Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: July 21, 2015

J. Kirsch
TU Muenchen
C. Grothoff
Inria Rennes Bretagne Atlantique
J. Appelbaum
Tor Project Inc.
H. Kenn, Ed.
Microsoft Deutschland GmbH
January 17, 2015

TCP Stealth
draft-kirsch-ietf-tcp-stealth-01

Abstract

TCP servers are visible on the Internet to unauthorized clients, as the existence of a TCP server is leaked in the TCP handshake before applications have a chance to authenticate the client.

We present a small modification to the initial TCP handshake that allows TCP clients to replace the TCP ISN in the TCP SYN packet with an authorization token. Based on this information, TCP servers may then chose to obscure their presence from unauthorized TCP clients.

This RFC documents the specific method for calculating the authorization token to ensure interoperability and to minimize interference by middleboxes. Mandating support for this method in operating system TCP/IP implementations will ensure that clients can connect to TCP servers protected by this method.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

As it has been shown in [ZMAP], it is feasible today to perform a port scan on all Internet hosts in less than an hour and specialized search engines perform such port scans regularily. At the same time, security issues in server implementations are exploited by various parties for espionage and commercial gain. Thus, it is increasingly important to minimize the visible footprint of services on Internet hosts, thereby reducing the attack surface.

Since the complete integrity of the internet infrastructure cannot be assumed, it follows that adversaries may be able to observe all traffic of an Internet host and perform man-in-the-middle attacks on traffic originating from specific clients. Furthermore, on the server side, an adversary looking for exploitable systems should be expected to have the ability to perform extensive port scans for TCP servers.

To help address this problem, we propose to standardize TCP Stealth, a stealthy port-knocking variant where an authenticator is embedded in the TCP SQN number. TCP Stealth enables authorized clients to perform a standard TCP handshake with the server, while obscuring the existence of the server from port scanners. The basic idea is to transmit an authorization token derived from a shared secret instead of a random value for the initial TCP SQN number in the TCP SYN packet. The token demonstrates to the server that the client is authorized and may furthermore protect the integrity of the beginning of the TCP payload to prevent man-in-the-middle attacks. If the token is incorrect, the operating system pretends that the port is closed. Thus, the TCP server is hidden from port scanners and the TCP traffic has no anomalies compared to a normal TCP handshake.

The TCP MD5 Signature Option defined in RFC 2385 defines a similar mechanism, except that RFC 2385 does not work in the presence of NATs (RFC 1631) and visibly changes the TCP wire protocol, and can thus be easily detected.

While TCP Stealth does not change the TCP wire protocol, the specific method for calculating the authorization token must be consistent across Internet hosts and their TCP/IP implementations to ensure interoperability. By embedding the port knocking logic into the TCP/ IP implementation of an operating system, we minimize the possibility of detecting hidden services via timing attacks, and avoid the pitfalls of applications trying to re-implement TCP in user-space. Implementors MUST make sure that the response to a connection request with wrong ISN value does not differ in any way from the response to a connection request to a closed port.

2. Terminology and Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

We use "TCP Stealth" to refer to the techniques described in this document.

The abbreviation "ISN" is used for the initial sequence number in the TCP header.

3. Description of the TCP Stealth Option

TCP Stealth implementations MUST support two variants for the TCP ISN calculation. The first variant is to only provide access authorization, the second also ensures the integrity of the first octets of the TCP payload. Which method is used depends on the application protocol, as only certain application protocols can benefit from the TCP payload protection.

Using the TCP Timestamp option (see RFC 7323, section 3.2) is recommended but optional in combination with TCP Stealth. If the client does not include a timestamp in the request, the following calculations should be done using a timestamp value of zero.

Note that if the TCP Timestamp option is enabled then the TCP implementation MUST make sure that potential retransmits of TCP SYN segments carry the same value for the TSVal field as the initial SYN segment that was sent out by TCP Stealth.

3.1. 32-bit Access Authorization

For TCP Stealth without payload integrity protection, the 32 bit ISN is calculated using a single round of MD5 (the procedure explained in RFC 1321, section 3.4) using the 64-byte shared secret as the argument for the hash function and a 16 byte initialization vector IV computed as follows:

First, the 16 byte are set to the destination address, in the case of IPv4 padded with zeros. Then, (if existent) the timestamp value (in network byte order) is XORed at offsets 8 to 11, and finally the destination port (in network byte order) is XORed at offsets 12 to 13.

