Limiting the number of concurrent branches for a Session Initiation Protocol request
draft-sparks-sipping-max-breadth-01

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on December 3, 2007.

Copyright Notice

Copyright (C) The IETF Trust (2007).

Abstract

This document defines the Max-Breadth mechanism for limiting the number of concurrent branches pursued for any given request. This
tool helps protect against certain amplification attacks that leverage the Session Initiation Protocol's request forking mechanism.

Table of Contents

1. Conventions and Definitions ................................... 3
2. Introduction ....................................................... 3
3. Examples ........................................................... 4
4. Formal Mechanism .................................................. 5
   4.1. "Max-Breadth" Header ........................................ 5
   4.2. Terminology .................................................. 5
   4.3. Proxy Behavior ................................................ 5
       4.3.1. Reusing Max-Breadth .................................... 6
   4.4. UAC Behavior ................................................ 6
   4.5. UAS behavior ................................................ 6
5. Implementor Notes .................................................. 6
   5.1. Treatment of CANCEL ......................................... 6
   5.2. Reclamation of Max-Breadth on 2xx Responses ............... 7
   5.3. Max-Breadth and Automaton UAs ............................. 7
6. Parallel and Sequential Forking .................................... 7
7. Max-Breadth Split Weight Selection ............................... 7
8. Max-Breadth's Effect on Forking-based Amplification Attacks .... 8
9. Header Field Definition ............................................ 8
10. IANA Considerations ............................................. 8
    10.1. Max-Forwards Header Field ................................. 8
    10.2. 4xx Max-Breadth Exceeded response ....................... 8
11. Security Considerations ......................................... 9
    11.1. Loop-Detection and the Wide-Fork Loop Attack ............. 9
    11.2. Inefficacy of Securing AORs ............................... 10
12. Open Issues ...................................................... 10
13. References ....................................................... 11
    13.1. Normative References ..................................... 11
    13.2. Informative References ................................... 11
Authors' Addresses .................................................. 11
Intellectual Property and Copyright Statements .................... 13
1. Conventions and Definitions

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 [RFC2119].

2. Introduction

This document defines the Max-Breadth mechanism to limit the total number of concurrent branches caused by a forked SIP request. The mechanism only limits concurrency. It does not limit the total number of branches a request can traverse over its lifetime.

With this mechanism, all proxyable requests are assigned a positive integral Max-Breadth value, which denotes the maximum number of concurrent branches this request may spawn through parallel forking as it is forwarded from its current point. When a proxy forwards a request, its Max-Breadth value is divided among the outgoing requests. In turn, each of the forwarded requests has a limit on how many concurrent branches they may spawn. As branches complete, their portion Max-Breadth value becomes available for subsequent branches, if needed. If there is insufficient Max-Breadth to carry out a desired parallel fork, a proxy can return a new 4xx Max-Breadth Exceeded response (the actual 4xx value will be selected later in the consensus process).

This mechanism operates independently from Max-Forwards. Max-Forwards limits the depth of the tree a request may traverse as it is forwarded from its origination point to each destination it may be forked to. As [I-D.ietf-sip-fork-loop-fix] shows, the number of branches in a tree of even limited depth can be made large (exponential with depth) by leveraging forking. Each such branch has a pair of SIP transaction state machines associated with it. The Max-Breadth mechanism limits the number of branches that are active (those that have running transaction state machines) at any given point in time.

Max-Breadth does not prevent forking. It only limits the number of concurrent parallel forked branches. In particular, a Max-Breadth of 1 restricts a request to pure serial forking rather than restricting it from being forked at all.
### 3. Examples

<table>
<thead>
<tr>
<th>UAC</th>
<th>Proxy A</th>
<th>Proxy B</th>
<th>Proxy C</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVITE</td>
<td>Max-Breadth: 60</td>
<td>Max-Breadth: 30</td>
<td>Max-Breadth: 69</td>
</tr>
<tr>
<td>Max-Forwards: 70</td>
<td>Max-Forwards: 69</td>
<td>Max-Forwards: 69</td>
<td></td>
</tr>
</tbody>
</table>

Parallel forking

<table>
<thead>
<tr>
<th>UAC</th>
<th>Proxy A</th>
<th>Proxy B</th>
<th>Proxy C</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVITE</td>
<td>Max-Breadth: 60</td>
<td>Max-Breadth: 60</td>
<td>Max-Breadth: 60</td>
</tr>
<tr>
<td>Max-Forwards: 70</td>
<td>Max-Forwards: 69</td>
<td>Max-Forwards: 68</td>
<td></td>
</tr>
</tbody>
</table>

Sequential forking

<table>
<thead>
<tr>
<th>UAC</th>
<th>Proxy A</th>
<th>Proxy B</th>
<th>Proxy C</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVITE</td>
<td>Max-Breadth: 60</td>
<td>Max-Breadth: 60</td>
<td>Max-Breadth: 60</td>
</tr>
<tr>
<td>Max-Forwards: 70</td>
<td>Max-Forwards: 69</td>
<td>Max-Forwards: 68</td>
<td></td>
</tr>
</tbody>
</table>

No forking
Max-Breadth and Max-Forwards working together

4. Formal Mechanism

4.1. "Max-Breadth" Header

The Max-Breadth header takes a single positive integer as its value. The Max-Breadth header takes no parameters.

