GlobalLogic

A Hitachi Group Company

EDUCATION

Typical application network stack

Andrii Beregovenko

- Typical TCP connection based communication
  - example HTTP query
  - analyze HTTP in Wireshark
  - trivial SMTP mail session
- Typical TCP session multi-connection pattern
  - Use telnet and ftp client for FTP protocol
- Typical UDP based protocol
  - Analyze DNS query in Wireshark
- Routing
  - redirect connection to the internet host through corporate VPN
- Filtering
  - blocking outcoming connection to mail server
  - rejecting with ICMP message
  - redirecting outcoming connection to the other host

Simple TCP based communication

Commands:

- Find out the mail server for a particular domain:

  ```
  dig mx protonmail.com
  ```
- Connect to SMTP server:

  ```
  telnet mail.protonmail.ch 25
  ```
- Say hello to the server:

  ```
  HELO globallogic.com
  ```
- Specify who is a mail sender

  ```
  MAIL FROM:<aberegovenko@globallogic.com>
  ```
- Specify the receiver of the email

  ```
  RCPT TO:<aberegovenko@protonmail.com>
  ```
- Start mail body

  ```
  DATA
  ```
- Finish message body with a following pattern: `<CR>.<CR>`
- Say goodbye to the server:

  ```
  QUIT
  ```

HTTP

Connect to server using telnet client:

```
telnet www.google.com 80
```

As soon as you will be connected you can start typing

```
GET / HTTP/1.1
Host: www.google.com
```

After that double-enter to signal remote server you are finished with your input

The web page with all its content and built-in scripts will appear.

TCP session with multi-connection

Using telnet for FTP

- To initiate connection run:

```
telnet ftp.us.debian.org 21
```

- Authorize on the server:

```
USER anonymous
PASS anonymous
```

- Check current directory and got to debian:

```
PWD
CWD debian
```

- Start passive mode:

```
PASV
```

- Calculate fetch port using two last numbers
- Initiate file transfer from remote server:

```
RETR README
```

- In a different terminal run data fetching:

```
telnet PROVIDED-IP CALCULATED-PORT > README
```

- Finish session:

```
QUIT
```

FTP client

Use of ftp client is way simpler and more friendly:

- To initiate connection run:
  ```
  ftp ftp.us.debian.org
  ```
- Use following as a login and a password: `anonymous`
- Show current directory:
  ```
  pwd
  ```
- Go to director on the server:
  ```
  cd debian
  ```
- List files in current directory:
  ```
  ls
  ```
- Initiate file transfer from remote site:
  ```
  get README
  ```
- Finish session:
  ```
  quit
  ```

UDP communication example

DNS query

- Run Wireshark application
- Start capturing on corresponding interface
- In a filter string enter filtering expression: `dns`
- Go to the console and run following command:

```
dig @8.8.8.8 mx google.com
```

