PERRY METZGER, WILLIAM ALLEN SIMPSON, AND PAUL VIXIE

# improving TCP security with robust cookies

Perry E. Metzger is the managing partner of Metzger, Dowdeswell & Co. and is also pursuing a doctorate in computer science at the University of Pennsylvania. He desperately needs more sleep.

*perry@piermont.com*

William Allen Simpson is a very independent consultant, involved in design, implementation, and operation of Internet routing, network security, network protocols, wireless networking, game networking, real-time data collection and distribution, and many other projects for over 30 years.

*William.Allen.Simpson@GMail.com*

Paul Vixie took over BIND maintainance after Berkeley gave it up in 1989. He rewrote it (BIND8, 1995) and then hired other people to rewrite it again (BIND9, 2000). He has recently hired a new team to rewrite it yet again (BIND10, 201?). Between BIND releases, Paul founded the first neutral commercial Internet exchange (PAIX, 1998) and spent a small fortune fighting lawsuits by spammers (MAPS, 1998).

*vixie@isc.org*

Some men dream of fortunes, others dream of cookies.

—*fortune cookie*

**THERE'S AN IMPENDING CRISIS, DRIVEN** by the deployment of Domain Name System (DNS) Security (DNSSEC). DNS responses no longer fit in small UDP datagrams and are subsequently repeated in TCP sessions. We urgently need to robustly handle high rates of short-lived transactional TCP traffic and seamlessly segue to longer-lived sessions. TCP Cookie Transactions solve these problems and some denial-of-service attacks against TCP as well.

Current TCP implementations store enormous amounts of internal state for every connection. Heavily loaded servers can run out of memory and other resources simply by receiving too many connections (or bogus connection attempts) too quickly. TCP Cookie Transactions (TCPCT) deter spoofing of client connections and prevent server resource exhaustion by eliminating the need to maintain server state during establishment and after termination of connections. The TCPCT cookie exchange itself may optionally carry <SYN> data, limited in size to inhibit denial-of-service (DoS) attacks.

## Motivation

Common DNSSEC-signed responses are as long as 1749 bytes. During key rollover, the response could be more than twice that size, much larger than the default UDP data size of 512 bytes.

Large DNS replies over UDP permit an attacker to amplify a denial-of-service attack. By spoofing a DNS request from a victim's IP address, an attacker can turn relatively short queries over a low bandwidth connection into a far more devastating amplification attack. That's potentially more potent than a generic attack, as operators cannot filter root server responses. Currently, only 2% of DNS root server queries over UDP are legitimate [33].

DNSSEC over UDP results in multiple IP fragments where the UDP headers and port numbers are only present in the first fragment. Badly implemented middleboxes [6]—such as stateless firewalls and network address translators (NAT) [28]—either drop all the fragments or pass the first and block the rest.

A horrific number of badly implemented middleboxes rewrite DNS over UDP messages according to local policy. These middleboxes assume that packets are not fragmented. Such middleboxes are likely to remain in place for many years.

UDP has no reliable signal that large datagrams won't work. Often the only symptom is a timeout, without any hint about which of the many possible problems occurred.

The burden is on DNS resolvers to try a protocol with less interference from middleboxes. Therefore, DNS resolvers repeat the same query over TCP.

Figure 1 shows that standard TCP creates server state immediately and retains it after the connection has closed during the TCP TIME-WAIT interval (usually 4 minutes).

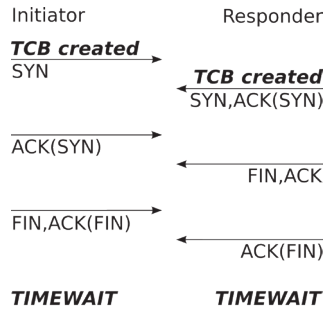


**FIGURE 1: WHEN A ◄SYN► IS RECEIVED, CURRENT SERVER IMPLEMENTATIONS CREATE STATE (THE TCB) AND MAINTAIN THAT STATE UNTIL THE TIME-WAIT INTERVAL HAS EXPIRED (USUALLY 4 MINUTES).**

Unfortunately, existing traffic patterns indicate that repeating most DNSSEC root UDP queries again over TCP would dramatically increase server load. After DNSSEC deployment in one major top-level domain, a 600% increase in TCP requests was reported [32] at the North American Network Operators Group (NANOG) meeting of June 2009.

