Network Security (NetSec)

Summer 2017
Exercise 04
Part 02: Reconnaissance

Dennis Giese| dennis.giese@seemoo.tu-darmstadt.de

Overview

- Introduction to Exercise 4: Part 2
- Foundations Nmap

Task

- Your Task
  - Collect as much as possible information about servers
    - 16 tokens are available
    - Version numbers
    - Used software/os
  - Where possible: leave messages

- Servers
  - 130.83.194.149, 130.83.194.150, 130.83.194.151
  - Available after 14.06.2016 18:59 until deadline
  - Generation of a „session id" is required
    - OTP-seed required for login

Session Generation & Login

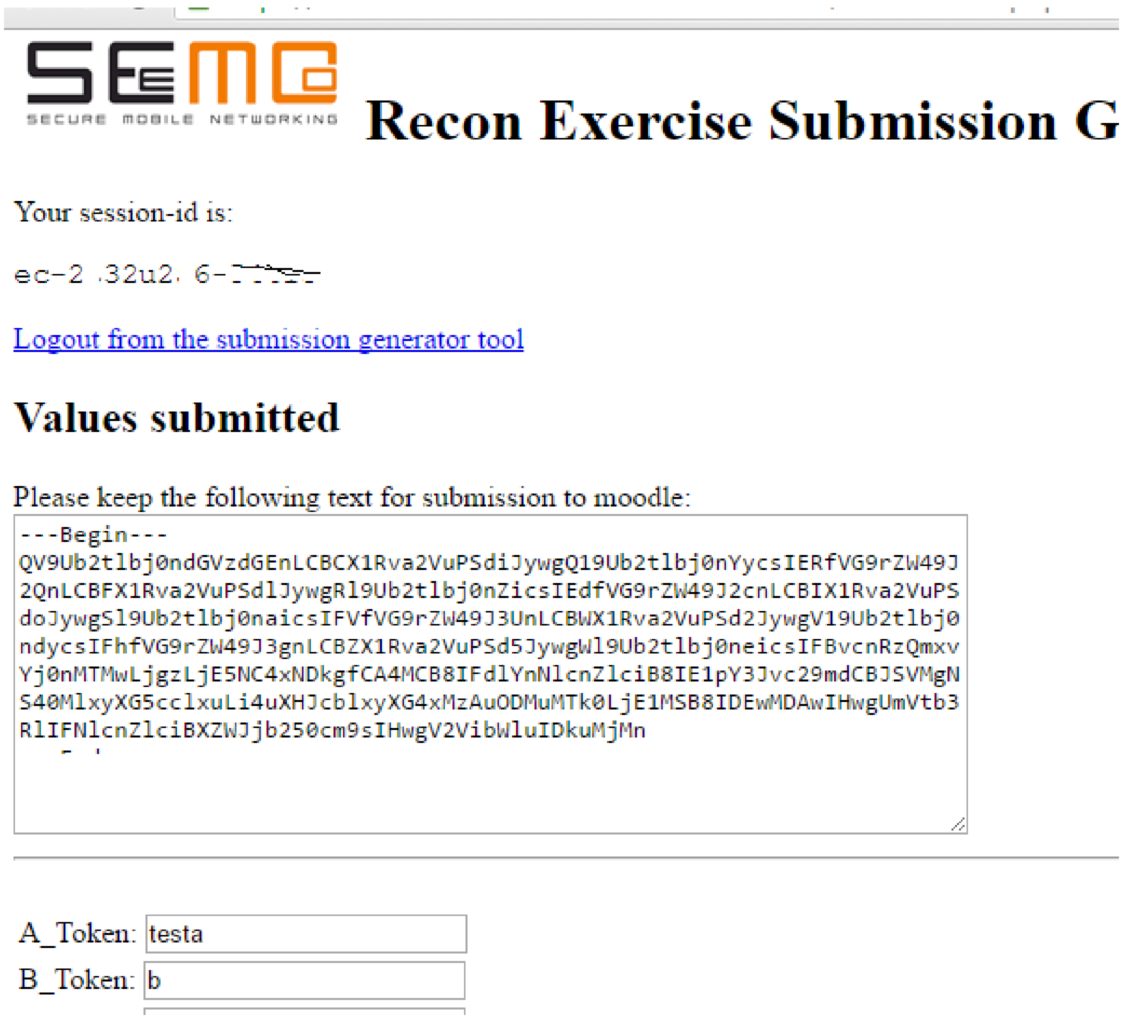
- Generation of the unique „session id"
  - xx-xxxxxxxx-xxxxx
  - One-time generation required
  - **without „session id": tokens are invalid**
  - Tokens and services bound to „session id"
    - Write the session id down, you can restore your session afterwards
  - Webinterface 130.83.194.149
  - Known to be vulnerable to the Heartbleed attack
  - Several times in an hour: an authorized users is logging in
    - Login page uses TOTP
- Firewall
  - Creates dynamically rules and bind services to session
    - -> IP address is very important
    - -> session valid for 2 hours, must be refreshed afterwards