4.2. Terminology

For each "response context" (see [RFC3261] Sec 16) in a proxy, this mechanism defines two positive integral values; Incoming Max-Breadth and Outgoing Max-Breadth. Incoming Max-Breadth is the value of the Max-Breadth header field value in the request that formed the response context. Outgoing Max-Breadth is the sum of the Max-Breadth of all forwarded requests in the response context, that have not received a final response.

4.3. Proxy Behavior

If a SIP proxy receives a request with no Max-Breadth header field value, it MUST add one, with a value that is RECOMMENDED to be 60. Proxies MUST have a maximum allowable Incoming Max-Breadth value,
which is RECOMMENDED to be 60. If this maximum is exceeded in a
received request, the proxy MUST overwrite with a value that SHOULD
be no greater than its allowable maximum (ISSUE 2).

All proxied requests MUST contain a single Max-Breadth header field
value.

SIP proxies MUST NOT allow the Outgoing Max-Breadth to exceed the
Incoming Max-Breadth in a given response context.

If a SIP proxy determines a response context has insufficient
Incoming Max-Breadth to carry out a desired parallel fork, and the
proxy is unwilling/unable to compensate by forking serially or
sending a redirect, that proxy MUST return a 4xx Max-Breadth Exceeded
response.

A SIP proxy MAY distribute Max-Breadth in an arbitrary fashion
between active branches. A proxy SHOULD NOT use a smaller amount of
Max-Breadth than was present in the original request, unless the
Incoming Max-Breadth exceeded the proxy's maximum acceptable value.
A proxy MUST NOT decrement Max-Breadth for each hop or otherwise use
it to restrict the "depth" of a request's propagation.

4.3.1. Reusing Max-Breadth

Because forwarded requests that have received a final response do not
count towards the Outgoing Max-Breadth, whenever a final response
arrives, the Max-Breadth that was used on that branch becomes
available for reuse. Proxies SHOULD be prepared to reuse this Max-
Breadth in cases where there may be elements left in the target-set.

4.4. UAC Behavior

A UAC MAY place a Max-Breadth header field value in outgoing
requests. If so, this value is RECOMMENDED to be 60 (ISSUE 2).

4.5. UAS behavior

This mechanism does not affect UAS behavior.

5. Implementor Notes

5.1. Treatment of CANCEL

Since CANCEL requests are never proxied, a Max-Breadth header-field-
value is meaningless in a CANCEL request. Sending a CANCEL in no way
effects the Outgoing Max-Breadth in the associated INVITE response
5.2. Reclamation of Max-Breadth on 2xx Responses

Whether 2xx responses free up Max-Breadth is mostly a moot issue, since proxies are forbidden to start new branches in this case. But, there is one caveat. For INVITE, we may receive multiple 2xx for a single branch. Also, 2543 implementations may send back a 6xx followed by a 2xx on the same branch. Implementations that subtract from the Outgoing Max-Breadth when they receive an INVITE/2xx must be careful to avoid bugs caused by subtracting multiple times for a single branch.

5.3. Max-Breadth and Automaton UAs

Designers of automaton UAs (including B2BUAs, gateways, exploders, and any other element that programmatically sends requests as a result of incoming SIP traffic) should consider whether Max-Breadth limitations should be placed on outgoing requests. For example, it is reasonable to design B2BUAs to carry the Max-Breadth value from incoming requests over into requests that are sent as a result. Also, it is reasonable to place Max-Breadth constraints on sets of requests sent by exploders, when they may be leveraged in an amplification attack.

6. Parallel and Sequential Forking

Inherent in this proposal is the ability of a proxy to reclaim apportioned Max-Breadth while forking sequentially. The limitation on outgoing Max-Breadth is applied to concurrent branches only.

For example, if a proxy receives a request with a Max-Breadth of 4, and has 8 targets to forward it to, that proxy may parallel fork to 4 of these targets initially (each with a Max-Breadth of 1, totaling an Outgoing Max-Breadth of 4). If one of these transactions completes with a failure response, the outgoing Max-Breadth drops to 3, allowing the proxy to forward to one of the 4 remaining targets (again, with a Max-Breadth of 1).

7. Max-Breadth Split Weight Selection

There are a variety of mechanisms for controlling the weight of each fork branch. Fork branches that are given more Max-Breadth are more likely to complete quickly (because it is less likely that a proxy down the line will be forced to fork sequentially). By the same
token, if it is known that a given branch will not fork later on, a Max-Breadth of 1 may be assigned with no ill effect. This would be appropriate, for example, if a proxy knows the branch is using the SIP outbound extension [I-D.ietf-sip-outbound].