To avoid overload, some operators are turning off the TCP port for DNS. That violates underlying DNS protocol expectations [2]. The inability to use TCP after missing or truncated UDP responses will prevent successful DNSSEC deployment.

TCP Cookie Transactions (TCPCT) permit TCP to be used in place of UDP for high-transaction-rate services without burdening servers. Operators can mitigate load by selective rejection of connection attempts without the Cookie option. Moreover, using the cryptologically secure robust cookie mechanism instead of UDP prevents the exploitation of amplification and fragmentation DoS attacks.

## Robust Cookies

In 1994, Phil Karn described a mechanism to avoid accumulating server state during an initial protocol handshake. The client sends an opaque anti-clogging token (a "cookie"). The server responds to each communication attempt by issuing its own cookie that is dependent on the client cookie, and it retains no state about the attempt.

The client returns this pair of cookies to the server, demonstrating a complete communications path. If the client fails to reply, the server has no state to free.

Karn and Simpson set forth explicit design criteria:

> The computing resources themselves must also be protected against malicious attack or sabotage. . . . These attacks are mitigated through using time-variant cookies, and the elimination of receiver state during initial exchanges of the protocol. [15, pp. 2–3]
>
> It MUST NOT be possible for anyone other than the issuing entity to generate cookies that will be accepted by that entity. This implies that the issuing entity will use local secret information in the generation and subsequent verification of a cookie. [15, p. 12; 16, p. 19]
>
> The Responder secret value that affects its cookies MAY remain the same for many different Initiators. However, this secret SHOULD be changed periodically to limit the time for use of its cookies (typically each 60 seconds). [16, p. 20]

This use of the term *cookie* should not be confused with other uses of "cookie" or "magic cookie," such as by HTTP or X Window systems, and other security protocol attempts [27]. Each of these is missing one or more of the requirements: (1) eliminating responding server state; (2) using a local secret; (3) having a time limit.

## Previous Papers and Proposals

Over the past 35 years, hundreds (perhaps thousands) of articles, papers, and reports have described network attacks using TCP: address and port spoofing, amplification, fragmentation, resource exhaustion, and others less commonly publicized [12]. Various incremental approaches have been proposed.

T/TCP [4] permits lightweight TCP transactions for applications that traditionally have used UDP. However, T/TCP has unacceptable security issues [13, 26].

By September 1996, the long anticipated DoS attacks in the form of TCP SYN floods were devastating popular (and unpopular) servers and sites. Phil Karn informally mentioned adapting anti-clogging cookies to TCP. Perry Metzger proposed adding Karn's cookies as part of a "TCP++" effort [22], and two years later as part of a "TCPng" discussion [23].

Daniel J. Bernstein implemented "SYN cookies," small cookies embedded in the TCP SYN initial sequence number. This technique was exceptionally clever, because it did not require cooperation of the remote party and could be deployed unilaterally. However, SYN cookies can only be used in emergencies; they are incompatible with most TCP options. As there is insufficient space in the sequence number, the cookie is not considered cryptologically secure. The SYN cookie mechanism remains inactive until the system is under attack, and thus is not well tested in operation. Because of these deficiencies, SYN cookies were not accepted for publication in the Internet Engineering Task Force (IETF) RFC series until recently [7].

In 1999, Faber, Touch, and Yue [9] proposed using an option to negotiate the party that would maintain TIME-WAIT state. This permits a server to entirely eliminate state after closing a connection.

In 2000, the Stream Control Transmission Protocol (SCTP) [29] was published with a mechanism partially based on Karn's ideas. There have been a number of barriers to deployment of SCTP [15].

In 2006, the Datagram Congestion Control Protocol (DCCP) [18] was published with a mechanism analogous to SYN cookies.

Medina, Allman, and Floyd [21] found that the vast majority of modern TCP implementations correctly handle unknown TCP options passing through middleboxes. A new TCP option sent in <SYN> and returned in <SYN,ACK(SYN)> will reliably indicate that both parties understand the extension. But it is still prudent to follow the [RFC 793] "general principle of robustness: be conservative in what you do, be liberal in what you accept from others."

## Solving <SYN> Spoofing

The initial TCP <SYN> exchange is vulnerable to forged IP addresses, predictable ports, and discoverable sequence numbers [25]. A complete fix requires that IP sources be checked as they enter the provider network, ensuring that they match those assigned to the provider's customers. Unfortunately, this ingress-filtering best current practice [11] is not widely enforced, and source address forged attacks continue at growing rates.

