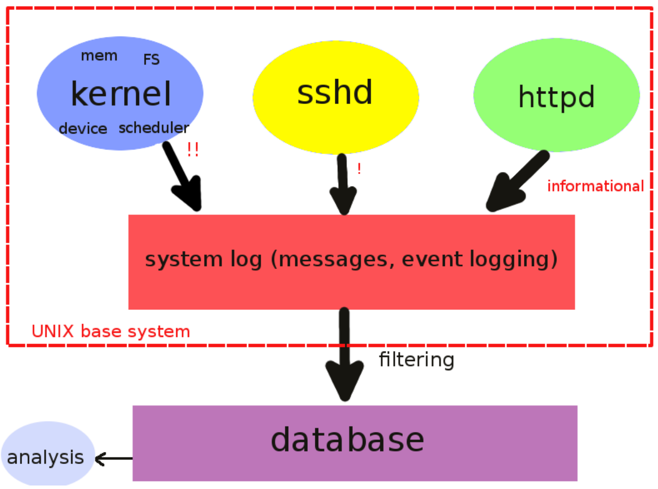
9. System Logging



Nuclear World Example

International Atomic Energy Agency (IAEA)

Chernobyl, Fukushima etc

Priorities: how important is the message

Facility: which subsystem

Configuration: where should the message be delivered

the log file is the newspaper

Nuclear World: INES scale

```
Level 7: Major accident
         Chernobyl, 26 April 1986
         Fukushima, 11 March 2011
Level 6: Serious accident
         Kyshtym, Mayak, Soviet Union, 29 September 1957
Level 5: Accident with wider consequences
         Harrisburg, Pennsylvania, 28 March 1979
         Goiânia, Brazil, 13 September 1987
Level 4: Accident with local consequences
         Tokaimura, Japan, 1999
Level 3: Serious incident
Level 2: Incident
         Cattenom, 18. Januar 2012
Level 1: Anomaly
Level 0: Deviation
```

Auditing

Auditing = the ability to tell **when who** *did* **what** *to* **what**

Example:

- on **Monday, June 1st, 2014**,
- user `video-wizard`
- did **overflow**
- the `/home` filesystem

Resource: system wide log file

(extension: network wide log file)

How to Write to the System Log (1)

`open()`, `fopen()` ? No! (Do not even think about it.)

Assume two processes writing simultaneously.

Serialization needed!

Assume you want to store the logs somewhere else.

Configurability needed!

How to Write to the System Log (2)

Solution: a special process, called `syslogd` *(syslog daemon)*

- serializes write requests
- can be configured in various ways
- may be reached over a network
- is supported by the C library (`syslog(3)`)

System Messages: Facility

which **subsystem** causes the message

- Kernel
- Mail System
- System Daemons
- Printer System

⋮

-

Keywords:

auth, authpriv, console, cron, daemon, ftp, kern, lpr, mail, mark, news, ntp, security, syslog, user, uucp, local0 through local7

System Messages: Priority

how **important** is the message

value	constant	name	description
0	`LOG_EMERG`	emergency	system is unusable
1	`LOG_ALERT`	alert	action must be taken immediately
2	`LOG_CRIT`	critical	critical conditions (probably hardware)
3	`LOG_ERR`	error	error conditions
4	`LOG_WARN`	warning	warning conditions
5	`LOG_NOTICE`	notice	normal but significant condition
6	`LOG_INFO`	info	informational message
7	`LOG_DEBUG`	debug	debug-level message

Keywords:

emerg, alert, crit, err, warning, notice, info, debug

System Messages: Examples (FreeBSD Kernel)

emergency	Killing all existing sessions... (going single-user)
alert	reboot after panic
critical	RAM parity error (hardware failure)
error	network card: Loss of carrier during transmit
warning	attempted source route from %s to %s
notice	ktrace write failed, errno %d, tracing stopped
info	pid %d (%s), uid %d: exited on signal %d
debug	arplookup %s failed

System Messages: Examples (Linux Kernel)

emergency	system to be rebooted, memory shortage for kernel
alert	kernel programming errors (NULL pointer etc)
critical	FS: corrupted header, SMP: 2nd CPU doesn't work
error	Out of Memory: Killed process
warning	network card: i82586 not responding, giving up
notice	network card: promiscuous mode enabled
info	TCP: time wait bucket table overflow
debug	UDP: IPv4 hw checksum failure

