Internet Engineering Task Force (IETF)
Request for Comments: 7332
Category: Standards Track
ISSN: 2070-1721

H. Kaplan
Oracle
V. Pascual
Quobis

August 2014

Loop Detection Mechanisms for Session Initiation Protocol (SIP) Back-to-Back User Agents (B2BUAs)

Abstract

SIP Back-to-Back User Agents (B2BUAs) can cause unending SIP request routing loops because, as User Agent Clients, they can generate SIP requests with new Max-Forwards values. This document discusses the difficulties associated with loop detection for B2BUAs and the requirements for them to prevent infinite loops.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7332.

Table of Contents

1. Introduction

SIP provides a means of preventing infinite request forwarding loops in [RFC3261], and a means of mitigating parallel forking amplification floods in [RFC5393]. Neither document normatively defines specific behavior for B2BUAs, however.

Unbounded SIP request loops have actually occurred in SIP deployments numerous times. The cause of loops is usually misconfiguration, but the reason they have been unbounded/unending is they crossed B2BUAs that reset the Max-Forwards value in the SIP requests they generated on their User Agent Client (UAC) side. Although such behavior is technically legal per [RFC3261] because a B2BUA is a UAC, the resulting unbounded loops have caused service outages and make troubleshooting difficult.

Furthermore, [RFC5393] also provides a mechanism to mitigate the impact of parallel forking amplification issues, through the use of a "Max-Breadth" header field. If a B2BUA does not pass this header field on, parallel forking amplification is not mitigated with the [RFC5393] mechanism.

This document defines normative requirements for Max-Forwards and Max-Breadth header field behaviors of B2BUAs, in order to mitigate the effect of loops and parallel forking amplification.

2. Conventions

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 BCP 14, RFC 2119 [RFC2119].

B2BUA terminology and taxonomy used in this document is based on [RFC7092].

3. Background

Within the context of B2BUAs, the scope of the SIP protocol ends at the User Agent Server (UAS) side of the B2BUA, and a new one begins on the UAC side. A B2BUA is thus capable of choosing what it wishes to do on its UAC side independently of its UAS side, and still remains compliant with [RFC3261] and its extensions. For example, any B2BUA type defined in [RFC7092] other than Proxy-B2BUA may create the SIP request on its UAC side without copying any of the Via header field values received on its UAS side. Indeed there are valid reasons for it to do so; however, this prevents the Via-based loop-detection mechanism defined in [RFC3261] and updated by [RFC5393] from detecting SIP request loops any earlier than by reaching a Max-Forwards limit.

Some attempts have been made by B2BUA vendors to detect request loops in other ways: by keeping track of the number of outstanding dialog-forming requests for a given caller/called URI pair; or by detecting when they receive and send their own media addressing information too many times in certain cases when they are a signaling/media-plane B2BUA; or by encoding a request instance identifier in some field they believe will pass through other nodes, and detecting when they see the same value too many times.

All of these methods are brittle and prone to error, however. They are brittle because it is very hard to accurately define when a value has been seen "too many times". Requests can and do fork before and after B2BUAs process them, and requests legitimately spiral in some cases, leading to incorrect determination of loops. The mechanisms are prone to error because there can be other B2BUAs in the loop’s path that interfere with the particular mechanism being used.

Ultimately, the last defense against loops becoming unbounded is to limit how many SIP hops any request can traverse, which is the purpose of the SIP Max-Forwards field value. If B2BUAs were to at least copy and decrement the Max-Forwards header field value from their UAS to the UAC side, loops would not continue indefinitely.

4. B2BUA Loop-Detection Behavior

It is RECOMMENDED that B2BUAs implement the loop-detection mechanism for the Via header field, as defined for a proxy in [RFC5393].

5. B2BUA Max-Forwards Behavior

This section applies for dialog-forming and out-of-dialog SIP requests. B2BUAs MAY perform the same actions for in-dialog requests, but doing so may cause issues with devices that set Max-Forwards values based upon the number of received Via or Record-Route headers.

All B2BUA types MUST copy the received Max-Forwards header field from the received SIP request on their UAS side, to any request(s) they generate on their UAC side, and decrement the value, as if they were a proxy following the requirements described in [RFC3261].

Being a UAS, B2BUAs MUST also check the received Max-Forwards header field and reject or respond to the request if the value is zero, as defined in [RFC3261].

If the received request did not contain a Max-Forwards header field, one MUST be created in any request generated in the UAC side, as described for proxies in Section 16.6, Step 3 of [RFC3261]. As in that specification, the value of the new Max-Forwards header SHOULD be 70.

6. B2BUA Max-Breadth Behavior

All B2BUA types MUST copy the received Max-Breadth header field from the received SIP request on their UAS side, to any request(s) they generate on their UAC side, as if they were a proxy following the requirements described in [RFC5393].

B2BUAs of all types MUST follow the requirements imposed on Proxies as described in Section 5.3.3 of [RFC5393], including generating the header field if none is received, limiting its maximum value, etc.

B2BUAs that generate parallel requests on their UAC side for a single incoming request on the UAS side MUST also follow the rules for Max-Breadth handling in [RFC5393] as if they were a parallel forking proxy.

7. Security Considerations

The security implications for parallel forking amplification are documented in Section 7 of [RFC5393]. This document does not introduce any additional issues beyond those discussed in [RFC5393].

Some B2BUAs reset the Max-Forwards and Max-Breadth header field values in order to obfuscate the number of hops a request has already traversed, as a privacy or security concern. Such goals are at odds

with the mechanisms in this document, and administrators can decide which they consider more important: obfuscation vs. loop detection. In order to comply with this RFC, manufacturers MUST comply with the normative rules defined herein by default, but MAY provide user-configurable overrides as they see fit.

8. Acknowledgments

Thanks to Brett Tate (Broadsoft), Andrew Hutton (Unify), and Anton Roman (Quobis) for their review of the document.

9. References

9.1. Normative References

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

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC5393] Sparks, R., Lawrence, S., Hawrylyshen, A., and B. Campen, "Addressing an Amplification Vulnerability in Session Initiation Protocol (SIP) Forking Proxies", RFC 5393, December 2008.

9.2. Informative References

[RFC7092] Kaplan, H. and V. Pascual, "A Taxonomy of Session Initiation Protocol (SIP) Back-to-Back User Agents", RFC 7092, December 2013.

Authors' Addresses

Hadriel Kaplan
Oracle

EMail: hadrielk@yahoo.com

Victor Pascual
Quobis

EMail: victor.pascual@quobis.com