Groupe de travail Réseau
Request for Comments : 5393
RFC mise à jour : 3261
Catégorie : Sur la voie de la normalisation
Traduction Claude Brière de L'Isle

R. Sparks, éd., Tekelec
S. Lawrence, Nortel Networks
A. Hawrylyshen, Ditech Networks
B. Campen, Tekelec
décembre 2008

Correction d'une faiblesse de la sécurité des mandataires de fourchement dans le protocole d'initialisation de session (SIP)

Statut du présent mémoire
Le présent document spécifie un protocole de l'Internet sur la voie de la normalisation pour la communauté de l'Internet, et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "Protocoles officiels de l'Internet" (STD 1) pour voir l'état de normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé
Le présent document met à jour normativement la RFC 3261, du protocole d'initialisation de session (SIP, *Session Initiation Protocol*) pour traiter une vulnérabilitéde la sécurité identifiée dans le comportement de mandataire SIP. Cette vulnérabilité permet une attaque contre les réseaux SIP où un petit nombre de demandes SIP légitimes et même autorisées peut stimuler des quantités massives de trafic de mandataire à mandataire.

Le présent document renforce les exigences de détection de boucle sur les mandataires SIP quand ils fourchent les demandes (c'est-à-dire, transmettent une demande à plus d'une destination). Il corrige et précise aussi la description de l'algorithme de détection de boucle afin qu'il soit exigé des mandataires qu'ils le mettent en œuvre. De plus, le document définit un mécanisme Max-Breadth pour limiter le nombre de branches concurrentes poursuivies pour toute demande.

Table des matières

1. Introduction

Les essais d'interopérabilité ont fait découvrir une vulnérabilité dans le comportement des mandataires SIP de fourchement comme défini dans la [RFC3261]. Cette vulnérabilité peut être renforcée pour faire qu'un petit nombre de demandes SIP valides génèrent un nombre extrêmement grand de messages de mandataire à mandataire. Une version de cette attaque montre que moins de dix messages stimulent potentiellement 2^71 messages.

Le présent document spécifie des changements normatifs au protocole SIP pour traiter cette vulnérabilité. Conformément à cette mise à jour, quand un mandataire SIP fourche une demande en plus d'une destination, il est exigé qu'il s'assure de ne pas participer à une demande en boucle.

Cette mise à jour normative seule est insuffisante pour protéger contre les variantes élaborées de l'attaque décrite ici qui impliquent plusieurs adresses d'entretien (AOR, *Address of Record*). Pour mieux traiter cette vulnérabilité, ce document définit le mécanisme Max-Breadth *(largeur maximale)* pour limiter le nombre total de branches concurrentes causées par une demande SIP fourchée. Le mécanisme limite seulement la concurrence, il ne limite pas le nombre total de branches qu'une demande peut traverser dans sa vie.

Les mécanismes de cette mise à jour vont protéger contre les variantes de l'attaque décrites ici qui utilisent un petit nombre de ressources, incluant les variantes les plus involontaires auto infligées qui se produisent par une mauvaise configuration accidentelle. Cependant, un attaquant qui a accès à un nombre suffisant de ressources distinctes va quand même être capable de stimuler un très grand nombre de messages. Le nombre de messages concurrents va être limité par le mécanisme Max-Breadth, de sorte que l'ensemble entier va être étalé sur une longue durée, donnant aux opérateurs une meilleure opportunité de détecter l'attaque et de prendre des mesures correctives en dehors du protocole. Un futur travail sur le protocole est nécessaire pour empêcher cette forme de l'attaque.

2. Conventions et définitions

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" en majuscules dans ce document sont à interpréter comme décrit dans le BCP 14, [RFC2119].

3. Vulnérabilité : levier de fourchement pour arroser un réseau

Cette section décrit l'établissement d'une attaque avec une hypothèse simplificatrice : que deux comptes sur chacun de deux serveurs différents mandataires/registraires conformes à la RFC 3261 qui ne font pas de détection de boucle sont disponibles à un attaquant. Cette hypothèse n'est pas nécessaire pour l'attaque mais rend la représentation du scénario plus simple. La même attaque peut être réalisée avec un seul compte sur un seul serveur.

Considérons deux services de mandataire/registraire, P1 et P2, et quatre adresses d'entretien, a@P1, b@P1, a@P2, et b@P2. En utilisant des demandes REGISTER normales, on établit les liens à ces AOR comme suit (les détails non essentiels ne sont pas montrés) :

```
REGISTER sip:P1 SIP/2.0
To: <sip:a@P1>
Contact: <sip:a@P2>, <sip:b@P2>

REGISTER sip:P1 SIP/2.0
To: <sip:b@P1>
Contact: <sip:a@P2>, <sip:b@P2>
```

REGISTER sip:P2 SIP/2.0
To: <sip:a@P2>
Contact: <sip:a@P1>, <sip:b@P1>

REGISTER sip:P2 SIP/2.0
To: <sip:b@P2>
Contact: <sip:a@P1>, <sip:b@P1>

Ces liens étant en place, on introduit une demande INVITE à une des quatre AOR, disons a@P1. Cette demande va être fourchée en deux demandes traitées par P2, qui va les fourcher en quatre demandes traitées par P1, qui va les fourcher en huit messages traités par P2, et ainsi de suite. Ce flux de messages est représenté dans la Figure 1.

