

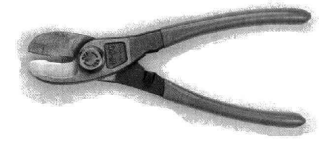
Velkommen til

Introduktion til Firewalls

Juli 2010

Flemming Jacobsen
fj@batmule.dk
Henrik Lund Kramshøj
hlk@kramse.org

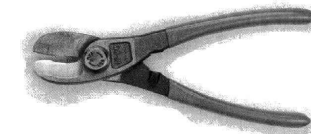
Formål



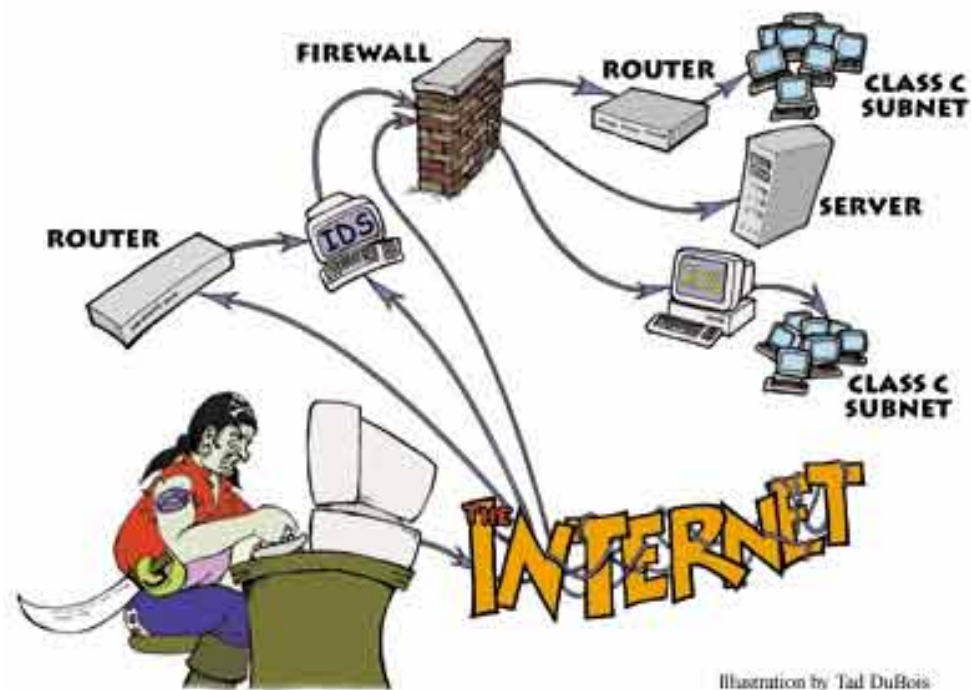
At introducere Firewall begrebet

At vise de sikkerhedsmæssige aspekter af firewalls og netsikkerhed

At give jer information om et par konkrete firewall eksempler



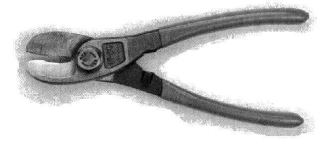
Er Firewalls og netsikkerhed interessant?



Sikkerhedsproblemerne i netværk er mange

Mange services - mange sårbare services

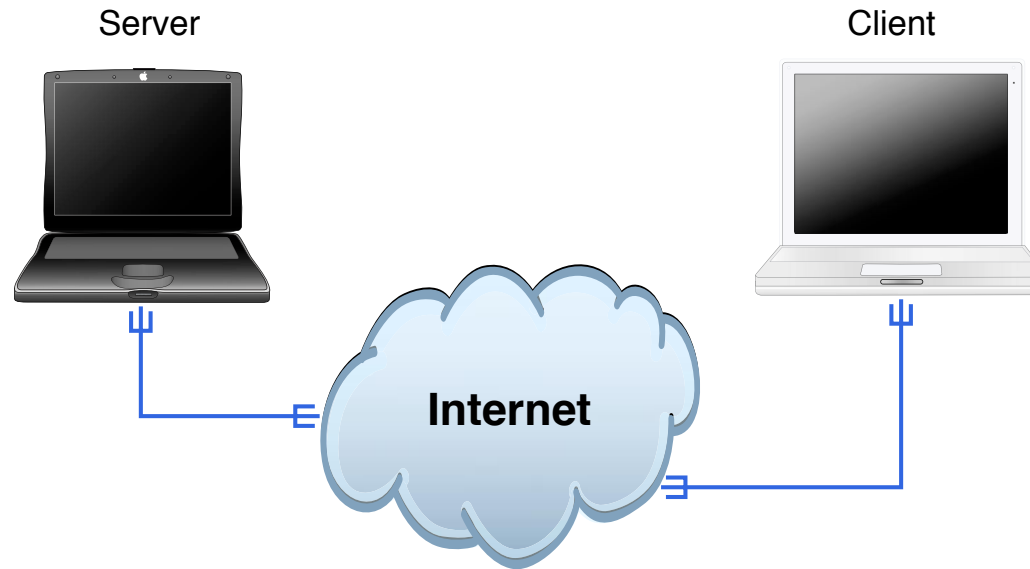
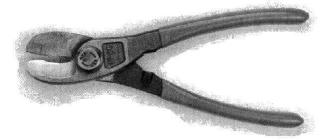
Hackerværktøjer



Der benyttes og henvises til en del værktøjer:

- nmap - <http://www.insecure.org> portscanner
- Wireshark - <http://www.wireshark.org> avanceret netværkssniffer
- OpenBSD - <http://www.openbsd.org> operativsystem med fokus på sikkerhed
- FreeBSD - <http://www.freebsd.org> operativsystem med fokus
- Juniper SSG og SRX
- Cisco Access Control Lists (ACL) IOS og ASA

Internet idag

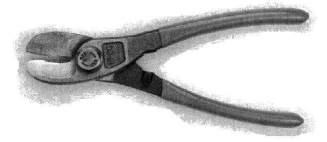


Klienter og servere

Rødder i akademiske miljøer

Protokoller der er op til 20 år gamle

Meget lidt kryptering, mest på http til brug ved e-handel



The Internet Worm 2. nov 1988

Udnyttede følgende sårbarheder

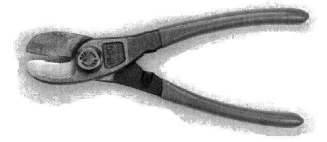
- buffer overflow i fingerd - VAX kode
- Sendmail - DEBUG
- Tillid mellem systemer: rsh, rexec, ...
- dårlige passwords

Avanceret + camouflage!

- Programnavnet sat til 'sh'
- Brugte fork() til at skifte PID jævnligt
- password cracking med intern liste med 432 ord og /usr/dict/words
- Fandt systemer i /etc/hosts.equiv, .rhosts, .forward, netstat ...