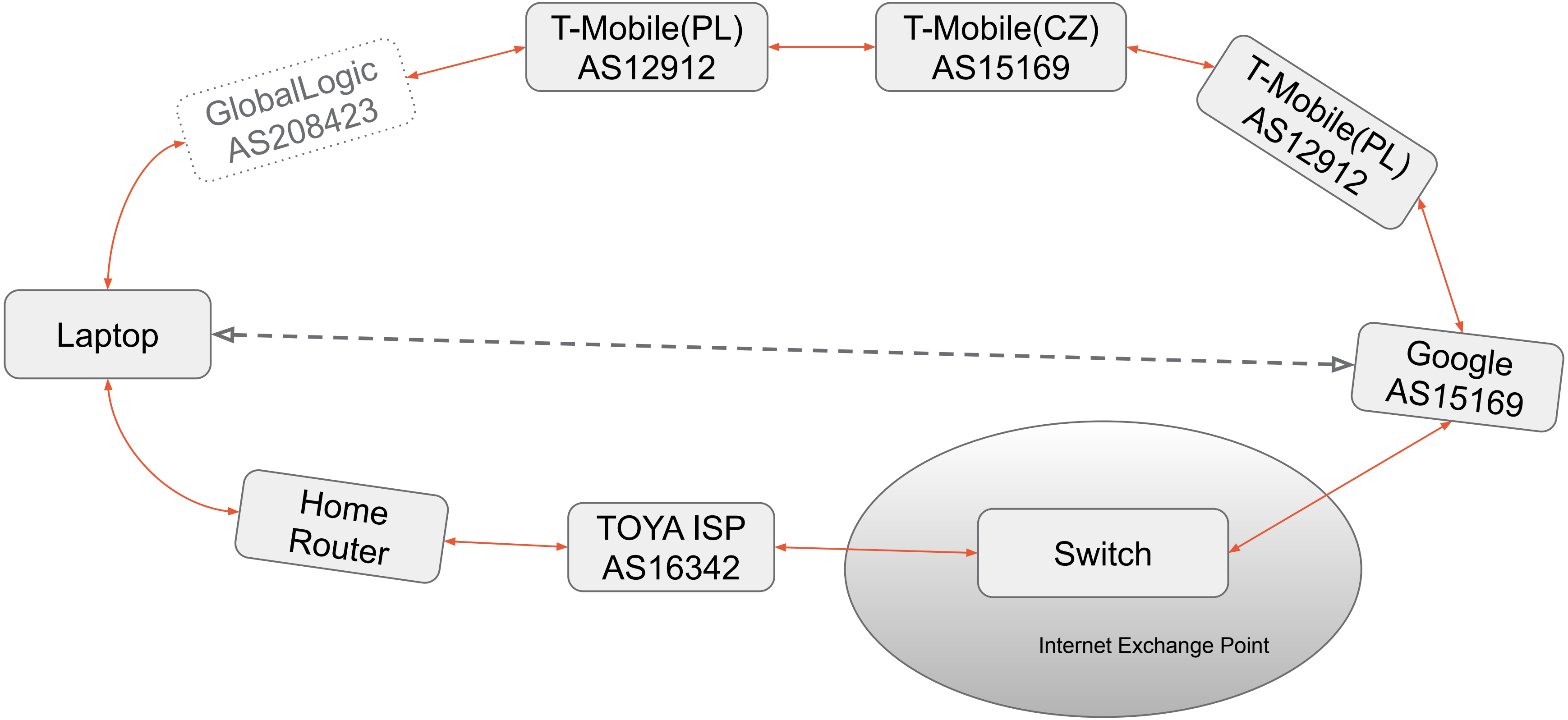
- Switch back to Wireshark and stop capturing
- Click on captured packet that contains a DNS query
  - using dissector window overview described request
- Click on captured packet that contains a DNS server response
  - using dissector window look over described reply

Routing example

Redirect traffic to go different path

- Using default system config analize what is the current path:
  ```
  ip ro get 8.8.8.8/32
  ```
- Using default config check the traffic path across the internet:
  `traceroute -I 8.8.8.8` or `mtr -r -c1 -n 8.8.8.8`
- Activate second connection, for example a VPN connection
- Setup a ip routing rule to route queries to IP host 8.8.8.8 through second link:
  ```
  sudo ip ro add 8.8.8.8/32 dev vpn0
  ```
- Check if not routing subsystem changed its routing table:
  ```
  ip ro get 8.8.8.8/32
  ```
- Run the traceroute commands again:
  `traceroute -I 8.8.8.8` or `mtr -r -c1 -n 8.8.8.8`
- Compare two received traces
- Using MTR option `-z` analyze the path across different autonomous systems

How it looks like on



Filtering example

Blocking outgoing connection

Example command how to block outcoming connection to any mail server:

```
iptables -t filter -A OUTPUT -o eth0 \
    -m tcp -p tcp --dport 25 \
    -j DROP
```

- `-t filter -A OUTPUT` - add to **OUTPUT** chain of the **filter** table
- `-m tcp -p tcp --dport 25`
  - `-m tcp` - activate iptables module for **TCP** connections
  - `-p tcp` - match only **TCP** connections
  - `--dport 25` - where **TCP** destination **port** is **25** (SMTP)
- For matched packets do the action `DROP`
  - Action `**DROP**` drops the packet with no actual notification of the remote side about that. So most probably sending side would try to resend packet one more time.

Blocking incoming connection

Rejecting with ICMP message

```
    iptables -t filter -A OUTPUT -o eth0 \
        -m tcp -p tcp --dport 25 \
        -j REJECT --reject-with icmp-host-unreachable
```

- `-t filter -A OUTPUT` - add to **OUTPUT** chain of the **filter** table
- `-m tcp -p tcp --dport 25`
  - `-m tcp` - activate iptables module for **TCP** connections
  - `-p tcp` - match only **TCP** connections
  - `--dport 25` - where **TCP** destination **port** is **25** (SMTP)
- For matched packets do the action `**REJECT**`
  - Action `**REJECT**` drops the packet, but comparing to action **DROP** behaves better.
    It sends back to sender an **ICMP** message that packets was rejected.

