Privacy and Security Program
Internet-Draft
Intended status: Informational
July 29, 2016
Expires: January 30, 2017

B. Trammell
ETH Zurich

Detecting and Defeating TCP/IP Hypercookie Attacks
draft-trammell-privsec-defeating-tcpip-meta-00

Abstract

The TCP/IP stack provides protocol features that can potentially be abused by on-path attackers to inject metadata about a traffic flow into that traffic flow in band. When this injected metadata is provided by an entity with knowledge about the natural person associated with a traffic flow, it becomes a grave threat to privacy, which we term a hypercookie.

This document defines a threat model for hypercookie injection and hypercookie coercion attacks, catalogs protocol features that may be used to achieve them, and provides guidance for defeating these attacks, with an analysis of protocol features that are disabled by the proposed defeat mechanism.

The deployment of firewalls that detect and reject abuse of protocol features can help, but the relative ease of injecting metadata for attackers on path, and trivial combination of metadata injection attacks, leads to a recommendation to add cryptographic integrity protection to transport layer headers to defend against injection attacks.

tl;dr: at least with respect to metadata injection in the current Internet protocol stack, everything is ruined.

This Internet-Draft will expire on January 30, 2017.

Copyright Notice

Table of Contents

1. Introduction

This document considers a specific threat model related to the pervasive surveillance threat model defined in [RFC7624] and correlation and identification of users as defined in sections 5.2.1 and 5.2.2, respectively, of [RFC6973]. The attacker has access to the access network(s) connecting a user to the Internet, by collaborating with, coopting, or otherwise exercising influence over the user's access provider. It can see all inbound and outbound traffic from the user via that network, and can modify inbound and outbound packets to the user. The attacker would like to add metadata to the user's traffic flows in order to expose that metadata to networks the user communicates with, where it will be passively observed, and it would like this metadata to appear in layers 3 or 4, in order to be completely transparent to the application. For purposes of this analysis, we presume this metadata is a user identifier or partial user identifier. We propose a colloquial term for this type of sub-application identification: "hypercookie". This can be seen as a third-party implementation of the metadata insertion pattern described in [I-D.hardie-privsec-metadata-insertion].

The attacker is variably interested in avoiding detection of hypercookie injection techniques, and is variably interested in metadata reliability, but requires that the injected metadata not interfere with normal protocol operation, even if the exposed metadata is not used by any far endpoint.

The hypercookie injection attack is related to another, largely equivalent attack, hypercookie coercion. In this attack, the attacker requires the client endpoint to expose the hypercookie itself, and uses in-band verification techniques to determine whether the hypercookie was correctly applied, blocking traffic which does not carry it.

This document is concerned only with identification through hypercookie injection at the transport and network layers, as this is possible even when the application layer is encrypted using TLS or other encryption schemes that operate above the transport layer. Application layer hypercookie injection is out of scope, as are identification methods using traffic fingerprinting. It is also concerned only with TCP as defined, not as implemented and deployed; exploitation of other behaviors in implemented TCP stacks (e.g. as outlined in [blind-tcp-attacks] may also be used for hypercookie exposure, albeit with further risk of connection disruption.

Further, out-of-band identification methods, e.g. linking a flow's five- or six-tuple with an identifier and using some other protocol

to export this linkage, is also not considered, as it is practically impossible for users and far endpoints to detect and defeat.

The metadata injection techniques presented in this document are EMPHATICALLY NOT RECOMMENDED for use on the Internet; this document is intended to educate members of the Internet engineering community about the potential for abuse in TCP as defined and deployed.

2. Terminology

As used in this document:

- "Stateless TCP firewall" refers to a middlebox [RFC3234] that selectively drops single malformed TCP packets. A stateless TCP firewall can defeat TCP metadata injection techniques which rely on noncompliant formation of single TCP packets.

- "Stateful TCP firewall" refers to a middlebox that selectively drops TCP packets not conforming to the protocol by modeling the TCP state machine on both endpoints. A stateful TCP firewall can defeat TCP metadata injection techniques which relies on noncompliant formation of TCP packets and/or flows.

- "Split TCP proxy" refers to a middlebox which terminates a TCP connection on one its Internet-facing side and opens a separate TCP connection on the other side. Split TCP firewalls defeat most of the TCP-specific metadata injection techniques in this document.

3. General Mitigation Techniques and Related Work

The metadata injection techniques described in Section 4 share some general properties: each places data into bits in the IP or TCP header, injection of which is insignificant to the connectivity or performance of the connection between the endpoints. To some extent, this is a consequence of cleartext headers in IP and TCP and of Postel's maxim [RFC1122]. Being liberal in what one accepts leaves space between what the sender SHOULD/MUST send and what the receiver will silently ignore, and these techniques exploit that space. Changing transport stacks to fail fast and hard on the receiver side, as recommended in [I-D.thomson-postel-was-wrong] would reduce this space, but at the possible risk of connectivity instability during the transition.