Configuring `syslogd`

configuration file `/etc/syslog.conf`

- facility/priority
  example: `kern.crit` are all kernel messages with priority *critical* or higher
- destination
  - a file (starts with „/")
    typical filename for all messages is `/var/log/messages`
  - a host (starts with „`@`")
  - a user

see manual page of `syslog.conf`

after changing the configuration file ⇝send SIGHUP to `syslogd`

Shell Command Line Interface

```
logger -p local0.notice -t HOSTIDM "Message"
```

Standard: IEEE Std 1003.2 ("POSIX.2")

Programming Interface

- use openlog() to set
  - the name of your process
  - some options (normally logging the PID)
  - the facility
- use syslog() to
  - set the priority
  - write the actual message (printf–style format)

```
void openlog( char *ident, int option, int  facility);
example: openlog("inetd",LOG_PID,LOG_DAEMON);

void syslog( int priority, char *format, ...);
example: syslog(LOG_WARNING,"invalid host %s",ip_address);

void closelog( void );
```

Side note: What is a Daemon

process running in background

often started at boot time
and terminated at shutdown

system daemons: name ends with `d`

examples: syslogd, sshd, inetd, in.telnetd, httpd, lpd, nfsd,. . .

exceptions: sendmail, portmap, ypserv, . . .

10. Network

Network Configuration

subtle differences between UNIX systems

1. Network Interface Card (NIC)
   - must be recognized by the kernel

   ⇝ kernel configuration
   - is then available under a name like
     - *fxp0, em0, vr0, . . .* depends on driver (BSD)
     - *eth0, eth1, . . .* (Linux)
2. IP address (broadcast, netmask)
   - must be configured via `ifconfig`
   - example (Linux/Solaris/BSD)

```
ifconfig eth0 134.96.216.97 netmask 255.255.255.0 \
              broadcast 134.96.216.255
```

3. Routing
   - adding a default gateway
   - example `route add default gw 134.96.216.1` (Linux)
   - example `route add default 134.96.216.1` (BSD)

4. DNS
   - add entry `nameserver` in `/etc/resolv.conf`
   - add entry `search` in `/etc/resolv.conf`
   - use DNS diagnosis tools `dig` and `host`
   - do **not** use `nslookup`

DNS Configuration at HTW

```
domain  htw-saarland.de
nameserver      134.96.208.98
nameserver      134.96.7.100
```

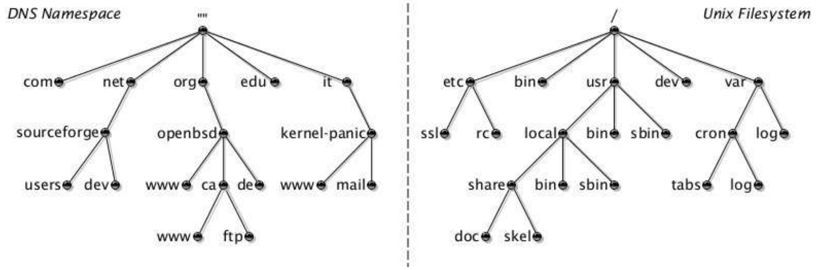
need not type in `htw-saarland.de` for local hosts

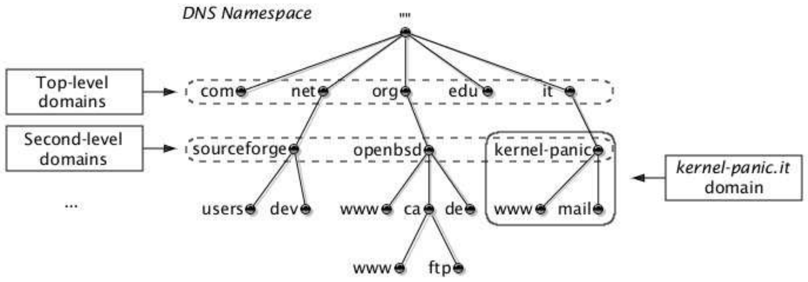
`search` keyword allows this for up to 6 domains