Submission of solution & Teamwork

- Submission of solution
  - There is a specific form in moodle
    - Session id, tokens, ports, versions
    - Use specific submission generator tool
  - Additionally: an short report is required (attach to IDS-submission)

- Teamwork
  - This exercise is not a team exercise!
  - We expect you to work on the excercise yourself
    - do not do the exercise for someone else or exchange tokens
  - However: Exchanging your ideas and approaches is ok(e.g. how to use nmap, usage of metasploit, etc.)

Submission of solution & Teamwork



Recon Exercise Submission G

Your session-id is:

`ec-2 .32u2. 6-`[redacted]

[Logout from the submission generator tool]()

Values submitted

Please keep the following text for submission to moodle:

```
---Begin---
QV9Ub2tlbj0ndGVzdGEnLCBCX1Rva2VuPSdiJywgQ19Ub2tlbj0nYycsIERfVG9rZW49J
2QnLCBFX1Rva2VuPSdlJywgRl9Ub2tlbj0nZicsIEdfVG9rZW49J2cnLCBIX1Rva2VuPS
doJywgSl9Ub2tlbj0naicsIFVfVG9rZW49J3UnLCBWX1Rva2VuPSd2JywgV19Ub2tlbj0
ndycsIFhfVG9rZW49J3gnLCBZX1Rva2VuPSd5JywgWl9Ub2tlbj0neicsIFBvcnRzQmxv
Yj0nMTMwLjgzLjE5NC4xNDkgfCA4MCB8IFdlYnNlcnZlciB8IE1pY3Jvc29mdCBJSVMgN
S40MlxyXG5cclxuLi4uXHJcblxyXG4xMzAuODMuMTk0LjE1MSB8IDEwMDAwIHwgUmVtb3
RlIFNlcnZlciBXZWJjb250cm9sIHwgV2VibWluIDkuMjMn
```

A_Token:	testa
B_Token:	b

Technical Information (I)

- Servers reachable from inside the TU campus (e.g. Wifi)
  - However: Access from outside the campus might be possible, but it is not supported
  - Do not use the TU VPN (L3-VPN vs. Nmap)

- If you dare to do the exercise from outside
  - Do not use the TU VPN (again)
  - Beware of IPv6-to-IPv4 NATs (e.g. Unity Media)
  - Beware of IPv4-Carrier-grade NAT
    - Check if your router has a public IPv4 address
  - Results might be slighly different because of TU firewall

- Do not try to use different sessions on a shared computer
  - Always destroy the session/log out the submission generator tool

Technical Information (II)

- Do not attack infrastructure services/sites
  - All sites with SEEMOO-logo are not subject to attacks
  - I am a horribly bad programmer, please do not abuse bugs ;)
    - If you find any bugs: contact me (dennis.giese@seemoo.tu-darmstadt.de)

Nmap (1)

Extremely popular

- usually run over Linux
- rich feature set, exploiting raw sockets
- need root to use all features

Ping sweeping

- over any range of IP addresses
- with ICMP, SYN, ACK
- OS determination

Port scanning

- Over any range of ports
- Almost any type of TCP, UDP packet

Source IP address spoofing

- Decoy scanning

Packet fragmentation

Timing Options

Further information:
http://nmap.org/book/man.html

Nmap (2)

Input

```
nmap [Scan Type] [Options] <target hosts>
```

- Default for port scanning: ports 1-1024 plus ports listed in nmap service file

Output

- open ports: syn/ack returned; port is open
- unfiltered ports: RST returned: port is closed but not blocked by firewall
- filtered ports: nothing returned; port is blocked by firewall

See Appendix for further examples

TCP: Reset packet

If machine receives a TCP packet it is not expecting, it responds with TCP packet with RST bit set.