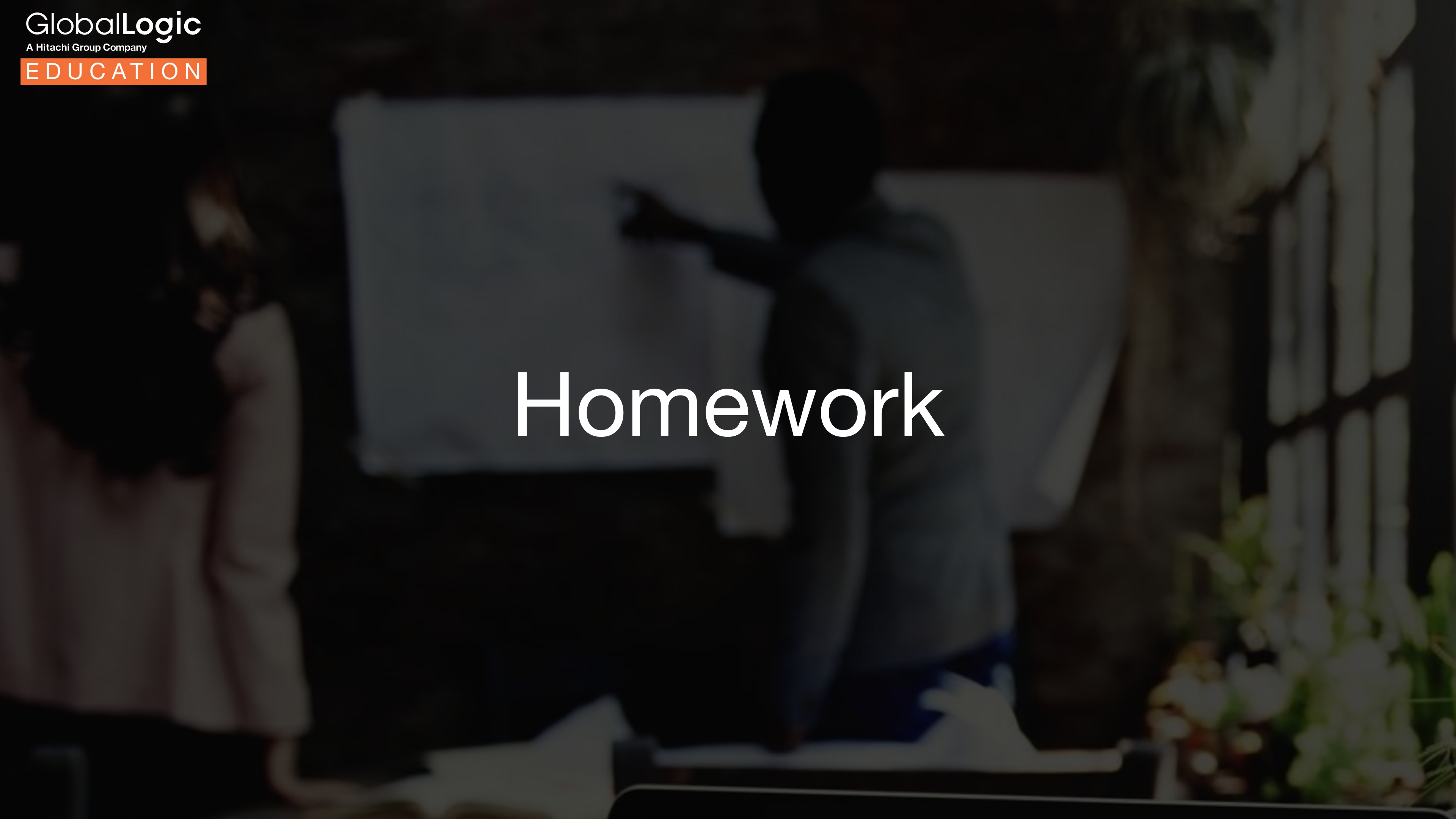
Redirecting connection

Redirecting incoming connection from the external host to us to a different port

```
iptables -t nat -A PREROUTING -i eth0 \
    -m tcp -p tcp --dport 25 \
    -j REDIRECT --to-port 25025
```

- `-t nat -A PREROUTING` - add to **PREROUTING** chain of the **nat** table
- `-m tcp -p tcp --dport 25`
  - `-m tcp` - activate iptables module for TCP connections
  - `-p tcp` - match only TCP connections
  - `--dport 25` - where TCP destination port is 25 (SMTP)
- For matched packets do the action `REDIRECT`
  - Action `REDIRECT` changes the TCP header. In our situation it changed TCP destination port, so that when the packet will reach routing phase, the new value would be considered.

