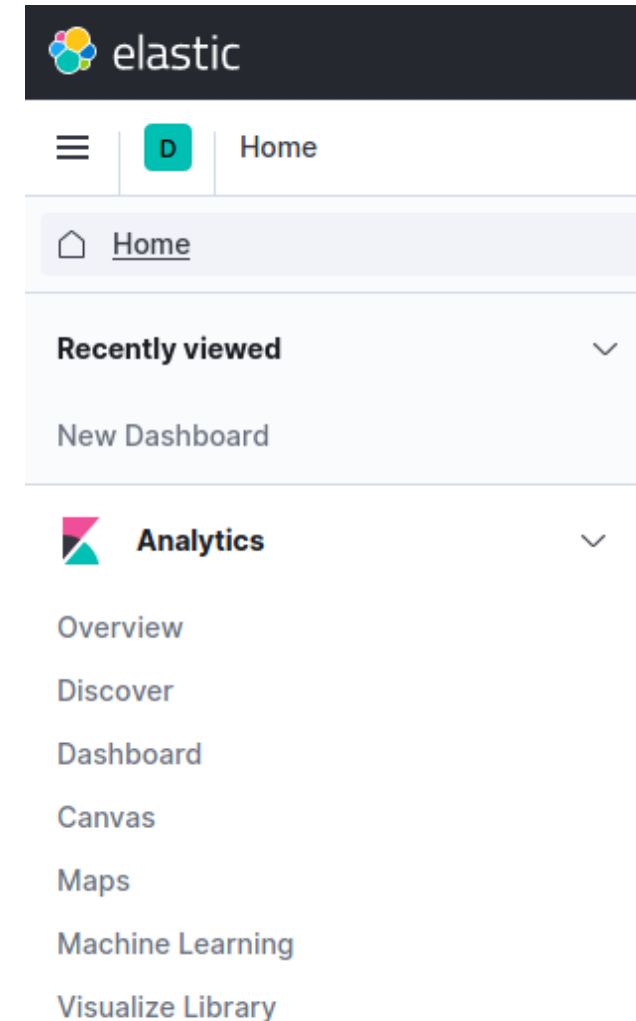
Elasticsearch and Kibana

ELK stack

- Acronym for three **open-source** projects:
 - **Elasticsearch** – RESTful **search** and analytical **engine**
 - **Kibana** – **visualizes data** from Elasticsearch
 - **Logstash** – **processing pipeline** that ingests data from multiple sources, transforms them, and sends to Elasticsearch
- **Beats** – lightweight single-purpose **data shippers**
- One of use cases is **security**
 - **Elastic SIEM** to prevent, detect, and respond to threats

Kibana – main menu

- **Quick start tutorial** for the most recent version of **Kibana** on [\[1\]](#)
- **Analytical** section in menu
 - **Discover** section
 - **Dashboard** section



Discover section in Kibana – I

The screenshot shows the Kibana Discover interface. At the top, a search bar contains the query `applicationName: DNS HTTP`. Below it, a filter bar shows `flow-2019-03-19` selected, with a dropdown menu open showing options: `flow-2019-03-19` (checked) and `syslog-2019-03-19`. A sidebar on the left lists field types: `_score` (number), `_type` (text), `applicationId` (text), `applicationName` (text), `bgpDestinationAsNumber` (number), and `bgpSourceAsNumber` (number). The main area displays 13 hits of search results. Each hit is a JSON object with fields like `applicationName`, `applicationId`, `biFlowStartMilliseconds`, `destinationIPv4Address`, and `extendedFlow.http`. The `applicationName` field is highlighted in yellow in each hit, indicating the search term.

```
applicationName: HTTP applicationId: 50331728 applicationId.keyword: 50331728 applicationName.keyword: HTTP bgpDestinationAsNumber: 3,356
biFlowStartMilliseconds: 1,552,989,982,120 destinationIPv4Address: 4.122.55.2 destinationIPv4Address.keyword: 4.122.55.2 destinationTranspc
exercise_dst_ipv4_segment.keyword: global extendedFlow.http:
{"uaAppMin":65535,"ua0s":65535,"ua0sBld":65535,"host":"0x63686d656c676c66f62652e6578","uaApp":65535,"methodMask":32769,"uaAppBld":65535,"ua0sMaj":65535}
extendedFlow.http.keyword:

applicationName: HTTP applicationId: 50331728 applicationId.keyword: 50331728 applicationName.keyword: HTTP bgpDestinationAsNumber: 3,356
biFlowStartMilliseconds: 1,552,989,982,133 destinationIPv4Address: 4.122.55.2 destinationIPv4Address.keyword: 4.122.55.2 destinationTranspc
exercise_dst_ipv4_segment.keyword: global extendedFlow.http:
{"uaAppMin":65535,"ua0s":65535,"ua0sBld":65535,"host":"0x676f766636572742e6578","uaApp":65535,"methodMask":32769,"uaAppBld":65535,"ua0sMaj":65535}
extendedFlow.http.keyword:

> applicationName: HTTP applicationId: 50331728 applicationId.keyword: 50331728 applicationName.keyword: HTTP bgpDestinationAsNumber: 0 bgpSourceAsNumber: 0
biFlowStartMilliseconds: 1,552,989,982,090 destinationIPv4Address: 9.66.11.12 destinationIPv4Address.keyword: 9.66.11.12 destinationTranspc
exercise_dst_ipv4_segment.keyword: blue-team-1 extendedFlow.http:
```