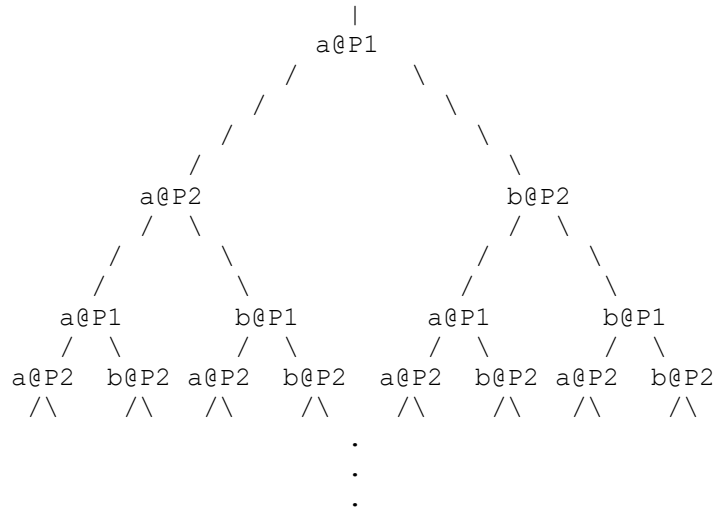


Figure 1 : Attaque de propagation de demandes

Les demandes vont continuer de se propager le long de cette arborescence jusqu'à ce que Max-Forwards atteigne zéro. Si le point d'extrémité et deux mandataires impliqués suivent les recommandations de la RFC 3261, l'arborescence va être profonde de 70 rangées, représentant 2^71-1 demandes. Le nombre réel de messages peut être bien plus grand si le temps de traitement de la totalité des demandes de l'arborescence est plus long que le temporisateur C chez l'un des mandataires. Dans ce cas, une tempêtre de 408 réponses et/ou une tempête de demandes CANCEL va aussi être propagée à travers l'arborescence avec les demandes INVITE. On se souvient que ce sont seulement deux mandataires qui sont impliqués dans ce scénario – chacun devant conserver l'état de toutes les transactions qu'il voit (au moins 2^70 transactions simultanément actives vers la fin du scénario).

L'attaque peut être simplifiée à un compte à un serveur si le service peut être convaincu que les contacts avec divers attributs (paramètres, schémas, en-têtes incorporés) sont suffisamment distincts, et que ces paramètres ne sont pas utilisés au titre des comparaisons d'AOR lors de la transmission d'une nouvelle demande. Comme la RFC 3261 exige que tous les paramètres d'URI soient supprimés d'un URI avant de le chercher dans un service de localisation et que les URI provenant du champ d'en-tête Contact soient comparés en utilisant l'égalité d'URI, l'enregistrement suivant devrait être suffisant pour établir cette attaque en utilisant une seule demande REGISTER à un seul compte :

REGISTER sip:P1 SIP/2.0
To: <sip:a@P1>
Contact: <sip:a@P1;unknown-param=whack>,<sip:a@P1;unknown-param=thud>

Cette attaque a été réalisée en pratique durant une session d'essai d'interopérabilité SIIP (SIPit). Le scénario a été étendu pour inclure plus de deux mandataires, et les mandataires participants ont tous limité les Max-Forwards à ne pas dépasser 20. Après une poignée de messages pour construire l'attaque, les mandataires participants ont commencé à se bombarder les uns les autres. En extrapolant des heures pendant lesquelles il était permis à l'expérience de fonctionner, le scénario se serait terminé en un peu moins de 10 jours. Si les mandataires avaient utilisé la valeur recommandée par la RFC 3261 de 70 Max-Forwards, et en supposant qu'ils aient fonctionné de façon linéaire alors que l'état qu'ils conservent augmentait, il leur aurait fallu 3 milliards d'années pour achever le traitement de la seule demande INVITE qui initiait l'attaque. Il est intéressant de noter que quelques mandataires ont réamorcé durant le scénario et ont rejoint l'attaque quand ils ont redémarré (pour autant qu'ils aient conservé l'état d'enregistrement à travers les réamorçages). Cela souligne que si cette

attaque était lancée sur l'Internet, elle pourrait exiger une coordination entre tous les éléments affectés pour l'arrêter.

La détection de boucle, comme spécifiée dans ce document, chez un des mandataires dans les scénarios décrits jusqu'à présent aurait arrêté l'attaque immédiatement. (Si tous les mandataires impliqués mettaient en œuvre cette détection de boucle, le nombre total de messages stimulés dans le premier scénario décrit aurait été réduit à 14 ; dans la variante impliquant un serveur, le nombre de messages stimulés aurait été réduit à 10.) Cependant, il y a une variante de l'attaque qui utilise plusieurs AOR où la détection de boucle seule est une protection insuffisante. Dans cette variante, chaque AOR participante fourche sur toutes les autres AOR participantes. Pour de petits nombre d'AOR participantes (10, par exemple) les chemins à travers l'arborescence résultante ne vont pas faire de boucle jusqu'à ce que de très grands nombres de messages aient été générés. Acquérir le nombre suffisant d'AOR pour lancer de telles attaques sur les réseaux actuellement disponibles est assez faisable.

Dans ce scénario, les demandes vont souvent faire de nombreux bonds pour achever une boucle, et il y a un très grand nombre de boucles différentes qui vont se produire durant l'attaque. En fait, si N est le nombre d'AOR participantes, et pourvu que N soit inférieur ou égal à Max-Forwards, la quantité de trafic généré par l'attaque est supérieure à N!, même si tous les mandataires impliqués font la détection de boucle.

Supposons qu'on ait un ensemble de N AOR, dont toutes sont établies à fourcher sur l'ensemble entier. Pour être clair, on suppose que AOR 1 est là où l'attaque commence. Chaque permutation des N-1 AOR restantes va s'exécuter, définissant (N-1)! chemins distincts, sans répéter aucune AOR. Alors, chacun de ces chemins va fourcher N chemins une dernière fois, et une boucle va être détectée sur chacune de ces branches. Ces branches finales seules totalisent N! demandes ((N-1)! chemins, avec N fourches à la fin de chaque chemin).

N	Demandes
1	1
2	4
3	15
4	64
5	325
6	1956
7	13699
8	109600
9	986409
10	9864100

Demandes transmises pour nombre d'AOR participantes

Dans un réseau où tous les mandataires effectuent la détection de boucle, un attaquant peut toujours se permettre des retours augmentant rapidement sur le nombre des AOR qui sont capables de faire levier. Le mécanisme Max-Breadth défini dans ce document est conçu pour limiter l'effectivité de cette variante de l'attaque.