TCP HICCUPS [hiccups] proposes a method for cooperative discovery and mitigation of middlebox manipulation. It uses many of the bits in the header that could also be used for metadata injection, and as

such provides a concrete implementation of fail fast and hard, mitigating TCP attacks as in Section 4.3.

The deployment of middleboxes to drop malformed packets or zero fields that may be used in hypercookie attacks may help to reduce the rate of success and therefore the incentive to perform hypercookie injection. However, this must be balanced against the cost of additional management complexity and the risk of further ossification of the Internet protocol stack through even more widespread deployment of transport-aware, stateful, packet-modifying middleboxes.

The best defense comes from evolving the stack: Widespread deployment transport protocol proposals that encrypt most or all of the transport layer headers such as QUIC, or proposals to enable generalized transport layer encapsulation and encryption such as PLUS, would effectively mitigate the TCP attacks in Section 4.3.

4. Metadata Injection Techniques

This section describes metadata injection techniques against the TCP/IP stack, separated by whether they abuse the IPv4, IPv6, or TCP protocols.

4.1. Abusing Internet Protocol features

Four attacks abuse the IPv6 header: three by injecting information into IPv6 source addresses, one abusing the IPv6 flow label.

4.1.1. Identification using EUI-64 addressing

[RFC4291] section 2.5.1 required IPv6 interface identifiers for Stateless Address Autoconfiguration (SLAAC) to be constructed using modified EUI-64 format. This leaks the hardware address of a user's terminal to the receiver and all devices along the path. Such addresses are easily recognized, as well, given the presence of the bytes 0xff and 0xfe at byte offsets 11 and 12 of the address. Though [RFC7136] deprecates the significance of the IPv6 interface identifier and [RFC4941] specifies a standard method for assigning privacy addresses when using SLAAC, these addresses may still be in use on the Internet and as such can be passively used as identifying information along the path.

When present, this technique provides 47 bits of identifying information on a per-node basis, present on each packet from the node. Access network providers cannot force the use of EUI-64 addressing; however, see Section 4.1.3 for a related technique.

The mitigation is to disable EUI-64 based SLAAC at end hosts, replacing it with [RFC4941] privacy addressing and/or DHCPv6 [RFC3315]. This is current recommended practice in any event. Both of these mitigations come with limited additional overhead and/or network management complexity.

4.1.2. Identification using DHCPv6

An attacker which runs or can influence the configuration of a DHCPv6 server from which a node gets its address can assign a source address to that node, the interface identifier part of which can contain identifying information.

When successful, this technique provides approximately 64 bits of identifying information on a per-node basis, present on each packet from the node. Access network providers can influence the use of DHCPv6 addresses, depending on access network architecture.

The mitigation is to disable DHCPv6. In situations when a user cannot practically do so without losing connectivity, this technique can be identified in some cases through an analysis of the addresses assigned to node(s) belonging to a user and determination of the persistence of the linkage between an address or addresses and a user.

4.1.3. Identification using IPv6 network address translation

An attacker which cannot influence the configuration of a DHCPv6 server can use network address translation to rewrite the interface identifier part of an address to contain identifying information.

When successful, this technique provides approximately 64 bits of identifying information on a per-node basis, present on each packet from the node.

No user-initiated mitigation is possible with the present stack. This technique can be detected by connecting to a remote host via IPv6, which can then analyze the addresses assigned to node(s) belonging to a user and determination of the persistence of the linkage between an address or addresses and a user.

4.1.4. Identification using Flow ID

[RFC6437] defines the IPv6 flow label, a 20-bit field in every IPv6 packet. It is intended to replace source and destination port in equal-cost multipath routing (ECMP) and other load distribution schemes. However, the flow label can be freely rewritten by middleboxes on path.

This technique provides up to 20 bits of identifying information per packet, with the caveat that applying different flow labels to different packets within a flow may impair transport layer performance due to reordering.

No user-initiated mitigation is possible with the present stack. Header modification detection as in [hiccups], and/or the deployment of middleboxes that monitor and/or zero the flow label may provide detection and mitigation.

4.2. Abusing legacy Internet Protocol features

One attack injects information into the IPv4 fragment ID header.

4.2.1. Fragment Identification Rewriting

[RFC6864] defines the Identification field in the IPv4 header, which is used for fragmentation and fragment reassembly. While the field is only defined when a packet is fragmented, middleboxes can freely fill identifying information into this field. [RFC6864] section 4.1 states that the value MUST be ignored by middleboxes, so it will tend to be preserved along the path assuming compliant devices.

This technique provides up to 16 bits of identifying information per packet, with a caveat that it may be difficult to implement on networks with large amounts of fragmented IPv4 traffic.