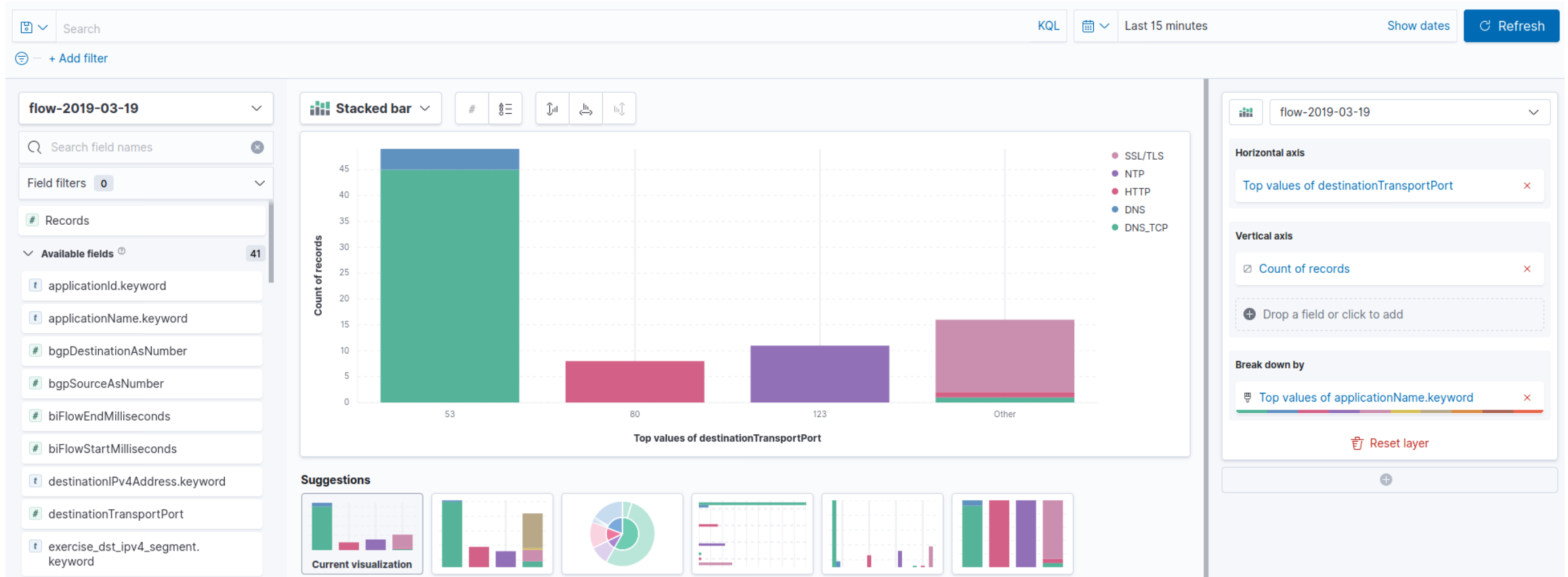

Discover section in Kibana – II

- Lists all entries in specific **index patterns**
- Uses queries in **Kibana Query Language (KQL)**
- **KQL**
 - Data **field name** is followed by **:** and **values separated by spaces**
 - Other parts can be added using **and / or keywords**
 - Kibana **automatically suggests** options for your query

Dashboard in Kibana – I

- **Lens** – dashboard panel **recommended** for most users
- Provides several **chart types**
- Possibility to **correlate several fields** in one chart
- **Video tutorial** about Kibana Lens by Elastic [\[1\]](#)
- Lens panel **can be saved** and modified later

