Nebula Security Domains

Cybersecurity Project

Samuel Vojtáš

What is Nebula Network?

- Why do we need Nebula?
- Peer-to-peer software-defined network
- How does it work?
  - Certificate Authority (CA)
  - Cryptographic keys & certificates for the network hosts
  - Configuration files for the network hosts
- Connections between network clients
  - Lighthouse architecture

Goal of the Project

- We want to create *security domains*
  - = set of logically related resources that can communicate together
- Requirements
  - Resource can have multiple security domains
  - Security domains are visible from the certificate
  - Resources can only connect to resources sharing at least one security domain
- How to design this process?
- How to automate this process?

Design of Security Domains

- We will use Nebula *groups* and *firewall rules*
- Resource has *groups* field in the certificate
- For every *group* resource has, we can create a firewall rule in its configuration file to allow connections from resources having this *group*

Global Configuration File

- Stored in `PROJECT_DIR/conf/network-config.yaml`

```yaml
# Lighthouse configuration
lighthouse:
    # Name of the lighthouse
    name: "lighthouse"
    # IP address of the lighthouse on Nebula network
    ip: "192.168.100.1/24"
    # Public IP address of the lighthouse
    routable_ip: "164.90.172.129"
    routable_port: "4242"

# Resource configuration
resources:
    # Name of the resource
  - name: "laptop1"
    # IP address of the resource on Nebula network
    ip: "192.168.100.2/24"
    # Groups/security domains resource is part of
    groups:
    - laptops

    ...
```

Automatiotion Scripts

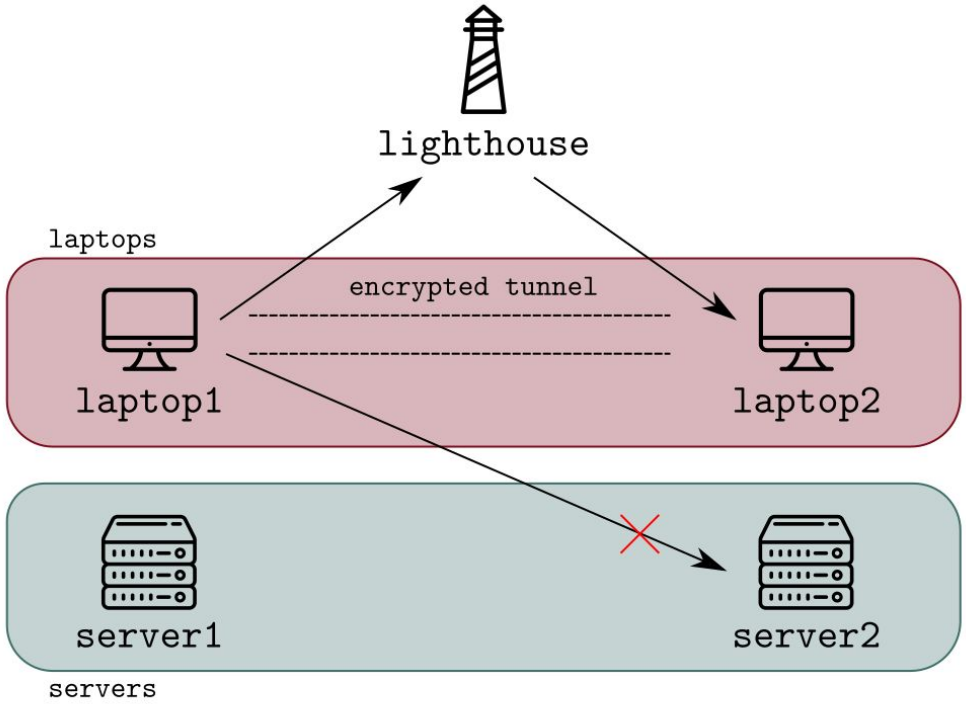
- `main.py`
  - `argument_parser.py`
  - `network_starter.py`
  - `container_starter.py`
- For network instantiation + testing phase, we can use `main.py`
- For network instantiation (to generate network files), `network_starter.py` is sufficient
- For container instantiation (testing with containers), `container_starter.py` is sufficient
- Each resource will have `host.crt`, `host.key`, `config.yaml`, and `ca.crt`

Resource’s Config File & Firewall List

```
# config.yaml: Configuration file of resource belonging to security domains:
# group1 & group2
...
firewall:
    outboud:
      # Allow all outbound traffic from the resource
      - port: any
        proto: any
        host: any
    inbound:
      # By default, allow all ICMP traffic to the resource
      - port: any
        proto: icmp
        host: any

      # Allow resources from group1 to contact the resource
      - port: any
        proto: any
        group: group1

      # Allow resources from group2 to contact the resource
      - port: any
        proto: any
        group: group2
      ...
...
```


lighthouse
laptops
encrypted tunnel
laptop1
laptop2
server1
server2
servers

Testing: Option #1

- Generate network files, instantiate containers for resources, instantiate container for lighthouse and run Nebula service

```
$ python3 ./src/main.py
```

Testing: Option #2

- Generate network files, instantiate containers for resources, wait for remote lighthouse and run Nebula service

```
$ python3 ./src/main.py --no-lighthouse-container
```

Testing: Option #3

- Only generate network files that can be later checked

```
$ python3 ./src/main.py --no-containers
```

- Information about security domain is included in certificate

```
$ ./bin/nebula-cert print -path ./generated/laptop1.crt

NebulaCertificate {
    …
    Groups: [
        “laptops”
    ]
    …
}
```

- Generated files then can be distributed to containers

```
$ python3 ./src/container_starter.py
```

Conclusion

- Additional security concept implemented on top of Nebula network – *security domains*
- Nebula network can be instantiated with automated scripts
- Several testing scenarios implemented

Questions