There is no user-initiated mitigation possible with deployed IPv4 stacks. Header modification detection as in [hiccups] may provide detection and mitigation

4.3. Abusing Transmission Control Protocol Features

A multitude of techniques exist to abuse TCP. These can be roughly classified into per-packet injection, where metadata can be added to header bits in each packet; and per-flow injection, where packets not part of the normal flow are generated and ignored by the receiver. Per-flow injection techniques generally provide much more space for metadata injection, and are sufficient for user identification for access control and user tracking on a per-flow basis.

4.3.1. Initial Sequence Number Rewriting

A middlebox can rewrite the initial sequence number (ISN) of flows it sees the SYN packet for, in order to place identifying information therein.

This technique provides up to 32 bits of identifying information per flow, with the caveat that it requires a stateful middlebox to translate all sequence and acknowledgment numbers on subsequent packets on the flow. It also does not work if there are other proxies which rewrite the ISN (e.g. for security, to mitigate poor randomness in 1990s era TCP stace ISN selection algorithms) on the path between the middlebox and the Internet. The identification provided by this technique also does not traverse split-TCP proxies.

Header modification detection as in [hiccups] or the aggressive deployment of split-TCP proxies can mitigate this attack. We note that the aggressive deployment of split-TCP proxies in the Internet is an undesirable solution, as it implies an acceleration and deepening of middlebox-related transport protocol ossification.

4.3.2. **Urgent Pointer Identification**

A middlebox can rewrite the urgent pointer of TCP packets without the URG flag set, in order to place identifying information therein. The urgent pointer is only intepreted when the URG flag is set, according to section 3.1 of [RFC0791]; compliant implementations will therefore ignore the urgent pointer when used in this manner.

This technique provides up to 16 bits of identifying information per packet.

Information exposed using this technique may not traverse TCP firewalls or split TCP proxies. The aggressive deployment of stateless TCP firewalls that zero the urgent pointer on all packets with the URG flag not set can mitigate this attack, at the cost of increased operational complexity and further middlebox-related transport protocol ossification.

4.3.3. **Piggybacked Experimental TCP Options**

A middlebox can piggyback an experimental TCP option onto a TCP packet with enough headroom, and place identifying information in that option. This option could even be given a IANA identifier using the ExId mechanism [RFC6994], registered with IANA on a First-Come, First-Served [RFC5226] basis, with an innocuous name, in order to deflect suspicion about its use.

Assuming a 4-byte ExId, sufficient headroom between the segment size and the path MTU, and no other TCP options on a packet, this technique can provide up to 288 bits of identifying information per packet given limitations on TCP options size. We note that this is an upper bound, and that the transparency of Internet paths to

unknown and experimental TCP options is not perfect, which reduce the applicability of this technique somewhat.

Information exposed using this technique may not traverse TCP firewalls or split TCP proxies. The aggressive deployment of stateless TCP firewalls that strip experimental options not in use on a given network can mitigate this attack. We note that some deployed TCP Fast Open [RFC7413] implementations use an experimental option, and would be affected by this mitigation. This mitigation also incurs the cost of increased operational complexity and further middlebox-related transport ossification.

4.3.4. Bare ACK Segments with Experimental TCP Options

As with the attack in Section 4.3.3, above, a middlebox could simply generate a suitable bare ACK packet within a flow, but not initiated by the sender, and place information in an experimental TCP option. The bare ACK would be processed by the receiver and the option ignored.

This technique can provide up to 288 bits of identifying information per flow given limitations on TCP options size. Note that multiple bare ACKs can be used to extend the amount of information injected per flow.

Mitigations and caveats thereon are as in Section 4.3.3, above.

4.3.5. Out of Window Segments

A middlebox that keeps state for each TCP connection traversing it can place out-of-window segments sharing a given 5-tuple but not initiated by the sender on the wire. These segments should traverse any device not looking at TCP state, and be ignored by the receiver.

This technique can provide over 11000 bits of identifying information per flow given a 1500 byte MTU. Note that multiple out of window segments can be used to extend the amount of information injected per flow.

Information exposed using this technique may not traverse stateful TCP firewalls or split TCP proxies. Existing stateful TCP firewalls already provide out-of-window segment dropping, due to their usefulness in TCP session hijacking attacks (see [blind-tcp-attacks] for more). The aggressive deployment of stateful TCP firewalls that drop and warn on out- of-window segments can mitigate this attack. This mitigation incurs the cost of increased operational complexity and further middlebox-related transport ossification.

4.3.6. Bad Checksum Segments

Similar to Section 4.3.5, a middlebox can place segments with bad checksums sharing a given 5-tuple on the wire. These segments should traverse any device not looking at TCP state, and be ignored by the receiver.

Per-flow information and mitigations along with caveats are as in Section 4.3.5.

