

# Network security

## Exercise 10

### Network monitoring

Tobias Limmer

Computer Networks and Communication Systems  
Dept. of Computer Sciences, University of Erlangen-Nuremberg, Germany

22. – 25.01.2008

# Network monitoring

dividable in:

- passive monitoring: monitor does not send anything to network → undetectable!
  - e.g. tcpdump, ethereal/wireshark, ...
- active monitoring: monitor sends “requests” to hosts and analyses responses
  - e.g. nmap, xprobe2, ...

# Active monitoring

- Nmap: scans for hosts and/or open ports, OS fingerprinting
- Xprobe2: sophisticated OS fingerprinting
- Nessus: vulnerability scanner, contains lots of exploits/attacks on systems and tries them out on targets
- lots of other vulnerability scanners . . .

# Passive monitoring

- raw packets:
  - Tcpdump: recording and visualization/interpretation of raw packets, storage
  - Ethereal/Wireshark: visual GUI, compatible with Tcpdump
- flows:
  - NProbe: generation of Netflow v9 data
  - Vermont: generation of IPFIX data (among other things)
- Intrusion detection systems (IDS)
  - dividable in anomaly-based and signature-based systems (see lecture)
  - Snort: payload-based IDS with lots of support and signatures
  - Bro: successor of Snort with better technology but less support by community

# IDS Snort

- parses payload of packets according to rules:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"WEB-CGI SWSOFT ASPSeek Overflow attempt";
flow:to_server,established; uricontent:"/s.cgi"; nocase;
content:"tmpl="; metadata:service http;
reference:bugtraq,2492; reference:cve,2001-0476;
classtype:web-application-attack; sid:804; rev:10;)
```

- preprocessing:
  - IP defragmentation
  - TCP reassembly
  - portscan detection
  - protocol analysis: SMTP, HTTP, FTP, SSH, ...

# IDS Snort 2

- output modules:
  - syslog, database, tcpdump, prelude, idmef, ...
- alert example:

```
[**] [122:1:0] (portscan) TCP Portscan [**]  
01/22-00:33:22.216582 192.168.101.1 -> 192.168.101.2  
PROTO255 TTL:0 TOS:0x0 ID:0 IpLen:20 DgmLen:164 DF
```

# flow aggregation

- data is summarized to relevant information
- standard flow keys are usually 5 tuple:  
protocol, srcIP, dstIP, srcPort, dstPort
- additional aggregated fields are e.g. number of packets, bytes, start-/endtime of flow
- flow keys may be varied arbitrarily
- available protocols:
  - Netflow v5/v9: introduced by Cisco
  - IPFIX (Internet Protocol Flow Information Export):  
standardized by IETF

Prot.	srcIP	srcPort	dstIP	dstPort	Oct.
TCP	10.0.0.2	50000	10.0.0.1	80	10
TCP	10.0.0.5	50012	10.1.23.2	80	42
TCP	10.0.0.2	50000	10.0.0.1	80	52
TCP	10.0.0.7	50023	10.2.38.1	80	67
TCP	10.0.0.2	50000	10.0.0.1	80	12

aggregated flow key

Prot.	srcIP	srcPort	dstIP	dstPort	Oct.
TCP	10.0.0.5	50012	10.1.23.2	80	42
TCP	10.0.0.7	50023	10.2.38.1	80	67
TCP	10.0.0.2	50000	10.0.0.1	80	74

## flow aggregation 2

- incoming packets are stored in hashtable
- if flow keys of incoming packet are identical to stored flow, flows are aggregated
- timeout of flows: when are flows taken from hashtable?
  - passive timeout: if flow is inactive, take this timeout
  - active timeout: if still packets for flow arrive, export this flow after this time nevertheless
- IPFIX:
  - templates: regularly sent packets which describe structure of IPFIX data records
  - data records: contain one or more flows
  - usually sent via UDP, SCTP is recommended
- biflows: aggregation of flows which represent both directions of a connection

# Vermont