`nameserver` is a host waiting for queries on UDP/53

Background on DNS

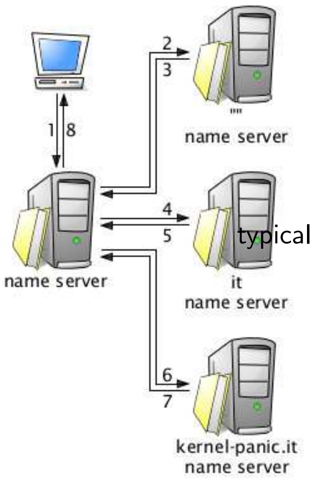
pictures from http://www.kernel-panic.it/openbsd/dns/





searching `www.kernel-panic.it`

1. client: dear local NS, address `www.kernel-panic.it`?
2. local NS: dear root server, NS for `.it`?
3. root server: that's `m.dns.it`, `r.dns.it`,. . . !
4. local NS: dear `m.dns.it`, NS for `kernel-panic.it`?
5. m.dns.it: that's `dns.technorail.com`
6. local NS asks `dns.technorail.com`: host `www.kernel-panic.it`?
7. `dns.technorail.com`: 62.149.140.23 !
8. local NS to client: your answer . . . 62.149.140.23 !



recursive query of client

Root Servers

```
$ dig +short . NS | sort
a.root-servers.net.
b.root-servers.net.
c.root-servers.net.
d.root-servers.net.
e.root-servers.net.
f.root-servers.net.
g.root-servers.net.
h.root-servers.net.
i.root-servers.net.
j.root-servers.net.
k.root-servers.net.
l.root-servers.net.
m.root-servers.net.
```

```
$ for x in ‘dig +short . NS | sort‘; do
echo $x" "‘dig +short $x‘ ;
  done
a.root-servers.net. 198.41.0.4
b.root-servers.net. 192.228.79.201
c.root-servers.net. 192.33.4.12
d.root-servers.net. 128.8.10.90
e.root-servers.net. 192.203.230.10
f.root-servers.net. 192.5.5.241
g.root-servers.net. 192.112.36.4
h.root-servers.net. 128.63.2.53
i.root-servers.net. 192.36.148.17
j.root-servers.net. 192.58.128.30
k.root-servers.net. 193.0.14.129
l.root-servers.net. 199.7.83.42
m.root-servers.net. 202.12.27.33
```

DNS Records

there are different *types* of addresses

- A records: request *host*, reply *IP*

  ```
  $ dig +short isl-s-01.htw-saarland.de
  134.96.216.91
  ```

- MX records: request *mail-domain*, reply *mail server* (with prio)

  ```
  $ dig +short htw-saarland.de MX
  80 m-relay2.rz.uni-saarland.de.
  90 m-relay3.rz.uni-saarland.de.
  20 m-relay.htw-saarland.de.
  80 m-relay.rz.uni-saarland.de.
  ```

- SOA records: request *domain*, reply *administrative parameters*

```
$ dig +short htw-saarland.de SOA
ns.rz.uni-saarland.de. Margit\.Meyer.htw-saarland.de. ...
```

- NS records: request *domain*, reply *name-server*

```
$ dig +short htw-saarland.de NS
ns.rz.uni-saarland.de.
ns1.htw-saarland.de.
ns.htw-saarland.de.
ws-ber1.win-ip.dfn.de.
```

- PTR records: reverse DNS lookup

```
$ dig +short 81.216.96.134.in-addr.arpa ptr
isl-c-01.htw-saarland.de.
```

- CNAME records: alias names

```
$ dig +short www.htw-saarland.de cname
www-portal.htw-saarland.de.
```

Problem with Message Authentication in DNS

no proof that the DNS replies are correct

most often not a problem, but attack may be invisible

September 2011:
TurkGuvenligi hackers at `NetNames`
(SQL injection, large DNS database)

⇝

The Register, Daily Telegraph, UPS, Vodafone, National Geographic unreachable

Solution for Message Authentication in DNS

DNSSEC

cryptographically signed replies

need verification of keys at upper level domain

⇝ Internet Root Key (key signing ceremony June 2010 at ICANN)