Lavet af Robert T. Morris, Jr.

Medførte dannelsen af CERT, <http://www.cert.org>

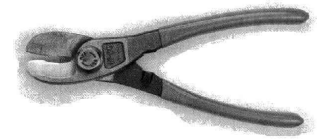


Hvad er en firewall

En firewall er noget som **blokerer** trafik på Internet

■ En firewall er noget som **tillader** trafik på Internet

Firewall

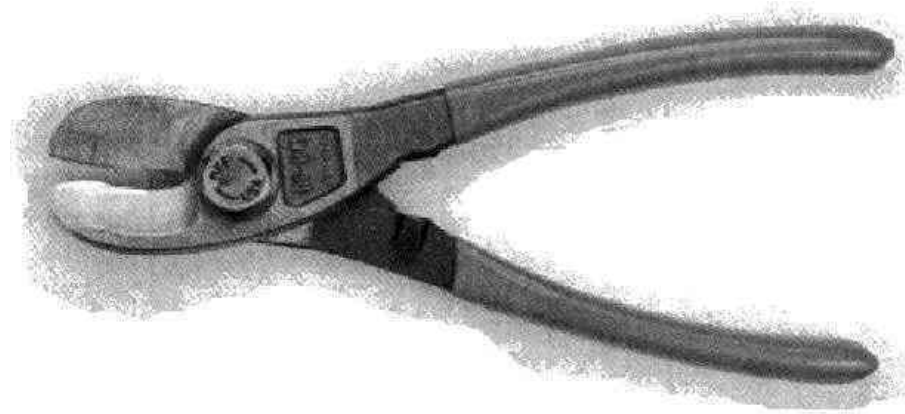
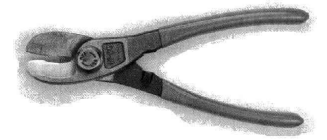


Indeholder typisk:

- Grafisk brugergrænseflade til konfiguration
- TCP/IP filtermuligheder - pakkernes afsender, modtager, retning ind/ud, porte, protokol, ...
- både IPv4 og IPv6 for Open Source firewalls: IPFW, OpenBSD PF, Linux firewalls, ...
- kun IPv4 for de kommercielle firewalls, enkelte kan IPv6
- foruddefinerede regler/eksempler
- typisk NAT funktionalitet indbygget
- typisk mulighed for nogle serverfunktioner:
DHCP-server, DNS caching server og lignende

En router med Access Control Lists - kaldes ofte netværksfilter, mens en dedikeret maskine kaldes en firewall

100% sikkerhed?



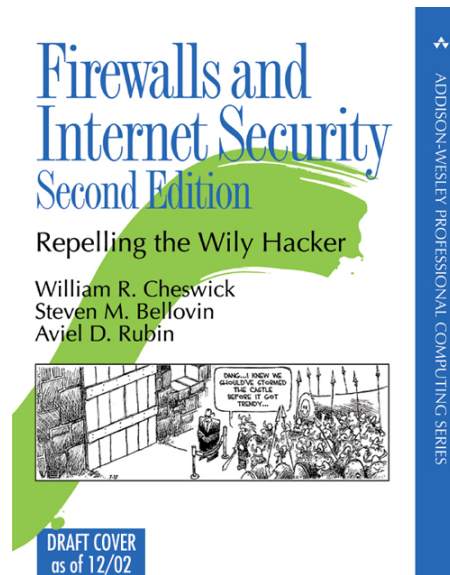
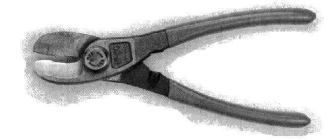
Hvor skal en firewall placeres for at gøre størst nytte?

Hvad er forudsætningen for at en firewall virker?
At der er konfigureret et sæt fornuftige regler!

Hvor kommer reglerne fra? Sikkerhedspolitikken!

Kilde: Billedet er fra Marcus Ranum The ULTIMATELY Secure Firewall

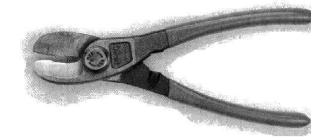
Firewall historik



Firewalls har været kendt siden starten af 90'erne

Den første bog *Firewalls and Internet Security* udkom i 1994 men der findes mange akademiske artikler om firewalls

Bogen *Firewalls and Internet Security* anbefales, William R. Cheswick, Steven M. Bellovin, Aviel D. Rubin, Addison-Wesley, 2nd edition, 2003



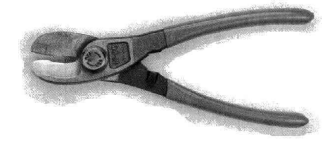
Moderne firewalls

Kommercielle firewalls:

- Cisco IOS routere og ASA firewalls <http://www.cisco.com>
- Juniper SSG og SRX <http://www.juniper.net>

Open source:

- Linux firewalls - fra begyndelsen til det nuværende netfilter til kerner version 2.4 og 2.6
<http://www.netfilter.org>
- Firewall GUIs ovenpå Linux - mange - findes også som kommercielle produkter
- IP Filter (IPF) <http://coombs.anu.edu.au/~avalon/>
- OpenBSD PF - findes idag på andre operativsystemer <http://www.openbsd.org>
- FreeBSD IPFW <http://www.freebsd.org> - IPFW benyttes også på Mac OS X. FreeBSD inkluderer ligeledes PF

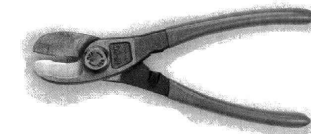


Personlige firewalls

Personlige firewalls kendes fra Windows (windows gennemgås ikke)

De firewalls vi omtaler kan bruges til både servere og klienter

Det anbefales at bruge en firewall på din laptop



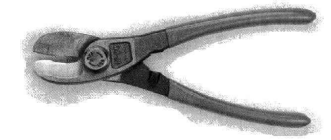
OSI og Internet modellerne

OSI Reference Model

Application
Presentation
Session
Transport
Network
Link
Physical

Internet protocol suite

Applications HTTP, SMTP, FTP, SNMP,	NFS
	XDR
	RPC
TCP UDP	
IPv4	IPv6 ICMPv6 ICMP
ARP RARP	
MAC	
Ethernet token-ring ATM ...	