4.4. Combination of Techniques

Note that multiple techniques above may be combined on any given packet or over the sequence of packets in any given flow in order to increase the number of bits available and/or increase the resilience of the injected information to mitigation.

5. Recommendations and Outlook

An analysis of the hypercookie attacks listed in this document, and the ability to combine them freely to improve hypercookie resilience and capacity, leads to a relatively bleak outlook. Mitigating the threat at scale with the stack as presently deployed requires impractically aggressive, altruistic deployment of TCP-modifying firewalls.

We therefore conclude that the most practical mitigation of this threat is the development and deployment of transport protocols that provide cryptographic integrity protection and/or confidentiality for their headers, in order to prevent hypercookie injection at the transport layer.

Note that these mitigations can only detect, but not prevent, hypercookie coercion attacks: if an attacker can successfully block a client's access to the Internet to enforce hypercookie coercion, removal of metadata will not restore that access, as the attack is carried out through nontechnical relationships between the attacker and the target. We can only hope that raising awareness and bringing transparency to the potential for hypercookie coercion attacks makes them less likely to be successful.

6. IANA Considerations

This document has no actions for IANA [EDITOR'S NOTE: please remove this section at publication.]

7. Security Considerations

This document outlines vulnerabilities in the TCP/IP protocol stack as deployed to a type of attack described in Section 1. Exploitation of these vulnerabilities can be used to expose identifying information about users of a network to third parties; the document discusses general and specific techniques to mitigate the impact of these exploits.

8. Acknowledgments

This work is supported by the European Commission under Horizon 2020 grant agreement no. 688421 Measurement and Architecture for a Middleboxed Internet (MAMI), and by the Swiss State Secretariat for Education, Research, and Innovation under contract no. 15.0268. This support does not imply endorsement.

Thanks to Ted Hardie, Joe Hildebrand, Mirja Kuehlewind, and the participants at the PLUS BoF at IETF 96 in Berlin for the conversations leading to and informing the publication of this document.

9. References

9.1. Normative References

[RFC3234] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, DOI 10.17487/RFC3234, February 2002, <http://www.rfc-editor.org/info/rfc3234>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <http://www.rfc-editor.org/info/rfc6973>.

[RFC7624] Barnes, R., Schneier, B., Jennings, C., Hardie, T., Trammell, B., Huitema, C., and D. Borkmann, "Confidentiality in the Face of Pervasive Surveillance: A Threat Model and Problem Statement", RFC 7624, DOI 10.17487/RFC7624, August 2015, <http://www.rfc-editor.org/info/rfc7624>.

9.2. Informative References

[blind-tcp-attacks]
Luckie, M., Beverly, R., Wu, T., Allman, M., and K. Claffy, "Resilience of Deployed TCP to Blind Attacks", 2015, <http://www.caida.org/~mjl/pubs/blind.pdf>.

[hiccups] Craven, R., Beverly, R., and M. Allman, "A Middlebox-Cooperative TCP for a non End-to-End Internet", 2014, <http://rbeverly.net/research/papers/hiccups-sigcomm14.pdf>.

[I-D.hardie-privsec-metadata-insertion]
Hardie, T., "Design considerations for Metadata Insertion", draft-hardie-privsec-metadata-insertion-02 (work in progress), March 2016.

[I-D.thomson-postel-was-wrong]
Thomson, M., "The Harmful Consequences of Postel's Maxim", draft-thomson-postel-was-wrong-00 (work in progress), March 2015.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <http://www.rfc-editor.org/info/rfc791>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <http://www.rfc-editor.org/info/rfc1122>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <http://www.rfc-editor.org/info/rfc3315>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <http://www.rfc-editor.org/info/rfc4291>.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <http://www.rfc-editor.org/info/rfc4941>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC6093] Gont, F. and A. Yourtchenko, "On the Implementation of the TCP Urgent Mechanism", RFC 6093, DOI 10.17487/RFC6093, January 2011, <http://www.rfc-editor.org/info/rfc6093>.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <http://www.rfc-editor.org/info/rfc6437>.

[RFC6864] Touch, J., "Updated Specification of the IPv4 ID Field", RFC 6864, DOI 10.17487/RFC6864, February 2013, <http://www.rfc-editor.org/info/rfc6864>.

[RFC6994] Touch, J., "Shared Use of Experimental TCP Options", RFC 6994, DOI 10.17487/RFC6994, August 2013, <http://www.rfc-editor.org/info/rfc6994>.

[RFC7136] Carpenter, B. and S. Jiang, "Significance of IPv6 Interface Identifiers", RFC 7136, DOI 10.17487/RFC7136, February 2014, <http://www.rfc-editor.org/info/rfc7136>.

[RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <http://www.rfc-editor.org/info/rfc7413>.

Author's Address

Brian Trammell
ETH Zurich

Email: ietf@trammell.ch