CTF Training
Defense and network monitoring

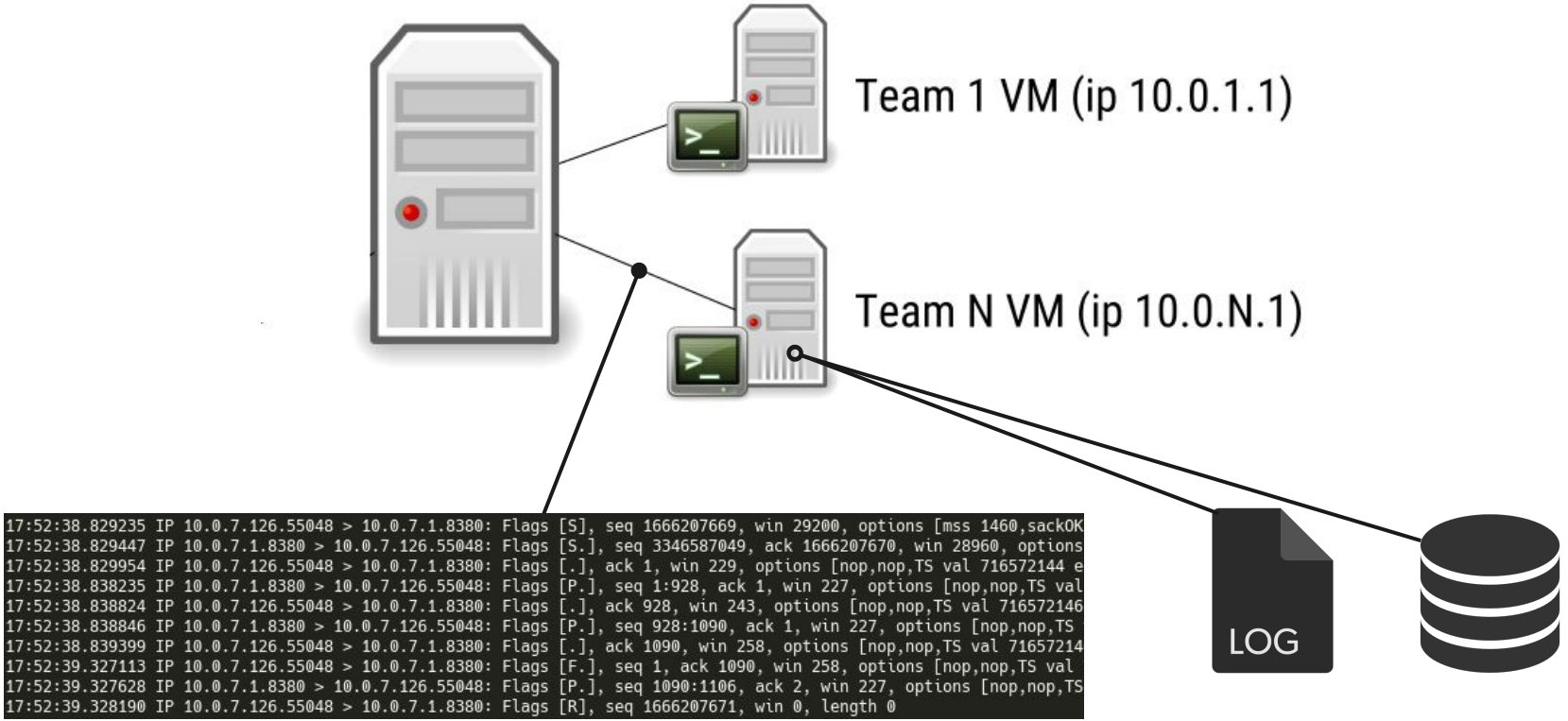
Security 2 2018-19

Università Ca' Foscari Venezia
www.dais.unive.it/~focardi
secgroup.dais.unive.it

Defense

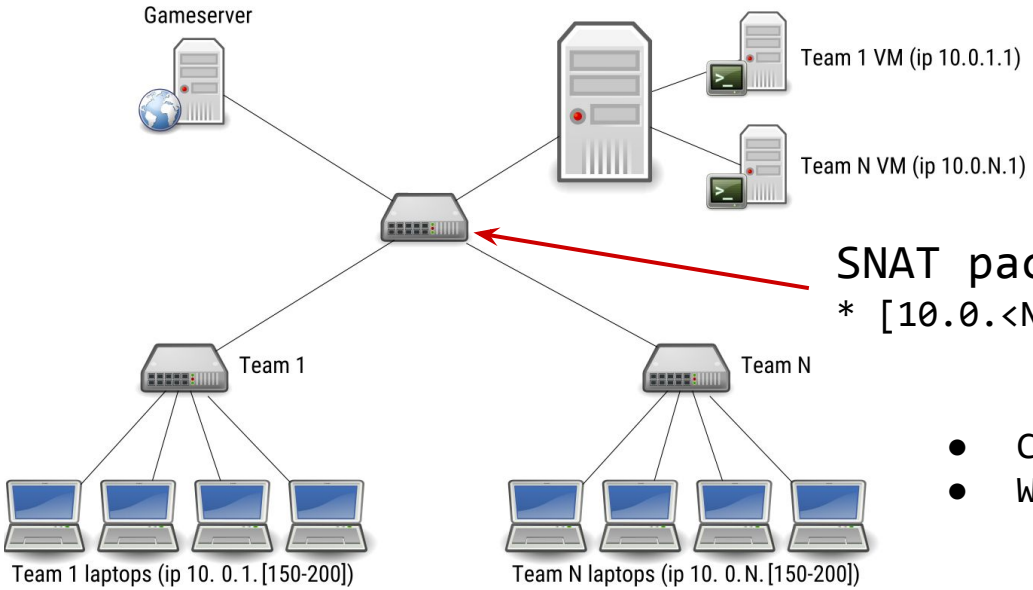
Strategies

1. Detection
2. Mitigation
3. Patching



Defense

Network monitoring



SNAT packets directed to VMs

* [10.0.<N>.126] > 10.0.<N>.0/24

- Cannot use addresses to detect attacks
- We must look at the payloads
  - Spot suspicious behavior