Dans tous les scénarios, il est important de remarquer qu'à chaque mandataire qui fourche, une branche supplémentaire pourrait être ajoutée qui pointe sur une seule victime (qui pourrait même n'être pas un élément à capacité SIP) résultant en une quantité massive de trafic dirigé vers la victime de potentiellement autant de sources qu'il y a d'AOR participant à l'attaque.

4. Mises à jour à la RFC 3261

4.1 Renforcement de l'exigence d'effectuer la détection de boucle

Les exigences suivantes atténuent le risque qu'un mandataire soit la victime de l'attaque décrite dans ce document.

Quand un mandataire SIP fourche une demande particulière à plus d'une localisation, il DOIT s'assurer que cette demande ne fait pas une boucle à travers ce mandataire. Il est RECOMMANDÉ que les mandataires satisfassent cette exigence en effectuant les étapes de détection de boucle définies dans le présent document.

L'exigence d'utiliser les précisions de ce document sur l'algorithme de détection de boucle de la RFC 3261 est un DEVRAIT pour que de futurs mécanismes sur la voie de la normalisation permettent à un mandataire de déterminer qu'il n'est pas en train de faire une boucle. Par exemple, un mandataire qui fourche sur des destinations établies en utilisant le mécanisme sip-outbound [RFC5626] vont savoir les branches qui ne vont pas faire de boucle.

Un mandataire SIP qui transmet une demande à une seule localisation PEUT effectuer la détection de boucle mais n'est pas obligé de le faire. Quand on transmet à une seule localisation, le risque d'exploitation de l'amplification est absent, et le mécanisme Max-Forwards va protéger le réseau dans la mesure prévue (se souvenir toujours de la constante de multiplication due à l'épuisement des Max-Forwards tout en ne fourchant pas). Un mandataire n'est pas obligé d'effectuer la détection de boucle quand il transmet une demande à une seule localisation même si il se trouve avoir précédemment fourché cette demande (et effectué la détection de boucle) dans sa progression à travers le réseau.

4.2 Correction et éclaircissements à l'algorithme de détection de boucle de la RFC 3261

4.2.1 Mise à jour du paragraphe 16.6

Ce paragraphe remplace tout le point 8 du paragraphe 16.6 de la RFC 3261 (le point 8 commence à la page 59 et se termine à la page 60 de la RFC 3261).

8. Ajout d'une valeur de champ d'en-tête Via
Le mandataire DOIT insérer une valeur de champ d'en-tête Via dans la copie avant les valeurs existantes de champ d'en-tête Via. La construction de cette valeur suit les mêmes lignes directrices du paragraphe 8.1.1.7. Cela implique que le mandataire va calculer son propre paramètre de branche, qui va être mondialement unique pour cette branche, et va contenir le mouchard magique requis. Noter que suivre seulement les lignes directrices du paragraphe 8.1.1.7 va résulter en un paramètre de branche qui va être différent pour les différentes instances d'une demande en spirale ou en boucle à travers un mandataire.

Les mandataires qui sont obligés d'effectuer la détection de boucle par la RFC 5393 ont une contrainte supplémentaire sur la valeur qu'ils placent dans le champ d'en-tête Via. Ces mandataires DEVRAIENT créer une valeur de branche séparable en deux parties d'une façon qui dépend de la mise en œuvre.

Le reste de la description de ce paragraphe suppose l'existence de ces deux parties. Si un mandataire choisit d'employer un autre mécanisme, il est de la responsabilité de la mise en œuvre de vérifier que les propriétés de détection définies par les exigences placées sur ces deux parties sont réalisées.

La première partie de la valeur de branche DOIT satisfaire les contraintes du paragraphe 8.1.1.7. La seconde partie est utilisée pour effectuer la détection de boucle et distinguer les boucles des spirales.

Cette seconde partie DOIT varier avec tout champ utilisé par la logique de service de localisation pour déterminer où recibler ou transmettre cette demande. Ceci est nécessaire pour distinguer des demandes en boucle de demandes en spirale en permettant au mandataire de reconnaître si des valeurs qui affectent le traitement de la demande ont changé. Donc, la seconde partie DOIT dépendre au moins de l'URI de demande reçu et de toute valeur de champ d'en-tête Route utilisée lors du traitement de la demande reçue. Les mises en œuvre doivent veiller à inclure tous les champs utilisés par la logique de service de localisation dans cette mise en œuvre particulière.

Cette seconde partie NE DOIT PAS varier avec la méthode de demande. Les demandes CANCEL et les ACK non 200 DOIVENT avoir la même valeur de paramètre de branche que la demande correspondante qu'elles annullent ou acquittent. Cette valeur de paramètre de branche est utilisée pour corréler ces demandes chez le serveur qui les traite (voir les paragraphes 17.2.3 et 9.2).

4.2.2 Mise à jour du paragraphe 16.3

Ce paragraphe remplace tout le point 4 du paragraphe 16.3 de la RFC 3261 (le point 4 apparaît à la page 51 de la RFC 3261).

4. Vérification de détection de boucle
Les mandataires obligés d'effectuer la détection de boucle par la RFC 5393 DOIVENT effectuer l'essai de déctection de boucle suivant avant de transmettre une demande. Chaque valeur de champ d'en-tête Via dans la demande dont la valeur "sent-by" correspond à une valeut placée dans les demandes précédentes par ce mandataire DOIT être inspectée dans la "seconde partie" définie au paragraphe 4.2.1 de la RFC 5393. Cette seconde partie ne sera pas présente si le message n'a pass fourché quand cette valeur de champ d'en-tête Via a été ajoutée. Si le second champ est présent, le mandataire DOIT effectuer le calcul de seconde partie décrit au paragraphe 4.2.1 de la RFC 5393 sur cette demande et comparer le résultat à la valeur provenant du champ d'en-tête Via. Si ces valeurs sont égales, la demande est en boucle et le mandataire DOIT

rejeter la demande avec une réponse 482 (Boucle détectée). Si les valeurs diffèrent, la demande fait une spirale et le traitement se continue à l'étape suivante.

