*Complexity Attacks on **ipfw***

David Malone <dwmalone@freebsd.org>
Josh Tobin <rjt@maths.tcd.ie>

20 September 2009

Complexity Attaks

- Working on some code — too slow.
- You implement some clever algorithm — nice and fast.
- Algorithm Performance: *average* and *worst case*.
- E.g. Quicksort.
- Who controls what performance you get?

See "Denial of Service via Algorithmic Complexity Attacks" by Scott A Crosby and Dan S Wallach in 2003 Usenix Security Symposium.

Hash Table

Lookup scheme to avoid cost of searching full list.

`carrot zuchinni ...` `...apricot apple`

becomes:

a `apricot, apple`

b `banana`

c `carrot`

$.$ `...`

z `zuchinni`

Hash function $h(x)$, XOR typical. Cost: $O(N) \rightarrow O(N/H)$.

Algorithmic Attacks

Suppose attacker controls keys.

a `abduce, abducens, abducent, abduct, abduction, abductor` ...

b

$.$

z

Attacker finds x_i so that $h(x_1) = h(x_2) = \ldots = h(x_i)$.

Crosby and Wallach

Bro XOR based hash to store state. Dual core Pentium drop traffic using 30s of 160kbps or 6 minutes of 16kbps.

Perl Hash used for (uh) Hashes. Insert 90k short strings. Usually took under 2s. With crafted input took almost 2 hours.

Also attacks on directory cache, Python, GLIB, Java, ...
Related attacks on browsers, web servers, ...

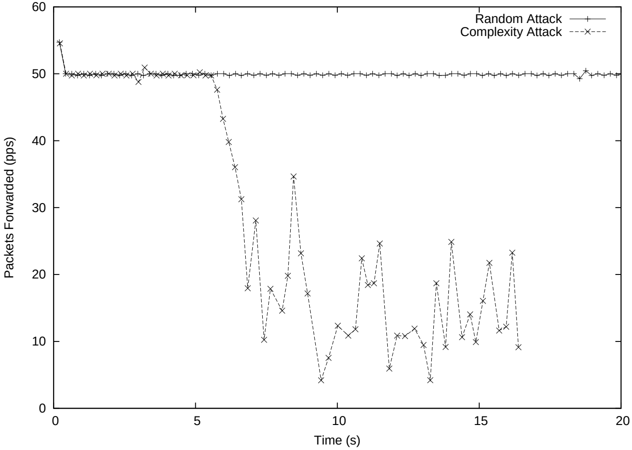
Fixes?

- Don't use hashes, choose good worst case performance.
- — can be hard to fit into existing code.
- Use cryptographic hashes.
- — output truncated
- Use keyed hash.
- — sometimes use misnomer "universal".

ipfw Flow Lookup

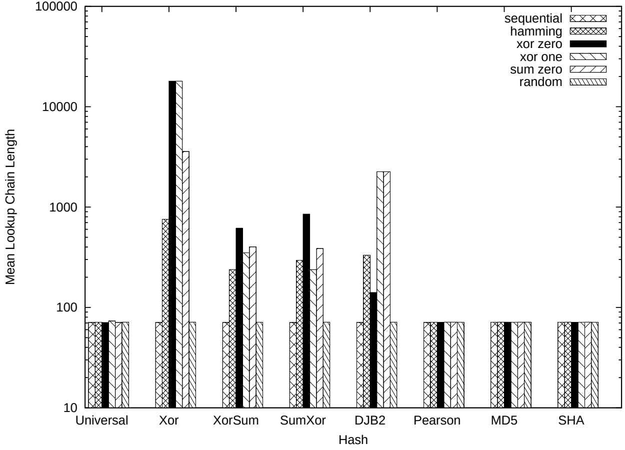
- Long standing firewall in FreeBSD.
- About 4.0 it grew stateful features.
- Lookup single flow by tuple (src IP, dst IP, src port, dst port).
- Uses hash table as optimisation.
- For IPv4 96 bits of input.
- For IPv6 288 bits of input.
- Note inexact flow matching different!