2048 Bit RSA key, exponent 65537

DNSSEC Public Root RSA Key n, 2048 bit

```
212080981482271179603433218336128380960348702216471804375634
752963886261174948922549730742230010889227176334389671001187
445061333321597244394586900211829034119507953509095412664181
880790068560654086737273589530048917616594142313599848269018
791334251777528334768898329417222606946611460349624548052675
823461094216078028921375693190158930463129431365424202829722
518280188940007805468651294368334724306795996666724315293828
780021358722372730781995405833538022370296023031578773132968
577112651044811609937159576661897359094365846815525820603432
963173913867839939291085454025649921514522604028740120613078
01750574916077373
```

protects DNS replies from top-level-domain servers

DNSSEC Public de Key

```
$ dig +short de DNSKEY

257 3 8 AwEAAYbcKo2IA8l6arSIiSC+l97v2vgNXrxjBJK+XkX5FYMPDfr2QgtU
        MHfjLPfMKiSxEXT0uL+SucI1ohv5I0C/pgz9e9NFDhMCpHLPA5s9LIzQ
        MHEs7Y+idlsRnBKe9Kw/B1RxzSZKxMd8UyAeA6j0vlZIKrokc1nr4ouv
        DhoYR3JDd7vCcvV08EIuaPgL0ijUYk07100jRFG+waRZnVPAwFZsgDIg
        BJqDl/nRVRBI8k3YFVPka6Rls/EIDYloqG+X5VZC/VXbBb7fams8misz
        3MsLeVy/fiH0j8SJMAZSbQxqo+/zWUJogl4Tyb5TbT1LRTfbyxII2zQ/
        ATXocW0ohSU=

256 3 8 AwEAAY0x2KKtTfeuIf/F6/W74mU3TSZMh4t+ARboRgxg0k5BK/kZ3slF
        zolY9t+jMIzqX+Rrjl0cHq6W+ERBEzsSvgjUwd3ZwJbWhvI4H4APgxLu
        oHv5p65SdtLT6nTUoKxGjRCEQexAn/MmxWQM37iHqi2ELVFABWDikKZg
        CZRGpQM9
```

DNSSEC Public de RSA Key n, 2048 bit

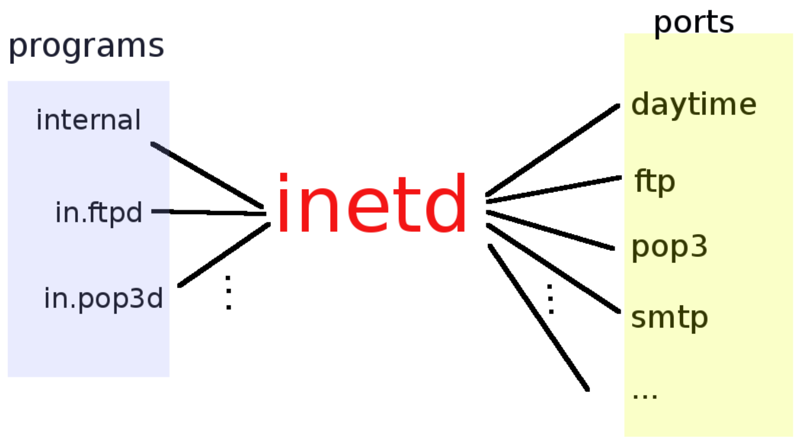
```
170245008759527409571354880559973545773595045766035877521381
230192254190427952179633423487450196083111936869630316806390
088168778768822124997586494803323322293779879574098656014691
587356760471071208951608891340174037164074411723193663727424
543918222399195337278617652557403471599164042884474959681049
427688604642089577745088553655249251823251030333632514792530
627275510648595234632942190926343227157475045064381270790970
639662414615393403148254991906430531798810260850123810638940
976251076124954202462300130573519166325909971423832778506192
958556496794408655823629060591884047534038987999318933441655
46169719203595557
```

protects DNS replies from servers within de domain

Network Services

- standalone services with own startscript, for example
  - `sshd`,
  - `dhcpd`,
  - `ypserv`,
  - `portmap`,
  - ...
- *inetd*–managed services (see `/etc/inetd.conf`)

