

## Memory

### Identify memory usage of TIME\_WAIT sockets

```
slabtop -o | grep -E '(^ OBJS|tw_sock_TCP|tcp_bind_bucket)'
```

OBJS	ACTIVE	USE	OBJ	SIZE	SLABS	OBJ/SLAB	CACHE	SIZE	NAME
13104	5813	44%	0.03K	117	112	468K	tcp_bind_bucket		
6900	6408	92%	0.19K	345	20	1380K	tw_sock_TCP		

## OpenSSH

Because SSH is incredible.

### Tunneling

```
# Using connect-proxy as a socks proxy (tunneling ssh over http proxy to an  
ssh server on port 1.2.3.4:443)  
Host 1.2.3.4  
ProxyCommand connect-proxy -H proxy:3128 %h %p  
Port 443  
User shaun  
  
# -W can be used for raw port-forwarding on OpenSSH 5.4 and higher  
Host internal.server  
HostName internal.server.com  
User shaun  
ProxyCommand ssh shaun@intermediary.server.com -W %h:%p  
  
# Here's the old way, with netcat  
Host internal.server  
HostName internal.server.com  
User shaun  
ProxyCommand ssh shaun@intermediary.server.com nc %h %p  
  
# New to OpenSSH 7.3 and higher is the ProxyJump command, which does the  
same, but with multiple possible intermediaries  
Host internal.server  
HostName internal.server.com  
ProxyJump shaun@intermediary1.server:22,shaun@intermediary2.server:22  
User shaun
```

### SSH as a VPN

```
ssh -NTcf -w 0:0 <destination>
```

```
# Machine A
ip link set tun0 up
ip addr add 10.0.0.100/32 peer 10.0.0.200 dev tun0

# Machine B
ip link set tun0 up
ip addr add 10.0.0.200/32 peer 10.0.0.100 dev tun0

# Add a route for target network on Machine B
ip route add 10.0.0.0/24 via 10.0.0.200

#This allows us to send packets from Machine B to any IP address on Network
A, via Machine A.
#To ensure that packets have a route back to Machine B add an arp entry on
Machine A:

arp -sD 10.0.0.200 eth0 pub

#This sets a published arp destination for 10.0.0.200 to Machine A (proxy-
ARP).

# Kernel packet forwarding must be enabled for the routing bits
echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward
```

### Copy and install public key

```
ssh-copy-id -i .ssh/id_rsa.pub user@server.com
```

## Filesystems

[xfs](#)  
[ext4](#)  
[recovery](#)

### Get detailed memory chip information

```
dmidecode --type 17
```

Sample output:

```
# dmidecode 2.12
SMBIOS 2.7 present.

Handle 0x003B, DMI type 17, 34 bytes
Memory Device
    Array Handle: 0x002C
    Error Information Handle: Not Provided
```

```
Total Width: 72 bits
Data Width: 64 bits
Size: 32 GB
Form Factor: DIMM
Set: None
Locator: D0
Bank Locator: /SYS/MB/P0
Type: DDR3
Type Detail: Synchronous
Speed: 1066 MHz
Manufacturer: Samsung
Serial Number: 366112E5
Asset Tag:
Part Number: M393B4G70BM0-YH9
Rank: 1
Configured Clock Speed: 1066 MHz
```

## OpenSSL

### Convert .crt to .pem

```
openssl x509 -in certificate.crt -out certificate.pem -outform PEM
```

### View certificate and key

```
openssl x509 -noout -text -in server.crt
openssl rsa -noout -text -in server.key
```

### Verify certificate matches key

The `modulus` and the `public exponent` portions in the key and the Certificate must match. But since the public exponent is usually 65537 and it's bothering comparing long modulus you can use the following approach:

```
openssl x509 -noout -modulus -in server.crt | openssl md5
openssl rsa -noout -modulus -in server.key | openssl md5
```

### Check which key or cert belongs to a CSR

```
openssl req -noout -modulus -in server.csr | openssl md5
```

### Show local certificate details

```
openssl s_client -showcerts -servername www.virtualhost.co.za -connect
localhost:443 </dev/null | openssl x509 -text
```

## Tcpdump

Dump TCP Rsets

```
tcpdump -fnni bond0:-nvvS 'tcp[tcpflags] & (tcp-rst) != 0'
```

## Other

Conceal process in 'ps'

```
echo FakeProcName > /tmp/cmdline
mount -n --bind -o ro /tmp/cmdline /proc/<pid>/cmdline

ps -ef | grep FakeProcName
```

speedtest

```
curl -s https://raw.githubusercontent.com/sivel/speedtest-
cli/master/speedtest.py | python -
```

get kernel debuginfo packages for systemtap and crash

For Unbreakable Enterprise Kernel:

```
export DLP="https://oss.oracle.com/ol7/debuginfo"
wget ${DLP}/kernel-uek-debuginfo-`uname -r`.rpm
wget ${DLP}/kernel-uek-debuginfo-common-`uname -r`.rpm
```

For Red Hat Compatible Kernel:

```
export DLP="https://oss.oracle.com/ol7/debuginfo"
wget ${DLP}/kernel-debuginfo-`uname -r`.rpm
# wget ${DLP}/kernel-debuginfo-common-`uname -r`.rpm
```

Install

```
rpm -Uhv kernel-uek-debuginfo-4.1.12-112.14.15.el7uek.x86_64.rpm \
kernel-uek-debuginfo-common-4.1.12-112.14.15.el7uek.x86_64.rpm
```

## Get interrupts causing high system time

```
sar -I XALL 1 | grep -v 0.00
```

## Iptables

### icmp rate limiting

```
iptables -A INPUT -p icmp -m icmp --icmp-type 8 -m limit --limit 1/second  
--limit-burst 1 -j ACCEPT  
iptables -A OUTPUT -p icmp -m icmp --icmp-type 8 -m limit --limit 1/second  
--limit-burst 1 -j ACCEPT
```

### Oracle Enterprise Linux

From:  
<https://wiki.dewberry.co.za/> - Shaun's Wiki

Permanent link:  
<https://wiki.dewberry.co.za/doku.php?id=linux>

Last update: **2020/03/06 13:53**

