W A Simpson
DayDreamer

Intended status: Informational

7 April 2011

TCP Cookie Transactions (TCPCT) Sockets Application Program Interface (API) draft-simpson-tcpct-api-04

Abstract

TCP Cookie Transactions (TCPCT) [RFC6013] deter spoofing of connections and prevent resource exhaustion, eliminating Responder (server) state during the initial handshake. The Initiator (client) has sole responsibility for ensuring required delays between connections. The cookie exchange may carry data, limited to inhibit amplification and reflection denial of service attacks.

This document provides a sockets Application Program Interface (API) to support "basic" and "advanced" TCPCT applications. TCP/IP applications written using the sockets API have enjoyed a high degree of portability.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is

at http://datatracker.ietf.org/drafts/current.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

Table of Contents

1. Introduction

This specification is based upon prior work product by the author for a Linux implementation, and previous internal documents Copyright (C) William Allen Simpson (2009-2011).

1.1. Terminology

The key words "MAY", "MUST, "MUST NOT", "OPTIONAL", "RECOMMENDED", "REQUIRED", "SHOULD", and "SHOULD NOT" in this document are to be interpreted as described in [RFC2119].

2. sysctl

tcp_cookie_size - INTEGER
   Default: 16. Maximum: 16. System default size of TCP Cookie Transactions (TCPCT) option, that may be overridden by the TCPCT socket option. Values greater than the maximum (16) are interpreted as the maximum. Values greater than zero and less than the minimum (8) are interpreted as the minimum. Odd values are interpreted as the next even value.

When tcp_cookie_size is 0, the TCP stack will not send the Cookie Option, and any incoming Cookie Option will be silently discarded.

When tcp_cookie_size is greater than 0, the TCP stack will always send the Cookie Option, and any incoming Cookie Option will be processed. This is most useful for single purpose systems, such as DNS or HTTP servers.

Either case may be modified for each socket or connection.

tcp_syn_data_limit - INTEGER
   Default: 496. Maximum: 496. System maximum amount of data transmitted with the <SYN>.

tcp_syn_ack_data_limit - INTEGER
   Default: 80. Maximum: 1220. System maximum amount of data transmitted with the <SYN,ACK(SYN)>. As a matter of security policy, keep the initial setting small for most systems to avoid their use in amplification denial of service attacks.

Note that usage of <SYN>-data requires the socket option.

As a matter of policy, these limits MAY be considerably smaller.

3. Socket Option

```
#define TCP_COOKIE_TRANSACTIONS
```

When this symbol is defined, the TCP Cookie Transactions (TCPCT) socket option is available.

3.1. struct tcp_cookie_transactions

```
/* for TCP_COOKIE_TRANSACTIONS (TCPCT) socket option */
#define TCP_COOKIE_MIN           8              /*  64-bits */
#define TCP_COOKIE_MAX          16              /* 128-bits */
#define TCP_COOKIE_PAIR_SIZE    (2*TCP_COOKIE_MAX)

/* Flags for both getsockopt and setsockopt */
#define TCPCT_IN_ALWAYS         (1 << 0)
#define TCPCT_OUT_NEVER         (1 << 1)

#define TCPCT_IN_DATA           (1 << 2)
#define TCPCT_OUT_DATA          (1 << 3)
/* reserved for future use: bits 4 .. 6 */

#define TCPCT_EXTEND            (1 << 7)
#define TCPCT_RETAIN            (1 << 15)

/* Extended Option flags for both getsockopt and setsockopt */
#define TCPCT_EXTEND_SIZE       (0x7)           /* mask */
#define TCPCT_EXTEND_TS32       (0x1)           /* default */
#define TCPCT_EXTEND_TS64       (0x2)
#define TCPCT_EXTEND_TS128      (0x4)

/* TCP_COOKIE_TRANSACTIONS socket option header */
struct tcp_cookie_transactions {
        uint16_t        tcpct_flags;
        uint8_t         tcpct_extended;
        uint8_t         tcpct_cookie_desired;
        uint16_t        tcpct_s_data_desired;
        uint16_t        tcpct_used;
};
```

The structure is 64 bits.

tcpct_flags

TCPCT_IN_ALWAYS
Default: 0 (off). Silently discard any incoming <SYN> or <SYN,ACK(SYN)> that is missing the Cookie option.

TCPCT_OUT_NEVER
Default: 0 (off). Refuse to send (override) the Cookie option. This supercedes any other settings to the contrary.

TCPCT_IN_DATA

getsockopt()
Indicates <SYN> or <SYN,ACK(SYN)> data has been received.

setsockopt()
Ignored.

TCPCT_OUT_DATA

getsockopt()
Indicates <SYN> or <SYN,ACK(SYN)> data has been sent on the connection.