Stateless vs Stateful

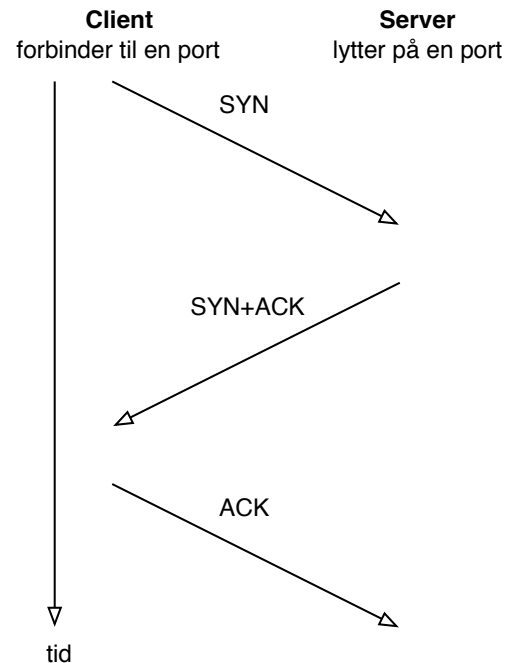
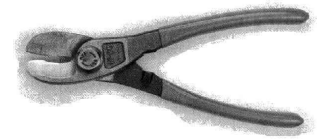
Stateless har ingen hukommelse, ser pakker enkeltvis

Stateful husker hvilke forbindelser der er etableret og tillader svar til disse
Denne type kræver dog mere processorkraft og hukommelse

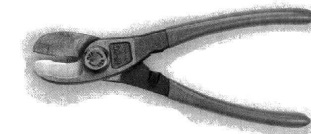
NB: stateful filtering bruges til TCP, UDP, ICMP osv. ikke *kun* til TCP

På de fleste netværk, servere og klienter bør der udelukkendes bruges stateful filtrering

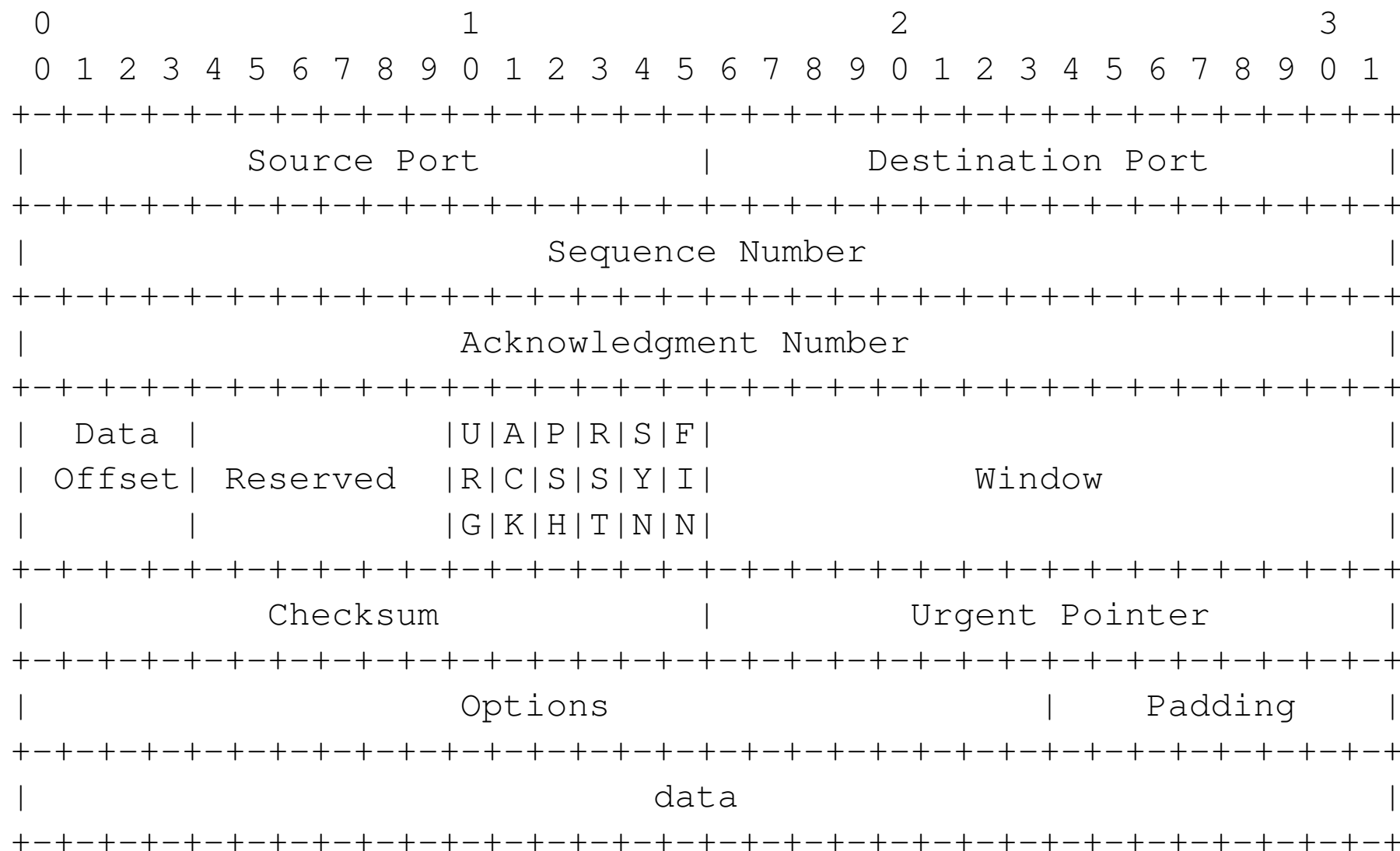
TCP three way handshake



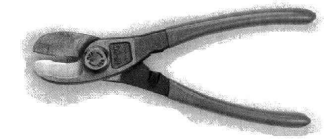
- **TCP SYN half-open scans**
- Tidligere loggede systemer kun når der var etableret en fuld TCP forbindelse - dette kan/kunne udnyttes til *stealth*-scans
- Hvis en maskine modtager mange SYN pakker kan dette fylde tabellen over connections op - og derved afholde nye forbindelser fra at blive oprette - **SYN-flooding**



TCP Transmission Control Protocol



TCP Header Format



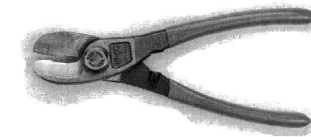
First or Last

To typer af firewalls:

First match - eksempelvis IPFW

Last match - eksempelvis PF

Det er ekstremt vigtigt at vide hvilken type firewall man bruger!

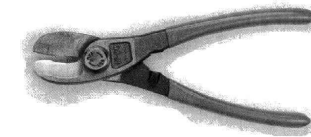


First match - IPFW

```
00100 16389 1551541 allow ip from any to any via lo0
00200      0          0 deny log ip from any to 127.0.0.0/8
00300      0          0 check-state
...
```

```
65435      36      5697 deny log ip from any to any
65535     865     54964 allow ip from any to any
```

Den sidste regel nås aldrig!