The resulting four 32-bit words of the MD5 hash are combined using XOR to calculate the 32-bit ISN in network byte order.

3.1.1. Test Vectors

	IPv4	IPv6
Shared secret	"Magic secret string"	"Magic secret string"
Destination IP	192.18.42.42	2001:db8::2a:2a
Destination Port	4242	4242
Timestamp	0x11223344	0x11223344
Resulting ISN	0xedea1325	0x4923a842

3.2. 16-bit Access Authorization and 16-bit Payload Protection

For TCP Stealth with payload integrity protection, the protected portion of the payload is limited to at most the first segment. Additionally, both client and server must know the exact number of bytes to be protected beforehand. The secret is prepended to the protected payload and hashed using full MD5 and the resulting eight 16-bit words of the MD5 hash are combined using XOR to a 16-bit integrity hash (IH).

Then, the 32 bit ISN is calculated using a single round of MD5 (the procedure explained in RFC 1321, section 3.4) using the 64-byte shared secret as the argument for the hash function and a 16 byte initialization vector IV computed as follows:

First, the 16 bytes are set to the destination address, in the case of IPv4 padded with zeros. The IH is XORed at offset 4 to 5. Then, (if existent) the timestamp value (in network byte order) is XORed at offsets 8 to 11, and finally the destination port (in network byte order) is XORed at offsets 12 to 13.

The resulting four 32-bit words of the MD5 hash are combined using XOR to calculate at 32-bit authenticator value (AV). The upper 16 bit of the AV are concatenated with the 16 bit of the IH to create the final 32-bit ISN in network byte order

The TCP server uses the 16 bit IH from the ISN to calculate the AV when receiving the TCP SYN packet. If the AV matches, the handshake is allowed to proceed. Then, when the TCP server receives the first segment, it checks that the included payload is of sufficient length and in combination with the shared secret hashes to the IH. If these checks fail, the connection is closed (using TCP RST).

3.2.1. Test Vectors

	IPv4	IPv6
Payload	"Protected payload goes here."	"Protected payload goes here."
Protected length	28	28
Shared secret	"Magic secret string"	"Magic secret string"
Destination IP	192.18.42.42	2001:db8::2a:2a
Destination Port	4242	4242
Timestamp	0x11223344	0x11223344
Resulting ISN	0x2153ff96	0x3ae5ff96

4. Timestamps and TCP SYN Retransmits

Implementations of TCP Stealth MUST take care to NOT modify the value of the TCP timestamp (if present) when retransmitting the TCP SYN packet. Otherwise, a different TCP timestamp would change the result of the ISN calculation and the receiver would fail to interpret the SYN packet correctly as a retransmission, which would be problematic in cases where high delay or a loss of the SYN ACK caused the retransmission.

TCP implementations in general SHOULD strongly consider preserving the original TCP timestamp value for TCP SYN retransmissions to make it impossible to detect the use of TCP Stealth by forcing TCP SYN retransmissions.

5. TCP Stealth and TCP SYN Cookies

Implementations of TCP Stealth can support the use of TCP SYN Cookies as described in RFC 4987, even in combination with content integrity protection. For this, the TCP server needs to recover the original ISN by subtracting 1 from the SQN of the first acknowledgement from the client. This way, the TCP server does not need to keep any state after receiving just the SYN packet.

6. Integration with Applications

Given operating system support, nearly every network application can be easily adapted to use TCP Stealth. Operating systems MUST use their existing facilities for setting TCP options to enable TCP clients to specify the shared secret and optionally the first octets of the TCP payload that are to be integrity protected. Similarly, for the TCP server it MUST be possible to set the shared secret and the expected number of protected octets of the TCP payload.

Furthermore, support for these options MUST be provided for both IPv4 and IPv6 and MAY be provided for other address families.

7. TCP Stealth and destination network address translation (DNAT)

As TCP Stealth relies on the destination port and address being preserved across the network, TCP Stealth cannot directly be used with hosts behind a DNAT. In case a host behind a DNAT should be protected by TCP Stealth the TCP Stealth implementation should be moved directly into the DNAT implementation.