Demonstration attack

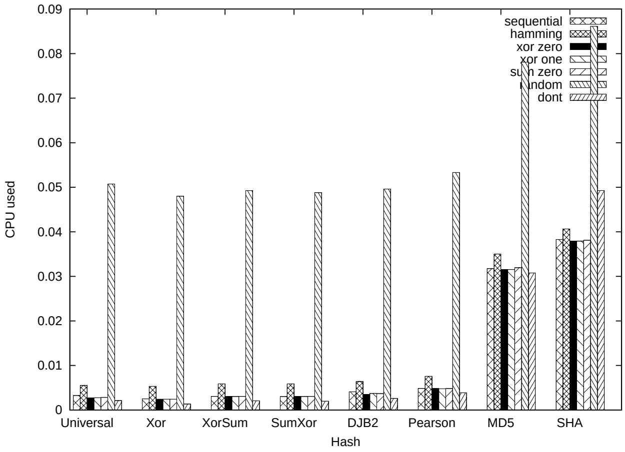


Xor	$h \leftarrow 0$	
	foreach $(byte[i])$	$h \leftarrow h \oplus byte[i]$
	return h	
DJB2	$h \leftarrow 5381$	
	foreach $(byte[i])$	$h \leftarrow 33 * h + byte[i]$
	return h	
XorSum	$h \leftarrow 0$	
	foreach $(byte[i])$	$h \leftarrow h + (byte[i] \oplus K[i])$
	return h	
SumXor	$h \leftarrow 0$	
	foreach $(byte[i])$	$h \leftarrow h \oplus (byte[i] + K[i])$
	return h;	
Universal	$h \leftarrow 0$	
	foreach $(byte[i])$	$h \leftarrow h + K[i] * byte[i]$
	return $h \mod 65537$	
Pearson	$h_1 \leftarrow h_2 \leftarrow 0$	
	foreach $(byte[i])$	$h_1 \leftarrow T_1[byte[i] \oplus h_1]$
	$h_2 \leftarrow T_2[byte[i] \oplus h_2]$	
	return $h_1 + h_2 * 256$	
MD5	return two bytes of MD5(bytes)	
SHA	return two bytes of SHA(bytes)	

Hash Chain Length



CPU Cost



sequential
hamming
xor zero
xor one
dont
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
CPU used
Universal
Xor
XorSum
SumXor
DJB2
Pearson
MD5
SHA
Hash

Other options

Don't need to use hash.

Tree Use lexical order to insert into tree.

Red/Black Tree Tree balanced by colouring.

Splay Tree Moves frequently accessed to top.

Treap Tree balanced using random heap.

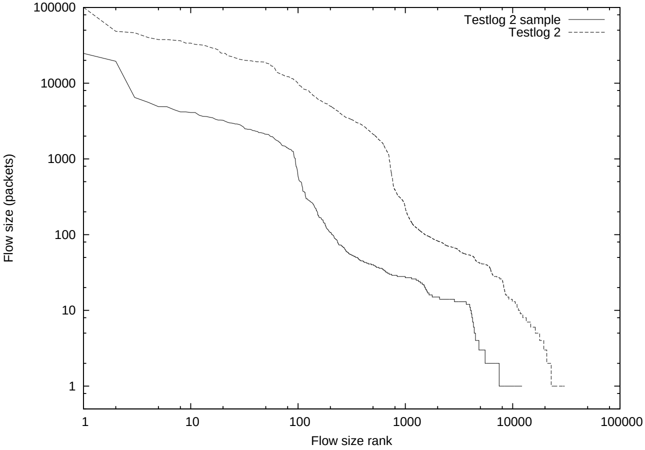
Tree is baseline (and subject to attack). Others are not (obviously) subject to attack.

