Mac OS X Application and Network Firewalls

Emory Lundberg
Security Friend
Information Security & Policy Office
University of Iowa
Good Morning

* Firewallapalooza
* Recap of Episode 1
* Host-based firewalls
* Lion’s ipfiltering and application firewall
* a closer look at Lion’s network-level firewall
IN EPISODE ONE

* TCP/IP basics
* Network Architecture 101 (Switching, Routing)
* Netmasks and CIDR notation (/32, /16, /24)
* Stateless packet filters
* Stateful firewalls
speak up if you have a question or comment
Lion
Lion Security

- improved aslr
- wde (filevault 2)
- sandbox for safari
- no java. no flash.
- Apple ID as authorization
why does my mac need a firewall?

* system hardening
* brute forcing attacks :(
* protecting institutional data by applying additional layers of defense
* do you know what your Mac is doing?
<table>
<thead>
<tr>
<th>App</th>
<th>Status</th>
<th>Users</th>
<th>Download rate</th>
<th>Upload rate</th>
<th>Total in</th>
<th>Total out</th>
<th>Last activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>vpagentd</td>
<td>Active</td>
<td>root</td>
<td>1.55 MB/s</td>
<td>71 KB/s</td>
<td>1.94 GB</td>
<td>84.01 MB</td>
<td>11:07:27 AM</td>
</tr>
<tr>
<td>Vidalia</td>
<td>Active</td>
<td>emory</td>
<td>690 B/s</td>
<td>690 B/s</td>
<td>42.09 MB</td>
<td>5.54 MB</td>
<td>11:07:27 AM</td>
</tr>
<tr>
<td>MobileMe</td>
<td>Idle</td>
<td>2 users</td>
<td>-</td>
<td>-</td>
<td>203.23 KB</td>
<td>132.61 KB</td>
<td>11:06:49 AM</td>
</tr>
<tr>
<td>Back To My Mac</td>
<td>Idle</td>
<td>root</td>
<td>-</td>
<td>-</td>
<td>283.81 MB</td>
<td>27.22 MB</td>
<td>11:07:21 AM</td>
</tr>
<tr>
<td>Safari</td>
<td>Active</td>
<td>emory</td>
<td>-</td>
<td>-</td>
<td>21.09 MB</td>
<td>2.42 MB</td>
<td>11:07:25 AM</td>
</tr>
<tr>
<td>System</td>
<td>Active</td>
<td>root</td>
<td>-</td>
<td>-</td>
<td>20.89 MB</td>
<td>8.62 MB</td>
<td>11:07:23 AM</td>
</tr>
<tr>
<td>Mail</td>
<td>Active</td>
<td>emory</td>
<td>-</td>
<td>-</td>
<td>4.84 MB</td>
<td>393.75 MB</td>
<td>11:07:23 AM</td>
</tr>
<tr>
<td>usbmuxd</td>
<td>Active</td>
<td>_usbmuxd</td>
<td>-</td>
<td>-</td>
<td>3.60 MB</td>
<td>3.17 MB</td>
<td>11:07:18 AM</td>
</tr>
<tr>
<td>imageng</td>
<td>Active</td>
<td>emory</td>
<td>-</td>
<td>-</td>
<td>570.87 KB</td>
<td>160.50 KB</td>
<td>11:07:23 AM</td>
</tr>
<tr>
<td>FileSyncAgent</td>
<td>Idle</td>
<td>liz</td>
<td>-</td>
<td>-</td>
<td>168.92 KB</td>
<td>411.04 KB</td>
<td>11:06:40 AM</td>
</tr>
<tr>
<td>Dropbox</td>
<td>Active</td>
<td>emory</td>
<td>-</td>
<td>-</td>
<td>93.45 KB</td>
<td>74.59 KB</td>
<td>11:06:40 AM</td>
</tr>
<tr>
<td>Google Chrome</td>
<td>Active</td>
<td>liz</td>
<td>-</td>
<td>-</td>
<td>18.12 KB</td>
<td>17.38 KB</td>
<td>11:06:36 AM</td>
</tr>
</tbody>
</table>

---

Download:
- 1.55 MB/s
- 72 KB/s

Upload:
- 1.55 MB/s
- 72 KB/s

Users:
- root
- emory
- liz
- _usbmuxd
Yikes, 3 Firewalls?!

* alf
* ipfw
* pf
* + a mystery guest for OS X Server
alf
alf features

* highest level of filtering in OS X
* trusted binaries
* can be scripted
* has a (limited) GUI
* unusual love of cats
Firewall: Off

This computer’s firewall is currently turned off. All incoming connections to this computer are allowed.

Start  Click Start to turn the firewall on.

Click the lock to make changes.
Firewall: On

The firewall is turned on and set up to prevent unauthorized applications, programs, and services from accepting incoming connections.

Stop  Click Stop to turn the firewall off.

Click the lock to prevent further changes.
didn’t you say scripting?
this was
terminal
screenshot
/usr/libexec/ApplicationFirewall/socketfilterfw -h

... 
-t app set trusted app, e.g. -t app1 app2 app3
-s file sign file
-v file verify file

... 
--listapps display a list of paths of added applications
`socketfilterfw \--listapps`

- show exceptions
- explicitly allowed apps
- signed exceptions
- Trusted Apps list ($TRUSTEDAPPS) populated by Apple
- httpd, smbd, etc.
e.g.