`inetd`



inetd

the ,,internet superserver”

one process to listen on all ports

after accepting an incoming connection,

inetd invokes a program

to handle the connection

/etc/inetd.conf

```
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
echo       stream  tcp  nowait  root  internal
echo       dgram   udp  wait    root  internal
# discard  stream  tcp  nowait  root  internal
# discard  dgram   udp  wait    root  internal
ftp        stream  tcp  nowait  root  /usr/sbin/in.ftpd  in.ftpd
telnet     stream  tcp  nowait  root  /usr/sbin/in.telnetd  in.telnetd
#telnet    stream  tcp  nowait  root  /usr/sbin/tcpd  in.telnetd
#login   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rlogind
```

tcpd is a so–called tcp–wrapper

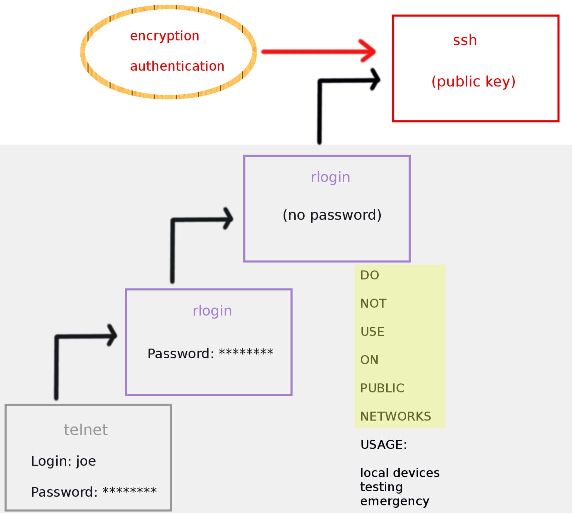
which can be configured to deny connections from certain IPs

but this can be (more efficiently) achieved by firewalling

Configuring inetd

- inetd must be installed
  - program /usr/sbin/inetd
  - script /etc/rc.d/inetd
- /etc/inetd.conf must be correct
  - use comment char ,,#” to disable services
  - read service name/port from /etc/services
  - check if corresponding programs are installed
- inetd must be running
  - startscript in boot sequence of runlevel or
  - started by hand
- inetd must know about configuration changes
  - send SIGHUP signal after editing /etc/inetd.conf

Login via Network



encryption
authentication

ssh
(public key)

rlogin
(no password)

rlogin
Password: ********

telnet
Login: joe
Password: ********

DO
NOT
USE
ON
PUBLIC
NETWORKS

USAGE:

local devices
testing
emergency

Login via Network

- telnet : a virtual terminal, first form of remote login
- rlogin : simplified remote login
- ssh : encrypted, authenticated remote login

telnet

Client side: telnet

Server side: inetd (in.telnetd)

```
$ telnet isl-s-01
Trying 134.96.216.91...
Connected to isl-s-01.
Escape character is '^]'.

FreeBSD/i386 (isl-s-01.htw-saarland.de) (ttyp1)

login: dweber
Password:
Last login: Thu Jun 19 10:07:54 from isl-s-02.htw-sa
Copyright (c) 1992-2007 The FreeBSD Project.
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
The Regents of the University of California. All rights reserved.

FreeBSD 6.2-RELEASE (GENERIC) #0: Fri Jan 12 10:40:27 UTC 2007

$
```

rlogin

Client side: `rlogin`

Server side: `inetd (in.rlogind)`

uses same user name

```
$ rlogin buddy
Password:
Last login: Sun Jul  4 11:27:50 from buddy.local
Linux buddy 2.4.24 #7 Fri Feb 13 23:25:00 CET 2004
dw@buddy(1):~$
```

rlogin: .rhosts

write *trusted host/user combinations* into `$HOME/.rhosts`

example: entry on host buddy

```
$ cat .rhosts
somehost dw
```

then `dw` may login to buddy without password

```
dw@somehost$ rlogin buddy
Last login: Sun Jul  4 11:31:36 from buddy.local
Linux buddy 2.4.24 #7 Fri Feb 13 23:25:00 CET 2004
dw@buddy(1):~$
```

Problems of rlogin/telnet

the network connection is

- unauthenticated
  - is the target host genuine?
  - is the connecting host genuine?
- unencrypted
  everybody sniffing on the wire
  (for example promiscious mode NIC)
  can read passwords, transmitted data, . . .