- For example when no process is listening on destination port

For UDP, machine returns ICMP “port unreachable” instead

Nmap (3): ping sweep

```
nmap -sP -v 116.27.38/24
```

Sends ICMP echo request (ping) to 256 addresses

Can change options so that pings with SYNs, ACKs…

- `-sP = ping`
- `-v = verbose`

Nmap (4): polite port scan

```
nmap -sT -v target.com
```

Attempts to complete 3-way handshake with each target port

Sends SYN, waits for SYNACK, sends ACK, then sends FIN to close connection

If target port is closed, no SYNACK returned

- Instead RST packet is typically returned

TCP connect scans are easy to detect

- Target (e.g. Web server) may log completed connections
- Gives away attacker's IP address

Nmap (5) : TCP SYN port scan

```
nmap -sS -v target.com
```

Stealthier than polite scan

Send SYN, receive SYNACK, send RST

- Send RST segment to avoid an accidental DoS attack

Stealthier: hosts do not record connection

- But routers with logging enabled will record the SYN packet

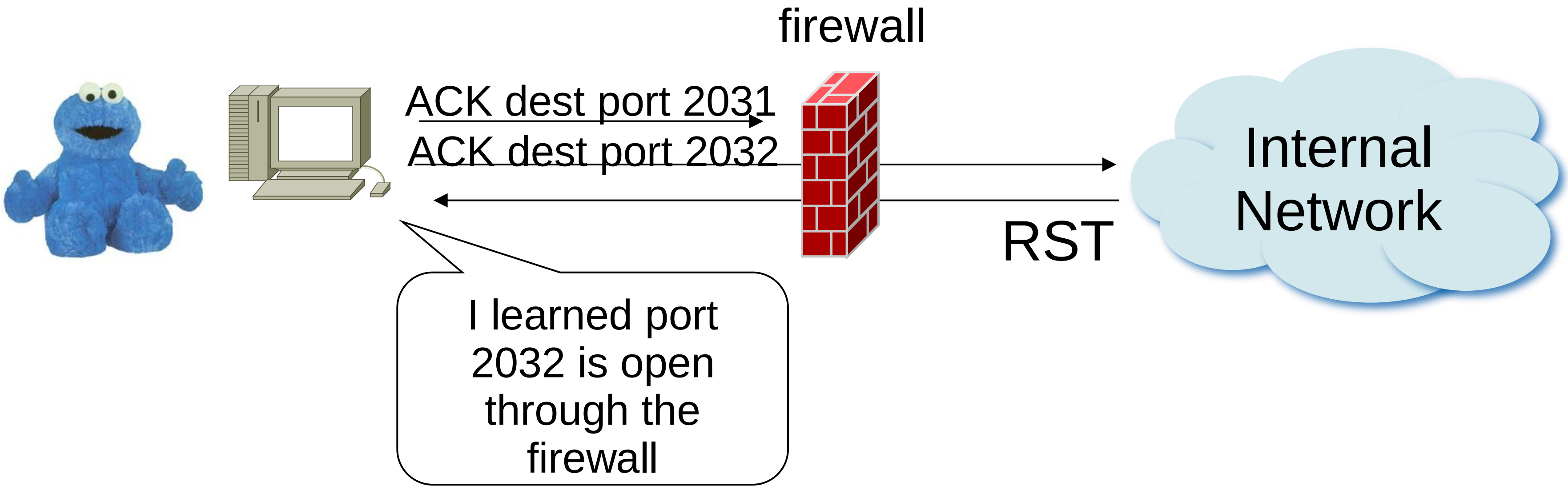
Faster: don’t need to send FIN packet

Nmap (6): TCP ACK scans

Many filters (in firewalls and routers) only let internal systems hosts initiate TCP connections

- Drop packets for which ACK=0 (ie SYN packet): no sessions initiated externally

To learn what ports are open through firewall, try an ACK scan (segments with ACK=1)



Nmap (7): UDP port scans

UDP doesn’t have SYN, ACK, RST packets

nmap simply sends UDP packet to target port

- ICMP Port Unreachable: interpret port closed
- Nothing comes back: interpret port open
  - False positives common

Nmap (8): Obscure Source

Attacker can enter list of decoy source IP addresses into Nmap

For each packet it sends, Nmap also sends packets from decoy source IP addresses

- For 4 decoy sources, send five packets

Attacker’s actual address must appear in at least one packet, to get a result

If there are 30 decoys, victim network will have to investigate 31 different sources!