4.2.3 Impact de la détection de boucle sur les performances globales du réseau

Ces exigences et la recommandation d'utiliser les mécanisme de détection de boucle du présent document rendent le compromis favorable de croissance exponentielle de message pour un travail qui est, au pire, d'ordre n^2 lorsque un message traverse n mandataires. Précisément, ce travail est d'ordre m*n où m est le nombre de mandataires dans le chemin qui fourche la demande à plus d'une localisation. En pratique, m est supposé être petit.

L'algorithme de détection de boucle exprimé dans ce document exige qu'un mandataire inspecte chaque élement Via dans la demande reçue. Dans le pire des cas, lorsque un message traverse N mandataires, dont chacun détecte les boucles, le mandataire k fait k inspections, et le nombre total d'inspections qui s'étale à travers les mandataires traitant cette demande est la somme de k de k=1 à k=N qui est N(N+1)/2.

4.2.4 Note de mise en œuvre

Une façon courante de créer la seconde partie de la valeur de paramètres de branche quand une demande est fourchée est de calculer un hachage sur l'enchaînement de l'URI de demande, et toutes les valeurs de champ d'en-tête Route utilisées durant le traitement de la demande, et toutes les autres valeurs utilisées par la logique de service de localisation lors du traitement de cette demane. Le hachage devrait être choisi de telle façon qu'il y ait une faible probabilité que deux ensembles distincts de ces paramètres entrent en collision. Parce que le nombre maximum d'entrées qui doivent être comparées est de 70, les chances d'une collision sont faibles même avec une valeur de hachage relativement basse, comme de 32 bits. Le CRC-32c spécifié dans la [RFC4960] est une fonction acceptable spécifique, comme l'est MD5 [RFC1321]. Noter que MD5 est choisi pour ses propriétés purement non cryptographiques. Un attaquant qui peut contrôler les entrées afin de produire une collision de hachage peut attaquer la connexion de diverses autres façons. Quand on forme la seconde partie en utilisant un hachage, les mises en œuvre DEVRAIENT inclure au moins un champ dans l'entrée du hachage qui varie entre les différentes transactions qui tentent d'atteindre la même destination pour éviter de répéter l'échec si il devrait y avoir une collision de hachage. Les champs Call-ID et CSeq seraient de bonnes entrées pour cela.

Un point commun de défaillance d'interopération aux événememts SIPit a été dû aux analyseurs qui refusent le contenu des valeurs de champ d'en-tête Via d'un autre élément quand ils inspectent la pile de Via à la recherche de boucles. Les mises en œuvre doivent faire attention d'éviter de faire des hypoyhèses sur le format de la valeur de champ d'en-tête Via d'un autre élément au delà des contraintes de base imposées à ce format par la RFC 3261. En particulier, analyser une valeur de champ d'en-tête avec des noms de paramètre inconnus, des paramètres sans valeurs, ou des valeurs de paramètres avec ou sans chaînes entre guillemets ne doit pas causer l'échec d'une mise en œuvre.

Retirer, masquer, ou de quelque autre façon modifier les valeurs des paramètres de branche dans les champs d'en-tête Via dans une demande reçue avant sa transmission supprime la capacité du nœud qui a placé de paramètre de branche dans le message d'effectuer la détection de boucle. Si deux éléments dans une boucle modifient de cette façon les paramètres de branche, une boucle ne peut jamais être détectée.

5. Max-Breadth

5.1 Généralités

Le mécanisme Max-Breadth défini ici limite le nombre total de branches concurrentes causées par une demande SIP fourchée. Avec ce mécanisme, toutes les demandes mandatables reçoivent une valeur de Max-Breadth positive entière, qui note le nombre maximum de branches concurrentes sur lequel cette demande peut s'étendre à travers les fourchements parallèles lorsque elle est transmise à partir de son point actuel. Quand un mandataire transmet une demande, sa valeur de Max-Breadth est divisée entre les demandes sortantes. À leur tour, chacune des demandes transmises a une limite sur le nombre de branches concurrentes auxquelles elle peut s'étendre. Lorsque les branches se terminent, leur portion de la valeur de Max-Breadth devient disponible pour les branches suivantes, si nécessaire. Si il n'y a pas assez de Max-Breadth pour exécuter un fourchement parallèle désiré, un mandataire peut retourner la réponse 440 (Max-Breadth excédé) définie dans ce document.

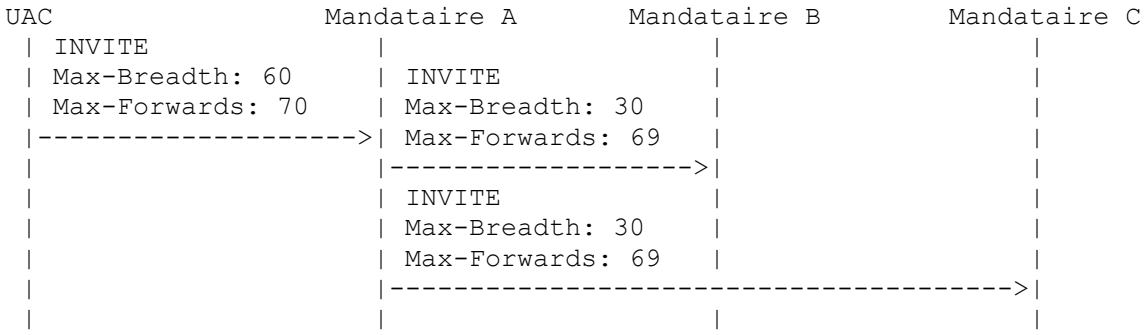
Ce mécanisme opère indépendamment de Max-Forwards. Max-Forwards limite la profondeur de l'arborescence qu'une demande peut traverser lorsque elle est transmise de son point d'origine à chaque destination à laquelle elle est fourchée. Comme le montre la Section 3, le nombre de branches dans une arborescence même de profondeur limitée peut être rendu

grand (exponentiel à la profondeur) en redoublant le fourchement. Chacune de ces branches a une paire d'automates à états de transaction SIP associé. Le mécanisme Max-Breadth limite le nombre de branches qui sont actives (celles qui font fonctionner des automates à états de transaction) à tout moment.