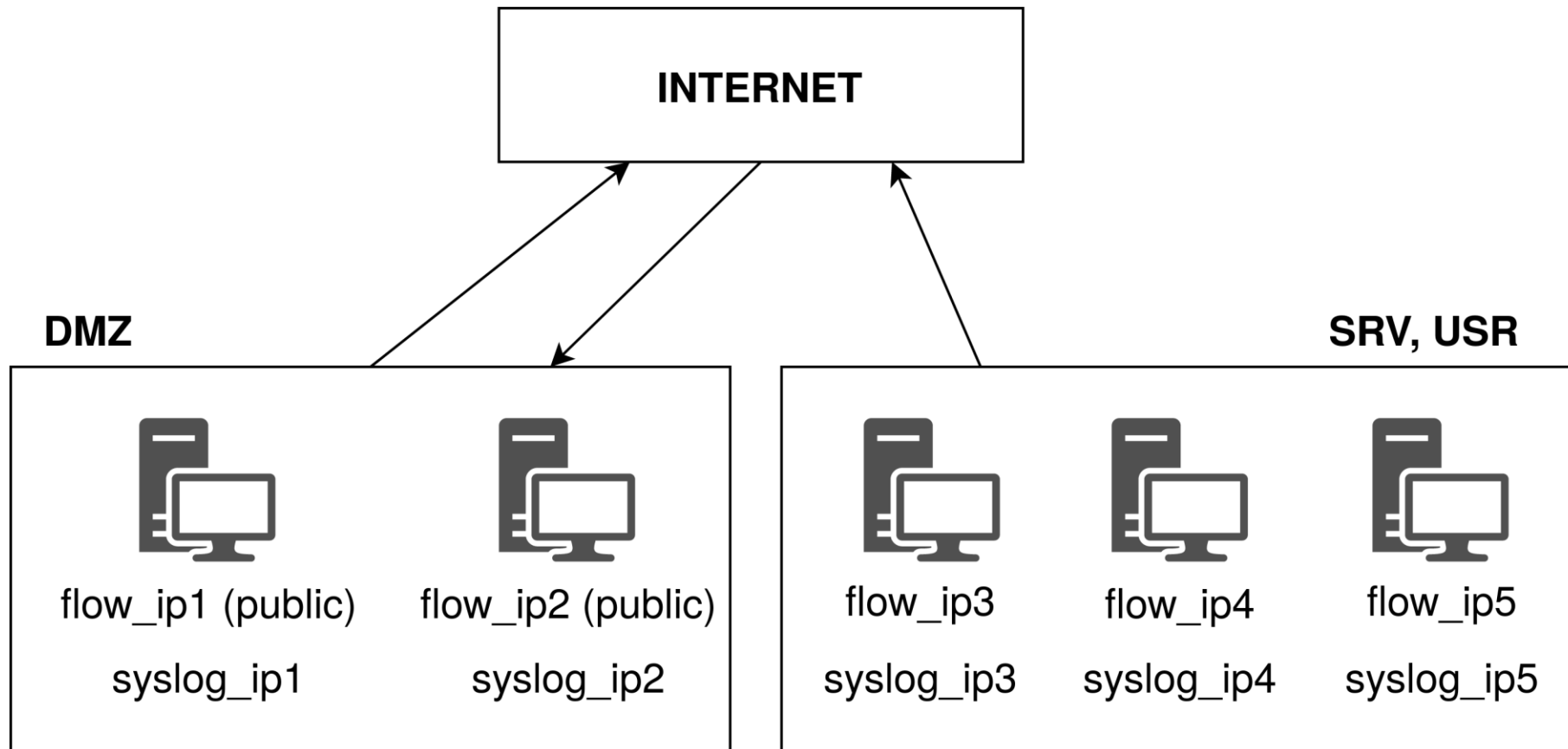
Dashboard in Kibana – II



Task 6 – multiple data sources (description)

- Your network subnet contains **5 devices**
- Identify their properties using captured **IP flow** and **syslog** dataset
- **IP flow dataset**
 - Source and destination **IP addresses** and **ports**
 - Protocol, timestamp, and other information
- **Syslog dataset**
 - Host IP
 - Hostname
 - Timestamp and other information

Task 6 – topology



Task 6 – IP addresses

- a) Determine IP addresses from **syslog dataset** for each device. Add only their **IP addresses, hostnames, and name of /24 network subnet** to the inventory.
- b) IP flow dataset contains **different IP addresses** than syslog dataset. **Determine two IP addresses** of devices from DMZ zone and the other three IP addresses. **Hint: arrows** in Figure containing topology denote **direction of communication** captured in **IP flow** dataset.