Nmap (9): TCP Stack Fingerprinting

In addition to determining open ports, attacker wants to know OS on targeted machine:

- exploit machine's known vulnerabilities
- sophisticated hacker may set up lab environment similar to target network

TCP implementations in different OSes respond differently to (illegal) combinations of TCP flag bits

Nmap (10): Fingerprinting

Nmap sends

- SYN to open port
- NULL to open port (no flag bits set)
- SYN/FIN/URG/PSH to open port
- SYN to closed port
- ACK to closed port
- FIN/PSH/URG to closed port
- UDP to closed port

Nmap includes a database of OS fingerprints for hundreds of platforms

- See nmap.org for further details

Nmap (11): examples

```
nmap -v target.com
```

- Scans all TCP default ports on target.com; verbose mode

```
nmap -sS -O target.com/24
```

- First pings addresses in target network to find hosts that are up. Then scans default ports at these hosts; stealth mode (doesn't complete the connections); tries to determine OS running on each scanned host

```
nmap -sX -p 22,53,110,143 198.116.*.1-137
```

- Sends an Xmas tree scan to the first half of each of the 255 possible subnets in the 198.116/16. Testing whether the systems run ssh, DNS, pop3, or imap

```
nmap -v -p 80 *.*.2.3-5
```

- finds all web servers on machines with IP addresses ending in .2.3, .2.4, or .2.5

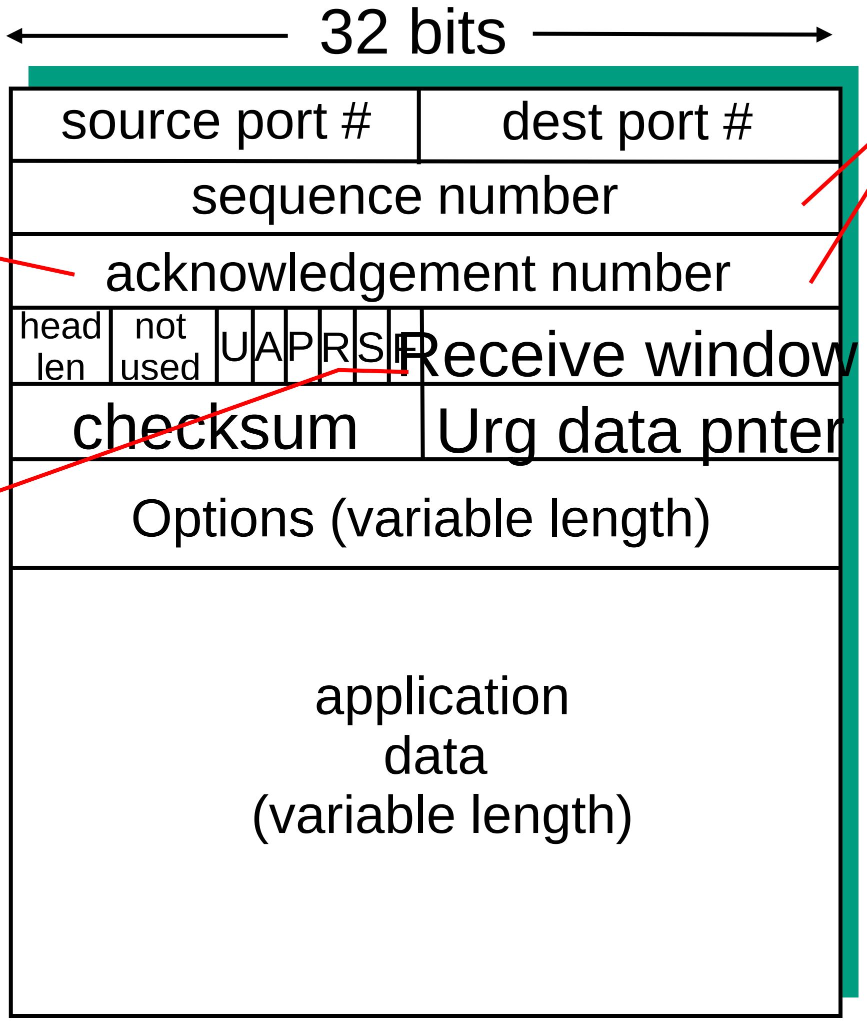
Contact

Prof. Dr.-Ing. Matthias Hollick
matthias.hollick@seemoo.tu-darmstadt.de

Technische Universität Darmstadt
Secure Mobile Networking Lab – SEEMOO
Department of Computer Science
Mornewegstr. 32
D-64293 Darmstadt

Phone: +49 6151 16-25472
Fax: +49 6151 16-25471
Web: https://seemoo.de

Excursus:
TCP Segment Structure



ACK # valid

counting by bytes of data (not segments!)