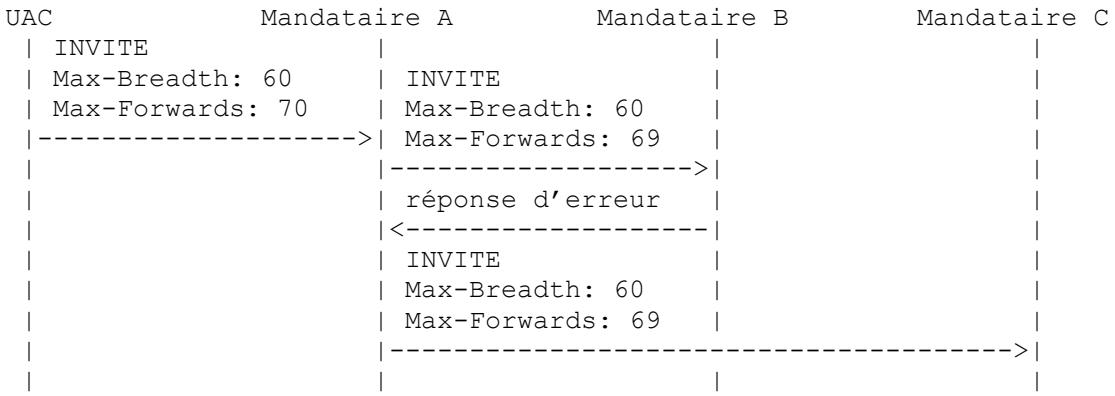
Max-Breadth n'empêche pas le fourchement. Il limite seulement le nombre de branches fourchées concurrentes en parallèle. En particulier, un Max-Breadth de 1 restreint une demande à un pur fourchement en série plutôt que de la restreindre à n'être pas fourchée du tout.

Un client qui reçoit une réponse 440 (Max-Breadth excédé) peut déduire que sa demande n'a pas atteint toutes les destinations possibles. Les options de récupération sont similaires à celle de la réception d'une réponse 483 (Trop de bonds) et incluent d'affecter les décisions d'acheminement par tout mécanisme approprié pour résulter en une recherche moins large, ou de redéfinir la demande elle-même avant la soumission pour réduire l'espace de recherche.

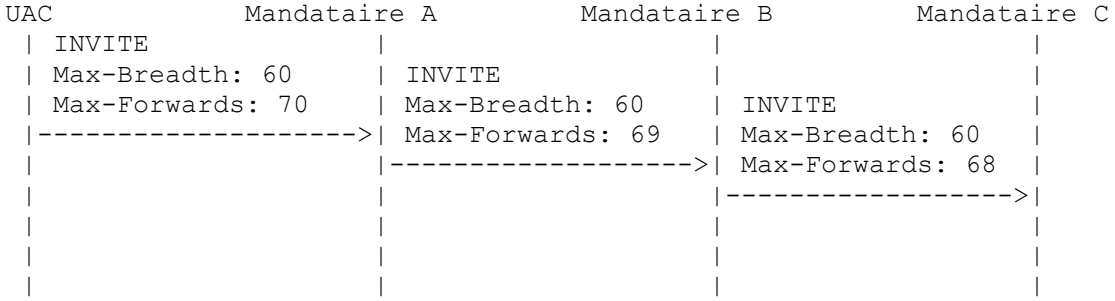
5.2 Exemples



Fourchement parallèle

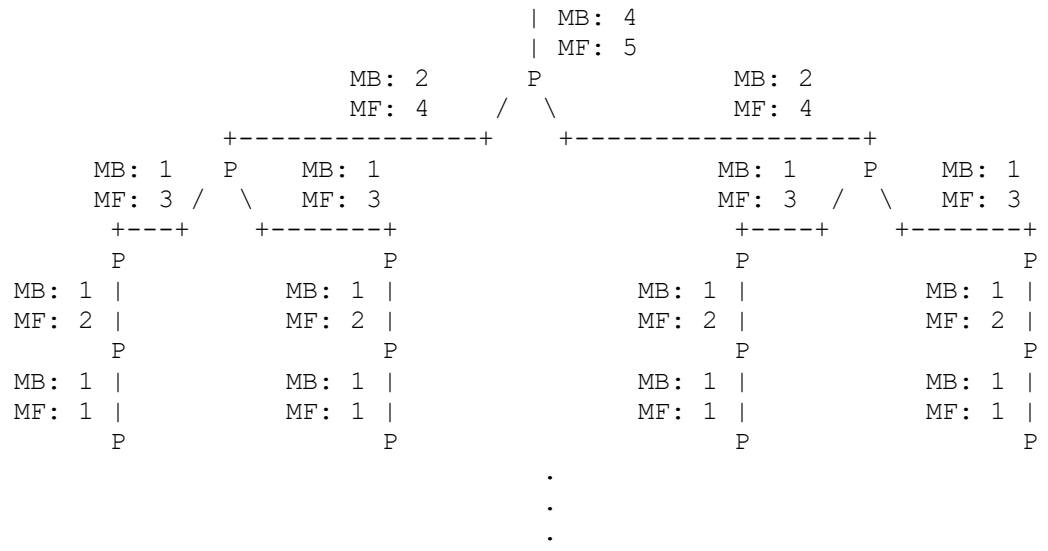


Fourchement séquentiel



Pas de fourchement

MB == Max-Breadth ; MF == Max-Forwards



Max-Breadth et Max-Forwards travaillent ensemble

5.3 Mécanisme formel

5.3.1 Champ d'en-tête Max-Breadth

Le champ d'en-tête Max-Breadth prend un seul entier positif comme valeur. La valeur du champ d'en-tête Max-Breadth ne prend pas de paramètre.