Public Key Cryptography (1)

solves both problems

every user U has

- a public key P_U
- a secret (private) key S_U

Example: To send a message m to Alice, Bob must compute

$$m' = E(P_{Alice}, m)$$

Alice decrypts m' by computing

$$D(S_{Alice}, m')$$

Public Key Cryptography (2)

The encryption function $E()$

and

the decryption function $D()$

are public.

$\rightsquigarrow$ it must be impossible to compute S_U from P_U

Public Key Cryptography (3)

There are three algorithms which are more or less used in PKC:

- RSA (based on factoring, 1978)
- DSA (based on discrete logs in Galois fields, 1985)
- ECDSA (based on discrete logs on elliptic curves, 1989)

World records for breaking these schemes:

- factoring 768 bits (232 decimal digits) in 2010 (Uni Bonn)
- factoring 663 bits (200 decimal digits) in 2005 (Uni Bonn)
- discrete log in $GF(p)$, p with 596 bits in 2014 (Loria, FR)
- DL on EC over $GF(p)$, p with 109 bits in 2002

Recommended key sizes for these schemes

- RSA 2048 bits
- DSA 2048 bits
- ECDSA 160 bits

ssh

Client side: `ssh`

Server side: `sshd`

Implementation: OpenSSH and others

Properties

- authenticated
  - connecting host must prove its identity (public key)
  - accepting host must prove its identity (public key)
  - user must prove his identity (public key, password)
- encrypted connection (especially no plain text passwords)

Public Key authentication:

```
$ ssh isl-1-01
Enter passphrase for key '/home/dweber/.ssh/id_dsa':
Last login: Mon Jul 16 15:46:13 2012 from stl-s-studwork.htw-saarland.de
FreeBSD 9.0-STABLE (ISL-S-01) #0: Wed Jun 13 01:32:10 CEST 2012
```

SSH Keys: Host Key

each host with `sshd` has an SSH key

```
$  cat /etc/ssh/ssh_host_rsa_key.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDqLXoHYuKw4m/
PjYO9OoqQjevFkAUpxK3YFVntCDjwoC+R8QB6d7CTguqTDNW9h1
1tDB2OxIz9UZvG6bjVb6Gj7cr6QoRe3K6JMah4My6tdufM+W8Mc
MTE6r/vo/OFgMuJ0juPKD9sjjXP3yfjNSaE1qU+RBCVnEcFSCHM
1uYmIuGl0OEOFFTbbiTETY2A6PCzV3EUD1vUmXIOEZBQmqSIkxF
8AUsttTRDbcLaWK32hhnQpjM4agTSqBIjNGzv8OrA/JIkThn7+A
ZAWccvziqqMDrdyB+539S42rbusY2h9ImZmIeHbOllfY0zy5E5y
q3Ied7CL0CpUwfjhLAa7h+H root@isl-1-01
$ cat /etc/ssh/ssh_host_rsa_key
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEA6i16B2LisOJvz42DvTqKkI3rxZAFKcSt2BVZ7Qg48KAvkfEA
enewk4LqkwzVvYddbQwdtMSM/VGbxum41W+ho+3K+kKEXtyuiTGoeDMurXbnzPlv
...
b4ctkgXh0dvNMVVMoFOBR8xY4YDgPwVBN6+Yo4NsppEaujfG4A==
-----END RSA PRIVATE KEY-----
```

SSH Keys: User Key

user may use a key for authentication

⇝key may replace password

SSH Keys: Generate User Key

```
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/dw/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/dw/.ssh/id_rsa.
Your public key has been saved in /home/dw/.ssh/id_rsa.pub.
The key fingerprint is:
8f:d6:39:5e:d5:9e:cb:62:9f:8f:64:cb:a8:37:7b:66 dw@buddy
$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/dw/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/dw/.ssh/id_dsa.
Your public key has been saved in /home/dw/.ssh/id_dsa.pub.
The key fingerprint is:
99:38:a4:10:e0:bb:a5:27:5f:48:a1:67:33:28:e3:cc dw@buddy
```

SSH Keys: Use User Key Instead of Password

Add the public user key of your system

- $HOME/.ssh/id_rsa.pub
- $HOME/.ssh/id_dsa.pub

to $HOME/.ssh/authorized_keys of the remote system

```
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAA... dw@somehost
```

Now dw@somehost can login without giving the password.

But dw@somehost must type the passphrase on his system.

Use ssh-agent to avoid typing the passphrase.