8. Max-Breadth's Effect on Forking-based Amplification Attacks

Max-Breadth limits the total number of active branches spawned by a given request at any one time, while placing no constraint on the distance (measured in hops) that the request can propagate. (ie, receiving a request with a Max-Breadth of 1 means that any forking must be sequential, not that forking is forbidden)

This limits the effectiveness of any amplification attack that leverages forking, because the amount of state/bandwidth needed to process the traffic at any given point in time is capped.

9. Header Field Definition

This specification extends the grammar for the Session Initiation Protocol by adding the following extension-header:

Max-Breadth = "Max-Breadth" HCOLON 1*DIGIT

10. IANA Considerations

This specification registers a new SIP header field and a new SIP response according to the processes defined in [RFC3261].

10.1. Max-Forwards Header Field

This information should appear in the header sub-registry under http://www.iana.org/assignments/sip-parameters.

RFC XXXX (this specification)

Header Field Name: Max-Breadth

Compact Form: none

10.2. 4xx Max-Breadth Exceeded response

This information should appear in the response-code sub-registry under http://www.iana.org/assignments/sip-parameters.
Response code: 4xx (The actual response code will be chosen later in the process)

Default Reason Phrase: Max-Breadth Exceeded

11. Security Considerations

11.1. Loop-Detection and the Wide-Fork Loop Attack

In [I-D.ietf-sip-fork-loop-fix], the forking-loop attack presented involves only 2 participating AORs. In this case, loop-detection is sufficient to stop the attack before significant amplification is achieved. However, in cases where the number of participating AORs is large, loop-detection gives insufficient protection against the attack. (It is worth noting here that acquiring several AORs to participate in an attack is quite easy, and networks are vulnerable to this attack regardless of the degree of control they maintain over their AORs; see Section 11.2.)

Loop-detection is insufficient protection against the wide-fork loop attack because requests will often take many hops to complete a loop, and there are a very large number of different loops that will occur during the attack. In fact, if N is the number of participating AORs, and provided N is less than or equal to Max-Forwards, the amount of traffic generated by the attack is greater than N!, even if all proxies involved are performing loop-detection.

Suppose we have a set of N AORs, all of which are set up to fork to the entire set. For clarity, assume AOR 1 is where the attack begins. Every permutation of the remaining N-1 AORs will play out, defining (N-1)! distinct paths, without repeating any AOR. Then, each of these paths will fork N ways one last time, and a loop will be detected on each of these branches. These final branches alone total N! requests ((N-1)! paths, with N forks at the end of each path).
### Forwarded Requests vs. Number of Participating AORs

<table>
<thead>
<tr>
<th>N</th>
<th>Requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>64</td>
</tr>
<tr>
<td>5</td>
<td>325</td>
</tr>
<tr>
<td>6</td>
<td>1956</td>
</tr>
<tr>
<td>7</td>
<td>13699</td>
</tr>
<tr>
<td>8</td>
<td>109600</td>
</tr>
<tr>
<td>9</td>
<td>986409</td>
</tr>
<tr>
<td>10</td>
<td>9864100</td>
</tr>
</tbody>
</table>

In a network where all proxies are performing loop-detection, an attacker is still afforded rapidly increasing returns on the number of AORs they are able to leverage.

#### 11.2. Inefficacy of Securing AORs

Additionally, networks that maintain "perfect" control of their AORs are vulnerable, because an attacker may easily compromise a less secure network, and direct traffic from the attack anywhere they choose. This is accomplished by adding additional bindings to the participating AORs. Note that this method may be used to attack SIP networks that do not use AORs, or even non-SIP networks.

#### 12. Open Issues

**ISSUE 1** As it is presently written, this mechanism does not decrease the aggregate traffic caused by the forking-loop attack. This mechanism only serves to spread the traffic caused by the attack over a longer period, by limiting the number of concurrent branches that are being processed at the same time. This may not be good enough, considering that an attacker may continue to pump requests into the amplifier they have constructed, causing the traffic to gradually build to unreasonable levels. An alternative would be to forbid reuse of Max-Breadth from completed branches, but this could break existing deployments. Should the mechanism allow breadth from completed branches to be reused or not?
ISSUE 2 The document currently recommends a default starting value of 60 for Max-Forwards. Is this the right value? (60=2^2*3*5 - it was chosen to make small divisions of the value easy to do). Is it necessary to recommend a starting value at all?

ISSUE 3 The document will need to pick a 400-class code and replace the occurrences of 4xx here with it.

13. References

13.1. Normative References


13.2. Informative References


Authors' Addresses

Robert Sparks
Estacado Systems
17210 Campbell Road
Suite 250
Dallas, Texas 75254-4203
USA

Email: RjS@nostrum.com