Last match - OpenBSD PF

```
ext_if="ext0"
```

```
int_if="int0"
```

```
block in
```

```
pass out keep state
```

```
pass quick on lo $int_if
```

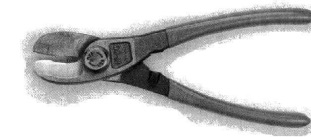
```
# Tillad forbindelser ind på port 80=http og port 53=domain
```

```
# på IP-adressen for eksterne netkort ($ext_if) syntaksen
```

```
pass in on $ext_if proto tcp to ($ext_if) port http keep state
```

```
pass in on $ext_if proto tcp, udp to ($ext_if) port domain keep st
```

Pakkerne markeres med `block` eller `pass` indtil sidste regel
nøgleordet *quick* afslutter match - god til store regelsæt



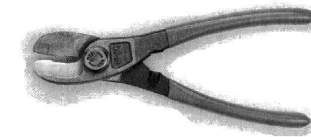
Vores firewall giver optimal sikkerhed

Vi har en server og har sat den bagved firewalls!

Den er sikker!



Desværre ikke!



Firewall står ikke alene

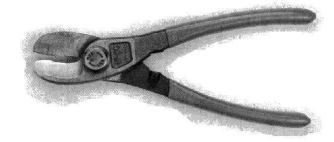
Firewall alene er ikke nok

Eksempelvis kan den ikke inspicere alle emails og websider du besøger

Sørg for at opdatere dit system jævnligt og husk backup

På Windows vil anti-virus være nødvendigt sammen med firewall

Forsvaret er som altid - flere lag af sikkerhed!



Firewall konfiguration

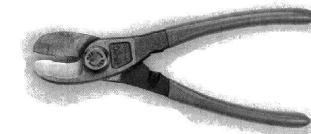
Den bedste firewall konfiguration starter med:

- Papir og blyant
- En fornuftig adressestruktur

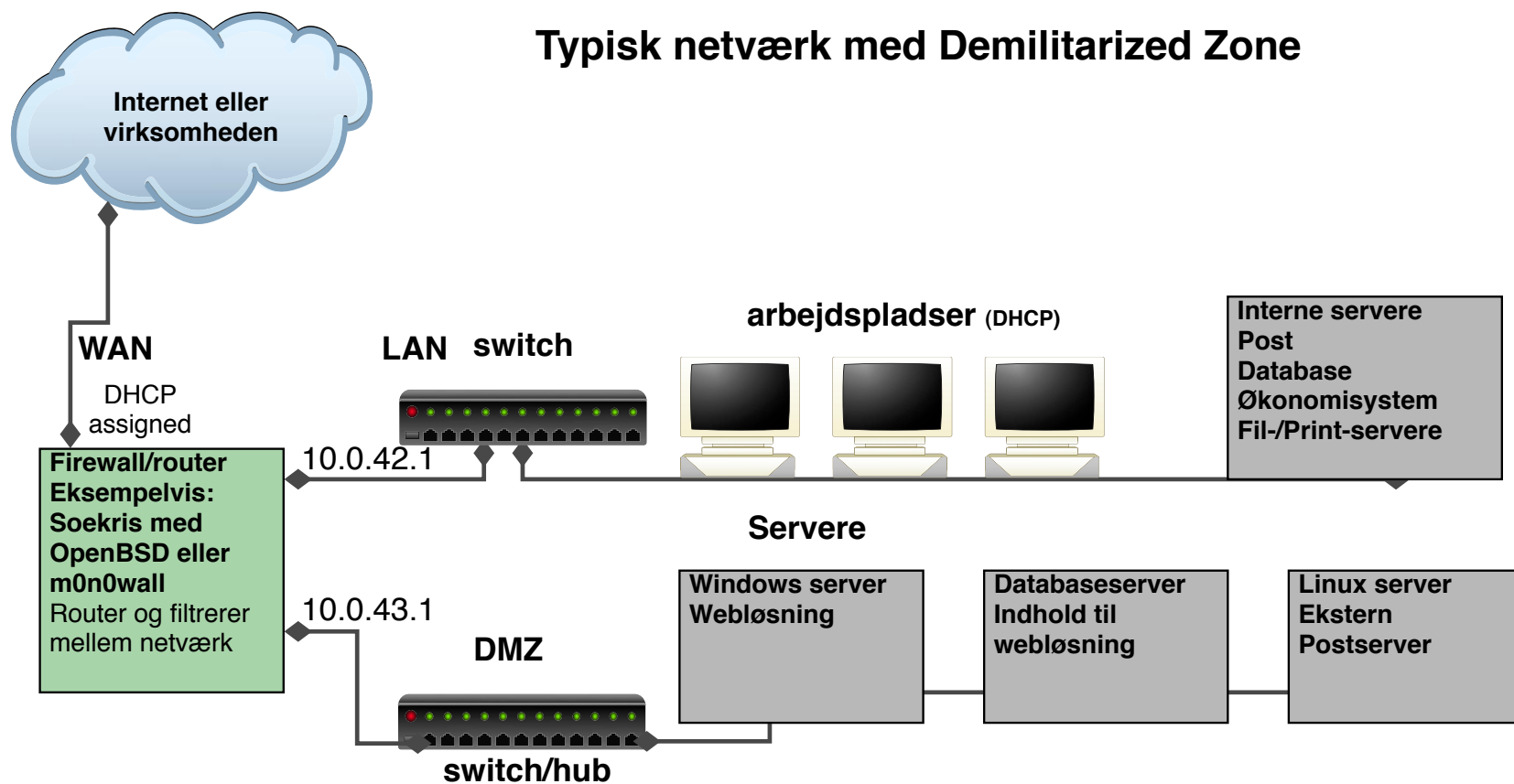
Brug dernæst en firewall med GUI første gang!

Husk dernæst:

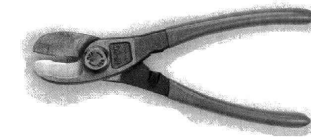
- En firewall skal passes
- En firewall skal opdateres
- Systemerne bagved skal hærdes!



En typisk firewall konfiguration



Opdeling i separate netværkssegmenter!



Bloker indefra og ud

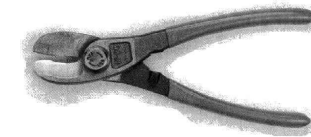
Der er porte og services som altid bør blokeres

Det kan være kendte sårbare services

- Windows SMB filesharing - ikke til brug på Internet!
- UNIX NFS - ikke til brug på Internet!

