ANSWERS

# A12.7 Internet security

Networks and the web · A level · OCR H446 1.3.3, AQA 7517 4.9.3.2 · about 30 min

## What this lesson is about

Firewalls, packet filtering, proxies and stateful inspection; encryption, certificates and signatures; worms, trojans and viruses.

- Firewalls
- Encryption
- Digital signatures
- Worms, trojans and viruses

## Questions

**6 marks in all**

**1.** Which kind of firewall allows an incoming packet only if it belongs to a connection that was properly started from inside? [1 mark]

- ☐ **A** Stateful inspection
- ☐ **B** Static packet filtering
- ☐ **C** A proxy server's cache
- ☐ **D** A MAC address allow list

> **Answer:** A. Stateful inspection keeps a table of open connections and judges each packet in that context.

**2.** What does a proxy server do that a packet filter does not? [1 mark]

- ☐ **A** Makes requests on behalf of internal clients, hiding their addresses, and can inspect content and cache pages
- ☐ **B** Checks only the source and destination ports
- ☐ **C** Encrypts all traffic with the receiver's public key
- ☐ **D** Assigns IP addresses to clients

> **Answer:** A. The outside server only sees the proxy; handling whole requests lets it filter content and cache responses.

**3.** Ada wants to send Ben a message only Ben can read. Which key should she encrypt it with? [1 mark]

- ☐ **A** Ben's public key
- ☐ **B** Ben's private key
- ☐ **C** Ada's private key
- ☐ **D** Ada's public key

> **Answer:** A. Only Ben's private key can decrypt something encrypted with Ben's public key.

**4.** Put the steps of creating and checking a digital signature in order. [1 mark]

Number the lines 1 to 5 to put them in the right order.

___ `The receiver decrypts the signature with the sender's public key`
___ `The receiver compares the two digests`
___ `The receiver hashes the received message`
___ `The sender hashes the message to make a digest`
___ `The sender encrypts the digest with their private key`

**Answer:**

```
The sender hashes the message to make a digest
The sender encrypts the digest with their private key
The receiver decrypts the signature with the sender's public key
The receiver hashes the received message
The receiver compares the two digests
```

Matching digests show the message came from the holder of the private key and was not changed.

**5.** Which kind of malware spreads through a network by itself, exploiting vulnerabilities, with no host file or user action? [1 mark]

- ☐ **A** Worm
- ☐ **B** Virus
- ☐ **C** Trojan
- ☐ **D** Proxy

**Answer:** A. A virus needs a host file and a user to spread it; a trojan relies on the user installing it and does not replicate.

**6.** What is the purpose of a digital certificate? [1 mark]

- ☐ **A** To show, with a certificate authority's signature, that a public key really belongs to a named owner
- ☐ **B** To store a user's private key on a website
- ☐ **C** To speed up symmetric encryption
- ☐ **D** To list the ports a firewall blocks

**Answer:** A. The CA signs the owner's identity and public key, so a browser that trusts the CA can trust the key.

## The task: a stateful firewall

Each packet is a tuple `(direction, source IP, source port, destination IP, destination port)`, where direction is `"out"` (leaving the network) or `"in"` (arriving). Check each packet against these rules in order, and use the first that matches: 1. If its source IP is `203.0.113.9` (a blocked address), the verdict is `drop blocked`. 2. If it is going `out`, record the connection (its source IP, source port, destination IP and destination port) and the verdict is `allow outbound`. 3. If it is coming `in` and is a reply to a recorded connection (its source is that connection's destination, and its destination is that connection's source, IP and port both), the verdict is `allow reply`. 4. If it is coming `in` to IP `192.168.4.10` port `443` (the school's web server), the verdict is `allow service`. 5. Otherwise the verdict is `drop`. Print `<packet number> <verdict>` for each packet, numbering from 1. Then print `allowed <a>, dropped <d>`, counting verdicts that begin `allow` and `drop`. That is 8 lines. The robot does not drive.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

BLOCKED = "203.0.113.9"
SERVER = ("192.168.4.10", 443)
packets = [
    ("out", "192.168.4.23", 49152, "203.0.113.80", 443),
    ("in", "203.0.113.80", 443, "192.168.4.23", 49152),
    ("in", "203.0.113.80", 443, "192.168.4.23", 49153),
    ("in", "203.0.113.9", 443, "192.168.4.23", 49152),
    ("in", "198.51.100.7", 51000, "192.168.4.10", 443),
    ("in", "198.51.100.7", 51000, "192.168.4.10", 22),
    ("in", "203.0.113.80", 443, "192.168.4.23", 49152),
]
connections = []
```

**The hint students can ask for:** Check the rules in the order they are listed and stop at the first that matches. An outgoing packet is remembered as a connection; an incoming packet is a reply only if its addresses and ports are that connection's with source and destination swapped.

### A solution

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

BLOCKED = "203.0.113.9"
SERVER = ("192.168.4.10", 443)
packets = [
    ("out", "192.168.4.23", 49152, "203.0.113.80", 443),
    ("in", "203.0.113.80", 443, "192.168.4.23", 49152),
    ("in", "203.0.113.80", 443, "192.168.4.23", 49153),
    ("in", "203.0.113.9", 443, "192.168.4.23", 49152),
    ("in", "198.51.100.7", 51000, "192.168.4.10", 443),
    ("in", "198.51.100.7", 51000, "192.168.4.10", 22),
    ("in", "203.0.113.80", 443, "192.168.4.23", 49152),
]
connections = []
allowed = 0
dropped = 0
for n, (direction, src, sport, dst, dport) in enumerate(packets, start=1):
    if src == BLOCKED:
```

```
        verdict = "drop blocked"
    elif direction == "out":
        connections.append((src, sport, dst, dport))
        verdict = "allow outbound"
    elif (dst, dport, src, sport) in connections:
        verdict = "allow reply"
    elif (dst, dport) == SERVER:
        verdict = "allow service"
    else:
        verdict = "drop"
    if verdict.startswith("allow"):
        allowed = allowed + 1
    else:
        dropped = dropped + 1
    print(n, verdict)
print(f"allowed {allowed}, dropped {dropped}")
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.