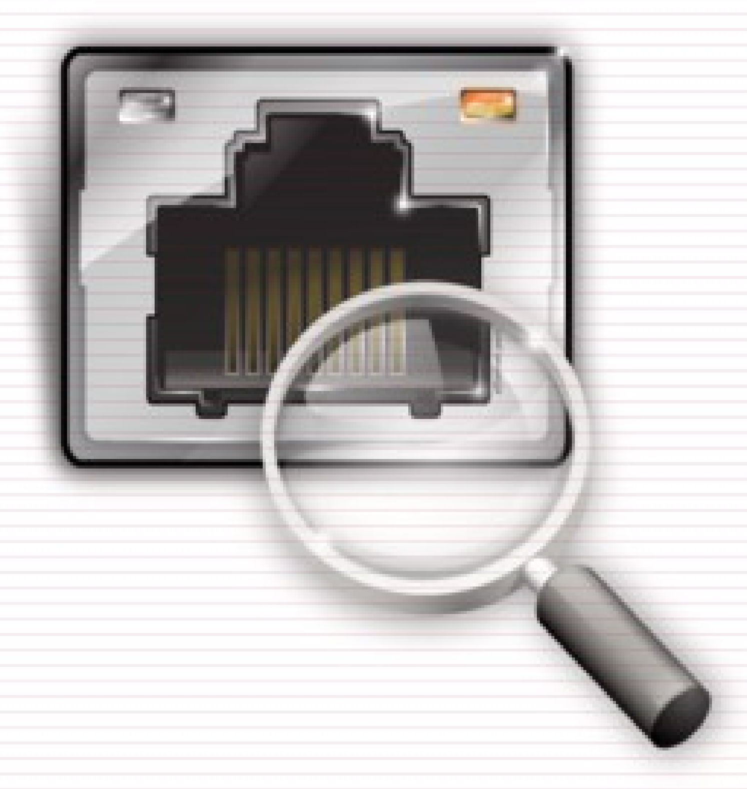
Portscanner



Applied Information Security

12.01.2016

Simon Targa

Mirko Bez

Aim of the project

- Implement a C port scanner
  - Various scan methods e.g. TCP connect scan, TCP SYN scan …
  - Console program like nmap