Kendte problemer:

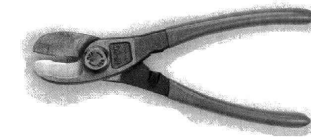
- KaZaA og andre P2P programmer - hvis muligt!
- Portmapper - port 111



Sådan beskytter du et netværk

Antagelser:

- Du har et hjemmenetværk
- Du har en dedikeret firewall
- Du har en Webserver og en emailserver på din DMZ



Regler

Netværkssegmenter:

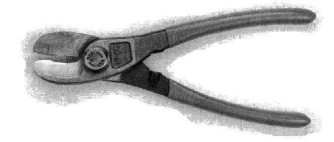
```
192.168.1.0/24 LAN  
192.168.10.0/24 DMZ
```

Netværkskort:

```
vr0 10.1.2.3 outside/internet  
vr1 192.168.1.254 LAN  
vr2 192.168.10.254 DMZ  
(antager man bruger en Soekris 5501)
```

Services:

```
SMTP port 25/tcp  
HTTP port 80/tcp  
HTTPS port 443/tcp
```



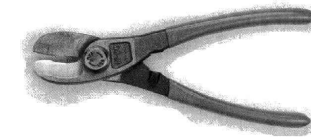
Definitioner og makroer

```
ext_if="vr0"  
int_if="vr1"  
dmz_if="vr2"
```

```
webserver="192.168.10.2"  
mailserver="192.168.10.3"
```

```
table <homenet> { 192.168.1.0/24 }  
table <dmznet> { 192.168.10.0/24 }
```

```
table <webserver> { 192.168.10.2 }  
table <mailserver> { 192.168.10.3 }
```

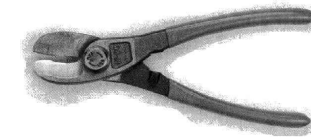


NAT og basics

```
nat on $ext_if from !($ext_if) -> ($ext_if:0)
rdr on $ext_if proto tcp from any to ($ext_if) port 25 -> $mailserver port 25
rdr on $ext_if proto tcp from any to ($ext_if) port 80 -> $webserver port 80
rdr on $ext_if proto tcp from any to ($ext_if) port 443 -> $webserver port 443
```

```
block in
pass out
```

```
# no internal filtering done, currently
pass quick on lo
pass in quick on $int_if
```

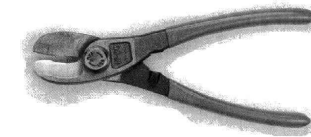


DMZ services

```
# incoming
pass in on $ext_if proto tcp to <mailserver> port 25
pass in on $ext_if proto tcp to <webserver> port 80
pass in on $ext_if proto tcp to <webserver> port 443

# outgoing
pass in on $dmz_if proto tcp from <mailserver> to any port 25
pass in on $dmz_if proto tcp udp from <mailserver> to any port 53
pass in on $dmz_if proto tcp from <mailserver> to any port 123

# This example is not complete!
```

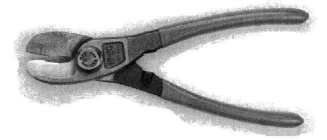


Cisco ASA definitioner

```
names
name 10.1.2.3 externalip
name 192.168.10.2 webserver
name 192.168.10.3 mailserver

interface GigabitEthernet0/0
  nameif outside
  security-level 0
  ip address 10.1.2.3 255.255.255.0
!
interface GigabitEthernet0/1
  nameif dmz
  security-level 10
  ip address 192.168.10.254 255.255.255.0
!
interface GigabitEthernet0/2
  nameif inside
  security-level 100
  ip address 192.168.1.254 255.255.255.0
```

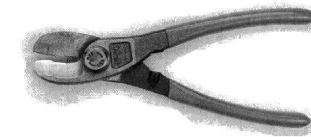
Cisco ASA



```
access-list outside_access extended permit tcp any host mailserver eq smtp
access-list outside_access extended permit tcp any host webserver eq http
access-list outside_access extended permit tcp any host webserver eq https
```

```
access-group outside_access in interface outside
access-group dmz_access in interface dmz
access-group inside_access in interface inside
```

! NB: NAT and inside+dmz rules not shown

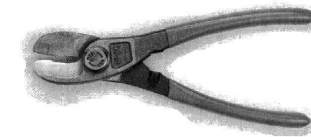


Cisco IOS ACL

```
access-list 101 permit tcp any any established
access-list 101 permit icmp any any
access-list 101 permit tcp any host 192.168.90.3 eq 22
access-list 101 permit tcp any host 192.168.90.3 eq www
```

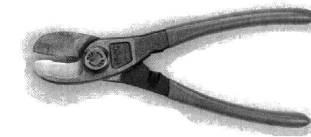
Added som en 'in' regel på det yderste IF. :

```
interface GigabitEthernet0
  description $FW_OUTSIDE$$ES_WAN$
  ip address dhcp client-id GigabitEthernet0
  ip access-group 101 in
  no ip redirects
  no ip unreachable
  no ip proxy-arp
  ip flow ingress
```



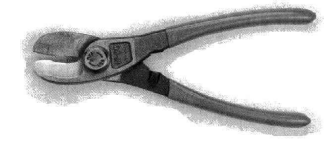
Juniper SRX definitioner

```
security {
  zones {
    security-zone trust {
      tcp-rst;
      address-book {
        address local-net 10.0.100.0/24;
        address dktest 10.0.100.101/32;
        address notest 10.0.100.102/32
        ...
      }
    }
  }
}
security-zone untrust {
  address-book {
    address notest 193.88.xxx.26/32;
    address vpnno 193.88.xxx.6/32;
    address vpndk 193.88.xxx.7/32;
  }
}
```



Juniper SRX Basic services

```
screen untrust-screen;
host-inbound-traffic {
    system-services {
        ssh;
        ping;
        http;
        https;
        ike;
    }
}
```



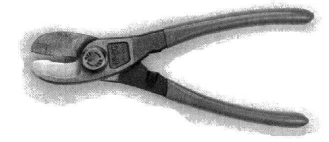
Juniper SRX policy

```

policies {
  from-zone trust to-zone trust {
    policy default-permit {
      match {
        source-address any;
        destination-address any;
        application any;
      }
      then {
        permit;
      }
    }
  }
}

```

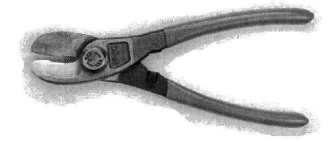
Note: husk at trust til trust zone også skal tillades



Juniper SRX policy

```
from-zone trust to-zone untrust {  
  policy any-permit {  
    match {  
      source-address any;  
      destination-address any;  
      application any;  
    }  
    then {  
      permit;  
    }  
  }  
}
```

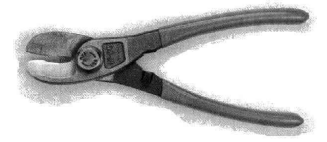
...