TCP Cookie Transactions (TCPCT) bolster the defense against such attacks. A cookie option is exchanged as the connection is opened. These cookies are larger and more unpredictable than addresses, ports, sequence numbers, and timestamps. They validate the connection between two parties.

Figure 2 demonstrates the TCPCT cookie exchange.

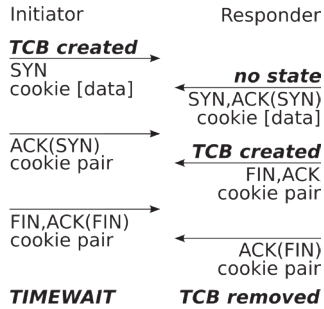


**FIGURE 2: A NEW TCP OPTION CARRIES THE COOKIES, FOLLOWED BY OPTIONAL TRANSACTION DATA IN THE INITIAL EXCHANGE (◄SYN► AND ◄SYN,ACK►).**

### AMPLIFICATION ATTACKS

TCP does not have the amplification problems of UDP [31]. A falsified source address on a <SYN> query results in a <SYN,ACK> response that is usually the same size as the query.

Unlike SCTP, TCPCT cookies are the same size in each direction, so the cookies themselves do not provide amplification.

Both T/TCP and a more recently proposed option [19] allow data to be carried on the <SYN,ACK> response, potentially allowing amplification. TCPCT enables sending this limited amount of data, as seen in Figure 2.

However, this optional feature is off by default, and is only enabled by an application on a per-port basis. Moreover, the feature may be temporarily disabled during periods of congestion and/or other resource limitations, transparently returning to default TCP behavior.

## FRAGMENTATION FAILURES

Problems with IP fragmentation have long been well known [17]. For example, IP fragmentation doesn't work reliably and, more importantly, doesn't fail reliably. UDP has no segmentation and relies entirely on unreliable IP for fragmentation support.

TCPCT requires the TCP Timestamps Option [5], which in turn requires Path MTU Discovery [24] and that the Don't Fragment (DF) bit is always set in the IP header.

## PORT PROBLEMS

Busy servers that deal with a large number of short transactions can experience port exhaustion.

For example, a Network Address Translator (NAT) maps routed hosts to its address, commonly implemented by assigning each connection to a different port [28]. When many hosts behind a NAT communicate with a common server, a port number must be assigned to each transaction. If too many transactions happen in rapid succession, the NAT will run out of port numbers.

DNS caching resolvers provide another example. When many hosts make queries through a caching resolver to a common server, a port number must be assigned to each transaction. If too many queries happen in rapid succession, the resolver will run out of port numbers. Repeated querying and aggressive retransmission [20] exacerbate these problems.

A closed TCP port must not be reused until a (TCP TIME-WAIT) timeout period has expired. If old port numbers are recycled too quickly, messages intended for the closed session cannot be distinguished from a newly opened session, appearing to be delayed duplicate transmissions.

TCPCT obviates antique duplicate transmissions by entirely eliminating server state after the <FIN> exchange. Only the client retains prior connection state for the required TCP TIME-WAIT period (see Figure 2).

TCPCT also handles reusing prior port numbers, by defining procedures that safely emulate persistent connections. Cookies and timestamps easily differentiate new sessions.

Most applications already follow the end-to-end principle and use the TCP close only as an optimization. Their data format provides all the necessary semantics for their needs.

TCPCT treats any closing <FIN> as advisory until it has been acknowledged by both parties. Like the <SYN>, each <FIN> is accompanied by the session cookies and timestamps. This inhibits a connection assassination attack with <FIN>.

## RESOURCE RECYCLING

When a TCP <SYN> arrives with an unreachable source address, the target reserves transmission control block (TCB) resources and waits for a response to its <SYN,ACK>. These are called half-open connections. An attacker can repeatedly open connections with bogus source addresses, causing a target to retain state for each half-open connection until there are no resources for legitimate connections.

Moreover, busy servers that deal with a large number of short transactions can have legitimate problems with TCB exhaustion. If the number of different clients connecting to a server locks up too much server memory, then persistent connections will make the problem worse.

Using a different strategy, attackers need only open some long-running Initiators that do nothing or do things very slowly as a different form of DoS attack. TCPCT works with the TCP User Timeout Option [8] to limit accumulation of inactive connections.