  - Patch/Mitigate
  - Copy and replicate the attack ;)

Defense

Network monitoring

Avoid saving your own ssh session

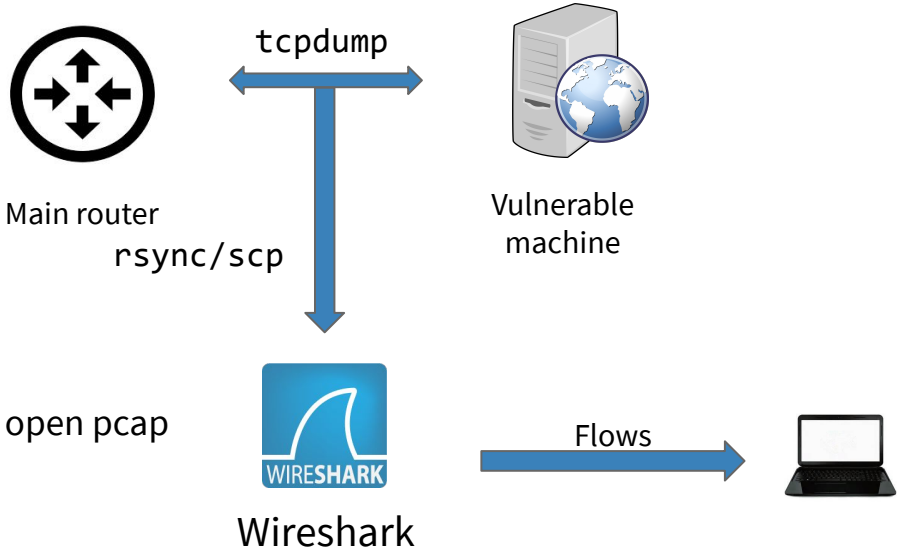
- use the tcpdump tool
  `tcpdump -i ens4 -s0 -w traffic.pcap not port 22`
  - Save the network traffic from interface ens4 into the `traffic.pcap` file
- Analyze the collected packets using `wireshark`

or

- use the tcpflow tool
  `tcpflow -i eth0`
  - saves in the current directory the reassembled flows