- Implement a C port scan detector
  - Console program

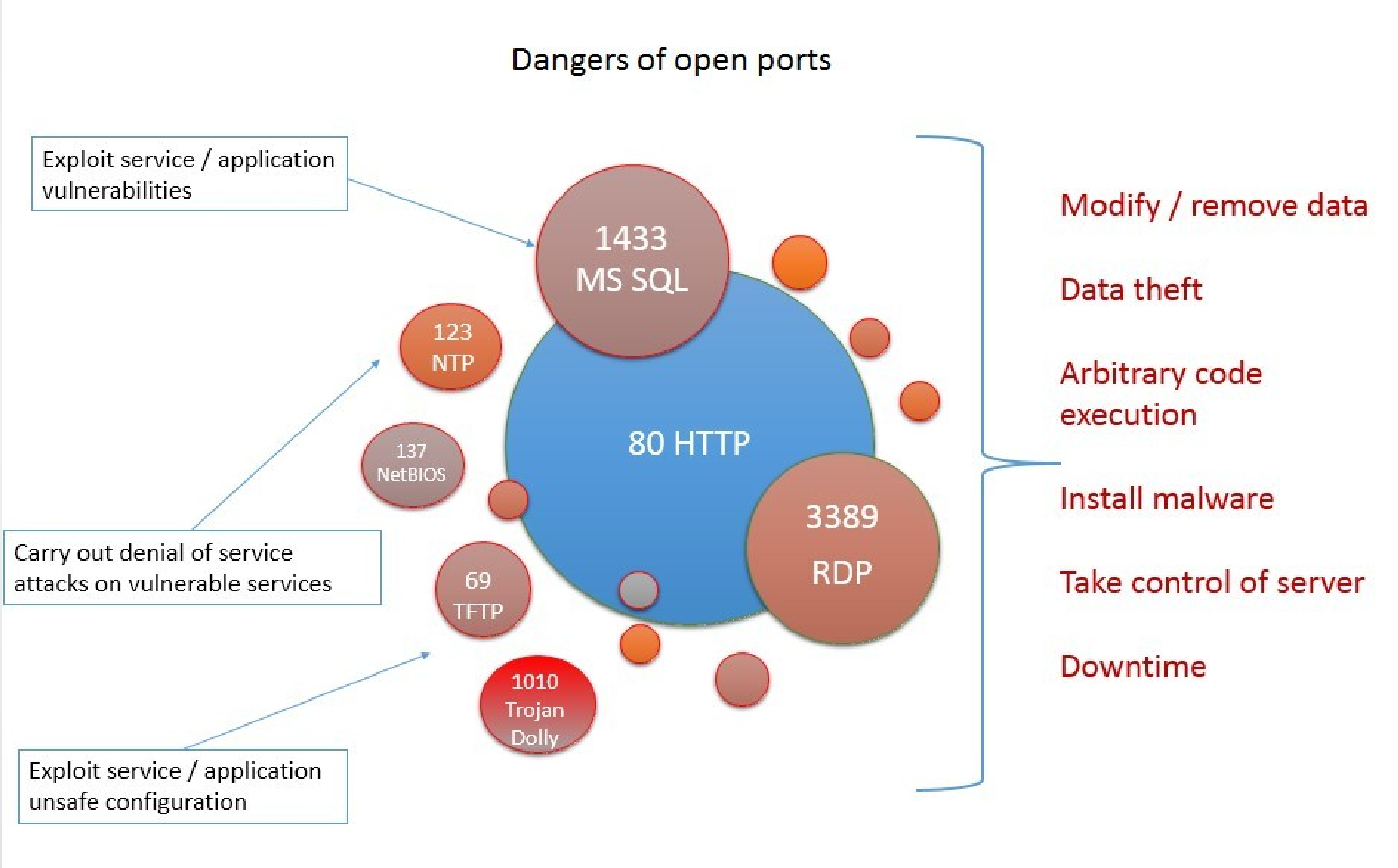
Motivation

- Which ports are open?

- Admins scan system/network:
  - Chek security of a network/system
  - Check intrusion attempts

- Attackers scan victims:
  - Identify services that are running on a system
  - Check vulnerabilities of systems
  - Exploit vulnerabilities in services running on open ports

Danger of open ports



Dangers of open ports

Exploit service / application vulnerabilities

Carry out denial of service attacks on vulnerable services

Exploit service / application unsafe configuration

Modify / remove data

Data theft

Arbitrary code execution

Install malware

Take control of server

Downtime

Implemented Scan Methods

- TCP connect scan
- TCP SYN scan
- NULL scan
- XMAS scan
- FIN scan
- Maimon scan

TCP connect scan

- Simple approach
- Use system call connect() to try to establish a connection
- If connection successful port is open
- Else port is closed

Advantages/Disadvantages

- Advantages:

  + Accurate

  + Fast

  + No root privileges required

  + Easy implementation

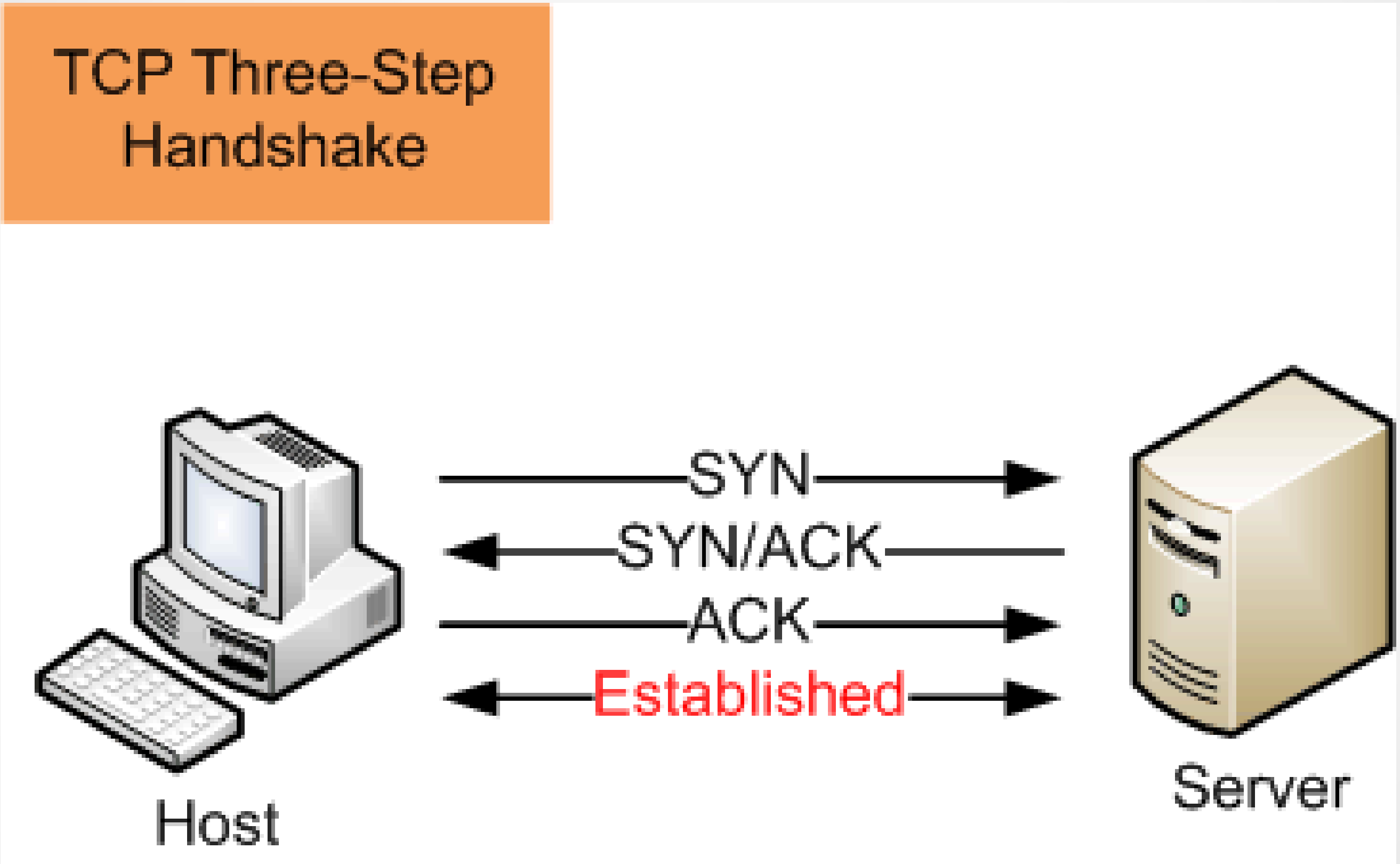
- Disadvantages

  - Easy detectable/logged

Implementation

- SOCK_STREAM socket (TCP socket)
- Use connect() function to try to establish a connection with the target machine
- Check return value of connect() function
  - 0 -> then port is open
  - Otherwise -> port is close

TCP Handshake



Syn scan

- Half-open scanning →

  doesn't open full TCP connection
- Send SYN packet
- SYN/ACK response = port is listening
- RST = port is close
- TIMEOUT(no answer) = port is filtered
- Standard scan method of nmap

Advantages/Disadvantages

- Advantages:

  + Reliable

  + Fast

  + No TCP three-way handshake → No logs

- Disadvantages

  - Requires root privileges

Raw socket

- Allows direct sending and receiving of Internet Protocol packets without any protocol-specific transport layer formatting
- Programmer builds packet to send
- Full control over headers (IP and TCP)
- Allows sophisticated scan techniques

Implementation

- Use RAW socket and set IP and TCP header
- Only set SYN Flag in TCP Header
- Send the packet to target machine
- Receive packets from target machine and check flags:
  - SYN/ACK flag set = port is open
  - RST flag = port is closed
  - TIMEOUT (no response) = port is filtered

XMAS, FIN, NULL scan

- Exploits a subtle loophole in the TCP RFC to differentiate between open and closed port
- RFC 793 says:
  - “if the [destination] port state is CLOSED .... an incoming segment not containing a RST causes a RST to be sent in response”
  - If packets are sent to open ports without the SYN, RST, or ACK bits set it states: “you are unlikely to get here, but if you do, drop the segment, and return.”

Advantages/Disadvantages

- Advantages:

  + More stealthy than SYN scan

  + No TCP three-way handshake → No logs

- Disadvantages

  - Requires root privileges

  - Slow false positives

Flags

- NULL scan: Does not set any bits (TCP flag header is 0)
- FIN scan: Sets just the TCP FIN bit
- XMAS scan: Sets the FIN, PSH, and URG flags, lighting the packet up like a Christmas tree
- Maimon scan: Sets FIN and ACK flags

Implementation

- Use RAW socket and set IP and TCP header
- Set Flags depending on method in TCP Header
- Send the packet to target machine
- Try to get a response:
  - RST packet received = port is closed
  - No response = Port is open/filtered

Port Scan Detector

- Aim: detect port scan of the different types
- Main Idea:
  You are (potentially) port scanned if in a short interval you receive a lot of UDP or TCP request from the same IP-Address

Port Scan Detector Functionality

- Sophos Approach (Implemented)
  - “A port scan is detected when a detection score of 21 points in a time range of 300ms for one individual source IP-Address is exceeded”

Source https://www.sophos.com/it-it/support/knowledgebase/115153.aspx

- Scanlogd Approach
  - At least 7 different privileged or 21 non-privileged ports, or a weighted combination of those, have to be accessed with no longer than 3 seconds between the accesses to be treated as a scan.
  - If more than 5 scans are detected within 20 seconds, that event will be logged and logging will be stopped temporarily.

Source http://www.openwall.com/scanlogd/scanlogd.8.shtml

Point Score

- Detection Score can be calculated as follows:

- Scan of a TCP destination port < 1024: 3 points
- Scan of a TCP destination port >= 1024: 1 point
- Scan of ports 11, 12, 13, 2000: 10 points

Source https://www.sophos.com/en-us/support/knowledgebase/115153.aspx

Implementation Notes

- PCAP library used
  - Used by Wireshark and TCPDump to get packets!
  - Easy API
  - Possibility to choose which packets to get:

    e.g. “dst host 192.168.0.1 && dst portrange 1-1024”

- For each potential attacker (i.e. different IP-Source) a thread is started
- This thread checks each 300 ms if a port scan is detected

Little Demonstration

Thank you for your attention!

Summary

- Port Scan are used to determine which ports are open
  - Open ports can be used for different attacks
  - To defend from port scans → use a Port Scan Detector and close as many ports as possible!
- We implemented
  - A TCP Port Scanner with 6 scan methods available
  - A TCP/UDP Port Scan Detector