Solution 6 a)

Data table ▾

Top values of fromhost_ip.keyword ▾	Top values of hostname.keyword ▾	Top values of exercise_segment.keyword ▾
10.7.101.44	desktop3	blue-team-1
10.7.101.12	mail	blue-team-1
10.7.101.26	db	blue-team-1
10.7.101.13	dns	blue-team-1
10.7.101.49	admin4	blue-team-1

syslog-2019-03-19 ▾

Break down by

- Top values of fromhost_ip.keyword ×
- Top values of hostname.keyword ×
- Top values of exercise_segment.keyword ×

+ Drop a field or click to add

Metrics

- Count of records ×

Solution 6 a)

IP Address

Family	IPv4
VRF	Global
Tenant	—
Status	Active
Role	—
DNS Name	mail
Description	—
Assignment	—
NAT (inside)	—
NAT (Outside)	—

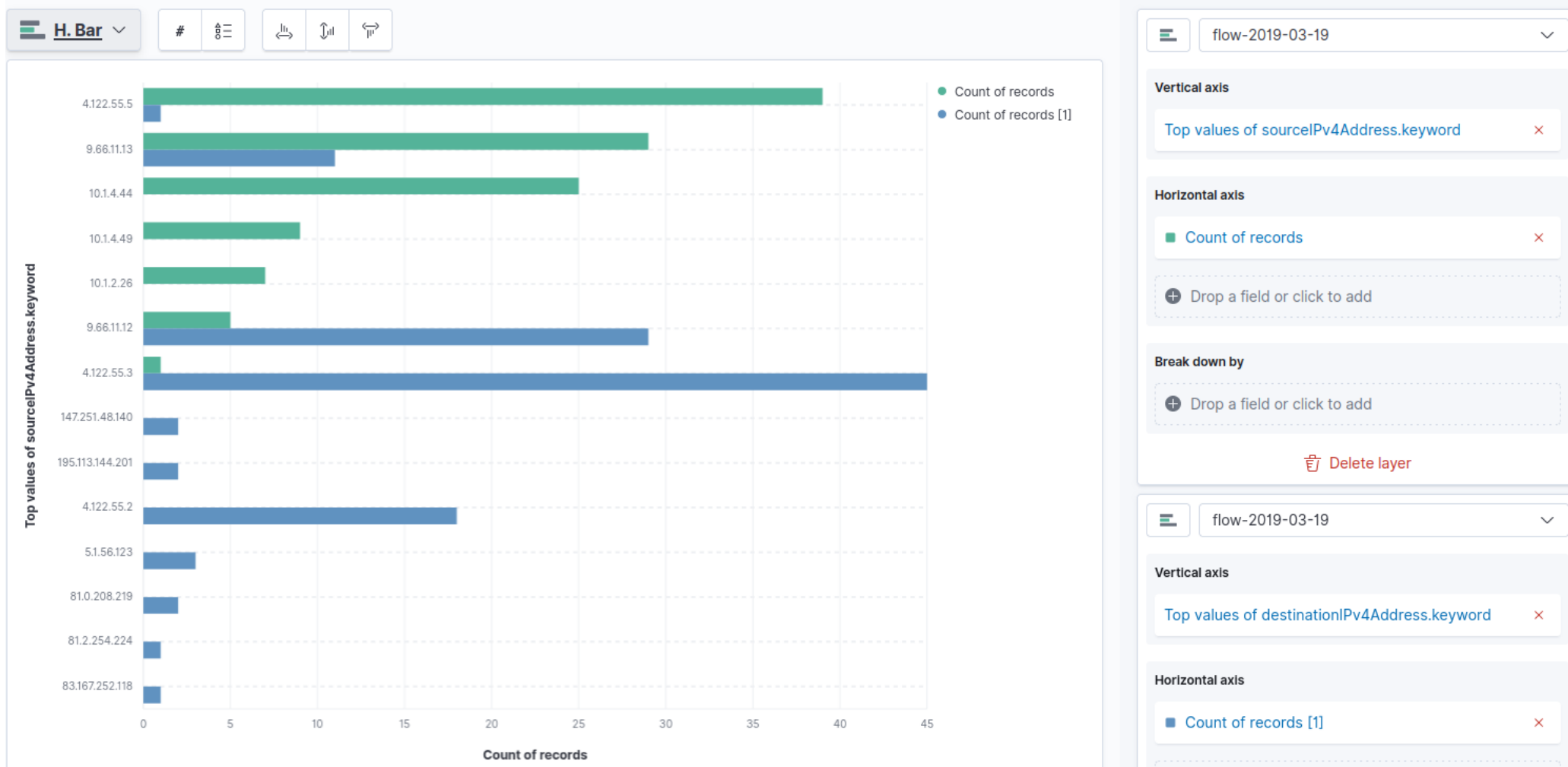
Parent Prefixes

Prefix	Status	Children	Tenant	Site	VLAN	Role	Description	
10.7.101.0/24	Active	0	—	—	—	—	blue-team-1	

Related IP Addresses

IP Address	VRF	Status	Role	Tenant	Assigned	DNS Name	Description	
10.7.101.13/24	Global	Active	—	—	—	dns	—	
10.7.101.26/24	Global	Active	—	—	—	db	—	
10.7.101.44/24	Global	Active	—	—	—	desktop3	—	
10.7.101.49/24	Global	Active	—	—	—	admin4	—	

Solution 6 b) – I



Solution 6 b) – II

- Consider the **direction** of communication
- **Two** devices from DMZ can be **sources** and **destinations** of communication
- **Three** devices from SRV and USR are **only sources** of communication
- Source IP addresses contain **three groups** based on **prefixes**
 - 4.122.55.5, 4.122.55.3
 - 9.66.11.12, 9.66.11.13
 - 10.1.4.26, 10.1.4.44, 10.1.4.49

Solution 6 b) – III

- Only the third group does not contain **destination IP addresses**
- The **first** and the **second** group contain **also destination** addresses
- **Destination IP** addresses contain the address `4.122.55.2`
 - It has the **same prefix** as the **first** group
- **DMZ zone** contains only **two** devices
 - Its **IP addresses** belong to the **second** group
- **DMZ IP addresses:** `9.66.11.12, 9.66.11.13`
- **SRV, USR IP addresses:** `10.1.4.26, 10.1.4.44, 10.1.4.49`

Task 7 – merging data