Network Monitoring

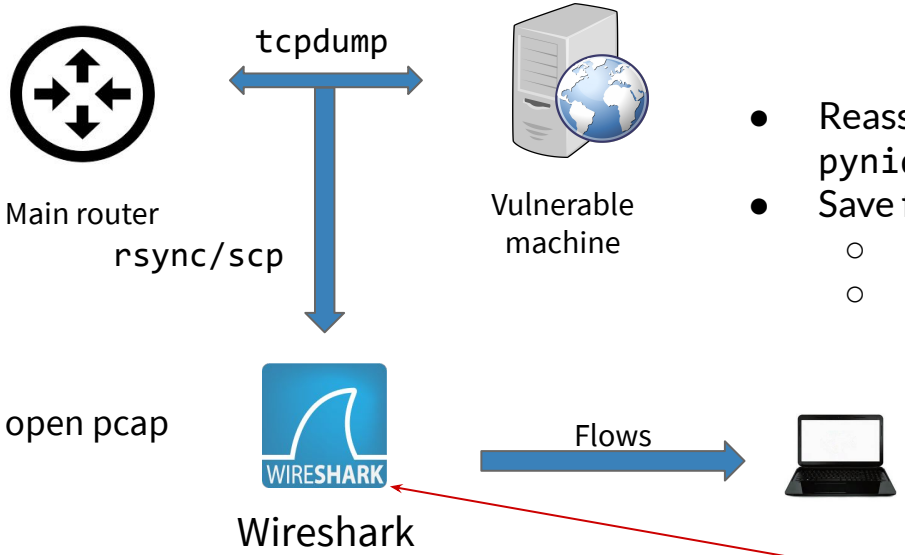
Suggested/Typical flow



DEMO

Network Monitoring

Suggested/Typical flow



- Reassemble tcp flows and make the search faster! `pynids` (https://github.com/MITRECND/pynids)
- Save flows to file or to a DB
  - file -> same as tcpflow
  - DB -> faster search + better visualization + scripting

Network Monitoring

pyNIDS example

Show raw http requests from pcap files

```
import sys, nids

def handle_tcp_stream(tcp):
	if tcp.nids_state == nids.NIDS_JUST_EST:                                        # New connection
		((src, sport), (dst, dport)) = tcp.addr
		tcp.client.collect = 0 # Collect only one side
		tcp.server.collect = 1
	elif tcp.nids_state == nids.NIDS_DATA:                                          # New packet
		tcp.discard(0) # keep all of the stream's new data
	elif tcp.nids_state in (nids.NIDS_CLOSE, nids.NIDS_TIMEOUT, nids.NIDS_RESET): # Connection closed
		print "addr:", tcp.addr
		print "Request:"
		print tcp.server.data[:tcp.server.count]

def main():
	nids.param("pcap_filter", "tcp port 80")  # Filter only packets to port 80
	nids.chksum_ctl([('0.0.0.0/0', False)])   # Disable checksumming
	nids.param("filename", sys.argv[1])    # read file at argv[1]
	nids.init()
	nids.register_tcp(handle_tcp_stream)
	nids.run()

if __name__ == '__main__':
	main()
```

Attack Mitigation

firewalls

- We cannot DROP connections using IP addresses
  - We must match packet payloads
  - and DROP them if they contain a particular sequence of bytes

IPTABLES

http://ipset.netfilter.org/iptables-extensions.man.html

```
iptables -N log_drop
iptables -A INPUT -p tcp --dport 80 -m string --algo bm --string ‘<string>’ -j log_drop
...
iptables -A log_drop -j LOG --log-prefix "DROP string: "
iptables -A log_drop -j DROP
```

MIGNIS

```
* / local:80 | -m string --algo bm --string ‘<string>’
```

System Administration

Small intro to docker

`docker-compose.yaml`
Defines a multi-container application

```yaml
version: '3'
services:

  nginx:
    image: nginx
    volumes:
     - ./htdocs:/var/www/htdocs
     - ./nginx.conf:/etc/nginx/nginx.conf
    ports:
     - 80:80
  php:
    build: ./php-mysql
    volumes:
     - ./htdocs:/var/www/htdocs
  mysql:
    build: ./db
```

`Dockerfile`
Defines a single container

```
FROM php:5-fpm
RUN docker-php-ext-install mysql mysqli pdo pdo_mysql
```

`docker-compose up -d` start the application
`docker-compose restart` restart the application
`docker-compose stop` stop the application
`docker-compose exec mysql bash`
start bash inside mysql container

All above commands must be run in the folder that contains the `docker-compose.yaml` file.

`docker ps` show running containers and TCP/UDP ports
`docker exec -it <cnt> bash` start bash inside the container

Training CTF

- Team Extraction
- Training service: `tweeter`
- Network Monitoring
- Patch/Mitigation