TCPCT ameliorates TCB exhaustion by eliminating server state during the <SYN> exchange and again after the <FIN> exchange. Optional <SYN> data entirely eliminates TCB state for short transactions. After the connection has closed, state is retained only by the client for the required TCP TIME-WAIT period (see Figure 2, above).

### TERMINATION TROUBLES

Perhaps the greatest security vulnerability of TCP itself is using an error indication (<RST>) to affect the operation of the protocol. This leads to TIME-WAIT assassination by antique duplicates [3], and connection assassination by third parties [10, 30].

TCPCT treats <RST> as advisory. Like the <SYN> and <FIN>, each <RST> is accompanied by the session cookies and timestamps. This inhibits a connection assassination attack with <RST>.

While cookies prevent most spoofed assassination attacks, the initial <SYN> exchange is particularly vulnerable. An attacker that can guess other fields could send a <RST> before the Responder <SYN,ACK> arrives with a proper cookie. The Initiator will not know about the attack.

Furthermore, cookies cannot defend against monkey-in-the-middle (MITM) attackers—where an attacker can record and/or reflect cookie, sequence, and timestamp values. Figure 3 demonstrates a standard TCP <SYN> assassination using <RST>.

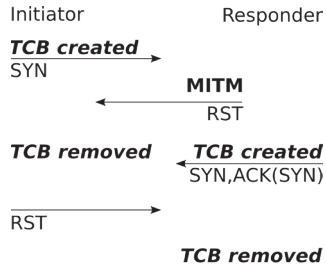


**FIGURE 3: USING AN INJECTED ◂RST▸ PACKET TO ASSASSINATE A TCP CONNECTION**

Therefore, receipt of <RST> has no effect on the operation of the protocol. All <RST> segments are merely counted [1, sec. 1.2.3]. Transmission will continue until a timeout expires [1, secs. 4.2.2.20(h), 4.2.3.5]. Arguably, this is the only substantial TCPCT change in TCP semantics.

## Summary and Exhortation

TCP Cookie Transactions (TCPCT) provide a cryptologically secure mechanism to guard against simple flooding attacks sent with bogus IP sources or TCP ports.

TCPCT entirely eliminates server state during connection establishment and after termination, and it inhibits premature closing of connections. Also, implementations may optionally exchange limited amounts of transaction data during the initial cookie exchange, reducing round trips in short transactions.

Finally, implementations may optionally rapidly recycle prior connections. For otherwise stateless applications, this transparently facilitates persistent connections and pipelining of requests over each connection, reducing network latency and host task context switching.

We expect soon to have TCPCT implementations tested and deployed in some root and TLD servers. As caching resolvers and clients are updated, the load on servers should decrease.

Gentle reader, we hope that the numerous benefits of TCPCT will inspire you to request implementation and deployment on your favorite systems.

## ACKNOWLEDGMENTS

Many thanks to Ted Faber, J. Bruce Fields, Fernando Gont, Craig Partridge, David P. Reed, Joe Touch, and other participants at the end2end-interest and namedroppers mailing lists for helpful comments on this topic.

## REFERENCES

[1] R. Braden, ed., "Requirements for Internet Hosts—Communication Layers," STD 3, RFC 1122, October 1989: http://tools.ietf.org/html/rfc1122.

[2] R. Braden, "Requirements for Internet Hosts—Application and Support," STD 3, RFC 1123, October 1989. See section 6.1.3.2: http://tools.ietf.org/html/rfc1123.

[3] R. Braden, "TIME-WAIT Assassination Hazards in TCP," RFC 1337, May 1992: http://tools.ietf.org/html/rfc1337.

[4] R. Braden, "T/TCP—TCP Extensions for Transactions—Functional Specification," RFC 1644, July 1994: http://tools.ietf.org/html/rfc1644.

[5] V. Jacobson, R. Braden, and D. Borman, "TCP Extensions for High Performance," RFC 1323, May 1992: http://tools.ietf.org/html/rfc1323. Updating: work in progress, March 4, 2009: http://tools.ietf.org/html/draft-ietf-tcpm-1323bis-01.

[6] B. Carpenter and S. Brim, "Middleboxes: Taxonomy and Issues," RFC 3234, February 2002.

[7] W. Eddy, "TCP SYN Flooding Attacks and Common Mitigations," RFC 4987, August 2007.