5.3.2 Terminologie

Pour chaque "contexte de réponse" (voir la Section 16 de la [RFC3261]) chez un mandataire, ce mécanisme définit deux valeurs d'entier positif : Max-Breadth entrante et Max-Breadth sortante. Max-Breadth entrante est la valeur dans le champ d'en-tête Max-Breadth de la demande qui a formé le contexte de réponse. Max-Breadth sortante est la somme des valeurs de champ d'en-tête Max-Breadth dans toutes les demandes transmises dans le contexte de réponse qui n'ont pas reçu de réponse finale.

5.3.3 Comportement de mandataire

Si un mandataire SIP reçoit une demande sans valeur de champ d'en-tête Max-Breadth, il DOIT en ajouter une, avec une valeur dont il est RECOMMANDÉ qu'elle soit 60. Les mandataires DOIVENT avoir une valeur maximum permise de Max-Breadth entrante, dont il est RECOMMANDÉ qu'elle soit de 60. Si ce maximum est excédé dans une demande reçue, le mandataire DOIT l'écraser avec une valeur qui DEVRAIT n'être pas supérieure à son maximum permis.

Toutes les demandes soumises à un mandataire DOIVENT contenir une seule valeur de champ d'en-tête Max-Breadth.

Les mandataires SIP NE DOIVENT PAS permettre que le Max-Breadth sortant excède le Max-Breadth entrant dans un contexte de réponse donné.

Si un mandataire SIP détermine qu'un contexte de réponse a un Max-Breadth entrant insuffisant pour exécuter un fourchement parallèle désiré, et si le mandataire ne veut pas ou n'est pas capable de compenser en fourchant en série ou en envoyant une redirection, ce mandataire DOIT retourner une réponse 440 (Max-Breadth excédé).

On remarque que ces exigences signifient qu'un mandataire qui reçoit une demande avec un Max-Breadth de 1 peut seulement fourcher en série, mais il n'est pas obligé de fourcher du tout -- il peut retourner une réponse 440. Donc, ce mécanisme n'est pas un outil qu'un agent d'utilisateur peut utiliser pour forcer tous les mandataires du chemin d'une demande à fourcher en série.

Un mandataire SIP PEUT distribuer le Max-Breadth de façon arbitraire entre les branches actives. Un mandataire Ne DEVRAIT PAS utiliser une plus petite quantité de Max-Breadth que ce qui était présent dans la demande originale sauf si

le Max-Breadth entrant excédait la valeur maximum acceptable du mandataire. Un mandataire NE DOIT PAS décrémenter Max-Breadth pour chaque bond ou autrement l'utiliser pour restreindre la "profondeur" de la propagation d'une demande.

5.3.3.1 Réutilisation de Max-Breadth

Parce que les demandes transmises qui ont reçu une réponse finale ne comptent pas pour le Max-Breadth sortant, chaque fois qu'une réponse finale arrive, le Max-Breadth qui a éte utilisé sur cette branche devient disponible pour réutilisation. Les mandataires DEVRAIENT être prêts à réutiliser ce Max-Breadth dans les cas où il pourrait y avoir des éléments laissés dans l'ensemble cible.

5.3.4 Comportement de l'UAC

Un client d'agent d'utilisateur (UAC, *User Agent Client*) PEUT placer une valeur de champ d'en-tête Max-Breadth dans les demandes sortantes. Si il le fait, il est RECOMMANDÉ que cette valeur soit 60.

5.3.5 Comportement de l'UAS

Ce mécanisme n'affecte pas le comportement du serveur d'agent d'utilisateur (UAS, *User Agent Server*). Un UAS qui reçoit une demande avec un champ d'en-tête Max-Breadth va ignorer ce champ lors du traitement de la demande.

5.4 Notes de mise en œuvre

5.4.1 Traitement de CANCEL

Comme les demandes CANCEL ne sont jamais soumises à un mandataire, une valeur de champ d'en-tête Max-Breadth n'a aucun sens dans une demande CANCEL. L'envoi d'une demande CANCEL n'affecte en aucune façon le Max-Breadth sorant dans le contexte de réponse INVITE associé. La réception d'un CANCEL n'affecte en aucune façon le Max-Breadth entrant du contexte de réponse INVITE associé.

5.4.2 Réclamation de Max-Breadth sur des réponses 2xx

Savoir si les réponses 2xx libèrent le Max-Breadth est une question indécise, car il est interdit aux mandataires de commencer de nouvelles branches dans ce cas. Mais, il faut faire attention. Un mandataire peut recevoir plusieurs réponses 2xx pour une seule demande INVITE transmise. Aussi, les mises en œuvre de la [RFC2543] peuvent renvoyer une 6xx suivie par une 2xx sur la même branche. Les mises en œuvre qui soustraient du Max-Breadth sortant quand elles reçoivent une réponse 2xx à une demande INVITE doivent être attentives à éviter les fautes causées par la soustraction plusieurs fois d'une seule branche.

5.4.3 Max-Breadth et UA automatiques

Les concepteurs d'agents d'utilisateurs (UA, *user agent*) automatiques (incluant des passerelles B2BUA,, des exploseurs, et tout autre élément qui envoie des demandes programmées par suite d'un trafic SIP entrant) devraient examiner si des limitations de Max-Breadth devraient être imposées aux demandes sortantes. Par exemple, il est raisonnable de concevoir des B2BUA pour porter la valeur de Max-Breadth provenant des demandes entrantes dans les demandes qui sont envoyées à leur suite.

Aussi, il est raisonnable de mettre des contraintes de Max-Breadth sur les ensembles de demandes envoyées par les exploseurs quand elles peuvent être démultipliées dans une attaque d'amplification.