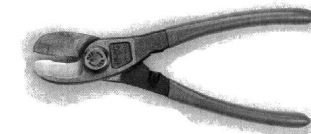
Juniper SRX policy - untrust to trust

```
from-zone untrust to-zone trust {
  policy notest {
    match {
      source-address any;
      destination-address notest;
      application [ junos-http junos-https myssh ];
    }
    then {
      permit;
    }
  }
  policy dktest {
    match {
      source-address any;
      destination-address dktest;
      application [ junos-http junos-https myssh ];
    }
    then {
      permit;
    }
  }
}
```

IPFW



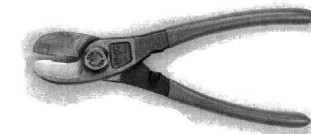
<http://www.batmule.dk/TheCamp/FW/fw.ipfw>



Hvordan virker DNS - query

```
# dig www.freebsd.org
```

```
IP 192.168.18.135.53419 > 192.5.6.30.53 53986% [1au] A? ns1.isc-sns.net. (44)
IP 192.168.18.135.61860 > 192.42.93.30.53 56088% [1au] A? ns2.isc-sns.com. (44)
IP 192.168.18.135.53453 > 199.249.113.1.53 18401% [1au] A? ns3.isc-sns.info. (45)
IP 199.249.113.1.53 > 192.168.18.135.53453 18401- 0/3/3 (161)
IP 192.168.18.135.63533 > 63.243.194.1.53 6080% [1au] A? ns3.isc-sns.info. (45)
IP 63.243.194.1.53 > 192.168.18.135.63533 6080*- 2/3/7 A 63.243.194.1, (1036)
IP 192.5.6.30.53 > 192.168.18.135.53419 53986- 0/3/4 (177)
IP 192.168.18.135.49650 > 63.243.194.1.53 5937 [1au] A? www.freebsd.org. (44)
IP 192.168.18.135.55042 > 63.243.194.1.53 27651% [1au] A? ns1.isc-sns.net. (44)
IP 192.42.93.30.53 > 192.168.18.135.61860 56088- 0/3/4 (177)
IP 192.168.18.135.54655 > 38.103.2.1.53 33354% [1au] A? ns2.isc-sns.com. (44)
IP 63.243.194.1.53 > 192.168.18.135.49650 5937*- 1/3/11 A 69.147.83.33 (1109)
IP 63.243.194.1.53 > 192.168.18.135.55042 27651*- 2/3/5 A 72.52.71.1, (831)
IP 38.103.2.1.53 > 192.168.18.135.54655 33354*- 2/4/9 A 38.103.2.1, (1249)
IP 192.168.18.135.51539 > 38.103.2.1.53 15185 [1au] MX? www.freebsd.org. (44)
IP 38.103.2.1.53 > 192.168.18.135.51539 15185*- 1/3/11 MX . 0 (1108)
```



Hvordan virker DNS - response

;; QUESTION SECTION:

;www.freebsd.org. IN A

;; ANSWER SECTION:

www.freebsd.org. 3455 IN A 69.147.83.33

;; AUTHORITY SECTION:

freebsd.org. 3455 IN NS ns2.isc-sns.com.

freebsd.org. 3455 IN NS ns1.isc-sns.net.

freebsd.org. 3455 IN NS ns3.isc-sns.info.

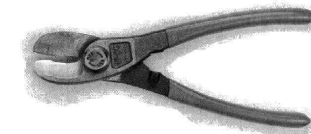
;; ADDITIONAL SECTION:

ns1.isc-sns.net. 3455 IN A 72.52.71.1

ns2.isc-sns.com. 3455 IN A 38.103.2.1

ns3.isc-sns.info. 3455 IN A 63.243.194.1

<http://www.batmule.dk/TheCamp/FW/DNS>

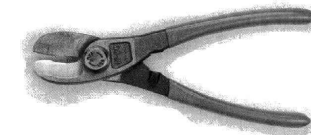


Hvordan virker TCP/HTTP

```
# fetch http://www.batmule.dk/test.txt
```

```
23:36:39.363598 IP 192.168.18.135.17934 > 195.215.30.71.80      Flags [S], seq 1025202606, win 65535,
options [mss 1460,nop,wscale 3,sackOK,TS
val 291138673 ecr 0],length 0
23:36:39.397177 IP 195.215.30.71.80      > 192.168.18.135.17934  Flags [S.], seq 805263203, ack 1025202607,
win 65535, options [mss 1460,sackOK,eol],
length 0
23:36:39.397290 IP 192.168.18.135.17934 > 195.215.30.71.80      Flags [.], ack 1, win 65535, length 0
23:36:39.398321 IP 192.168.18.135.17934 > 195.215.30.71.80      Flags [P.], ack 1, win 65535, length 99
23:36:39.413495 IP 195.215.30.71.80      > 192.168.18.135.17934  Flags [P.], ack 100, win 65535, length 262
23:36:39.413564 IP 195.215.30.71.80      > 192.168.18.135.17934  Flags [F.], seq 263, ack 100, win 65535,
length 0
23:36:39.413619 IP 192.168.18.135.17934 > 195.215.30.71.80      Flags [.], ack 264, win 65438, length 0
23:36:39.421152 IP 192.168.18.135.17934 > 195.215.30.71.80      Flags [F.], seq 100, ack 264, win 65535,
length 0
23:36:39.432576 IP 195.215.30.71.80      > 192.168.18.135.17934  Flags [.], ack 101, win 65534, length 0
```

```
http://www.batmule.dk/TheCamp/FW/HTTP
```



TCP/HTTP response

```
hlk@bigfoot:hlk$ telnet www.batmule.dk 80
```

```
Trying 195.215.30.71...
```

```
Connected to fj.batmule.dk.
```

```
Escape character is '^]'.  
GET /test.txt HTTP/1.1
```

```
Host: www.batmule.dk
```

```
HTTP/1.1 200 OK
```

```
Date: Wed, 28 Jul 2010 11:03:02 GMT
```

```
Server: Apache
```

```
Last-Modified: Tue, 27 Jul 2010 21:35:03 GMT
```

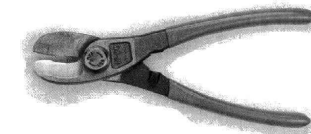
```
ETag: "b28e9-1e-4c4f5107"
```

```
Accept-Ranges: bytes
```

```
Content-Length: 30
```

```
Content-Type: text/plain
```

```
Tue Jul 27 23:35:03 CEST 2010
```



TCPDUMP - protokolanalyse pakkesniffer

The screenshot shows a web browser window with the address bar containing `http://www.tcpdump.org/tcpdump_man.html`. The browser's address bar also shows a search bar with the text "Google". The browser's address bar also shows a search bar with the text "Google". The browser's address bar also shows a search bar with the text "Google".

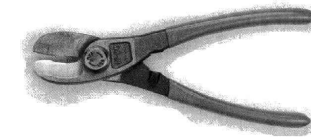
SYNOPSIS

```
tcpdump [ -adeflnNOpqRStuvxX ] [ -c count ]  
[ -C file_size ] [ -F file ]  
[ -i interface ] [ -m module ] [ -r file ]  
[ -s snaplen ] [ -T type ] [ -w file ]  
[ -E algo:secret ] [ expression ]
```

DESCRIPTION

Tcpdump prints out the headers of packets on a network interface that match the boolean *expression*. It can also be run with the *-w* flag, which causes it to save the packet data to a file for later analysis, and/or with the *-b* flag, which causes it to read from a saved packet file rather than to read packets from a network interface. In all cases, only packets that match *expression* will be processed by *tcpdump*.