RST, SYN, FIN: connection estab (setup, teardown commands)

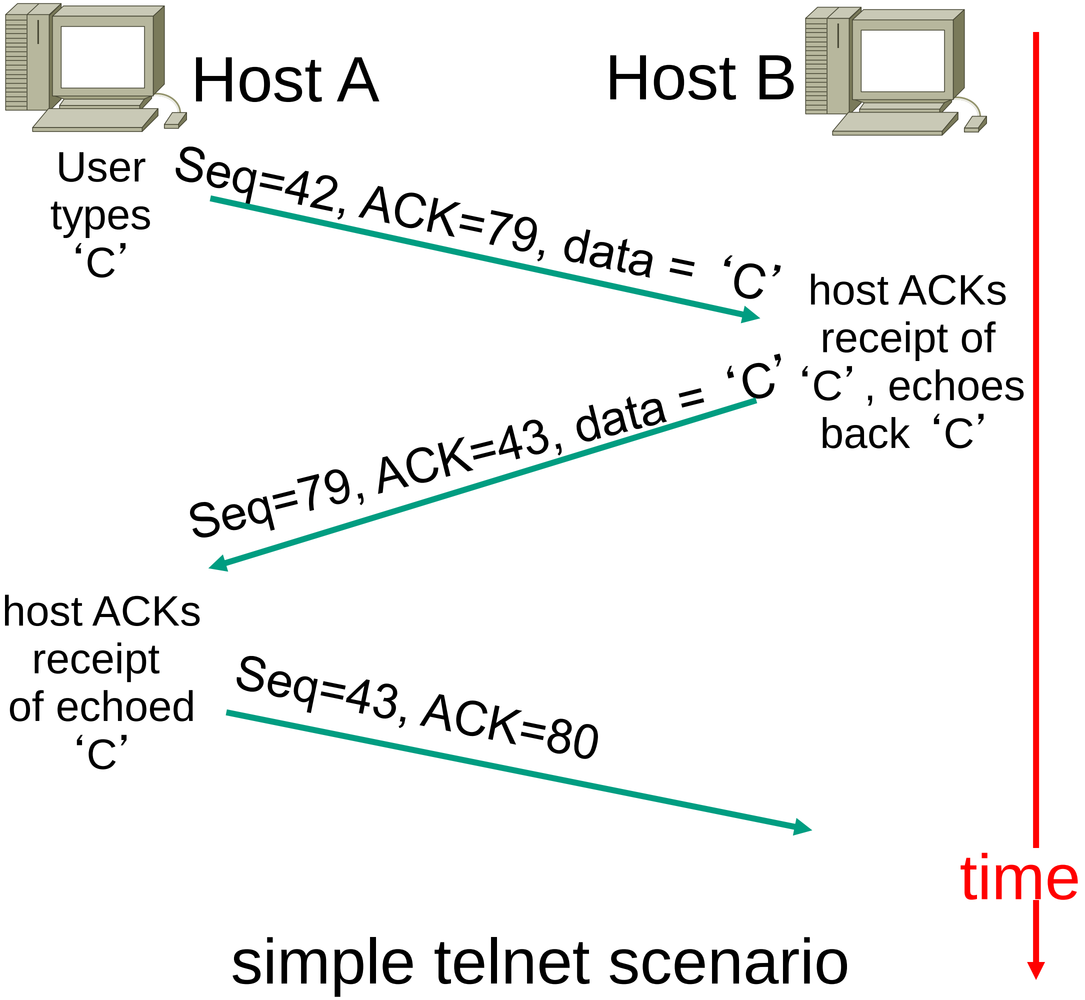
Excursus:
TCP seq. #’s and ACKs

Seq. #’s:

- byte stream “number” of first byte in segment’s data

ACKs:

- seq # of next byte expected from other side


Host A
Host B
User types ‘C’
Seq=42, ACK=79, data = ‘C’
host ACKs receipt of ‘C’, echoes back ‘C’
Seq=79, ACK=43, data = ‘C’
host ACKs receipt of echoed ‘C’
Seq=43, ACK=80
time

simple telnet scenario

Excursus: TCP Connection Establishment

Three way handshake:

Step 1: client host sends TCP SYN segment to server

- SYN=1, ACK=0
- specifies initial seq #
- no data

Step 2: server host receives SYN, replies with SYN-ACK segment

- SYN=1, ACK=1
- server host allocates buffers
- specifies server initial seq. #

Step 3: client receives SYN-ACK, replies with ACK segment, which may contain data

- SYN=0, ACK=1

Copyright Notice