5.5 Fourchement parallèle et séquentiel

Inhérente à la définition de ce mécanisme est la capacité d'un mandataire de réclamer un Max-Breadth proportionné lorsque il fourche séquentiellement. La limitation sur le Max-Breadth sortant est appliquée seulement sur les branches concurrentes.

Par exemple, si un mandataire reçoit une demande avec un Max-Breadth de 4 et a 8 cibles à la transmettre, ce mandataire peut fourcher initiallement en parallèle à 4 de ces cibles (chacune avec un Max-Breadth de 1, totalisant un Max-Breadth

sortant de 4). Si une de ces transactions s'achève avec une réponse d'échec, le Max-Breadth sortant tombe à 3, permettant au mandataire de transmettre à une des 4 cibles restantes (là encore, avec un Max-Breadth de 1).

5.6 Choix de pondération partagée de Max-Breadth

Il y a divers mécanismes pour contrôler la pondération de chaque branche de fourchement. Les branches fourchées qui reçoivent plus de Max-Breadth vont probablement s'achever rapidement (parce qu'il est moins probable qu'un mandataire le long de la ligne soit forcé de fourcher séquentiellement). Par le même jeton, si il est connu qu'une branche donnée ne va pas fourcher plus tard, un Max-Breadth de 1 peut être alloué sans effet contraire. Cela serait approprié, par exemple, si un mandataire sait que la branche utilise l'extension SIP de sortie [RFC5626].

5.7 Effet de Max-Breadth sur les attaques d'amplification fondées sur le fourchement

Max-Breadth limite le nombre total de branches actives engendrées par une certaine demande à un moment donné, tout en ne faisant peser aucune contrainte sur la distance (mesurée en bonds) sur laquelle la demande peut se propager. (C'est-à-dire que recevoir une demande avec un Max-Breadth de 1 signifie que tout fourchement doit être séquentiel, et non que le fourchement est interdit.)

Cela limite l'efficacité de toute attaque d'amplification qui s'appuie sur le fourchement à cause de la quantité d'état/bande passante nécessaire pour traiter le trafic qui est contrôlée à tout moment.

5.8 Définition ABNF du champ d'en-tête Max-Breadth

Cette spécification étend la grammaire pour le protocole d'initialisation de session en ajoutant un en-tête d'extension. La définition ABNF [RFC5234] est la suivante.

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

6. Considérations relatives à l'IANA

Cette spécification enregistre un nouveau champ d'en-tête SIP et une nouvelle réponse SIP conformément aux processus définis dans la [RFC3261].

6.1 Champ d'en-tête Max-Breadth

Cette information apparaît dans le sous registre Champs d'en-tête du registre des paramètres SIP.

RFC 5393 (cette spécification)
Nom de champ d'en-tête : Max-Breadth
Forme compacte : aucune

6.2 Réponse 440 Max-Breadth excédé

Cette information apparaît dans le sous registre Codes de réponse du registre des paramètres SIP.

Code de réponse : 440
Phrase de raison par défaut : Max-Breadth Excédé

7. Considérations sur la sécurité

Ce document est entièrement consacré à la documentation et au règlement d'une vulnérabilité des mandataires SIP comme définis par la RFC 3261, qui peut conduire à une attaque d'échange de message à croissance exponentielle.

Le mécanisme Max-Breadth défini ici ne diminue pas le trafic agrégé causé par l'attaque de boucle fourchée. Il sert

seulement à étaler le trafic causé par l'attaque sur une plus longue période en limitant le nombre de branches concurrentes qui sont traitées en même temps. Un attaquant pourrait pomper plusieurs demandes dans un réseau qui utilise le mécanisme Max-Breadth et construire graduellement le trafic à des niveaux déraisonnables. Les déploiements devraient surveiller avec soin et réagir aux augmentations graduelles du nombre de transactions concurrentes en instance relatives à une ressource donnée pour protéger contre cette possibilité. Les opérateurs devraient anticiper d'être capable de désactiver temporairement toutes les ressources identifiées comme étant utilisées dans de telles attaques. Une augmentation rapide des transactions concurrentes en instance à l'échelle du système peut être l'indication de la présence de cette sorte d'attaque sur de nombreuses ressources. Les déploiements dans lesquels il est faisable qu'un attaquant obtienne un très grand nombre de ressources sont particulièrement risqués. Si la détection et l'intervention sur chaque instance de l'attaque est insuffisante pour réduire la charge, une surcharge peut se produire.

Les déployeurs et opérateurs sont encouragés à suivre les recommandations développées pour traiter les conditions de surcharge (voir les [RFC5390] et [RFC6357]).

Les concepteurs de passerelles de protocole devraient examiner avec attention les implications de cette sorte d'attaque. Par exemple, si un message transite d'un réseau SIP au réseau téléphonique public commuté (RTPC) et revient ensuite dans un réseau SIP, et si les informations sur l'historique de la demande sur l'un ou l'autre côté de la traduction de protocole sont perdues, il devient possible de construire des boucles dont ni Max-Forwards ni la détection de boucle ne peuvent protéger. Ceci, combiné avec l'amplification de fourchement sur le côté SIP de la boucle, va résulter en une attaque comme décrite dans ce document, que ces mécanismes ne vont pas réduire, pas même au point de limiter le nombre de messages concurrents dans l'attaque. Ces considérations sont particulièrement importantes pour les concepteurs de passerelles de SIP à SIP (comme on en trouve dans les B2BUA, par exemple). De nombreuses mises en œuvre existantes de B2BUA sont sous pression pour cacher autant d'informations que possible sur les deux côtés communiquants. Les mises en œuvre peuvent être tentées de supprimer les données qui pourraient être utilisées par les mécanisme de détection de boucle, Max-Forwards, ou Max-Breadth à d'autres points dans le réseau, prenant en charge la responsabilité de la détection de boucles (ou de formes de cette attaque). Cependant, si deux de ces mises en œuvre sont impliquées dans l'attaque, aucune d'elles ne sera capable de la détecter.