8. Old duplicate segments

A client using the pseudo-random ISN generation of TCP Stealth without payload protection must respect TCP TIME-WAIT and use a different source port for a second connection to the same destination until sufficient time has past for old duplicate segments of the previous TCP connection to not be in the network anymore. TCP Stealth has no impact on the TCP PAWS mechanism (RFC 7323), as PAWS makes no assumptions about the ISN.

9. Middlebox Considerations

To avoid interfering with TCP Stealth, middleboxes SHOULD NOT modify TCP SQN numbers and SHOULD preserve the TCP timestamp option (if present). Recent studies [OSSIFICATION] [Knock] have shown that almost all middleboxes already satisfy these constraints.

10. IANA Considerations

None.

11. Security Considerations

Implementations using a predictable shared secret and predictable values for the TCP timestamp (such as 0 if timestamps are not used at all) may be susceptible to attacks based on ISN prediction. Thus, applications MUST be careful to not set the option with an empty or default value for the shared secret (especially in the case where the user did not to specify a shared secret). Similarly, static sequence numbers occur when TCP timestamps are disabled for each combination of secret, destination IP address and destination port.

Using predictable ISNs can be used as a vector for denial-of-service attacks on TCP (such as CVE-2005-0356). However, with TCP Stealth, only the authorized client and the providing server share the secret necessary to predict the ISN. Routers that are in a position to observe the ISN from the handshake can always perform a Denial of Service attack on TCP. TCP Stealth does not change this fundamental issue with TCP.

Middleboxes MAY modify the TCP SQN number or tamper with the TCP timestamp option. In this case, services protected by the TCP Stealth option would be unavailble to clients accessing the network from behind such a NAT. While such implementations are rare in practice, operators should be aware of the resulting possible limitations in availablilty of TCP Stealth protected services.

TCP Stealth MAY protect against malicious attacks in which security flaws of software operating above the TCP protocol are exploited. However TCP Stealth SHOULD be considered as an additional layer of security, not as a replacement of IP or higher level based authentication.

TCP Stealth does not protect against passive port scan scenarios in which an attacker can tell a closed from an open port by eavesdropping on the connection made by a legit client to a TCP Stealth protected server. TCP Stealth MAY protect against active man-in-the-middle attacks in which an attacker tries to intercept the TCP connection after the knock if TCP Stealth operates with payload integrity protection and the application layer protocol is designed in a sane way.

The protections offered by TCP Stealth are limited by the fact that the TCP SQN number field is only 32 bits. Thus, an adversary may still be able to connect to TCP Stealth protected services with low probability by chance, or high probability using brute-force methods. Thus, TCP Stealth SHOULD only be used as an additional layer of security.

12. References

12.1. Normative References

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4987] Eddy, W., "TCP SYN Flooding Attacks and Common Mitigations", RFC 4987, August 2007.

[RFC7323] Borman, D., Braden, B., Jacobson, V., and R. Scheffenegger, "TCP Extensions for High Performance", RFC 7323, September 2014.

12.2. Informative References

[Knock] Kirsch, J. and C. Grothoff, "Knock: Practical and Secure Stealthy Servers", August 2014, <https://gnunet.org/knock>.

[OSSIFICATION] Honda, M., Nishida, Y., Raiciu, C., Greenhalgh, A., Handley, M., and H. Tokuda, "Is It Still Possible to Extend TCP?", 2011, <http://doi.acm.org/10.1145/2068816.2068834>.

[ZMAP] Durumeric, Z., Wustrow, E., and J. Halderman, "ZMAP: The Internet Scanner", August 2013, <https://zmap.io/>.

Authors' Addresses

Julian Kirsch
TU Muenchen
Chair for IT-Security
Boltzmannstrasse 3
Technische Universitaet Muenchen
Garching bei Muenchen, Bayern D-85748
DE

Email: kirschju@sec.in.tum.de

Christian Grothoff
Inria Rennes Bretagne Atlantique
Equipe Decentralise
Inria Rennes Bretagne Atlantique
263 Avenue du General Leclerc
Campus Universitaire de Beaulieu
Rennes Cedex, Bretagne F-35042
FR

Email: christian@grothoff.org

Jacob Appelbaum
Tor Project Inc.

Email: jacob@appelbaum.net

Holger Kenn (editor)
Microsoft Deutschland GmbH
Konrad Zuse Strasse 1
Unterschleissheim D-85716
DE

Email: holger.kenn@cubeos.org