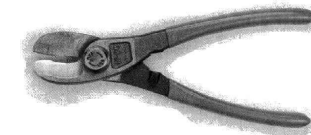
<http://www.tcpdump.org> - både til Windows og UNIX



tcpdump - normal brug

- tekstmode
- kan gemme netværkspakker i filer
- kan læse netværkspakker fra filer
- er de-facto standarden for at gemme netværksdata i filer

```
[root@otto hlk]# tcpdump -i en0
tcpdump: listening on en0
13:29:39.947037 fe80::210:a7ff:fe0b:8a5c > ff02::1: icmp6: router advertisement
13:29:40.442920 10.0.0.200.49165 > dns1.cybercity.dk.domain: 1189+[|domain]
13:29:40.487150 dns1.cybercity.dk.domain > 10.0.0.200.49165: 1189 NXDomain* [|domain]
13:29:40.514494 10.0.0.200.49165 > dns1.cybercity.dk.domain: 24765+[|domain]
13:29:40.563788 dns1.cybercity.dk.domain > 10.0.0.200.49165: 24765 NXDomain* [|domain]
13:29:40.602892 10.0.0.200.49165 > dns1.cybercity.dk.domain: 36485+[|domain]
13:29:40.648288 dns1.cybercity.dk.domain > 10.0.0.200.49165: 36485 NXDomain* [|domain]
13:29:40.650596 10.0.0.200.49165 > dns1.cybercity.dk.domain: 4101+[|domain]
13:29:40.694868 dns1.cybercity.dk.domain > 10.0.0.200.49165: 4101 NXDomain* [|domain]
13:29:40.805160 10.0.0.200 > mail: icmp: echo request
13:29:40.805670 mail > 10.0.0.200: icmp: echo reply
...
```



TCPDUMP syntaks - udtryk

filtre til husbehov

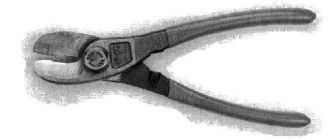
- type - host, net og port
- src pakker med afsender IP eller afsender port
- dst pakker med modtager IP eller modtager port
- host - afsender eller modtager
- proto - protokol: ether, fddi, tr, ip, ip6, arp, rarp, decnet, tcp og udp

IP adresser kan angives som dotted-decimal eller navne

porte kan angives med numre eller navne

komplekse udtryk opbygges med logisk and, or, not

Vildt eksempel http://wiki.tyk.nu/index.php/Tcpdump_patterns



tcpdump udtryk eksempler

Host 10.1.2.3

Alle pakker hvor afsender eller modtager er 10.1.2.3

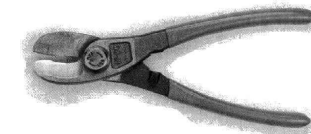
host 10.2.3.4 and not host 10.3.4.5

Alle pakker til/fra 10.2.3.4 undtagen dem til/fra 10.3.4.5

- meget praktisk hvis man er logget ind på 10.2.3.4 via netværk fra 10.3.4.5

host foo and not port ftp and not port ftp-data

trafik til/fra maskine *foo* undtagen hvis det er FTP trafik



Wireshark - grafisk pakkesniffer

WIRESHARK

HOME ABOUT WHAT'S NEW DOWNLOAD FAQ

Get It

[Download](#)

Get Help

[FAQs](#)

[Documentation](#)

[Mailing Lists](#)

[Wiki](#)

[Bug tracker](#)

Develop

[Developer Info](#)

Products

[AirPcap](#)

[Network Toolkit](#)

[OEM WinPcap](#)

Sniffing Problems A Mile Away

The Ethereal network protocol analyzer has changed its name to Wireshark.

The name might be new, but the software is the same. Wireshark's powerful features make it the tool of choice for network troubleshooting, protocol development, and education worldwide.

Wireshark was written by networking experts around the world, and is an example of the power of open source. It runs on Windows, Linux, UNIX, and other platforms.

Download Now

0.99.3

Q:

How do I capture 802.11 traffic on Windows?

A:

AirPcap

News

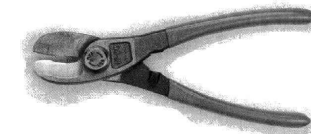
Wireshark 0.99.3 Released

Aug 23, 2006

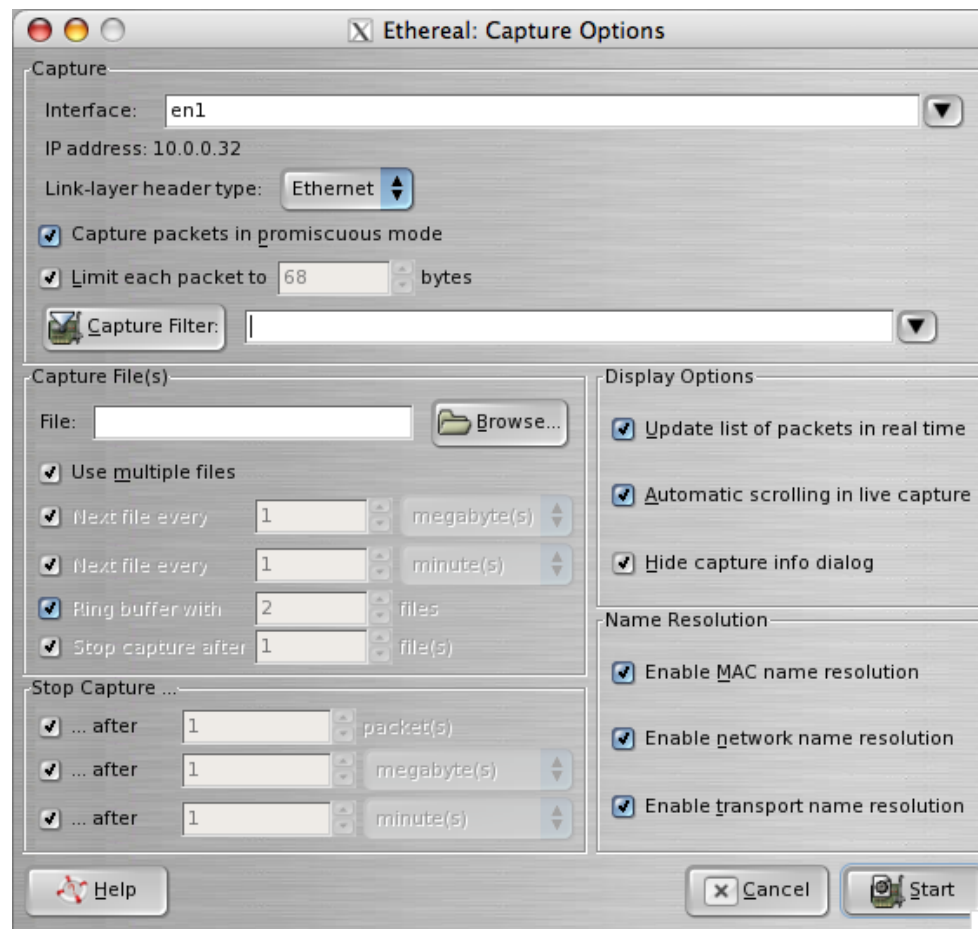
Wireshark 0.99.3 has been released. Security-related vulnerabilities in the SCSI, DHCP, ESP, and Q.2931 dissectors have been fixed. See the [advisory](#) for details.