7.1 Autres solutions considérées et rejetées

Les autres solutions discutées incluent :

Ne rien faire, compter sur des poursuites contre l'agresseur : bien que les systèmes qui ont des comptes aient des enregistrements qui peuvent être exploités pour localiser les agresseurs, il n'est pas clair que cela fournisse une dissuasion crédible ou une défense contre l'attaque décrite dans ce document. Les systèmes qui ne reconnaissent pas la situation et prennent des actions correctives/préventives vont probalement expérimenter des défaillances d'une magnitude qui empêche la restitution des enregistrements qui documentent l'établissement de l'attaque. (Dans un scénario, les enregistrements peuvent se produire à un moment radicalement différent de celui de la transaction INVITE. La demande INVITE elle-même peut être venue d'un innocent). Il est même possible que le scénario puisse être établi de façon non intentionnelle. De plus, pour certains déploiements existants, le coût et la capacité d'audit d'un compte est simplement une adresse de messagerie. Trouver quelqu'un à punir peut être impossible. Finalement, il y a des individus qui ne vont pas répondre à une menace d'action légale, et l'effet de même une seule instance réussie de cette sorte d'attaque pourrait être dévastateur pour un fournisseur de services.

Mettre un plus petit maximum à Max-Forwards : l'effet de l'attaque est exponentiel par rapport à la valeur initiale de Max-Forwards. Diminuer cette valeur limite l'effet de l'attaque. Cela se fait aux dépens d'une limitation sévère de la portée des demandes dans le réseau, éventuellement au point que les architectures existantes commencent à avoir des défaillances.

Interdire les liens d'enregistrement à des contacts arbitraires : la façon dont le lien d'enregistrement est actuellement défini est une partie clé du succès de la sorte d'attaque documentée ici. La solution de remplacement de la limitation des liens d'enregistrement pour ne permetttre que le lien à l'élément de réseau effectuant l'enregistrement, peut-être jusqu'au point extrême d'ignorer les bits fournis dans le Contact en faveur des artifices de transport observés dans la demande d'enregistrement, a été discutée (en particulier dans le contexte des mécanismes définis dans la [RFC5626]). Des mécanismes comme ceux-là peuvent être reconsidérés à l'avenir, mais sont actuellement insuffisamment développés pour régler la menace présente.

Interdire le fourchement : cette attaque n'existe pas dans un système qui s'appuie entièreement sur la redirection et l'initiation de nouvelles demandes par le point d'extrémité d'origine. Supprimer un tel grand composant architectural du système a été estimé une solution trop extrême pour l'instant.

Ne pas réclamer plus de largeur : une solution de remplacement au mécanisme de Max-Breadth qui a été examinée et rejetée était de ne pas permettre que la largeur provenant des branches complétées soit réutilisée (voir le paragraphe 5.3.3.1). Selon cette solution de remplacement, une demande introduite causerait, au plus, que la valeur initiale des transactions Max-Breadth soit générée dans le réseau. Bien que cette approche limite à un multiplicateur constant toute variante de la vulnérabilité à l'amplification décrite ici, elle changerait radicalement la portée potentielle des demandes, et on estime que cela casserait les déploiements existants.

8. Remerciements

Merci aux développeurs qui ont soumis leur code à ce scénario et aidé à analyser les résultats à la réunion SIPit 17. Eric Rescorla a fourni des indications et du texte pour la note de recommandation de hachage.

9. Références

9.1 Références normatives

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997. *(MàJ par* RFC8174)

[RFC3261] J. Rosenberg et autres, "SIP : Protocole d'initialisation de session", juin 2002, DOI 10.17487/RFC3261. (*Mise à jour par* 3265, 3853, 4320, 4916, 5393, 6665, 8217, 8760)

[RFC5234] D. Crocker, P. Overell, "BNF augmenté pour les spécifications de syntaxe : ABNF", janvier 2008. (STD0068)

9.2 Références pour information

[RFC1321] R. Rivest, "Algorithme de résumé de message MD5", avril 1992, DOI 10.17487/RFC1321, *(Information)*

[RFC2543] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg, "SIP : protocole d'initialisation de session", mars 1999. (*Obsolète, voir* RFC3261, RFC3262, RFC3263, RFC3264, RFC3265) *(P.S.)*

[RFC4960] R. Stewart, éd., "Protocole de transmission de commandes de flux (SCTP)", septembre 2007. (*Remplace* RFC2960, RFC3309 ; *P.S. ; Remplacée par* RFC9260)

[RFC5390] J. Rosenberg, "Exigences pour la gestion des surcharges dans le protocole d'initialisation de session", décembre 2008. *(Info.)*

[RFC5626] C. Jennings, R. Mahy, F. Audet, éd., "Gestion des connexions initiées par le client dans le protocole d'initialisation de session (SIP)", octobre 2009, DOI 10.17487/RFC5626. *(MàJ* RFC3261, RFC3327) *(P. S.)*

[RFC6357] V. Hilt, E. Noel, C. Shen, A. Abdelal, "Considérations sur la conception du contrôle de surchage pour le protocole d'initialisation de session (SIP)", août 2011. *(Information)*

Adresse des auteurs

Robert Sparks (éditeur)
Tekelec
17210 Campbell Road
Suite 250
Dallas, Texas 75254-4203
USA
mél : RjS@nostrum.com

Scott Lawrence
Nortel Networks, Inc.
600 Technology Park
Billerica, MA 01821
USA
mél : scott.lawrence@nortel.com

Alan Hawrylyshen
Ditech Networks Inc.
823 E. Middlefield Rd
Mountain View, CA 94043
USA
mél : alan.ietf@polyphase.ca

Byron Campen
Tekelec
17210 Campbell Road
Suite 250
Dallas, Texas 75254-4203
USA
mél : bcampen@estacado.net