SSH: Trust the Accepting Host?

Remember: You are going to type a password now!

first time connect:

```
$ ssh buddy
The authenticity of host 'buddy (192.168.1.5)' can't be established.
RSA key fingerprint is 1c:c8:74:7d:39:8f:35:ba:f4:d9:57:86:c2:1c:f3:4c.
Are you sure you want to continue connecting (yes/no)?
```

fingerprint = MD5-hash

fingerprint check on remote system

```
buddy: # ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key.pub
1024 1c:c8:74:7d:39:8f:35:ba:f4:d9:57:86:c2:1c:f3:4c ssh_host_rsa_key
```

12. Firewalling

Keep the bad guys out,

and let the good guys in.

Firewalls have rules to *refuse* IP packets,

and *accept* them.

First–match–logic: `iptables`, `ipfw`

- rule after rule
- until a rule (positively or negatively) matches

Last–match–logic: `pf`, `ipfilter`

- rule after rule
- the last rule that matches determines target

⇝firewalling is part of TCP/IP code, therefore part of kernel

ipfw: the FreeBSD Way of Firewalling

enable firewalling in `/etc/rc.conf`

```
firewall_enable="YES"
firewall_type="client"
```

add rules to `/etc/rc.firewall` for the chosen firewall type

- open – no rules
- client – no servers on this machine
- simple – basic server configuration (DNS, HTTP, NTP)
- closed – all IP services disabled, except loopback

ipfw: the FreeBSD Way of Firewalling

adding rules

```
# allow local net
ipfw add pass all from 134.96.216.0/24 to me

# Allow IP fragments to pass through
ipfw add pass all from any to any frag

# Allow setup of outgoing TCP connections only
ipfw add pass tcp from  me to any setup keep-state
```

ipfw (2)

```
# Disallow setup of all other TCP connections
ipfw add deny tcp from any to any setup

# Allow DNS queries out in the world
ipfw add pass udp from me to any 53
ipfw add pass udp from any 53 to me

# Everything else is denied by default, unless the
# IPFIREWALL_DEFAULT_TO_ACCEPT option is set in your kernel
# config file.
```

ipfw: statistic of usage (per rule)

used to verify correctness of rules

```
# ipfw show
00100   1960    209528 allow ip from any to any via lo0
00200      0         0 deny ip from any to 127.0.0.0/8
00300      0         0 deny ip from 127.0.0.0/8 to any
00400 628018 379969470 allow ip from me to 134.96.216.0/24
...
65535      0         0 deny ip from any to any
```

shows rule and number of packets/bytes since last counter reset

pf: the OpenBSD Way of Firewalling

. . . but ported to FreeBSD

`http://www.de.openbsd.org/faq/pf/`

- fast
- low system resources
- secure
- pfauth: can accept IP with valid SSH authentication
- passive OS detection

iptables: the Linux Way of Firewalling

first–match–logic

a *chain* is a list of rules

there are 3 built–in chains

chain	meaning
INPUT	for all incoming packets
OUTPUT	for all outgoing packets
FORWARD	for all packets routed through

iptables: Targets

a target is an action in case a rule matches

- ACCEPT the packet is processed as normal
- DROP the packet is discarded
- REJECT the packet is discarded, send ICMP to source
- LOG write packet to syslog
- DNAT destination address rewriting
- SNAT source address rewriting
- MASQUERADE source address rewriting

iptables: Examples (1)

show rules

```
iptables -L
```

drop some broadcasts in LAN

```
iptables -A INPUT -p udp -d 255.255.255.255 -j DROP
iptables -A INPUT -p udp -d 134.96.255.255  -j DROP
iptables -A INPUT -p udp -d 134.96.214.255  -j DROP
```

iptables: Examples (2)

disallow specific source address

```
iptables -A INPUT -s 64.94.110.0/24 -j REJECT
```

iptables: Examples (3)

allow specific source address

```
# allow localhost
iptables -A INPUT --protocol ip -s 127.0.0.1 -j ACCEPT

# allow stl-k-16 ssh
iptables -A INPUT -s 134.96.216.26  --protocol tcp
               --destination-port 22 -j ACCEPT
```

Good Luck while Defending Against Hackers

http://www.claybennett.com/pages/info_superhighway.html