<http://www.wireshark.org>

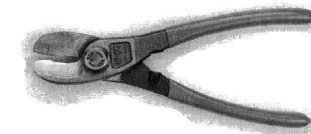
både til Windows og UNIX, tidligere kendt som Ethereal



Brug af Wireshark



Man starter med Capture - Options



Brug af Wireshark

The screenshot shows the Wireshark interface with the following data in the packet list:

No.	Time	Source	Destination	Protocol	Info
561	6.760947	10.0.0.32	sunny.kramse.dk	TCP	54021 > imap [ACK] Seq=426 Ack=10775 Win=65535 Len=0
562	6.763144	sunny.kramse.dk	10.0.0.32	TLS	Continuation Data, [Unreassembled Packet]
563	6.820037	10.0.0.32	sunny.kramse.dk	TCP	54021 > imap [ACK] Seq=426 Ack=11106 Win=65535 Len=0
564	6.919635	10.0.0.32	sunny.kramse.dk	TCP	54023 > imap [SYN] Seq=0 Ack=0 Win=65535 Len=0
565	6.921708	sunny.kramse.dk	10.0.0.32	TCP	imap > 54023 [SYN, ACK] Seq=0 Ack=1 Win=1638 Len=0
566	6.921794	10.0.0.32	sunny.kramse.dk	TCP	54023 > imap [ACK] Seq=1 Ack=1 Win=65535 Len=0
567	6.922614	10.0.0.32	sunny.kramse.dk	TLS	Client Hello

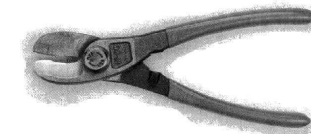
The detailed view for packet 563 shows the following structure:

- Frame 563 (66 bytes on wire, 66 bytes captured)
- Ethernet II, Src: AppleCom_86:7c:3f (00:0d:93:86:7c:3f), Dst: Olicom_c3:57:d8 (00:00:24:c3:57:d8)
- Internet Protocol, Src: 10.0.0.32 (10.0.0.32), Dst: sunny.kramse.dk (217.157.20.131)
- Transmission Control Protocol, Src Port: 54021 (54021), Dst Port: imap (993), Seq: 426, Ack: 11106, Len: 0

The raw packet bytes are displayed in hexadecimal and ASCII:

```
0000 00 00 24 c3 57 d8 00 0d 93 86 7c 3f 08 00 45 00  ..$.W... ..|?..E.  
0010 00 34 7e 8b 40 00 40 06 c3 f8 0a 00 00 20 d9 9d  .4~.@@. .... ..  
0020 14 83 d3 05 03 e1 cd 31 c9 ea 0d 7b a2 bf 80 10  .....1 ...{....  
0030 ff ff 32 0d 00 00 01 01 08 0a 62 e0 c3 42 bb e3  ..2..... ..b..B..
```

Læg mærke til filtermulighederne

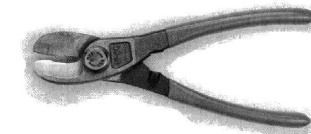


Nmap / Zenmap <http://www.nmap.org>

The screenshot shows the Zenmap application window. The 'Target' field contains '192.168.18.158' and the 'Profile' is set to 'Intense Scan'. The command entered is 'nmap -T4 -A -v 192.168.18.158'. The main display area shows the Nmap output for the scan.

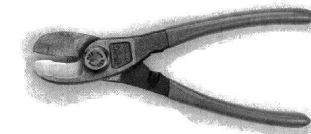
```
nmap -T4 -A -v 192.168.18.158

Nmap scan report for xman (192.168.18.158)
Host is up (0.030s latency).
Not shown: 996 filtered ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 4.3 (protocol 2.0)
| ssh-hostkey: 1024 c9:b5:f5:32:f8:51:bb:91:d5:32:e3:ad:17:48:5b:1e (DSA)
|_ 2048 c3:c7:6c:78:83:cd:87:c8:75:9f:2b:97:5b:a4:b7:1b (RSA)
80/tcp    open  http     Apache httpd
|_html-title: Observium :: Network Observation and Monitoring
|_http-methods: No Allow or Public header in OPTIONS response (status code 200)
|_http-favicon: Unknown favicon MD5: 2AE73A875A9FFA274B16F72BBF72B6D7
443/tcp   open  ssl/http Apache httpd
|_html-title: Index of /
|_http-methods: GET HEAD POST OPTIONS TRACE
|_Potentially risky methods: TRACE
|_See http://nmap.org/nsedoc/scripts/http-methods.html
631/tcp   closed ipp
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.9 - 2.6.18
Uptime guess: 2.712 days (since Sun Jul 25 20:35:12 2010)
Network Distance: 2 hops
```



Scapy <http://www.secdev.org/projects/scapy/>

```
h1k@bigfoot:h1k$ sudo scapy-2.6
WARNING: No route found for IPv6 destination :: (no default route?)
Welcome to Scapy (2.1.0)
>>> ans,unans=sr(IP(dst="192.168.18.50-100")/ICMP())
Begin emission:
.....*.*.....*.*.....*.....Finished to send 51 pa
.*.....*.....
....
.....^C
Received 168 packets, got 14 answers, remaining 37 packets
>>> ans.summary(lambda (s,r): r.sprintf("%IP.src% is alive") )
192.168.18.50 is alive
192.168.18.54 is alive
192.168.18.60 is alive
192.168.18.62 is alive
192.168.18.68 is alive
192.168.18.72 is alive
192.168.18.78 is alive
192.168.18.86 is alive
....
```

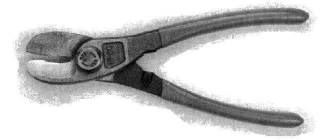


System information

Name	aske.kramse.dk
Version	1.1b16 built on Sun Jul 18 10:25:58 CEST 2004
Platform	net45xx
Uptime	1:35PM up 2 mins, load averages: 0.11, 0.08, 0.03

Brug en GUI firewall første gang.

Spørgsmål?



Flemming Jacobsen
fj@batmule.dk
Henrik Lund Kramshøj
hlk@kramse.org

You are always welcome to send me questions later via email