- a) For two devices from **demilitarized zone**, determine their **public IP address** from IP flow dataset and **IP address from syslog dataset** according to the most relevant **ports, service names, or function** in the network.
- b) **Optional:** Store both IP addresses in Netbox using NAT (Network Address Translation).
- c) **Optional:** You obtained **alert about incident** from some IP address. The asset inventory directly provides contact to device's owner. What other sources of data would you use if you do not have asset inventory?

Solution 7 a) – I

– Query:

– destinationIPv4Address : 9.66.11.12 or destinationIPv4Address : 9.66.11.13

– Destination ports:

– 9.66.11.13 mainly 53,
– 9.66.11.12 uses mainly ports for transmission of emails [\[1\]](#)

– **Hostnames** from syslog: mail, dns, db, desktop3, admin4

– Possible to use also **application names** and **messages** from syslog

Solution 7 a) – II

– Results:

- 10.7.101.12 and 9.66.11.12, **hostname:** mail
- 10.7.101.13 and 9.66.11.13, **hostname:** dns

Optional: Solution 7 b)

<input type="checkbox"/> IP Address	VRF	Status	Role	Tenant	Assigned	DNS Name	Description	NAT (Inside)	NAT (Outside)
<input type="checkbox"/> 9.66.11.12/24	Global	Active	—	—	—	mail	—	10.7.101.12/24	—
<input type="checkbox"/> 9.66.11.13/24	Global	Active	—	—	—	dns	—	10.7.101.13/24	—
<input type="checkbox"/> 10.7.101.12/24	Global	Active	—	—	—	mail	—	—	9.66.11.12/24
<input type="checkbox"/> 10.7.101.13/24	Global	Active	—	—	—	dns	—	—	9.66.11.13/24
<input type="checkbox"/> 10.7.101.26/24	Global	Active	—	—	—	db	—	—	—
<input type="checkbox"/> 10.7.101.44/24	Global	Active	—	—	—	desktop3	—	—	—
<input type="checkbox"/> 10.7.101.49/24	Global	Active	—	—	—	admin4	—	—	—

Optional: Solution 7 c)

- Asset inventory provides all information for **quick access**
- Example source are **authentication logs**
 - Contain **IP address** and **login name**
 - Login names must be **unified**, e.g., **UČO** for **eduroam**

How was it today?

Please fill in an **anonymous** exit ticket:

<https://muni.cz/go/pa211-22-02>



M U N I

F I