

Re: ipfw uid/gid to match listening TCP sockets?

Re: ipfw uid/gid to match listening TCP sockets?

Source: <http://unix.derkeiler.com/Mailing-Lists/FreeBSD/net/2008-04/msg00071.html>

- *From:* "Yar Tikhyy" <yar@xxxxxxxxxxxxxxxxxxx>
 - *Date:* Tue, 8 Apr 2008 16:43:31 +0400
-

On Tue, Apr 8, 2008 at 3:19 PM, Robert Watson <rwatson@xxxxxxxxxxx> wrote:

On Mon, 7 Apr 2008, Yar Tikhyy wrote:

Our ipfw currently doesn't seem to match this host's traffic by uid/gid if

the traffic goes to a listening TCP socket. E.g., if one tries to allow passive data connections to a local anonymous FTP server as follows, it won't work:

```
ipfw add 10000 allow tcp from any to me dst-port 49152-65535 uid
ftp in keep-state
```

This behaviour is obvious from ip_fw2.c:

```
2009 if (proto == IPPROTO_TCP) {
2010 wildcard = 0;
2011 pi = &tcbinfo;
2012 } else if (proto == IPPROTO_UDP) {
2013 wildcard = INPLOOKUP_WILDCARD;
2014 pi = &udbinfo;
2015 } else
2016 return 0;
```

I.e., it is OK for UDP to match PCBs (essentially sockets) with a wildcard foreign (remote) address, but not for TCP.

I wonder if there will be any security or whatever issues if the wildcard flag is set for TCP, too. The only peculiarity I can see now is that

Re: ipfw uid/gid to match listening TCP sockets?

listening sockets shouldn't generate outbound traffic; as soon a 3-way handshake starts, a separate PCB is created. Thus a listening socket can match inbound packets only.

Are there any other points I missed? Thanks!

None of this code really makes very much sense anyway, and is vulnerable to a number of races and semantic inconsistencies, not to mention application behavior that confuses it (such as sshd's opening forwarded sockets using a privileged credential). I'm not sure I agree with your analysis that listen sockets don't generate packets, btw: the syncache generates packets that are not yet from a specific socket, so arguably they are from the listen socket. All that said, I don't see any reason not to match listen sockets in the pcb lookup here.

Thank you for these points! Matching packets from listen sockets makes the case even simpler; then it's the matter of changing the "wildcard = 0;" to "wildcard = INPLOOKUP_WILDCARD;". At least matching listen sockets doesn't seem to break things not already broken.

Be aware that uid/gid/jail rules may become less maintainable as our TCP locking becomes more mature. We already jump through some uncomfortable hoops to keep it working, but I'm not sure how long that can go on.

I've always viewed uid/gid rules as a hack that works for now. In the long run we may want to consider an API allowing privileged apps to punch holes in the firewall in a controllable manner. Of course, the API should be agnostic of the particular firewall type. Then, e.g., ftpd(8) would be able to open its current passive data port only and to a single remote IP, and the whole port range wouldn't need to be exposed. Such holes could be handled as dynamic rules/states so that they don't stay there forever if the app crashes.

--

Yar

freebsd-net@xxxxxxxxxxxxx mailing list
<http://lists.freebsd.org/mailman/listinfo/freebsd-net>
To unsubscribe, send any mail to "freebsd-net-unsubscribe@xxxxxxxxxxxxx"