Q&A

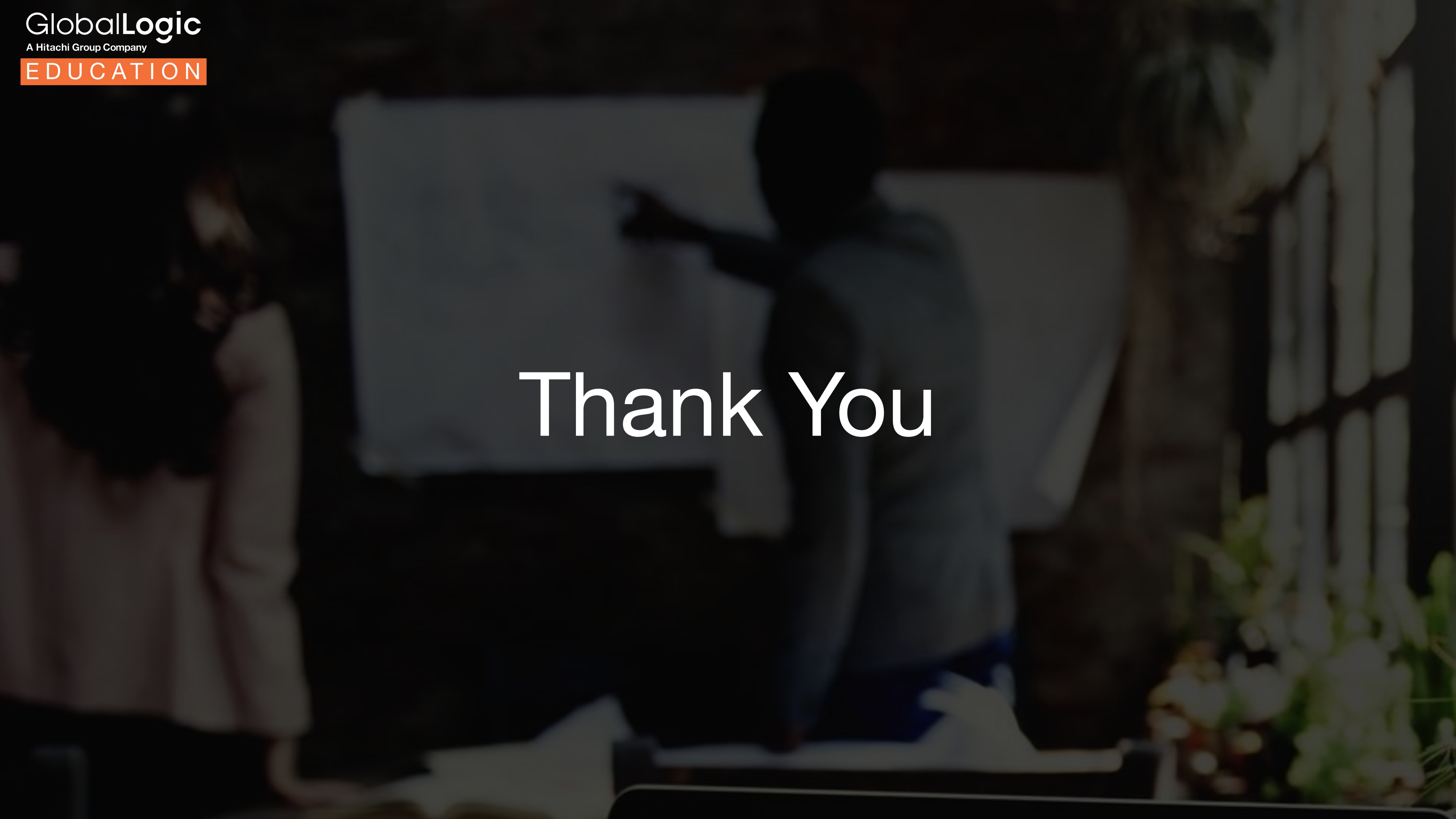


Homework

Homework

- Use netcat command start server which is listening on a TCP port 27664
    - Whatever netcat will receive as a server should be written/dumped into a file
- Using a telnet command connect to that TCP server (port 27664) and send some example message
- While doing so, capture all the traffic using wireshark
    - Investigate and apply necessary filter expression to filter
- Create a redirect rule which would redirect incoming traffic to port 21 to our server on a port 27664. Demonstrate how telnet successfully connecting to the port 21 and reaching our server instead
- For all of the tasks show all the commands and necessary screenshots to demonstrate how you've done that task
- Provide a traffic dump of communication between the TCP server (created using netcat) and telnet client. The dump shall be in pcap format.

Good luck!



Thank You

GlobalLogic

A Hitachi Group Company

EDUCATION