

Re: Question about tcp hash function tcp_hashfn()

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-06/msg00119.html>

- *From:* Evgeniy Polyakov <johnpol@xxxxxxxxxxxx>
 - *Date:* Thu, 1 Jun 2006 12:38:05 +0400
-

On Thu, Jun 01, 2006 at 01:11:25AM -0600, Brian F. G. Bidulock (bidulock@xxxxxxxxxxxx) wrote:

Evgeniy,

On Thu, 01 Jun 2006, Evgeniy Polyakov wrote:

On Thu, Jun 01, 2006 at 12:46:08AM -0600, Brian F. G. Bidulock (bidulock@xxxxxxxxxxxx) wrote:

Since pseudo-randomness affects both folded and not folded hash distribution, it can not end up in different results.

Yes it would, so to rule out pseudo-random effects the pseudo-random number generator must be removed.

You are right that having test with 2^{48} values is really interesting, but it will take ages on my test machine :)

Try a usable subset; no pseudo-random number generator.

I've run it for 2^{30} – the same result: folded and not folded Jenkins hash behave the same and still both results produce exactly the same artifacts compared to XOR hash.

But not without the pseudo-random number generation... ?

How can I obtain $(2^{30}) * 6$ bytes of truly random bytes?

Re: Question about tcp hash function tcp_hashfn()

Btw, XOR hash, as completely stateless, can be used to show how Linux pseudo-random generator works for given subset – it's average of distribution is very good.

But its distribution might auto-correlate with the Jenkins function.
The only way to be sure is to remove the pseudo-random number generator.

Just try incrementing from, say, 10.0.0.0:10000 up, resetting port number to 10000 at 16000, and just incrementing the IP address when the port number wraps, instead of pseudo-random, through 2^{30} loops for both.
If the same artifacts emerge, I give in.

I've run it with following source ip/port selection algo:

```
if (++sport == 0) {  
  saddr++;  
  sport++;  
}
```

Starting IP was 1.1.1.1 and sport was 1.
Destination IP and port are the same 192.168.0.1:80

Jenkins hash started to show different behaviour:
it does not have previous artefacts, but instead it's dispersion is much wider than in XOR case.

With following ip/port selection algo:

```
if (++sport == 0) {  
  //saddr++;  
  sport += 123;  
}
```

I see yet another jenkins artefacts, but again different from previous two.

But each time both folded and not folded hashes behave exactly the same.

Can you show the same artifacts for jenkins_3word?

What should be used as starting point there?
If I use 0 it is the same as jhash_2words().
If I use 123123 – artefacts are the same, just slightly shifted (I tested only the latest test above though).

Looking into the code we can see that jhash_2words() is jhash_3words() with zero "C" value, so it will show the same nature.

Re: Question about tcp hash function tcp_hashfn()

Re: Question about tcp hash function tcp_hashfn()

--

Evgeniy Polyakov

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>