When tcpct_s_data_desired is zero, constant data was sent.

setsockopt()
Indicates <SYN> or <SYN,ACK(SYN)> constant data MAY be sent; tcpct_used contains the total number of bytes.

Do not wait for additional data before sending; tcpct_s_data_desired MUST be zero.

Setting the data value only has utility for a Responder. + Initiators MUST use tcpct_s_data_desired instead.

TCPCT_EXTEND
Indicates tcpct_extended field contains a non-default value.

TCPCT_RETAIN +
When this symbol is defined, the optional TCPCT Rapid Restart + [TCPCTRR] feature is available. +

Default: 0 (off). Indicates TCPCT TCB SHOULD be retained for + rapid restart.

Unused bits MUST be zero.

tcpct_extended

TCPCT_EXTEND_SIZE
Mask for Size field in Extended Option. If Size is set to an

unrecognized value (or zero), the TCP Timestamps Extended Option MUST NOT be sent, and MUST be ignored upon receipt.

TCPCT_EXTEND_TS32
Use 32-bit timestamps extension (default).

TCPCT_EXTEND_TS64
Use 64-bit timestamps extension.

TCPCT_EXTEND_TS128
Use 128-bit timestamps extension.

If the TCPCT_EXTEND flag is not set (default), this field MUST be ignored, but SHOULD be set to either the default value or zero.

tcpct_cookie_desired
Values: 0 (default), 8, 10, 12, 14, 16. Send the Cookie option with the <SYN> or <SYN,ACK(SYN)>.

Whenever this field is zero, use tcp_cookie_size instead.

tcpct_s_data_desired
Default: 0. The maximum amount of data transmitted with the <SYN> (up to tcp_syn_data_limit) or the <SYN,ACK(SYN)> (up to tcp_syn_ack_data_limit).

Whenever this field is non-zero, wait for data before sending. Unlike tcpct_cookie_desired, this field MUST be set explicitly; there is no system value.

tcpct_used
Default: 0. Number of bytes used in the variable length data value.

This header is optionally followed by variable length data.

3.2. struct tcpct_full and struct tcpct_half

```
/* TCP_COOKIE_TRANSACTIONS cookie values */
struct tcpct_full {
        struct tcp_cookie_transactions soh;
        uint8_t         tcpct_value[TCP_COOKIE_PAIR_SIZE];
};


struct tcpct_half {
        struct tcp_cookie_transactions soh;
        uint8_t         tcpct_value[TCP_COOKIE_MAX];
};
```

The structures are a multiple of 64 bits. These are provided for convenience, such as stack allocation.

tcpct_value

getsockopt()
: The current cookie or cookie pair value; tcpct_used contains the total number of bytes. Use tcpct_full.

  For an established connection, tcpct_cookie_desired will contain the cookie size.

setsockopt()
: The constant cookie or cookie pair value; tcpct_used contains the total number of bytes. Use tcpct_full or tcpct_half as appropriate.

  TCPCT_OUT_DATA is not set (MUST be zero). Any previously specified constant data is discarded.

  To distinguish a maximal cookie from a minimal cookie pair, tcpct_cookie_desired MUST equal tcpct_used or half of tcpct_used.

  Setting the cookie value only has utility for an Initiator, as receipt of an Initiator cookie and calculation of the response will replace the cookie pair in the Responder socket.

Unused bytes MUST be zero.

3.3. more ...

Acknowledgments

Andre Broido informally described utilizing cookies for Transport Layer Security (TLS) session identifiers, in place of the [RFC5077] ticket. Rapid TLS session resumption would improve both latency and privacy, but is beyond the scope of this specification.

IANA Considerations

This specification registers two new TCP options. These (previously unpublished) options were specifically selected to avoid conflicts, and IANA was advised in advance.

TCP Cookie Option
Kind: 31
Length: variable.

TCP Timestamps Extended Option
Kind: 32
Length: 4.

Notes: Kind 31 is a prime number, particularly appropriate for a security enhancement. Kind 32 is a multiple of TCP Timestamps Option (Kind 8); Kind 16 was already registered.

Security Considerations

First and foremost, the cookie exchange improves operational security for vulnerable servers against flooding attacks. All other semantics are subordinate.

TCPCT provides a number of new security mechanisms that need to be accessible to applications.

Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, March 1997.

[RFC6013] Simpson, W. A., "TCP Cookie Transactions (TCPCT)", January 2011.

[TCPCTRR] Simpson, W. A., "TCPCT Rapid Restart", March 2011. http://tools.ietf.org/html/draft-simpson-tcpct-rr

Informative References

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", January 2008.

Author's Address

Questions about this document can be directed to:

William Allen Simpson
DayDreamer
Computer Systems Consulting Services
1384 Fontaine
Madison Heights, Michigan 48071

William.Allen.Simpson@Gmail.com