[8] L. Eggert and F. Gont, "TCP User Timeout Option," RFC 5482, March 2009.

[9] T. Faber, J. Touch, and W. Yue, "The TIME-WAIT State in TCP and Its Effect on Busy Servers," IEEE INFOCOM 99, pp. 1573–1584.

[10] S. Floyd, "Inappropriate TCP Resets Considered Harmful," BCP 60, RFC 3360, August 2002.

[11] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing," BCP 38, RFC 2827, May 2000.

[12] F. Gont, "Security Assessment of the Transmission Control Protocol (TCP)," February 2009: https://www.cpni.gov.uk/Docs/tn-03-09-security-assessment-TCP.pdf.

[13] C. Hannum, "Security Problems Associated with T/TCP," unpublished work in progress, September 1996: http://www.mid-way.org/doc/ttcp-sec.txt.

[14] Internet Engineering Task Force (IETF), "Intellectual Property Rights Disclosures": https://datatracker.ietf.org/ipr/.

[15] P. Karn and W. Simpson, "The Photuris Session Key Management Protocol," March 1995: draft-karn-photuris-01.txt.sp. Published as "Photuris: Design Criteria," in *Proceedings of Sixth Annual Workshop on Selected Areas in Cryptography*, LNCS 1758, (Springer-Verlag, August 1999).

[16] P. Karn and W. Simpson, "Photuris: Session-Key Management Protocol," RFC 2522, March 1999.

[17] C. Kent and J. Mogul, "Fragmentation Considered Harmful," 1987: http://www.hpl.hp.com/techreports/Compaq-DEC/WRL-87-3.pdf.

[18] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," RFC 4340, March 2006.

[19] A. Langley, "Faster Application Handshakes with SYN/ACK Payloads," work in progress, August 5, 2008: http://tools.ietf.org/html/draft-agl-tcpm-sadata-01.

[20] M. Larson and P. Barber, "Observed DNS Resolution Misbehavior," BCP 123, RFC 4697, October 2006.

[21] A. Medina, M. Allman, and S. Floyd, "Measuring Interactions Between Transport Protocols and Middleboxes," *Proceedings of the 4th ACM SIGCOMM/USENIX Conference on Internet Measurement*, October 2004: http://www.icsi.berkeley.edu/pubs/networking/tbit-Aug2004.pdf.

[22] P. Metzger, "Re: SYN floods (was: does history repeat itself?)," September 9, 1996: http://www.merit.net/mail.archives/nanog/1996-09/msg00235.html.

[23] P. Metzger, "Re: what a new TCP header might look like," May 12, 1998: ftp://ftp.isi.edu/end2end/end2end-interest-1998.mail.

[24] J. Mogul, and S. Deering, "Path MTU Discovery," RFC 1191, November 1990.

[25] R. Morris, "A Weakness in the 4.2BSD Unix TCP/IP Software," Technical Report CSTR-117, AT&T Bell Laboratories, February 1985: http://pdos.csail.mit.edu/~rtm/papers/117.pdf.

[26] route [at] infonexus [dot] com, "T/TCP vulnerabilities," *Phrack Magazine*, vol. 8, no. 53, July 8, 1998: http://www.phrack.org/issues.html?issue=53&id=6.

[27] W. Simpson, "IKE/ISAKMP Considered Harmful," USENIX *;login:*, December 1999: http://www.usenix.org/publications/login/1999-12/features/harmful.html.

[28] P. Srisuresh and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," RFC 3022, January 2001.

[29] R. Stewart, ed., "Stream Control Transmission Protocol," RFC 4960, September 2007.

[30] J. Touch, "Defending TCP against Spoofing Attacks," RFC 4953, July 2007.

[31] R. Vaughn and G. Evron, "DNS Amplification Attacks," March 17, 2006: http://www.isotf.org/news/DNS-Amplification-Attacks.pdf.

[32] D. Wessels, "DNSSEC, EDNS, and TCP," June 2009: http://www.nanog.org/meetings/nanog46/presentations/Wednesday/wessels_light_N46.pdf.

[33] D. Wessels and M. Fomenkov, "Wow, That's a Lot of Packets," *Proceedings of the Passive and Active Measurement Workshop (PAM),* April 2003: http://www.caida.org/publications/papers/2003/dnspackets/wessels-pam2003.pdf.