NAME

syncache, **syncookies** - sysctl(8) MIBs for controlling TCP SYN caching

SYNOPSIS

sysctl net.inet.tcp.syncookies
sysctl net.inet.tcp.syncookies_only

sysctl net.inet.tcp.syncache.hashsize
sysctl net.inet.tcp.syncache.bucketlimit
sysctl net.inet.tcp.syncache.cachelimit
sysctl net.inet.tcp.syncache.rexmtlimit
sysctl net.inet.tcp.syncache.count
sysctl net.inet.tcp.syncache.see_other

DESCRIPTION

The **syncache** sysctl(8) MIB is used to control the TCP SYN caching in the system, which is intended to handle SYN flood Denial of Service attacks.

When a TCP SYN segment is received on a port corresponding to a listen socket, an entry is made in the **syncache**, and a SYN,ACK segment is returned to the peer. The **syncache** entry holds the TCP options from the initial SYN, enough state to perform a SYN,ACK retransmission, and takes up less space than a TCP control block endpoint. An incoming segment which contains an ACK for the SYN,ACK and matches a **syncache** entry will cause the system to create a TCP control block with the options stored in the **syncache** entry, which is then released.

The **syncache** protects the system from SYN flood DoS attacks by minimizing the amount of state kept on the server, and by limiting the overall size of the **syncache**.

Syncookies provides a way to virtually expand the size of the **syncache** by keeping state regarding the initial SYN in the network. Enabling **syncookies** sends a cryptographic value in the SYN,ACK reply to the client machine, which is then returned in the client’s ACK. If the corresponding entry is not found in the **syncache**, but the value passes specific security checks, the connection will be accepted. This is only used if the **syncache** is unable to handle the volume of incoming connections, and a prior entry has been evicted from the cache.

Syncookies have a certain number of disadvantages that a paranoid administrator may wish to take note of. Since the TCP options from the initial SYN are not saved, they are not applied to the connection, precluding use of features like window scale, timestamps, or exact MSS sizing. As the returning ACK establishes the connection, it may be possible for an attacker to ACK flood a machine in an attempt to create a connection. While steps have been taken to mitigate this risk, this may provide a way to bypass firewalls which filter incoming segments with the SYN bit set.

To disable the **syncache** and run only with **syncookies**, set *net.inet.tcp.syncookies_only* to 1.

The **syncache** implements a number of variables in the *net.inet.tcp.syncache* branch of the sysctl(3) MIB. Several of these may be tuned by setting the corresponding variable in the loader(8).

hashsize	Size of the **syncache** hash table, must be a power of 2. Read-only, tunable via loader(8).
bucketlimit	Limit on the number of entries permitted in each bucket of the hash table. This should be left at a low value to minimize search time. Read-only, tunable via loader(8).
cachelimit	Limit on the total number of entries in the **syncache**. Defaults to (*hashsize* x *bucketlimit*), may be set lower to minimize memory consumption. Read-only, tunable via loader(8).
rexmtlimit	Maximum number of times a SYN,ACK is retransmitted before being discarded. The default of 3 retransmits corresponds to a 45 second timeout, this value may be increased depending on the RTT to client machines. Tunable via sysctl(3).
count	Number of entries present in the **syncache** (read-only).
see_other	If set to true value, all **syncache** entries will be visible via *net.inet.tcp.pcblist* sysctl, or via netstat(1), ignoring all of security(7) UID/GID, jail(2) and mac(4) checks. If turned off, the visibility checks are enforced. However, extra ucred(9) referencing is required on every incoming SYN packet processed. The default is off.

Statistics on the performance of the **syncache** may be obtained via netstat(1), which provides the following counts:

syncache entries added	Entries successfully inserted in the **syncache**.
retransmitted	SYN,ACK retransmissions due to a timeout expiring.
dupsyn	Incoming SYN segment matching an existing entry.
dropped	SYNs dropped because SYN,ACK could not be sent.
completed	Successfully completed connections.

bucket overflow	Entries dropped for exceeding per-bucket size.
cache overflow	Entries dropped for exceeding overall cache size.
reset	RST segment received.
stale	Entries dropped due to maximum retransmissions or listen socket disappearance.
aborted	New socket allocation failures.
badack	Entries dropped due to bad ACK reply.
unreach	Entries dropped due to ICMP unreachable messages.
zone failures	Failures to allocate new **syncache** entry.
cookies received	Connections created from segment containing ACK.

SEE ALSO

netstat(1), jail(2), mac(4), tcp(4), security(7), loader(8), sysctl(8), ucred(9)

HISTORY

The existing **syncache** implementation first appeared in FreeBSD 4.5. The original concept of a **syncache** originally appeared in BSD/OS, and was later modified by NetBSD, then further extended here.

AUTHORS

The **syncache** code and manual page were written by Jonathan Lemon <*jlemon@FreeBSD.org*>.