Design Aims/Method

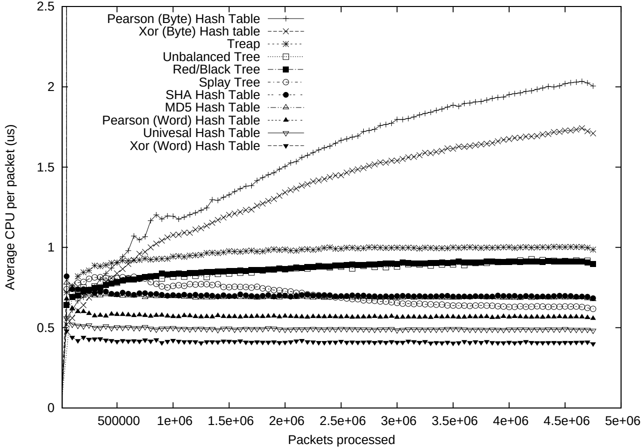
Want flow lookup to:

- Should perform OK under typical traffic.
- Should not degrade badly under attack.
- . . . typical performance depends on keys.
- . . . collect trace of traffic,
- . . . assess using pcap framework,
- . . . check performance in kernel.

Traffic Trace

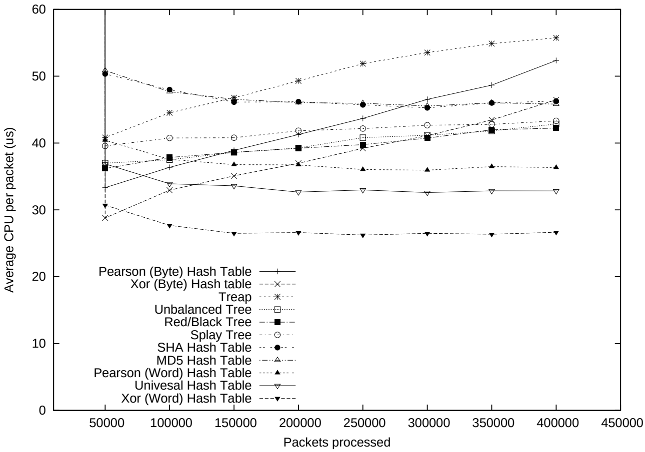


Big CPU



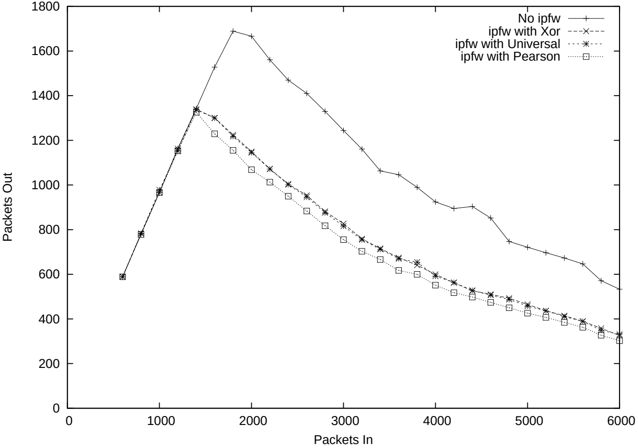
Pearson (Byte) Hash Table
Xor (Byte) Hash table
Treap
Unbalanced Tree
Red/Black Tree
Splay Tree
SHA Hash Table
MD5 Hash Table
Pearson (Word) Hash Table
Univesal Hash Table
Xor (Word) Hash Table
Average CPU per packet (us)
Packets processed
0
0.5
1
1.5
2
2.5
500000
1e+06
1.5e+06
2e+06
2.5e+06
3e+06
3.5e+06
4e+06
4.5e+06
5e+06

Small CPU



Average CPU per packet (us)
Packets processed
60
50
40
30
20
10
0
50000
100000
150000
200000
250000
300000
350000
400000
450000
Pearson (Byte) Hash Table
Xor (Byte) Hash table
Treap
Unbalanced Tree
Red/Black Tree
Splay Tree
SHA Hash Table
MD5 Hash Table
Pearson (Word) Hash Table
Univesal Hash Table
Xor (Word) Hash Table

Peak Forwarding



Conclusion

- Take care choosing algorithms.
- XOR doesn't look so good.
- Universal hash actually pretty cheap.
- Looking at attacking some hashes:
- ...Simple Pearson.
- ...RSS Hash on Ethernet Cards.