* 22 : /System/Library/CoreServices/RemoteManagement/screensharingd.bundle/Contents/MacOS/screensharingd
   ( Block incoming connections )

* 23 : /System/Library/CoreServices/ServerManagerDaemon.bundle/Contents/MacOS/servermgrd
   ( Allow incoming connections )
other socketfilterfw options

* --getglobalstate, --setglobalstate
* --blockapp, --unblockapp, --getappblocked
* --getstealthmode, --setstealthmode
alf’s glue

* talks to other processes to negotiate
* configd(8)
* design goals
* OS X Server’s `afctl` and possible future
network firewalls

* ipfw in 10.7
* FreeBSD
* since Mac OS X Server 1.0-2 / Rhapsody!
* pf in 10.7
still using ipfw
the OpenBSD Packet Filter: pf

* introduced in 2000, OpenBSD 3.0
* Daniel Hartmeier
* replaces IPFilter due to license concerns
* stateful firewall, stateful filtering
What uses pf in Lion?

* alf
* connection sharing
* ? airdrop ?
* VPN services on Mac OS X Server (NAT etc.)
writing pf rules
(the hard part)
remember!

Pragmatic Notes: TCP/IP

To make network firewall rules you need to know:
- IP addresses involved
- Ports involved
- Protocol used (TCP/UDP/ICMP)

Useful Information:
- IPv4 addresses are 32bits & IPv6 addresses are 128bits
- Ports are 16bits (which can represent 0-65535)
- Ports 0-1023 are privileged ports.
- Client applications dynamically use high-number ports
pf rule syntax

action [direction] [log] [quick] \ 
[on interface] [proto protocol] 
[from src_addr [port src_port]] \ 
[to dst_addr] [port dst_port]] \ 
[flags tcp_flags] [state]
sample pf rule: outside > web server

* pass in on $ext_if proto tcp to $web_server port 80 keep state

* pass in on $ext_if proto tcp to $web_server port 80 keep state (max 200, source-track rule, max-src-nodes 100, max-src-states 3)
state

* stateful inspection
* state options in pf
  * max #, no state
  * source track (# per address)
* max-src nodes | states (limit # of addresses or state entries)
quick

* pf is last match

* you can use 'quick' to skip evaluating other rules
udp has no state

* ...but pf can figure it out
* timeouts from prior traffic
* can control state for udp in pf policy
Lion’s pf

* a bit dated, but still good!
* no altq
* no authpf
syn proxy

* A way to prevent attacks like syn floods
  * source > firewall > destination
  * source --> firewall --> destination
applied pf
fort awesome

* ipv6 via hurricane electric
* ipv4 rfc1918
* MediaCom > AirPort > Gigabit Ethernet & 802.11n 5GHz
* VMWare (bridged interfaces)
/etc/pf.conf
pretty sparse!

```bash
# Mac OS X Lion glue from Apple for anchors
#
# com.apple anchor point
#
# nat-anchors "com.apple/*"
# rdr-anchors "com.apple/*"
# anchor "com.apple/*"
# load anchor "com.apple" from "/etc/pf.anchors/com.apple"
```
pfctl (8)

- connects to the packet filter device
- kernel still handles forwarding
- you’ll need to create pflog0 (pflog (8))
pfctl -s
(show)

* -s nat
  shows all NAT rules

* -s States
  shows all state information

* tables, memory, sources, anchors

* -s all
  shows everything
-E, -d, and -f

* `pfctl -E` enables the ruleset

* `pfctl -d` disables the ruleset

* `-f` specifies the rules file, e.g. `pfctl -E -f /etc/pf.conf`
variables and macros
block all incoming, allow outbound

# block everything
#
block in log
block out

pass out quick on $onboard_if keep state
scanning lindsay
using "include"

include "/etc/pf.fortawesome.conf"
allow ssh, and my campus workstation

sample rules
letting my home network talk to me

sample rules
a host behind Lion on VMWare

sample rules
lindsay and edie

lindsay

edie

imaps, smtps, ssh, privoxy, web applications
allow the traffic

sample rules to allow edie and lindsay to chat
there we go!

it's working!
aren't these great redacted slides?!
talking to edie's

sample rules
sample network diagram showing a “server” system, an iPad, a MBP, a bunch of Windows systems on a building net, and a management system
common services

* syslog udp:514
* http tcp:80
  https tcp:443
* ssh tcp:22 (5522 in my house)
* imaps tcp:993 submission tcp:587 smtp tcp:25
which of these rules...

a)  
pass in on $onboard_if proto udp from 128.255.76.0/23 to any port 514

b)  
pass in on $onboard_if proto tcp from any to any port {993,587,25} keep state

c)  
pass in on $onboard_if proto tcp from 128.255.0.0/16 to any port {80,443} keep state

d)  
pass in quick on $onboard_if proto tcp from workstation1 to server.test.uiowa.edu port {22,5522} keep state
fun pf tricks

* kill states for networks or hosts

```bash
# pfctl -k 192.168.1.0/24 -k \
172.16.0.0/16
```

* filter based on OS fingerprinting (!?)

```
pf.conf:
block in on $ext_if from any os
"Windows 2000"
```

* ...or any other fingerprint, like nmap
wrap-up

* help is available
* GUI tools will catch up
* any remaining questions?